UNIVERSIDAD DE EL SALVADOR

FACULTAD MULTIDISCIPLINARIA DE OCCIDENTE

DEPARTAMENTO DE INGENIERIA Y ARQUITECTURA

TRABAJO DE GRADO

DISEÑO Y DESARROLLO DE UN SISTEMA INFORMATICO DE EVALUACION DE PROYECTOS DE RENTABILIDAD SOCIAL PARA ALCALDIAS MUNICIPALES DE EL SALVADOR

PARA OPTAR AL GRADO DE

INGENIERO(A) DE SISTEMAS INFORMATICOS

PRESENTADO POR

CARLOS EDUARDO CÁRCAMO MENDOZA
RICARDO ANTONIO LÓPEZ MARTÍNEZ
JOSUÉ BENJAMÉN MORÁN JIMÉNEZ
YANCI KARINA NERIO
PABLO JOSÉ VINUEZA CAMPOS

DOCENTE ASESOR

INGENIÉRO ERNESTO ALEXANDER CALDERÓN PERAZA

DICIEMBRE, 2019

SANTA ANA, EL SALVADOR, CENTROAMÉRICA

UNIVERSIDAD DE EL SALVADOR

AUTORIDADES

M.Sc. ROGER ARMANDO ARIAS ALVARADO

RECTOR

DR. RAÚL ERNESTO AZCÚNAGA LÓPEZ

VICERRECTOR ACADÉMICO

ING. JUAN ROSA QUINTANILLA QUINTANILLA

VICERRECTOR ADMINISTRATIVO

ING. FRANCISCO ANTONIO ALARCÓN SANDOVAL

SECRETARIO GENERAL

LICDO. LUIS ANTONIO MEJÍA LIPE

DEFENSOR DE LOS DERECHOS UNIVERSITARIOS

LICDO. RAFAEL HUMBERTO PEÑA MARÍN

FISCAL GENERAL

FACULTAD MULTIDISCIPLINARIA DE OCCIDENTE

AUTORIDADES

M.Ed. ROBERTO CARLOS SIGÜENZA CAMPOS

DECANO

M.Ed. RINA CLARIBEL BOLAÑOS DE ZOMETA

VICEDECANA

LICDO. JAIME ERNESTO SERMEÑO DE LA PEÑA

SECRETARIO

ING. DOUGLAS GARCÍA RODEZNO

JEFE DEL DEPARTAMENTO DE INGENIERÍA

AGRADECIMIENTOS

Este trabajo de graduación está dedicado a:

A mi abuela Emilia Alas (Q.E.P.D.): Por todo su amor y paciencia que siempre recordaré.

Mi tia Carmen Cárcamo: Eternamente agradecido con mucho amor y respeto, por todo su amor, apoyo incondicional y todo su esfuerzo por sacar adelante a nuestra familia.

A mi primo hermano Hugo Nelson Cárcamo: Por todo su apoyo y por haberme motivado en todo momento a seguir adelante con mis estudios.

A mi prima hermana Norma Ivonne Cárcamo: Por todos sus consejos y apoyo incondicional.

A mis padres: José Ángel Cárcamo (Q.E.P.D.), María Ester Mendoza

Por darme la vida y una familia que siempre me han apoyado

A mis primas hermanas Celina Cárcamo y sus hijas

Abigail Cárcamo

Stephania Cárcamo

Con mucho cariño, por su comprensión y paciencia en todo momento.

A mi querida amiga Jennifer López: Por su comprensión, confianza y apoyo incondicional durante todo este proceso

A mis compañeros de tesis: Por habernos comprendido, apoyado y por todo esfuerzo dedicado a la culminación de este trabajo.

A nuestro asesor: Por su desinteresada colaboración y apoyo durante la realización de este documento.

Carlos Eduardo Cárcamo Mendoza

Agradezco a Dios, por haberme permitido cosechar este logo, a mi familia por el apoyo que a lo largo de estos años me ha brindado, a mis amigos y compañeros de tesis por su esfuerzo y dedicación, a los catedráticos que en todos estos años de una u otra manera abonaron a mi proceso de aprendizaje y formación, a cada uno de mis compañeros de estudio con los cuales compartimos alegrías, preocupaciones, decepciones, enojos y triunfos. Gracias por haber formado parte de este largo camino.

Ricardo Antonio López Martínez

Agradecer: en primer lugar, a **Dios** porque sé que sin Él mis proyecciones de lograr estar en este momento serian nulas, donde hubo limitaciones Él puso el carácter para transformarlas en oportunidades.

Segundo lugar a mis Padres, **Armando Morán y Deysi Jiménez** por enseñarme a confiar en Dios, por intentar ayudarme en lo que pudieron, sé que desearon hacer mucho más, pero más que un aporte económico, siempre modelaron una vida correcta, de esfuerzo, fe, disciplina y a que no es necesario tener dinero, sino, tener voluntad; siempre serán mi modelo a seguir.

Mis hermanos: **Cesiah**, por darme un apoyo sin que lo pidiera, por muchas veces tomar el papel de madre, que cuida, protege, provee, ayuda y corrige. **Edwin**, por estar ahí, por enseñarme a tener carácter.

Mi familia, **Mario Cruz**, como un hermano, siempre a mi lado; **Rodrigo Jiménez**, un apoyo sin falta; **Salvador Iraheta** (QEPD) sé que desearías estar acá; **Familias Morán-Jiménez.**

Mi novia y amiga: **Gabriela Castro** por acompañarme en este reto.

Amigos: **Sra. Marinela,** por estar para mi familia siempre, jamás podré pagarle por tanto; **Jonathan Abrego, Luis Martínez y familia, Carlos David Ramos, Cindy Perlera y familia, Ingrid Pérez, Alejandro Delgado, Francisco Aquino,** con mucho agradecimiento.

A la familia OVNICO S.A. de C.V. por proveerme un empleo cuando más lo necesite.

A mi grupo de tesis y asesor Ing. Calderón, sin duda, mis amigos.

Josué Benjamín Morán Jiménez

Agradezco a Dios, a mi mamá por todo el apoyo, a mis compañeros de tesis, a mi asesor y a mí por mi esfuerzo. Gracias, gracias, gracias.

Yanci Nerio

Señoras y señores docentes, compañeros de equipo, tengo el gusto de expresar estas palabras de agradecimiento, como mi único medio de expresar el éxito logrado junto con mis compañeros de equipo.

Por grandes razones, agradezco primeramente a Dios, quien me dio la oportunidad de la vida, de estudiar en la Universidad de El Salvador y concluir ahora mis estudios superiores.

A mis padres, quienes me apoyaron en todo lo indispensable, a mis hermanos que me apoyan en lo que otras personas jamás me ayudarían, que me dieron apoyo moral cuando me sentía caer y a mis amigos, que me apoyaron con su tiempo y comprensión en mis estudios.

Agradezco a la Universidad de El Salvador, que, con sus docentes, lograron pulir mis módicas facultades y lograr mi titulación en la carrera de Ingeniería de Sistemas Informáticos.

Gracias,

Pablo Vinueza

INDICE

INTRODUCCION

En el presente documento se plantea el diseño y desarrollo de un sistema de evaluación de proyectos con enfoque social para las instituciones municipales, con el fin de dar una referencia acertada a la toma de decisión que en la actualidad le compete al consejo municipal para ayudar a utilizar de una manera más eficiente los fondos que el estado distribuye a las alcaldía, las cuales, tienen un gran reto al saber que la población tiene muchas demandas y necesidades que deben ser satisfechas y los recursos tienen la particularidad de ser limitados o también no son aprovechados al máximo. También como este trabajo busca que con la creación de esta herramienta no solo las decisiones sobre qué proyecto realizar sea más fácil, sino también, que aumente la plusvalía de la ciudad, que aporte al desarrollo y avance integral de la municipalidad.

Debido a estas necesidades, hemos decidido realizar este trabajo de grado, apoyándonos con la tecnología para resolver esta problemática, el saber elegir lo mejor para todos. Denominamos este trabajo como: **Diseño y desarrollo de un Sistema informático de Evaluación de Proyectos de Rentabilidad Social para Alcaldías Municipales de El Salvador**, el cual describimos el contenido capitular de la siguiente manera:

Capítulo I: Generalidades, se explica él porque es necesaria la realización del siguiente trabajo, planteamiento del problema, objetivos, justificación del desarrollo, coberturas y alcance.

Capitulo II: Marco Teórico, explicamos algunos conceptos fundamentales como también la idea lógica del funcionamiento del software, descripción de indicadores.

Capitulo III: Diseño del Sistema de Evaluación de Proyectos, la descripción general del software, diseño de base de datos, interfaz, seguridad, acceso, comunicación de interfaces, etc.

Capitulo IV: Desarrollo del Sistema de Evaluación Social de Proyectos, descripción de desarrollo en componentes de Base de datos, FrontEnd, BackEnd, ACL, y respaldo.

Se presentan finalmente, las conclusiones, recomendaciones, anexos y manual de usuario.

CAPITULO I: GENERALIDADES

1.1 Planteamiento del Problema

Uno de los componentes del desarrollo de los países se sustenta en la base de la inversión, tanto privada como pública. En la inversión pública, debido a lo limitado de los fondos asignados para tal fin, surge la imperiosa necesidad de invertir en forma metódica y estratégica.

Las Alcaldías de los Municipios de los Departamentos de El Salvador, tienen asignados fondos para inversión, a través del Fondo para el Desarrollo Económico y Social de las Municipalidades de El Salvador (FODES), Esta institución fue creada por mandato constitucional y destinada a la inversión de proyectos que beneficien al desarrollo de los municipios, así como para sufragar algunos gastos de funcionamiento de la entidad municipal. La asignación para cada municipalidad obedece a lo establecido en la Ley de Creación del Fondo para el Desarrollo Económico y Social de los Municipios de El Salvador (FODES), aprobada por la Asamblea Legislativa en 1988, y por el Reglamento de la Ley que fue aprobado en 1998. Las asignaciones de los últimos años se observan en la tabla 1.

MUNICIPIO	2018	2019
SANTA ANA	$3,599,658.75	3880910.68
CHALCHUAPA	$2,900,294.25	3126902.78
METAPAN	$3,627,774.73	3911223.44
COATEPEQUE	$2,770,185.31	2986628.04

EL CONGO	$1,681,769.98	1813,171.61
TEXISTEPEQUE	$1,762,200.35	1,899,886.25
CANDELARIA DE LA FRONTERA	$1,824,952.54	1,967,541.45
SAN SEBASTIAN SALITRILLO	$1,124,040.73	1,211,865.34
SANTA ROSA GUACHIPILIN	$869,964.08	937,936.93
SANTIAGO DE LA FRONTERA	$890,790.41	960,390.48
EL PORVENIR	870,890.69	938,935.95
MASAHUAT	737,453.80	795,073.23
SAN ANTONIO PAJONAL	668,623.52	720,865.05

Tabla 1.1 Fuente: Distribución FODES compartida en portal web del Instituto Salvadoreño de Desarrollo Municipal ISDEM.

Uno de los principales objetivos de la Ley del FODES es asegurar justicia en la distribución de los recursos, tomando en cuenta las necesidades sociales, económicas y culturales de cada municipio, siendo competencia de las alcaldías municipales invertir en forma óptima dicho fondo.

En el Reglamento de la Ley del FODES se establece, en el artículo 10, que los municipios utilizarán el 80% de los fondos asignados para desarrollar proyectos de obras de infraestructura, en beneficio de sus habitantes; y el 20% para gastos de funcionamiento. Los fondos necesarios para financiar este 20%, se tomarán del aporte que otorgue el Estado, por medio del Instituto Salvadoreño de Desarrollo Municipal.

De ese 80% para inversión, se podrá utilizar hasta el 5% para gastos de pre-inversión. Se entenderán como gastos de pre-inversión para los efectos del presente reglamento, los

siguientes: Elaboración del Plan de Inversión del municipio; Elaboración de carpetas técnicas; Consultorías; Publicación de Carteles de Licitación Pública y Privada.

Los proyectos deben ser formulados de conformidad a las normas técnicas de elaboración de proyectos, contenidas en las guías proporcionadas por el Fondo de Inversión Social para el Desarrollo Local de El Salvador y acorde a la reglamentación de la Corte de Cuentas de la República.

La Ley del FODES no incluye los mecanismos internos, procedimientos o criterios para elegir los proyectos ni mucho menos para priorizar su ejecución.

De allí que el problema se aborda desde tres vías: la ausencia de criterios de evaluación por los tomadores de decisiones, la falta de competencia en evaluación de proyectos de los encargados de ejecutarlos y la falta de procedimientos establecidos para llevar a cabo una evaluación con criterios técnicos.

La ausencia de criterios técnicos de evaluación de proyectos para definir cuáles proyectos ejecutar prioritariamente en las alcaldías de los municipios, desde un punto de vista financiero, que permita el uso eficiente del Fondo asignado, es uno de los principales problemas a los que se enfrentan los Concejos Municipales cada año.

La Ley del FODES establece que los Concejos Municipales serán responsables de administrar y utilizar eficientemente los recursos asignados en una forma transparente, en caso contrario responderán conforme a la Ley pertinente por el mal uso de dichos fondos. Aún con esta consideración, en las alcaldías se desconocen mecanismos técnicos para decidir ejecutar un proyecto.

Los proyectos que se aprueban se seleccionan y priorizan según criterios personales de los miembros del Concejo Municipal, muchas veces con fines de ganar popularidad política; otro de los factores que influyen en la toma de decisiones es el momento político en que se analizan

y algunos datos históricos o experiencias en proyectos similares; sin embargo, estas decisiones solo quedan asentadas en acta, sin someterse a una evaluación de proyectos basado en criterios técnicos o sociales previamente cuantificados.

Por otra parte, los encargados de las unidades de proyectos, en su mayoría, son empleados sin la formación profesional o técnica requerida, por lo que no pueden emitir un dictamen sobre la factibilidad económica de cada proyecto al carecer de herramientas que les permitan dicha evaluación previa a la decisión de ejecutarse.

A pesar que el Instituto Salvadoreño de Desarrollo Municipal (ISDEM), encargado de la distribución del FODES, puede proporcionar asesoría técnica para la elaboración de carpetas técnicas, no existe un procedimiento establecido para la evaluación de proyectos; las municipalidades en su mayoría no cuentan con recursos adecuados para la evaluación de proyectos, por lo que se limitan a calcular carpetas técnicas con presupuestos establecidos, donde las decisiones de ejecución de los proyectos se realizan de forma discrecional basada en las áreas de inversión establecidas y número de beneficiarios calculado.

Considerando la situación planteada, el presente estudio se orientó a responder la siguiente pregunta de investigación:

¿Generando criterios de evaluación de proyectos municipales, se hará un uso eficiente de los Fondos asignados para las Alcaldías de los Municipios?

El sistema propuesto está diseñado para ser aplicado en las Alcaldías Municipales de El Salvador, sin embargo, la recopilación de la información que sirve de base para el estudio, se delimita a las alcaldías del Departamento de Santa Ana, en la Zona Occidental de El Salvador, considerando principalmente las condiciones especiales de la Alcaldía de San Sebastián Salitrillo como base para su aplicación y posterior comparación.

La Ley de creación del FODES (Asamblea Legislativa El Salvador, 1988) establece que los recursos provenientes del Fondo Municipal podrán invertirse en la adquisición de vehículos para el servicio de recolección y transporte de basura, maquinaria, equipo y mobiliario y su mantenimiento para el buen funcionamiento; instalación, mantenimiento y tratamiento de aguas negras, construcción de servicios sanitarios, baños y lavaderos públicos, obras de infraestructura relacionada con tiangues, rastros o mataderos, cementerios, puentes, carreteras y caminos vecinales o calles urbanas y la reparación de éstas. Industrialización de basuras o sedimento de aguas negras, construcción y equipamiento de escuelas, centros comunales, bibliotecas, teatros, guarderías, parques, instalaciones deportivas, recreativas, turísticas y campos permanentes de diversiones; así como también para ferias, fiestas patronales, adquisición de inmuebles destinados a las obras descritas; y al pago de las deudas institucionales contraídas por la municipalidad y por servicios prestados por empresas estatales o particulares; incluyéndose del desarrollo de infraestructura, mobiliario y funcionamiento relacionados con servicios públicos de educación, salud y saneamiento ambiental, así como también para el fomento y estímulo a las actividades productivas de beneficio comunitario y programas de prevención a la violencia.

De todos los proyectos en los que las municipalidades pueden invertir el 80% del FODES surge una variedad grande y se debe elegir bien para lograr realizar una mayor cantidad y/o con niveles aceptables de calidad, se debe considerar que eso depende de la ubicación, extensión y costo de cada proyecto, sin excluir a la población beneficiada. De lo contrario, se puede caer en el vicio de asignar fondos importantes a proyectos, que, si bien pueden ser relevantes, son extremadamente costosos y que no representen beneficios sociales que justifiquen su ejecución y el uso de los recursos.

1.2 Justificación de la Investigación

Las Alcaldías Municipales son responsables de la inversión pública en sus municipios, de tal manera, que deben procurar una adecuada gestión de sus recursos limitados; de allí la importancia de aplicar una técnica de evaluación de proyectos sociales que suponen un beneficio a las comunidades pertenecientes a las municipalidades.

El Código Municipal de El Salvador (Asamblea Legislativa El Salvador, 1986), en el Título III, De la Competencia Municipal, Capítulo Único, expresa las veintiocho competencias de los Municipios, entre las que se destacan las enfocadas a la elaboración y ejecución de planes y programas de desarrollo económico y social a nivel local, tales como la que menciona el Artículo 4:

Numeral 1: Compete a los Municipios la elaboración, aprobación y ejecución de planes de desarrollo urbanos y rurales de la localidad;

Numeral 4: La promoción y desarrollo de la educación, la cultura, el deporte, la recreación, las ciencias y las artes;

Numeral 5: La promoción y desarrollo de programas de salud, como saneamiento ambiental, prevención y combate de enfermedades; entre otros.

En razón de lo anterior toma importancia la adecuada inversión que los gobiernos municipales de El Salvador deben y pueden realizar, cuando se tienen recursos limitados, empleándolos adecuadamente buscando maximizar las inversiones.

La evaluación de proyectos debe incorporar criterios sociales; pues la evaluación de la rentabilidad económica privada es muy limitada al intentar medir los períodos de retorno de proyectos de carreteras, agua potable, electrificación, entre otros, debido al factor social que estos proyectos poseen, los proyectos de rentabilidad social resuelven necesidades que están

concatenadas entre sí, y que al suplirse adecuadamente, redunda en múltiples beneficios para la sociedad en general.

La evaluación de proyectos reviste vital importancia en función de optimizar los exiguos recursos que administran las alcaldías, ante esa situación es necesario establecer un proceso sistemático de evaluación de sus proyectos con el propósito de priorizar las necesidades más apremiantes de la comunidad, basado en los criterios técnicos.

Se considera de suma importancia el crear una metodología que facilite la evaluación, el ordenamiento lógico de los criterios tomados a través de archivos digitales y el cálculo económico de los criterios considerados; haciendo más sencillo cuantificar y cualificar los proyectos analizados, permitiendo así dar prioridad a aquellos proyectos que tienen mayor rentabilidad, pero no solo rentabilidad desde el aspecto privado, no simplemente analizar los proyectos por medio de su tasa interna de retorno o de su valor presente, sino que se incorporen otros criterios, tales como el número de beneficiarios y el área prioritaria de inversión (rentabilidad social).

Con el sistema propuesto se pretende aportar una herramienta que permita conocer y decidir sobre los beneficios y costos al invertir en un determinado proyecto, considerando su período de recuperación, su tiempo de retorno, sus costos de mantenimiento a lo largo de su vida útil y evaluar si conviene realizarlo o buscar otras alternativas; orientando la toma de decisiones de inversión facilitando la clasificación de factores que por su naturaleza matemática suelen ser de difícil cálculo, y de complicada comprensión y análisis.

1.3 Cobertura y Alcance

Cobertura Espacial

El proyecto se delimita al área o unidad de Proyectos de cualquier Alcaldía Municipal de El Salvador, utilizando los recursos presentados en el apartado correspondiente en este mismo documento, pretendiendo ejecutarlo en un período máximo de un año calendario.

Cobertura Temporal

Se considerarán los proyectos presentados durante los años 2018 y 2019 años en que se desarrolla la investigación.

Alcance

El alcance del proyecto se limita a la evaluación de proyectos Ex ante, que se realiza en la primera fase del ciclo de vida de un proyecto, es decir, antes de la ejecución del mismo, con el objetivo de evaluar sus costes y beneficios tanto económicos como sociales.

No se considerará el estudio de metodologías y cálculos econométricos para determinar precios sociales, divisas sociales y rentabilidades sociales exigidas, dichos datos se asumen

serán proporcionados por el Estado o en su defecto se tomarán datos de mercado. (Se asumirá un entorno sin distorsiones)

Se desarrollará el sistema para el uso de cualquier alcaldía, sin embargo, no se incluye la implementación a nivel nacional del sistema de evaluación municipal.

No se incluyen los mecanismos o metodologías de diagnóstico de necesidades de las comunidades de cada Municipio, partiendo de la premisa que las Municipalidades identifican adecuadamente los problemas de sus pobladores.

1.4 Objetivos

1.4.1 Objetivo General

Analizar, diseñar y desarrollar un Software de Evaluación basado en la Rentabilidad Social para los proyectos impulsados en las alcaldías municipales.

1.4.2 Objetivos Específicos

Desarrollar una aplicación que potencialmente pueda ser utilizada por todas las alcaldías del país, independiente de la ideología política.

Realizar un diagnóstico de la metodología y criterios actuales que se utilizan en la evaluación, elección y priorización de los proyectos que se presentan en las Municipalidades, considerando los criterios establecidos para tal fin.

Diseñar una base de datos que responda a las exigencias de almacenamiento y concurrencia.

Proveer un mecanismo de seguridad que garantice la consistencia, integridad y privacidad de los datos.

1.5 Metodología de la investigación

1.5.1Tipo de investigación realizada

El Tipo de estudio realizado para llevar a cabo la investigación es proyectivo, este tipo de investigación consiste en la elaboración de una propuesta, un plan, un programa o un modelo, como solución a un problema o necesidad de tipo práctico, ya sea de un grupo social, o de una institución, o de una región geográfica, en un área particular del conocimiento, a partir de un diagnóstico preciso de las necesidades del momento, los procesos explicativos o generadores involucrados y de las tendencias futuras, es decir, con base en los resultados de un proceso investigativo. (Hurtado de Barrera, 2008)

1.5.2 Instrumentos y/o técnicas para la investigación

El estudio se diseñó por fases, en cada una de las cuales se desarrollaron actividades específicas.

La fase 1 consistió en establecer los contactos en las diferentes municipalidades, para iniciar la investigación sobre cómo evalúan actualmente los proyectos que cada año llegan desde las comunidades a las alcaldías. En esta fase se solicitó audiencia con los alcaldes y con los encargados de proyectos para exponer el objetivo del estudio. Esta es la fase descriptiva.

La fase 2 consistió en la observación directa sobre el manejo de proyectos, revisando las carpetas técnicas y analizando la información contenida en ellas para establecer qué criterios utilizan en la actualidad para su aprobación y ejecución. Realizando el análisis de los criterios y

definir criterios de evaluación económica que permitan sustentar la decisión a tomar. Esta fase es la analítica y comparativa (de gabinete).

La fase 3 consistió en la construcción de la propuesta del sistema de evaluación de proyectos, a través de criterios económicos y criterios de orden social que puedan establecerse en un procedimiento sistemático para priorizar, con base en esos criterios la conveniencia de ejecutar un determinado proyecto o no, con el fin de optimizar los recursos con los que cuentan las municipalidades, garantizando la inversión pública eficiente. Esta fase es la predictiva.

La fase 4 consistió en el diseño de la herramienta informática que, al introducir los datos numéricos provenientes de las carpetas técnicas de los proyectos, pueda calcular los valores económicos que permitan priorizar los proyectos a ejecutar según el análisis multicriterio establecido. Esta es la fase proyectiva propiamente.

1.5.3 Definición de la población

El Universo total consistió en 1 Alcaldía Municipal del Departamento de Santa Ana.

Se tomó como muestra la alcaldía del municipio de Santa Ana. La muestra se estableció por conveniencia, debido a que se tuvo acceso a la información pertinente para el estudio a través de las unidades de Proyectos y la Unidad de Adquisiciones y Contrataciones (UACI) y por la diversidad de proyectos que en estas alcaldías se ejecutan.

1.5.4 Metodología utilizada

Se utilizó la entrevista estructurada; la cuál fue dirigida al personal involucrado en proyectos de la municipalidad, el cual en la práctica puede ser un delegado de proyectos, el edil, un concejal o el jefe de la UACI.

La entrevista tiene una estructura piramidal, iniciando con preguntas cerradas para luego pasar a preguntas más abiertas.

CAPITULO II: MARCO TEÓRICO

2.1 Antecedentes

La inversión social cada día va tomando una importancia mayor, no podríamos comenzar este capítulo sin antes dar una pequeña descripción que amplíe el panorama acerca de la concepción que tenemos acerca de los proyectos sociales.

En su primer momento, como es de esperarse, la situación política y económica de nuestro país y lo que el poder ejerce a través de estos van generando que los puestos públicos sean utilizados por personajes que mediante su popularidad social obtienen los votos necesarios para sentarse en los escritorios más importantes de las municipalidades: Las Alcaldías. Segundo, la falta de preparación acerca del puesto de un alcalde o su consejo Municipal (aunque esto podría ser relativo) aumenta la probabilidad de que la gestión de los recursos económicos no sea aprovechados de la manera óptima. Tercero, según lo que hemos venido describiendo, podemos afirmar que el poder municipal de la municipalidades están cedidos a empresarios que aunque con capacidades y muchos graduados de instituciones de prestigio aun no conocen de la gestión de proyectos de rentabilidad social, puesto que sus empresas(si la tienen), predominará las políticas mercadológicas que involucran el fin del lucro y la rentabilidad económica de la organización en sí, descuidando en sus objetivos alcanzar la excelencia en cuanto a la sostenibilidad de áreas importantes de las personas.

2.1.1 Proyecto

Es importante definir qué se entiende por proyecto, y para eso se presentan las definiciones que aportan los especialistas en el tema. (Cohen, 2010) establece que la ONU ha definido un proyecto como "una empresa planificada que consiste en un conjunto de actividades interrelacionadas y coordinadas para alcanzar objetivos específicos dentro de los límites de un presupuesto y un periodo dados".

2.1.2 Proyecto social

Una definición aceptable es que un proyecto social es la unidad mínima de asignación de recursos, que a través de un conjunto integrado de procesos y actividades pretende transformar una dificultad de la realidad, disminuyendo o eliminando un déficit, o solucionando un problema (Cohen, 2010).

Las condiciones que debe cumplir un proyecto social pueden establecerse de la siguiente manera:

En primer lugar, se debe definir el problema que se pretende resolver.

Definir y establecer objetivos de impacto claramente definidos.

Identificar a la población objetivo a la que está destinada el proyecto.

Localización espacial de los beneficiarios.

(Cohen, 2010) Afirma que "los problemas sociales se definen como carencias o déficits existentes en un grupo poblacional. Constituyen una brecha entre lo deseado por la sociedad y la realidad (Más que un planteamiento filosófico es importante destacar que muchas necesidades de la sociedad pueden ser ficticias y efímeras, por tal razón las necesidades sociales a las cuales se debe realizar el enfoque deben consistir en elementales y fundamentales para la subsistencia y el adecuado funcionamiento de la sociedad). Por tal razón la decisión de ejecución de

proyectos sociales no se puede fundamentar en meras suposiciones o creencias". Un programa o proyecto social es sustentable en la medida que exista capacidad instalada (recursos físicos, humanos y financieros) para que los procesos requeridos sean adecuadamente implementados. Es sostenible cuando los impactos producidos perduran en el tiempo.

2.2 La evaluación de proyectos de rentabilidad social en El Salvador

Luego de recolectar la información, por medio de entrevistas con el área de UACI de la alcaldía de Santa Ana, y analizar las respuestas obtenidas, se presentan los siguientes resultados:

Tabla 2.1: Tabla de Resultados Obtenidos

PREGUNTA	RESULTADOS
¿Cuántos proyectos son ejecutados por la municipalidad en un año?	La cantidad de proyectos ejecutados depende de las dimensiones de la municipalidad, un municipio como Santa Ana, ejecuta más proyectos que un municipio como San Sebastián Salitrillo, debido a que la cantidad de proyectos está relacionada con el presupuesto y la asignación del Fondo FODES que la municipalidad cuenta. En un año promedio se ejecutan alrededor de 10 proyectos. Sin embargo, la cantidad de proyectos a veces no es lo relevante, debido a que podrían en un caso ejemplo ejecutarse 3 proyectos cuyos montos sean grandes.
¿Qué tipo de proyectos son ejecutados por la Municipalidad?	Los gobiernos municipales ejecutan diferentes tipos de proyectos dentro de los que encontramos, proyectos de vialidad urbana, introducción de agua potable, alumbrado público, electrificación, edificaciones deportivas, edificaciones de escuelas, reparación de calles, etc.
¿Cuál es el proceso de evaluación de proyectos que realizan?	No existe un proceso formal como tal, sino que el Concejo Municipal establece prioridades sobre las cuales trabajan, muchas de estas están vinculadas a la campaña electoral previa a conformar el gobierno municipal, los pobladores de diferentes comunidades expresan sus necesidades, las cuales son tomadas en cuenta. Pero un proceso formal de evaluación de proyectos NO existe.

¿Cuáles criterios son tomados en cuenta al momento de decidir ejecutar un proyecto?	Existen varios elementos principales a la hora de decidir ejecutar un proyecto como: La cantidad de beneficiarios, se busca que el proyecto beneficie a la mayor cantidad de familias; monto del proyecto: proyecto con montos accesibles son más fáciles de financiar, naturaleza de la comunidad: algunas comunidades debido a su situación de vulnerabilidad serán prioritarias; impacto: la relevancia y la importancia que dicho proyecto puede representar en el municipio.
¿De todos los criterios mencionados, existe una cuantificación formal sobre ellos?	Los gobiernos municipales están muy cerca de la población, y por tal razón, aunque no se tenga una escala de evaluación o criterios cuantificables, los mismos son conocidos de forma discrecional debido a que los concejales y ediles conocen su municipio.
¿Quién toma la decisión final para la ejecución de un proyecto?	El Concejo Municipal y lo realiza de una forma discrecional.
¿Existen proyectos municipales que hayan fracasado?	En la gestión actual NO, pero en gestiones anteriores se mencionaron varios proyectos que resultaron mal, calles que se arruinaron bien rápido, proyectos de alcantarillado que no se ejecutaron en su totalidad, y críticas a proyectos emergentes sobre el bajo impacto alcanzado en las comunidades.
¿Considera importante el establecer un procedimiento formal para la evaluación de proyectos?	Sería muy importante, debido a que en las municipalidades a veces no se cuenta con una unidad de proyectos como tal, a veces un concejal se hace cargo de dicha función, pero no cuenta con herramientas necesarias para ello.
¿Qué características desearía que tenga la metodología de evaluación de proyectos municipales?	Que permita evaluar proyectos de todas las naturalezas, de agua potable, de electrificación, deportes, escuelas, en fin, de todo tipo de proyectos municipales, que además que permita manejar diferentes vidas útiles para los proyectos, y que se puedan evaluar dos proyectos aunque sus vidas útiles y naturales sean diferentes, que usen los indicadores que actualmente tomamos, la cantidad de beneficiarios, las áreas prioritarias definidas por el Concejo Municipal.

2.3 Características de las Municipalidades en El Salvador, su estructura y funcionamiento.

En los Municipios de El Salvador son utilizados conceptos básicos dentro de lo legal y administrativo, los cuales sirven para identificar las actividades y plazas que son propias del quehacer Municipal en general; entre ellas podemos mencionar las siguientes:

El Código Municipal emitido según decreto legislativo número 274 del 31 de enero de 1986, publicado en el Diario Oficial número 3 tomo 290 del 05 de febrero del mismo año, establece los siguientes conceptos:

Municipio

Es la Unidad Política Administrativa primaria dentro del órgano estatal, establecida en un territorio determinado que le es propio, organizado bajo un ordenamiento jurídico que garantiza la participación popular en la formación y conducción de la sociedad local, con autonomía para darse su propio Gobierno".

(Art. 2 C.M.).

Concejo Municipal

"Es la máxima autoridad del Municipio y será presidido por el alcalde. El Gobierno municipal estará ejercido por un Concejo que tiene carácter deliberante y normativo, lo integra un alcalde, un Síndico y dos o más Regidores según el número de habitantes." (Art. 24 CM.).

Hacienda Pública Municipal

"Comprende los bienes, ingresos y obligaciones del Municipio. Gozarán de las mismas exoneraciones, garantías y privilegios que los bienes del Estado". (Art. 60 CM.)

Presupuesto Municipal

"Es una exposición detallada de los ingresos y gastos previstos por el Concejo Municipal, para el ejercicio a que se refiere". (Art. 73 CM)

Ejercicio del Presupuesto

"Es el período para el cual se prepara el presupuesto; comprende el año fiscal de Enero a Diciembre de cada año". (Art. 72 CM)

Fondos Municipales

"Es el término genérico que agrupa los ingresos provenientes de la aplicación de las leyes, ordenanzas, reglamentos y disposiciones relativas a impuestos, tasas, derechos, compensaciones y otras contribuciones, así como las que le resultan de operaciones comerciales, donativos o de cualquier otro título, que pertenezcan a la municipalidad o estén bajo su custodia".

Arbitrio

"Sinónimo de impuestos, el derecho o imposición con que se obtienen fondos para los gastos públicos, por lo general de las Municipalidades".

Impuesto

"Prestación en dinero obligatoria y unilateral que exige coactivamente el Poder Público a todas aquellas personas naturales o jurídicas, cuya situación coincida con las que la ley señala como hecho generador del crédito fiscal o Municipal".

Impuesto Municipal

"Son Impuestos Municipales, los tributos exigidos por los Municipios, sin contraprestación algún a individualizada" (Art. 4 Ley General Tributaria Municipal)

Tasas Municipales

"Son tasas Municipales, los tributos que se generan en ocasión de los servicios públicos de naturaleza administrativa o jurídica prestados por los Municipios"

(Art. 5 Ley General Tributaria Municipal)

2.4 Características de la evaluación de proyectos en las Municipalidades de El Salvador.

Según nuestra investigación, se realizó el análisis de información, identificando las siguientes características y/o retos a vencer en las evaluaciones de proyectos en las alcaldías, se enumeraron y se les indico la frecuencia con la que se encontró el problema.

Nº	Problema principal	Frecuencia encontrada
1	No existe un procedimiento sistemático para la evaluación de proyectos ni una persona preparada exclusivamente para realizarlo.	100%
2	No se unifican criterios para la toma de decisiones en la elección de los proyectos a ejecutar	100%

3	No existe un instrumento que permita evaluar los diferentes tipos de proyectos que pueden ejecutarse por parte de la Alcaldía.	100%
3	No se cuantifican los datos de los proyectos, es decir no hay una escala de valoración para respaldar la toma de decisiones en cuanto a su ejecución.	100%

Tabla 2.2 . Priorización de problemas encontrados a través del diagnóstico

2.5 Conceptos sobre inversión

Es necesario para entender la problemática planteada, explicar algunos conceptos y definiciones relacionados a los proyectos y evaluaciones.

2.5.1 Proyecto

Un proyecto es una actividad grupal temporal para producir un producto, servicio, o resultado, que es único. (Según PMI).

2.5.2 Dirección de proyectos

Es la aplicación del conocimiento, de las habilidades, y de las técnicas para ejecutar los proyectos en forma eficiente y efectiva. Es una competencia estratégica para las organizaciones, y les permite atar los resultados de los proyectos a las metas del negocio, y así competir mejor en su mercado. (Según PMI).

2.5.3 Proyecto social

Una definición aceptable es que un proyecto social es la unidad mínima de asignación de recursos, que a través de un conjunto integrado de procesos y actividades pretende transformar una dificultad de la realidad, disminuyendo o eliminando un déficit, o solucionando un problema (Cohen, 2010).

Las condiciones que debe cumplir un proyecto social pueden establecerse de la siguiente manera:

En primer lugar, se debe definir el problema que se pretende resolver.

Definir y establecer objetivos de impacto claramente identificados.

Identificar a la población objetivo a la que está destinada el proyecto.

Localización espacial de los beneficiarios.

(Cohen, 2010) Afirma que "los problemas sociales se definen como carencias o déficits existentes en un grupo poblacional. Constituyen una brecha entre lo deseado por la sociedad y la realidad (Más que un planteamiento filosófico es importante destacar que muchas necesidades de la sociedad pueden ser ficticias y efímeras, por tal razón las necesidades sociales a las cuales se debe realizar el enfoque deben consistir en elementales y fundamentales para la subsistencia y el adecuado funcionamiento de la sociedad). Por tal razón la decisión de ejecución de proyectos sociales no se puede fundamentar en meras suposiciones o creencias". Un programa o proyecto social es sustentable en la medida que exista capacidad instalada (recursos físicos, humanos y financieros) para que los procesos requeridos sean adecuadamente implementados. Es sostenible cuando los impactos producidos perduran en el tiempo

2.6 Evaluación Ex ante

Existen dos tipos de evaluación según el momento que se realiza y el objetivo perseguido (Hugo Navarro, 2006):

a) La evaluación ex-ante, que se realiza antes de la inversión y la operación. Ella permite estimar tanto los costos como el impacto (o beneficios) y así adoptar la decisión (cualitativa) de implementar o no el proyecto.

A partir de ella resulta posible priorizar distintos proyectos e identificar la alternativa óptima para alcanzar los objetivos de impacto perseguidos.

La evaluación ex-ante trata de simular el efecto de un proyecto antes de que este se ponga en práctica o entre en operación. El objetivo de la evaluación ex-ante es proporcionar elementos de juicio para determinar cuál es el proyecto o la combinación de proyectos que más conviene a la población en términos del cambio de las condiciones de vida de los beneficiarios.

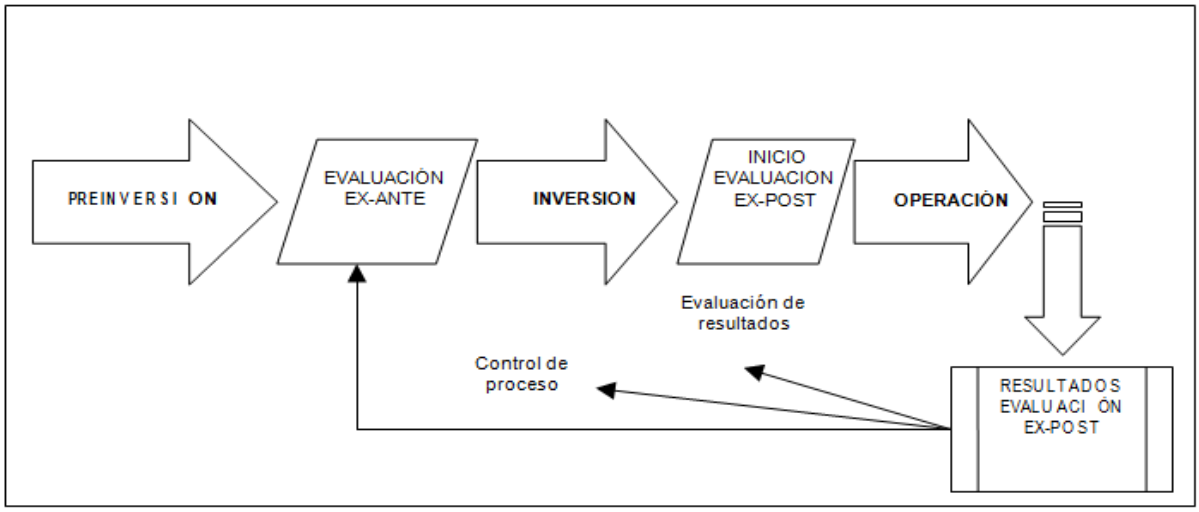


Figura 1: Evaluaciones de Impacto y el Ciclo del Proyecto

Fuente: Navarro, Hugo. Pauta metodológica de evaluación de impacto ex-ante y ex-post de programas sociales de lucha contra la pobreza

Para las evaluaciones ex-ante, el diseño del grupo de control será representado por la situación actual de pobreza de los beneficiarios del proyecto (el "antes", línea base o situación sin proyecto), y el grupo de tratamiento corresponderá a la simulación de la situación con proyecto (el "después"). Así, el impacto del proyecto será la diferencia en cualquier variable de resultado antes y después de la ejecución del proyecto.

Definidas las alternativas de proyecto, estas deben ser evaluadas para seleccionar la que presenta una mejor relación entre los costos de su implementación y el impacto estimado.

2.6.1 El Análisis Costo-Impacto (ACI) es la metodología que permite seleccionar la alternativa que maximiza el impacto al menor costo posible, en otras palabras, escoger la opción que presenta el menor costo por unidad de impacto. Por consiguiente, requiere el análisis de costos y del impacto. Ambos presentan diferencias de operacionalización según la naturaleza del proyecto.

En los proyectos de gran escala, se deben realizar comparando todas sus alternativas. En un programa que incluye un conjunto de pequeños proyectos con una sola alternativa, que comparten los objetivos de impacto y la población objetivo.

En programas en los que concursan pequeños proyectos con diferencias de población objetivo y/u objetivos de impacto, la evaluación se debe complementar con un análisis multi criterio.

2.6.2 Calcular los costos (análisis de la eficiencia). Los costos de un proyecto aluden al valor económico de cada uno de los bienes y servicios utilizados, independientemente de quién afronte su financiamiento. No se debe confundir costo con egreso. En cada alternativa se deben identificar los costos relevantes que se deben afrontar durante la vida del proyecto.

Para evaluar proyectos grandes (tipo a) hay que identificar los factores diferenciales (aquellos que implican mayores o menores costos). Es posible que existan actividades comunes (como el desarrollo e implementación de un sistema de monitoreo), que, por lo mismo, no contribuyen a

la toma decisiones, por lo que es preferible posponer su estudio para la fase de Programación. En proyectos pequeños se deben considerar todos los costos.

Los costos pueden clasificarse en:

2.6.3 Costos de capital: son los que se deben afrontar para adquirir bienes cuya duración en el proyecto (vida útil) es superior a un año. Normalmente, el desembolso debe hacerse durante la ejecución (inversión) para que puedan ser utilizados en la operación. Si es necesario reponer dichos bienes o realizar ampliaciones, tales erogaciones también forman parte de los costos de capital. Estos siempre se consignan en el período anterior a su utilización. Los costos de capital más comunes en los proyectos sociales son los de terreno, construcción, equipamiento e inversiones complementarias.

2.6.4 Costos de mantenimiento: son los requeridos para mantener el estándar de calidad y volumen de producción de los bienes de capital (equipos, edificios, etc.). Normalmente, se calculan como una proporción de los costos de capital del proyecto para cada período.

2.6.5 Costos de operación: se derivan de la compra de bienes y/o servicios cuya vida útil es inferior a un año. En los costos de operación se distinguen los:

a) Directos: Derivados de los insumos y personal imprescindibles para la realización del proyecto. Forman parte de los procesos principales (en el proyecto atención primaria de salud serían las enfermeras, paramédicos, fármacos, etc.).

b) Indirectos: No son imprescindibles, pero permiten aumentar la eficiencia. Forman parte de los procesos de apoyo (en el mismo proyecto, serían la supervisión, capacitación, etc.).

2.6.6 Costos adicionales de los usuarios: En los proyectos sociales es necesario tener en cuenta los costos en que debe incurrir la población objetivo para recibir los productos del proyecto. Normalmente, estos son los costos de movilización y el valor del tiempo de traslado y de espera (medida en horas hombre, dividiendo el sueldo mínimo mensual por 240 horas/mes).

Siempre se deben considerar los costos de oportunidad. Las donaciones y el trabajo voluntario, que son gratuitos, implican costos económicamente cuantificables. Si no se los incluye como tales, se asume que los recursos aportados son infinitos. Dichos costos deben imputarse en los costos de capital o de operación, según sea el caso.

Los recursos financieros utilizados en el proyecto tienen un costo de oportunidad calculado en base a lo que podrían rendir si se los destinara a inversiones alternativas (depósitos, acciones u otro tipo de proyectos). Mientras mayor es el horizonte del proyecto, mayor es su importancia. Normalmente, en los proyectos sociales, este costo se traduce en una tasa de descuento de 12% anual o aquella que este publicada como tasa social de descuento del país.

Hay que tomar en cuenta sólo los costos relevantes. Los egresos menores (como útiles de oficina) se deben agrupar en rubros genéricos. Los costos se deben imputar a precios de mercado, en el lugar del proyecto. Esto elimina la necesidad de realizar estimaciones sobre la inflación futura. La evaluación ex-ante se realiza utilizando los costos de oportunidad, pero para elaborar el presupuesto de la alternativa seleccionada (en la etapa de la programación) se deben considerar

Para elaborar un flujo de costos hay que tener presente:

1. Los períodos parten del "año cero", que corresponde a la etapa de ejecución, en la que se realizan las inversiones. Los períodos siguientes incluyen los costos de operación, de mantenimiento y reposición o ampliación de la inversión.
2. Los montos consignados en cada período deben expresarse en moneda de igual poder adquisitivo, por ejemplo, colones del 01/01/2000, dólares del 31/01/1999, unidades de fomento, etc.
3. En los proyectos que requieren inversión se debe considerar su valor residual, que es la estimación del precio al cual se pueden vender los bienes de capital al término de

su vida útil o al finalizar el proyecto. Este se debe consignar como un ingreso en el período correspondiente. El terreno tiene un valor residual igual a su valor nominal inicial, es decir, se recupera el 100%, salvo el caso en que el proyecto produzca una variación en el valor económico del mismo.

4. La vida útil de los bienes de capital son los años estimados de su potencial operación. Depende de sus especificaciones técnicas y la intensidad de su uso en el proyecto. Por ejemplo, una camioneta podría funcionar sin mayores problemas por 6 o más años, pero si, su costo de mantenimiento luego del tercer año es mayor que el de reposición, será preferible consignar una vida útil de 3 años. Para las construcciones se suelen considerar entre 20 y 30 años.
5. Si el proyecto dura más que la vida útil de alguno de los bienes de capital que requiere, será necesario hacer una inversión de reposición. Si la situación es la inversa, existirá un valor residual equivalente al tiempo de vida útil que le resta.
6. Los bienes de capital se imputan en la proporción en que van a ser utilizados en el proyecto. Por ejemplo, si en un proyecto de educación de adultos, se utiliza, a ciertas horas, una escuela básica existente, se considerará como costos de capital sólo el porcentaje en que se usa dicha infraestructura.

2.7 Principios económicos para la evaluación social de proyectos

2.7.1 Valor presente neto (VPN)

El Valor Presente Neto es el método más conocido a la hora de evaluar proyectos de inversión a largo plazo. El Valor Presente Neto permite determinar si una inversión cumple con el objetivo

básico financiero: MAXIMIZAR la inversión[7]. El Valor Presente Neto permite determinar si dicha inversión puede incrementar o reducir el valor de las Pymes (Valquiro, 2010). Ese cambio en el valor estimado puede ser positivo, negativo o continuar igual. Si es positivo significará que el valor de la firma tendrá un incremento equivalente al monto del Valor Presente Neto. Si es negativo quiere decir que la firma reducirá su riqueza en el valor que arroje el VPN. Si el resultado del VPN es cero, la empresa no modificará el monto de su valor. (Cohen, 2010)

Es importante tener en cuenta que el monto del Valor Presente Neto depende de las siguientes variables:

La inversión inicial previa, las inversiones durante la operación, los flujos netos de efectivo, la tasa de descuento y el número de periodos que dure el proyecto.

La inversión inicial previa: corresponde al monto o valor del desembolso que la empresa hará en el momento de contraer la inversión. En este monto se pueden encontrar: El valor de los activos fijos, la inversión diferida y el capital de trabajo.

Los activos fijos serán todos aquellos bienes tangibles necesarios para el proceso de transformación de materia prima (edificios, terrenos, maquinaria, equipos, etc.) o que pueden servir de apoyo al proceso. Estos activos fijos conforman la capacidad de inversión de la cual dependen la capacidad de producción y la capacidad de comercialización.

La inversión diferida es aquella que no entra en el proceso productivo y que es necesaria para poner a punto el proyecto: construcción, instalación y montaje de una planta, la papelería que se requiere en la elaboración del proyecto como tal, los gastos de organización, patentes y documentos legales necesarios para iniciar actividades son ejemplos de la inversión diferida.

El capital de trabajo es el monto de activos corrientes que se requiere para la operación del proyecto: el efectivo, las cuentas por cobrar, los inventarios se encuentran en este tipo de activos. Cabe recordar que las empresas deben tener niveles de activos corrientes necesarios tanto para

realizar sus transacciones normales, como también para tener la posibilidad de especular y prever situaciones futuras impredecibles que atenten en el normal desarrollo de sus operaciones. Los niveles ideales de activos corrientes serán aquellos que permita reducir al máximo posible los costos de oportunidad (costos por exceso + costos por insuficiencia + costos por administración).

Los activos fijos son bienes sujetos al desgaste por el uso o también por el paso del tiempo. La depreciación juega papel importante pues afecta positivamente a los flujos netos de efectivo por ser ésta deducible de impuestos lo que origina un ahorro fiscal. Importante recordar que los terrenos no son activos depreciables. Los activos nominales o diferidos por su parte también afectan al flujo neto de efectivo pues son inversiones susceptibles de amortizar, tarea que se ejecutará con base a las políticas internas de la compañía. Estas amortizaciones producirán un ahorro fiscal muy positivo para determinar el flujo neto de efectivo.

Las inversiones durante la operación: Son las inversiones en reemplazo de activos, las nuevas inversiones por ampliación e incrementos en capital de trabajo (Fontaine, La evaluación privada y social de proyectos, el rol del estado, 2007).

Los flujos netos de efectivo: es importante tener en cuenta la diferencia existente entre las utilidades contables y el flujo neto de efectivo. Las primeras es el resultado neto de una empresa tal y como se reporta en el estado de resultados; en otras palabras, es la utilidad sobre un capital invertido. El flujo neto de efectivo es la sumatoria entre las utilidades contables con la depreciación y la amortización de activos nominales, partidas que no generan movimiento alguno de efectivo y, que por lo tanto, significa un ahorro por la vía fiscal debido a que son deducibles para propósitos tributarios. Cuanto mayor sea la depreciación y mayor sea la amortización de activos nominales menor será la utilidad antes de impuestos y por consiguiente menor los impuestos a pagar.

Los flujos netos de efectivo son aquellos flujos de efectivo que el proyecto debe generar después de poner en marcha el proyecto, de ahí la importancia en realizar un pronóstico muy acertado con el fin de evitar errores en la toma de decisiones (Fontaine, Evaluación Social de Proyectos, 2010).

2.7.2 La tasa de descuento

La tasa de descuento es la tasa de retorno requerida sobre una inversión. La tasa de descuento refleja la oportunidad perdida de gastar o invertir en el presente por lo que también se le conoce como costo o tasa de oportunidad. Su operación consiste en aplicar en forma contraria el concepto de tasa compuesta. Es decir, si a futuro la tasa de interés compuesto capitaliza el monto de intereses de una inversión presente, la tasa de descuento revierte dicha operación. En otras palabras, esta tasa se encarga de descontar el monto capitalizado de intereses del total de ingresos percibidos en el futuro. Ejemplo:

Inversión: $1.000 Tasa de descuento periódica: 15% anual. Años a capitalizar: 2

Valor futuro al final del periodo 2 = 1.000 x (1.15)2 = 1.322,50

Valor presente de 1.322,50 a una tasa de descuento del 15% durante 2 años = 1.000

1.322,50 ÷ (1.15)2 = 1.000

En evaluación de proyectos un inversionista puede llegar a tener dificultad para determinar la tasa de descuento. Es este quizás el mayor problema que tiene el VPN. ¿La tasa de descuento puede ser el costo de capital de las utilidades retenidas? O, ¿Puede ser también el costo de emitir acciones comunes? y ¿Por qué no la tasa de deuda? Algunos expertos opinan que una de las mejores alternativas es aplicar la tasa promedio ponderada de capital, pues ella reúne todos los componentes de financiamiento del proyecto. Pero también el inversionista puede aplicar su

costo de oportunidad, es decir aquella tasa que podría ganar en caso de elegir otra alternativa de inversión con igual riesgo.

2.7.3 Tasa interna de retorno (TIR)

Conocida también como tasa interna de rendimiento, es un instrumento o medida usada como indicador al evaluar la eficacia de una inversión.

La TIR sirve para identificar claramente el tiempo en que recuperaremos el capital asignado a una inversión. Para su cálculo también se requiere proyectar los gastos por efectuar (valores negativos) e ingresos por recibir (valores positivos) que ocurren en períodos regulares (Hugo Navarro, 2006).

¿Cómo se calcula? Como se citó inicialmente es necesario establecer el monto de inversión, los flujos de ingreso y la inversión periódica (gastos) para cada uno de los períodos establecidos en el proyecto, a fin de considerar sólo los beneficios netos en cada periodo (utilidades brutas o utilidades antes de impuestos).

El monto de las utilidades calculadas nos ayudará a determinar el plazo en que recuperaremos la inversión inicial requerida para el proyecto, ejemplo:

Inversión Inicial: 100.000

	AÑO 1	AÑO 2	AÑO 3	AÑO 4	AÑO 5
COSTOS	$500,000	$500,000	$500,000	$500,000	$500,000
INGRESOS	$520,000	$520,000	$520,000	$520,000	$520,000

UTILIDAD	$20,000	$20,000	$20,000	$20,000	$20,000
INVERSIÓN INICIAL	$100,000	$100,000	$100,000	$100,000	$100,000
TIR	-80%	-60%	-40%	-20%	0%

Tabla 2.3 Ejemplo de cálculo TIR

Como se puede observar la inversión requerida o inicial, de acuerdo a la proyección de las utilidades anuales, podrá recuperarse en su totalidad al final del quinto año. Lo que podría interpretarse como una buena inversión si la relacionamos con la tasa de interés pagada por otros instrumentos financieros. Por lo que en éste ejemplo se puede interpretar como que la TIR es de 5 años.

La TIR, se puede calcular con los beneficios antes o después de impuestos.

Es recomendable aplicar esta facilidad para determinar la productividad de cualquier proyecto que nos propongan, pero también es aplicable a inversiones por realizar en herramientas financieras tradicionales. La TIR nos dará información adicional para tomar decisiones analizadas a profundidad cuando de inversiones se trata.

2.8 Tasa social de descuento

En economía se reconoce que existe una dicotomía entre consumo e inversión. El consumo es el uso directo de un bien económico para el disfrute actual, mientras que la inversión comporta dedicar una cierta cantidad de recursos a una actividad o proyecto que no proporcionará beneficios o satisfacción inmediata, sino que sólo pasado un tiempo generará recursos que

podrán ser usados directamente para la satisfacción de necesidades o deseos. Obviamente el consumo y la inversión se contraponen, y un problema de los individuos y las sociedades es cómo dividir los recursos actuales (tiempo, dinero, tierra, etc.) entre consumo presente e inversión que producirá beneficios futuros. La tasa de descuento tiene que ver con la equivalencia entre el consumo presente y los beneficios futuros. Una tasa de descuento grande implica una gran preferencia por el consumo presente y un menor interés en disfrutar de beneficios futuros. Ese análisis de tasa de descuento puede llevarse a cabo sobre las inversiones públicas en bienes de interés social.

La tasa social de descuento mide la tasa a la cual una sociedad está dispuesta a cambiar consumo presente por consumo futuro o, dicho de otra manera, el patrón de consumo ahorro de una sociedad en cada momento; lo cual no es otra cosa que el valor tiempo que le asigna la sociedad a la postergación. Esta es la razón por el cual toma relevancia la tasa social en la evaluación de proyectos del sector público, sobre todo cuando se están evaluando proyectos cuyos beneficios afectan a toda la sociedad, como es el caso de proyectos generadores de bienes públicos, y cuando los proyectos arrojan resultados ``que se extienden por muchos períodos y, por tanto, afectan a más de una generación.

2.9 Diseño teórico de análisis de proyectos

Las municipalidades gestionan, administran y ejecutan proyectos de diferentes naturalezas, desde proyectos que van desde introducción de agua potable, infraestructura vial, proyectos de salud, electrificación etc. Cada uno con características técnicas distintas. Sin embargo, es viable el abordarlos a todos ellos por medio de una metodología que integre una caracterización de los mismos, facilitando la priorización.

Luego de analizar exhaustivamente la teoría sobre la evaluación de proyectos, y de observar los procesos que se ejecutan en las municipalidades de la zona occidental, se propone la metodología multicriterio donde sus ejes o indicadores sean:

2.9.1 Tasa Interna de Retorno

Es el promedio de los rendimientos de una inversión, se expresa por medio de una tasa de interés (Ehrhardt, 2007), con la cual los ingresos futuros se equilibran con la inversión, este indicador es muy empleado y de fácil comprensión, cuando toma un valor positivo indica una rentabilidad del proyecto, cuando su valor es negativo refleja una perdida.

2.9.2 Valor Actual Neto

Permite calcular el valor presente de los flujos futuros de efectivo, originados por una inversión, este indicador descuenta los montos del flujo de caja futuros convirtiéndolos al presente, empleando una tasa de descuento para ello.

2.9.3 Tiempo de recuperación de la inversión

Se refiere al periodo de tiempo en el cual una inversión inicial llega a recuperarse, es un indicador que suele expresarse en años. Dada una inversión inicial, este indicador calcula el tiempo en el cual los flujos futuros llegan a equipararse con la inversión inicial.

2.9.4 Área Prioritaria

Una Municipalidad manejará un conjunto de áreas claves en la referencia de clasificación de proyectos, áreas que se han priorizado debido a una agenda de país o una agenda municipal, con el objeto de darle énfasis a un conjunto de necesidades que son trascendentales, un ejemplo de ello es un área que se priorice debido a que conforma un riesgo o un peligro latente para una comunidad de pobladores, y debido a ello dicha área se convierte en importante por sobre otras.

2.9.5 Promedio de inversión por beneficiario

Se refiere al promedio per cápita invertido, donde el objetivo de un proyecto es beneficiar a la mayor cantidad de pobladores. Se calcula dividiendo el monto de la inversión entre la cantidad de beneficiarios.

2.10 Metodología para la consolidación de evaluación de proyectos.

Para la evaluación de proyectos surge una dificultad inherente, en razón de la naturaleza diversa de los proyectos realizados por las Municipalidades, pues evaluar un proyecto de agua potable con un proyecto de informática tiene la complejidad referida a la naturaleza de cada uno de los proyectos.

De ahí que surge la necesidad de una metodología multicriterio, la cual emplee indicadores que sean comunes para cada uno de los proyectos, y que permita, evaluar los mismos bajo un contexto de una escala donde se clasifiquen cada proyecto independientemente de su naturaleza.

El proceso de evaluación de los proyectos municipales es el siguiente:

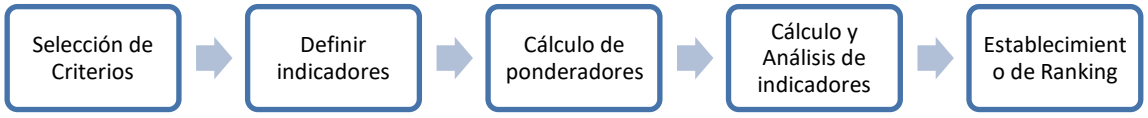


Figura 2: Proceso de evaluación de proyectos propuesto. **Fuente: creación propia**.

2.10.1 Selección de Criterios

La identificación de los criterios es una parte fundamental para la correcta evaluación de proyectos multicriterio. La selección de criterios dependerá del nivel jerárquico donde se quiera

implementar la iniciativa, ya sea que la prioridad de inversión sea política, desarrollo social, desarrollo económico o enfoque técnico.

2.10.2 Definir Indicadores

Una vez que se han identificado los criterios, se necesita definir cómo serán medidos. Para la construcción de los indicadores que serán empleados, se requiere que los objetivos sean desglosados en variables que puedan ser "medidas" y evaluadas. Se definen con claridad y con la mayor rigurosidad posible cuáles serán las variables que serán empleadas para medir los objetivos. A partir de las variables definidas, para los criterios, se construyen los indicadores que los representen adecuadamente, ya sean estos objetivos institucionales, programáticos o del proyecto.

2.10.3 Cálculo de ponderadores

A partir de los juicios expresados en las matrices de comparaciones y las fórmulas para los indicadores se calculan los ponderadores correspondientes a cada criterio.

2.10.4 Análisis de Indicadores

Los indicadores de las iniciativas poseen diferentes características según sea la variable que estén describiendo. La información cualitativa debe ser "cuantificada" para que sea aplicable a los métodos. Para esto debe construir tablas que permitan su homologación numérica. En estas se deben establecer los grados del atributo y su correspondiente numérico. Es importante que la escala aplicada, que indica la cota superior, inferior y los valores intermedios, sea igual para todos los indicadores.

La información cuantitativa de los indicadores proviene de diferentes medidas, y por lo tanto, de diferentes escalas que hacen imposible una comparación objetiva entre ellas. Se debe crear una escala con valores mínimo, máximo e intermedios tales que agrupen todo el rango de valores del indicador cuantitativo.

2.10.4 Establecimiento del Ranking

Se ordenan jerárquicamente los índices calculados para cada alternativa o proyecto de mayor a menor. Para que la municipalidad seleccione de una manera sencilla el proyecto o los proyectos que tienen mejor puntuación.

Un ejemplo del ranking sería:

Tabla 2.4: Ejemplo de Ranking de proyectos.

Ranking	Proyecto	Puntuación
1	Creación de un sistema automatizado para el registro tributario Municipal.	49
2	Electrificación de la lotificación Santa Gertrudis.	46
3	Construcción de la escuela de educación básica en la colonia El Calvario	42
4	Pavimentación de la calle principal de la Colonia el Pedregal	38
5	Construcción de la cancha de basquetball en la colonia El Calvario	32

Fuente: Ing. Quevedo, Ing. Calderon, E (2014). Diseño de un sistema de Evaluacion de proyectos de inversion, impulsados por las Alcaldias Municipales del departamento de Santa Ana, en la zona oocidental de El Salvador (Tesis de posgrado). Universidad de El Salvador, Santa Ana, El Salvador, Página 52

La municipalidad contará con un ranking de proyectos, priorizados en razón de su mejor evaluación, y con una fácil interpretación, en este caso con estrellas, facilitando la lectura del

ranking de proyectos. De modo que se ejecuten los proyectos en el orden de la clasificación realizada.

Los indicadores que son la base del ranking son los siguientes:

- Rentabilidad (con enfoque social).
- Valor Neto de Inversión.
- Tiempo de retorno.
- Beneficiarios.
- Área Prioritaria.

Los primeros 4 indicadores son cuantitativos y tienen una definición matemática formal además de requerir de insumos para su cálculo, el indicador de área prioritaria es cualitativo, pero que recibirá una adaptación para cuantificar en igual ponderación con el resto de los indicadores. Cada indicador tendrá la misma escala, y brindará una serie de puntos para el proyecto evaluado, la suma de los puntos de cada indicador generará el puntaje final, mismo que se refleja en forma de estrellas para su fácil interpretación:

Tabla 2.5: Ejemplo de Indicadores y Puntaje.

Indicador	Escala	Puntaje Ejemplo
Rentabilidad(con enfoque social)	0 – 10	7.5
Valor Neto Inversión	0 – 10	3.2
Tiempo de retorno	0 – 10	5.7
Beneficiarios	0 – 10	8
Área Prioritaria	0 – 10	7.5
	Total	31.9

Fuente: Ing. Quevedo, Ing. Calderon, E (2014). Diseño de un sistema de Evaluacion de proyectos de inversion, impulsados por las Alcaldias Municipales del departamento de Santa Ana, en la zona oocidental de El Salvador (Tesis de posgrado). Universidad de El Salvador, Santa Ana, El Salvador, Página 53.

Para los indicadores financieros se requiere de la cuantificación de una serie de elementos vitales para su cálculo: Inversión inicial, beneficios, costo, tasa de interés entre otros.

La inversión inicial se expresa a nivel municipal en la realización de la carpeta técnica, documento que brinda el diseño del proyecto y cuantifica todos los elementos de la inversión inicial, dicha carpeta técnica muestra el monto total que cuesta la inversión inicial de dicho proyecto.

La tasa de interés refleja el costo de capital, y es de vital importancia para el cálculo de los indicadores. La cuantificación de los beneficios y los costos durante toda la vida útil del proyecto es quien define la rentabilidad del mismo, por tal razón los beneficios sugeridos agrupan tanto el enfoque privado del proyecto, como también la parte social del mismo, incluyendo los costos y beneficios indirectos y sociales de la ejecución del proyecto. Los costos y beneficios tienen diferentes fuentes dependiendo de la naturaleza del proyecto.

Tabla 2.6: Recopilación Financiera del Proyecto (información de cabecera)

Nombre del proyecto:	Proyecto ABC
Monto de la Inversión:	$xxx,xx.xx
Vida Útil del Proyecto:	5 años
Costos anuales:	$x,xxx.xx
Beneficios anuales:	$x,xxx.xx

Antes de entrar al detalle y desglose de las fórmulas matemáticas para la metodología se brindará un desglose de los costos y beneficios a tomar en cuenta según la naturaleza de cada proyecto, ello con el objetivo que en los mismos se aplique un enfoque social, donde no solo se incluyan los beneficios privados, sino que también se incluyan los beneficios sociales.

2.11 Beneficios y Costos según tipo de proyecto a ejecutar

A continuación, se citan los costos y los beneficios identificables para los tipos de proyectos más comunes, sin embargo el principio para la evaluación de proyectos es el mismo, cuantificar los costos y los beneficios, ya sea que los mismos sean directos o indirectos, privados o sociales. Se detallan los siguientes tipos de proyectos:

- Proyectos de Agua Potable.
- Proyectos de Evacuación y Drenaje de Aguas lluvias.
- Proyectos de Alumbrado Público.
- Proyectos de Atención en Salud Primaria.
- Proyectos de Edificación Pública.
- Proyectos de Vialidad.
- Proyectos de Educación.
- Proyectos de Electrificación.
- Proyectos de Residuos Sólidos.
- Proyectos de Seguridad Policial.

2.11.1 Beneficios y costo de proyectos de agua potable

Tabla 2.7: Beneficios y Costos Proyectos de Agua Potable

COSTOS	
Costos de Operación.	Se registran a lo largo de la vida útil del proyecto y están asociados al buen funcionamiento del mismo, por ejemplo: ● Energía eléctrica. ● Agua cruda. ● Productos químicos. ● Mantenimiento de equipo. ● Mano de obra.
BENEFICIOS	

Beneficios por mayor consumo.	Originados por el consumo de agua potable, se incluyen: ● Ingresos por consumo de agua. ● Ahorro en aseo personal. ● Ahorro en actividades domésticas. ● Oportunidad de realizar actividades productivas.
Liberación de recursos.	Son beneficios originados al comparar la situación sin proyecto, pues el agua es un recurso vital que implica esfuerzos de los pobladores por su aprovisionamiento: ● Ahorro en tiempo de obtención de agua. ● Ahorro en mejora de la salud.

Fuente: (Mideplan, 2013)

2.11.2 Beneficios y costos de proyectos de evacuación y drenaje de aguas lluvias

Tabla 2.8: Beneficios y Costos Proyectos de Evacuación y Drenaje de Aguas lluvias

COSTOS	
Costos de Operación.	Se registran a lo largo de la vida útil del proyecto y están asociados al buen funcionamiento del mismo, por ejemplo: ● Mantenimiento y limpieza de infraestructura. ● Mano de obra.
BENEFICIOS	
Beneficio por menor daño en propiedades residenciales.	El beneficio por menor daño en propiedades residenciales será equivalente al cambio en el precio de la vivienda al mejorar la condición de evacuación y drenaje de aguas lluvias.
Beneficio por recuperación de terrenos baldíos anegadizos.	El beneficio de las propiedades que no cuentan con construcción que se ven mejorados por el proyecto de aguas lluvias.
Beneficio por menor daño de propiedades comerciales o industriales.	Son los beneficios originados para los comercios o industrias, las cuales con el proyecto ejecutado ahorraran en pérdidas ocasionadas por las inundaciones.
Beneficio por menor daño de propiedades del estado o públicas.	Desde el punto de vista inmobiliario las propiedades de organismos públicos se asemejan a las propiedades comerciales, en cuanto a su valoración e impacto por inundaciones.
Beneficio por menor deterioro de infraestructura vial.	Este se compone de la diferencia entre los costos anuales de reposición de la infraestructura vial deteriorada con proyecto y sin proyecto. $CD_{sin\ proyecto} - CD_{con\ proyecto}$

Beneficios por reducción de los costos de viaje.	Se calcula la diferencia de los costos de viaje sin proyecto y los costos de viaje con el proyecto. Tomando en cuenta las afectaciones en costos de viaje originadas por inundaciones en la infraestructura vial.
Beneficios por menor ausentismo laboral.	Beneficio originado por el ahorro en pérdidas debidas a la suspensión de actividades productivas asociadas a una inundación de un comercio, una industria o una institución pública.
Beneficios por reducción de enfermedades.	Este beneficio puede ser difícil de calcular en muchos casos, por lo que puede indicarse como un beneficio intangible.

Fuente: (Mideplan, 2013).

2.11.3 Beneficios y costos de proyectos de alumbrado pubico

Tabla 2.9: Beneficios y Costos Proyectos de Alumbrado Público

COSTOS	
Costos de Operación.	Se registran a lo largo de la vida útil del proyecto y están asociados al buen funcionamiento del mismo, por ejemplo: ● Mantenimiento y limpieza del sistema de alumbrado. ● Mano de obra. ● Gasto de energía eléctrica. ● Costo de reemplazo.
BENEFICIOS	
Disminución de costos de operación y mantenimiento.	Acorde al avance tecnológico, se produce un ahorro de operación y de mantenimiento en función que los componentes del sistema de alumbrado tienen mayor vida útil, y menor consumo de energía eléctrica.
Disminución de gases de efecto invernadero.	Hay un impacto asociado en los gases de efecto invernadero asociados a la situación sin proyecto, dado que puede emplearse quema de combustibles para proveer de iluminación.
Beneficio por disminución de contaminación lumínica.	Este beneficio puede considerarse intangible si se dificulta su cálculo.
Ahorro por disposición de lámparas contaminantes.	Muchos sistemas de alumbrado público antiguas estaban conformados por lámparas de mercurio y/o sodio, las cuales requieren un proceso adecuado de disposición final debido a sus contaminantes.

Fuente: (Mideplan, 2013).

2.11.4 Beneficios y costos de proyectos de atención en salud primaria

Tabla 2.10: Beneficios y Costos de Proyectos de Atención en Salud Primaria

COSTOS	
Costos de Operación.	Son los costos fijos del establecimiento: ● Remuneraciones del personal. ● Viáticos. ● Farmacia. ● Materiales e insumos. ● Gastos de servicios básicos.
Costos de Mantención.	Considera los gastos de mantención de la infraestructura y de los equipos.
BENEFICIOS	
Ahorro por pronta y oportuna atención médica.	Entre más pronto sea atendida una enfermedad, requiere de menor atención médica, menor medicamente empleado y de menor tiempo de recuperación del paciente.
Ahorro en reducción de tiempos de espera y atención.	Un paciente para acudir a un centro asistencial de salud suspende su actividad productiva, dejando su trabajo y gastando ese tiempo en la atención médica.
Ahorro en costos de traslado.	Al acercar la atención médica a una población, se reducen los costos de traslado, que en muchos casos implica traslados en animales de carga como caballos, o camionetas.
Beneficio en la mejora de los indicadores de salud.	Beneficios vinculados a la mejora en indicadores como la tasa de mortalidad infantil, número de controles por cada mil habitantes, etc.

Fuente: (Mideplan, 2013).

2.11.5 Beneficios y costos de proyectos de edificación publica

Tabla 2.11: Beneficios y Costos de Proyectos de Edificación Pública

COSTOS	
Costos de Operación.	Son los costos fijos del establecimiento: ● Remuneración de personal. ● Gastos de servicios básicos.
Costos de Mantención.	Considera los gastos de mantención de la infraestructura y de los equipos.

BENEFICIOS	
Ahorro en Costos de Operación.	Corresponde a ahorros en gastos por concepto de remuneraciones de personal, servicios básicos (agua, energía eléctrica, gas, combustible), etc.
Ahorro en costos de reparación.	La reparación de edificios contempla los trabajos destinados a subsanar el deterioro sufrido en un inmueble en forma ocasional o por falta de conservación y que se traduce en la reposición de elementos fundamentales dañados; es decir, aquellos que afectan a la estructura física, a sus instalaciones y/o modifican substancialmente la concepción arquitectónica artística del edificio y que por lo tanto requieren de una asistencia técnica especializada del nivel profesional.
Ahorro en costos de remodelación.	La remodelación de edificios contempla los trabajos destinados a reorganizar el espacio interior de un edificio y/o adecuar sus instalaciones.
Mejoramiento de la eficiencia del personal.	Aumento de productividad del personal que labora en la institución: El aumento de productividad proviene de un ahorro de tiempo en el desplazamiento de los funcionarios. Este beneficio se percibe más claramente en aquellos proyectos que persiguen concentrar, en un solo edificio, dependencias que funcionan en localizaciones separadas.
Ahorro por liberación de activos.	Si la institución es propietaria de uno o más inmuebles y éstos quedaran liberados a causa del proyecto, deberá considerarse como beneficio el valor de su venta, siempre y cuando éste tenga un uso alternativo. Por otra parte, si la institución arrienda un inmueble y por causa del proyecto deja de hacerlo, el ahorro del costo de arriendo deberá considerarse como beneficio del proyecto.

Fuente: (Mideplan, 2013).

2.11.6 Beneficios y costos de proyectos de vialidad

Tabla 2.12: Beneficios y Costos de Proyectos de Vialidad

COSTOS	
Costos de Mantención.	Considera los gastos de mantención de la infraestructura vial como el mantenimiento de baches, y la limpieza de la vía.
BENEFICIOS	

Ahorro en tiempo de viaje.	Al contar con caminos más directos o más amplios, los usuarios demoran menos en desplazarse de un lugar a otro. El cálculo del ahorro de tiempo, medido en horas, se realiza obteniendo la diferencia entre el tiempo de viaje con y sin proyecto.
Ahorro en costos de operación vehicular.	Al mejorar la calidad de la carpeta de rodadura de la vía, se producen una mejora en el rendimiento de lubricantes, aumenta la vida útil de los neumáticos y del vehículo mismo.
Ahorro de combustible.	Debido a la mejora de las condiciones de la calle, habrá un ahorro de combustible en razón de la velocidad de desplazamiento de los automotores.
Ahorro en costos de mantenimiento.	Esto se produce cuando en la situación sin proyecto se requieren constantes y recurrentes reparaciones para poder continuar con la vía operativa. Para su estimación y valoración, deberá estimarse los costos de mantenimiento de las situaciones sin y con proyecto.

Fuente: (Mideplan, 2013).

2.11.7 Beneficios y costos de proyectos de educacion

Tabla 2.13: Beneficios y Costos de Proyectos de Educación

COSTOS	
Costos de Mantención.	Considera los gastos de mantención de la infraestructura educativa.
Costos de Transporte.	Los beneficiarios del proyecto recurren a gastos para desplazarse desde sus hogares hasta su centro educativo.
BENEFICIOS	
Aumento en el nivel de productividad de los beneficiarios.	Al mejorar el nivel educativo las oportunidades de trabajo crecen, y por ende una mejora en el nivel de ingresos de las familias.
Disminución de conductas antisociales.	Debido a que los jóvenes logran adquirir nuevas oportunidades, se reduce la probabilidad de buscar la alternativa delincuencial.
Ahorro en costos de transporte.	Muchos de los jóvenes requieren de esfuerzos mayores para poder optar a educación, viéndose obligados a recorrer grandes distancias en el escenario sin proyecto para cubrir su necesidad educativa.

Fuente: (Mideplan, 2013).

2.11.8 Beneficios y costos de proyectos de elictrificacion

Tabla 2.14: Beneficios y Costos. Proyectos de Electrificación

COSTOS	
Costos de Operación.	Son los costos fijos del establecimiento: ● Remuneración de personal. ● Gastos de servicios básicos.
Costos de Mantención.	Considera los gastos de mantención de la infraestructura y de los equipos.
BENEFICIOS	
Ingresos por ventas y tasas.	Tanto privados como el estado percibe beneficios a raíz de la venta del servicio de energía eléctrica, así como del gravamen a la infraestructura del mismo.
Ahorro en recursos alternos.	En el escenario sin proyecto, las familias recurren al uso de alternativas de iluminación, como la quema de combustibles.
Oportunidades de productividad.	El servicio de energía eléctrica brinda un abanico de oportunidades para el crecimiento económico como por ejemplo el poder poner un taller de mecánica, un molino, etc.

Fuente: (Mideplan, 2013).

2.11.9 Beneficios y costos de proyectos de residuos solidos

Tabla 2.15: Beneficios y Costos. Proyectos de Residuos Solidos

COSTOS	
Costos de Operación.	Son los costos fijos del establecimiento: ● Remuneración de personal. ● Gastos de servicios básicos.
Costos de Mantención.	Considera los gastos de mantención de la infraestructura y de los equipos.
Costos de transporte.	Se refiere a los costos implicados en recoger y transportar los desechos sólidos hasta el lugar de su adecuado tratamiento.
BENEFICIOS	
Ingresos por venta de servicio.	Se produce un ingreso por el cobro de tonelada de disposición final de desechos. En algunos casos también se produce una venta a raíz de los desechos transformados.

Ahorro en recursos alternos.	En el escenario sin proyecto, las familias recurren al uso de alternativas como el entierro de desechos o la quema de los mismos.

Fuente: (Mideplan, 2013).

2.11.10 Beneficios y costos de proyectos de seguridad policial

Tabla 2.16: Beneficios y Costos de Proyectos de Seguridad Policial

COSTOS	
Costos de Operación.	Son los costos fijos del establecimiento: ● Remuneración de personal. ● Gastos de servicios básicos.
Costos de Mantención.	Considera los gastos de mantención de la infraestructura y de los equipos.
BENEFICIOS	
Ahorro por reducción de delincuencia.	En términos de Economía, se puede aducir que los mayores niveles de seguridad ciudadana que se logran con niveles adicionales de vigilancia policial constituyen bienes de consumo, en tanto las personas se gratifican por el hecho objetivo de desenvolverse en un medio ambiente más seguro, al igual que por percibir un entorno de mayor seguridad. Puede cuantificarse por medio de los costos implicados por delitos.
Mejora en el ambiente de inversión.	Condiciones de mayor seguridad generan ambientes favorables al desarrollo de actividades productivas.

Fuente: (Mideplan, 2013).

2.11.11 Otros tipos de proyectos

Independientemente del tipo de proyecto que se ejecute, los elementos a cuantificar son los mismos: Inversión Inicial, Costos, Beneficios, etc. Por ello puede generalizarse para cualquier tipo de proyecto el cálculo de los mismos. La importancia radica en se realice una aproximación lo más cercana posible, para que la evaluación se acerque a la realidad, sin dejar de lado ningún costo o beneficio.

Tabla 2.17: Beneficios y Costos de Otros Proyectos

COSTOS	
Costos Privados.	Aquellos que implican una erogación de fondos para por parte del ejecutor o beneficiario privado del proyecto.
Costos Sociales.	Aquellos costos que están asociados a las comunidades o personas alrededor del proyecto.
BENEFICIOS	
Beneficios Privados.	Son los ingresos que se producirán a partir de la ejecución del proyecto, los que serán captados por la organización ejecutora del mismo.
Beneficios Sociales.	Son beneficios recibidos por las comunidades o personas que interactúan con el proyecto.

Fuente: Ing. Quevedo, Ing. Calderon, E (2014). Diseño de un sistema de Evaluacion de proyectos de inversion, impulsados por las Alcaldias Municipales del departamento de Santa Ana, en la zona oocidental de El Salvador (Tesis de posgrado). Universidad de El Salvador, Santa Ana, El Salvador, Página 63.

2.12 Definicion de indicadores

Una vez conocida la lógica de evaluación describiremos la parte que se encargara de dar un valor a las evaluaciones de los proyectos, los indicadores se detallan a continuación:

2.12.1 Rentabilidad con enfoque social

Este indicador es calculado por medio de la Tasa Interna de Retorno (TIR) y por definición es la tasa de interés, con la cual el valor actual neto se iguala a cero. Su fórmula es la siguiente:

$$VAN = \sum_{t=1}^{n} \frac{F_t}{(1+TIR)^t} - I = 0$$

Para el cálculo de la TIR los beneficios y costos de los flujos deben tomar en cuenta tanto los flujos Privados como Sociales.

Una vez calculada la TIR se le asigna el puntaje en razón de los siguientes valores:

Tabla 2.18: Puntaje asignado a TIR

TIR	PUNTAJE
Menor o igual a 1%	0
1.1% a 3%	1
3.01% a 5%	2
5.01% a 7%	3
7.01% a 9%	4
9.01% a 11%	5
11.01 a 13%	6
13.01% a 15%	7
15.01 a 17%	8
17.01 a 19%	9
Mayor de 19%	10

Fuente: Ing. Quevedo, Ing. Calderon, E (2014). Diseño de un sistema de Evaluacion de proyectos de inversion, impulsados por las Alcaldias Municipales del departamento de Santa Ana, en la zona oocidental de El Salvador (Tesis de posgrado). Universidad de El Salvador, Santa Ana, El Salvador, Página 65

2.12.2 Valor Neto de la Inversión

Se calcula por medio del Valor Actual Neto (VAN). Trae al presente todos los flujos futuros, su fórmula es la siguiente:

$$\text{VAN} = \sum_{t=1}^{n} \frac{V_t}{(1+k)^t} - I_0$$

V_t representa los flujos de caja en cada periodo t.

I_0 es el valor del desembolso inicial de la inversión.

n es el número de períodos considerado.

k, ***d*** o ***TIR*** es el tipo de interés.

Para dicha operación se va a tomar una tasa de descuento igual a la inflación de El Salvador en el presente año, para 2013 fue de un 1% y ese comportamiento se espera para el año 2014 (Trading Economics, 2014).

Para asignar el puntaje en razón del cálculo del Valor Actual Neto se calculará la división entre el VAN del proyecto dividido por la inversión inicial (con signo positivo) multiplicado por 100, de modo que se expresa el VAN como un porcentaje de la inversión inicial y dependiendo de ese valor se asignará el puntaje.

Tabla 2.19: Puntaje Asignado al VAN

(VAN / InversionInicial) x 100	PUNTAJE
Menor o igual a 1	0
1.1 a 10	1
10.1 a 20	2
20.1 a 30	3
30.1 a 50	4
50.1 a 70	5
70.1 a 90	6
90.1 a 110	7
110.1 a 130	8
130.1 a 150	9
Mayor a 150	10

Fuente: Ing. Quevedo, Ing. Calderon, E (2014). Diseño de un sistema de Evaluacion de proyectos de inversion, impulsados por las Alcaldias Municipales del departamento de Santa Ana, en la zona oocidental de El Salvador (Tesis de posgrado). Universidad de El Salvador, Santa Ana, El Salvador, Página 66

2.12.3 Tiempo de retorno de la inversión

Es el tiempo que tarda una inversión en recuperar sus costos. El resultado del cálculo es una cantidad de tiempo, expresada en años, meses y/o días. Dicha cantidad de tiempo no es estándar para cada proyecto pues los proyectos tienen naturaleza distinta, para realizar una estandarización del indicador la cual exprese este indicador de forma equitativa para cualquier

tipo de proyecto, el tiempo de recuperación de la inversión se expresará de forma porcentual en comparación con el tiempo de vida útil del proyecto.

Fórmula= (1 – (Periodo de Recuperación de Inversión / VidaUtilProyecto)) X 10

De este modo se expresa en una escala de 0-10

2.12.4 Beneficiarios

Por principio de inversión, entre más beneficiarios tenga un proyecto es mejor para el desarrollo municipal, por ejemplo una inversión de $1,000 para 100 beneficiarios dará un monto de $10 por beneficiario, si la misma inversión alcanza a 200 beneficiarios, se tendría $5 por beneficiario, donde cuando menor es el promedio de inversión por beneficiario mejor es el indicador.

Para poder asignar un valor o puntaje en este indicador se requiere de 2 o más proyectos, pues la cantidad de puntos dependerá de la escala que fluctúa en razón de los proyectos que se ejecuten. Para puntuarlo se tomara el valor máximo y valor mínimo del promedio de inversión por beneficiario donde el máximo tendrá el puntaje de 10 y el mínimo el puntaje de 0.

La fórmula para el puntaje es la siguiente:

$$\text{Puntaje} = 1 - \left(\frac{\text{PromInvBenef-MIN(PromInvBenef)}}{\text{MAX(PromInvBenef)-MIN(PromInvBenef)}} \right) \text{ x } 10$$

PromInvBenefi= Promedio de Inversión por Beneficiario del proyecto Actual

MIN(PromInvBenef)= Mínimo de todos los Promedios de Inversión por Beneficiario

MAX(PromInvBenef)= Máximo de todos los Promedios de Inversión por Beneficiario

Por ejemplo:

Proyectos	PromInvBe	Puntaje
● Proyecto A	$ 5.00	10
● Proyecto B	$ 7.50	6.42
● Proyecto C	$12.00	0

2.12.5 Área Prioritaria

Se realizará una clasificación para cada proyecto, asignando un área para el mismo, aquellos proyectos que se determinen como claves en razón de su área prioritaria, los mismos serán puntuados más alto. Las áreas y sus puntajes son los siguientes:

Tabla 2.20: Puntaje Asignado por Area Prioritaria

Área	**Puntaje**
Medio Ambiente y Recursos Naturales	10
Generación Energética	10
Seguridad Alimentaria y Nutricional	9
Salud	8
Educación	7
Mitigación de Riesgos	7
Energía Eléctrica	6
Proyectos Tecnológicos	6
Ordenamiento territorial	5
Protección al patrimonio cultural	5
Construcción	4
Ordenamiento urbano	4
Deportes	4
Otros proyectos	3

Fuente: Ing. Quevedo, Ing. Calderon, E (2014). Diseño de un sistema de Evaluacion de proyectos de inversion, impulsados por las Alcaldias Municipales del departamento de Santa Ana, en la zona oocidental de El Salvador (Tesis de posgrado). Universidad de El Salvador, Santa Ana, El Salvador, Página 68.

Estas áreas y sus puntajes pueden ser modificados por una municipalidad, lo importante es que se establezcan áreas prioritarias de modo que permitan clasificar cada proyecto en una de dichas áreas.

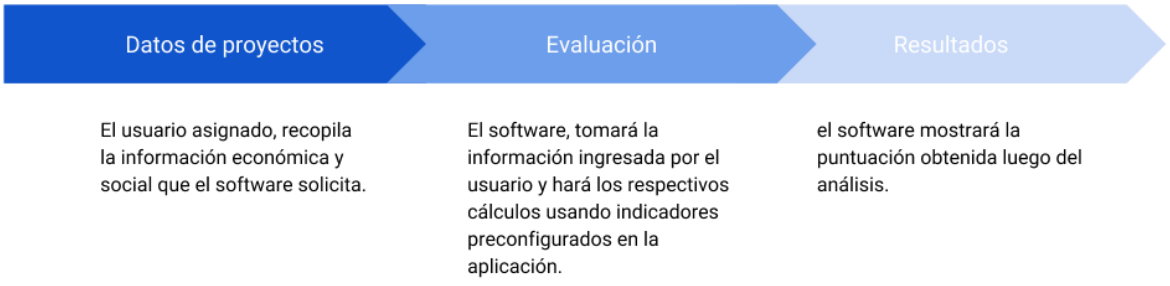
CAPITULO III: DISEÑO DEL SISTEMA DE EVALUACIÓN SOCIAL DE PROYECTOS

Este capítulo describe todo lo necesario para poder llevar a cabo una solución al tema planteado en los capítulos anteriores. Se dará una muestra iniciando por dar forma a la idea lógica del desarrollo, es decir, se llevará el problema de evaluación de proyectos a lenguaje informático, en pocas palabras las bases del sistema desde el punto de vista de ingeniería de sistemas.

3.1 Perspectiva del Software

Este software es una solución independiente de otros a los cuales una institución o entidad pueda apoyarse, que como objetivo busca el reemplazo de las tomas de decisiones basadas en métodos empíricos o que buscan el beneficio de un sector en específico. Como se ha descrito en el capítulo 2, se busca automatizar y unificar la manera en la que se toman las decisiones, con el apoyo de este, se tendrá una recomendación, una referencia como también, un parámetro altamente veraz, conciso y confiable. Basado en una interfaz que facilite el uso del usuario, implementado en una plataforma web, apoyándose en tecnologías flexibles que garantizaran la calidad del producto. Un esquema resumido de lo que el software hará, se muestra en la figura:

Figura 3.1 Resumen de perspectiva de software



3.2 Función del producto

La función general del software es el proveer al usuario de un análisis de proyectos con rentabilidad social en base a un multicriterio previamente estudiado, en donde, conceda luego de ingresar la información económica y social de un proyecto, devuelva un puntaje, con el cual de un apoyo a la toma de decisiones.

El siguiente diagrama mostrará un flujo de usuario de la lógica general de la aplicación:

Tabla 3.1 Descripción de Procesos generales de sistema

Descripción	Proceso
Esta actividad dará la oportunidad de acceder al home de la aplicación, en la cual se ofrecerá una interfaz personalizada de acuerdo con el usuario que se ingresado.	Inicio de Sesión
El software considera el uso de indicadores agrupados en "reglas". Aunque ya existe una configuración inicial de esta, el usuario puede editarlas de acuerdo con su utilidad.	Administrar indicadores
Luego de tener el acceso correcto a nuestra aplicación, podremos crear proyectos, los cuales será necesario completar la información requerida por el sistema, por ejemplo: inversiones iniciales, costos, beneficios, beneficiarios, etc.	Crear Proyectos
Se crea este proceso el cual ayudará al usuario a tener sus proyectos en un solo contenedor y así mantener un orden y una mejor visibilidad de sus proyectos y puntuaciones.	Crear Carpetas
El asignar proyectos a carpetas, ayudará a que las reglas sean aplicadas en conjunto a diferentes proyectos.	Asignar proyectos a carpetas
El núcleo de la aplicación se encuentra en este proceso, el cual permitirá utilizar reglas preconfiguradas para el análisis multicriterio (que se han descrito en el Cap II) y dar una puntuación de 0 a 50, siendo 50 el proyecto mejor calificado.	Análisis de priorización de proyectos

Los reportes tendrán como objetivo dar un panorama más amplio a una puntuación obtenida, un resumen de las áreas puntuadas para un análisis más objetivo.	Vista de reporte de puntuación

3.3 Clases de Usuarios y Características

El software considerará 3 diferentes tipos de usuarios, los cuales serán descritos en la siguiente tabla:

Tabla 3.2 Descripción de Usuarios

USUARIO	DESCRIPCIÓN	ROLES	PERMISOS
ROOT	Este usuario es único, y tiene acceso a todas las funcionalidades del sistema, además de poder crear usuarios con roles más limitados.	Administrador General (Usuarios y gestión general del sistema)	Ver Crear Actualizar Eliminar
ADMIN	Este usuario tiene autorización para configurar la entidad de la institución en la aplicación.	Administrador Local (Usuarios y gestión general de alcaldía)	Ver Crear Actualizar Eliminar
USER	Este usuario tendrá uso limitado al sistema, su fin es el administrar la información requerida para que el software haga los análisis correspondientes, para eliminar información, será necesario que se comunique con un usuario ADMIN	Usuario Final (Gestión de proyectos)	Ver Crear Actualizar

3.4 Entorno Operativo

El sistema de priorización de proyectos deberá estar capacitado para ser desplegado ya sea en un servidor local o un servidor en la nube. Dicho servidor deberá contar con laS siguientes características de hardware:

Arquitectura de 64 bits

4 o más núcleos de CPU’s disponibles

16 GB de RAM mínimo, 32 GB de RAM recomendado

25 GB de espacio de disco disponible mínimo.

Puertos habilitados para TCP.

El acceso al sistema solamente necesitará el acceso desde cualquier dispositivo que disponga de un navegador web con las versiones actualizadas de los mismos.

3.5 Restricciones de diseño e implementación

El sistema se limitará a usar las siguientes tecnologías:

Tabla 3.3 Descripción de tecnologías por componente

Componente de software	Tecnología
Gestor de base de datos	MongoDB
Framework Frontend	Angular Material
Framework BackEnd	NodeJs
Framework de servidor	ExpressJs

3.6 Lineamientos generales de interfaces externas

Para la elaboración de la interfaz se necesitará contar con algunos elementos comunes basados en Angular Material para las diferentes interfaces, estos elementos deberán aplicarse de manera general al software con el objetivo de estandarizar las vistas:

3.6.1 Fuentes y colores:

La tipografia utilizada en este software será Roboto/Sans Serif o en su defecto la que el navegador donde se utilice el software maneje como estándar.

Los colores utilizados para los títulos (Azul), títulos de tablas (Gris) y texto en general (Negro).

3.6.2 Botones:

Se utilizarán los componentes <button>:

3.6.3 Notificaciones:

Las notificaciones al usuario serán mostradas a través del componente **snack-bar** de la siguiente manera:

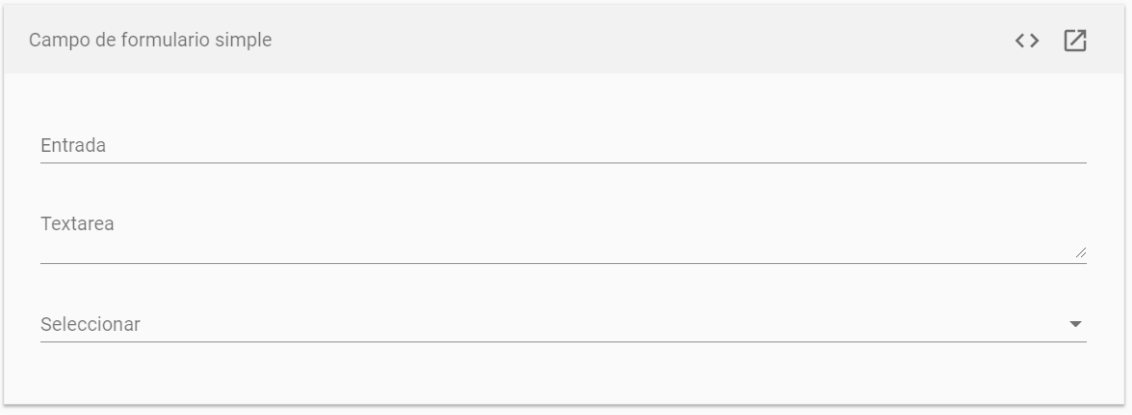
Éxito
SnackBar color celeste confirmando que la acción solicitada ha sido realizada.

Advertencia
SnackBar color celeste confirmando que la acción se ha advertido por alguna acción de información al usuario.

Error
SnackBar color gris informando el error como comentario.

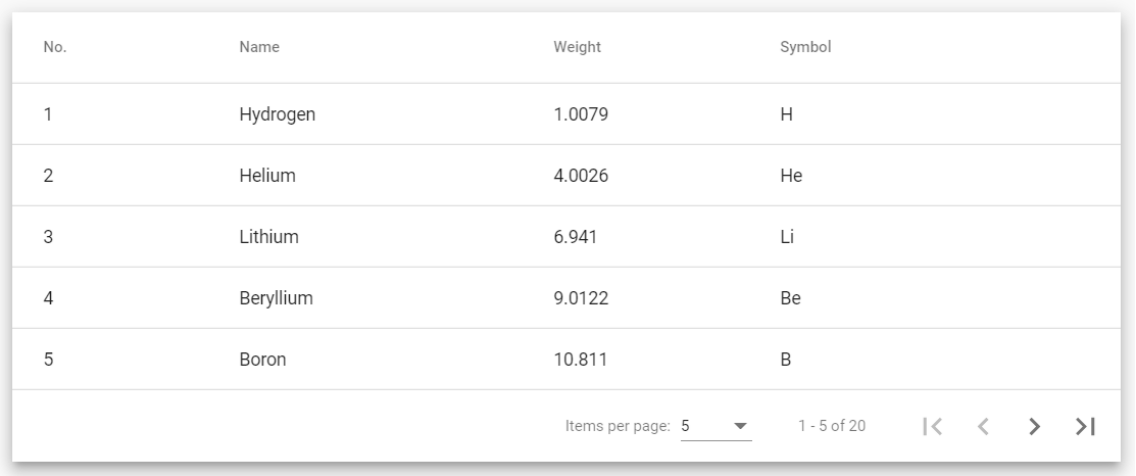
3.6.4 Formularios de ingreso de datos:

Para el ingreso de información y relleno de formularios se utilizará el elemento **<mat-form-field>** que permite agrupar dentro de él elementos como **<input>, <textarea>, <mat-select>,** etc.



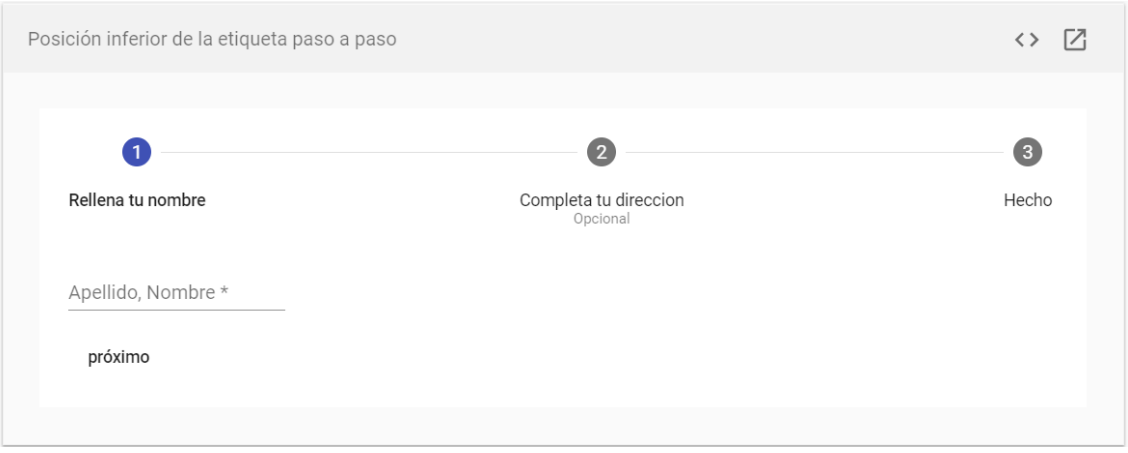
3.6.5 Listas o tablas:

El ordenamiento de información en lista se presentará a través del componente **<table mat-table>.** La siguiente imagen muestra un ejemplo de dicho componente:



3.6.6 Paso a paso

Algunos registros serán agregados siguiendo un flujo de trabajo que permitirá al usuario ingresar información ordenada y agrupada lógicamente. Esto a través del componente **`<mat-horizontal-stepper>`**. La siguiente imagen muestra un ejemplo de este componente:



3.6.7 Spinner de progreso

Algunos procesos necesitaran indicar al usuario que el sistema está realizando una labor o ejecutando un script, para eso utilizaremos el componente **`<mat-progress-spinner>`** en modalidad indeterminada. La siguiente imagen muestra un ejemplo del componente:

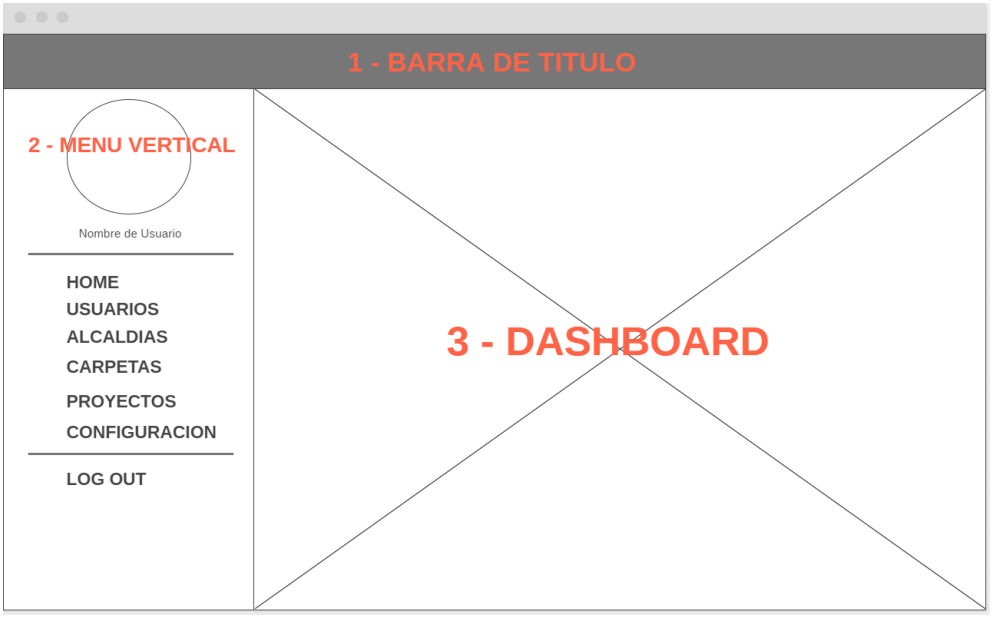
3.7 Interfaces de usuario

Se describen a continuación los requerimientos para cada interfaz.

<table>
<tr><td>Interfaz 1</td><td>INICIO DE SESIÓN</td></tr>
<tr><td>Descripción</td><td>Se requiere que el software inicie con la interfaz que muestre el acceso a los usuarios registrados en la base de datos para lo cual es necesario contar con los siguientes elementos:
1. Un campo de ingreso de usuario.
2. Un campo de ingreso de contraseña.
3. Un botón que ejecute la acción de validación e ingrese al software.
4. Un mensaje que indique si el usuario ha ingresado las credenciales correctamente.</td></tr>
<tr><td colspan="2">Diseño del componente</td></tr>
<tr><td colspan="2"> 
</td></tr>
</table>

Interfaz 2	HOME
Descripción	Se requiere para esta interfaz los siguientes elementos: 1. Una barra de título que contenga: a. El nombre de la aplicación. b. Boton de Log out. c. Un botón para ocultar el menú. 2. Un menú vertical al lado izquierdo con los ítems: a. Foto y nombre de usuario. b. Inicio: Que regrese al home de la aplicación. c. Usuarios: vinculado a la interfaz de gestión de usuarios. d. Alcaldías: vinculado a interfaz de gestión de alcaldías. e. Carpetas: vinculado a interfaz de gestión de carpetas. f. Proyectos: vinculado a interfaz de gestión de proyectos g. Reglas: vinculado a interfaz de reglas h. Log Out: cerrar sesión de software. 3. Dashboard: área de trabajo, para mostrara el área de trabajo.

Diseño del componente



Interfaz 3	CREAR USUARIOS
Descripción	Se requiere que el sistema provea una interfaz que sirva para la creación de usuarios. Para esto se necesitará contar con las siguientes entradas de texto:

	1. Alcaldía (Usar componente **<mat select>)** 2. Nombre. 3. Apellido. 4. Username 5. Email 6. Cargo 7. Rol(Usar componente **<mat select**>) 8. Contraseña 9. Confirmar Contraseña 10. Botón “Crear” para guardar al nuevo usuario
Diseño	Se utilizará el elemento **`<mat-form-field>`** Ver formulario de entrada descrito en sección **3.6.4**

Interfaz 4	**LISTAR USUARIOS**
Descripción	Se requiere que el sistema provea una lista de usuarios para administrarse (editar y eliminar), en donde pueda mostrarse la siguiente información de los usuarios: 1. Nombre 2. Apellido 3. UserName 4. Email 5. Rol Cada fila listada deberá contener botones de acción para editar y eliminar.
Diseño	Se utilizará el elemento **`<table-mat-table>`** Ver descripción del elemento listas en sección **3.6.5**

Interfaz 5	**CREAR ALCALDÍAS**
Descripción	Se requiere que el sistema provea una interfaz que sirva para la creación de Alcaldías. Para esto se necesitará contar con las siguientes entradas de texto requeridas para el ingreso de la información: 1. Nombre 2. Descripción 3. Dirección 4. Municipio 5. Ciudad

	6. Botón “Crear” para guardar la información
Diseño	Se utilizará el elemento **`<mat-form-field>`** Ver formulario de entrada descrito en sección **3.6.4**

Interfaz 6	**ADMINISTAR ALCALDIAS**
Descripción	Se requiere que el sistema provea una lista de alcaldias para administrarse (editar y eliminar), en donde pueda mostrarse la siguiente información de los usuarios: 1. Nombre 2. Descripción 3. Opciones (Cada fila listada deberá contener botones de acción para editar y eliminar).
Diseño	Se utilizará el elemento **`<table-mat-table>`** Ver descripción del elemento listas en sección **3.6.5**

Interfaz 7	**CREAR CARPETAS**
Descripción	Se requiere que el sistema provea una interfaz que sirva para la creación de contenedores denominado: “Carpetas”. Estas almacenarán los diferentes proyectos evaluados y se crearán usando un flujo de trabajo y se deberá crear en base a la siguiente estructura: Paso 1. Datos Generales: 1. Nombre 2. Descripción. Paso 2. Reglas: Se seleccionará una regla previamente creada en el menú de configuración utilizando el componente **`<mat-select>`**). Paso 3. Proyectos: Se facilitará al usuario el elegir los proyectos creados Paso 4. Guardar.
Diseño	Se utilizará el elemento **`<mat-horizontal-stepper>`.** Ver descripción del elemento paso a paso en sección **3.6.6** Se utilizará el elemento **`<mat-form-field>`** Ver descripción del elemento formulario de entrada en sección **3.6.4**

Interfaz 8	**ADMINISTRAR CARPETAS**
Descripción	Se requiere que el sistema provea una lista de carpetas para administrarse (editar y eliminar), en donde pueda mostrarse la siguiente información de las carpetas: 1. Nombre 2. Descripción 3. Opciones (Cada fila listada deberá contener botones de acción para editar, eliminar y realizar evaluación)
Diseño	Se utilizará el elemento **`<table-mat-table>`** Ver elemento de lista en sección **3.6.5**

Interfaz 9	**PROYECTOS**
Descripción	Se requiere que el sistema provea una interfaz que sirva para ingresar la información relacionada con los proyectos a evaluar. Se hará uso de un flujo de trabajo y se deberá crear en base a la siguiente estructura: Paso 1. General 1. Tipo de proyecto (Se deberá seleccionar utilizando el componente **`<mat-select>`**). 2. Nombre 3. Área Prioritaria (Se deberá seleccionar utilizando el componente **`<mat-select>`**). 4. Costo variable (Se deberá marcar esta opción utilizando el componente **`<mat-checkbox>`**) 5. Porcentaje. 6. Descripción. Paso 2. Ubicación: registro de coordenadas. Se ubicará una extensión de maps que facilite el ingreso de las coordenadas. 1. X (Ingreso de Latitud) 2. Y (Ingreso de Altitud) Paso 3. Detalles: 1. Beneficiarios. 2. Duración (en años) 3. Vida Útil 4. Inversión (Inversión Inicial) 5. Presupuesto 6. Valor Residual (Fijado en 10% pero habilitado para edición) Paso 4. Responsable

	1. Nombre del responsable 2. Apellido del responsable 3. Cargo Paso 5. Costos de operación: (Estos costos deberán mostrarse de manera dinámica en base a los tipos de proyectos establecidos en la ***sección 2.11***) Paso 6. Beneficios: (Estos beneficios deberán mostrarse de manera dinámica en base a los tipos de proyectos establecidos en la ***seccion 2.11***) Paso 7. Flujo de Caja: deberá mostrarse una matriz de acuerdo al cálculo de TIR del proyecto, basada en el cálculo del indicador número 1 detallado en la ***sección 2.12.1*** Paso 7. Finalizar. 1. Aceptar (Se utilizará el componente **`<mat-button>`**)
Diseño	Se utilizará el elemento **`<mat-horizontal-stepper>`.** Ver descripción del elemento paso a paso en sección **3.6.6** Se utilizará el elemento **`<mat-form-field>`** Ver descripción del elemento formulario de entrada en sección **3.6.4**

Interfaz 10	**LISTAR PROYECTOS**
Descripción	Se requiere que el sistema provea una lista de proyectos para administrarse (editar y eliminar), en donde pueda mostrarse la siguiente información de las carpetas: 1. Nombre 2. Descripción 3. Opciones (Cada fila listada deberá contener botones de acción para editar y eliminar)
Diseño	Se utilizará el elemento **`<table-mat-table>`** Ver elemento de lista en sección **3.6.5**

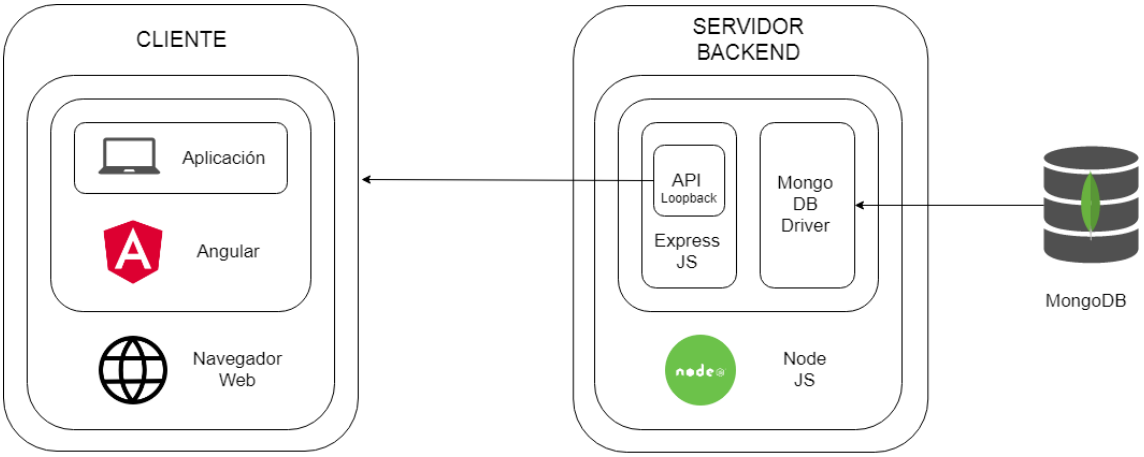
Interfaz 11	**CREAR REGLAS**
Descripción	Se requiere que el sistema provea una interfaz para la creación de reglas aplicables a los proyectos. Estas configuraciones se harán utilizando un flujo de trabajo para facilitar el ingreso de la información. La información requerida será en base a la siguiente estructura: Paso 1: Datos Generales

	1. Nombre 2. Descripción Paso 2: Áreas Prioritarias (Se listarán las áreas prioritarias descritas en la ***sección 2.12.5***) Paso 3: Indicadores. Se mostrará los indicadores TIR y VAN con un diseño de escala modificable de 0 a 10 para cada uno con su respectivo rango. Paso 4: Finalizar. Mostrar el botón guardar.
Diseño	Se utilizará el elemento **`<mat-horizontal-stepper>`.** Ver descripción del elemento paso a paso en sección **3.6.6** Se utilizará el elemento **`<mat-form-field>`** Ver descripción del elemento formulario de entrada en sección **3.6.4**

Interfaz 12	**LISTAR CONFIGURACIÓN**
Descripción	Se requiere que el sistema provea una lista de reglas para administrarse (editar y eliminar), en donde pueda mostrarse la siguiente información de las carpetas: 1. Nombre 2. Descripción Opciones (Cada fila listada deberá contener botones de acción para editar y eliminar)
Diseño	Se utilizará el elemento **`<table-mat-table>`** Ver elemento de lista en sección **3.6.5**

3.8 Interfaces de Software

La interacción de las tecnologías utilizadas en el desarrollo de este software viene dada por el diagrama, en el cual podemos resaltar las siguientes características:



3.9 Interfaces de Comunicación

En este apartado se definirán los requisitos de las funciones relacionados a la comunicación. Para lo cual detallamos las siguientes especificaciones:

3.9.1 Navegador web: el sistema debe de poder ser implementado en cualquier navegador web actualizado a su última versión.

3.9.2 Servicios en red: es necesaria la conexión estable del usuario al servidor mientras se administre algún contenido dentro del sistema.

3.9.3 Estándar de comunicación: se utilizó el estándar de comunicación HTTP. Debido a que la implementación de la aplicación es de manera local, no es necesaria la utilización de certificados de seguridad SSL.

3.9.4 Modo de autenticación: debido a que la aplicación no trabaja con sesiones para este requerimiento es necesaria la utilización de tokens. El backend es incapaz de recordar los datos de usuario. Para evitar que el backend en cada llamada deba recordar reiteradamente los datos de usuario y contraseña deberá generarse un token. Al iniciar la aplicación, el backend genera un token al usuario y el cliente en futuros accesos entrega ese token para certificar que está autenticado.

3. 10 Características del Software

En esta sección, incluiremos a detalle todas las características del software, su descripción, como también, sus requerimientos para que estas características puedan ser desarrolladas y como resultado, generen el objetivo del software.

Característica		**Descripción**
INICIO DE SESIÓN Y USUARIOS		Esta función describe las configuraciones iniciales para el acceso al software, los diferentes niveles de usuarios, roles y permisos.
No	**Nombre del requerimiento**	**Descripción**
RF01	Inicio de Sesión	Se requiere que el sistema disponga de un método que garantice el acceso a la aplicación con la utilización de un usuario y contraseña.
RF02	Alcaldías	Se requiere que el sistema sea capaz de asignarse una Alcaldía, tomando de un catálogo previamente configurado. El catálogo contará con la siguiente información: ● Nombre de Alcaldía ● Descripción ● Municipio ● Departamento
RF02	Usuarios	Se requiere que el sistema disponga de tres niveles de usuarios. ● ROOT ● ADMIN ● USER ***Ver sección 3.3***
RF03	ROOT	Se requiere la creación del usuario tipo ROOT, el cual tendrá permisos completos sobre la aplicación.
RF04	ADMIN	Se requiere la creación del usuario tipo ADMIN, este tendrá permisos sobre el CRUD dentro de la aplicación implementada en la alcaldía. También será el encargado de la creación de los usuarios USER.
RF05	USER	Se requiere la creación del usuario tipo USER, este usuario tendrá

		permisos limitados los cuales están descritos en la *sección 3.3*

Característica		**Descripción**
REGLAS E INDICADORES		Esta función describe la manera en la que la lógica del software evaluará los proyectos, también como deberá emplearse la administración de indicadores y la agrupación en reglas
No	**Nombre del requerimiento**	**Descripción**
RF06	Definición general de indicadores	Se requiere que el sistema implemente los indicadores: ● TIR (Tasa Interna de Retorno). ● VAN (Valor Actual Neto). ● Tiempo de Retorno de Inversión. ● Beneficiarios. ● Área Prioritaria.
RF07	Definición general de escalas de indicador	Se requiere que el sistema permita al usuario crear y configurar las escalas por indicador definidas por: ● Valor Inicial. ● Valor Final. ● Puntuación. Las escalas tendrán un valor permitido de 0 a 10, siendo 10 la mejor puntuación. ***Ver sección 2.10.4.***
RF08	Indicador TIR	Se requiere que el sistema sea capaz de calcular con los datos almacenados de un proyecto la Tasa Interna de Retorno.
RF09	Puntaje TIR	Se requiere como configuración estándar a este indicador, se agregue una escala dada por la ***tabla 2.18.*** Ver ***sección 2.12.1.***
RF10	Indicador VAN	Se requiere que el sistema sea capaz de calcular con los datos almacenados de un proyecto el Valor Actual Neto siguientes datos: ● Inversión ● Periodo ● Costos y Beneficios ● Interés

RF11	Puntaje VAN	Se requiere como configuración estándar a este indicador, se agregue una escala dada por la ***tabla 2.19*** descrita en la ***sección 2.12.2***.
RF12	Indicador: Tiempo de Retorno de Inversión	Se requiere que el sistema sea capaz de calcular con los datos almacenados de un proyecto el Tiempo de retorno de inversión con los siguientes datos: ● Periodo de recuperación de inversión ● Vida útil del proyecto
RF13	Puntaje Tiempo de Retorno de Inversión	Se requiere como configuración estándar a este indicador, se agregue una escala en base al cálculo descrito en la ***sección 2.12.3***
RF14	Indicador: Beneficiarios	Se requiere que el sistema sea capaz de almacenar la información de beneficiarios para ser utilizada en este indicador de acuerdo con los siguientes datos: ● Promedio de inversión por beneficiario del proyecto actual. ● Mínimo de todos los proyectos de inversión por beneficiario ● Máximo de todos los proyectos de inversión por beneficiario. ***Ver sección 2.12.4***
RF15	Puntaje Beneficiarios	Se requiere como configuración estándar a este indicador, se agregue una escala en base al cálculo descrito en la ***sección 2.12.4***
RF16	Indicador: Área Prioritaria	Se requiere que el sistema permita al usuario elegir un área prioritaria para cada proyecto.
RF17	Puntaje Área Prioritaria	Se requiere como configuración estándar la puntuación sugerida en la ***tabla 2.20*** de la ***sección 2.12.5***
RF18	Reglas	Se requiere que el sistema permite la creación de diferentes reglas para almacenar las diferentes configuraciones de indicadores. Estas estarán identificadas por un Nombre.

Característica		Descripción
CREAR PROYECTOS		Esta función es la encargada de almacenar la información general, económica y social de un proyecto.
No	**Nombre del requerimiento**	**Descripción**
RF19	Generales de proyecto	Se requiere que el sistema almacene la información general de un proyecto. La información a almacenar será la siguiente: ● Nombre de Alcaldía ● Tipo de proyecto ● Nombre del Proyecto ● Descripción del proyecto
RF20	Ubicación de Proyecto	Se requiere que el sistema brinde la opción de agregar la ubicación geográfica del proyecto ya sea utilizando una herramienta (Google Maps) o ingresada manualmente por el usuario.
RF21	Detalles de proyecto	Se requiere que el sistema almacene detalles económicos y sociales del proyecto: ● Beneficiarios ● Duración ● Vida útil (Años) ● Inversión Inicial ● Presupuesto ● Valor residual
RF22	Datos del responsable del Proyecto	Se requiere que el sistema almacene datos del responsable del proyecto: ● Nombre del responsable ● Apellido del responsable ● Cargo
RF23	Costos de Proyecto	Se requiere que el sistema almacene los costos económicos y que sean mostrados dinamicamente de acuerdo con el tipo de proyecto que se seleccionó en el RF14. Ver sección 2.11
RF24	Beneficios del Proyecto	Se requiere que el sistema almacene los beneficios económicos y que sean mostrados dinamicamente de acuerdo con el tipo de proyecto que se seleccionó en el RF14. Ver ***sección 2.11***
RF25	Guardar Proyecto	Se requiere que el sistema almacene la información en la base de datos

		luego que el usuario confirme que los datos se han ingresado correctamente

Característica		**Descripción**
CREAR CARPETAS		Esta función tiene como objetivo el agrupar en contenedores un grupo de proyectos para tener un mejor manejo de la aplicación de las reglas.
No	**Nombre del requerimiento**	**Descripción**
RF26	Generales de Carpeta	Se requiere el almacenamiento de información general de carpeta como identificador de estas. ● Nombre de Alcaldía ● Nombre de Carpeta ● Descripción
RF27	Selección de regla	Se requiere que a las carpetas se les pueda asignar las reglas deseadas para poder realizar el análisis multicriterio.
RF28	Selección de Proyectos	Se requiere que el sistema brinde el catálogo de proyectos ya creados para que sean agregados a las carpetas, de modo que al usuario se le facilite el agregar dichos proyectos a los contenedores. Ver detalle en RF26
RF29	Guardar Carpeta	Se requiere que el sistema brinde la función de almacenar los cambios a la carpeta siempre y cuando el usuario esté de acuerdo con la información almacenada.
RF30	Administrar Carpeta	Se requiere que el sistema brinde la opción de administrar las carpetas ya creadas y que se den permisos para editar o eliminar las carpetas.

Característica		Descripción
ASIGNAR PROYECTOS A CARPETAS		Esta característica proveerá la facilidad de poder agregar proyectos a carpetas para tener un mejor orden de los proyectos, además, el poder aplicar reglas de manera global a los proyectos incluidos en las carpetas.
No	**Nombre del requerimiento**	**Descripción**
RF31	Asignar proyectos a carpetas	Se requiere que el sistema brinde el listado de proyectos ingresados al software y almacenados para poder ser elegidos a una carpeta determinada.
RF32	Edición de asignación	El sistema debe de permitir al usuario eliminar (Esta acción no eliminará de la base de datos la información del proyecto elegido) de la carpeta los proyectos que han sido asignados a la carpeta.

Característica		Descripción
EVALUACIÓN DE PROYECTOS		Esta función describe la manera en la que la lógica del software evaluará los proyectos, también como deberá emplearse la administración de indicadores y la agrupación en reglas
No	**Nombre del requerimiento**	**Descripción**
RF33	Evaluación de proyectos	Se requiere que el sistema calcule los datos de inversión del proyecto en los diferentes indicadores y devuelva las ponderaciones calculadas.
RF34	Puntaje	Se requiere que el sistema asigne la sumatoria de los cálculos de los puntajes de los indicadores a cada proyecto evaluado.
RF35	Dashboard	Se requiere que el sistema muestre de manera ordenada el listado de proyectos por carpeta junto con us puntuación de forma descendente.

3.11 Lineamientos generales de Base de Datos

Los requerimientos básicos para la realización de la base de datos vienen dados de acuerdo a las siguientes características:

Tipo de base de datos: la base de datos se realizará bajo un esquema no relacional, utilizando como gestor a MongoDB.

Independencia de datos: es necesario que la base de datos no dependa del uso de la aplicación. Aunque los datos solamente serán utilizados por SmartProject es necesario que el objeto JSON sea independiente de la aplicación.

Seguridad y auditoría de acceso de datos: se requiere que el sistema cuente con un mecanismo de acceso a los datos de manera segura, implementado un login con usuarios y contraseñas y basada en token que se almacenen en el backend para garantizar la utilización segura de los mismos como también el respectivo log de usuarios para identificar los posibles cambios realizados dentro de la aplicación.

Reducción de la redundancia: se requiere que la base de datos evite la redundancia de información, duplicidad y/o datos que no sean requeridos en las características del software y así mejorar el rendimiento y reducir el espacio de memoria utilizada por esta.

Acceso concurrente: se requiere que la aplicación permita el uso de la información de manera simultánea para múltiples usuarios.

Integridad de los datos: se requiere que la base de datos almacene la información real e integra introducida por los usuarios. La aplicación debe de ejercer un control total sobre los datos en cuanto a escritura, lectura, modificación y eliminación que garantice las correcciones de manera permanente y sin cambios.

Acceso a datos mediante consultas programadas: la consulta de los datos sea eficiente, que garantice la respuesta rápida al usuario y en dado caso exista un error de lectura de datos este pueda ser mostrado al usuario e indique cómo resolver el problema.

3.12 Diseño de Seguridad

Según la tabla de requerimientos la seguridad está dada en base a los siguientes puntos:

3.12.1 Acceso al sistema.

El acceso al sistema será posible mediante la configuración de distintos niveles de acceso creados para cada tipo de usuario, cada usuario tendrá un rol diferente los cuales se describen en la siguiente tabla:

USUARIO	**DESCRIPCIÓN**
ROOT	Este usuario es único, y tiene acceso a todas las funcionalidades del sistema, además de poder crear usuarios con roles más limitados.
ADMIN	Este usuario tiene autorización para configurar la entidad de la institución en la aplicación.
USER	Este usuario tendrá uso limitado al sistema, su fin es el administrar la información requerida para que el software haga los análisis correspondientes, para eliminar información, será necesario que se comunique con un usuario ADMIN

3.12 Presupuesto estimado

La inversión necesaria para que este proyecto sea realizado viene detallado en el siguiente cuadro.

Recurso	**Tiempo invertido (Horas)**	**Costo por hora**	**Inversión**
Personal Logístico de Software	120	$10.00	$1200.00
Personal de desarrollo de Software*	960	$10.00	$9600.00
Equipos (Laptop)			$4000.00
Insumos y suministros			$500.00
****TOTAL**			**$15300.00**

*El costo por hora se ha calculado en base al pago de un empleado promedio dentro de El Salvador.

**El total es un cálculo estimado en base a nuestra experiencia en el rubro de la consultoría y desarrollo de software.

CAPITULO IV: DESARROLLO DEL SISTEMA WEB DE EVALUACION SOCIAL DE PROYECTOS

El presente capítulo describe los distintos planes que se idearon para llevar a cabo el proceso de desarrollo del sistema web de evaluación social de proyectos.

4.1 Plan de disponibilidad

En base a los requerimientos descritos en el capítulo anterior, se implementó un plan para garantizar que el sistema estará altamente disponible. Defínase disponibilidad como la cantidad de tiempo que el sistema realiza su función específica y permanece en estado operativo.

Cabe mencionar que el plan de disponibilidad debería de variar de acuerdo con la etapa de vida en la que se encuentra el sistema; por ejemplo, en la etapa de desarrollo se implementan soluciones que proporcionen la mayor cantidad de disponibilidad en tiempo y recursos para ejecutar código y desarrollar pruebas, así como las herramientas necesarias para la captura de información pertinente al desarrollador, con el fin de llevar a cabo mejoras y adecuaciones al código. Por otro lado, el plan de disponibilidad en la etapa de implementación final debe proponer soluciones que aumenten la confianza del usuario final en que el sistema cumplirá con los requisitos de disponibilidad y será estable y confiable en su entorno operativo.

En base a lo anterior delimitamos que el actual plan de disponibilidad corresponde a la etapa de desarrollo únicamente y no contempla la etapa de implementación del sistema.

4.1.1 Disponibilidad de hardware

Para la solución del presente desarrollo del sistema se buscó una solución que no requiriera la adquisición de hardware físico local, ya que eso incide en los costos de mantenimiento de un sistema. Gracias a las nuevas funcionalidades que brindan diversos servicios en línea, fue

posible encontrar un ambiente que brindara toda una plataforma de desarrollo, prueba e implementación en un mismo lugar. El servicio web implementado en este desarrollo fue Heroku. Heroku es una plataforma web que se encarga de proveer al desarrollador de todos los recursos necesarios para ejecutar una aplicación, liberando a la persona encargada de realizar instalaciones y configuraciones en los servidores web. Por lo tanto, en cuanto a la disponibilidad de hardware Heroku brinda diversas opciones dependiendo de la etapa de vida en la que se encuentre el sistema. Por ejemplo, durante la etapa de desarrollo, no es necesario contar con un servidor de grandes capacidades de concurrencia o disponibilidad continua, por lo que Heroku configura automáticamente recursos limitados básicos al entorno en donde se ha desplegado el sistema.

Por otro lado, finalizado el proceso de desarrollo, Heroku brinda la posibilidad de escalar el hardware que se está utilizando y adaptarlo a la necesidad de demanda que sea requerido. Esto es una enorme ventaja, ya que, si el sistema requiere soportar una concurrencia de miles de usuarios, Heroku se encarga de gestionar los recursos idóneos para satisfacer ese requerimiento.

Cabe mencionar que Heroku permite utilizar gratuitamente su plataforma durante el proceso de desarrollo, no así cuando sea necesario escalar recursos.

4.1.2 Disponibilidad de software

En seguimiento con la disponibilidad de hardware, Heroku se encarga internamente de gestionar, instalar e implementar las dependencias necesarias para la ejecución adecuada del sistema. Esto evita pérdida de tiempo del lado del equipo de desarrollo y sólo es requerido de parte de ellos el brindar a Heroku de un listado de las dependencias necesarias.

Se concluye entonces, que Heroku, se encarga a su vez de disponer del software requerido para la ejecución del sistema, así como también del hardware adecuado de acuerdo a la etapa en la que se encuentre el sistema.

4.2 Plan de comunicaciones

Con el fin de garantizar la recopilación, distribución y gestión oportuna y adecuada de la información del sistema para todos los participantes del proyecto, se utilizó un medio de comunicación que pudiera brindar el estado y las noticias sobre el avance del desarrollo del sistema a todas las partes interesadas apropiadas.

Como plataforma de comunicación se utilizó a Trello. Trello es una herramienta para la gestión de proyectos que permite la colaboración grupal. Trello utiliza un tablero (board) en donde se puede visualizar de manera general todo lo pertinente a las tareas y responsabilidades en el desarrollo del sistema. Además, es posible crear tarjetas (cards) que especifican las tareas a realizar dentro del proyecto y asignar dichas tarjetas a ciertos usuarios dentro del equipo de desarrollo.

Con la medida anterior se busca mantener una comunicación efectiva entre los miembros del equipo de desarrollo, asegurando que todos los participantes estén bien informados y tengan a su vez la información necesaria para tomar decisiones.

4.3 Plan de Desarrollo

En el presente plan se describe el proceso de desarrollo del sistema. Este plan complementa las especificaciones funcionales que proporcionan los detalles técnicos de lo que se codificó.

4.3.1 Pautas y estándares

Se establecieron pautas y estándares de desarrollo para garantizar que todos los miembros del equipo de desarrollo sean conscientes de las expectativas que se les imponen. A continuación, presentamos las pautas y estándares establecidos en el presente desarrollo.

4.3.2 Nombramiento

Para la nomenclatura de variables y archivos se definieron las siguientes normas:

- Evita nombres de una sola letra. Sé descriptivo con tus nombres.

```
// mal
function q() {
  // ...algo...
}

// bien
function query() {
  // ...algo...
}
```

- Usa camelCase cuando nombres tus objetos, funciones e instancias.

```
// mal
var OBJEcttssss = {};
var this_is_my_object = {};
function c() {};
var u = new user({
  name: 'Bob Parr'
});

// bien
var thisIsMyObject = {};
function thisIsMyFunction() {};
var user = new User({
  name: 'Bob Parr'
});
```

- Usa PascalCase cuando nombres constructores o clases.

```
// mal
function user(options) {
  this.name = options.name;
```

```
}

var bad = new user({
  name: 'nope'
});

// bien
function User(options) {
  this.name = options.name;
}

var good = new User({
  name: 'yup'
});
```

- Usa un guión bajo "_" adelante de la variable cuando nombres propiedades privadas.

```
// mal
this.__firstName__ = 'Panda';
this.firstName_ = 'Panda';

// bien
this._firstName = 'Panda';
```

- Cuando guardes una referencia a this usa _this.

```
// mal
function() {
  var self = this;
  return function() {
    console.log(self);
  };
}

// mal
function() {
  var that = this;
  return function() {
    console.log(that);
  };
}

// bien
function() {
  var _this = this;
  return function() {
    console.log(_this);
```

```
  };
}
```

- Nombra tus funciones. Esto será de ayuda cuando hagas seguimiento de la pila de llamadas (e.g. en caso de errores).

```
// mal
function() {
  var self = this;
  return function() {
    console.log(self);
  };
}

// mal
function() {
  var that = this;
  return function() {
    console.log(that);
  };
}

// bien
function() {
  var _this = this;
  return function() {
    console.log(_this);
  };
}
```

4.3.3 Control de versiones y fuente

Como sistema de control de versiones se utilizó GitHub, ya que posee todas las herramientas necesarias para el trabajo colaborativo con todo el equipo de desarrollo.

Básicamente se manejaron dos ramas en el desarrollo del sistema: “testing” y “master”. Testing fue utilizada como la rama de pruebas; por ejemplo, cuando un nuevo componente o funcionalidad finalizaba su proceso de desarrollo era probada en esta rama. Cuando esta nueva porción de código pasaba todas las pruebas realizadas por el personal encargado del control de calidad, se realizaba un “merge” o “unión” con la rama “master”.

De esta forma se mantuvo un control sobre el código y el avance general del sistema.

4.4 Componentes

A continuación, se describen los componentes que se desarrollaron de acuerdo con los lineamientos del diseño definido en el capítulo anterior. Cabe mencionar, que el código fuente que se muestra a continuación es el que se encuentra a la fecha en la rama “master” del control de código fuente. Es posible que en el momento de la lectura de este documento se hayan realizados actualizaciones en las tecnologías utilizadas o adecuaciones al código fuente; sin embargo, el funcionamiento y lógica del sistema no se verán afectados.

4.4.1 Base de datos

La base de datos del sistema fue desarrollada de acuerdo con los requerimientos de diseño. A continuación, presentamos la representación escrita de la base de datos no relacional:

```
{
   "name":"Alcaldia",
   "idInjection":true,
   "properties":{
      "nombre":{
         "type":"string",
         "required":true
      },
      "descripcion":{
         "type":"string",
         "required":true
      },
      "direccion":{
         "type":"string",
         "required":true
      },
      "municipio":{
         "type":"string",
         "required":true
      },
      "ciudad":{
         "type":"string",
```

```
            "required":true
         }
      }
   },
   {
      "name":"Carpeta",
      "idInjection":true,
      "properties":{
         "nombre":{
            "type":"string",
            "required":true
         },
         "descripcion":{
            "type":"string",
            "required":true
         },
         "proyectos":{
            "type":[
               "object"
            ],
            "required":false
         },
         "regla":{
            "type":"object",
            "required":true
         },
         "status":{
            "type":"boolean",
            "required":true,
            "default":false
         },
         "created":{
            "type":"date",
            "required":false,
            "defaultFn":"now"
         },
         "updated":{
            "type":"date",
            "required":false,
            "defaultFn":"now"
         },
         "idAlcaldia":{
            "type":"string",
            "required":true
         }
      }
```

```
  },
  {
     "name":"Evaluaciones",
     "idInjection":true,
     "properties":{
        "carpetaId":{
           "type":"string",
           "required":true
        },
        "reglaId":{
           "type":"string",
           "required":true
        },
        "carpeta":{
           "type":"object",
           "required":true
        },
        "regla":{
           "type":"object",
           "required":true
        },
        "created":{
           "type":"date",
           "required":false,
           "defaultFn":"now"
        },
        "updated":{
           "type":"date",
           "required":false,
           "defaultFn":"now"
        }
     }
  },
  {
     "name":"Proyecto",
     "idInjection":true,
     "properties":{
        "nombre":{
           "type":"string",
           "required":true
        },
        "descripcion":{
           "type":"string",
           "required":true
        },
        "beneficiarios":{
```

```
            "type":"number",
            "required":true
         },
         "beneficios":{
            "type":[
               "object"
            ],
            "required":false
         },
         "costos":{
            "type":[
               "object"
            ],
            "required":false
         },
         "objectivos":{
            "type":[
               "string"
            ],
            "required":false
         },
         "vidaUtil":{
            "type":"number",
            "required":true,
            "descripcion":"Vida util del proyecto ya implementado, en
años"
         },
         "duracion":{
            "type":"number",
            "required":true,
            "descripcion":"Meses"
         },
         "inversion":{
            "type":"number",
            "required":true,
            "descripcion":"Inversion inicial"
         },
         "presupuesto":{
            "type":"number",
            "required":true
         },
         "costoTotal":{
            "type":"number",
            "required":true,
            "descripcion":"Suma de todos los costos implicados"
         },
```

```
        "valorResidual":{
           "type":"number",
           "required":true
        },
        "ubicacion":{
           "type":"GeoPoint",
           "required":true
        },
        "tipoProyecto":{
           "type":"object",
           "required":true
        },
        "areaPrioritaria":{
           "type":"object",
           "required":true
        },
        "responsable":{
           "type":"object",
           "required":true
        },
        "status":{
           "type":"number",
           "required":true,
           "default":2
        },
        "created":{
           "type":"date",
           "required":false,
           "defaultFn":"now"
        },
        "updated":{
           "type":"date",
           "required":false,
           "defaultFn":"now"
        },
        "idAlcaldia":{
           "type":"string",
           "required":true
        },
        "costoConstante":{
           "type":"boolean",
           "required":false,
           "default":true
        },
        "porcentajeCostoVariable":{
           "type":"number",
```

```
            "required":false,
            "default":0
         },
         "flujoCaja":{
            "type":"object",
            "required":true
         }
      }
   },
   {
      "name":"Regla",
      "idInjection":true,
      "properties":{
         "nombre":{
            "type":"string",
            "required":true
         },
         "descripcion":{
            "type":"string",
            "required":true
         },
         "areasPrioritarias":{
            "type":[
               "object"
            ],
            "required":true
         },
         "indicadores":{
            "type":[
               "object"
            ],
            "required":true
         },
         "status":{
            "type":"boolean",
            "required":true,
            "default":false
         },
         "created":{
            "type":"date",
            "required":false,
            "defaultFn":"now"
         },
         "updated":{
            "type":"date",
            "required":false,
```

```
            "defaultFn":"now"
         },
         "idAlcaldia":{
            "type":"string",
            "required":true
         }
      }
   },
   {
      "name":"Users",
      "base":"User",
      "idInjection":true,
      "options":{
         "validateUpsert":true,
         "strictObjectIDCoercion":true
      },
      "properties":{
         "nombre":{
            "type":"string",
            "required":true
         },
         "apellido":{
            "type":"string",
            "required":true
         },
         "username":{
            "type":"string",
            "required":true
         },
         "email":{
            "type":"string",
            "required":true
         },
         "password":{
            "type":"string",
            "required":true
         },
         "cargo":{
            "type":"string",
            "required":false
         },
         "rol":{
            "type":"string",
            "required":true,
            "default":"user"
         },
```

```
        "status":{
           "type":"boolean",
           "required":true,
           "default":false
        },
        "created":{
           "type":"date",
           "required":false,
           "defaultFn":"now"
        },
        "updated":{
           "type":"date",
           "required":false,
           "defaultFn":"now"
        },
        "idAlcaldia":{
           "type":"string",
           "required":true
        }
     }
}
```

4.4.2 Back-end

La estructura de la aplicación del back-end es la presentada a continuación:

1. Client
    a. README.md

```
## Client

This is the place for your application front-end files.
```

2. Common
    a. Models
        i. alcaldia.js

```
"use strict";
module.exports = (Alcaldia) => {}
```

        ii. alcaldia.json

```
{
  "name": "Alcaldia",
  "idInjection": true,
```

```
    "options": {
      "validateUpsert": true
    },
    "properties": {
                                                "nombre": {
        "type": "string",
        "required": true
      },
      "descripcion": {
        "type": "string",
        "required": true
      },
      "direccion": {
        "type": "string",
        "required": true
      },
      "municipio": {
        "type": "string",
        "required": true
      },
      "ciudad": {
        "type": "string",
        "required": true
      }
    },
    "validations": [],
    "relations": {},
    "acls": [
      {
        "accessType": "*",
        "principalType": "ROLE",
        "principalId": "$unauthenticated",
        "permission": "DENY"
      },
      {
        "accessType": "*",
        "principalType": "ROLE",
        "principalId": "user",
        "permission": "DENY"
      },
      {
        "accessType": "*",
        "principalType": "ROLE",
        "principalId": "admin",
        "permission": "DENY"
      },
      {
        "accessType": "READ",
        "principalType": "ROLE",
        "principalId": "user",
        "permission": "ALLOW"
      },
      {
        "accessType": "READ",
        "principalType": "ROLE",
        "principalId": "admin",
        "permission": "ALLOW"
      },
      {
        "accessType": "*",
        "principalType": "ROLE",
        "principalId": "root",
        "permission": "ALLOW"
      }
    ],
    "methods": {}
  }
```

iii. calculos.js

```
"use strict";

const R = require('ramda');
const calculos = require('../../helpers/calculos');

module.exports = (Calculos) => {
  Calculos.evaluacion = (carpetaId, cb) => {
    const { Carpeta, Regla, Evaluaciones } = require('../../server/server').models;

    function getRelga(carpeta) {
      return new Promise((resolve, reject) => {
        Regla.findById(carpeta.regla.id, (err, regla) => {
          if (err || R.isNil(regla)) {
            if (R.isNil(regla)) {
              err = new Error(`Regla con ID: ${carpeta.regla[0].id}, no existe!`);
              err.status = 500;
            }
            return reject(err);
          }

          resolve({regla, carpeta});
        });
      });
    }

    function process(data) {
      return new Promise((resolve, reject) => {
        try {
          let regla = data.regla;
          let proyectos = data.carpeta.proyectos.map((proyecto) => {
            let puntajes = [];
            let TIR = calculos.TIR(proyecto);
            let VAN = calculos.VAN(TIR, proyecto);
            puntajes.push({         indicador:         "Area         Prioritaria",         puntaje:
calculos.puntajeAreaPrioritaria(proyecto, regla) });
            puntajes.push({ indicador: "TIR", puntaje: calculos.puntajeTIR(regla.indicadores, TIR)
});
            puntajes.push({ indicador: "VAN", puntaje: calculos.puntajeVAN(regla.indicadores, VAN,
proyecto.inversion) });
            puntajes.push({                indicador:                "TRI",                puntaje:
calculos.puntajeTiempoRetornoInversion(proyecto) });
            if (data.carpeta.proyectos.length > 1) {
              puntajes.push({           indicador:           "beneficiarios",           puntaje:
calculos.puntajeBeneficiarios(proyecto, data.carpeta.proyectos) });
            }
            proyecto["puntajes"] = puntajes;

            return proyecto;
          });

          data.carpeta.proyectos = proyectos;
          resolve(data);
        } catch (e) {
          reject(e);
        }
      });
    }

    function storeEvaluation(data) {
      return new Promise((resolve, reject) => {
        let evaluacion = {
          carpetaId: data.carpeta.id,
          reglaId: data.regla.id,
          carpeta: data.carpeta,
          regla: data.regla
        };
```

```
          Evaluaciones.create(evaluacion, (err, result) => {
            if (err) return reject (err);

            resolve(result);
          });
        });
      }

      let promise = new Promise((resolve, reject) => {
        Carpeta.findById(carpetaId, (err, carpeta) => {
          if (err || R.isNil(carpeta)) {
            if (R.isNil(carpeta)) {
              err = new Error(`Carpeta con ID: ${id}, no existe!`);
              err.status = 500;
            }
            return reject(err);
          }

          resolve(carpeta);
        });
      });

      promise
        .then(getRelga)
        .then(process)
        .then(storeEvaluation)
        .then(data => cb(null, data))
        .catch(err => cb(err, null));
    };

    Calculos.remoteMethod(
      'evaluacion',
      {
        description: 'Evaluar proyectos y asignar puntajes',
        http: { path: '/evaluacion', verb: 'post' },
        accepts: [
          {
            arg: 'carpetaId',
            type: 'string',
            required: true
          }
        ],
        returns: [
          { arg: 'evalucion', type: 'Evaluaciones', root: true }
        ]
      }
    );
  }
```

iv. calculos.json

```
  {
    "name": "Calculos",
    "options": {
      "idInjection": false,
      "validateUpsert": false
    },
    "properties": {},
    "validations": [],
    "relations": {},
    "acls": [
      {
        "accessType": "*",
        "principalType": "ROLE",
        "principalId": "$everyone",
```

```
      "permission": "ALLOW"
    }
  ],
  "methods": {}
}
```

v. carpeta.js

```
"use strict";

module.exports = (Carpeta) => {}
```

vi. carpeta.json

```
{
  "name": "Carpeta",
  "idInjection": true,
  "options": {
    "validateUpsert": true
  },
  "properties": {
    "nombre": {
      "type": "string",
      "required": true
    },
    "descripcion": {
      "type": "string",
      "required": true
    },
    "proyectos": {
      "type": [
        "object"
      ],
      "required": false
    },
    "regla": {
      "type": "object",
      "required": true
    },
    "status": {
      "type": "boolean",
      "required": true,
      "default": false
    },
    "created": {
      "type": "date",
      "required": false,
      "defaultFn": "now"
    },
    "updated": {
      "type": "date",
      "required": false,
      "defaultFn": "now"
    },
    "idAlcaldia": {
      "type": "string",
      "required": true
    }
  },
  "validations": [],
  "relations": {},
  "acls": [
    {
```

```
        "accessType": "*",
        "principalType": "ROLE",
        "principalId": "$unauthenticated",
        "permission": "DENY"
      },
      {
        "accessType": "*",
        "principalType": "ROLE",
        "principalId": "user",
        "permission": "DENY"
      },
      {
        "accessType": "deleteById",
        "principalType": "ROLE",
        "principalId": "user",
        "permission": "DENY"
      },
      {
        "accessType": "*",
        "principalType": "ROLE",
        "principalId": "user",
        "permission": "ALLOW"
      },
      {
        "accessType": "*",
        "principalType": "ROLE",
        "principalId": "admin",
        "permission": "ALLOW"
      },
      {
        "accessType": "*",
        "principalType": "ROLE",
        "principalId": "root",
        "permission": "ALLOW"
      }

    ],
    "methods": {}
  }
```

vii. evaluaciones.js

```
  "use strict";

  module.exports = (Evaluaciones) => {}
```

viii. evaluaciones.json

```
  {
    "name": "Evaluaciones",
    "idInjection": true,
    "options": {
      "validateUpsert": true,
      "strictObjectIDCoercion": true
    },
    "properties": {
      "carpetaId": {
        "type": "string",
        "required": true
      },
      "reglaId": {
        "type": "string",
        "required": true
```

```
      },
      "carpeta": {
        "type": "object",
        "required": true
      },
      "regla": {
        "type": "object",
        "required": true
      },
      "created": {
        "type": "date",
        "required": false,
        "defaultFn": "now"
      },
      "updated": {
        "type": "date",
        "required": false,
        "defaultFn": "now"
      }
    },
    "validations": [],
    "relations": {},
    "acls": [
      {
        "accessType": "*",
        "principalType": "ROLE",
        "principalId": "$unauthenticated",
        "permission": "DENY"
      },
      {
        "accessType": "READ*",
        "principalType": "ROLE",
        "principalId": "$authenticated",
        "permission": "ALLOW"
      },
      {
        "accessType": "WRITE*",
        "principalType": "ROLE",
        "principalId": "$authenticated",
        "permission": "ALLOW"
      }
    ],
    "methods": {}
  }
```

ix. proyecto.js

```
  "use strict";

  Const calculos= require('../../helpers/calculos');
  const R = require('ramda');

  module.exports = (Proyecto) => {

    /**
     * Update Proyecto
     *
     * @param {string} id - Identificador del Proyecto
     * @param {Object} params - Parametros a actualizar
     * @param {loopbackCallback} cb - La funcion callback a ejecutar
     */

     /**
      * Loopback Callback que devuelve los resultados
      *
      * @callback loopbackCallback
```

```
    * @param {Error} err - Error object If any
    * @param {Object} datra - Response Object
    */
  Proyecto.updateOne = function (id, params, cb) {
    function recalculate(proyecto) {
      try {
        let vidaUtil = proyecto.vidaUtil != params.vidaUtil;
        let inversion = proyecto.inversion != params.inversion;
        let tipoProyecto = proyecto.tipoProyecto.id != params.tipoProyecto.id;
        let valorResidual = proyecto.valorResidual != params.valorResidual;
        let costoConstante = proyecto.costoConstante != params.costoConstante;
        return vidaUtil || inversion || tipoProyecto || valorResidual || costoConstante;
      } catch (err) {
        console.error(error);
        return false;
      }

    }

    function update(proyecto) {
      return new Promise((resolve, reject) => {
        if (recalculate(proyecto)) {
          params.flujoCaja = [];
        }

        params.flujoCaja = calculos.flujoCaja(params);

        proyecto.updateAttributes(params, (err, instance) => {
          if (err) return reject(err);
          resolve(instance);
        });
      });
    }

    let promise = new Promise((resolve, reject) => {
      Proyecto.findById(id, (err, proyecto) => {
        if (err || R.isNil(proyecto)) {
          if (R.isNil(proyecto)) {
            err = new Error(`Proyecto con ID: ${id}, no existe!`);
            err.status = 500;
          }
          return reject(err);
        }

        resolve(proyecto);
      });
    });

    promise
      .then(update)
      .then(data => cb(null, data))
      .catch(err => cb(err, null));
  };

  Proyecto.remoteMethod(
    'updateOne',
    {
      description: 'Patch attributes for a model instance and persist it into the data source.',
      http: { path: '/updateOne/:id', verb: 'patch' },
      accepts: [
        {
          arg: 'id',
          type: 'string',
          required: true
        },
        {
          arg: 'params',
          type: 'Proyecto',
          http: { source: 'body' },
```

```
            description: 'Model instance data'
          }
        ],
        returns: { arg: 'data', type: 'Proyecto', root: true }
      }
    );

    Proyecto.observe('before save', function(ctx, next) {
      let flujo = (ctx.instance) ? ctx.instance.flujoCaja: ctx.data.flujoCaja;
      flujo = R.isNil(flujo) ? []: flujo;
      if (R.isEmpty(flujo)) {
        if (ctx.instance) {
          let flujoCaja = calculos.flujoCaja(ctx.instance);
          ctx.instance.flujoCaja = flujoCaja;
        } else  {
          let flujoCaja = calculos.flujoCaja(ctx.data);
          ctx.data.flujoCaja = flujoCaja;
        }
      }
      next();
    });
  }
```

x. proyecto.json

```
  {
    "name": "Proyecto",
    "idInjection": true,
    "options": {
      "validateUpsert": true
    },
    "properties": {
      "nombre": {
        "type": "string",
        "required": true
      },
      "descripcion": {
        "type": "string",
        "required": true
      },
      "beneficiarios": {
        "type": "number",
        "required": true
      },
      "beneficios": {
        "type": [
          "object"
        ],
        "required": false
      },
      "costos": {
        "type": [
          "object"
        ],
        "required": false
      },
      "objectivos": {
        "type": [
          "string"
        ],
        "required": false
      },
      "vidaUtil": {
        "type": "number",
        "required": true,
        "descripcion": "Vida util del proyecto ya implementado, en años"
```

```
        },
        "duracion": {
          "type": "number",
          "required": true,
          "descripcion": "Meses"
        },
        "inversion": {
          "type": "number",
          "required": true,
          "descripcion": "Inversion inicial"
        },
        "presupuesto": {
          "type": "number",
          "required": true
        },
        "costoTotal": {
          "type": "number",
          "required": true,
          "descripcion": "Suma de todos los costos implicados"
        },
        "valorResidual": {
          "type": "number",
          "required": true
        },
        "ubicacion": {
          "type": "GeoPoint",
          "required": true
        },
        "tipoProyecto": {
          "type": "object",
          "required": true
        },
        "areaPrioritaria": {
          "type": "object",
          "required": true
        },
        "responsable": {
          "type": "object",
          "required": true
        },
        "status": {
          "type": "number",
          "required": true,
          "default": 2
        },
        "created": {
          "type": "date",
          "required": false,
          "defaultFn": "now"
        },
        "updated": {
          "type": "date",
          "required": false,
          "defaultFn": "now"
        },
        "idAlcaldia": {
          "type": "string",
          "required": true
        },
        "costoConstante": {
          "type": "boolean",
          "required": false,
          "default": true
        },
        "porcentajeCostoVariable": {
          "type": "number",
          "required": false,
          "default": 0
        },
```

```
      "flujoCaja": {
        "type": "object",
        "required": true
      }
    },
    "validations": [],
    "relations": {},
    "acls": [
      {
        "accessType": "*",
        "principalType": "ROLE",
        "principalId": "$unauthenticated",
        "permission": "DENY"
      },
      {
        "accessType": "*",
        "principalType": "ROLE",
        "principalId": "user",
        "permission": "DENY"
      },
      {
        "accessType": "deleteById",
        "principalType": "ROLE",
        "principalId": "user",
        "permission": "DENY"
      },
      {
        "accessType": "*",
        "principalType": "ROLE",
        "principalId": "user",
        "permission": "ALLOW"
      },
      {
        "accessType": "*",
        "principalType": "ROLE",
        "principalId": "admin",
        "permission": "ALLOW"
      },
      {
        "accessType": "*",
        "principalType": "ROLE",
        "principalId": "root",
        "permission": "ALLOW"
      }
    ],
    "methods": {}
  }
```

xi. regla.js

```
  "use strict";

  module.exports = (Regla) => {}
```

xii. regla.json

```
  {
    "name": "Regla",
    "idInjection": true,
    "options": {
      "validateUpsert": true
    },
    "properties": {
```

```
        "nombre": {
          "type": "string",
          "required": true
        },
        "descripcion": {
          "type": "string",
          "required": true
        },
        "areasPrioritarias": {
          "type": [
            "object"
          ],
          "required": true
        },
        "indicadores": {
          "type": [
            "object"
          ],
          "required": true
        },
        "status": {
          "type": "boolean",
          "required": true,
          "default": false
        },
        "created": {
          "type": "date",
          "required": false,
          "defaultFn": "now"
        },
        "updated": {
          "type": "date",
          "required": false,
          "defaultFn": "now"
        },
        "idAlcaldia": {
          "type": "string",
          "required": true
        }
      },
      "validations": [],
      "relations": {},
      "acls": [
        {
          "accessType": "*",
          "principalType": "ROLE",
          "principalId": "$unauthenticated",
          "permission": "DENY"
        },
        {
          "accessType": "*",
          "principalType": "ROLE",
          "principalId": "user",
          "permission": "DENY"
        },
        {
          "accessType": "*",
          "principalType": "ROLE",
          "principalId": "root",
          "permission": "ALLOW"
        },
        {
          "accessType": "*",
          "principalType": "ROLE",
          "principalId": "admin",
          "permission": "ALLOW"
        },
        {
          "accessType": "READ",
```

```
        "principalType": "ROLE",
        "principalId": "user",
        "permission": "ALLOW"
      }
    ],
    "methods": {}
  }
```

xiii. settings.js

```
"use strict";

const config = require('../../config')

module.exports = (Settings) => {
  Settings.getAreasPrioritarias = (cb) => {
    cb(null, config.areasPrioritarias);
  };

  Settings.remoteMethod(
    'getAreasPrioritarias',
    {
      description: 'Get Areas Prioritarias',
      http: { path: '/getAreasPrioritarias', verb: 'get' },
      returns: [
        { arg: 'data', type: 'array', root: true }
      ]
    }
  );

  Settings.getIndicadores = (cb) => {
    cb(null, config.indicadores);
  };

  Settings.remoteMethod(
    'getIndicadores',
    {
      description: 'Get indicadores',
      http: { path: '/getIndicadores', verb: 'get' },
      returns: [
        { arg: 'data', type: 'array', root: true }
      ]
    }
  );

  Settings.getTipoProyectos = (cb) => {
    cb(null, config.tipoProyectos);
  };

  Settings.remoteMethod(
    'getTipoProyectos',
    {
      description: 'Get tipo proyectos',
      http: { path: '/getTipoProyectos', verb: 'get' },
      returns: [
        { arg: 'data', type: 'array', root: true }
      ]
    }
  );
}
```

xiv. settings.json

```
{
  "name": "Settings",
  "options": {
    "idInjection": false,
    "validateUpsert": false
  },
  "properties": {},
  "validations": [],
  "relations": {},
  "acls": [
    {
      "accessType": "*",
      "principalType": "ROLE",
      "principalId": "$unauthenticated",
      "permission": "DENY"
    },
    {
      "accessType": "*",
      "principalType": "ROLE",
      "principalId": "user",
      "permission": "DENY"
    },
    {
      "accessType": "*",
      "principalType": "ROLE",
      "principalId": "root",
      "permission": "ALLOW"
    },
    {
      "accessType": "*",
      "principalType": "ROLE",
      "principalId": "admin",
      "permission": "ALLOW"
    },
    {
      "accessType": "READ",
      "principalType": "ROLE",
      "principalId": "user",
      "permission": "ALLOW"
    }
  ],
  "methods": {}
}
```

xv. users.js

```
"use strict";

const regexp = require('../../helpers/regex');
const Promise = require('bluebird');
const { ObjectId } = require('mongodb')
const R = require('ramda');
var ObjectID = require('mongodb').ObjectID;

module.exports = (Users) => {
  Users.validatesFormatOf('email', { with: regexp.email, message: 'Ingrese un email valido' });

  /**
   * Create new user
   *
   * @param {Object} params Parametros desde el cliente
   * @callback {Function} cb La funcion callback
   * @param {Error} err Error object
   * @param {String} message Mensaje cuando no hay errores
   */
  Users.new = (params, cb) => {
```

```
    if (R.isNil(params.password) || R.isEmpty(params.password) ||
        R.isEmpty(params.username) || R.isNil(params.username)) {
      let message = R.isNil(params.password) ? 'El password ': 'El username ';
      message += 'es requerido.';
      return cb(new Error(message));
    }

    if (params.password.trim().toLowerCase() === params.username.trim().toLowerCase()) {
      return cb(new Error('Contraseña y nombre de usuario deben ser diferentes!'));
    }

    Users.create(params, (error, model) => {
      if (error) {
        console.log(error);
        return cb(error);
      } else {
        cb(null, 'Usuario creado correctamente!');
      }
    });
  }

  Users.observe('after save', (ctx, next) => {
    if (ctx.instance) {
      const { Role, RoleMapping } = require('../../server/server').models;
      let rol = (R.isNil(ctx.instance.rol) || R.isEmpty(ctx.instance.rol))
                ? 'user'
                : ctx.instance.rol.toLowerCase();

      Role.findOne({ where: { name: rol } }, (err, model) => {
        if (err) return console.error(err);
        if (model === null) return console.log('No role were found!');

        let role = {
          principalType: ctx.instance.rol,
          principalId: ObjectID(ctx.instance.id),
          roleId: model.id
        };

        RoleMapping.create(role, (err, roleMapping) => {
          if (err) return console.error(err);

          console.log('RoleMapping for user id: ' + ctx.instance.id);
          console.log('Result:', roleMapping);
        });
      });
    }

    next();
  });

  Users.remoteMethod(
    'new',
    {
      description: 'Crear un nuevo usuario',
      http: { path: '/new', verb: 'post' },
      accepts: {
        arg: 'params',
        type: Users,
        http: { source: 'body' },
        description: 'Model instance data'
      },
      returns: { arg: 'message', type: 'string' }
    }
  )

  /**
   * Login
   *
   * @param {Object} params Parametros desde el cliente
```

```
     * @callback {Function} cb La funcion callback
     * @param {Error} err Error object
     * @param {String} message Mensaje cuando no hay errores
     */
    Users.signin = (username, password, cb) => {

      function login(user) {
        return new Promise((resolve, reject) => {
          Users.login({username: username, password: password}, (err, token) => {
            if (err) return reject(err);
            let results = {
              token,
              user: {
                id: user.id,
                rol: user.rol,
                cargo: user.cargo,
                email: user.email,
                nombre: user.nombre,
                apellido: user.apellido,
                username: user.username,
                idAlcaldia: user.idAlcaldia,
              },
            }

            resolve(results);
          });
        });
      }

      function getAlcaldia(data) {
        const { Alcaldia } = require('../../server/server').models;
        if (data.user.rol === "root") {
          data.user.alcaldia = {};
          return Promise.resolve(data);
        } else {
          return new Promise((resolve, reject) => {
            Alcaldia.findById(ObjectId(data.user.idAlcaldia), (err, alcaldia) => {
              if (err) return reject(err);
              if (alcaldia) {
                data.user.alcaldia = alcaldia;
              }
              resolve(data);
            });
          });
        }
      }

      let promise = new Promise((resolve, reject) => {
        Users.findOne({ where: { username: username, status: 1 } }, (err, user) => {
          if (err) return reject(err);
          if (user) {
            resolve(user);
          } else {
            reject(new Error('Usuario bloquedo, favor comunicarse con soporte!'));
          }
        });
      });

      promise
      .then(login)
      .then(getAlcaldia)
      .then(token => cb(null, token))
      .catch(err => cb(err, null));
    };

    Users.remoteMethod(
      'signin',
      {
        description: 'Iniciar session',
```

```
        http: { path: '/signin' },
        accepts: [
          { arg: 'username', type: 'string', required: true },
          { arg: 'password', type: 'string', required: true }
        ],
        returns: { arg: 'accessToken', type: 'object', root: true }
      }
    );
  }
```

xvi. users.json

```
{
  "name": "Users",
  "base": "User",
  "idInjection": true,
  "options": {
    "validateUpsert": true,
    "strictObjectIDCoercion": true
  },
  "properties": {
    "nombre": {
      "type": "string",
      "required": true
    },
    "apellido": {
      "type": "string",
      "required": true
    },
    "username": {
      "type": "string",
      "required": true
    },
    "email": {
      "type": "string",
      "required": true
    },
    "password": {
      "type": "string",
      "required": true
    },
    "cargo": {
      "type": "string",
      "required": false
    },
    "rol": {
      "type": "string",
      "required": true,
      "default": "user"
    },
    "status": {
      "type": "boolean",
      "required": true,
      "default": false
    },
    "created": {
      "type": "date",
      "required": false,
      "defaultFn": "now"
    },
    "updated": {
      "type": "date",
      "required": false,
      "defaultFn": "now"
    },
    "idAlcaldia": {
```

```
        "type": "string",
        "required": true
      }
    },
    "validations": [],
    "relations": {
      "accessTokens": {
        "type": "hasMany",
        "model": "AccessToken",
        "foreignKey": "userId"
      }
    },
    "acls": [
      {
        "accessType": "READ",
        "principalType": "ROLE",
        "principalId": "$everyone",
        "permission": "ALLOW"
      },
      {
        "accessType": "EXECUTE",
        "principalType": "ROLE",
        "principalId": "$everyone",
        "permission": "ALLOW",
        "property": [
          "new",
          "login",
          "signin"
        ]
      },
      {
        "accessType": "WRITE",
        "principalType": "ROLE",
        "principalId": "$authenticated",
        "permission": "ALLOW"
      }
    ],
    "methods": {}
  }
```

3. Config

    a. areasPrioritarias.json

```
  {
      "data": [
        {
          "id": 1,
          "nombre": "Medio Ambiente y Recursos Naturales",
          "descripcion": ""
        },
        {
          "id": 2,
          "nombre": "Generación Energética",
          "descripcion": ""
        },
        {
          "id": 3,
          "nombre": "Seguridad Alimentaria y Nutricional",
          "descripcion": ""
        },
        {
          "id": 4,
          "nombre": "Salud",
          "descripcion": ""
```

```
        },
        {
          "id": 5,
          "nombre": "Educación",
          "descripcion": ""
        },
        {
          "id": 6,
          "nombre": "Mitigación de Riesgos",
          "descripcion": ""
        },
        {
          "id": 7,
          "nombre": "Energía Eléctrica",
          "descripcion": ""
        },
        {
          "id": 8,
          "nombre": "Proyectos Tecnológicos",
          "descripcion": ""
        },
        {
          "id": 9,
          "nombre": "Ordenamiento territorial",
          "descripcion": ""
        },
        {
          "id": 10,
          "nombre": "Protección al patrimonio cultural",
          "descripcion": ""
        },
        {
          "id": 11,
          "nombre": "Construcción",
          "descripcion": ""
        },
        {
          "id": 12,
          "nombre": "Ordenamiento urbano",
          "descripcion": ""
        },
        {
          "id": 13,
          "nombre": "Deportes",
          "descripcion": ""
        },
        {
          "id": 14,
          "nombre": "Otros proyectos",
          "descripcion": ""
        }
      ]
}
```

b. index.js

```
const areasPrioritarias = require('./areasPrioritarias.json').data;
const indicadores = require('./indicadores.json').data;
const tipoProyectos = require('./tipoProyectos.json').data;

module.exports = {
  areasPrioritarias: areasPrioritarias,
  indicadores: indicadores,
  tipoProyectos: tipoProyectos
};
```

c. indicadores.json


```
   {
     "data": [
       {
         "id": 1,
         "nombre": "TIR",
         "descripcion": "Tasa Interna de Retorno",
         "objectivos": "",
         "formula": {
           "parametros": {
             "Fn": "Flujo caja en periodo n",
             "n": "Numero de periodos",
             "i": "Inversion inicial"
           },
           "calculo": "TIR = Σ(Fn/(1+i)^n) = 0"
         },
         "editable": true
       },
       {
         "id": 2,
         "nombre": "VAN",
         "descripcion": "Valor Actual Neto",
         "objectivos": "",
         "formula": {
           "parametros": {
             "n": "numero de periodos",
             "I": "Inversión Inicial",
             "Fn": "Flujo de caja en periodo n",
             "TIR": "Tasa Interna de Retorno"
           },
           "calculo": "VAN = Σ[(Fn/(1+TIR)^n) - I] = 0"
         },
         "editable": true
       },
       {
         "id": 3,
         "nombre": "tiempoRetorno",
         "descripcion": "Tiempo de Recuperacion de la Inversion",
         "objectivos": "",
         "formula": {
           "parametros": {
             "x": "",
             "y": ""
           },
           "calculo": ""
         },
         "editable": false
       },
       {
         "id": 4,
         "nombre": "beneficiarios",
         "descripcion": "Promedio per cápita invertido, donde el objetivo de un proyecto es beneficiar
a la mayor cantidad de pobladores",
         "objectivos": "",
         "formula": {
           "parametros": {
             "x": "",
             "y": ""
           },
           "calculo": ""
         },
         "editable": false
       }
     ]
```

```
    }
```

d. tipoProyectos.json

```
   {
     "data": [
       {
         "id": 1,
         "nombre": "Agua Potable",
         "nombreCompleto": "Proyectos de Agua Potable",
         "costos": [
           {
             "nombre": "Costos de Operacion",
             "descripcion": "Se registran a lo largo de la vida útil del proyecto y están asociados al
buen funcionamiento del mismo, por ejemplo: Energía eléctrica, Agua cruda, Productos químicos",
             "monto": ""
           }
         ],
         "beneficios": [
           {
             "nombre": "Beneficios por mayor consumo",
             "descripcion": "Originados por el consumo de agua potable, se incluyen: Ingresos por
consumo de agua, Ahorro en aseo personal, Ahorro en actividades domésticas, Oportunidad de realizar
actividades productivas",
             "monto": ""
           },
           {
             "nombre": "Liberación de recursos",
             "descripcion": "Son beneficios originados al comparar la situación sin proyecto, pues el
agua es un recurso vital que implica esfuerzos de los pobladores por su aprovisionamiento: Ahorro en
tiempo de obtención de agua, Ahorro en mejora de la salud",
             "monto": ""
           }
         ]
       },
       {
         "id": 2,
         "nombre": "Evacuacion y Drenaje de Aguas lluvias",
         "nombreCompleto": "Proyectos de Evacuación y Drenaje de Aguas lluvias",
         "costos": [
           {
             "nombre": "Costos de Operacion",
             "descripcion": "Se registran a lo largo de la vida útil del proyecto y están asociados al
buen funcionamiento del mismo, por ejemplo: Mantenimiento y limpieza de infraestructura, Mano de obra",
             "monto": ""
           }
         ],
         "beneficios": [
           {
             "nombre": "Beneficio por menor daño en propiedades residenciales",
             "descripcion": "El beneficio por menor daño en propiedades residenciales será equivalente
al cambio en el precio de la vivienda al mejorar la condición de evacuación y drenaje de aguas lluvias.",
             "monto": ""
           },
           {
             "nombre": "Beneficio por recuperación de terrenos baldíos anegadizos",
             "descripcion": "El beneficio de las propiedades que no cuentan con construcción que se
ven mejorados por el proyecto de aguas lluvias.",
             "monto": ""
           },
           {
             "nombre": "Beneficio por menor daño de propiedades comerciales o industriales",
             "descripcion": "Son los beneficios originados para los comercios o industrias, las cuales
con el proyecto ejecutado ahorraran en pérdidas ocasionadas por las inundaciones.",
             "monto": ""
           },
```

```
            {
              "nombre": "Beneficio por menor deterioro de infraestructura vial",
              "descripcion": "Este se compone de la diferencia entre los costos anuales de reposición
de la infraestructura vial deteriorada con proyecto y sin proyecto.",
              "monto": ""
            },
            {
              "nombre": "Beneficios por reducción de los costos de viaje",
              "descripcion": "Se calcula la diferencia de los costos de viaje sin proyecto y los costos
de viaje con el proyecto. Tomando en cuenta las afectaciones en costos de viaje originadas por
inundaciones en la infraestructura vial.",
              "monto": ""
            },
            {
              "nombre": "Beneficios por menor ausentismo laboral",
              "descripcion": "Beneficio originado por el ahorro en pérdidas debidas a la suspensión de
actividades productivas asociadas a una inundación de un comercio, una industria o una institución
pública.",
              "monto": ""
            },
            {
              "nombre": "Beneficios por reducción de enfermedades.",
              "descripcion": "Este beneficio puede ser difícil de calcular en muchos casos, por lo que
puede indicarse como un beneficio intangible",
              "monto": ""
            }
          ]
        },
        {
          "id": 3,
          "nombre": "Alumbrado Público",
          "nombreCompleto": "Proyectos de Alumbrado Público",
          "costos": [
            {
              "nombre": "Costos de Operacion",
              "descripcion": "Se registran a lo largo de la vida útil del proyecto y están asociados al
buen funcionamiento del mismo, por ejemplo: Mantenimiento y limpieza del sistema de alumbrado, Mano de
obra, etc.",
              "monto": ""
            }
          ],
          "beneficios": [
            {
              "nombre": "Disminución de costos de operación y mantenimiento",
              "descripcion": "Acorde al avance tecnológico, se produce un ahorro de operación y de
mantenimiento en función que los componentes del sistema de alumbrado, tienen mayor vida útil, y menor
consumo de energía eléctrica.",
              "monto": ""
            },
            {
              "nombre": "Disminución de gases de efecto invernadero",
              "descripcion": "Hay un impacto asociado en los gases de efecto invernadero asociados a la
situación sin proyecto, dado que puede emplearse quema de combustibles para proveer de iluminación.",
              "monto": ""
            },
            {
              "nombre": "Beneficio por disminución de contaminación lumínica",
              "descripcion": "Este beneficio puede considerarse intangible si se dificulta su cálculo.",
              "monto": ""
            },
            {
              "nombre": "Ahorro por disposición de lámparas contaminantes",
              "descripcion": "Muchos sistemas de alumbrado público antiguas estaban conformadas por
lámparas de mercurio y/o sodio, las cuales requieren un proceso adecuado de disposición final debido
a sus contaminantes",
              "monto": ""
            }
          ]
        },
```

```
        {
          "id": 4,
          "nombre": "Atención en Salud Primaria",
          "nombreCompleto": "Proyectos de Atención en Salud Primaria",
          "costos": [
            {
              "nombre": "Costos de Operacion",
              "descripcion": "Son los costos fijos del establecimiento: Remuneraciones del personal,
Viáticos, Farmacia, Materiales e insumos, Gastos de servicios básicos",
              "monto": ""
            },
            {
              "nombre": "Costos de Mantención",
              "descripcion": "Considera los gastos de mantención de la infraestructura y de los
equipos.",
              "monto": ""
            }
          ],
          "beneficios": [
            {
              "nombre": "Ahorro por pronta y oportuna atención médica",
              "descripcion": "Entre más pronto sea atendida una enfermedad, requiere de menor atención
médica, menor medicamente empleado y de menor tiempo de recuperación del paciente.",
              "monto": ""
            },
            {
              "nombre": "Ahorro en reducción de tiempos de espera y atención",
              "descripcion": "Un paciente para acudir a un centro asistencial de salud, suspende su
actividad productiva, dejando su trabajo y gastando ese tiempo en la atención médica.",
              "monto": ""
            },
            {
              "nombre": "Ahorro en costos de traslado",
              "descripcion": "Al acercar la atención médica a una población, se reducen los costos de
traslado, que en muchos casos implica traslados en animales de carga como caballos, o camionetas.",
              "monto": ""
            },
            {
              "nombre": "Beneficio en la mejora de los indicadores de salud",
              "descripcion": "Beneficios vinculados a la mejora en indicadores como la tasa de mortalidad
infantil, número de controles por cada mil habitantes, etc.",
              "monto": ""
            }
          ]
        },
        {
          "id": 5,
          "nombre": "Edificación Pública",
          "nombreCompleto": "Proyectos de Edificación Pública",
          "costos": [
            {
              "nombre": "Costos de Operacion",
              "descripcion": "Son los costos fijos del establecimiento: Remuneración de personal, Gastos
de servicios básicos",
              "monto": ""
            },
            {
              "nombre": "Costos de Mantención",
              "descripcion": "Considera los gastos de mantención de la infraestructura y de los
equipos.",
              "monto": ""
            }
          ],
          "beneficios": [
            {
              "nombre": "Ahorro en Costos de Operación",
              "descripcion": "Corresponde a ahorros en gastos por concepto de remuneraciones de
personal, servicios básicos (agua, energía eléctrica, gas, combustible), etc.",
              "monto": ""
```

```
            },
            {
              "nombre": "Ahorro en costos de reparación",
              "descripcion": "La reparación de edificios contempla los trabajos destinados a subsanar el deterioro sufrido en un inmueble en forma ocasional o por falta de conservación y que se traduce en la reposición de elementos fundamentales dañados; es decir, aquellos que afectan a la estructura física, a sus instalaciones y/o modifican substancialmente la concepción arquitectónica artística del edificio y que por lo tanto requieren de una asistencia técnica especializada del nivel profesional.",
              "monto": ""
            },
            {
              "nombre": "Ahorro en costos de remodelación",
              "descripcion": "La remodelación de edificios contempla los trabajos destinados a reorganizar el espacio interior de un edificio y/o adecuar sus instalaciones.",
              "monto": ""
            },
            {
              "nombre": "Mejoramiento de la eficiencia  del personal.",
              "descripcion": "Aumento de productividad del personal que labora en la institución: El aumento de productividad proviene de un ahorro de tiempo en el desplazamiento de los funcionarios. Este beneficio se percibe más claramente en aquellos proyectos que persiguen concentrar, en un solo edificio, dependencias que funcionan en localizaciones separadas.",
              "monto": ""
            },
            {
              "nombre": "Ahorro por liberación de activos",
              "descripcion": "Si la institución es propietaria de uno o más inmuebles y éstos quedaran liberados a causa del proyecto, deberá considerarse como beneficio el valor de su venta, siempre y cuando éste tenga un uso alternativo.",
              "monto": ""
            }
          ]
        },
        {
          "id": 6,
          "nombre": "Vialidad",
          "nombreCompleto": "Proyectos de Vialidad",
          "costos": [
            {
              "nombre": "Costos de Mantención",
              "descripcion": "Considera los gastos de mantención de la infraestructura y de los equipos.",
              "monto": ""
            }
          ],
          "beneficios": [
            {
              "nombre": "Ahorro en tiempo de viaje",
              "descripcion": "Al contar con caminos más directos o más amplios, los usuarios demoran menos en desplazarse de un lugar a otro. El cálculo del ahorro de tiempo, medido en horas, se realiza obteniendo la diferencia entre el tiempo de viaje con y sin proyecto.",
              "monto": ""
            },
            {
              "nombre": "Ahorro en costos de operación vehicular",
              "descripcion": "Al mejorar la calidad de la carpeta de rodadura de la vía, se producen una mejora en el rendimiento de lubricantes, aumenta la vida útil de los neumáticos y del vehículo mismo.",
              "monto": ""
            },
            {
              "nombre": "Ahorro de combustible.",
              "descripcion": "Debido a la mejora de las condiciones de la calle, habrá un ahorro de combustible en razón de la velocidad de desplazamiento de los automotores.",
              "monto": ""
            },
            {
              "nombre": "Ahorro en costos de mantenimiento.",
```

```
            "descripcion": "Esto se produce cuando en la situación sin proyecto se requieren constantes
y recurrentes reparaciones para poder continuar con la vía operativa. Para su estimación y valoración,
deberá estimarse los costos de mantenimiento de las situaciones sin y con proyecto.",
            "monto": ""
          }
        ]
      },
      {
        "id": 7,
        "nombre": "Educación",
        "nombreCompleto": "Proyectos de Educación",
        "costos": [
          {
            "nombre": "Costos de Mantención",
            "descripcion": "Considera los gastos de mantención de la infraestructura educativa.",
            "monto": ""
          },
          {
            "nombre": "Costos de Transporte",
            "descripcion": "Los beneficiarios del proyecto recurren a gastos para desplazarse desde
sus hogares hasta su centro educativo.",
            "monto": ""
          }
        ],
        "beneficios": [
          {
            "nombre": "Aumento en el nivel de productividad de los beneficiarios",
            "descripcion": "Al mejorar el nivel educativo las oportunidades de trabajo crecen, y por
ende una mejora en el nivel de ingresos de las familias.",
            "monto": ""
          },
          {
            "nombre": "Disminución de conductas antisociales",
            "descripcion": "Debido a que los jóvenes logran adquirir nuevas oportunidades, se reduce
la probabilidad de buscar la alternativa delincuencial.",
            "monto": ""
          },
          {
            "nombre": "Ahorro en costos de transporte",
            "descripcion": "Muchos de los jóvenes requieren de esfuerzos mayores para poder optar a
educación, viéndose obligados a recorrer grandes distancias en el escenario sin proyecto para cubrir
su necesidad educativa.",
            "monto": ""
          }
        ]
      },
      {
        "id": 8,
        "nombre": "Electrificación",
        "nombreCompleto": "Proyectos de Electrificación",
        "costos": [
          {
            "nombre": "Costos de Operación",
            "descripcion": "Son los costos fijos del establecimiento: Remuneración de personal, Gastos
de servicios básicos.",
            "monto": ""
          },
          {
            "nombre": "Costos de Mantención",
            "descripcion": "Considera los gastos de mantención de la infraestructura y de los
equipos.",
            "monto": ""
          }
        ],
        "beneficios": [
          {
            "nombre": "Ingresos por ventas y tasas",
            "descripcion": "Tanto privados como el estado percibe beneficios a raíz de la venta del
servicio de energía eléctrica así como del gravamen a la infraestructura del mismo.",
```

```
              "monto": ""
            },
            {
              "nombre": "Ahorro en recursos alternos",
              "descripcion": "En el escenario sin proyecto, las familias recurren al uso de alternativas
de iluminación, como la quema de combustibles.",
              "monto": ""
            },
            {
              "nombre": "Oportunidades de productividad",
              "descripcion": "El servicio de energía eléctrica brinda un abanico de oportunidades para
el crecimiento económico como por ejemplo el poder poner un taller de mecánica, una molino, etc.",
              "monto": ""
            }
          ]
        },
        {
          "id": 9,
          "nombre": "Residuos Solidos",
          "nombreCompleto": "Proyectos de Residuos Solidos",
          "costos": [
            {
              "nombre": "Costos de Operación",
              "descripcion": "Son los costos fijos del establecimiento: Remuneración de personal, Gastos
de servicios básicos.",
              "monto": ""
            },
            {
              "nombre": "Costos de Mantención",
              "descripcion": "Considera los gastos de mantención de la infraestructura y de los
equipos.",
              "monto": ""
            },
            {
              "nombre": "Costos de transporte",
              "descripcion": "Se refiere a los costos implicados en recoger y transportar los desechos
sólidos hasta el lugar de su adecuado tratamiento.",
              "monto": ""
            }
          ],
          "beneficios": [
            {
              "nombre": "Ingresos por venta de servicio",
              "descripcion": "Se produce un ingreso por el cobro de tonelada de disposición final de
desechos. En algunos casos también se produce una venta a raíz de los desechos transformados.",
              "monto": ""
            },
            {
              "nombre": "Ahorro en recursos alternos",
              "descripcion": "En el escenario sin proyecto, las familias recurren al uso de alternativas
como el entierro de desechos o la quema de los mismos.",
              "monto": ""
            }
          ]
        },
        {
          "id": 10,
          "nombre": "Seguridad Policial",
          "nombreCompleto": "Proyectos de Seguridad Policial",
          "costos": [
            {
              "nombre": "Costos de Operación",
              "descripcion": "Son los costos fijos del establecimiento: Remuneración de personal, Gastos
de servicios básicos.",
              "monto": ""
            },
            {
              "nombre": "Costos de Mantención",
```

```
            "descripcion": "Considera los gastos de mantención de la infraestructura y de los
equipos.",
            "monto": ""
          }
        ],
        "beneficios": [
          {
            "nombre": "Ahorro por reducción de delincuencia",
            "descripcion": "En términos de Economía, se puede aducir que los mayores niveles de
seguridad ciudadana que se logran con niveles adicionales de vigilancia policial constituyen bienes de
consumo, en tanto las personas se gratifican por el hecho objetivo de desenvolverse en un medio ambiente
más seguro, al igual que por percibir un entorno de mayor seguridad. Puede cuantificarse por medio de
los costos implicados por delitos.",
            "monto": ""
          },
          {
            "nombre": "Mejora en el ambiente de inversión",
            "descripcion": "Condiciones de mayor seguridad generan ambientes favorables al desarrollo
de actividades productivas",
            "monto": ""
          }
        ]
      },
      {
        "id": 11,
        "nombre": "Otros",
        "nombreCompleto": "Otros Proyectos",
        "costos": [
          {
            "nombre": "Costos Privados",
            "descripcion": "Aquellos que implican una erogación de fondos para por parte del ejecutor
o beneficiario privado del proyecto.",
            "monto": ""
          },
          {
            "nombre": "Costos Sociales",
            "descripcion": "Aquellos costos que están asociados a las comunidades o personas alrededor
del proyecto.",
            "monto": ""
          }
        ],
        "beneficios": [
          {
            "nombre": "Beneficios Privados",
            "descripcion": "Son los ingresos que se producirán a partir de la ejecución del proyecto,
los que serán captados por la organización ejecutora del mismo.",
            "monto": ""
          },
          {
            "nombre": "Beneficios Sociales",
            "descripcion": "Son beneficios recibidos por las comunidades o personas que interactúan
con el proyecto.",
            "monto": ""
          }
        ]
      }
    ]
  }
```

4. Helpers

    a. calculos.js

```
  const Big = require('big.js');
  const R = require('ramda');
```

```
    const DP = 6; // Decimal places
    Big.DP = DP;
    Big.RM = 1;

    module.exports = {
      flujoCaja: (proyecto, porcetaje = 0) => {
        let  beneficios  =  proyecto.beneficios.reduce((prev,  curr)  =>  (prev.plus(curr.monto)),  new
Big(0));
        let costos = proyecto.costos.reduce((prev, curr) => (prev.plus(curr.monto)), new Big(0));
        let residual = new Big(proyecto.valorResidual);
        let porcetajeVariable = new Big(0);
        let flujo = [];
        let recalcular = !R.isEmpty(proyecto.flujoCaja) || !R.isNil(proyecto.flujoCaja);
        if (!R.isNil(proyecto.porcentajeCostoVariable)) {
          porcetajeVariable = new Big(proyecto.porcentajeCostoVariable).div(100);
        }
        for (let i = 0; i < proyecto.vidaUtil; i++) {
          if (i === 0) {
            flujo.push({
              anio: i,
              monto: new Big(proyecto.inversion).toFixed(DP),
              costos: 0,
              beneficios: 0
            });
          } else if (i === (proyecto.vidaUtil - 1)) {
            if (!porcetajeVariable.eq(0)) {
              costos = costos.plus(costos.times(porcetajeVariable));
              beneficios = beneficios.plus(beneficios.times(porcetajeVariable));
            }

            if (recalcular) {
              costos = new Big(proyecto.flujoCaja[i].costos);
              beneficios = new Big(proyecto.flujoCaja[i].beneficios);
            }

            var monto = new Big(0);

            if (residual.gt(0)) {
              monto = beneficios.minus(costos).plus(residual.div(100).plus( new Big(proyecto.inversion)
));
            } else {
              monto = beneficios.minus(costos);
            }

            flujo.push({
              anio: i,
              monto: monto.toFixed(DP),
              costos: costos.toFixed(DP),
              beneficios: beneficios.toFixed(DP)
            });
          } else {
            if (!porcetajeVariable.eq(0)) {
              costos = costos.plus(costos.times(porcetajeVariable));
              beneficios = beneficios.plus(beneficios.times(porcetajeVariable));
            }

            if (recalcular) {
              costos = new Big(proyecto.flujoCaja[i].costos);
              beneficios = new Big(proyecto.flujoCaja[i].beneficios);
            }

            flujo.push({
              anio: i,
              monto: beneficios.minus(costos).toFixed(DP),
              costos: costos.toFixed(DP),
              beneficios: beneficios.toFixed(DP)
            });
          }
        }
```

```
    return flujo;
  },

  TIR: (proyecto = null) => {
    if (R.isNil(proyecto)) {
      return new Big(0).toFixed(DP);
    }

    let sum = proyecto.flujoCaja.reduce((sum, flujo, n) => {
      let fn = new Big(flujo.monto);
      let divisor = new Big(1 + proyecto.inversion).pow(proyecto.vidaUtil);
      return sum.plus(fn.div(divisor));
    }, new Big(0));

    return sum.toFixed(DP);
  },

  puntajeTIR: (indicadores, TIR) => {
    let tir = new Big(TIR);
    let indicador = indicadores.find((item) => { return item.id == 1});
    let puntaje = indicador.valores.find((valores) => {
      let min = new Big(valores.min);
      let max = new Big(valores.max);
      return tir.gte(min) && tir.lte(max);
    });

    return puntaje ? puntaje.valor : 0;
  },

  VAN: (TIR = 0, proyecto = null) => {
    if (R.isNil(proyecto)) {
      return new Big(0).toFixed(DP);
    }

    let sum = proyecto.flujoCaja.reduce((sum, flujo, n) => {
      let fn = new Big(flujo.monto);
      let divisor = new Big(new Big(1).plus(TIR)).pow(proyecto.vidaUtil);
      return sum.plus(fn.div(divisor));
    }, new Big(0));

    return sum.minus(proyecto.vidaUtil).toFixed(DP);
  },

  puntajeVAN: (indicadores, VAN, inv) => {
    let van = new Big(VAN);
    let inversion = new Big(inv);
    let valor = van.div(inversion).times(100);
    let indicador = indicadores.find((item) => { return item.id == 2});
    let puntaje = indicador.valores.find((valores) => {
      let min = new Big(valores.min);
      let max = new Big(valores.max);
      return valor.gte(min) && valor.lte(max);
    });

    return puntaje ? puntaje.valor : 0;
  },

  puntajeTiempoRetornoInversion: (proyecto) => {
    let vidaUtil = new Big(proyecto.vidaUtil);
    let recuperacionInversion = vidaUtil;
    let inversionIni = new Big(proyecto.inversion);

    for (var i = 0; i < proyecto.flujoCaja.length - 1; i++) {
      if (i == 0) continue;

      let val = new Big(proyecto.flujoCaja[i].monto);
      if (val.gte(inversionIni)) {
        recuperacionInversion = new Big(i);
```

```
            break;
          }
          console.log(proyecto.flujoCaja[i + 1])
          let nexVal = new Big(proyecto.flujoCaja[i + 1].monto);
          if (val.plus(nexVal).gte(inversionIni)) {
            recuperacionInversion = new Big(i);
            break;
          }
        }

        return   new   Big(10).times(   new   Big(1).minus(   recuperacionInversion.div(vidaUtil)   )
).toFixed(DP);
      },

      puntajeBeneficiarios: (proyecto, proyectos) => {
        if (proyectos.length <= 1) {
          return false;
        }

        let inversion = new Big(proyecto.inversion);
        let beneficiarios = new Big(proyecto.beneficiarios);
        let promInvBeneficiarios = inversion.div(beneficiarios);

        let listPromInvBeneficiarios = proyectos.map((proyecto) => {
          let inv = new Big(proyecto.inversion);
          let benef = new Big(proyecto.beneficiarios);

          return inv.div(benef);
        });

        listPromInvBeneficiarios.sort((a, b) => {
          if (a.lt(b)) {
            return -1;
          }
          if (a.gt(b)) {
            return 1;
          }

          return 0;
        });

        let one = new Big(1);
        let maxPromInvBenef = listPromInvBeneficiarios[0];
        let minPromInvBenef = listPromInvBeneficiarios[listPromInvBeneficiarios.length - 1];

        return   new   Big(10).times(   one.minus(   promInvBeneficiarios.minus(   minPromInvBenef.div(
maxPromInvBenef.minus(minPromInvBenef) ) ) ) );
      },

      puntajeAreaPrioritaria: (proyecto, regla) => {
        let areaPrioritaria = regla.areasPrioritarias.find((area) => {
          return area.id == proyecto.areaPrioritaria.id;
        });

        return areaPrioritaria.puntaje;
      },

      sumaPuntajes: (puntajes) => {
        return //puntos, ranking
      }
    }
```

b. regex.js

```
    module.exports = {
```

```
      /*
       * Email RegExp
       */
      email:        /^(([^<>()\[\]\\.,;:\s@"]+(\.[^<>()\[\]\\.,;:\s@"]+)*)|(".+"))@((\[[0-9]{1,3}\.[0-
9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}])|(([a-zA-Z\-0-9]+\.)+[a-zA-Z]{2,}))$/,

      /*
       * Lowercase Alphanumeric RegExp
       * At least 1 number
       * At least 1 characters
       */
      atLeastOneAlphanumericLower: /^(?=.*[0-9])(?=.*[a-z])([a-z0-9]+)$/,

      /*
       * Lower and uppercase Alphanumeric RegExp
       * At least 1 characters
       * At least 1 number
       * Optional uppercase
       * Optional Lowercase
       */
      atLeastOneAlphanumeric: /^(?=.*[0-9])(?=.*[a-zA-Z])([a-zA-Z0-9]*)$/,

      /*
       * Alphanumeric RegExp
       * At least 1 lower
       * At least 1 uppercase
       * At least 1 number
       * At least 1 characters
       */
      atLeastOneAlphanumericLowerUpper: /^(?=.*[0-9])(?=.*[a-z])(?=.*[A-Z])([a-zA-Z0-9]+)$/,

      /*
       * Alphanumeric and special RegExp
       * At least 1 lower
       * At least 1 uppercase
       * At least 1 number
       * At least 1 characters
       * Allows: '_', '.', '-'
       * Doesn't allow '__', '--', '..' and any consecutive combination of these
       * Doesn't allow '_', '.', '-' as the first character
       * Doesn't allow '_', '.', '-' at the end
       * Does not allow any other special character different of the above
       */
      atLeastOneNumLowerAndUpperAllowSpecial:   /^(?![_.-])(?=.*[0-9])(?!.*?[._-]{2})(.*[A-Z])(?=.*[a-
z])(((?!__)(?!--)(?!\.-|-\.)(?!\._|_\.)(?!_-|-_)(?!.\.{2})[a-z0-9_.-])*[a-zA-Z0-9])?$/,

      /*
       * DUI RegExp
       * Match DUI (with hyphen)
       */
      dui: /^([0-9]{8}[-])([0-9]{1})$/
    }
```

5. Server

    a. Boot

        i. authentication.js

```
    'use strict';

    module.exports = function enableAuthentication(server) {
      // enable authentication
      server.enableAuth();
```

```
};
```

ii. create-roles.js

```
"use strict";

module.exports = (app) => {
  const Role = app.models.Role;
  let roles = [
    { "name": "admin", "description": "Administradores" },
    { "name": "user", "description": "Usuario Normal" },
    { "name": "root", "description": "Super usuario" }
  ]

  Role.create(roles, (err, role) => {
    if (err) return;
  });
};
```

iii. root.js

```
'use strict';

module.exports = function(server) {
  // Install a `/` route that returns server status
  var router = server.loopback.Router();
  router.get('/', server.loopback.status());
  server.use(router);
};
```

b. component-config.json

```
{
  "loopback-component-explorer": {
    "mountPath": "/explorer",
    "generateOperationScopedModels": true
  }
}
```

c. config.json

```
{
  "restApiRoot": "/api",
  "host": "0.0.0.0",
  "port": 3000,
  "remoting": {
    "context": false,
    "rest": {
      "handleErrors": false,
      "normalizeHttpPath": false,
      "xml": false
    },
    "json": {
      "strict": false,
      "limit": "100kb"
    },
    "urlencoded": {
```

```
        "extended": true,
        "limit": "100kb"
      },
      "cors": false
    }
  }
```

d. datasources.development.json

```
  {
    "mem": {
      "name": "mem",
      "connector": "memory"
    },
    "db": {
      "host": "localhost",
      "port": 27017,
      "database": "tesis",
      "password": "",
      "name": "db",
      "user": "",
      "connector": "mongodb"
    }
  }
```

e. datasources.json

```
  {
    "mem": {
      "name": "mem",
      "connector": "memory"
    },
    "db": {
      "name": "db",
      "url": "${MONGODB_URI}",
      "connector": "mongodb"
    }
  }
```

f. middleware.development.json

```
  {
    "final:after": {
      "strong-error-handler": {
        "params": {
          "debug": true,
          "log": true
        }
      }
    }
  }
```

g. middleware.json

```
  {
    "initial:before": {
      "loopback#favicon": {}
```

```
    },
    "initial": {
      "compression": {},
      "cors": {
        "params": {
          "origin": true,
          "credentials": true,
          "maxAge": 86400
        }
      },
      "helmet#xssFilter": {},
      "helmet#frameguard": {
        "params": [
          "deny"
        ]
      },
      "helmet#hsts": {
        "params": {
          "maxAge": 0,
          "includeSubdomains": true
        }
      },
      "helmet#hidePoweredBy": {},
      "helmet#ieNoOpen": {},
      "helmet#noSniff": {},
      "helmet#noCache": {
        "enabled": false
      }
    },
    "session": {},
    "auth": {},
    "parse": {},
    "routes": {
      "loopback#rest": {
        "paths": [
          "${restApiRoot}"
        ]
      }
    },
    "files": {},
    "final": {
      "loopback#urlNotFound": {}
    },
    "final:after": {
      "strong-error-handler": {}
    }
  }
```

h. model-config.json

```
  {
    "_meta": {
      "sources": [
        "loopback/common/models",
        "loopback/server/models",
        "../common/models",
        "./models"
      ],
      "mixins": [
        "loopback/common/mixins",
        "loopback/server/mixins",
        "../common/mixins",
        "./mixins"
      ]
    },
    "AccessToken": {
```

```
    "dataSource": "db",
    "public": false,
    "relations": {
      "user": {
        "type": "belongsTo",
        "model": "Users",
        "foreignKey": "userId"
      }
    }
  },
  "ACL": {
    "dataSource": "db",
    "public": false
  },
  "RoleMapping": {
    "dataSource": "db",
    "public": false,
    "options": {
      "strictObjectIDCoercion": true
    }
  },
  "Role": {
    "dataSource": "db",
    "public": false
  },
  "Users": {
    "dataSource": "db",
    "public": true
  },
  "Proyecto": {
    "dataSource": "db",
    "public": true
  },
  "Carpeta": {
    "dataSource": "db",
    "public": true
  },
  "Regla": {
    "dataSource": "db",
    "public": true
  },
  "Settings": {
    "dataSource": "db",
    "public": true
  },
  "Alcaldia": {
    "dataSource": "db",
    "public": true
  },
  "Evaluaciones": {
    "dataSource": "db",
    "public": true
  },
  "Calculos": {
    "dataSource": "db",
    "public": true
  }
}
```

i. server.js

```
'use strict';

const loopback = require('loopback');
const boot = require('loopback-boot');
```

```
const app = module.exports = loopback();

app.start = function() {
  // start the web server
  return app.listen(function() {
    app.emit('started');
    let baseUrl = app.get('url').replace(/\/$/, '');
    console.log('Web server listening at: %s', baseUrl);
    if (app.get('loopback-component-explorer')) {
      let explorerPath = app.get('loopback-component-explorer').mountPath;
      console.log('Browse your REST API at %s%s', baseUrl, explorerPath);
    }
  });
};

// Bootstrap the application, configure models, datasources and middleware.
// Sub-apps like REST API are mounted via boot scripts.
boot(app, __dirname, function(err) {
  if (err) throw err;

  // start the server if `$ node server.js`
  if (require.main === module)
    app.start();
});
```

6. package.json

```
{
  "name": "tesis-backend",
  "version": "1.0.0",
  "main": "server/server.js",
  "engines": {
    "node": ">=6 <=8"
  },
  "scripts": {
    "lint": "eslint .",
    "start": "NODE_ENV=production node server/server.js",
    "dev": "NODE_ENV=development DEBUG=loopback:connector:* nodemon server/server.js",
    "posttest": "npm run lint && nsp check"
  },
  "dependencies": {
    "big.js": "^5.2.2",
    "bluebird": "^3.5.1",
    "compression": "^1.0.3",
    "cors": "^2.5.2",
    "helmet": "^1.3.0",
    "loopback": "^3.0.0",
    "loopback-boot": "^2.6.5",
    "loopback-component-explorer": "^5.0.0",
    "loopback-connector-mongodb": "^3.3.0",
    "ramda": "^0.25.0",
    "serve-favicon": "^2.0.1",
    "strong-error-handler": "^2.0.0"
  },
  "devDependencies": {
    "eslint": "^3.17.1",
    "eslint-config-loopback": "^8.0.0",
    "nsp": "^2.1.0"
  },
  "repository": {
    "type": "",
    "url": ""
  },
  "license": "UNLICENSED",
  "description": "tesis-backend"
}
```

4.3.3 Front-end

La estructura de la aplicación del front-end es la presentada a continuación:

1. src
    a. app
        i. alcaldia
            1. create
                a. create.component.css

```
.form-container {
  display: flex;
  flex-direction: column;
  align-items: center;
}
.buttons-container {
  display: flex;
  flex-direction: row;
  align-items: center;
  margin-top: 100px;
}

.mat-form-field {
  width: 30%;
}

::ng-deep .mat-horizontal-stepper-header {
  pointer-events: none !important;
}
h2 {
  color: #3f51b5;
  text-align: center;
}
/*
  ##Device = Tablets, Ipads (portrait)
  ##Screen = B/w 768px to 1024px
*/

@media (min-width: 768px) and (max-width: 1024px) {
  .mat-form-field {
    width: 80%;
  }
}

/*
  ##Device = Tablets, Ipads (landscape)
  ##Screen = B/w 768px to 1024px
*/

@media (min-width: 768px) and (max-width: 1024px) and (orientation: landscape) {
  .mat-form-field {
    width: 80%;
  }
}
```

```
/*
  ##Device = Low Resolution Tablets, Mobiles (Landscape)
  ##Screen = B/w 481px to 767px
*/

@media (min-width: 481px) and (max-width: 767px) {
  .mat-form-field {
    width: 80%;
  }
}

/*
  ##Device = Most of the Smartphones Mobiles (Portrait)
  ##Screen = B/w 320px to 479px
*/

@media (min-width: 320px) and (max-width: 480px) {
  .mat-form-field {
    width: 80%;
  }
}
```

b. create.component.html

```
<h2>Crear Alcaldía</h2>
<form #f="ngForm" (ngSubmit)="onSubmit(f)">
  <ng-template>Datos generales</ng-template>
  <div ngModelGroup="generales" #generalesGroup="ngModelGroup" class="form-container">
    <mat-form-field>
      <input type="text" matInput ngModel placeholder="Nombre" name="nombre" nombre required
#nombreInput="ngModel">
      <mat-error>Este campo es requerido</mat-error>
    </mat-form-field>
    <mat-form-field>
      <textarea cols="30" rows="7" matInput placeholder="Descripcion" name="descripcion"
descripcion required ngModel
        #descripcionInput="ngModel"></textarea>
      <mat-error>Este campo es requerido</mat-error>
    </mat-form-field>
    <mat-form-field>
      <input type="text" matInput ngModel placeholder="Direccion" name="direccion" direccion
required #direccionInput="ngModel">
      <mat-error>Este campo es requerido</mat-error>
    </mat-form-field>
    <mat-form-field>
      <input type="text" matInput ngModel placeholder="Municipio" name="municipio" municipio
required #municipioInput="ngModel">
      <mat-error>Este campo es requerido</mat-error>
    </mat-form-field>
    <mat-form-field>
      <input type="text" matInput ngModel placeholder="Ciudad" name="ciudad" ciudad required
#ciudadInput="ngModel">
      <mat-error>Este campo es requerido</mat-error>
    </mat-form-field>
    <button mat-button type="submit" color="primary" mat-raised-button>Guardar</button>
  </div>
</form>
```

c. create.component.ts

```
import { Component, OnInit } from "@angular/core";
import { NgForm } from "@angular/forms";
```

```
import { AlcaldiaService, UIService } from "../../services";

import { Alcaldia } from "../../models";

import { Store } from "@ngrx/store";
import * as fromRoot from "../../store/app.reducers";
import * as UI from "../../store/ui/ui.actions";
import { Router } from "@angular/router";

@Component({
  selector: "app-create",
  templateUrl: "./create.component.html",
  styleUrls: ["./create.component.css"]
})
export class CreateComponent implements OnInit {
  alcaldiaArray: any = [];

  constructor(
    private alcaldiaService: AlcaldiaService,
    private uiService: UIService,
    private store: Store<fromRoot.State>,
    private router: Router
  ) {}

  ngOnInit() {}

  onSubmit(form: NgForm) {
    this.alcaldiaArray = [
      {
        nombre: form.value.generales.nombre,
        descripcion: form.value.generales.descripcion,
        direccion: form.value.generales.direccion,
        municipio: form.value.generales.municipio,
        ciudad: form.value.generales.ciudad,
        status: true
      }
    ];

    this.alcaldiaService.crear(this.alcaldiaArray).subscribe(
      data => {
        this.store.dispatch(new UI.StopLoading());
        this.uiService.showSnackbar(
          "Alcaldia creada",
          null,
          3000,
          "custom-snack-bar"
        );
        this.router.navigate(["/alcaldias/administrar"]);
      },
      error => {
        this.store.dispatch(new UI.StopLoading());
        this.uiService.showSnackbar(
          "Problemas creando alcaldia",
          null,
          3000,
          "custom-snack-bar-error"
        );
      }
    );
  }
}
```

2. list

   a. delete

i. delete.component.html

```
<h1 mat-dialog-title>Eliminando alcaldia: {{ passedData.alcaldia.nombre }}</h1>
<mat-dialog-content>
  <mat-list>
    <mat-list-item> Esta seguro que quiere eliminarla? </mat-list-item>
  </mat-list>
</mat-dialog-content>
<mat-dialog-actions  align="center">
  <button  *ngIf="!(isLoading$  |  async)"  type="button"  mat-raised-button  color="primary"
(click)="onSubmit()">Eliminar</button>
  <button mat-raised-button [mat-dialog-close]="false">Cancelar</button>
  <mat-spinner *ngIf="isLoading$ | async"></mat-spinner>
</mat-dialog-actions>
```

ii. delete.component.ts

```
import { Component, Inject, OnInit } from "@angular/core";
import { MAT_DIALOG_DATA, MatDialogRef } from "@angular/material";
import { Observable } from "rxjs";
import { AlcaldiaService, UIService, UsersService } from "../../../services";
import { User } from "../../../models";

import { Store } from "@ngrx/store";
import * as fromRoot from "../../../store/app.reducers";
import * as UI from "../../../store/ui/ui.actions";

@Component({
  selector: "app-delete-alcaldia",
  templateUrl: "delete.component.html",
  styleUrls: ["../list.component.css"]
})
export class DeleteComponent implements OnInit {
  isLoading$: Observable<boolean>;
  cargoSel: any;
  rolSel: any;
  alcaldiaArray: any = [];
  inactivateUserArr: any = [];
  constructor(
    public dialogRef: MatDialogRef<DeleteComponent>,
    @Inject(MAT_DIALOG_DATA) public passedData: any,
    private alcaldiaService: AlcaldiaService,
    private userService: UsersService,
    private uiService: UIService,
    private store: Store<fromRoot.State>
  ) {}

  ngOnInit() {
    this.isLoading$ = this.store.select(fromRoot.getIsLoading);
  }

  onSubmit() {
    this.alcaldiaArray = {
      status: false
    };
    this.alcaldiaService
      .updateAlcaldia(this.passedData.id, this.alcaldiaArray)
      .subscribe(
        data => {
          this.store.dispatch(new UI.StopLoading());
          this.uiService.showSnackbar(
            "Alcaldía eliminada!",
            null,
            3000,
            "custom-snack-bar"
```

```
            );
            this.dialogRef.close();
          },
          error => {
            this.inactivateUserArr = {
              status: false
            };
            this.userService
              .getUsersByAlcaldia(this.passedData.id)
              .subscribe((users: User[]) => {
                users.forEach(x => {
                  this.userService
                    .updateUser(x.id, this.inactivateUserArr)
                    .subscribe(data => {});
                });
              });
            this.store.dispatch(new UI.StopLoading());
            this.uiService.showSnackbar(
              "Problemas eliminando alcaldía",
              null,
              3000,
              "custom-snack-bar-error"
            );
            this.dialogRef.close();
          }
        );
    }
  }
```

b. edit

i. edit.component.html

```
  <h2 mat-dialog-title>Editando alcaldia:</h2>
  <form #f="ngForm" (ngSubmit)="onSubmit(f)">
    <mat-dialog-content>
      <ng-template>Datos generales</ng-template>
      <div ngModelGroup="generales" #generalesGroup="ngModelGroup" class="form-container">
        <mat-form-field>
          <input    type="text"    matInput    [(ngModel)]="passedData.alcaldia.nombre"    ngModel
placeholder="Nombre" name="nombre"
            nombre required #nombreInput="ngModel">
          <mat-error>Este campo es requerido</mat-error>
        </mat-form-field>
        <mat-form-field>
          <textarea  cols="30"  rows="7"  matInput  placeholder="Descripcion"  name="descripcion"
[(ngModel)]="passedData.alcaldia.descripcion"
            descripcion required ngModel #descripcionInput="ngModel"></textarea>
          <mat-error>Este campo es requerido</mat-error>
        </mat-form-field>
        <mat-form-field>
          <input type="text" matInput ngModel placeholder="Direccion" name="direccion" direccion
required #direccionInput="ngModel"
            [(ngModel)]="passedData.alcaldia.direccion">
          <mat-error>Este campo es requerido</mat-error>
        </mat-form-field>
        <mat-form-field>
          <input type="text" matInput ngModel placeholder="Municipio" name="municipio" municipio
required #municipioInput="ngModel"
            [(ngModel)]="passedData.alcaldia.municipio">
          <mat-error>Este campo es requerido</mat-error>
        </mat-form-field>
        <mat-form-field>
          <input type="text" matInput ngModel placeholder="Ciudad" name="ciudad" ciudad required
#ciudadInput="ngModel"
```

```
            [(ngModel)]="passedData.alcaldia.ciudad">
          <mat-error>Este campo es requerido</mat-error>
        </mat-form-field>
      </div>
    </mat-dialog-content>
    <mat-dialog-actions align="end">
      <button  *ngIf="!(isLoading$  |  async)"  type="submit"  mat-raised-button  color="primary"
[disabled]="f.invalid">Guardar</button>
      <button mat-raised-button [mat-dialog-close]="false">Cancelar</button>
      <mat-spinner *ngIf="isLoading$ | async"></mat-spinner>
    </mat-dialog-actions>
  </form>
```

ii. edit.component.ts

```
  import {
    Component,
    Inject,
    OnInit
  } from "@angular/core";
  import { NgForm, FormGroup, FormControl, Validators } from "@angular/forms";

  import {
    MAT_DIALOG_DATA,
    MatDialogRef
  } from "@angular/material";
  import { AlcaldiaService, UIService } from "../../../services";
  import { Observable } from "rxjs";
  import { Store } from "@ngrx/store";
  import * as fromRoot from "../../../store/app.reducers";
  import * as UI from "../../../store/ui/ui.actions";
  import { Router } from "@angular/router";

  @Component({
    selector: "app-edit-alcaldia",
    templateUrl: "edit.component.html",
    styleUrls: ["../list.component.css"]
  })
  export class EditComponent implements OnInit {
    alcaldiaArray: any = [];
    isLoading$: Observable<boolean>;

    disabled = false;

    public closeModal: boolean;

    constructor(
      public dialogRef: MatDialogRef<EditComponent>,
      @Inject(MAT_DIALOG_DATA) public passedData: any,
      private alcaldiaService: AlcaldiaService,
      private uiService: UIService,
      private store: Store<fromRoot.State>,
      private router: Router
    ) {}

    ngOnInit() {
      this.isLoading$ = this.store.select(fromRoot.getIsLoading);
    }
    onSubmit(form: NgForm) {
      this.alcaldiaArray = {
        nombre: form.value.generales.nombre,
        descripcion: form.value.generales.descripcion,
        direccion: form.value.generales.direccion,
        municipio: form.value.generales.municipio,
        ciudad: form.value.generales.ciudad,
        status: true
```

```
      };
      this.alcaldiaService
        .updateAlcaldia(this.passedData.id, this.alcaldiaArray)
        .subscribe(
          data => {
            console.log(data)
            this.store.dispatch(new UI.StopLoading());
            this.uiService.showSnackbar(
              "Alcaldia actualizada!",
              null,
              3000,
              "custom-snack-bar"
            );
            this.dialogRef.close();
            this.closeModal = true;
            this.router.navigate(["/alcaldias/administrar"]);
          },
          error => {
            this.store.dispatch(new UI.StopLoading());
            this.uiService.showSnackbar(
              "Problemas actualizando alcaldia",
              null,
              15000,
              "custom-snack-bar-error"
            );
            this.dialogRef.close();
          }
        );
    }
  }
```

c. list.component.css

```
.form-container {
  display: flex;
  flex-direction: column;
}

::ng-deep .mat-snack-bar-container {
  background-color: #36a5e6;
  color: #fff;
}

.mat-table {
  overflow: auto;
}
h2{
  color: #3f51b5;
  font-weight: bold;
  text-align: center;
}
.filter{
  margin-left: 15px;
}
```

d. list.component.html

```
<h2>Listado de alcaldías</h2>
<div fxLayoutAlign="center center" class="filter">
  <mat-form-field fxFlex="50%">
    <input
      matInput
      type="text"
```

```
          (keyup)="doFilter($event.target.value)"
          placeholder="Filter"
        />
      </mat-form-field>
    </div>
    <mat-table [dataSource]="dataSource" matSort>
      <ng-container matColumnDef="nombre">
        <mat-header-cell mat-header-cell *matHeaderCellDef>
          Nombre
        </mat-header-cell>
        <mat-cell *matCellDef="let element"> {{ element.nombre }} </mat-cell>
      </ng-container>

      <ng-container matColumnDef="descripcion">
        <mat-header-cell mat-header-cell *matHeaderCellDef>
          Descripcion
        </mat-header-cell>
        <mat-cell *matCellDef="let element"> {{ element.descripcion }} </mat-cell>
      </ng-container>

      <ng-container matColumnDef="actions">
        <mat-header-cell *matHeaderCellDef mat-sort-header>Actions</mat-header-cell>
        <mat-cell *matCellDef="let element">
          <button mat-icon-button (click)="onEdit(element.id)">
            <mat-icon>edit</mat-icon>
          </button>
          <button mat-icon-button (click)="onDelete(element.id)">
            <mat-icon>delete</mat-icon>
          </button>
        </mat-cell>
      </ng-container>
      <mat-header-row *matHeaderRowDef="displayedColumns"></mat-header-row>
      <mat-row *matRowDef="let row; columns: displayedColumns"></mat-row>
    </mat-table>
    <div *ngIf="notResults" align="center">No se encontraron resultados</div>

    <mat-paginator [pageSize]="5" [pageSizeOptions]="[5, 10, 20]"></mat-paginator>
    <mat-spinner
      *ngIf="(isLoading$ | async)"
      align="center"
      [diameter]="48"
      style="margin:0 auto;"
    ></mat-spinner>
```

e. list.component.ts

```
    import {
      Component,
      OnInit,
      ViewChild,
      AfterViewInit,
      ChangeDetectorRef
    } from "@angular/core";
    import { Observable } from "rxjs";
    import { toArray } from 'rxjs/operators';
    import { AlcaldiaService, AuthService, UIService } from "../../services";
    import {
      MatTableDataSource,
      MatSort,
      MatPaginator,
      MatDialog
    } from "@angular/material";
    import { Alcaldia, User } from "../../models";
    import { EditComponent } from "./edit/edit.component";
    import { DeleteComponent } from "./delete/delete.component";
    import { Store } from "@ngrx/store";
```

```
import * as fromRoot from "../../store/app.reducers";
import * as UI from "../../store/ui/ui.actions";

interface Reglas {
  id: any;
  nombre: string;
}

@Component({
  selector: "app-list",
  templateUrl: "./list.component.html",
  styleUrls: ["./list.component.css"]
})
export class ListComponent implements OnInit, AfterViewInit {
  alcaldia: Alcaldia;
  displayedColumns = ["nombre", "descripcion", "actions"];
  dataSource = new MatTableDataSource<Alcaldia>();
  currentUser: Observable<User>;
  idAlcaldia: String;
  rol: String;
  @ViewChild(MatSort) sort: MatSort;
  @ViewChild(MatPaginator) paginator: MatPaginator;
  isLoading$: Observable<boolean>;
  notResults = false;

  constructor(
    private alcaldiaService: AlcaldiaService,
    private authService: AuthService,
    private uiService: UIService,
    private dialog: MatDialog,
    private changeDetectorRefs: ChangeDetectorRef,
    private store: Store<fromRoot.State>
  ) {
    this.idAlcaldia = this.authService.currentUserValue.user.idAlcaldia;
    this.rol = this.authService.currentUserValue.user.rol;
  }

  ngOnInit() {
    this.isLoading$ = this.store.select(fromRoot.getIsLoading);
    this.onRefresh();
  }

  ngAfterViewInit() {
    this.dataSource.sort = this.sort;
    this.dataSource.paginator = this.paginator;
  }
  doFilter(filterValue: string) {
    this.dataSource.filter = filterValue.trim().toLowerCase();
  }

  onRefresh() {
    this.store.dispatch(new UI.StartLoading());
    if (this.rol == "root") {
      this.alcaldiaService
        .getAllAlcaldias()
        .pipe()
        .subscribe(
          (alcaldia: Alcaldia[]) => {
            this.dataSource.data = alcaldia;
            if(this.dataSource.data.length == 0){
              this.notResults = true;
            }
            this.store.dispatch(new UI.StopLoading());
          },
          error => {
            this.store.dispatch(new UI.StopLoading());
            this.uiService.showSnackbar(error, null, 3000, "custom-snack-bar-error");
          }
        );
```

```
      } else {
        this.alcaldiaService
          .getAlcaldia(this.idAlcaldia)
          .pipe()
          .subscribe(
            (alcaldia: Alcaldia[]) => {
              let arr = [];
              arr.push(alcaldia);
              this.dataSource.data = arr;
              if(this.dataSource.data.length == 0){
                this.notResults = true;
              }
              this.store.dispatch(new UI.StopLoading());
            },
            error => {
              this.store.dispatch(new UI.StopLoading());
              this.uiService.showSnackbar(error, null, 3000, "custom-snack-bar-error");
            }
          );
      }
    }
    onEdit(id) {
      this.alcaldiaService
        .getAlcaldia(id)
        .subscribe((alcaldiaData: Alcaldia[]) => {
          const dialogRef = this.dialog
            .open(EditComponent, {
              // height: '700px',
              width: "900px",
              data: {
                alcaldia: alcaldiaData,
                id: id
              }
            })
            .afterClosed()
            .subscribe(result => {
              this.onRefresh();
            });
        });
    }
    onDelete(id) {
      this.alcaldiaService
        .getAlcaldia(id)
        .subscribe((alcaldiaData: Alcaldia[]) => {
          const dialogRef = this.dialog
            .open(DeleteComponent, {
              data: {
                alcaldia: alcaldiaData,
                id: id
              }
            })
            .afterClosed()
            .subscribe(result => {
              this.onRefresh();
            });
        });
    }
  }
```

3. alcaldia.module

```
  import { NgModule } from '@angular/core';
  import { CommonModule } from '@angular/common';
  import { RouterModule, Routes } from '@angular/router';

  import { CreateComponent } from './create/create.component';
```

```
import { ListComponent } from './list/list.component';
import { EditComponent } from './list/edit/edit.component';
import { DeleteComponent } from './list/delete/delete.component';

import { AlcaldiaService } from '../services';
import { MaterialModule } from '../material.module';
import { ReactiveFormsModule, FormsModule } from '@angular/forms';

const routes: Routes = [
  { path: "crear", component: CreateComponent },
  { path: "administrar", component: ListComponent }
];

@NgModule({
  imports: [
    CommonModule,
    MaterialModule,
    ReactiveFormsModule,
    FormsModule,
    RouterModule.forChild(routes),

  ],
  declarations: [
    CreateComponent,
    ListComponent,
    EditComponent,
    DeleteComponent
  ],
  entryComponents: [
    EditComponent,
    DeleteComponent
  ],
  providers: [AlcaldiaService]
})
export class AlcaldiaModule { }
```

ii. carpeta

1. create

a. create.component.css

```
.form-container {
  display: flex;
  flex-direction: column;
  align-items: center;
}
.form-container-reglas {
  display: flex;
}
.buttons-container {
  display: flex;
  flex-direction: row;
  align-items: center;
  align-content: space-between;
  margin-top: 100px;
}
.buttons-container button {
    margin-right: 20px;
}
.mat-form-field {
  width: 100%;
}
```

```
.mat-table {
  overflow: auto;
}

.mat-column-select {
  overflow: visible;
}

::ng-deep .mat-horizontal-stepper-header {
  pointer-events: none !important;
}
h3{
  color: #3f51b5;
  font-weight: bold;
  text-align: center;
}
/*
  ##Device = Desktops
  ##Screen = 1281px to higher resolution desktops
*/

@media (min-width: 1281px) {
  .mat-form-field {
    width: 30%;
  }
}

/*
  ##Device = Laptops, Desktops
  ##Screen = B/w 1025px to 1280px
*/

@media (min-width: 1025px) and (max-width: 1280px) {
  .mat-form-field {
    width: 30%;
  }

}

/* results table */
table {
  width: 100%;
}

tr.detail-row-tr {
  height: 0;
}

tr.element-row:not(.expanded-row):hover {
  background: #f5f5f5;
}

tr.element-row:not(.expanded-row):active {
  background: #efefef;
}

.element-row td {
  border-bottom-width: 0;
}

.row-detail {
  overflow: hidden;
  display: flex;
}


/* /results table */
```

b. create.component.html

```
<h3>Crear Carpetas</h3>
<form #f="ngForm" (ngSubmit)="onSubmit(f)">
  <ng-template [ngIf]="mobileScreen" [ngIfElse]="desktopScreen">
    <mat-vertical-stepper linear #stepper="matVerticalStepper">
      <mat-step>
        <ng-template matStepLabel>Datos generales</ng-template>
        <div
          ngModelGroup="generales"
          #generalesGroup="ngModelGroup"
          class="form-container"
        >
          <mat-form-field *ngIf="(currentUser$ | async)?.user.rol == 'root'">
            <mat-select
              matInput
              ngModel
              placeholder="Alcaldia"
              idAlcaldia
              name="idAlcaldia"
              required
              #idAlcaldiaInput="ngModel"
            >
              <mat-option>Seleccione una opcion</mat-option>
              <mat-option
                *ngFor="let alcaldia of alcaldias; let in = index"
                [value]="alcaldia.id"
              >
                {{ alcaldia.nombre }}
              </mat-option>
            </mat-select>
            <mat-error>Este campo es requerido</mat-error>
          </mat-form-field>
          <mat-form-field>
            <input
              type="text"
              matInput
              ngModel
              placeholder="Nombre"
              name="nombre"
              nombre
              required
              #nombreInput="ngModel"
            />
            <mat-error>Este campo es requerido</mat-error>
          </mat-form-field>
          <mat-form-field>
            <textarea
              cols="30"
              rows="7"
              matInput
              placeholder="Descripcion"
              name="descripcion"
              descripcion
              required
              ngModel
              #descripcionInput="ngModel"
            ></textarea>
            <mat-error>Este campo es requerido</mat-error>
          </mat-form-field>
        </div>
        <div>
          <button
            mat-button
            matStepperNext
            type="button"
            color="primary"
            mat-raised-button
```

```
            [disabled]="!generalesGroup.valid"
          >
            Siguiente
          </button>
        </div>
      </mat-step>
      <mat-step>
        <ng-template matStepLabel>Reglas</ng-template>
        <div
          ngModelGroup="reglas"
          #reglasGroup="ngModelGroup"
          class="form-container-reglas"
          fxLayoutGap="10px"
        >
          <mat-form-field>
            <mat-select
              matInput
              ngModel
              placeholder="Reglas"
              regla
              name="regla"
              required
              #reglaInput="ngModel"
            >
              <mat-option>Seleccione una opción</mat-option>
              <mat-option
                *ngFor="let regla of reglas2; let in = index"
                [value]="regla.id"
              >
                {{ regla.nombre }}
              </mat-option>
            </mat-select>
          </mat-form-field>
          <div style="min-height:200px"></div>
        </div>
        <div>
          <button mat-button matStepperPrevious type="button" mat-raised-button>
            Atras
          </button>
          <button
            mat-button
            matStepperNext
            type="button"
            color="primary"
            mat-raised-button
            [disabled]="!reglasGroup.valid"
          >
            Siguiente
          </button>
        </div>
      </mat-step>
      <!-- <mat-step formGroupName="2" [stepControl]="formArray?.get([2])"> -->
      <mat-step>
        <ng-template matStepLabel>Proyectos</ng-template>

        <div fxLayoutAlign="center center">
          <mat-form-field fxFlex="40%">
            <input
              matInput
              type="text"
              (keyup)="doFilter($event.target.value)"
              placeholder="Filter"
            />
          </mat-form-field>
        </div>

        <mat-table [dataSource]="dataSource" matSort>
          <!-- Checkbox Column -->
          <ng-container matColumnDef="select">
```

```
              <mat-header-cell *matHeaderCellDef>
                <mat-checkbox
                  (change)="$event ? masterToggle() : null"
                  [checked]="selection.hasValue() && isAllSelected()"
                  [indeterminate]="selection.hasValue() && !isAllSelected()"
                >
                </mat-checkbox>
              </mat-header-cell>
              <mat-cell *matCellDef="let row">
                <mat-checkbox
                  (click)="$event.stopPropagation()"
                  (change)="$event ? selection.toggle(row) : null"
                  [checked]="selection.isSelected(row)"
                  [value]="row"
                >
                </mat-checkbox>
              </mat-cell>
            </ng-container>

            <ng-container matColumnDef="nombre">
              <mat-header-cell *matHeaderCellDef mat-sort-header
                >Nombre</mat-header-cell
              >
              <mat-cell *matCellDef="let element">{{ element.nombre }}</mat-cell>
            </ng-container>

            <mat-header-row *matHeaderRowDef="displayedColumns"></mat-header-row>
            <mat-row
              *matRowDef="let row; columns: displayedColumns"
              (click)="selection.toggle(row)"
            ></mat-row>
          </mat-table>

          <mat-paginator
            [pageSize]="5"
            [pageSizeOptions]="[5, 10, 20]"
          ></mat-paginator>

          <div>
            <button mat-button matStepperPrevious type="button" mat-raised-button>
              Atras
            </button>
            <button
              mat-button
              matStepperNext
              type="button"
              color="primary"
              mat-raised-button
            >
              Siguiente
            </button>
          </div>
        </mat-step>
        <mat-step>
          <ng-template matStepLabel>Finalizar</ng-template>
          <div class="buttons-container" fxLayoutGap="20px">
            <button mat-button matStepperPrevious type="button" mat-raised-button>
              Atras
            </button>
            <!-- <button mat-button (click)="stepper.reset()" mat-raised-button>Reset</button> -->
            <button
              mat-button
              matStepperNext
              type="submit"
              color="primary"
              mat-raised-button
            >
              Guardar
            </button>
```

```
            <button mat-button color="primary" mat-raised-button>Evaluar</button>
          </div>
        </mat-step>
      </mat-vertical-stepper>
    </ng-template>
    <ng-template #desktopScreen>
      <mat-horizontal-stepper linear #stepper="matHorizontalStepper">
        <mat-step>
          <ng-template matStepLabel>Datos generales</ng-template>
          <div
            ngModelGroup="generales"
            #generalesGroup="ngModelGroup"
            class="form-container"
          >
            <mat-form-field *ngIf="(currentUser$ | async)?.user.rol == 'root'">
              <mat-select
                matInput
                ngModel
                placeholder="Alcaldia"
                idAlcaldia
                name="idAlcaldia"
                required
                #idAlcaldiaInput="ngModel"
              >
                <mat-option>Seleccione una opcion</mat-option>
                <mat-option
                  *ngFor="let alcaldia of alcaldias; let in = index"
                  [value]="alcaldia.id"
                >
                  {{ alcaldia.nombre }}
                </mat-option>
              </mat-select>
              <mat-error>Este campo es requerido</mat-error>
            </mat-form-field>
            <mat-form-field>
              <input
                type="text"
                matInput
                ngModel
                placeholder="Nombre"
                name="nombre"
                nombre
                required
                #nombreInput="ngModel"
              />
              <mat-error>Este campo es requerido</mat-error>
            </mat-form-field>
            <mat-form-field>
              <textarea
                cols="30"
                rows="7"
                matInput
                placeholder="Descripcion"
                name="descripcion"
                descripcion
                required
                ngModel
                #descripcionInput="ngModel"
              ></textarea>
              <mat-error>Este campo es requerido</mat-error>
            </mat-form-field>
          </div>
          <div>
            <button
              mat-button
              matStepperNext
              type="button"
              color="primary"
              mat-raised-button
```

```
            [disabled]="!generalesGroup.valid"
          >
            Siguiente
          </button>
        </div>
      </mat-step>
      <mat-step>
        <ng-template matStepLabel>Reglas</ng-template>
        <div
          ngModelGroup="reglas"
          #reglasGroup="ngModelGroup"
          class="form-container"
          fxLayout="column"
          fxLayoutAlign="center center"
          fxLayoutGap="10px"
        >
          <mat-form-field>
            <mat-select
              matInput
              ngModel
              placeholder="Reglas"
              regla
              name="regla"
              required
              #reglaInput="ngModel"
            >
              <mat-option>Seleccione una opción</mat-option>
              <mat-option
                *ngFor="let regla of reglas2; let in = index"
                [value]="regla.id"
              >
                {{ regla.nombre }}
              </mat-option>
            </mat-select>
          </mat-form-field>
          <div style="min-height:200px"></div>
        </div>
        <div>
          <button mat-button matStepperPrevious type="button" mat-raised-button>
            Atras
          </button>
          <button
            mat-button
            matStepperNext
            type="button"
            color="primary"
            mat-raised-button
            [disabled]="!reglasGroup.valid"
          >
            Siguiente
          </button>
        </div>
      </mat-step>
      <!-- <mat-step formGroupName="2" [stepControl]="formArray?.get([2])"> -->
      <mat-step>
        <ng-template matStepLabel>Proyectos</ng-template>

        <div fxLayoutAlign="center center">
          <mat-form-field fxFlex="40%">
            <input
              matInput
              type="text"
              (keyup)="doFilter($event.target.value)"
              placeholder="Filter"
            />
          </mat-form-field>
        </div>

        <mat-table [dataSource]="dataSource" matSort>
```

```
            <!-- Checkbox Column -->
            <ng-container matColumnDef="select">
              <mat-header-cell *matHeaderCellDef>
                <mat-checkbox
                  (change)="$event ? masterToggle() : null"
                  [checked]="selection.hasValue() && isAllSelected()"
                  [indeterminate]="selection.hasValue() && !isAllSelected()"
                >
                </mat-checkbox>
              </mat-header-cell>
              <mat-cell *matCellDef="let row">
                <mat-checkbox
                  (click)="$event.stopPropagation()"
                  (change)="$event ? selection.toggle(row) : null"
                  [checked]="selection.isSelected(row)"
                  [value]="row"
                >
                </mat-checkbox>
              </mat-cell>
            </ng-container>

            <ng-container matColumnDef="nombre">
              <mat-header-cell *matHeaderCellDef mat-sort-header
                >Nombre</mat-header-cell
              >
              <mat-cell *matCellDef="let element">{{ element.nombre }}</mat-cell>
            </ng-container>

            <mat-header-row *matHeaderRowDef="displayedColumns"></mat-header-row>
            <mat-row
              *matRowDef="let row; columns: displayedColumns"
              (click)="selection.toggle(row)"
            ></mat-row>
          </mat-table>

          <mat-paginator
            [pageSize]="5"
            [pageSizeOptions]="[5, 10, 20]"
          ></mat-paginator>

          <div>
            <button mat-button matStepperPrevious type="button" mat-raised-button>
              Atras
            </button>
            <button
              mat-button
              matStepperNext
              type="button"
              color="primary"
              mat-raised-button
            >
              Siguiente
            </button>
          </div>
        </mat-step>
        <mat-step>
          <ng-template matStepLabel>Finalizar</ng-template>
          <div class="buttons-container" fxLayoutGap="20px">
            <button mat-button matStepperPrevious type="button" mat-raised-button>
              Atras
            </button>
            <!-- <button mat-button (click)="stepper.reset()" mat-raised-button>Reset</button> -->
            <button
              mat-button
              matStepperNext
              type="submit"
              color="primary"
              mat-raised-button
            >
```

```
              Guardar
            </button>
            <button mat-button color="primary" mat-raised-button>Evaluar</button>
          </div>
        </mat-step>
      </mat-horizontal-stepper>
    </ng-template>
  </form>
```

c. create.component.ts

```
import { Component, OnInit, ViewChild, AfterViewInit } from "@angular/core";
import { NgForm } from "@angular/forms";
import { MatTableDataSource, MatSort, MatPaginator } from "@angular/material";
import { Observable } from "rxjs";
import { Store } from "@ngrx/store";
import { Router, ActivatedRoute } from "@angular/router";

import {
  ReglaService,
  UIService,
  CarpetaService,
  ProyectoService,
  AlcaldiaService,
  AuthService
} from "../../services";
import { Proyecto, Regla, Alcaldia, Auth } from "../../models";
import * as fromRoot from "../../store/app.reducers";
import * as UI from "../../store/ui//ui.actions";
import { SelectionModel } from "@angular/cdk/collections";

@Component({
  selector: "app-create",
  templateUrl: "./create.component.html",
  styleUrls: ["./create.component.css"]
})
export class CreateComponent implements OnInit, AfterViewInit {
  displayedColumns = ["select", "nombre"];
  carpetaArray: any = [];
  dataSource = new MatTableDataSource<Proyecto>();
  selection = new SelectionModel<Proyecto>(true, []);
  /** list of reglas */
  public reglas: Regla[] = [];
  public reglas2: Regla[] = [];
  public proyectos: Proyecto[] = [];

  @ViewChild(MatSort)
  sort: MatSort;
  @ViewChild(MatPaginator)
  paginator: MatPaginator;

  proyectoSel: any[] = [];
  reglasData: any[] = [];
  reglasListData: any[] = [];

  disabled = false;
  ShowFilter = false;
  limitSelection = false;
  selectedItems: any = [];
  dropdownSettings: any = {};
  mobileScreen = false;
  desktopScreen = true;
  mHeight: any;
  mWidth: any;
  public alcaldias: Alcaldia[] = [];
  currentUser$: Observable<Auth>;
```

```
  idAlcaldia: string;

  constructor(
    private carpetaService: CarpetaService,
    private proyectoService: ProyectoService,
    private reglaService: ReglaService,
    private uiService: UIService,
    private store: Store<fromRoot.State>,
    private alcaldiaService: AlcaldiaService,
    private authService: AuthService,
    private route: ActivatedRoute,
    private router: Router,
  ) {
    this.currentUser$ = this.store.select(fromRoot.getCurrentUser);
    this.getAlcaldias();
    this.getReglas();
    this.mHeight = window.screen.height;
    this.mWidth = window.screen.width;
    this.desktopScreen = false;
    if (this.mWidth <= 700 || (this.mWidth == 768 && this.mHeight == 1024)) {
      this.mobileScreen = true;
    }
    this.idAlcaldia = this.authService.currentUserValue.user.idAlcaldia;
  }

  ngOnInit() {
    this.getReglas();
    this.getProyectos();
  }

  ngAfterViewInit() {
    this.dataSource.sort = this.sort;
    this.dataSource.paginator = this.paginator;
  }

  doFilter(filterValue: string) {
    this.dataSource.filter = filterValue.trim().toLowerCase();
  }

  /** Whether the number of selected elements matches the total number of rows. */
  isAllSelected() {
    const numSelected = this.selection.selected.length;
    const numRows = this.dataSource.data.length;
    return numSelected === numRows;
  }

  /** Selects all rows if they are not all selected; otherwise clear selection. */
  masterToggle() {
    this.isAllSelected()
      ? this.selection.clear()
      : this.dataSource.data.forEach(row => this.selection.select(row));
  }

  onSubmit(form: NgForm) {
    let alcaldia = form.value.generales.idAlcaldia;
    if(!alcaldia){
       alcaldia =  this.idAlcaldia
    }
    this.carpetaArray = [
      {
        idAlcaldia: alcaldia,
        nombre: form.value.generales.nombre,
        descripcion: form.value.generales.descripcion,
        regla: this.reglas2.filter(
            x => x.id == form.value.reglas.regla
          )[0],
        proyectos: this.selection.selected,
        status: true
      }
```

```
      ];
      this.carpetaService.crear(this.carpetaArray).subscribe(
        data => {
          this.store.dispatch(new UI.StopLoading());
          this.uiService.showSnackbar(
            "Carpeta creada",
            null,
            3000,
            "custom-snack-bar"
          );
          this.router.navigate(["/carpetas/administrar"]);
        },
        error => {
          this.store.dispatch(new UI.StopLoading());
          console.log('error', error)
          this.uiService.showSnackbar(
            "Problemas creando carpeta",
            null,
            3000,
            "custom-snack-bar-error"
          );
        }
      );
    }

    getReglas() {
      return this.reglaService.getAllReglas().subscribe((data: Regla[]) => {
        this.reglas2 = data;
      });
    }
    getProyectos() {
      return this.proyectoService
        .getAllProyectos()
        .subscribe((proyectoData: Proyecto[]) => {
          this.proyectos = proyectoData;
          this.dataSource.data = this.proyectos;
        });
    }
    getAlcaldias() {
      return this.alcaldiaService
        .getAllAlcaldias()
        .subscribe((data: Alcaldia[]) => {
          this.alcaldias = data;
        });
    }
  }
```

2. list

a. delete

i. delete.component.html

```
  <h1 mat-dialog-title>Eliminando carpeta: {{ passedData.carpeta.nombre }}</h1>
  <mat-dialog-content>
    <mat-list>
      <mat-list-item> Esta seguro que quiere eliminarla? </mat-list-item>
    </mat-list>
  </mat-dialog-content>
  <mat-dialog-actions>
    <button *ngIf="!(isLoading$ | async)" type="button" mat-raised-button color="primary"
(click)="onSubmit()">Delete</button>
    <button mat-button [mat-dialog-close]="false">Cancelar</button>
    <mat-spinner *ngIf="isLoading$ | async"></mat-spinner>
```

```
    </mat-dialog-actions>
```

ii. delete.component.ts

```
import { Component, Inject, OnInit } from "@angular/core";
import { MAT_DIALOG_DATA, MatDialogRef } from "@angular/material";
import { CarpetaService, UIService } from "../../../services";
import { Observable } from "rxjs";
import { Store } from "@ngrx/store";

import * as fromRoot from "../../../store/app.reducers";
import * as UI from "../../../store/ui/ui.actions";

@Component({
  selector: "app-delete-carpeta",
  templateUrl: "delete.component.html",
  styleUrls: ["../list.component.css"]
})
export class DeleteCarpetaComponent implements OnInit {
  isLoading$: Observable<boolean>;
  cargoSel: any;
  rolSel: any;
  constructor(
    public dialogRef: MatDialogRef<DeleteCarpetaComponent>,
    @Inject(MAT_DIALOG_DATA) public passedData: any,
    private carpetaService: CarpetaService,
    private uiService: UIService,
    private store: Store<fromRoot.State>
  ) {}

  ngOnInit() {
    this.isLoading$ = this.store.select(fromRoot.getIsLoading);
  }

  onSubmit() {
    this.carpetaService.deleteCarpeta(this.passedData.id).subscribe(
      data => {
        this.store.dispatch(new UI.StopLoading());
        this.uiService.showSnackbar(
          "Problemas eliminando carpeta",
          null,
          3000,
          "custom-snack-bar"
        );
        this.dialogRef.close();
      },
      error => {
        this.store.dispatch(new UI.StopLoading());
        this.uiService.showSnackbar(
          "Problemas eliminando carpeta",
          null,
          3000,
          "custom-snack-bar-error"
        );
        this.dialogRef.close();
      }
    );
  }
}
```

b. edit

i. edit.component.html

```
<h3 *ngIf="!(isLoading$ | async)">
  <a routerLink="/configuracion/listar" title="Atrás"
    ><mat-icon>arrow_back</mat-icon></a
  >
  Editando carpeta: {{ detailData.nombre }}
</h3>
<form #f="ngForm" (ngSubmit)="onSubmit(f)" *ngIf="!(isLoading$ | async)">
  <ng-template [ngIf]="mobileScreen" [ngIfElse]="desktopScreen">
    <mat-vertical-stepper linear #stepper="matVerticalStepper">
      <mat-step>
        <ng-template matStepLabel>Datos generales</ng-template>
        <div
          ngModelGroup="generales"
          #generalesGroup="ngModelGroup"
          class="form-container"
        >
          <mat-form-field>
            <input
              type="text"
              matInput
              [(ngModel)]="detailData.nombre"
              ngModel
              placeholder="Nombre"
              name="nombre"
              nombre
              required
              #nombreInput="ngModel"
            />
            <mat-error>Este campo es requerido</mat-error>
          </mat-form-field>
          <mat-form-field>
            <textarea
              cols="30"
              rows="7"
              matInput
              placeholder="Descripcion"
              name="descripcion"
              [(ngModel)]="detailData.descripcion"
              descripcion
              required
              ngModel
              #descripcionInput="ngModel"
            ></textarea>
            <mat-error>Este campo es requerido</mat-error>
          </mat-form-field>
        </div>
        <div>
          <button
            mat-button
            matStepperNext
            type="button"
            color="primary"
            mat-raised-button
            [disabled]="!generalesGroup.valid"
          >
            Siguiente
          </button>
        </div>
      </mat-step>
      <mat-step>
        <ng-template matStepLabel>Reglas</ng-template>
        <div
          ngModelGroup="reglas"
          #reglasGroup="ngModelGroup"
          class="form-container"
          fxLayout="column"
          fxLayoutAlign="center center"
          fxLayoutGap="10px"
        >
```

```
            <mat-form-field>
              <mat-select
                matInput
                ngModel
                placeholder="Reglas"
                regla
                name="regla"
                required
                #reglaInput="ngModel"
                [(ngModel)]="detailData.regla.id"
              >
                <mat-option>Seleccione una opción</mat-option>
                <mat-option
                  *ngFor="let regla of reglas; let in = index"
                  [value]="regla.id"
                >
                  {{ regla.nombre }}
                </mat-option>
              </mat-select>
            </mat-form-field>
            <div style="min-height:200px"></div>
          </div>
          <div>
            <button mat-button matStepperPrevious type="button" mat-raised-button>
              Atras
            </button>
            <button
              mat-button
              matStepperNext
              type="button"
              color="primary"
              mat-raised-button
              [disabled]="!reglasGroup.valid"
            >
              Siguiente
            </button>
          </div>
        </mat-step>
        <!-- <mat-step formGroupName="2" [stepControl]="formArray?.get([2])"> -->
        <mat-step>
          <ng-template matStepLabel>Proyectos</ng-template>

          <div fxLayoutAlign="center center">
            <mat-form-field fxFlex="40%">
              <input
                matInput
                type="text"
                (keyup)="doFilter($event.target.value)"
                placeholder="Filter"
              />
            </mat-form-field>
          </div>

          <mat-table [dataSource]="dataSource" matSort>
            <!-- Checkbox Column -->
            <ng-container matColumnDef="select">
              <mat-header-cell *matHeaderCellDef>
                <mat-checkbox
                  (change)="$event ? masterToggle() : null"
                  [checked]="selection.hasValue() && isAllSelected()"
                  [indeterminate]="selection.hasValue() && !isAllSelected()"
                >
                </mat-checkbox>
              </mat-header-cell>
              <mat-cell *matCellDef="let row">
                <mat-checkbox
                  (click)="$event.stopPropagation()"
                  (change)="$event ? selection.toggle(row) : null"
                  [checked]="row.isSelected"
```

```
                    [value]="row"
                  >
                  </mat-checkbox>
                </mat-cell>
              </ng-container>

              <ng-container matColumnDef="nombre">
                <mat-header-cell *matHeaderCellDef mat-sort-header
                  >Nombre</mat-header-cell
                >
                <mat-cell *matCellDef="let element">{{ element.nombre }}</mat-cell>
              </ng-container>

              <mat-header-row *matHeaderRowDef="displayedColumns"></mat-header-row>
              <mat-row
                *matRowDef="let row; columns: displayedColumns"
                (click)="selection.toggle(row)"
              ></mat-row>
            </mat-table>

            <mat-paginator
              [pageSize]="5"
              [pageSizeOptions]="[5, 10, 20]"
            ></mat-paginator>

            <div>
              <button mat-button matStepperPrevious type="button" mat-raised-button>
                Atras
              </button>
              <button
                mat-button
                matStepperNext
                type="button"
                color="primary"
                mat-raised-button
                (click)="evaluar()"
              >
                Evaluar y Continuar
              </button>
            </div>
          </mat-step>
          <mat-step>
            <ng-template matStepLabel>Finalizar</ng-template>
            <!-- results table -->
            <table
              mat-table
              [dataSource]="dataSource_"
              multiTemplateDataRows
              class="mat-elevation-z8"
            >
              <ng-container
                matColumnDef="{{ column }}"
                *ngFor="let column of columnsToDisplay"
              >
                <th mat-header-cell *matHeaderCellDef>{{ column }}</th>
                <td mat-cell *matCellDef="let element">{{ element[column] }}</td>
              </ng-container>

              <!--
                      Expanded Content Column - The detail row is made up of this one column that spans
across all columns
                    -->
              <ng-container matColumnDef="expandedDetail">
                <td
                  mat-cell
                  *matCellDef="let element"
                  [attr.colspan]="columnsToDisplay.length"
                >
                  <div
```

```
                class="row-detail"
                [@detailExpand]="
                  element == expandedElement ? 'expanded' : 'collapsed'
                "
              >
                <table>
                  <thead style="text-align: left; border:1px solid #cccc">
                    <tr>
                      <th>Indicador</th>
                      <th>Puntaje</th>
                    </tr>
                  </thead>
                  <tbody>
                    <tr
                      *ngFor="
                        let puntaje of element.tablaPuntajes;
                        let in = index
                      "
                    >
                      <td>{{ puntaje.indicador }}</td>
                      <td>{{ puntaje.puntaje }}</td>
                    </tr>
                    <tr style="border:1px solid #cccc">
                      <td style="text-align:right; font-weight: bold">
                        Total:
                      </td>
                      <td>100</td>
                    </tr>
                  </tbody>
                </table>
              </div>
            </td>
          </ng-container>

          <tr mat-header-row *matHeaderRowDef="columnsToDisplay"></tr>
          <tr
            mat-row
            *matRowDef="let element; columns: columnsToDisplay"
            class="element-row"
            [class.expanded-row]="expandedElement === element"
            (click)="expandedElement = element"
          ></tr>
          <tr
            mat-row
            *matRowDef="let row; columns: ['expandedDetail']"
            class="detail-row-tr"
          ></tr>
        </table>

        <!-- /results table -->
        <div class="buttons-container" fxLayoutGap="20px">
          <button mat-button matStepperPrevious type="button" mat-raised-button>
            Atras
          </button>
          <!--
                  <button mat-button (click)="stepper.reset()" mat-raised-button>Reset</button>
                -->
          <button
            mat-button
            [mat-dialog-close]="closeModal"
            matStepperNext
            type="submit"
            color="primary"
            mat-raised-button
          >
            Guardar
          </button>
          <button mat-button color="primary" mat-raised-button>Evaluar</button>
        </div>
```

```
        </mat-step>
      </mat-vertical-stepper>
    </ng-template>
    <ng-template #desktopScreen>
      <mat-horizontal-stepper linear #stepper="matHorizontalStepper">
        <mat-step>
          <ng-template matStepLabel>Datos generales</ng-template>
          <div
            ngModelGroup="generales"
            #generalesGroup="ngModelGroup"
            class="form-container"
          >
            <mat-form-field>
              <input
                type="text"
                matInput
                [(ngModel)]="detailData.nombre"
                ngModel
                placeholder="Nombre"
                name="nombre"
                nombre
                required
                #nombreInput="ngModel"
              />
              <mat-error>Este campo es requerido</mat-error>
            </mat-form-field>
            <mat-form-field>
              <textarea
                cols="30"
                rows="7"
                matInput
                placeholder="Descripcion"
                name="descripcion"
                [(ngModel)]="detailData.descripcion"
                descripcion
                required
                ngModel
                #descripcionInput="ngModel"
              ></textarea>
              <mat-error>Este campo es requerido</mat-error>
            </mat-form-field>
          </div>
          <div>
            <button
              mat-button
              matStepperNext
              type="button"
              color="primary"
              mat-raised-button
              [disabled]="!generalesGroup.valid"
            >
              Siguiente
            </button>
          </div>
        </mat-step>
        <mat-step>
          <ng-template matStepLabel>Reglas</ng-template>
          <div
            ngModelGroup="reglas"
            #reglasGroup="ngModelGroup"
            class="form-container"
            fxLayout="column"
            fxLayoutAlign="center center"
            fxLayoutGap="10px"
          >
            <mat-form-field>
              <mat-select
                matInput
                ngModel
```

```
                placeholder="Reglas"
                regla
                name="regla"
                required
                #reglaInput="ngModel"
                [(ngModel)]="detailData.regla.id"
              >
                <mat-option>Seleccione una opción</mat-option>
                <mat-option
                  *ngFor="let regla of reglas; let in = index"
                  [value]="regla.id"
                >
                  {{ regla.nombre }}
                </mat-option>
              </mat-select>
            </mat-form-field>
            <div style="min-height:200px"></div>
          </div>
          <div>
            <button mat-button matStepperPrevious type="button" mat-raised-button>
              Atras
            </button>
            <button
              mat-button
              matStepperNext
              type="button"
              color="primary"
              mat-raised-button
              [disabled]="!reglasGroup.valid"
            >
              Siguiente
            </button>
          </div>
        </mat-step>
        <!-- <mat-step formGroupName="2" [stepControl]="formArray?.get([2])"> -->
        <mat-step>
          <ng-template matStepLabel>Proyectos</ng-template>

          <div fxLayoutAlign="center center">
            <mat-form-field fxFlex="40%">
              <input
                matInput
                type="text"
                (keyup)="doFilter($event.target.value)"
                placeholder="Filter"
              />
            </mat-form-field>
          </div>

          <mat-table [dataSource]="dataSource" matSort>
            <!-- Checkbox Column -->
            <ng-container matColumnDef="select">
              <mat-header-cell *matHeaderCellDef>
                <mat-checkbox
                  (change)="$event ? masterToggle() : null"
                  [checked]="selection.hasValue() && isAllSelected()"
                  [indeterminate]="selection.hasValue() && !isAllSelected()"
                >
                </mat-checkbox>
              </mat-header-cell>
              <mat-cell *matCellDef="let row">
                <mat-checkbox
                  (click)="$event.stopPropagation()"
                  (change)="$event ? selection.toggle(row) : null"
                  [checked]="row.isSelected"
                  [value]="row"
                >
                </mat-checkbox>
              </mat-cell>
```

```
              </ng-container>

              <ng-container matColumnDef="nombre">
                <mat-header-cell *matHeaderCellDef mat-sort-header
                  >Nombre</mat-header-cell
                >
                <mat-cell *matCellDef="let element">{{ element.nombre }}</mat-cell>
              </ng-container>

              <mat-header-row *matHeaderRowDef="displayedColumns"></mat-header-row>
              <mat-row
                *matRowDef="let row; columns: displayedColumns"
                (click)="selection.toggle(row)"
              ></mat-row>
            </mat-table>

            <mat-paginator
              [pageSize]="5"
              [pageSizeOptions]="[5, 10, 20]"
            ></mat-paginator>

            <div class="buttons-container"  fxLayoutGap="20px">
              <button mat-button matStepperPrevious type="button" mat-raised-button>
                Atras
              </button>
              <button
                mat-button
                type="button"
                color="primary"
                mat-raised-button
                (click)="evaluar()"
                matStepperNext
              >
                Evaluar y continuar
              </button>

            </div>
          </mat-step>
          <mat-step>
            <ng-template matStepLabel>Finalizar</ng-template>
            <!-- results table -->
            <!-- {{evaluaciones |json}} -->
            <table
              mat-table
              [dataSource]="dataSource_"
              multiTemplateDataRows
              class="mat-elevation-z8"
            >
              <ng-container
                matColumnDef="{{ column }}"
                *ngFor="let column of columnsToDisplay"
              >
                <th mat-header-cell *matHeaderCellDef>{{ column }}</th>
                <td mat-cell *matCellDef="let element">{{ element[column] }}</td>
              </ng-container>

              <!--
                      Expanded Content Column - The detail row is made up of this one column that spans
across all columns
                    -->
              <ng-container matColumnDef="expandedDetail">
                <td
                  mat-cell
                  *matCellDef="let element"
                  [attr.colspan]="columnsToDisplay.length"
                >
                  <div
                    class="row-detail"
                    [@detailExpand]="
```

```
                    element == expandedElement ? 'expanded' : 'collapsed'
                  "
                >
                  <table>
                    <thead style="text-align: left; border:1px solid #cccc">
                      <tr>
                        <th><b>Indicador</b></th>
                        <th><b>Puntaje</b></th>
                      </tr>
                    </thead>
                    <tbody>
                      <tr
                        *ngFor="
                          let puntaje of element.puntajes;
                          let in = index
                        "
                      >
                        <td>{{ puntaje.indicador }}</td>
                        <td>{{ puntaje.puntaje }}</td>
                      </tr>
                    </tbody>
                  </table>
                </div>
              </td>
            </ng-container>

            <tr mat-header-row *matHeaderRowDef="columnsToDisplay"></tr>
            <tr
              mat-row
              *matRowDef="let element; columns: columnsToDisplay"
              class="element-row"
              [class.expanded-row]="expandedElement === element"
              (click)="expandedElement = element"
            ></tr>
            <tr
              mat-row
              *matRowDef="let row; columns: ['expandedDetail']"
              class="detail-row-tr"
            ></tr>
          </table>

          <!-- /results table -->
          <div class="buttons-container" fxLayoutGap="20px">
            <button mat-button matStepperPrevious type="button" mat-raised-button>
              Atras
            </button>
            <button
              mat-button
              [mat-dialog-close]="closeModal"
              matStepperNext
              type="submit"
              color="primary"
              mat-raised-button
            >
              Guardar
            </button>
          </div>
        </mat-step>
      </mat-horizontal-stepper>
    </ng-template>
  </form>
  <mat-spinner
    *ngIf="isLoading$ | async"
    align="center"
    [diameter]="48"
    style="margin:0 auto;"
  ></mat-spinner>
```

ii. edit.component.ts

```
import {
  Component,
  Inject,
  OnInit,
  ViewChild,
  AfterViewInit
} from "@angular/core";
import {
  NgForm
} from "@angular/forms";
import {animate, state, style, transition, trigger} from '@angular/animations';

import {
  MAT_DIALOG_DATA,
  MatDialogRef,
  MatTableDataSource,
  MatSort,
  MatPaginator
} from "@angular/material";
import {
  ReglaService,
  UIService,
  CarpetaService,
  ProyectoService,
  AuthService
} from "../../../services";

import { Observable } from "rxjs";
import { Store } from "@ngrx/store";

import * as fromRoot from "../../../store/app.reducers";
import * as UI from "../../../store/ui//ui.actions";
import { SelectionModel } from "@angular/cdk/collections";

import { Proyecto, Regla, Carpeta } from "../../../models";
import { Router, ActivatedRoute } from "@angular/router";

@Component({
  selector: "app-edit-carpeta",
  templateUrl: "edit.component.html",
  styleUrls: ["../../create/create.component.css"],
  animations: [
    trigger('detailExpand', [
      state('collapsed', style({height: '0px', minHeight: '0', display: 'none'})),
      state('expanded', style({height: '*'})),
      transition('expanded <=> collapsed', animate('225ms cubic-bezier(0.4, 0.0, 0.2, 1)')),
    ]),
  ],
})
export class EditCarpetaComponent implements OnInit, AfterViewInit {
  displayedColumns = ["select", "nombre"];
  carpetaArray: any = [];
  dataSource = new MatTableDataSource<Proyecto>();
  selection = new SelectionModel<Proyecto>(true, []);
  isLoading$: Observable<boolean>;

  public reglas: Regla[] = [];
  public proyectos: Proyecto[] = [];

  @ViewChild(MatSort) sort: MatSort;
  @ViewChild(MatPaginator) paginator: MatPaginator;

  proyectoSel: any[] = [];

  disabled = false;
  ShowFilter = false;
```

```
      limitSelection = false;
      public closeModal: boolean;

      ////expandable:
      dataSource_ = new MatTableDataSource();;
      columnsToDisplay = ['proyecto', 'puntaje'];
      expandedElement: PuntajesProyectos;
      public detailData: any;
      id: string;
      mobileScreen = false;
      desktopScreen = true;
      mHeight: any;
      mWidth: any;
      idAlcaldia: string;
      evaluaciones: any = [];
      proyectosEvaluados: any = [];
      constructor(
        private carpetaService: CarpetaService,
        private proyectoService: ProyectoService,
        private reglaService: ReglaService,
        private uiService: UIService,
        private store: Store<fromRoot.State>,
        private route: ActivatedRoute,
        private router: Router,
        private authService: AuthService
      ) {
        this.mHeight = window.screen.height;
        this.mWidth = window.screen.width;
        this.desktopScreen = false;
        if (this.mWidth <= 700 || (this.mWidth == 768 && this.mHeight == 1024)) {
          this.mobileScreen = true;
        }
        this.id = this.route.snapshot.paramMap.get("id");
        this.idAlcaldia = this.authService.currentUserValue.user.idAlcaldia; //pasar esto a un solo
lugar para reutilizarlo
      }

      ngOnInit() {
        this.store.dispatch(new UI.StartLoading());
        this.getDetails();
        this.getReglas();
        this.getProyectos();
        this.isLoading$ = this.store.select(fromRoot.getIsLoading);
        this.dataSource.data = this.proyectos;
      }

      ngAfterViewInit() {
        this.dataSource.sort = this.sort;
        this.dataSource.paginator = this.paginator;
      }

      doFilter(filterValue: string) {
        this.dataSource.filter = filterValue.trim().toLowerCase();
      }

      /** Whether the number of selected elements matches the total number of rows. */
      isAllSelected() {
        const numSelected = this.selection.selected.length;
        const numRows = this.dataSource.data.length;
        return numSelected === numRows;
      }

      /** Selects all rows if they are not all selected; otherwise clear selection. */
      masterToggle() {
        this.isAllSelected()
          ? this.selection.clear()
          : this.dataSource.data.forEach(row => this.selection.select(row));
      }
      onSubmit(form: NgForm) {
```

```
        let alcaldia = form.value.generales.idAlcaldia;
        if(!alcaldia){
           alcaldia =  this.idAlcaldia
        }
        this.carpetaArray = {
          nombre: form.value.generales.nombre,
          descripcion: form.value.generales.descripcion,
          regla: this.reglas.filter(
            x => x.id == form.value.reglas.regla
          )[0],
          proyectos: this.selection.selected,
          status: true
        };
        this.carpetaService
          .updateCarpeta(this.detailData.id, this.carpetaArray)
          .subscribe(
            data => {
              this.store.dispatch(new UI.StopLoading());
              this.uiService.showSnackbar(
                "Carpeta actualizada!",
                null,
                3000,
                "custom-snack-bar"
              );
              this.router.navigate(["/carpetas/administrar"]);
            },
            error => {
              this.store.dispatch(new UI.StopLoading());
              this.uiService.showSnackbar(
                "Problemas actualizando carpeta",
                null,
                15000,
                "custom-snack-bar-error"
              );
            }
          );
      }
      getReglas() {
        return this.reglaService.getAllReglas().subscribe((data: Regla[]) => {
          this.reglas = data;
        });
      }
      getProyectos() {
        return this.proyectoService
          .getAllProyectos()
          .subscribe((proyectoData: Proyecto[]) => {
            this.proyectos = proyectoData;
            this.dataSource.data = this.proyectos;
            setTimeout(() => {
                this.selection = new SelectionModel<Proyecto>(
                    true,
                    this.detailData.proyectos
                  );
                  this.proyectos.forEach((pro, y) => {
                    this.detailData.proyectos.forEach((element, x) => {
                      if (element.id == pro.id) {
                        this.proyectos[y].isSelected = true;
                      }
                    });
                  });
            }, 1500);
            //this.selection.selected.push();
          });
      }
      getDetails() {
        this.carpetaService.getCarpeta(this.id).subscribe((carpetaData: Carpeta[]) => {
         this.detailData = carpetaData;
         console.log("detailData333",this.detailData)
         this.store.dispatch(new UI.StopLoading());
```

```
      });
    }
    evaluar() {
      this.carpetaService
        .evaluarCarpeta(this.detailData.id)
        .subscribe(
          data => {
            let puntajeIndicadores = 0;
            this.evaluaciones = data['carpeta']['proyectos'];
            let puntajesPro = [];
            this.evaluaciones.forEach((pro, y) => {
              puntajesPro =  pro.puntajes;
              //TODO hay numeros negativos
              // puntajesPro.forEach((element, x) => {
              //     puntajeIndicadores = puntajeIndicadores + (+element.puntaje);

              // });
              this.proyectosEvaluados.push({
                  'proyecto': pro.nombre,
                  'puntaje': puntajeIndicadores,
                  'puntajes': pro.puntajes
              })
            });
            this.dataSource_.data = this.proyectosEvaluados;
          //   setTimeout(() => {

              this.store.dispatch(new UI.StopLoading());
              this.uiService.showSnackbar(
                  "Carpeta Evaluada!",
                  null,
                  3000,
                  "custom-snack-bar"
              );
          //   }, 1500);

            //this.router.navigate(["/carpetas/administrar"]);
          },
          error => {
            this.store.dispatch(new UI.StopLoading());
            this.uiService.showSnackbar(
              "Problemas evaluando carpeta",
              null,
              15000,
              "custom-snack-bar-error"
            );
          }
        );
    }
  }
  export interface Puntajes {
    indicador: string;
    puntaje: number;
  }
  export interface PuntajesProyectos {
    proyecto: string;
    puntaje: number;
    tablaPuntajes: Puntajes[];
  }
```

c. evaluaciones

i. evaluaciones.component.html

```
<h3>Listado de Evaluaciones de Carpeta: </h3>
<mat-table [dataSource]="dataSource" matSort>
```

```
    <ng-container matColumnDef="nombre">
      <mat-header-cell mat-header-cell *matHeaderCellDef>
        Proyecto
      </mat-header-cell>
      <mat-cell *matCellDef="let element"> {{ element.nombre }} </mat-cell>
    </ng-container>
    <ng-container matColumnDef="descripcion">
      <mat-header-cell mat-header-cell *matHeaderCellDef>
        Puntaje
      </mat-header-cell>
      <mat-cell *matCellDef="let element"> {{ element.descripcion }} </mat-cell>
    </ng-container>
    <ng-container matColumnDef="actions">
      <mat-header-cell *matHeaderCellDef mat-sort-header>Acciones</mat-header-cell>
      <mat-cell *matCellDef="let element">
        <a routerLink="{{element.id}}"><mat-icon>Detalles</mat-icon></a>
        <button mat-icon-button (click)="onDelete(element.id)">
          <mat-icon>delete</mat-icon>
        </button>
      </mat-cell>
    </ng-container>
    <mat-header-row *matHeaderRowDef="displayedColumns"></mat-header-row>
    <mat-row *matRowDef="let row; columns: displayedColumns"></mat-row>
  </mat-table>
  <div *ngIf="notResults" align="center">No se encontraron resultados</div>

  <mat-paginator [pageSize]="5" [pageSizeOptions]="[5, 10, 20]"></mat-paginator>
  <mat-spinner
    *ngIf="(isLoading$ | async)"
    align="center"
    [diameter]="48"
    style="margin:0 auto;"
  ></mat-spinner>
```

ii. evaluaciones.component.ts

```
  import {
      Component,
      OnInit,
      ViewChild,
      AfterViewInit,
      ChangeDetectorRef
    } from "@angular/core";
    import { Observable } from "rxjs";
    import {
      MatTableDataSource,
      MatSort,
      MatPaginator,
      MatDialog
    } from "@angular/material";
    import { CarpetaService } from "../../../services";
    import { Carpeta } from "../../../models";
    import { Store } from "@ngrx/store";
    import * as fromRoot from "../../../store/app.reducers";
    import * as UI from "../../../store/ui/ui.actions";
    import { ActivatedRoute } from "@angular/router";
    interface Reglas {
      id: any;
      nombre: string;
    }

    @Component({
      selector: "app-evaluaciones",
      templateUrl: "./evaluaciones.component.html",
      styleUrls: ["../../create/create.component.css"]
    })
```

```
export class EvaluacionesCarpetaComponent implements OnInit, AfterViewInit {
  carpeta: Carpeta;
  displayedColumns = ["nombre", "descripcion", "actions"];
  dataSource = new MatTableDataSource<Carpeta>();
  isLoading$: Observable<boolean>;

  @ViewChild(MatSort) sort: MatSort;
  @ViewChild(MatPaginator) paginator: MatPaginator;
  notResults = false;
  carpetaId;
  constructor(
    private carpetaService: CarpetaService,
    private dialog: MatDialog,
    private changeDetectorRefs: ChangeDetectorRef,
    private store: Store<fromRoot.State>,
    private route: ActivatedRoute,
  ) {}

  ngOnInit() {
    this.isLoading$ = this.store.select(fromRoot.getIsLoading);
    this.carpetaId = this.route.snapshot.paramMap.get("id");
    this.onRefresh();
  }

  ngAfterViewInit() {
    this.dataSource.sort = this.sort;
    this.dataSource.paginator = this.paginator;
  }
  doFilter(filterValue: string) {
    this.dataSource.filter = filterValue.trim().toLowerCase();
  }

  onRefresh() {
    this.store.dispatch(new UI.StartLoading());
    console.log(this.carpetaId)
    this.carpetaService.getEvaluacionesById(this.carpetaId).subscribe(data => {
      console.log("evaluacion", data)
      // this.dataSource.data = carpeta;
      // if(this.dataSource.data.length == 0){
      //   this.notResults = true;
      // }
      this.store.dispatch(new UI.StopLoading());
    });
  }

  // onDelete(id) {
  //   this.carpetaService.getCarpeta(id).subscribe((carpetaData: Carpeta[]) => {
  //     const dialogRef = this.dialog
  //       .open(DeleteCarpetaComponent, {
  //         data: {
  //           carpeta: carpetaData,
  //           id: id
  //         }
  //       })
  //       .afterClosed()
  //       .subscribe(result => {
  //         this.onRefresh();
  //       });
  //   });
  // }
}
```

d. list.component.css

```
.form-container {
```

```
  display: flex;
  flex-direction: column;
}

::ng-deep .mat-snack-bar-container {
  background-color: #36a5e6;
  color: #fff;
}

.mat-table {
  overflow: auto;
}
h3{
  color: #3f51b5;
  font-weight: bold;
  text-align: center;
}
```

e. list.component.html

```
<h3>Listado de carpetas</h3>
<mat-table [dataSource]="dataSource" matSort>
  <ng-container matColumnDef="nombre">
    <mat-header-cell mat-header-cell *matHeaderCellDef>
      Nombre
    </mat-header-cell>
    <mat-cell *matCellDef="let element"> {{ element.nombre }} </mat-cell>
  </ng-container>
  <ng-container matColumnDef="descripcion">
    <mat-header-cell mat-header-cell *matHeaderCellDef>
      Descripcion
    </mat-header-cell>
    <mat-cell *matCellDef="let element"> {{ element.descripcion }} </mat-cell>
  </ng-container>
  <ng-container matColumnDef="actions">
    <mat-header-cell *matHeaderCellDef mat-sort-header>Actions</mat-header-cell>
    <mat-cell *matCellDef="let element">
      <a routerLink="{{element.id}}"><mat-icon>edit</mat-icon></a>
      <button mat-icon-button (click)="onDelete(element.id)">
        <mat-icon>delete</mat-icon>
      </button>
      <a routerLink="/carpetas/evaluaciones/{{element.id}}"><mat-icon>bar_chart</mat-icon></a>
    </mat-cell>
  </ng-container>
  <mat-header-row *matHeaderRowDef="displayedColumns"></mat-header-row>
  <mat-row *matRowDef="let row; columns: displayedColumns"></mat-row>
</mat-table>
<div *ngIf="notResults" align="center">No se encontraron resultados</div>

<mat-paginator [pageSize]="5" [pageSizeOptions]="[5, 10, 20]"></mat-paginator>
<mat-spinner
  *ngIf="(isLoading$ | async)"
  align="center"
  [diameter]="48"
  style="margin:0 auto;"
></mat-spinner>
```

f. list.component.ts

```
import {
  Component,
  OnInit,
  ViewChild,
```

```
    AfterViewInit,
    ChangeDetectorRef
  } from "@angular/core";
  import { Observable } from "rxjs";
  import {
    MatTableDataSource,
    MatSort,
    MatPaginator,
    MatDialog
  } from "@angular/material";
  import { CarpetaService } from "../../services";
  import { Carpeta } from "../../models";
  import { EditCarpetaComponent } from "./edit/edit.component";
  import { DeleteCarpetaComponent } from "./delete/delete.component";
  import { Store } from "@ngrx/store";
  import * as fromRoot from "../../store/app.reducers";
  import * as UI from "../../store/ui/ui.actions";

  interface Reglas {
    id: any;
    nombre: string;
  }

  @Component({
    selector: "app-list",
    templateUrl: "./list.component.html",
    styleUrls: ["./list.component.css"]
  })
  export class ListComponent implements OnInit, AfterViewInit {
    carpeta: Carpeta;
    displayedColumns = ["nombre", "descripcion", "actions"];
    dataSource = new MatTableDataSource<Carpeta>();
    isLoading$: Observable<boolean>;

    @ViewChild(MatSort) sort: MatSort;
    @ViewChild(MatPaginator) paginator: MatPaginator;
    notResults = false;

    constructor(
      private carpetaService: CarpetaService,
      private dialog: MatDialog,
      private changeDetectorRefs: ChangeDetectorRef,
      private store: Store<fromRoot.State>
    ) {}

    ngOnInit() {
      this.isLoading$ = this.store.select(fromRoot.getIsLoading);
      this.onRefresh();
    }

    ngAfterViewInit() {
      this.dataSource.sort = this.sort;
      this.dataSource.paginator = this.paginator;
    }
    doFilter(filterValue: string) {
      this.dataSource.filter = filterValue.trim().toLowerCase();
    }

    onRefresh() {
      this.store.dispatch(new UI.StartLoading());
      this.carpetaService.getAllCarpetas().subscribe((carpeta: Carpeta[]) => {
        this.dataSource.data = carpeta;
        if(this.dataSource.data.length == 0){
          this.notResults = true;
        }
        this.store.dispatch(new UI.StopLoading());
      });
    }
```

```
    onDelete(id) {
      this.carpetaService.getCarpeta(id).subscribe((carpetaData: Carpeta[]) => {
        const dialogRef = this.dialog
          .open(DeleteCarpetaComponent, {
            data: {
              carpeta: carpetaData,
              id: id
            }
          })
          .afterClosed()
          .subscribe(result => {
            this.onRefresh();
          });
      });
    }
  }
```

3. carpeta.module.ts

```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { RouterModule, Routes } from '@angular/router';

import { CreateComponent } from './create/create.component';
import { ListComponent } from './list/list.component';
import { CarpetaService, ProyectoService, ReglaService } from '../services';
import { MaterialModule } from '../material.module';
import { EditCarpetaComponent } from './list/edit/edit.component';
import { DeleteCarpetaComponent } from './list/delete/delete.component';

import { EvaluacionesCarpetaComponent } from './list/evaluaciones/evaluaciones.component';

import { NgxMatSelectSearchModule } from 'ngx-mat-select-search';
import { NgMultiSelectDropDownModule } from 'ng-multiselect-dropdown';
import { ReactiveFormsModule, FormsModule } from '@angular/forms';

const routes: Routes = [
  { path: "crear", component: CreateComponent },
  { path: "administrar", component: ListComponent },
  { path: "administrar/:id", component: EditCarpetaComponent },
  { path: "evaluaciones/:id", component: EvaluacionesCarpetaComponent }

];
@NgModule({
  imports: [
    CommonModule,
    MaterialModule,
    NgxMatSelectSearchModule,
    NgMultiSelectDropDownModule.forRoot(),
    ReactiveFormsModule,
    FormsModule,
    RouterModule.forChild(routes),
  ],
  declarations: [
    CreateComponent,
    ListComponent,
    EditCarpetaComponent,
    DeleteCarpetaComponent,
    EvaluacionesCarpetaComponent
  ],
  entryComponents: [
    DeleteCarpetaComponent
  ],
  providers: [CarpetaService, ProyectoService, ReglaService]
})
export class CarpetaModule { }
```

4. custom.validator.ts

```
import { FormArray } from '@angular/forms';

export class CustomValidators {
  static multipleCheckboxRequireOne(fa: FormArray) {
    let valid = false;

    for (let x = 0; x < fa.length; ++x) {
      if (fa.at(x).value) {
        valid = true;
        break;
      }
    }
    return valid ? null : {
      multipleCheckboxRequireOne: true
    };
  }
}
```

iii. guards

1. auth.guard.ts

```
import { Injectable } from '@angular/core';
import { Router, CanActivate, ActivatedRouteSnapshot, RouterStateSnapshot } from '@angular/router';

import { AuthService } from '../services';
import { Store } from '@ngrx/store';
import { take } from 'rxjs/operators';

import * as fromRoot from '../store/app.reducers';

@Injectable({ providedIn: 'root' })
export class AuthGuard implements CanActivate {
    constructor(
        private router: Router,
        private authenticationService: AuthService,
        private store: Store<fromRoot.State>
    ) {}

    canActivate(route: ActivatedRouteSnapshot, state: RouterStateSnapshot) {
        const currentUser = this.authenticationService.currentUserValue;
        if (!currentUser) {
            this.router.navigate(['/login'], { queryParams: { returnUrl: state.url }});
        }
        return this.store.select(fromRoot.getIsAuth).pipe(take(1));

      }

}
```

2. index.ts

```
export * from './auth.guard';
```

iv. helpers

1. validators

a. index.ts

```
export * from './maxMinValidator';
```

b. maxMinValidator.ts

```
import {AbstractControl, NG_VALIDATORS, Validator} from "@angular/forms";
import {Directive, forwardRef, Input} from "@angular/core";
@Directive({
  selector: '[validateMaxMinValue][ngModel],[validateMaxMinValue][formControl]',
  providers: [
    { provide: NG_VALIDATORS, useExisting: forwardRef(() => maxMinValidator), multi: true }
  ]
})
export class maxMinValidator implements Validator {
  @Input() validateMaxMinValue: string;
  validate(c: AbstractControl): { [key: string]: any; } {
    const minMax = this.validateMaxMinValue.split(",");
    const v: number = +c.value;
    const min: number = +minMax[0];
    const max: number = +minMax[1];
    return v >= min && v <= max ? null : {validateMaxMin: true}
  }
}
```

2. error.interceptor.ts

```
import { Injectable } from '@angular/core';
import { HttpRequest, HttpHandler, HttpEvent, HttpInterceptor } from '@angular/common/http';
import { Observable, throwError } from 'rxjs';
import { catchError } from 'rxjs/operators';

import { AuthService } from '../services';

@Injectable()
export class ErrorInterceptor implements HttpInterceptor {
    constructor(private authService: AuthService) {}

    intercept(request: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> {
        return next.handle(request).pipe(catchError(err => {
            if (err.status === 401 || err.status === 500) {
                // auto logout if 401 response returned from api
                this.authService.logout();

                //location.reload(true);
            }
            console.log('errror: ',err)
            const error = err.error.error.message || err.statusText;
            return throwError(error);
        }))
    }
}
```

3. index.ts

```
export * from './error.interceptor';
export * from './jwt.interceptor';
export * from './validators/index'
```

4. jwt.interceptor.ts

```
import { Injectable } from '@angular/core';
import { HttpRequest, HttpHandler, HttpEvent, HttpInterceptor } from '@angular/common/http';
import { Observable } from 'rxjs';

import { AuthService } from '../services';

@Injectable()
export class JwtInterceptor implements HttpInterceptor {
    constructor(private authService: AuthService) {}

    intercept(request: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> {
        // add authorization header with jwt token if available
        let currentUser = this.authService.currentUserValue;
        if (currentUser && currentUser.token) {
            request = request.clone({
                setHeaders: {
                    'Content-Type': 'application/json',
                    Authorization: `${currentUser.token.id}`
                }
            });
        }

        return next.handle(request);
    }
}
```

v. home

1. navigation

a. header

i. header.component.css

```
a {
  text-decoration: none;
  color: white;
}

a:hover,
a:active {
  color: white;
}

.navigation-items {
  list-style: none;
  padding: 0;
  margin: 0;
}

.nav-caption {
  display: inline-block;
  padding-left: 6px;
}
```

```
.main-header {
  position: fixed;
  top: 0;
  left: 0;
  right: 0;
}
.mat-toolbar.mat-primary {
  background: #1e88e5;
  color: #fff;
}
.img-logo{
  margin-top: 10px;
  margin-left: 10px;
}
.logo-fch{
  letter-spacing: 2px;
  font-weight: bold;
}
.logo-sch{
  color: #becdda;
  font-weight: bolder;
}
```

ii. header.component.html

```
<mat-toolbar color="primary" class="main-header">
  <!-- <div fxHide.gt-xs> -->
  <div>
    <button mat-icon-button (click)="onToggleSidenav()">
      <mat-icon>menu</mat-icon>
    </button>
  </div>
  <div>
    <!-- <a routerLink="/"><img src="../../../assets/img/logo-white.png" width="60%" class="img-
logo"></a> -->
    <a routerLink="/"><span class="logo-fch">SMART</span><span class="logo-sch">PROJECT</span></a>

  </div>
  <div fxFlex fxLayout fxLayoutAlign="flex-end" fxHide.xs>
    <ul fxLayout fxLayoutGap="10px" class="navigation-items">
      <!-- <li>
        <a routerLink="/login"><mat-icon>power_settings_new</mat-icon></a>
      </li> -->
      <li>
        <a style="cursor: pointer" (click)="onLogout()" title="Log Out"><mat-icon>exit_to_app</mat-
icon></a>
      </li>
    </ul>
  </div>
</mat-toolbar>
```

iii. header.component.ts

```
import { Component, OnInit, EventEmitter, Output } from "@angular/core";
import { Observable } from "rxjs";
import { Router } from "@angular/router";
import { AuthService } from "../../../services";

@Component({
  selector: "app-header",
  templateUrl: "./header.component.html",
  styleUrls: ["./header.component.css"]
})
```

```
export class HeaderComponent implements OnInit {
  @Output() sidenavToggle = new EventEmitter<void>();
  isAuth: boolean;

  constructor(private authService: AuthService, private router: Router) {}

  ngOnInit() {
    this.isAuth = true;
  }

  onToggleSidenav() {
    this.sidenavToggle.emit();
  }

  onLogout() {
    this.authService.logout();
    this.router.navigate(["/login"]);

  }
}
```

b. sidenav-list

i. sidenav-list.component.css

```
a {
  text-decoration: none;
  color: #8f999e;
}

/* a:hover,
a:active {
  color: blue;
} */
.item, .mat-nav-list .mat-list-item, .mat-selection-list .mat-list-item {
  color: #8f999e;
}
.mat-list .mat-subheader, .mat-nav-list .mat-subheader, .mat-selection-list .mat-subheader {
  color: lightgray;
}
::ng-deep .mat-nav-list .mat-list-item {
  color: #8f999e;
}
/* 273a42 */
.mat-list-item:hover{
  background: #ece4e463;
  color: #1e88e5;
}
.mat-icon{
  color: #8f999e;
  margin-right: 10px;
}
.mat-card{
  background: #1e2c32;
  color: white;
}
```

ii. sidenav-list.component.html

```
<a mat-list-item *ngIf="!item.disabled" [ngStyle]="{'padding-left': (depth * 12) + 'px'}"
(click)="onItemSelected(item)"
                [ngClass]="{'active': item.route ? router.isActive(item.route, true): false,
'expanded': expanded}">
    <mat-icon class="routeIcon">{{item.iconName}}</mat-icon>
    {{item.displayName}}
```

```
    <span fxFlex *ngIf="item.children && item.children.length">
      <span fxFlex></span>
      <mat-icon [@indicatorRotate]="expanded ? 'expanded': 'collapsed'">
        expand_more
      </mat-icon>
    </span>
  </a>
  <div *ngIf="expanded">
    <app-sidenav-list *ngFor="let child of item.children" [item]="child" [depth]="depth+1">
    </app-sidenav-list>
  </div>
```

iii. sidenav-list.component.ts

```
import { Component, OnInit, EventEmitter, Output, Input, HostBinding } from '@angular/core';
import { Observable } from 'rxjs';
import { Router } from '@angular/router';
import { NavItem } from '../../../models';
// import { AuthService } from '../../auth/auth.service';
// import * as fromRoot from '../../app.reducer';
// import { User } from '../../auth/user.model';
import { NavService } from '../../../services';
import {animate, state, style, transition, trigger} from '@angular/animations';

@Component({
  selector: 'app-sidenav-list',
  templateUrl: './sidenav-list.component.html',
  styleUrls: ['./sidenav-list.component.css'],
  animations: [
    trigger('indicatorRotate', [
      state('collapsed', style({transform: 'rotate(0deg)'})),
      state('expanded', style({transform: 'rotate(180deg)'})),
      transition('expanded <=> collapsed',
        animate('225ms cubic-bezier(0.4,0.0,0.2,1)')
      ),
    ])
  ]
})
export class SidenavListComponent implements OnInit {

  @Output() closeSidenav = new EventEmitter<void>();
  isAuth$: Observable<boolean>;
  //currentUser$: Observable<User>;
  opened: any = false;
  rol: string;
  @Input() item: NavItem;
  expanded: boolean;
  @HostBinding('attr.aria-expanded') ariaExpanded = this.expanded;
  @Input() depth: number;
  mHeight: any;
  mWidth: any;
  constructor(
   // private authService: AuthService,
    private navService: NavService,
   // private store: Store<fromRoot.State>,
    public router: Router
  ) {
    if (this.depth === undefined) {
      this.depth = 0;
    }
    this.mHeight = (window.screen.height);
    this.mWidth = (window.screen.width);
  }

  ngOnInit() {
    // this.isAuth$ = this.store.select(fromRoot.getIsAuth);
```

```
      // this.currentUser$ = this.store.select(fromRoot.getCurrentUser);
    }

    onClose() {
      this.closeSidenav.emit();
    }

    onItemSelected(item: NavItem) {
      if (!item.children || !item.children.length) {
        this.router.navigate([item.route]);
        if(this.mWidth <= 700 || this.mWidth == 768 && this.mHeight == 1024){
          this.navService.closeNav();
        }
      }
      if (item.children && item.children.length) {
        this.expanded = !this.expanded;
      }
    }
  }
```

2. home.component.css

3. home.component.html

```
  <div *ngIf="((currentUser$ | async)?.user.rol != 'root'); else rootRole" class="welcome"
fxLayout="column" fxLayout.gt-md="row" fxLayoutGap.gt-md="20px" fxLayoutAlign="center center">
    <section>
      <h1>Bienvenid@ {{ (currentUser$ | async)?.user.nombre }} {{ (currentUser$ |
async)?.user.apellido }}</h1>
      <p>(Nombre de la alcaldia aqui) </p>
    </section>
  </div>
  <ng-template #rootRole>
    <div class="welcome" fxLayout="column" fxLayout.gt-md="row" fxLayoutGap.gt-md="20px"
fxLayoutAlign="center center">
      <section>
        <h1>PROYECTOS</h1>
        <p>Mostrar los proyectos recientes</p>
      </section>
      <section>
        <h1>REPORTES</h1>
        <p>La verdad no se que podemos mostrar en el home</p>
      </section>
    </div>
  </ng-template>
```

4. home.component.ts

```
  import { Component, OnInit } from '@angular/core';
  import { Observable } from 'rxjs';

  import { Store } from '@ngrx/store';
  import * as fromRoot from '../store/app.reducers';
  import { Auth } from '../models';
  @Component({
    selector: 'app-home',
    templateUrl: './home.component.html',
    styleUrls: ['./home.component.css']
  })
  export class HomeComponent implements OnInit {
    currentUser$: Observable<Auth>;
    constructor(private store: Store<fromRoot.State>) { }
```

```
  ngOnInit() {
    this.currentUser$ = this.store.select(fromRoot.getCurrentUser);
  }

}
```

5. index.ts

```
  export * from './home.component';
  export * from './navigation/header/header.component';
  export     *     from     './navigation/sidenav-list/sidenav-
list.component';
```

vi. login

1. index.ts

```
export * from './login.component';
```

2. login.component.css

```
.login-page{
  height: 100vh;
  width: 100%;
}

.login-spacer{
  height: 20vh;
}

.login-container{
  margin: auto;
  width: 300px;
  padding: 2vh;
  background-color: #fff;
}

.login-form{
  display: flex;
  flex-direction: column;
}

.card-heading{
  text-align: center;
  color: rgba(0, 0, 0, 0.31);
}

.buttons .login{
  padding-top: 2vh;
  padding-bottom: 2vh;
}

.buttons .login > button{
  width: 100%;
}
/* Absolute Center Spinner */
.loading-indicator {
  position: fixed;
```

```
    z-index: 999;
    height: 0.5em;
    width: 0.5em;
    overflow: visible;
    margin: auto;
    top: 0;
    left: 0;
    bottom: 0;
    right: 0;
  }

  /* Transparent Overlay */
  .loading-indicator:before {
    content: '';
    display: block;
    top: 0;
    left: 0;
    width: 50%;
    height: 50%;
  }
```

3. login.component.html

```
  <div class="login-page">
    <div class="login-spacer"></div>
    <div class="login-container">
      <header fxLayout="column" fxLayoutAlign="center center" fxLayoutGap="50px">
        <span class="logo-login">
          <img src="assets/img/logo.png" alt="SmartAdmin" />
        </span>
      </header>
      <form class="login-form" [formGroup]="loginForm" (ngSubmit)="onSubmit()">
        <mat-form-field>
          <input
            type="text"
            matInput
            placeholder="Nombre de usuario"
            formControlName="email"
          />
          <mat-error>Nombre de usuario inválido.</mat-error>
        </mat-form-field>
        <mat-form-field>
          <input
            type="password"
            matInput
            placeholder="Contraseña"
            formControlName="password"
          />
          <mat-hint>Ingrese su contraseña.</mat-hint>
          <mat-error>Este campo no puede estar vacío.</mat-error>
        </mat-form-field>
        <div class="buttons">
          <div class="login">
            <button
              *ngIf="!(isLoading$ | async)"
              type="submit"
              mat-raised-button
              color="primary"
              [disabled]="loginForm.invalid"
            >
              Ingresar
            </button>
          </div>
        </div>
        <div class="loading-indicator">
          <mat-progress-spinner
```

```
            style="margin:0 auto;"
            [diameter]="50"
            mode="indeterminate"
            *ngIf="(isLoading$ | async)"
          ></mat-progress-spinner>
        </div>
      </form>
    </div>
  </div>
```

4. login.component.ts

```
import { Component, OnInit } from "@angular/core";
import { FormGroup, FormControl, Validators } from "@angular/forms";
import { Observable } from "rxjs";
import { Router, ActivatedRoute } from "@angular/router";
import { first } from 'rxjs/operators';
import { Store } from '@ngrx/store';

import { AuthService, UIService } from "../services";
import * as fromRoot from '../store/app.reducers';
import * as UI from '../store/ui/ui.actions';
import * as Auth from '../store/auth/auth.actions';

import { MatSnackBar } from '@angular/material';

@Component({
  selector: "app-login",
  templateUrl: "./login.component.html",
  styleUrls: ["./login.component.css"]
})
export class LoginComponent implements OnInit {
  loginForm: FormGroup;
  isLoading$: Observable<boolean>;
  submitted = false;
  returnUrl: string;

  constructor(
    private route: ActivatedRoute,
    private router: Router,
    private authService: AuthService,
    private uiService: UIService,
    private store: Store<fromRoot.State>
  ) {
    // redirect to home if already logged in
    if (this.authService.currentUserValue) {
      this.store.dispatch(new Auth.SetAuthenticated());
      this.router.navigate(["/"]);
    }
  }

  ngOnInit() {
    this.isLoading$ = this.store.select(fromRoot.getIsLoading);
    this.loginForm = new FormGroup({
      email: new FormControl("", {
        validators: [Validators.required]
      }),
      password: new FormControl("", { validators: [Validators.required] })
    });
    this.returnUrl = this.route.snapshot.queryParams["returnUrl"] || "/";
  }

  onSubmit() {
    this.store.dispatch(new UI.StartLoading());
    this.authService
      .login({
```

```
          username: this.loginForm.value.email,
          password: this.loginForm.value.password
        })
        .pipe(first())
        .subscribe(
          data => {
            this.store.dispatch(new UI.StopLoading());
            this.authService.initAuthListener();
            this.router.navigate([this.returnUrl]);
          },
          error => {
            this.store.dispatch(new UI.StopLoading());
            this.uiService.showSnackbar("Credenciales Incorrectas", null, 3000, null);
          }
        );
    }
  }
```

vii. models

1. alcaldia.model.ts

```
export interface Alcaldia {
  nombre: string,
  descripcion: string,
  direccion: string,
  municipio: string,
  ciudad: string,
  id: string
}
```

2. area.model.ts

```
export interface Area {
  id: any,
  nombre: string,
  puntaje: number
}
```

3. auth.model.ts

```
import { User } from "./user.model";

export interface Auth {
    user:User,
    token:{
        created: string;
        id: string;
        ttl: number;
        userId: string;
    }
  }
```

4. beneficio.model.ts

```
export interface Beneficio {
```

```
    nombre: string,
    monto: string,
    descripcion?: string
}
```

5. carpeta.model.ts

```
export interface Carpeta {
  nombre: string;
  descripcion: string;
  proyectos: any[];
  regla: any[];
  status: true;
  id: string;
  idAlcaldia: string;
}
```

6. costo.model.ts

```
export interface Costo {
    nombre: string,
    monto: string,
    descripcion?: string
}
```

7. index.ts

```
export * from './user.model';
export * from './login.model';
export * from './auth.model';
export * from './nav-item.model';
export * from './alcaldia.model';
export * from './carpeta.model'
export * from './proyecto.model';
export * from './regla.model';
export * from './area.model';
export * from './indicador.model';
export * from './tipo-proyecto.model';
export * from './costo.model';
export * from './beneficio.model';
```

8. indicador.model.ts

```
export interface Indicador {
  id: any,
  nombre: string,
  valores: any[],
  editable: boolean
}
```

9. login.model.ts

```
export interface AuthData {
  username: string;
```

```
  password: string;
}
```

10. nav-item.model.ts

```
export interface NavItem {
    displayName: string;
    disabled?: boolean;
    iconName: string;
    route?: string;
    code?:string;
    children?: NavItem[];
  }
```

11. proyecto.model.ts

```
export interface Proyecto {
  idAlcaldia: string;
  nombre: string;
  descripcion?: string;
  beneficiarios?: number;
  beneficios?: any[];
  costos?: any[];
  objetivos?: string;
  vidaUtil?: number;
  duracion?: number;
  inversion?: number;
  presupuesto?: number;
  costoTotal?: number;
  valorResidual?: number;
  ubicacion?: {
    lat: 0,
    lng: 0
  };
  tipoProyecto?: {};
  areaPrioritaria?: {};
  puntaje?: number;
  responsable?: {};
  status?: true;
  id?: string;
  isSelected?: boolean;
  costoConstante: boolean;
  porcentajeCostoVariable: number;
  flujoCaja?: {};
}
```

12. regla.model.ts

```
import { Area } from "./area.model";
import { Indicador } from "./indicador.model";

export interface Regla {
  idAlcaldia: string;
  nombre: string;
  descripcion?: string;
  areasPrioritarias?: Area[];
  indicadores?: Indicador[];
  status: true;
  id?: string;
```

```
}
```

13. tipo-proyecto.model.ts

```
import { Costo } from "./costo.model";
import { Beneficio } from "./beneficio.model";

export interface TipoProyecto {
    id?: number,
    nombre: string,
    nombreCompleto: string,
    costos: any[],
    beneficios: any[],
}
```

14. user.model.ts

```
export interface User {
  email: string;
  userId: string;
  nombre: string;
  apellido: string;
  username: string;
  rol: string;
  cargo: string;
  status: true;
  idAlcaldia: string;
  id: string;
}
```

viii. proyectos

1. create

a. create.component.css

```
.form-container {
  display: flex;
  flex-direction: column;
  align-items: center;
}

.row-container {
  display: flex;
  flex-direction: row;
  align-items: center;
}

.mat-form-field {
  width: 100%;
}

.costos-item > .mat-form-field {
  width: 100%;
}

.costos-container {
  display: flex;
```

```
    flex-flow: row wrap;
    justify-content: space-around;
  }

  .costos-item {
    display: flex;
    flex-direction: column;
  }
  .stepper-btn {
      margin: 10px;
  }
  ::ng-deep .mat-horizontal-stepper-header {
    pointer-events: none !important;
  }

  agm-map {
    height: 300px;
  }

  ::ng-deep .mat-snack-bar-container {
    background-color: #36a5e6;
    color: #fff;
  }

  h3 {
    color: #3f51b5;
    font-weight: bold;
    text-align: center;
  }
  .icon-back {
      vertical-align: middle;
      padding: 20px;
  }
  .header-flujo-row{
      display: flex;
      align-content: space-between;
  }
  .header-flujo{
      padding: 30px;
      width: 20%;
  }
  .col-flujo{
      width: 20%;
      padding: 10px 30px;
  }
  /*
    ##Device = Desktops
    ##Screen = 1281px to higher resolution desktops
  */

  @media (min-width: 1281px) {
    .mat-form-field {
      width: 30%;
    }
    .costos-item {
      width: 40%;
    }

    .costos-item > .mat-form-field {
      width: 80%;
    }
  }

  /*
    ##Device = Laptops, Desktops
    ##Screen = B/w 1025px to 1280px
  */

  @media (min-width: 1025px) and (max-width: 1280px) {
```

```
    .mat-form-field {
      width: 30%;
    }
    .costos-item {
      width: 40%;
    }

    .costos-item > .mat-form-field {
      width: 80%;
    }
  }
```

b. create.component.html

```
  <h3>Proyectos</h3>
  <form #f="ngForm" (ngSubmit)="onSubmit(f)">
    <ng-template [ngIf]="mobileScreen" [ngIfElse]="desktopScreen">
      <mat-vertical-stepper
        #stepper="matVerticalStepper"
        linear
      >
        <!-- Step 1 : Datos generales -->
        <mat-step>
          <ng-template matStepLabel>General</ng-template>
          <div
            ngModelGroup="generales"
            #generalesGroup="ngModelGroup"
            class="form-container"
          >
            <!-- Alcaldia -->
            <mat-form-field>
              <mat-select
                matInput
                ngModel
                placeholder="Alcaldia"
                idAlcaldia
                name="idAlcaldia"
                required
                #idAlcaldiaInput="ngModel"
              >
                <mat-option>Seleccione una opcion</mat-option>
                <mat-option
                  *ngFor="let alcaldia of alcaldias; let in = index"
                  [value]="alcaldia.id"
                >
                  {{ alcaldia.nombre }}
                </mat-option>
              </mat-select>
              <mat-error>Este campo es requerido</mat-error>
            </mat-form-field>
            <!-- Tipo de proyecto -->
            <mat-form-field>
              <mat-select
                matInput
                ngModel
                placeholder="Tipo de Proyecto"
                tipoproyecto
                name="tipoproyecto"
                required
                #tipoproyectoInput="ngModel"
                (selectionChange)="onSelectTipo($event.value)"
              >
                <mat-option>Seleccione una opcion</mat-option>
                <mat-option
                  *ngFor="
                    let tipo of proyectoService.tiposProyecto;
```

```
                let in = index
              "
              [value]="tipo.id"
            >
              {{ tipo.nombre }}
            </mat-option>
          </mat-select>
          <mat-error>Este campo es requerido</mat-error>
        </mat-form-field>
        <!-- areas prioritarias -->
        <mat-form-field>
          <mat-select
            matInput
            ngModel
            placeholder="Areas Prioritarias"
            areasprioritarias
            name="areasprioritarias"
            required
            #areasprioritariasInput="ngModel"
          >
            <mat-option>Seleccione una opción</mat-option>
            <mat-option
              *ngFor="let area of proyectoService.areas; let in = index"
              [value]="area.id"
            >
              {{ area.nombre }}
            </mat-option>
          </mat-select>
          <mat-error>Este campo es requerido</mat-error>
        </mat-form-field>
        <!-- Costo variable -->
        <div style="align-items:initial">
          Costos Variables
          <input type="checkbox" (change)="toggleEditable($event)" />
        </div>
        <!-- Porcentaje -->
        <mat-form-field *ngIf="costosVariablesInput">
          <input
            type="text"
            matInput
            ngModel
            placeholder="Porcentaje"
            name="porcentaje"
            porcentaje
            required
            #porcentajeInput="ngModel"
            validateMaxMinValue="1,100"
            (keydown)="helperService.validateValue($event)"
          />
          <mat-hint>Valores entre 1 y 10.</mat-hint>
          <mat-error>
            Este campo es requerido (entre 1 y 10).
          </mat-error>
        </mat-form-field>
        <!-- Nombre Proyecto -->
        <mat-form-field>
          <input
            type="text"
            matInput
            ngModel
            placeholder="Nombre"
            name="nombre"
            nombre
            required
            #nombreInput="ngModel"
          />
          <mat-error>Este campo es requerido</mat-error>
        </mat-form-field>
        <!-- Descripcion -->
```

```
              <mat-form-field>
                <textarea
                  cols="30"
                  rows="7"
                  matInput
                  placeholder="Descripcion"
                  name="descripcion"
                  descripcion
                  required
                  ngModel
                  #descripcionInput="ngModel"
                ></textarea>
                <mat-error>Este campo es requerido</mat-error>
              </mat-form-field>
            </div>
            <div>
              <!--  <button  mat-button  matStepperNext  type="button"  color="primary"  mat-raised-
button>Siguiente</button> -->
              <button
                mat-button
                matStepperNext
                type="button"
                color="primary"
                mat-raised-button
                [disabled]="!generalesGroup.valid"
              >
                Siguiente
              </button>
            </div>
          </mat-step>
          <!-- Step 2 : Ubicacion -->
          <mat-step>
            <ng-template matStepLabel>Ubicacion</ng-template>
            <div ngModelGroup="ubicacion" #ubicacionGroup="ngModelGroup">
              <div class="costos-item">
                <div>
                  <input
                    matInput
                    placeholder="Busque la ubicacion"
                    autocorrect="off"
                    autocapitalize="off"
                    spellcheck="off"
                    type="text"
                    class="form-control"
                    #search
                    [formControl]="searchControl"
                  />
                </div>
                <br />
                <agm-map
                  [latitude]="latitude"
                  [longitude]="longitude"
                  [scrollwheel]="false"
                  [zoom]="zoom"
                >
                  <agm-marker
                    [latitude]="latitude"
                    [longitude]="longitude"
                  ></agm-marker>
                </agm-map>
              </div>
              <div class="costos-item">
                <label for=""
                  >Si sabe las coordenadas puede ingresarlas manualmente en los
                  campos de abajo</label
                >
                <mat-form-field>
                  <input
                    type="text"
```

```
                matInput
                [(ngModel)]="latitude"
                ngModel
                placeholder="x"
                name="latitud"
                latitud
                required
                #nombreresponsableInput="ngModel"
              />
              <mat-error>Este campo es requerido</mat-error>
            </mat-form-field>
            <mat-form-field>
              <input
                type="text"
                matInput
                [(ngModel)]="longitude"
                ngModel
                placeholder="y"
                name="longitud"
                longitud
                required
                #apellidoresponsableInput="ngModel"
              />
              <mat-error>Este campo es requerido</mat-error>
            </mat-form-field>
          </div>
        </div>
        <div>
          <button mat-button matStepperPrevious type="button" mat-raised-button>
            Atras
          </button>
          <button
            mat-button
            matStepperNext
            type="button"
            color="primary"
            mat-raised-button
            [disabled]="!ubicacionGroup.valid"
          >
            Siguiente
          </button>
        </div>
      </mat-step>
      <!-- Step 3 : Detalles generales -->
      <mat-step>
        <ng-template matStepLabel>Detalles</ng-template>
        <div
          ngModelGroup="detalles"
          #detallesGroup="ngModelGroup"
          fxLayout="column"
          fxLayoutAlign="center center"
          fxLayoutGap="10px"
        >
          <div class="costos-item">
            <mat-form-field>
              <input
                type="number"
                min="1"
                matInput
                placeholder="Beneficiarios"
                name="beneficiarios"
                required
                ngModel
              />
              <mat-error>Este campo es requerido</mat-error>
            </mat-form-field>
            <mat-form-field>
              <input
                type="text"
```

```
                matInput
                ngModel
                placeholder="Duracion"
                name="duracion"
                duracion
                required
                #duracionInput="ngModel"
              />
              <mat-hint>Expresado en años</mat-hint>
              <mat-error>Este campo es requerido</mat-error>
            </mat-form-field>
            <mat-form-field>
              <input
                type="text"
                matInput
                ngModel
                placeholder="Vida Util"
                name="vidautil"
                vidautil
                required
                #vidautilInput="ngModel"
              />
              <mat-error>Este campo es requerido</mat-error>
            </mat-form-field>
          </div>
          <div class="costos-item">
            <mat-form-field>
              <input
                type="text"
                matInput
                currencyMask
                [options]="{ prefix: '$ ', thousands: ',', decimal: '.' }"
                ngModel
                placeholder="Inversion"
                name="inversion"
                inversion
                required
                #presupuestoInput="ngModel"
              />
              <mat-error>Este campo es requerido</mat-error>
            </mat-form-field>
            <mat-form-field>
              <input
                type="text"
                matInput
                currencyMask
                [options]="{ prefix: '$ ', thousands: ',', decimal: '.' }"
                ngModel
                placeholder="Presupuesto"
                name="presupuesto"
                presupuesto
                required
                #presupuestoInput="ngModel"
              />
              <mat-error>Este campo es requerido</mat-error>
            </mat-form-field>
            <mat-form-field>
              <input
                type="number"
                min="1"
                max="100"
                matInput
                placeholder="Valor Residual"
                [(ngModel)]="valorresidual"
                name="valorresidual"
                required
                ngModel
              />
              <mat-hint>Valores entre 1% y 100%</mat-hint>
```

```
                <mat-error>Este campo es requerido</mat-error>
              </mat-form-field>
            </div>
          </div>
          <div>
            <button mat-button matStepperPrevious type="button" mat-raised-button>
              Atras
            </button>
            <button
              mat-button
              matStepperNext
              type="button"
              color="primary"
              mat-raised-button
              [disabled]="!detallesGroup.valid"
            >
              Siguiente
            </button>
          </div>
        </mat-step>
        <!-- Step 4 : Datos responsable -->
        <mat-step>
          <ng-template matStepLabel>Responsable</ng-template>
          <div
            ngModelGroup="responsable"
            #responsableGroup="ngModelGroup"
            fxLayout="column"
            fxLayoutAlign="center center"
            fxLayoutGap="10px"
          >
            <p>Datos del Responsable de Proyecto:</p>
            <mat-form-field>
              <input
                type="text"
                matInput
                ngModel
                placeholder="Nombre Responsable"
                name="nombreresponsable"
                nombreresponsable
                required
                #nombreresponsableInput="ngModel"
              />
              <mat-error>Este campo es requerido</mat-error>
            </mat-form-field>
            <mat-form-field>
              <input
                type="text"
                matInput
                ngModel
                placeholder="Apellido Responsable"
                name="apellidoresponsable"
                apellidoresponsable
                required
                #apellidoresponsableInput="ngModel"
              />
              <mat-error>Este campo es requerido</mat-error>
            </mat-form-field>
            <mat-form-field>
              <input
                type="text"
                matInput
                ngModel
                placeholder="Cargo"
                name="cargo"
                cargo
                required
                #cargoInput="ngModel"
              />
            </mat-form-field>
```

```
            </div>
            <div>
              <button mat-button matStepperPrevious type="button" mat-raised-button>
                Atras
              </button>
              <button
                mat-button
                matStepperNext
                type="button"
                color="primary"
                mat-raised-button
                [disabled]="!responsableGroup.valid"
              >
                Siguiente
              </button>
            </div>
          </mat-step>
          <!-- Step 5 : Costos -->
          <mat-step>
            <ng-template matStepLabel>Costos</ng-template>
            <div
              ngModelGroup="costos"
              #costosGroup="ngModelGroup"
              class="form-container2"
              fxLayout="column"
              fxLayoutAlign="center center"
              fxLayoutGap="10px"
            >
              <div *ngFor="let costo of proyectoService.costos[0]; let in = index">
                <label class="max-input label-container"
                  >{{ costo.nombre }}:
                </label>
                <mat-form-field class="max-input">
                  <input
                    type="text"
                    currencyMask
                    [options]="{ thousands: ',', decimal: '.' }"
                    matInput
                    placeholder="Monto"
                    [(ngModel)]="costo.monto"
                    name="motnoc{{ in }}"
                    required
                    ngModel
                  />
                  <mat-error>Este campo es requerido</mat-error>
                </mat-form-field>
              </div>
            </div>
            <div>
              <button mat-button matStepperPrevious type="button" mat-raised-button>
                Atras
              </button>
              <button
                mat-button
                matStepperNext
                type="button"
                color="primary"
                mat-raised-button
                [disabled]="!costosGroup.valid"
              >
                Siguiente
              </button>
            </div>
          </mat-step>
          <!-- Step 6 : Beneficios -->
          <mat-step>
            <ng-template matStepLabel>Beneficios</ng-template>
            <div
              ngModelGroup="beneficios"
```

```
            #beneficiosGroup="ngModelGroup"
            class="form-container2"
            fxLayout="column"
            fxLayoutAlign="center center"
            fxLayoutGap="10px"
          >
            <div
              *ngFor="
                let beneficio of proyectoService.beneficios[0];
                let in = index
              "
            >
              <!-- <div class="row-container"> -->
              <label class="max-input  label-container">{{
                beneficio.nombre
              }}</label>
              <br />
              <!-- </div> -->
              <!-- <div class="row-container"> -->
              <mat-form-field class="max-input">
                <input
                  matInput
                  placeholder="Monto"
                  currencyMask
                  [options]="{ thousands: ',', decimal: '.' }"
                  [(ngModel)]="beneficio.monto"
                  name="montob{{ in }}"
                  class="form-control"
                  required
                />
                <!-- <input matInput placeholder="Valor" name="valor{{in}}" required ngModel>  -->
                <mat-error>Este campo es requerido</mat-error>
              </mat-form-field>
              <!-- </div> -->
            </div>
          </div>
          <div>
            <!-- [disabled]="!f.form.valid" -->
            <button mat-button matStepperPrevious type="button" mat-raised-button>
              Atras
            </button>
            <button
              mat-button
              matStepperNext
              type="button"
              color="primary"
              mat-raised-button
              [disabled]="!beneficiosGroup.valid"
              (click)="crearFlujoCaja(f)"
            >
              Siguiente
            </button>
          </div>
        </mat-step>
        <!-- Step 7 : Flujo de caja -->
        <mat-step>
          <ng-template matStepLabel>Flujo de caja</ng-template>
          <div
            ngModelGroup="flujocaja"
            #flujocajaGroup="ngModelGroup"
            class="form-container2"
            fxLayout="column"
            fxLayoutAlign="center center"
            fxLayoutGap="10px"
          >
            <div class="header-flujo-row">
              <div class="header-flujo">Año</div>
              <div class="header-flujo">Monto</div>
              <div class="header-flujo">Costo</div>
```

```
              <div class="header-flujo">Beneficio</div>
            </div>
            <div
              *ngFor="let flujo of proyectoFlujoCaja; let in = index"
              class="header-flujo-row"
            >
              <div class="col-flujo">
                <label class="max-input  label-container">{{ flujo.anio }}</label>
              </div>
              <div class="col-flujo">
                <mat-form-field class="max-input">
                  <input
                    matInput
                    currencyMask
                    [options]="{ thousands: ',', decimal: '.' }"
                    [(ngModel)]="flujo.monto"
                    name="montof{{ in }}"
                    class="form-control"
                    required
                  />
                  <mat-error>Este campo es requerido</mat-error>
                </mat-form-field>
              </div>
              <div class="col-flujo">
                <mat-form-field class="max-input">
                  <input
                    matInput
                    currencyMask
                    [options]="{ thousands: ',', decimal: '.' }"
                    [(ngModel)]="flujo.beneficios"
                    name="beneficiosf{{ in }}"
                    class="form-control"
                    required
                  />
                  <mat-error>Este campo es requerido</mat-error>
                </mat-form-field>
              </div>
              <div class="col-flujo">
                <mat-form-field class="max-input">
                  <input
                    matInput
                    currencyMask
                    [options]="{ thousands: ',', decimal: '.' }"
                    [(ngModel)]="flujo.costos"
                    name="costosf{{ in }}"
                    class="form-control"
                    required
                  />
                  <mat-error>Este campo es requerido</mat-error>
                </mat-form-field>
              </div>
            </div>
          </div>
          <div class="buttons-container" fxLayoutGap="20px">
            <button
              mat-button
              matStepperPrevious
              type="button"
              mat-raised-button
              class="stepper-btn"
            >
              Atras
            </button>
            <button
              mat-button
              matStepperNext
              type="submit"
              color="primary"
              mat-raised-button
```

```
              [disabled]="f.invalid"
              class="stepper-btn"
            >
              Guardar
            </button>
            <button
              mat-button
              type="button"
              (click)="salir()"
              color="primary"
              mat-raised-button
              [disabled]="f.invalid"
              class="stepper-btn"
            >
              Salir
            </button>
          </div>
        </mat-step>
      </mat-vertical-stepper>
    </ng-template>
    <ng-template #desktopScreen>
      <mat-horizontal-stepper
        #stepper="matHorizontalStepper"
        linear
      >
        <!-- Step 1 : Datos generales -->
        <mat-step>
          <ng-template matStepLabel>General</ng-template>
          <div
            ngModelGroup="generales"
            #generalesGroup="ngModelGroup"
            class="form-container"
          >
            <!-- Alcaldia -->
            <mat-form-field>
              <mat-select
                matInput
                ngModel
                placeholder="Alcaldia"
                idAlcaldia
                name="idAlcaldia"
                required
                #idAlcaldiaInput="ngModel"
              >
                <mat-option>Seleccione una opcion</mat-option>
                <mat-option
                  *ngFor="let alcaldia of alcaldias; let in = index"
                  [value]="alcaldia.id"
                >
                  {{ alcaldia.nombre }}
                </mat-option>
              </mat-select>
              <mat-error>Este campo es requerido</mat-error>
            </mat-form-field>
            <!-- tipo proyecto -->
            <mat-form-field>
              <mat-select
                matInput
                ngModel
                placeholder="Tipo de Proyecto"
                tipoproyecto
                name="tipoproyecto"
                required
                #tipoproyectoInput="ngModel"
                (selectionChange)="onSelectTipo($event.value)"
              >
                <mat-option>Seleccione una opcion</mat-option>
                <mat-option
                  *ngFor="
```

```
                let tipo of proyectoService.tiposProyecto;
                let in = index
              "
              [value]="tipo.id"
            >
              {{ tipo.nombre }}
            </mat-option>
          </mat-select>
          <mat-error>Este campo es requerido</mat-error>
        </mat-form-field>
        <!-- areas prioritarias -->
        <mat-form-field>
          <mat-select
            matInput
            ngModel
            placeholder="Areas Prioritarias"
            areasprioritarias
            name="areasprioritarias"
            required
            #areasprioritariasInput="ngModel"
          >
            <mat-option>Seleccione una opción</mat-option>
            <mat-option
              *ngFor="let area of proyectoService.areas; let in = index"
              [value]="area.id"
            >
              {{ area.nombre }}
            </mat-option>
          </mat-select>
          <mat-error>Este campo es requerido</mat-error>
        </mat-form-field>
        <div style="align-items:initial">
          Costos Variables
          <input
            type="checkbox"
            (change)="toggleEditable($event)"
            costoconstante
            name="costoconstante"
            #costoconstanteInput="ngModel"
            ngModel
          />
        </div>
        <mat-form-field *ngIf="costosVariablesInput">
          <input
            type="text"
            matInput
            ngModel
            placeholder="Porcentaje"
            name="porcentaje"
            porcentaje
            required
            #porcentajeInput="ngModel"
            validateMaxMinValue="1,100"
            (keydown)="helperService.validateValue($event)"
          />
          <mat-hint>Valores entre 1 y 10.</mat-hint>
          <mat-error>
            Este campo es requerido (entre 1 y 10).
          </mat-error>
        </mat-form-field>
        <!-- nombre proyecto -->
        <mat-form-field>
          <input
            type="text"
            matInput
            ngModel
            placeholder="Nombre"
            name="nombre"
            nombre
```

```
                required
                #nombreInput="ngModel"
              />
              <mat-error>Este campo es requerido</mat-error>
            </mat-form-field>
            <mat-form-field>
              <textarea
                cols="30"
                rows="7"
                matInput
                placeholder="Descripcion"
                name="descripcion"
                descripcion
                required
                ngModel
                #descripcionInput="ngModel"
              ></textarea>
              <mat-error>Este campo es requerido</mat-error>
            </mat-form-field>
          </div>
          <div>
            <button
              mat-button
              matStepperNext
              type="button"
              color="primary"
              mat-raised-button
              [disabled]="!generalesGroup.valid"
            >
              Siguiente
            </button>
          </div>
        </mat-step>
        <!-- Step 2 : Ubicacion -->
        <mat-step>
          <ng-template matStepLabel>Ubicacion</ng-template>
          <div
            ngModelGroup="ubicacion"
            #ubicacionGroup="ngModelGroup"
            class="costos-container"
          >
            <div class="costos-item">
              <div class="container">
                <div>
                  <input
                    matInput
                    placeholder="Busque la ubicacion"
                    autocorrect="off"
                    autocapitalize="off"
                    spellcheck="off"
                    type="text"
                    class="form-control"
                    #search
                    [formControl]="searchControl"
                  />
                </div>
                <br />
                <agm-map
                  [latitude]="latitude"
                  [longitude]="longitude"
                  [scrollwheel]="false"
                  [zoom]="zoom"
                >
                  <agm-marker
                    [latitude]="latitude"
                    [longitude]="longitude"
                  ></agm-marker>
                </agm-map>
              </div>
```

```
            </div>
            <div class="costos-item">
              <label for=""
                >Si sabe las coordenadas puede ingresarlas manualmente en los
                campos de abajo</label
              >
              <mat-form-field>
                <input
                  type="text"
                  matInput
                  [(ngModel)]="latitude"
                  ngModel
                  placeholder="x"
                  name="latitud"
                  latitud
                  required
                  #nombreresponsableInput="ngModel"
                />
                <mat-error>Este campo es requerido</mat-error>
              </mat-form-field>
              <mat-form-field>
                <input
                  type="text"
                  matInput
                  [(ngModel)]="longitude"
                  ngModel
                  placeholder="y"
                  name="longitud"
                  longitud
                  required
                  #apellidoresponsableInput="ngModel"
                />
                <mat-error>Este campo es requerido</mat-error>
              </mat-form-field>
            </div>
          </div>
          <div>
            <button mat-button matStepperPrevious type="button" mat-raised-button>
              Atras
            </button>
            <button
              mat-button
              matStepperNext
              type="button"
              color="primary"
              mat-raised-button
              [disabled]="!ubicacionGroup.valid"
            >
              Siguiente
            </button>
          </div>
        </mat-step>
        <!-- Step 3 : Detalles generales -->
        <mat-step>
          <ng-template matStepLabel>Detalles</ng-template>
          <div
            ngModelGroup="detalles"
            #detallesGroup="ngModelGroup"
            class="costos-container"
            fxLayout="column"
            fxLayoutAlign="center center"
            fxLayoutGap="10px"
          >
            <div class="costos-item">
              <mat-form-field>
                <input
                  type="number"
                  min="1"
                  matInput
```

```
                placeholder="Beneficiarios"
                name="beneficiarios"
                required
                ngModel
              />
              <mat-error>Este campo es requerido</mat-error>
            </mat-form-field>
            <mat-form-field>
              <input
                type="text"
                matInput
                ngModel
                placeholder="Duracion"
                name="duracion"
                duracion
                required
                #duracionInput="ngModel"
              />
              <mat-hint>Expresado en años</mat-hint>
              <mat-error>Este campo es requerido</mat-error>
            </mat-form-field>
            <mat-form-field>
              <input
                type="text"
                matInput
                ngModel
                placeholder="Vida Util"
                name="vidautil"
                vidautil
                required
                #vidautilInput="ngModel"
              />
              <mat-error>Este campo es requerido</mat-error>
            </mat-form-field>
          </div>
          <div class="costos-item">
            <mat-form-field>
              <input
                type="text"
                matInput
                currencyMask
                [options]="{ prefix: '$ ', thousands: ',', decimal: '.' }"
                ngModel
                placeholder="Inversion"
                name="inversion"
                inversion
                required
                #presupuestoInput="ngModel"
              />
              <mat-error>Este campo es requerido</mat-error>
            </mat-form-field>
            <mat-form-field>
              <input
                type="text"
                matInput
                currencyMask
                [options]="{ prefix: '$ ', thousands: ',', decimal: '.' }"
                ngModel
                placeholder="Presupuesto"
                name="presupuesto"
                presupuesto
                required
                #presupuestoInput="ngModel"
              />
              <mat-error>Este campo es requerido</mat-error>
            </mat-form-field>
            <mat-form-field>
              <input
                type="number"
```

```
                  min="1"
                  max="100"
                  matInput
                  placeholder="Valor Residual"
                  [(ngModel)]="valorresidual"
                  name="valorresidual"
                  required
                  ngModel
                />
                <mat-hint>Valores entre 1% y 100%</mat-hint>
                <mat-error>Este campo es requerido</mat-error>
              </mat-form-field>
            </div>
          </div>
          <div>
            <button mat-button matStepperPrevious type="button" mat-raised-button>
              Atras
            </button>
            <button
              mat-button
              matStepperNext
              type="button"
              color="primary"
              mat-raised-button
              [disabled]="!detallesGroup.valid"
            >
              Siguiente
            </button>
          </div>
        </mat-step>
        <!-- Step 4 : Datos responsable -->
        <mat-step>
          <ng-template matStepLabel>Responsable</ng-template>
          <div
            ngModelGroup="responsable"
            #responsableGroup="ngModelGroup"
            class="form-container"
            fxLayout="column"
            fxLayoutAlign="center center"
            fxLayoutGap="10px"
          >
            <p>Datos del Responsable de Proyecto:</p>
            <mat-form-field>
              <input
                type="text"
                matInput
                ngModel
                placeholder="Nombre Responsable"
                name="nombreresponsable"
                nombreresponsable
                required
                #nombreresponsableInput="ngModel"
              />
              <mat-error>Este campo es requerido</mat-error>
            </mat-form-field>
            <mat-form-field>
              <input
                type="text"
                matInput
                ngModel
                placeholder="Apellido Responsable"
                name="apellidoresponsable"
                apellidoresponsable
                required
                #apellidoresponsableInput="ngModel"
              />
              <mat-error>Este campo es requerido</mat-error>
            </mat-form-field>
            <mat-form-field>
```

```
              <input
                type="text"
                matInput
                ngModel
                placeholder="Cargo"
                name="cargo"
                cargo
                required
                #cargoInput="ngModel"
              />
            </mat-form-field>
          </div>
          <div>
            <button mat-button matStepperPrevious type="button" mat-raised-button>
              Atras
            </button>
            <button
              mat-button
              matStepperNext
              type="button"
              color="primary"
              mat-raised-button
              [disabled]="!responsableGroup.valid"
            >
              Siguiente
            </button>
          </div>
        </mat-step>
        <!-- Step 5 : Costos -->
        <mat-step>
          <ng-template matStepLabel>Costos</ng-template>
          <div
            ngModelGroup="costos"
            #costosGroup="ngModelGroup"
            class="form-container2"
            fxLayout="column"
            fxLayoutAlign="center center"
            fxLayoutGap="10px"
          >
            <div *ngFor="let costo of proyectoService.costos[0]; let in = index">
              <label class="max-input label-container"
                >{{ costo.nombre }}:
              </label>
              <mat-form-field class="max-input">
                <input
                  type="text"
                  currencyMask
                  [options]="{ thousands: ',', decimal: '.' }"
                  matInput
                  placeholder="Monto"
                  [(ngModel)]="costo.monto"
                  name="motnoc{{ in }}"
                  required
                  ngModel
                />
                <mat-error>Este campo es requerido</mat-error>
              </mat-form-field>
            </div>
          </div>
          <div>
            <button mat-button matStepperPrevious type="button" mat-raised-button>
              Atras
            </button>
            <button
              mat-button
              matStepperNext
              type="button"
              color="primary"
              mat-raised-button
```

```
            [disabled]="!costosGroup.valid"
          >
            Siguiente
          </button>
        </div>
      </mat-step>
      <!-- Step 6 : Beneficios -->
      <mat-step>
        <ng-template matStepLabel>Beneficios</ng-template>
        <div
          ngModelGroup="beneficios"
          #beneficiosGroup="ngModelGroup"
          class="form-container2"
          fxLayout="column"
          fxLayoutAlign="center center"
          fxLayoutGap="10px"
        >
          <div
            *ngFor="
              let beneficio of proyectoService.beneficios[0];
              let in = index
            "
          >
            <label class="max-input  label-container">{{
              beneficio.nombre
            }}</label>
            <br />
            <mat-form-field class="max-input">
              <input
                matInput
                placeholder="Monto"
                currencyMask
                [options]="{ thousands: ',', decimal: '.' }"
                [(ngModel)]="beneficio.monto"
                name="montob{{ in }}"
                class="form-control"
                required
              />
              <mat-error>Este campo es requerido</mat-error>
            </mat-form-field>
          </div>
        </div>
        <div>
          <button mat-button matStepperPrevious type="button" mat-raised-button>
            Atras
          </button>
          <button
            mat-button
            matStepperNext
            type="button"
            color="primary"
            mat-raised-button
            [disabled]="!costosGroup.valid"
            (click)="crearFlujoCaja(f)"
          >
            Crear Flujo de Caja
          </button>
        </div>
      </mat-step>
      <!-- Step 7 : Flujo de caja -->
      <mat-step>
        <ng-template matStepLabel>Flujo de caja</ng-template>
        <div
          ngModelGroup="flujocaja"
          #flujocajaGroup="ngModelGroup"
          class="form-container2"
          fxLayout="column"
          fxLayoutAlign="center center"
          fxLayoutGap="10px"
```

```
          >
            <div class="header-flujo-row">
              <div class="header-flujo">Año</div>
              <div class="header-flujo">Monto</div>
              <div class="header-flujo">Costo</div>
              <div class="header-flujo">Beneficio</div>
            </div>
            <div
              *ngFor="let flujo of proyectoFlujoCaja; let in = index"
              class="header-flujo-row"
            >
              <div class="col-flujo">
                <label class="max-input  label-container">{{ flujo.anio }}</label>
              </div>
              <div class="col-flujo">
                <mat-form-field class="max-input">
                  <input
                    matInput
                    currencyMask
                    [options]="{ thousands: ',', decimal: '.' }"
                    [(ngModel)]="flujo.monto"
                    name="montof{{ in }}"
                    class="form-control"
                    required
                  />
                  <mat-error>Este campo es requerido</mat-error>
                </mat-form-field>
              </div>
              <div class="col-flujo">
                <mat-form-field class="max-input">
                  <input
                    matInput
                    currencyMask
                    [options]="{ thousands: ',', decimal: '.' }"
                    [(ngModel)]="flujo.beneficios"
                    name="beneficiosf{{ in }}"
                    class="form-control"
                    required
                  />
                  <mat-error>Este campo es requerido</mat-error>
                </mat-form-field>
              </div>
              <div class="col-flujo">
                <mat-form-field class="max-input">
                  <input
                    matInput
                    currencyMask
                    [options]="{ thousands: ',', decimal: '.' }"
                    [(ngModel)]="flujo.costos"
                    name="costosf{{ in }}"
                    class="form-control"
                    required
                  />
                  <mat-error>Este campo es requerido</mat-error>
                </mat-form-field>
              </div>
            </div>
          </div>
          <div class="buttons-container" fxLayoutGap="20px">
            <button
              mat-button
              matStepperPrevious
              type="button"
              mat-raised-button
              class="stepper-btn"
            >
              Atras
            </button>
            <button
```

```
              mat-button
              matStepperNext
              type="submit"
              color="primary"
              mat-raised-button
              [disabled]="f.invalid"
              class="stepper-btn"
            >
              Guardar
            </button>
            <button
              mat-button
              type="button"
              (click)="salir()"
              color="primary"
              mat-raised-button
              [disabled]="f.invalid"
              class="stepper-btn"
            >
              Salir
            </button>
          </div>
        </mat-step>
      </mat-horizontal-stepper>
    </ng-template>
  </form>
  <mat-spinner
    *ngIf="isLoading$ | async"
    align="center"
    [diameter]="48"
    style="margin:0 auto;"
  ></mat-spinner>
```

c. create.component.ts

```
  /// <reference types="@types/googlemaps" />
  import {
    Component,
    OnInit,
    ViewChild,
    ElementRef,
    NgZone
  } from "@angular/core";
  import { NgForm, FormControl } from "@angular/forms";
  import {
    ProyectoService,
    UIService,
    AlcaldiaService,
    ReglaService,
    HelperService
  } from "../../services";
  import { Store } from "@ngrx/store";
  import { Observable } from "rxjs";
  import { Proyecto, TipoProyecto, Alcaldia, Area } from "../../models";
  import * as fromRoot from "../../store/app.reducers";
  import * as UI from "../../store/ui/ui.actions";
  import { MapsAPILoader } from "@agm/core";
  import { Router } from "@angular/router";

  @Component({
    selector: "app-create",
    templateUrl: "./create.component.html",
    styleUrls: ["./create.component.css"]
  })
  export class CreateComponent implements OnInit {
    public latitude: number;
```

```
      public longitude: number;
      public searchControl: FormControl;
      public zoom: number;
      public selectedValue;
      @ViewChild("search")
      public searchElementRef: ElementRef;
      public proyectoSelected: TipoProyecto[] = [];
      isLoading$: Observable<boolean>;
      public valorresidual = 10;
      public alcaldias: Alcaldia[] = [];
      mobileScreen = false;
      desktopScreen = true;
      mHeight: any;
      mWidth: any;
      costosVariablesInput = false;
      proyectoId: string;
      areas: Area[] = [];
      proyectoFlujoCaja: any;
      currentProjectId = null;
      stepperIndex:          any          =          (+localStorage.getItem("stepperIndex")          ?
+localStorage.getItem("stepperIndex"): 0);

      constructor(
        private proyectoService: ProyectoService,
        private reglaService: ReglaService,
        private uiService: UIService,
        private store: Store<fromRoot.State>,
        private mapsAPILoader: MapsAPILoader,
        private ngZone: NgZone,
        private alcaldiaService: AlcaldiaService,
        private helperService: HelperService,
        private router: Router
      ) {
        this.mHeight = window.screen.height;
        this.mWidth = window.screen.width;
        this.desktopScreen = false;
        if (this.mWidth <= 700 || (this.mWidth == 768 && this.mHeight == 1024)) {
          this.mobileScreen = true;
        }
      }

      ngOnInit() {
        this.getAlcaldias();
        this.proyectoService.getTipoProyectos();
        localStorage.removeItem("stepperIndex");
        // localStorage.removeItem("stepperIndex");
        this.zoom = 4;
        this.latitude = 39.8282;
        this.longitude = -98.5795;

        //create search FormControl
        this.searchControl = new FormControl();
        //set current position
        this.setCurrentPosition();
        //load Places Autocomplete
        this.mapsAPILoader.load().then(() => {
          let autocomplete = new google.maps.places.Autocomplete(
            this.searchElementRef.nativeElement,
            {
              types: ["address"]
            }
          );
          autocomplete.addListener("place_changed", () => {
            this.ngZone.run(() => {
              //get the place result
              let place: google.maps.places.PlaceResult = autocomplete.getPlace();

              //verify result
              if (place.geometry === undefined || place.geometry === null) {
```

```
            return;
          }

          //set latitude, longitude and zoom
          this.latitude = place.geometry.location.lat();
          this.longitude = place.geometry.location.lng();
          this.zoom = 12;
        });
      });
    });
  }

  private setCurrentPosition() {
    if ("geolocation" in navigator) {
      navigator.geolocation.getCurrentPosition(position => {
        this.latitude = position.coords.latitude;
        this.longitude = position.coords.longitude;
        this.zoom = 12;
      });
    }
  }

  onSubmit(form: NgForm) {
    if (form.valid &&  this.currentProjectId) {
        let proyectoObject = this.proyectoService.crearObjeto(form,this.proyectoFlujoCaja);
        console.log("this.proyectoFlujoCaja",this.proyectoFlujoCaja)
        console.log("this.currentProjectId",this.currentProjectId)
        this.store.dispatch(new UI.StartLoading());
        this.proyectoService
            .recalculoFlujoCaja(this.currentProjectId, proyectoObject[0])
            .subscribe(
            data => {
                this.store.dispatch(new UI.StopLoading());
                this.uiService.showSnackbar(
                "Flujo de caja actualizado!",
                null,
                15000,
                "custom-snack-bar"
                );
                console.log("data444",data)
                localStorage.setItem("stepperIndex", '6');
                this.router.navigate(["proyectos/listar/"+this.currentProjectId]);
            },
            error => {
                this.store.dispatch(new UI.StopLoading());
                this.uiService.showSnackbar(
                "Problemas actualizando flujo de caja",
                null,
                3000,
                "custom-snack-bar-error"
                );
            }
        );
    }
  }
  crearFlujoCaja(form: NgForm) {
    if (form.valid) {
      let proyectoObject = this.proyectoService.crearObjeto(form, this.proyectoFlujoCaja);
      this.proyectoService.crear(proyectoObject[0]).subscribe(
        data => {
          this.proyectoFlujoCaja = data['flujoCaja'];
          this.currentProjectId = data['id'];
          this.store.dispatch(new UI.StopLoading());
          this.uiService.showSnackbar(
            "Creando flujo de caja...",
            null,
            3000,
            "custom-snack-bar"
          );
```

```
            },
            error => {
              this.store.dispatch(new UI.StopLoading());
              this.uiService.showSnackbar(
                "Problemas procesando proyecto!",
                null,
                3000,
                "custom-snack-bar-error"
              );
            }
          );
        }
      }
      onSelectTipo (id){
          this.proyectoService.onSelectTipo(id);
      }
      getAlcaldias() {
        return this.alcaldiaService
          .getAllAlcaldias()
          .subscribe((data: Alcaldia[]) => {
            this.alcaldias = data;
          });
      }
      toggleEditable(event) {
        if (event.target.checked) {
          this.costosVariablesInput = true;
        } else {
          this.costosVariablesInput = false;
        }
      }
      validateValue(event: KeyboardEvent) {
        this.helperService.validateValue(event);
      }
      salir(){
        localStorage.removeItem("stepperIndex");
        this.router.navigate(["proyectos/listar"]);
      }
    }
```

2. list

a. delete

i. delete-proyecto.component.html

```
    <h1 mat-dialog-title>Eliminando Proyecto: {{ passedData.proyecto.nombre }}</h1>
    <mat-dialog-content>
      <mat-list>
        <mat-list-item> Esta seguro que quiere eliminarlo? </mat-list-item>
      </mat-list>
    </mat-dialog-content>
    <mat-dialog-actions>
      <button  *ngIf="!(isLoading$  |  async)"  type="button"  mat-raised-button  color="primary"
(click)="onSubmit()">Delete</button>
      <button mat-button [mat-dialog-close]="false">Cancelar</button>
      <mat-spinner *ngIf="isLoading$ | async"></mat-spinner>
    </mat-dialog-actions>
```

ii. delete-proyecto.component.ts

```
    import { Component, Inject, OnInit } from "@angular/core";
```

```
import { MAT_DIALOG_DATA, MatDialogRef } from "@angular/material";
import { NgForm, FormGroup, FormControl, Validators } from "@angular/forms";
import { ProyectoService, UIService } from "../../../services";
import { Observable } from "rxjs";
import { Store } from "@ngrx/store";

import * as fromRoot from "../../../store/app.reducers";
import * as UI from "../../../store/ui/ui.actions";

@Component({
  selector: "app-delete-proyecto",
  templateUrl: "delete-proyecto.component.html",
  styleUrls: ["../list.component.css"]
})
export class DeleteProyectoComponent implements OnInit {
  isLoading$: Observable<boolean>;

  constructor(
    public dialogRef: MatDialogRef<DeleteProyectoComponent>,
    @Inject(MAT_DIALOG_DATA) public passedData: any,
    private proyectoService: ProyectoService,
    private uiService: UIService,
    private store: Store<fromRoot.State>
  ) {}

  ngOnInit() {
    this.isLoading$ = this.store.select(fromRoot.getIsLoading);
  }

  onSubmit() {
    this.proyectoService.deleteProyecto(this.passedData.id).subscribe(
      data => {
        this.store.dispatch(new UI.StopLoading());
        this.uiService.showSnackbar(
          "Proyecto eliminado!",
          null,
          3000,
          "custom-snack-bar"
        );
        this.dialogRef.close();
      },
      error => {
        this.store.dispatch(new UI.StopLoading());
        this.uiService.showSnackbar(
          "Problemas eliminando proyecto",
          null,
          3000,
          "custom-snack-bar-error"
        );
        this.dialogRef.close();
      }
    );
  }
}
```

b. edit

i. edit-proyecto.component.html

```
<h3 *ngIf="!(isLoading$ | async)">
  <a routerLink="/configuracion/listar" title="Atrás"
    ><mat-icon class="icon-back">arrow_back</mat-icon></a
  >Editando Proyecto: {{ detailData.nombre }}
</h3>
<form #f="ngForm" (ngSubmit)="onSubmit(f)" *ngIf="!(isLoading$ | async)">
```

```
    <ng-template [ngIf]="mobileScreen" [ngIfElse]="desktopScreen">
      <mat-vertical-stepper #stepper="matVerticalStepper" linear [selectedIndex]="stepperIndex">
        <!-- Step 1 : Datos generales -->
        <mat-step>
          <ng-template matStepLabel>Datos generales</ng-template>
          <div
            ngModelGroup="generales"
            #generalesGroup="ngModelGroup"
            class="form-container"
          >
            <mat-form-field>
              <mat-select
                matInput
                ngModel
                placeholder="Tipo de Proyecto"
                tipoproyecto
                name="tipoproyecto"
                required
                #tipoproyectoInput="ngModel"
                [(ngModel)]="detailData.tipoProyecto.id"
                (selectionChange)="onSelectTipo($event.value)"
              >
                <mat-option>Seleccione una opcion</mat-option>
                <mat-option
                  *ngFor="let tipo of tiposProyecto; let in = index"
                  [value]="tipo.id"
                >
                  {{ tipo.nombre }}
                </mat-option>
              </mat-select>
              <mat-error>Este campo es requerido</mat-error>
            </mat-form-field>
            <!-- areas prioritarias -->
            <mat-form-field>
              <mat-select
                matInput
                ngModel
                placeholder="Areas Prioritarias"
                areasprioritarias
                name="areasprioritarias"
                required
                #areasprioritariasInput="ngModel"
                [(ngModel)]="detailData.areaPrioritaria.id"
              >
                <mat-option>Seleccione una opción</mat-option>
                <mat-option
                  *ngFor="let area of areas; let in = index"
                  [value]="area.id"
                >
                  {{ area.nombre }}
                </mat-option>
              </mat-select>
              <mat-error>Este campo es requerido</mat-error>
            </mat-form-field>
            <div style="align-items:initial">
              Costos Variables
              <input
                type="checkbox"
                (change)="toggleEditable($event)"
                name="costoConstante"
                costoConstante
                #costoConstanteInput="ngModel"
                [(ngModel)]="detailData.costoConstante"
              />
            </div>
            <mat-form-field
              *ngIf="costosVariablesInput || detailData.costoConstante"
            >
              <input
```

```
                type="text"
                matInput
                ngModel
                placeholder="Porcentaje"
                name="porcentaje"
                porcentaje
                required
                #porcentajeInput="ngModel"
                [(ngModel)]="detailData.porcentajeCostoVariable"
                validateMaxMinValue="1,100"
                (keydown)="helperService.validateValue($event)"
              />
              <mat-hint>Valores entre 1 y 10.</mat-hint>
              <mat-error>
                Este campo es requerido (entre 1 y 10).
              </mat-error>
            </mat-form-field>
            <mat-form-field *ngIf="!(isLoading$ | async)">
              <input
                type="text"
                matInput
                [(ngModel)]="detailData.nombre"
                ngModel
                placeholder="Nombre"
                name="nombre"
                nombre
                required
                #nombreInput="ngModel"
              />
              <mat-error>Este campo es requerido</mat-error>
            </mat-form-field>
            <mat-form-field>
              <textarea
                cols="30"
                rows="7"
                matInput
                placeholder="Descripcion"
                name="descripcion"
                [(ngModel)]="detailData.descripcion"
                descripcion
                required
                ngModel
                #descripcionInput="ngModel"
              ></textarea>
              <mat-error>Este campo es requerido</mat-error>
            </mat-form-field>
          </div>
          <div>
            <button
              class="stepper-btn"
              mat-button
              matStepperNext
              type="button"
              color="primary"
              mat-raised-button
              [disabled]="!generalesGroup.valid"
            >
              Siguiente
            </button>
          </div>
        </mat-step>
        <!-- Step 2 : Ubicacion -->
        <mat-step>
          <ng-template matStepLabel>Ubicacion</ng-template>
          <div
            ngModelGroup="ubicacion"
            #ubicacionGroup="ngModelGroup"
            class="costos-container"
          >
```

```
              <div class="costos-item">
                <div class="container">
                  <div>
                    <input
                      matInput
                      placeholder="Busque la ubicacion"
                      autocorrect="off"
                      autocapitalize="off"
                      spellcheck="off"
                      type="text"
                      class="form-control"
                      #search
                      [formControl]="searchControl"
                    />
                  </div>
                  <br />
                  <agm-map
                    [latitude]="latitude"
                    [longitude]="longitude"
                    [scrollwheel]="false"
                    [zoom]="zoom"
                  >
                    <agm-marker
                      [latitude]="latitude"
                      [longitude]="longitude"
                    ></agm-marker>
                  </agm-map>
                </div>
              </div>
              <div class="costos-item">
                <label for=""
                  >Si sabe las coordenadas puede ingresarlas manualmente en los
                  campos de abajo</label
                >
                <br />
                <mat-form-field>
                  <input
                    type="text"
                    matInput
                    [(ngModel)]="latitude"
                    ngModel
                    placeholder="x"
                    name="latitud"
                    latitud
                    required
                    #latitudInput="ngModel"
                  />
                  <mat-error>Este campo es requerido</mat-error>
                </mat-form-field>
                <mat-form-field>
                  <input
                    type="text"
                    matInput
                    [(ngModel)]="longitude"
                    ngModel
                    placeholder="y"
                    name="longitud"
                    longitud
                    required
                    #longitudInput="ngModel"
                  />
                  <mat-error>Este campo es requerido</mat-error>
                </mat-form-field>
              </div>
            </div>
            <div>
              <button mat-button matStepperPrevious type="button" mat-raised-button class="stepper-
btn">
                Atras
```

```
            </button>
            <button
              mat-button
              matStepperNext
              type="button"
              color="primary"
              mat-raised-button
              [disabled]="!ubicacionGroup.valid"
              class="stepper-btn"
            >
              Siguiente
            </button>
          </div>
        </mat-step>
        <!-- Step 3 : Detalles generales -->
        <mat-step>
          <ng-template matStepLabel>Detalles</ng-template>
          <div
            ngModelGroup="detalles"
            #detallesGroup="ngModelGroup"
            class="costos-container"
            fxLayout="column"
            fxLayoutAlign="center center"
            fxLayoutGap="10px"
          >
            <div class="costos-item">
              <mat-form-field>
                <input
                  type="number"
                  min="1"
                  matInput
                  placeholder="Beneficiarios"
                  [(ngModel)]="detailData.beneficiarios"
                  name="beneficiarios"
                  required
                  ngModel
                />
                <mat-error>Este campo es requerido</mat-error>
              </mat-form-field>
              <mat-form-field>
                <input
                  type="text"
                  matInput
                  [(ngModel)]="detailData.duracion"
                  ngModel
                  placeholder="Duracion"
                  name="duracion"
                  duracion
                  required
                  #duracionInput="ngModel"
                />
                <mat-error>Este campo es requerido</mat-error>
              </mat-form-field>
              <mat-form-field>
                <input
                  type="text"
                  matInput
                  [(ngModel)]="detailData.vidaUtil"
                  ngModel
                  placeholder="Vida Util"
                  name="vidautil"
                  vidautil
                  required
                  #vidautilInput="ngModel"
                />
                <mat-error>Este campo es requerido</mat-error>
              </mat-form-field>
            </div>
            <div class="costos-item">
```

```
                <mat-form-field>
                  <input
                    type="text"
                    matInput
                    currencyMask
                    [options]="{ prefix: '$ ', thousands: ',', decimal: '.' }"
                    [(ngModel)]="detailData.inversion"
                    ngModel
                    placeholder="Inversion"
                    name="inversion"
                    inversion
                    required
                    #presupuestoInput="ngModel"
                  />
                  <mat-error>Este campo es requerido</mat-error>
                </mat-form-field>
                <mat-form-field>
                  <input
                    type="text"
                    matInput
                    currencyMask
                    [options]="{ prefix: '$ ', thousands: ',', decimal: '.' }"
                    [(ngModel)]="detailData.presupuesto"
                    ngModel
                    placeholder="Presupuesto"
                    name="presupuesto"
                    presupuesto
                    required
                    #presupuestoInput="ngModel"
                  />
                  <mat-error>Este campo es requerido</mat-error>
                </mat-form-field>
                <mat-form-field>
                  <input
                    type="number"
                    min="1"
                    max="100"
                    matInput
                    placeholder="Valor Residual"
                    [(ngModel)]="detailData.valorResidual"
                    name="valorresidual"
                    required
                    ngModel
                  />
                  <mat-hint>Valores entre 1% y 100%</mat-hint>
                  <mat-error>Este campo es requerido</mat-error>
                </mat-form-field>
              </div>
            </div>
            <div>
              <button mat-button matStepperPrevious type="button" mat-raised-button class="stepper-
btn">
                Atras
              </button>
              <button
                mat-button
                matStepperNext
                type="button"
                color="primary"
                mat-raised-button
                [disabled]="!detallesGroup.valid"
                class="stepper-btn"
              >
                Siguiente
              </button>
            </div>
          </mat-step>
          <!-- Step 4 : Datos responsable -->
          <mat-step>
```

```
            <ng-template matStepLabel>Responsable</ng-template>
            <div
              ngModelGroup="responsable"
              #responsableGroup="ngModelGroup"
              class="form-container"
              fxLayout="column"
              fxLayoutAlign="center center"
              fxLayoutGap="10px"
            >
              <p>Datos del Responsable de Proyecto:</p>
              <mat-form-field>
                <input
                  type="text"
                  matInput
                  [(ngModel)]="detailData.responsable.nombre"
                  ngModel
                  placeholder="Nombre Responsable"
                  name="nombreresponsable"
                  nombreresponsable
                  required
                  #nombreresponsableInput="ngModel"
                />
                <mat-error>Este campo es requerido</mat-error>
              </mat-form-field>
              <mat-form-field>
                <input
                  type="text"
                  matInput
                  [(ngModel)]="detailData.responsable.apellido"
                  ngModel
                  placeholder="Apellido Responsable"
                  name="apellidoresponsable"
                  apellidoresponsable
                  required
                  #apellidoresponsableInput="ngModel"
                />
                <mat-error>Este campo es requerido</mat-error>
              </mat-form-field>
              <mat-form-field>
                <input
                  type="text"
                  matInput
                  [(ngModel)]="detailData.responsable.cargo"
                  ngModel
                  placeholder="Cargo"
                  name="cargo"
                  cargo
                  required
                  #cargoInput="ngModel"
                />
              </mat-form-field>
            </div>
            <div>
              <button mat-button matStepperPrevious type="button" mat-raised-button class="stepper-
btn">
                Atras
              </button>
              <button
                mat-button
                matStepperNext
                type="button"
                color="primary"
                mat-raised-button
                [disabled]="!responsableGroup.valid"
                class="stepper-btn"
              >
                Siguiente
              </button>
            </div>
```

```
          </mat-step>
          <!-- Step 5 : Costos -->
          <mat-step>
            <ng-template matStepLabel>Costos</ng-template>
            <div
              ngModelGroup="costos"
              #costosGroup="ngModelGroup"
              class="form-container2"
              fxLayout="column"
              fxLayoutAlign="center center"
              fxLayoutGap="10px"
            >
              <div *ngFor="let costo of costos; let in = index">
                <label class="max-input label-container"
                  >{{ costo.nombre }}:
                </label>
                <mat-form-field class="max-input">
                  <input
                    type="text"
                    currencyMask
                    [options]="{ thousands: ',', decimal: '.' }"
                    matInput
                    placeholder="Monto"
                    [(ngModel)]="costo.monto"
                    name="motnoc{{ in }}"
                    required
                    ngModel
                  />
                  <mat-error>Este campo es requerido</mat-error>
                </mat-form-field>
              </div>
            </div>
            <div>
              <button  mat-button  matStepperPrevious  type="button"  mat-raised-button  class="stepper-
btn">
                Atras
              </button>
              <button
                mat-button
                matStepperNext
                type="button"
                color="primary"
                mat-raised-button
                [disabled]="!costosGroup.valid"
                class="stepper-btn"
              >
                Siguiente
              </button>
            </div>
          </mat-step>
          <!-- Step 6 : Beneficios -->
          <mat-step>
            <ng-template matStepLabel>Beneficios</ng-template>
            <div
              ngModelGroup="beneficios"
              #beneficiosGroup="ngModelGroup"
              class="form-container2"
              fxLayout="column"
              fxLayoutAlign="center center"
              fxLayoutGap="10px"
            >
              <div *ngFor="let beneficio of beneficios; let in = index">
                <label class="max-input  label-container">{{
                  beneficio.nombre
                }}</label>
                <br />
                <mat-form-field class="max-input">
                  <input
                    matInput
```

```
                    placeholder="Monto"
                    currencyMask
                    [options]="{ thousands: ',', decimal: '.' }"
                    [(ngModel)]="beneficio.monto"
                    name="montob{{ in }}"
                    class="form-control"
                    required
                  />
                  <mat-error>Este campo es requerido</mat-error>
                </mat-form-field>
              </div>
            </div>

            <div>
              <button mat-button matStepperPrevious type="button" mat-raised-button class="stepper-
btn">
                Atras
              </button>
              <button
                mat-button
                matStepperNext
                type="button"
                color="primary"
                mat-raised-button
                [disabled]="!beneficiosGroup.valid"
                (click)="actualizarFlujoCaja(f)"
                class="stepper-btn"
              >
                Siguiente
              </button>
            </div>
          </mat-step>
          <!-- Step 7 : Flujo de caja -->
          <mat-step>
            <ng-template matStepLabel>Flujo de caja</ng-template>
            <div
              ngModelGroup="flujocaja"
              #flujocajaGroup="ngModelGroup"
              class="form-container2"
              fxLayout="column"
              fxLayoutAlign="center center"
              fxLayoutGap="10px"
            >
            <mat-table [dataSource]="dataSource">
                <ng-container matColumnDef="anio">
                    <mat-header-cell *matHeaderCellDef>Año</mat-header-cell>
                    <mat-cell *matCellDef="let element; let in = index">{{
                    element.anio
                    }}</mat-cell>
                </ng-container>
                <ng-container matColumnDef="monto">
                    <mat-header-cell *matHeaderCellDef>Monto</mat-header-cell>
                    <mat-cell *matCellDef="let element; let in = index">
                        <mat-form-field class="max-input">
                            <input
                                matInput
                                currencyMask
                                [options]="{ thousands: ',', decimal: '.' }"
                                [(ngModel)]="element.monto"
                                name="montoFlujo{{ in }}"
                                class="form-control"
                                required
                            />
                            <mat-error>Este campo es requerido</mat-error>
                        </mat-form-field>
                    </mat-cell>
                </ng-container>
                <ng-container matColumnDef="costos">
                    <mat-header-cell *matHeaderCellDef>Costos</mat-header-cell>
```

```
                    <mat-cell *matCellDef="let element; let in = index">
                        <mat-form-field class="max-input">
                            <input
                                matInput
                                currencyMask
                                [options]="{ thousands: ',', decimal: '.' }"
                                [(ngModel)]="element.costos"
                                name="costosFlujo{{ in }}"
                                class="form-control"
                                required
                            />
                            <mat-error>Este campo es requerido</mat-error>
                        </mat-form-field>
                    </mat-cell>
                </ng-container>
                <ng-container matColumnDef="beneficios">
                    <mat-header-cell *matHeaderCellDef>Beneficios</mat-header-cell>
                    <mat-cell *matCellDef="let element; let in = index">
                        <mat-form-field class="max-input">
                            <input
                                matInput
                                currencyMask
                                [options]="{ thousands: ',', decimal: '.' }"
                                [(ngModel)]="element.beneficios"
                                name="beneficiosFlujo{{ in }}"
                                class="form-control"
                                required
                            />
                            <mat-error>Este campo es requerido</mat-error>
                        </mat-form-field>
                    </mat-cell>
                </ng-container>

                <mat-header-row
                    *matHeaderRowDef="displayedColumns"
                ></mat-header-row>
                <mat-row
                    *matRowDef="let row; columns: displayedColumns"
                    (click)="selection.toggle(row)"
                ></mat-row>
            </mat-table>
            </div>
            <div class="buttons-container" fxLayoutGap="20px">
              <button mat-button matStepperPrevious type="button" mat-raised-button class="stepper-
btn">
                Atras
              </button>
              <button
                mat-button
                matStepperNext
                type="submit"
                color="primary"
                mat-raised-button
                [disabled]="f.invalid"
                class="stepper-btn"
              >
                Guardar
              </button>
            </div>
          </mat-step>
        </mat-vertical-stepper>
      </ng-template>
      <ng-template #desktopScreen>
        <mat-horizontal-stepper #stepper="matHorizontalStepper" linear [selectedIndex]="stepperIndex">
          <!-- Step 1 : Datos generales -->
          <mat-step>
            <ng-template matStepLabel>Datos generales</ng-template>
            <div
              ngModelGroup="generales"
```

```
            #generalesGroup="ngModelGroup"
            class="form-container"
          >
            <!-- tipo de proyecto -->
            <mat-form-field>
              <mat-select
                matInput
                ngModel
                placeholder="Tipo de Proyecto"
                tipoproyecto
                name="tipoproyecto"
                required
                #tipoproyectoInput="ngModel"
                [(ngModel)]="detailData.tipoProyecto.id"
                (selectionChange)="onSelectTipo($event.value)"
              >
                <mat-option>Seleccione una opcion</mat-option>
                <mat-option
                  *ngFor="
                    let tipo of proyectoService.tiposProyecto;
                    let in = index
                  "
                  [value]="tipo.id"
                >
                  {{ tipo.nombre }}
                </mat-option>
              </mat-select>
              <mat-error>Este campo es requerido</mat-error>
            </mat-form-field>
            <!-- Nombre proyecto -->
            <mat-form-field *ngIf="!(isLoading$ | async)">
              <input
                type="text"
                matInput
                [(ngModel)]="detailData.nombre"
                ngModel
                placeholder="Nombre"
                name="nombre"
                nombre
                required
                #nombreInput="ngModel"
              />
              <mat-error>Este campo es requerido</mat-error>
            </mat-form-field>
            <!-- areas prioritarias -->
            <mat-form-field>
              <mat-select
                matInput
                ngModel
                placeholder="Areas Prioritarias"
                areasprioritarias
                name="areasprioritarias"
                required
                #areasprioritariasInput="ngModel"
                [(ngModel)]="detailData.areaPrioritaria.id"
              >
                <mat-option>Seleccione una opción</mat-option>
                <mat-option
                  *ngFor="let area of areas; let in = index"
                  [value]="area.id"
                >
                  {{ area.nombre }}
                </mat-option>
              </mat-select>
              <mat-error>Este campo es requerido</mat-error>
            </mat-form-field>
            <!-- Costos variables -->
            <div style="align-items:initial">
              Costos Variables
```

```
              <input
                type="checkbox"
                (change)="toggleEditable($event)"
                name="costoConstante"
                costoConstante
                #costoConstanteInput="ngModel"
                [(ngModel)]="detailData.costoConstante"
              />
            </div>
            <!-- porcentaje costo variable -->
            <mat-form-field
              *ngIf="costosVariablesInput || detailData.costoConstante"
            >
              <input
                type="text"
                matInput
                ngModel
                placeholder="Porcentaje"
                name="porcentaje"
                porcentaje
                required
                #porcentajeInput="ngModel"
                [(ngModel)]="detailData.porcentajeCostoVariable"
                validateMaxMinValue="1,100"
                (keydown)="helperService.validateValue($event)"
              />
              <mat-hint>Valores entre 1 y 10.</mat-hint>
              <mat-error>
                Este campo es requerido (entre 1 y 10).
              </mat-error>
            </mat-form-field>
            <!-- descriptcion -->
            <mat-form-field>
              <textarea
                cols="30"
                rows="7"
                matInput
                placeholder="Descripcion"
                name="descripcion"
                [(ngModel)]="detailData.descripcion"
                descripcion
                required
                ngModel
                #descripcionInput="ngModel"
              ></textarea>
              <mat-error>Este campo es requerido</mat-error>
            </mat-form-field>
          </div>
          <div>
            <button
              mat-button
              matStepperNext
              type="button"
              color="primary"
              mat-raised-button
              [disabled]="!generalesGroup.valid"
              class="stepper-btn"
            >
              Siguiente
            </button>
          </div>
        </mat-step>
        <!-- Step 2 : Ubicacion -->
        <mat-step>
          <ng-template matStepLabel>Ubicacion</ng-template>
          <div
            ngModelGroup="ubicacion"
            #ubicacionGroup="ngModelGroup"
            class="costos-container"
```

```
            >
              <div class="costos-item">
                <div class="container">
                  <div>
                    <input
                      matInput
                      placeholder="Busque la ubicacion"
                      autocorrect="off"
                      autocapitalize="off"
                      spellcheck="off"
                      type="text"
                      class="form-control"
                      #search
                      [formControl]="searchControl"
                    />
                  </div>
                  <br />
                  <agm-map
                    [latitude]="latitude"
                    [longitude]="longitude"
                    [scrollwheel]="false"
                    [zoom]="zoom"
                  >
                    <agm-marker
                      [latitude]="latitude"
                      [longitude]="longitude"
                    ></agm-marker>
                  </agm-map>
                </div>
              </div>
              <div class="costos-item">
                <label for=""
                  >Si sabe las coordenadas puede ingresarlas manualmente en los
                  campos de abajo</label
                >
                <br />
                <mat-form-field>
                  <input
                    type="text"
                    matInput
                    [(ngModel)]="latitude"
                    ngModel
                    placeholder="x"
                    name="latitud"
                    latitud
                    required
                    #latitudInput="ngModel"
                  />
                  <mat-error>Este campo es requerido</mat-error>
                </mat-form-field>
                <mat-form-field>
                  <input
                    type="text"
                    matInput
                    [(ngModel)]="longitude"
                    ngModel
                    placeholder="y"
                    name="longitud"
                    longitud
                    required
                    #longitudInput="ngModel"
                  />
                  <mat-error>Este campo es requerido</mat-error>
                </mat-form-field>
              </div>
            </div>
            <div>
              <button mat-button matStepperPrevious type="button" mat-raised-button class="stepper-
btn">
```

```
            Atras
          </button>
          <button
            mat-button
            matStepperNext
            type="button"
            color="primary"
            mat-raised-button
            [disabled]="!ubicacionGroup.valid"
            class="stepper-btn"
          >
            Siguiente
          </button>
        </div>
      </mat-step>
      <!-- Step 3 : Detalles generales -->
      <mat-step>
        <ng-template matStepLabel>Detalles</ng-template>
        <div
          ngModelGroup="detalles"
          #detallesGroup="ngModelGroup"
          class="costos-container"
          fxLayout="column"
          fxLayoutAlign="center center"
          fxLayoutGap="10px"
        >
          <div class="costos-item">
            <mat-form-field>
              <input
                type="number"
                min="1"
                matInput
                placeholder="Beneficiarios"
                [(ngModel)]="detailData.beneficiarios"
                name="beneficiarios"
                required
                ngModel
              />
              <mat-error>Este campo es requerido</mat-error>
            </mat-form-field>
            <mat-form-field>
              <input
                type="text"
                matInput
                [(ngModel)]="detailData.duracion"
                ngModel
                placeholder="Duracion"
                name="duracion"
                duracion
                required
                #duracionInput="ngModel"
              />
              <mat-error>Este campo es requerido</mat-error>
            </mat-form-field>
            <mat-form-field>
              <input
                type="text"
                matInput
                [(ngModel)]="detailData.vidaUtil"
                ngModel
                placeholder="Vida Util"
                name="vidautil"
                vidautil
                required
                #vidautilInput="ngModel"
              />
              <mat-error>Este campo es requerido</mat-error>
            </mat-form-field>
          </div>
```

```
              <div class="costos-item">
                <mat-form-field>
                  <input
                    type="text"
                    matInput
                    currencyMask
                    [options]="{ prefix: '$ ', thousands: ',', decimal: '.' }"
                    [(ngModel)]="detailData.inversion"
                    ngModel
                    placeholder="Inversion"
                    name="inversion"
                    inversion
                    required
                    #presupuestoInput="ngModel"
                  />
                  <mat-error>Este campo es requerido</mat-error>
                </mat-form-field>
                <mat-form-field>
                  <input
                    type="text"
                    matInput
                    currencyMask
                    [options]="{ prefix: '$ ', thousands: ',', decimal: '.' }"
                    [(ngModel)]="detailData.presupuesto"
                    ngModel
                    placeholder="Presupuesto"
                    name="presupuesto"
                    presupuesto
                    required
                    #presupuestoInput="ngModel"
                  />
                  <mat-error>Este campo es requerido</mat-error>
                </mat-form-field>
                <mat-form-field>
                  <input
                    type="number"
                    min="1"
                    max="100"
                    matInput
                    placeholder="Valor Residual"
                    [(ngModel)]="detailData.valorResidual"
                    name="valorresidual"
                    required
                    ngModel
                  />
                  <mat-hint>Valores entre 1% y 100%</mat-hint>
                  <mat-error>Este campo es requerido</mat-error>
                </mat-form-field>
              </div>
            </div>
            <div>
              <button  mat-button  matStepperPrevious  type="button"  mat-raised-button  class="stepper-
btn">
                Atras
              </button>
              <button
                mat-button
                matStepperNext
                type="button"
                color="primary"
                mat-raised-button
                [disabled]="!detallesGroup.valid"
                class="stepper-btn"
              >
                Siguiente
              </button>
            </div>
          </mat-step>
          <!-- Step 4 : Datos responsable -->
```

```
          <mat-step>
            <ng-template matStepLabel>Responsable</ng-template>
            <div
              ngModelGroup="responsable"
              #responsableGroup="ngModelGroup"
              class="form-container"
              fxLayout="column"
              fxLayoutAlign="center center"
              fxLayoutGap="10px"
            >
              <p>Datos del Responsable de Proyecto:</p>
              <mat-form-field>
                <input
                  type="text"
                  matInput
                  [(ngModel)]="detailData.responsable.nombre"
                  ngModel
                  placeholder="Nombre Responsable"
                  name="nombreresponsable"
                  nombreresponsable
                  required
                  #nombreresponsableInput="ngModel"
                />
                <mat-error>Este campo es requerido</mat-error>
              </mat-form-field>
              <mat-form-field>
                <input
                  type="text"
                  matInput
                  [(ngModel)]="detailData.responsable.apellido"
                  ngModel
                  placeholder="Apellido Responsable"
                  name="apellidoresponsable"
                  apellidoresponsable
                  required
                  #apellidoresponsableInput="ngModel"
                />
                <mat-error>Este campo es requerido</mat-error>
              </mat-form-field>
              <mat-form-field>
                <input
                  type="text"
                  matInput
                  [(ngModel)]="detailData.responsable.cargo"
                  ngModel
                  placeholder="Cargo"
                  name="cargo"
                  cargo
                  required
                  #cargoInput="ngModel"
                />
              </mat-form-field>
            </div>
            <div>
              <button mat-button matStepperPrevious type="button" mat-raised-button class="stepper-
btn">
                Atras
              </button>
              <button
                mat-button
                matStepperNext
                type="button"
                color="primary"
                mat-raised-button
                [disabled]="!responsableGroup.valid"
                class="stepper-btn"
              >
                Siguiente
              </button>
```

```
            </div>
          </mat-step>
          <!-- Step 5 : Costos -->
          <mat-step>
            <ng-template matStepLabel>Costos</ng-template>
            <div
              ngModelGroup="costos"
              #costosGroup="ngModelGroup"
              class="form-container2"
              fxLayout="column"
              fxLayoutAlign="center center"
              fxLayoutGap="10px"
            >
              <div *ngFor="let costo of costos; let in = index">
                <label class="max-input label-container"
                  >{{ costo.nombre }}:
                </label>
                <mat-form-field class="max-input">
                  <input
                    type="text"
                    currencyMask
                    [options]="{ thousands: ',', decimal: '.' }"
                    matInput
                    placeholder="Monto"
                    [(ngModel)]="costo.monto"
                    name="motnoc{{ in }}"
                    required
                    ngModel
                  />
                  <mat-error>Este campo es requerido</mat-error>
                </mat-form-field>
              </div>
            </div>
            <div>
              <button  mat-button  matStepperPrevious  type="button"  mat-raised-button  class="stepper-
btn">
                Atras
              </button>
              <button
                mat-button
                matStepperNext
                type="button"
                color="primary"
                mat-raised-button
                [disabled]="!costosGroup.valid"
                class="stepper-btn"
              >
                Siguiente
              </button>
            </div>
          </mat-step>
          <!-- Step 6 : Beneficios -->
          <mat-step>
            <ng-template matStepLabel>Beneficios</ng-template>
            <div
              ngModelGroup="beneficios"
              #beneficiosGroup="ngModelGroup"
              class="form-container2"
              fxLayout="column"
              fxLayoutAlign="center center"
              fxLayoutGap="10px"
            >
              <div *ngFor="let beneficio of beneficios; let in = index">
                <label class="max-input  label-container">{{
                  beneficio.nombre
                }}</label>
                <br />
                <mat-form-field class="max-input">
                  <input
```

```
                    matInput
                    placeholder="Monto"
                    currencyMask
                    [options]="{ thousands: ',', decimal: '.' }"
                    [(ngModel)]="beneficio.monto"
                    name="montob{{ in }}"
                    class="form-control"
                    required
                   />
                   <mat-error>Este campo es requerido</mat-error>
                </mat-form-field>
              </div>
            </div>

            <div>
              <button mat-button matStepperPrevious type="button" mat-raised-button class="stepper-
btn">
                Atras
              </button>
              <button
                mat-button
                matStepperNext
                type="button"
                color="primary"
                mat-raised-button
                [disabled]="!beneficiosGroup.valid"
                (click)="actualizarFlujoCaja(f)"
                class="stepper-btn"
              >
                Siguiente
              </button>
            </div>
          </mat-step>
            <!-- Step 7 : Flujo de caja -->
            <mat-step>
                <ng-template matStepLabel>Flujo de caja</ng-template>
                <div
                  ngModelGroup="flujocaja"
                  #flujocajaGroup="ngModelGroup"
                  class="form-container2"
                  fxLayout="column"
                  fxLayoutAlign="center center"
                  fxLayoutGap="10px"
                >
                    <div class="header-flujo-row">
                        <div class="header-flujo">Año</div>
                        <div class="header-flujo">Monto</div>
                        <div class="header-flujo">Costo</div>
                        <div class="header-flujo">Beneficio</div>
                    </div>
                    <div *ngFor="let flujo of proyectoFlujoCaja; let in = index" class="header-flujo-
row">
                        <div class="col-flujo">
                            <label class="max-input  label-container">{{
                                    flujo.anio
                            }}</label>
                        </div>
                        <div class="col-flujo">
                            <mat-form-field class="max-input">
                                <input
                                matInput
                                currencyMask
                                [options]="{ thousands: ',', decimal: '.' }"
                                [(ngModel)]="flujo.monto"
                                name="montof{{ in }}"
                                class="form-control"
                                required
                                />
                                <mat-error>Este campo es requerido</mat-error>
```

```
                        </mat-form-field>
                    </div>
                    <div class="col-flujo">
                        <mat-form-field class="max-input">
                            <input
                            matInput
                            currencyMask
                            [options]="{ thousands: ',', decimal: '.' }"
                            [(ngModel)]="flujo.beneficios"
                            name="beneficiosf{{ in }}"
                            class="form-control"
                            required
                            />
                            <mat-error>Este campo es requerido</mat-error>
                        </mat-form-field>
                    </div>
                    <div class="col-flujo">
                        <mat-form-field class="max-input">
                            <input
                            matInput
                            currencyMask
                            [options]="{ thousands: ',', decimal: '.' }"
                            [(ngModel)]="flujo.costos"
                            name="costosf{{ in }}"
                            class="form-control"
                            required
                            />
                            <mat-error>Este campo es requerido</mat-error>
                        </mat-form-field>
                    </div>
                </div>
            </div>
            <div class="buttons-container" fxLayoutGap="20px">
              <button mat-button matStepperPrevious type="button" mat-raised-button class="stepper-
btn">
                Atras
              </button>
              <button
                mat-button
                matStepperNext
                type="submit"
                color="primary"
                mat-raised-button
                [disabled]="f.invalid"
                class="stepper-btn"
              >
                Guardar
              </button>
              <button
                mat-button
                type="button"
                (click)="salir()"
                color="primary"
                mat-raised-button
                [disabled]="f.invalid"
                class="stepper-btn"
              >
                Salir
              </button>
            </div>
          </mat-step>
        </mat-horizontal-stepper>
      </ng-template>
    </form>
  <mat-spinner
    *ngIf="isLoading$ | async"
    align="center"
    [diameter]="48"
    style="margin:0 auto;"
```

```
></mat-spinner>
```

ii. edit-proyecto.component.ts

```
/// <reference types="@types/googlemaps" />
import {
  Component,
  OnInit,
  ViewChild,
  ElementRef,
  NgZone
} from "@angular/core";
import { NgForm, Validators, FormControl } from "@angular/forms";
import {
  ProyectoService,
  UIService,
  ReglaService,
  HelperService
} from "../../../services";
import { Observable } from "rxjs";
import { Store } from "@ngrx/store";

import * as fromRoot from "../../../store/app.reducers";
import * as UI from "../../../store/ui/ui.actions";

import {
  Proyecto,
  TipoProyecto,
  Costo,
  Beneficio,
  Area
} from "../../../models";
import { Router, ActivatedRoute } from "@angular/router";

import { MapsAPILoader } from "@agm/core";

@Component({
  selector: "app-edit-proyecto",
  templateUrl: "edit-proyecto.component.html",
  styleUrls: ["../../create/create.component.css"]
})
export class EditProyectoComponent implements OnInit {
  public latitude: number;
  public longitude: number;
  public searchControl: FormControl;
  public zoom: number;
  @ViewChild("search")
  public searchElementRef: ElementRef;

  proyectoArray: any = [];
  isLoading$: Observable<boolean>;
  public costos: Costo[] = [];
  public beneficios: Beneficio[] = [];
  public costos1: Costo[] = [];
  public beneficios1: Beneficio[] = [];
  public tiposProyecto: TipoProyecto[] = [];
  public tipoId: number;
  public proyectoObject: Proyecto[] = [];
  public detailData: any;
  id: string;
  mobileScreen = false;
  desktopScreen = true;
  mHeight: any;
  mWidth: any;
  areas: Area[] = [];
  proyectoFlujoCaja: any;
```

```
      costosVariablesInput = false;
      idAlcaldia: string;


      stepperIndex:          any          =          (+localStorage.getItem("stepperIndex")          ?
+localStorage.getItem("stepperIndex"): 0);
      constructor(
        private proyectoService: ProyectoService,
        private uiService: UIService,
        private reglaService: ReglaService,
        private store: Store<fromRoot.State>,
        private mapsAPILoader: MapsAPILoader,
        private route: ActivatedRoute,
        private router: Router,
        private ngZone: NgZone,
        private helperService: HelperService
      ) {
        this.mHeight = window.screen.height;
        this.mWidth = window.screen.width;
        this.desktopScreen = false;
        if (this.mWidth <= 700 || (this.mWidth == 768 && this.mHeight == 1024)) {
          this.mobileScreen = true;
        }
        this.id = this.route.snapshot.paramMap.get("id");
      }

      ngOnInit() {
        this.isLoading$ = this.store.select(fromRoot.getIsLoading);
        this.store.dispatch(new UI.StartLoading());
        this.getDetails();
        this.zoom = 12;
        this.searchControl = new FormControl();
        this.areas = this.reglaService.getAreasList();
        setTimeout(() => {
            if(localStorage.getItem("stepperIndex")){
                this.stepperIndex = +localStorage.getItem("stepperIndex");
            }
        }, 1500);
      }

      onSubmit(form: NgForm) {
        //temporal mientras se define como se va a manejar el flujo de caja si es modiicado
        if (form.valid) {
          let proyectoObject = this.proyectoService.crearObjeto(form,this.proyectoFlujoCaja);
          this.store.dispatch(new UI.StartLoading());
          this.proyectoService
            .recalculoFlujoCaja(this.detailData.id, proyectoObject[0])
            .subscribe(
              data => {
                this.store.dispatch(new UI.StopLoading());
                this.uiService.showSnackbar(
                  "Flujo de caja actualizado!",
                  null,
                  15000,
                  "custom-snack-bar"
                );
                localStorage.setItem("stepperIndex", '6');
                let currentUrl = this.router.url;
                this.router.navigateByUrl('/', {skipLocationChange: true}).then(() => {
                    this.router.navigate([currentUrl]);
                });
              },
              error => {
                this.store.dispatch(new UI.StopLoading());
                this.uiService.showSnackbar(
                  "Problemas actualizando flujo de caja",
                  null,
                  3000,
                  "custom-snack-bar-error"
```

```
            );
          }
        );
    }
  }
  actualizarFlujoCaja(form: NgForm) {
    //   temporal mientras se define como se va a manejar el flujo de caja si es modiicado
    if (form.valid) {
      let proyectoObject = this.proyectoService.crearObjeto(form, this.proyectoFlujoCaja);
      if(proyectoObject[0].costos == null){
        proyectoObject[0].costos = this.costos;
      }
      if(proyectoObject[0].beneficios == null){
        proyectoObject[0].beneficios = this.beneficios;
      }
      if(proyectoObject[0].costoConstante == null){
        proyectoObject[0].costoConstante = this.costosVariablesInput;
      }
      if(proyectoObject[0].idAlcaldia == null){
        proyectoObject[0].idAlcaldia = this.idAlcaldia;
      }

    // this.store.dispatch(new UI.StartLoading());
      this.proyectoService
        .updateProyecto(this.detailData.id, proyectoObject[0])
        .subscribe(
            data => {
            this.proyectoFlujoCaja = data['flujoCaja'];
            this.store.dispatch(new UI.StopLoading());
            this.uiService.showSnackbar(
              "Actualizando flujo de caja",
              null,
              15000,
              "custom-snack-bar"
            );
            //this.router.navigate(["/proyectos/listar"]);
          },
          error => {
            this.store.dispatch(new UI.StopLoading());
            this.uiService.showSnackbar(
              "Problemas actualizando flujo de caja",
              null,
              3000,
              "custom-snack-bar-error"
            );
          }
        );
    }
  }
  onSelectTipo(id) {
    if (this.tipoId === id) {
      this.costos = this.detailData.costos;
      this.beneficios = this.detailData.beneficios;
    } else {
      this.costos = [];
      this.beneficios = [];
      this.costos = this.proyectoService.tiposProyecto.filter(
        x => x.id == id
      )[0].costos;
      this.beneficios = this.proyectoService.tiposProyecto.filter(
        x => x.id == id
      )[0].beneficios;
    }
  }
  getDetails() {
    this.proyectoService.getTipoProyectos();
    this.proyectoService
      .getProyecto(this.id)
      .subscribe((proyectoData: Proyecto[]) => {
```

```
          this.detailData = proyectoData;
          this.costos = this.detailData.costos;
          this.beneficios = this.detailData.beneficios;
          this.tipoId = this.detailData.tipoProyecto.id;
          this.costosVariablesInput = this.detailData.costoConstante;
          this.latitude = this.detailData.ubicacion.lat;
          this.longitude = this.detailData.ubicacion.lng;
          this.proyectoFlujoCaja = this.detailData.flujoCaja;
          this.idAlcaldia = this.detailData.idAlcaldia;
          this.proyectoFlujoCaja = this.detailData.flujoCaja;
         // this.loadMap();
          this.store.dispatch(new UI.StopLoading());
        });
    }
    loadMap() {
      this.mapsAPILoader.load().then(() => {
        let autocomplete = new google.maps.places.Autocomplete(
          this.searchElementRef.nativeElement,
          {
            types: ["address"]
          }
        );
        autocomplete.addListener("place_changed", () => {
          this.ngZone.run(() => {
            //get the place result
            let place: google.maps.places.PlaceResult = autocomplete.getPlace();
            //verify result
            if (place.geometry === undefined || place.geometry === null) {
              return;
            }
            //set latitude, longitude and zoom
            this.latitude = place.geometry.location.lat();
            this.longitude = place.geometry.location.lng();
            this.zoom = 12;
          });
        });
      });
    }
    toggleEditable(event) {
      if (event.target.checked) {
        this.costosVariablesInput = true;
      } else {
        this.costosVariablesInput = false;
      }
    }
    salir(){
      localStorage.removeItem("stepperIndex");
      this.router.navigate(["proyectos/listar"]);
    }
  }
```

c. list.component.css

```
  h3{
      color: #3f51b5;
      font-weight: bold;
      text-align: center;
    }
```

d. list.component.html

```
  <h3>Listado de Proyectos</h3>
  <mat-table [dataSource]="dataSource" matSort>
```

```
    <ng-container matColumnDef="nombre">
      <mat-header-cell mat-header-cell *matHeaderCellDef>
        Nombre
      </mat-header-cell>
      <mat-cell *matCellDef="let element"> {{ element.nombre }} </mat-cell>
    </ng-container>

    <ng-container matColumnDef="descripcion">
      <mat-header-cell mat-header-cell *matHeaderCellDef>
        Descripcion
      </mat-header-cell>
      <mat-cell *matCellDef="let element"> {{ element.descripcion }} </mat-cell>
    </ng-container>

    <ng-container matColumnDef="actions">
      <mat-header-cell *matHeaderCellDef mat-sort-header>Actions</mat-header-cell>
      <mat-cell *matCellDef="let element">
        <a routerLink="{{element.id}}"><mat-icon>edit</mat-icon></a>
        <button mat-icon-button (click)="onDelete(element.id)">
          <mat-icon>delete</mat-icon>
        </button>
      </mat-cell>
    </ng-container>
    <mat-header-row *matHeaderRowDef="displayedColumns"></mat-header-row>
    <mat-row *matRowDef="let row; columns: displayedColumns"></mat-row>
  </mat-table>
  <div *ngIf="notResults" align="center">No se encontraron resultados</div>

  <mat-paginator [pageSize]="5" [pageSizeOptions]="[5, 10, 20]"></mat-paginator>
  <mat-spinner
    *ngIf="(isLoading$ | async)"
    align="center"
    [diameter]="48"
    style="margin:0 auto;"
  ></mat-spinner>
```

e. list.component.ts

```
  import {
    Component,
    OnInit,
    ViewChild,
    AfterViewInit,
    ChangeDetectorRef
  } from "@angular/core";
  import { Observable } from "rxjs";
  import { ProyectoService } from "../../services";
  import {
    MatTableDataSource,
    MatSort,
    MatPaginator,
    MatDialog
  } from "@angular/material";
  import { Proyecto } from "../../models";
  import { EditProyectoComponent } from "./edit/edit-proyecto.component";
  import { DeleteProyectoComponent } from "./delete/delete-proyecto.component";
  import { Store } from "@ngrx/store";
  import * as fromRoot from "../../store/app.reducers";
  import * as UI from "../../store/ui/ui.actions";

  @Component({
    selector: "app-list",
    templateUrl: "./list.component.html",
    styleUrls: ["./list.component.css"]
  })
  export class ListComponent implements OnInit, AfterViewInit {
```

```
    proyecto: Proyecto;
    displayedColumns = ["nombre", "descripcion", "actions"];
    dataSource = new MatTableDataSource<Proyecto>();

    @ViewChild(MatSort) sort: MatSort;
    @ViewChild(MatPaginator) paginator: MatPaginator;
    isLoading$: Observable<boolean>;
    notResults = false;

    constructor(
      private proyectoService: ProyectoService,
      private dialog: MatDialog,
      private changeDetectorRefs: ChangeDetectorRef,
      private store: Store<fromRoot.State>
    ) {}

    ngOnInit() {
      this.isLoading$ = this.store.select(fromRoot.getIsLoading);
      this.onRefresh();
    }

    ngAfterViewInit() {
      this.dataSource.sort = this.sort;
      this.dataSource.paginator = this.paginator;
    }
    doFilter(filterValue: string) {
      this.dataSource.filter = filterValue.trim().toLowerCase();
    }

    onRefresh() {
      this.store.dispatch(new UI.StartLoading());

      this.proyectoService.getAllProyectos().subscribe((proyecto: Proyecto[]) => {
        this.dataSource.data = proyecto;
        if(this.dataSource.data.length == 0){
          this.notResults = true;
        }
        this.store.dispatch(new UI.StopLoading());

      });
    }
    onDelete(id) {
      this.proyectoService
        .getProyecto(id)
        .subscribe((proyectoData: Proyecto[]) => {
          const dialogRef = this.dialog
            .open(DeleteProyectoComponent, {
              data: {
                proyecto: proyectoData,
                id: id
              }
            })
            .afterClosed()
            .subscribe(result => {
              this.onRefresh();
            });
        });
    }
  }
```

3. proyectos.module.ts

```
import { NgModule, LOCALE_ID} from '@angular/core';
import { CommonModule } from '@angular/common';
import { RouterModule, Routes } from '@angular/router';
```

```
import { CreateComponent } from './create/create.component';
import { ListComponent } from './list/list.component';

import { EditProyectoComponent } from './list/edit/edit-proyecto.component';
import { DeleteProyectoComponent } from './list/delete/delete-proyecto.component';

import { MaterialModule } from '../material.module';
import { ReactiveFormsModule, FormsModule } from '@angular/forms';
import { ProyectoService } from '../services';
import { ReglaService } from '../services';
import { CurrencyMaskModule } from "ng2-currency-mask";
import { AgmCoreModule } from '@agm/core';

const routes: Routes = [
  { path: "crear", component: CreateComponent },
  { path: "listar", component: ListComponent },
  { path: "listar/:id", component: EditProyectoComponent }

];

@NgModule({
  imports: [
    CommonModule,
    MaterialModule,
    ReactiveFormsModule,
    FormsModule,
    CurrencyMaskModule,
    AgmCoreModule.forRoot({
      apiKey: "AIzaSyArt3ySEzc5bgcGGfRuNWmJ5eNjGAFhNZ8",
      libraries: ["places"]
    }),
    RouterModule.forChild(routes)
  ],
  declarations: [
    CreateComponent,
    ListComponent,
    EditProyectoComponent,
    DeleteProyectoComponent
  ],
  entryComponents: [
    DeleteProyectoComponent
  ],
  providers: [ProyectoService, ReglaService]
})
export class ProyectosModule { }
```

ix. reglas

1. create

a. create.component.css

```
.label-container {
  display: flex;
  flex-direction: column;
  align-items: flex-start;
  margin-top: 20px;
}

.mat-form-field {
  width: 100%;
}

.form-container {
```

```
    display: flex;
    flex-direction: column;
    /* align-items: center; */
  }

  .mat-icon-button {
    top: 10px;
    float: right;
  }

  ::ng-deep .mat-horizontal-stepper-header {
    pointer-events: none !important;
  }

  ::ng-deep .mat-snack-bar-container {
    background-color: #36a5e6;
    color: #fff;
  }

  h3 {
    color: #3f51b5;
    font-weight: bold;
    text-align: center;
  }
  .max-input {
    margin-left: 5px;
  }

  .max-input .mat-form-field {
    width: 10%;
  }
  /*
    ##Device = Desktops
    ##Screen = 1281px to higher resolution desktops
  */

  @media (min-width: 1281px) {
    .mat-form-field {
      width: 30%;
    }
    .max-input {
      margin-left: 50px;
    }

    .max-input .mat-form-field {
      width: 50%;
    }
    .form-container {
      display: flex;
      flex-direction: column;
      align-items: center;
    }

    .row-container {
      display: flex;
      flex-direction: row;
      align-items: center;
    }
  }

  /*
    ##Device = Laptops, Desktops
    ##Screen = B/w 1025px to 1280px
  */

  @media (min-width: 1025px) and (max-width: 1280px) {
    .mat-form-field {
      width: 30%;
    }
```

```
    .max-input {
      margin-left: 50px;
    }

    .max-input .mat-form-field {
      width: 50%;
    }
    .form-container {
      display: flex;
      flex-direction: column;
      align-items: center;
    }

    .row-container {
      display: flex;
      flex-direction: row;
      align-items: center;
    }
  }
```

b. create.component.html

```
  <h3>Reglas</h3>
  <form #f="ngForm" (ngSubmit)="onSubmit(f)">
    <ng-template [ngIf]="mobileScreen" [ngIfElse]="desktopScreen">
      <mat-vertical-stepper #stepper="matVerticalStepper" linear>
        <mat-step>
          <ng-template matStepLabel>Datos generales</ng-template>
          <div
            ngModelGroup="generales"
            #generalesGroup="ngModelGroup"
            class="form-container"
            fxLayout="column"
            fxLayoutAlign="center center"
            fxLayoutGap="10px"
          >
            <mat-form-field *ngIf="(currentUser$ | async)?.user.rol == 'root'">
              <mat-select
                matInput
                ngModel
                placeholder="Alcaldia"
                idAlcaldia
                name="idAlcaldia"
                required
                #idAlcaldiaInput="ngModel"
              >
                <mat-option>Seleccione una opcion</mat-option>
                <mat-option
                  *ngFor="let alcaldia of alcaldias; let in = index"
                  [value]="alcaldia.id"
                >
                  {{ alcaldia.nombre }}
                </mat-option>
              </mat-select>
              <mat-error>Este campo es requerido</mat-error>
            </mat-form-field>
            <mat-form-field>
              <input
                type="text"
                matInput
                [(ngModel)]="nombre"
                ngModel
                placeholder="Nombre"
                name="nombre"
                nombre
                required
```

```
                #nombreInput="ngModel"
              />
              <mat-error>Este campo es requerido</mat-error>
            </mat-form-field>
            <mat-form-field>
              <textarea
                cols="30"
                rows="7"
                matInput
                placeholder="Descripcion"
                name="descripcion"
                [(ngModel)]="descripcion"
                descripcion
                required
                ngModel
                #descripcionInput="ngModel"
              ></textarea>
              <mat-error>Este campo es requerido</mat-error>
            </mat-form-field>
          </div>
          <div>
            <button
              mat-button
              matStepperNext
              type="button"
              color="primary"
              mat-raised-button
              [disabled]="!generalesGroup.valid"
            >
              Siguiente
            </button>
          </div>
        </mat-step>
        <mat-step>
          <ng-template matStepLabel>Areas Prioritarias</ng-template>
          <div
            ngModelGroup="areas"
            #areasGroup="ngModelGroup"
            class="form-container2"
            fxLayout="column"
            fxLayoutAlign="center center"
            fxLayoutGap="10px"
          >
            <div *ngFor="let area of areas; let in = index">
              <label class="max-input label-container">{{ area.nombre }}: </label>
              <mat-form-field class="max-input">
                <input
                  type="number"
                  min="1"
                  max="10"
                  matInput
                  placeholder="Puntaje"
                  [(ngModel)]="area.puntaje"
                  name="puntaje{{in}}"
                  #areaId
                  maxlength="2"
                  required
                  ngModel
                  (keydown)="validateValue($event)"
                  validateMaxMinValue= "1,10"
                />
                <mat-hint>Valores entre 1 y 10.</mat-hint>
                <mat-error>
                    Este campo es requerido (entre 1 y 10).
                </mat-error>
              </mat-form-field>
            </div>
          </div>
          <div>
```

```
            <button mat-button matStepperPrevious type="button" mat-raised-button>
              Atras
            </button>
            <button
              mat-button
              matStepperNext
              type="button"
              color="primary"
              mat-raised-button
              [disabled]="!areasGroup.valid"
            >
              Siguiente
            </button>
          </div>
        </mat-step>
        <mat-step>
          <ng-template matStepLabel>Indicadores</ng-template>
          <div
            ngModelGroup="indicadores"
            #indicadoresGroup="ngModelGroup"
            class="form-container"
            fxLayout="column"
            fxLayoutAlign="center center"
            fxLayoutGap="10px"
          >
            <div *ngFor="let indicador of indicadores; let in = index">
              <div class="row-container">
                <label class="label-container">{{ indicador.nombre }}</label>
                <br />
              </div>
              <div *ngFor="let valores of indicador.valores; let i = index">
                <div class="row-container">
                  <mat-form-field>
                    <input
                      matInput
                      placeholder="Valor"
                      [(ngModel)]="valores.valor"
                      name="valor{{in}}-{{i}}"
                      class="form-control"
                      required
                    />
                    <!--
                      <input matInput placeholder="Valor" name="valor{{in}}" required ngModel>
                    -->
                    <mat-error>Este campo es requerido</mat-error>
                  </mat-form-field>
                  <mat-form-field class="max-input">
                    <input
                      matInput
                      placeholder="Min"
                      [(ngModel)]="valores.min"
                      name="min{{in}}-{{i}}"
                      class="form-control"
                      required
                    />
                    <!--
                      <input matInput placeholder="Min" name="min{{in}}" required ngModel>
                    -->
                    <mat-error>Este campo es requerido</mat-error>
                  </mat-form-field>
                  <mat-form-field class="max-input">
                    <!--
                      <input matInput placeholder="Max" name="max{{in}}" required ngModel>
                    -->
                    <input
                      matInput
                      placeholder="Max"
                      [(ngModel)]="valores.max"
                      name="max{{in}}-{{i}}"
```

```
                    class="form-control"
                    required
                  />
                  <mat-error>Este campo es requerido</mat-error>
                </mat-form-field>
                <button
                  *ngIf="i == 0"
                  mat-icon-button
                  type="button"
                  (click)="onAddIndicador(in)"
                >
                  <mat-icon>add</mat-icon>
                </button>
                <button
                  *ngIf="i > 0"
                  mat-icon-button
                  type="button"
                  (click)="onRemoveIndicador(in, i)"
                >
                  <mat-icon>delete</mat-icon>
                </button>
              </div>
            </div>
          </div>
        </div>

        <div>
          <!-- [disabled]="!f.form.valid" -->
          <button mat-button matStepperPrevious type="button" mat-raised-button>
            Atras
          </button>
          <button
            mat-button
            matStepperNext
            type="button"
            color="primary"
            mat-raised-button
            [disabled]="!indicadoresGroup.valid"
          >
            Siguiente
          </button>
        </div>
      </mat-step>
      <mat-step>
        <ng-template matStepLabel>Finalizar</ng-template>
        <div class="buttons-container" fxLayoutGap="20px">
          <button mat-button matStepperPrevious type="button" mat-raised-button>
            Atras
          </button>
          <button mat-button (click)="stepper.reset()" mat-raised-button>
            Reset
          </button>
          <button
            mat-button
            matStepperNext
            type="submit"
            color="primary"
            mat-raised-button
            [disabled]="f.invalid"
          >
            Guardar
          </button>
        </div>
      </mat-step>
    </mat-vertical-stepper>
  </ng-template>
  <ng-template #desktopScreen>
    <mat-horizontal-stepper #stepper="matHorizontalStepper" linear>
      <mat-step>
```

```
            <ng-template matStepLabel>Datos generales</ng-template>
            <div
              ngModelGroup="generales"
              #generalesGroup="ngModelGroup"
              class="form-container"
              fxLayout="column"
              fxLayoutAlign="center center"
              fxLayoutGap="10px"
            >
              <mat-form-field *ngIf="(currentUser$ | async)?.user.rol == 'root'">
                <mat-select
                  matInput
                  ngModel
                  placeholder="Alcaldia"
                  idAlcaldia
                  name="idAlcaldia"
                  required
                  #idAlcaldiaInput="ngModel"
                >
                  <mat-option>Seleccione una opcion</mat-option>
                  <mat-option
                    *ngFor="let alcaldia of alcaldias; let in = index"
                    [value]="alcaldia.id"
                  >
                    {{ alcaldia.nombre }}
                  </mat-option>
                </mat-select>
                <mat-error>Este campo es requerido</mat-error>
              </mat-form-field>
              <mat-form-field>
                <input
                  type="text"
                  matInput
                  [(ngModel)]="nombre"
                  ngModel
                  placeholder="Nombre"
                  name="nombre"
                  nombre
                  required
                  #nombreInput="ngModel"
                />
                <mat-error>Este campo es requerido</mat-error>
              </mat-form-field>
              <mat-form-field>
                <textarea
                  cols="30"
                  rows="7"
                  matInput
                  placeholder="Descripcion"
                  name="descripcion"
                  [(ngModel)]="descripcion"
                  descripcion
                  required
                  ngModel
                  #descripcionInput="ngModel"
                ></textarea>
                <mat-error>Este campo es requerido</mat-error>
              </mat-form-field>
            </div>
            <div>
              <button
                mat-button
                matStepperNext
                type="button"
                color="primary"
                mat-raised-button
                [disabled]="!generalesGroup.valid"
              >
                Siguiente
```

```
            </button>
          </div>
        </mat-step>
        <mat-step>
          <ng-template matStepLabel>Areas Prioritarias</ng-template>
          <div
            ngModelGroup="areas"
            #areasGroup="ngModelGroup"
            class="form-container2"
            fxLayout="column"
            fxLayoutAlign="center center"
            fxLayoutGap="10px"
          >
            <div *ngFor="let area of areas; let in = index">
              <label class="max-input label-container">{{ area.nombre }}: </label>
              <mat-form-field class="max-input">
                <input
                  type="number"
                  min="1"
                  max="10"
                  matInput
                  placeholder="Puntaje"
                  [(ngModel)]="area.puntaje"
                  name="puntaje{{in}}"
                  #areaId
                  maxlength="2"
                  required
                  ngModel
                  (keydown)="validateValue($event)"
                  validateMaxMinValue= "1,10"
                />
                <mat-hint>Valores entre 1 y 10.</mat-hint>
                <mat-error>
                    Este campo es requerido (entre 1 y 10).
                </mat-error>
              </mat-form-field>
            </div>
          </div>
          <div>
            <button mat-button matStepperPrevious type="button" mat-raised-button>
              Atras
            </button>
            <button
              mat-button
              matStepperNext
              type="button"
              color="primary"
              mat-raised-button
              [disabled]="!areasGroup.valid"
            >
              Siguiente
            </button>
          </div>
        </mat-step>
        <mat-step>
          <ng-template matStepLabel>Indicadores</ng-template>
          <div
            ngModelGroup="indicadores"
            #indicadoresGroup="ngModelGroup"
            class="form-container"
            fxLayout="column"
            fxLayoutAlign="center center"
            fxLayoutGap="10px"
          >
            <div *ngFor="let indicador of indicadores; let in = index">
              <div class="row-container">
                <label class="label-container">{{ indicador.nombre }}</label>
                <br />
              </div>
```

```
              <div *ngFor="let valores of indicador.valores; let i = index">
                <div class="row-container">
                  <mat-form-field>
                    <input
                      matInput
                      placeholder="Valor"
                      [(ngModel)]="valores.valor"
                      name="valor{{in}}-{{i}}"
                      class="form-control"
                      required
                    />
                    <!--
                      <input matInput placeholder="Valor" name="valor{{in}}" required ngModel>
                    -->
                    <mat-error>Este campo es requerido</mat-error>
                  </mat-form-field>
                  <mat-form-field class="max-input">
                    <input
                      matInput
                      placeholder="Min"
                      [(ngModel)]="valores.min"
                      name="min{{in}}-{{i}}"
                      class="form-control"
                      required
                    />
                    <!--
                      <input matInput placeholder="Min" name="min{{in}}" required ngModel>
                    -->
                    <mat-error>Este campo es requerido</mat-error>
                  </mat-form-field>
                  <mat-form-field class="max-input">
                    <!--
                      <input matInput placeholder="Max" name="max{{in}}" required ngModel>
                    -->
                    <input
                      matInput
                      placeholder="Max"
                      [(ngModel)]="valores.max"
                      name="max{{in}}-{{i}}"
                      class="form-control"
                      required
                    />
                    <mat-error>Este campo es requerido</mat-error>
                  </mat-form-field>
                  <button
                    *ngIf="i == 0"
                    mat-icon-button
                    type="button"
                    (click)="onAddIndicador(in)"
                  >
                    <mat-icon>add</mat-icon>
                  </button>
                  <button
                    *ngIf="i > 0"
                    mat-icon-button
                    type="button"
                    (click)="onRemoveIndicador(in, i)"
                  >
                    <mat-icon>delete</mat-icon>
                  </button>
                </div>
              </div>
            </div>

            <div>
              <!-- [disabled]="!f.form.valid" -->
              <button mat-button matStepperPrevious type="button" mat-raised-button>
                Atras
```

```
            </button>
            <button
              mat-button
              matStepperNext
              type="button"
              color="primary"
              mat-raised-button
              [disabled]="!indicadoresGroup.valid"
            >
              Siguiente
            </button>
          </div>
        </mat-step>
        <mat-step>
          <ng-template matStepLabel>Finalizar</ng-template>
          <div class="buttons-container" fxLayoutGap="20px">
            <button mat-button matStepperPrevious type="button" mat-raised-button>
              Atras
            </button>
            <button mat-button (click)="stepper.reset()" mat-raised-button>
              Reset
            </button>
            <button
              mat-button
              matStepperNext
              type="submit"
              color="primary"
              mat-raised-button
              [disabled]="f.invalid"
            >
              Guardar
            </button>
          </div>
        </mat-step>
      </mat-horizontal-stepper>
    </ng-template>
  </form>
```

c. create.component.ts

```
  import { Component, OnInit, ViewChild } from "@angular/core";
  import {
    NgForm,
    NgModel
  } from "@angular/forms";
  import { MatStepper } from "@angular/material";
  import { Observable } from "rxjs";
  import { Store } from "@ngrx/store";
  import { Regla, Area, Indicador, Auth } from "../../models";
  import { ReglaService, UIService, AuthService, AlcaldiaService, HelperService } from
"../../services";
  import * as fromRoot from "../../store/app.reducers";
  import * as UI from "../../store/ui/ui.actions";

  @Component({
    selector: "app-create",
    templateUrl: "./create.component.html",
    styleUrls: ["./create.component.css"]
  })
  export class CreateComponent implements OnInit {
    indicadores: Indicador[] = [];
    areas: Area[] = [];
    reglaObject: any = [];
    indicadoresObject: Indicador[] = [];
    valuesArray: any[] = [];
```

```
  nombre: string;
  descripcion: string;
  mobileScreen = false;
  desktopScreen = true;
  mHeight: any;
  mWidth: any;
  currentUser$: Observable<Auth>;
  public alcaldias: any;
  invalid = false;
  constructor(
    private reglaService: ReglaService,
    private uiService: UIService,
    private alcaldiaService: AlcaldiaService,
    private authService: AuthService,
    private store: Store<fromRoot.State>,
    private helperService: HelperService
  ) {
    this.mHeight = window.screen.height;
    this.mWidth = window.screen.width;
    this.desktopScreen = false;
    if (this.mWidth <= 700 || (this.mWidth == 768 && this.mHeight == 1024)) {
      this.mobileScreen = true;
    }
  }
  @ViewChild('areaId') areaId: NgModel;

  ngOnInit() {
    this.currentUser$ = this.store.select(fromRoot.getCurrentUser);
    this.getAlcaldias();
    this.areas = this.reglaService.getAreasList();
    this.getIndicadores();
  }

  getIndicadores() {
    return this.reglaService.getIndicadores().subscribe((data: Indicador[]) => {
        this.indicadoresObject = data;
        this.indicadoresObject.forEach((element, x) => {
            if(element.editable){
                this.indicadores.push({
                    id: element.id,
                    nombre: element.nombre,
                    valores: [],
                    editable: element.editable
                });
                this.indicadores[x].valores.push({ valor: null, min: null, max: null });
            }
      });
    });
  }
  onSubmit(form: NgForm) {
    if (form.valid) {
      let idAlcaldia = form.value.idAlcaldia;
      if (this.authService.currentUserValue.user.rol == "admin") {
        idAlcaldia = this.authService.currentUserValue.user.idAlcaldia;
      }
      this.reglaObject.push({
        idAlcaldia: idAlcaldia,
        nombre: this.nombre,
        descripcion: this.descripcion,
        areasPrioritarias: this.areas,
        indicadores: this.indicadores,
        status: true
      });

      console.log(this.reglaObject)
      this.reglaService.crear(this.reglaObject).subscribe(
        data => {
          this.store.dispatch(new UI.StopLoading());
          this.uiService.showSnackbar(
```

```
              "Regla creada con éxito",
              null,
              3000,
              "custom-snack-bar"
            );
          },
          error => {
            this.store.dispatch(new UI.StopLoading());
            this.uiService.showSnackbar(
              "Problemas creando regla",
              null,
              3000,
              "custom-snack-bar-error"
            );
          }
        );
      }
    }
    onAddIndicador(index) {
      this.indicadores[index].valores.push({ valor: null, min: null, max: null });
    }
    onRemoveIndicador(index, idx) {
      this.indicadores[index].valores.splice(idx, 1);
    }

    goForward(stepper: MatStepper) {
      if (this.nombre && this.descripcion) {
        stepper.next();
      }
    }
    getAlcaldias() {
      return this.alcaldiaService
        .getAllAlcaldias()
        .pipe()
        .subscribe(
          data => {
            this.store.dispatch(new UI.StartLoading());
            this.alcaldias = data;
            this.store.dispatch(new UI.StopLoading());
          },
          error => {
            this.uiService.showSnackbar(
              "Problemas obteniendo alcaldías",
              null,
              3000,
              "custom-snack-bar-error"
            );
            console.log("error: ", error);
          }
        );
    }
  }
```

2. list

   a. delete

      i. delete.component.html

```
  <h1 mat-dialog-title>Eliminando Regla: {{ passedData.regla.nombre }}</h1>
  <mat-dialog-content>
    <mat-list>
      <mat-list-item> Esta seguro que quiere eliminarla? </mat-list-item>
    </mat-list>
```

```
    </mat-dialog-content>
    <mat-dialog-actions>
      <button   *ngIf="!(isLoading$   |   async)"   type="button"   mat-raised-button   color="primary"
(click)="onSubmit()">Delete</button>
      <button mat-button [mat-dialog-close]="false">Cancelar</button>
      <mat-spinner *ngIf="isLoading$ | async"></mat-spinner>
    </mat-dialog-actions>
```

ii. delete.component.ts

```
    import { Component, Inject, OnInit } from "@angular/core";
    import { MAT_DIALOG_DATA, MatDialogRef } from "@angular/material";
    import { NgForm, FormGroup, FormControl, Validators } from "@angular/forms";
    import { ReglaService, UIService } from "../../../services";
    import { Observable } from "rxjs";
    import { Store } from "@ngrx/store";

    import * as fromRoot from "../../../store/app.reducers";
    import * as UI from "../../../store/ui/ui.actions";

    @Component({
      selector: "app-delete-regla",
      templateUrl: "delete.component.html",
      styleUrls: ["../list.component.css"]
    })
    export class DeleteComponent implements OnInit {
      isLoading$: Observable<boolean>;

      constructor(
        public dialogRef: MatDialogRef<DeleteComponent>,
        @Inject(MAT_DIALOG_DATA) public passedData: any,
        private reglaService: ReglaService,
        private uiService: UIService,
        private store: Store<fromRoot.State>
      ) {}

      ngOnInit() {
        this.isLoading$ = this.store.select(fromRoot.getIsLoading);
      }

      onSubmit() {
        this.reglaService.deleteRegla(this.passedData.id).subscribe(
          data => {
            console.log("error");
            this.store.dispatch(new UI.StopLoading());
            this.uiService.showSnackbar(
              "Problemas eliminando regla",
              null,
              3000,
              "custom-snack-bar"
            );
            this.dialogRef.close();
          },
          error => {
            console.log("success");
            this.store.dispatch(new UI.StopLoading());
            this.uiService.showSnackbar(
              "Regla eliminada!",
              null,
              3000,
              "custom-snack-bar-error"
            );
            this.dialogRef.close();
          }
        );
      }
```

```
}
```

b. edit

i. edit.component.css

```
::ng-deep .mat-snack-bar-container {
  background-color: #36a5e6;
  color: #fff;
}
h2{
  color: #3f51b5;
  text-align: center;
}
.label-container {
  display: flex;
  flex-direction: column;
  align-items: flex-start;
  margin-top: 20px;
}

.mat-form-field {
  width: 100%;
}

.mat-icon-button {
  top: 10px;
  float: right;
}

::ng-deep .mat-horizontal-stepper-header {
  pointer-events: none !important;
}

/*
  ##Device = Desktops
  ##Screen = 1281px to higher resolution desktops
*/

@media (min-width: 1281px) {
  .mat-form-field {
    width: 30%;
  }
  .max-input {
    margin-left: 50px;
  }

  .max-input .mat-form-field {
    width: 50%;
  }
  .form-container {
    display: flex;
    flex-direction: column;
    align-items: center;
  }

  .row-container {
    display: flex;
    flex-direction: row;
    align-items: center;
  }
}

/*
  ##Device = Laptops, Desktops
```

```
    ##Screen = B/w 1025px to 1280px
  */

  @media (min-width: 1025px) and (max-width: 1280px) {
    .mat-form-field {
      width: 30%;
    }
    .max-input {
      margin-left: 50px;
    }

    .max-input .mat-form-field {
      width: 50%;
    }
    .form-container {
      display: flex;
      flex-direction: column;
      align-items: center;
    }

    .row-container {
      display: flex;
      flex-direction: row;
      align-items: center;
    }
  }
```

ii. edit.component.html

```
  <h2  *ngIf="!(isLoading$  |  async)"><a  routerLink="/configuracion/listar"  title="Atrás"><mat-
icon>arrow_back</mat-icon></a> Editando regla: {{ passedData.nombre }}</h2>
  <!-- <mat-dialog-content> -->
  <form #f="ngForm" (ngSubmit)="onSubmit(f)" *ngIf="!(isLoading$ | async)">
    <ng-template [ngIf]="mobileScreen" [ngIfElse]="desktopScreen">
      <mat-vertical-stepper #stepper="matVerticalStepper" linear>
        <mat-step>
          <ng-template matStepLabel>Datos generales</ng-template>
          <div
            ngModelGroup="generales"
            #generalesGroup="ngModelGroup"
            class="form-container"
            fxLayout="column"
            fxLayoutAlign="center center"
            fxLayoutGap="10px"
          >
            <mat-form-field>
              <input
                type="text"
                matInput
                [(ngModel)]="passedData.nombre"
                ngModel
                placeholder="Nombre"
                name="nombre"
                nombre
                required
                #nombreInput="ngModel"
              />
              <mat-error>Este campo es requerido</mat-error>
            </mat-form-field>
            <mat-form-field>
              <textarea
                cols="30"
                rows="7"
                matInput
                placeholder="Descripcion"
                name="descripcion"
```

```
                [(ngModel)]="passedData.descripcion"
                descripcion
                required
                ngModel
                #descripcionInput="ngModel"
              ></textarea>
              <mat-error>Este campo es requerido</mat-error>
            </mat-form-field>
          </div>
          <div>
            <button
              mat-button
              matStepperNext
              type="button"
              color="primary"
              mat-raised-button
              [disabled]="!generalesGroup.valid"
            >
              Siguiente
            </button>
          </div>
        </mat-step>
        <mat-step>
          <ng-template matStepLabel>Areas Prioritarias</ng-template>
          <div
            ngModelGroup="areas"
            #areasGroup="ngModelGroup"
            class="form-container2"
            fxLayout="column"
            fxLayoutAlign="center center"
            fxLayoutGap="10px"
          >
            <div
              *ngFor="let area of passedData.areasPrioritarias; let in = index"
            >
              <label class="max-input label-container">{{ area.nombre }}: </label>
              <mat-form-field class="max-input">
                <input
                  type="number"
                  min="1"
                  max="10"
                  matInput
                  placeholder="Puntaje"
                  [(ngModel)]="area.puntaje"
                  name="puntaje{{in}}"
                  #areaId
                  maxlength="2"
                  required
                  ngModel
                  (keydown)="validateValue($event)"
                  validateMaxMinValue = "1,10"
                />
                <mat-hint>Valores entre 1 y 10.</mat-hint>
                <mat-error> Este campo es requerido (entre 1 y 10). </mat-error>
              </mat-form-field>
            </div>
          </div>
          <div>
            <button mat-button matStepperPrevious type="button" mat-raised-button>
              Atras
            </button>
            <button
              mat-button
              matStepperNext
              type="button"
              color="primary"
              mat-raised-button
              [disabled]="!areasGroup.valid"
            >
```

```
            Siguiente
          </button>
        </div>
      </mat-step>
      <mat-step>
        <ng-template matStepLabel>Indicadores</ng-template>
        <div
          ngModelGroup="indicadores"
          #indicadoresGroup="ngModelGroup"
          class="form-container"
          fxLayout="column"
          fxLayoutAlign="center center"
          fxLayoutGap="10px"
        >
          <div *ngFor="let indicador of passedData.indicadores; let in = index">
            <div class="row-container">
              <label class="label-container">{{ indicador.nombre }}</label>
              <br />
            </div>
            <div *ngFor="let valores of indicador.valores; let i = index">
              <div class="row-container">
                <mat-form-field>
                  <input
                    matInput
                    placeholder="Valor"
                    [(ngModel)]="valores.valor"
                    name="valor{{in}}-{{i}}"
                    class="form-control"
                    required
                  />
                  <!--
                    <input matInput placeholder="Valor" name="valor{{in}}" required ngModel>
                  -->
                  <mat-error>Este campo es requerido</mat-error>
                </mat-form-field>
                <mat-form-field class="max-input">
                  <input
                    matInput
                    placeholder="Min"
                    [(ngModel)]="valores.min"
                    name="min{{in}}-{{i}}"
                    class="form-control"
                    required
                  />
                  <!--
                    <input matInput placeholder="Min" name="min{{in}}" required ngModel>
                  -->
                  <mat-error>Este campo es requerido</mat-error>
                </mat-form-field>
                <mat-form-field class="max-input">
                  <!--
                    <input matInput placeholder="Max" name="max{{in}}" required ngModel>
                  -->
                  <input
                    matInput
                    placeholder="Max"
                    [(ngModel)]="valores.max"
                    name="max{{in}}-{{i}}"
                    class="form-control"
                    required
                  />
                  <mat-error>Este campo es requerido</mat-error>
                </mat-form-field>
                <button
                  *ngIf="i == 0"
                  mat-icon-button
                  type="button"
                  (click)="onAddIndicador(in)"
                >
```

```
                    <mat-icon>add</mat-icon>
                  </button>
                  <button
                    *ngIf="i > 0"
                    mat-icon-button
                    type="button"
                    (click)="onRemoveIndicador(in, i)"
                  >
                    <mat-icon>delete</mat-icon>
                  </button>
                </div>
              </div>
            </div>
          </div>

          <div>
            <!-- [disabled]="!f.form.valid" -->
            <button mat-button matStepperPrevious type="button" mat-raised-button>
              Atras
            </button>
            <button
              mat-button
              matStepperNext
              type="button"
              color="primary"
              mat-raised-button
              [disabled]="!indicadoresGroup.valid"
            >
              Siguiente
            </button>
          </div>
        </mat-step>
        <mat-step>
          <ng-template matStepLabel>Finalizar</ng-template>
          <div class="buttons-container" fxLayoutGap="20px">
            <button mat-button matStepperPrevious type="button" mat-raised-button>
              Atras
            </button>
            <button
              mat-button
              matStepperNext
              type="submit"
              color="primary"
              mat-raised-button
              [disabled]="f.invalid"
            >
              Guardar
            </button>
          </div>
        </mat-step>
      </mat-vertical-stepper>
    </ng-template>
    <ng-template #desktopScreen>
      <mat-horizontal-stepper #stepper="matHorizontalStepper" linear>
        <mat-step>
          <ng-template matStepLabel>Datos generales</ng-template>
          <div
            ngModelGroup="generales"
            #generalesGroup="ngModelGroup"
            class="form-container"
            fxLayout="column"
            fxLayoutAlign="center center"
            fxLayoutGap="10px"
          >
            <mat-form-field>
              <input
                type="text"
                matInput
                [(ngModel)]="passedData.nombre"
```

```
                ngModel
                placeholder="Nombre"
                name="nombre"
                nombre
                required
                #nombreInput="ngModel"
              />
              <mat-error>Este campo es requerido</mat-error>
            </mat-form-field>
            <mat-form-field>
              <textarea
                cols="30"
                rows="7"
                matInput
                placeholder="Descripcion"
                name="descripcion"
                [(ngModel)]="passedData.descripcion"
                descripcion
                required
                ngModel
                #descripcionInput="ngModel"
              ></textarea>
              <mat-error>Este campo es requerido</mat-error>
            </mat-form-field>
          </div>
          <div>
            <button
              mat-button
              matStepperNext
              type="button"
              color="primary"
              mat-raised-button
              [disabled]="!generalesGroup.valid"
            >
              Siguiente
            </button>
          </div>
        </mat-step>
        <mat-step>
          <ng-template matStepLabel>Areas Prioritarias</ng-template>
          <div
            ngModelGroup="areas"
            #areasGroup="ngModelGroup"
            class="form-container2"
            fxLayout="column"
            fxLayoutAlign="center center"
            fxLayoutGap="10px"
          >
            <div
              *ngFor="let area of passedData.areasPrioritarias; let in = index"
            >
              <label class="max-input label-container">{{ area.nombre }}: </label>
              <mat-form-field class="max-input">
                <input
                  type="number"
                  min="1"
                  max="10"
                  matInput
                  placeholder="Puntaje"
                  [(ngModel)]="area.puntaje"
                  name="puntaje{{in}}"
                  #areaId
                  maxlength="2"
                  required
                  ngModel
                  (keydown)="validateValue($event)"
                  validateMaxMinValue = "1,10"
                />
                <mat-hint>Valores entre 1 y 10.</mat-hint>
```

```
                <mat-error> Este campo es requerido (entre 1 y 10). </mat-error>
              </mat-form-field>
            </div>
          </div>
          <div>
            <button mat-button matStepperPrevious type="button" mat-raised-button>
              Atras
            </button>
            <button
              mat-button
              matStepperNext
              type="button"
              color="primary"
              mat-raised-button
              [disabled]="!areasGroup.valid"
            >
              Siguiente
            </button>
          </div>
        </mat-step>
        <mat-step>
          <ng-template matStepLabel>Indicadores</ng-template>
          <div
            ngModelGroup="indicadores"
            #indicadoresGroup="ngModelGroup"
            class="form-container"
            fxLayout="column"
            fxLayoutAlign="center center"
            fxLayoutGap="10px"
          >
            <div *ngFor="let indicador of passedData.indicadores; let in = index">
              <div class="row-container">
                <label class="label-container">{{ indicador.nombre }}</label>
                <br />
              </div>
              <div *ngFor="let valores of indicador.valores; let i = index">
                <div class="row-container">
                  <mat-form-field>
                    <input
                      matInput
                      placeholder="Valor"
                      [(ngModel)]="valores.valor"
                      name="valor{{in}}-{{i}}"
                      class="form-control"
                      required
                    />
                    <!--
                      <input matInput placeholder="Valor" name="valor{{in}}" required ngModel>
                    -->
                    <mat-error>Este campo es requerido</mat-error>
                  </mat-form-field>
                  <mat-form-field class="max-input">
                    <input
                      matInput
                      placeholder="Min"
                      [(ngModel)]="valores.min"
                      name="min{{in}}-{{i}}"
                      class="form-control"
                      required
                    />
                    <!--
                      <input matInput placeholder="Min" name="min{{in}}" required ngModel>
                    -->
                    <mat-error>Este campo es requerido</mat-error>
                  </mat-form-field>
                  <mat-form-field class="max-input">
                    <!--
                      <input matInput placeholder="Max" name="max{{in}}" required ngModel>
                    -->
```

```
                    <input
                      matInput
                      placeholder="Max"
                      [(ngModel)]="valores.max"
                      name="max{{in}}-{{i}}"
                      class="form-control"
                      required
                    />
                    <mat-error>Este campo es requerido</mat-error>
                  </mat-form-field>
                  <button
                    *ngIf="i == 0"
                    mat-icon-button
                    type="button"
                    (click)="onAddIndicador(in)"
                  >
                    <mat-icon>add</mat-icon>
                  </button>
                  <button
                    *ngIf="i > 0"
                    mat-icon-button
                    type="button"
                    (click)="onRemoveIndicador(in, i)"
                  >
                    <mat-icon>delete</mat-icon>
                  </button>
                </div>
              </div>
            </div>
          </div>

          <div>
            <!-- [disabled]="!f.form.valid" -->
            <button mat-button matStepperPrevious type="button" mat-raised-button>
              Atras
            </button>
            <button
              mat-button
              matStepperNext
              type="button"
              color="primary"
              mat-raised-button
              [disabled]="!indicadoresGroup.valid"
            >
              Siguiente
            </button>
          </div>
        </mat-step>
        <mat-step>
          <ng-template matStepLabel>Finalizar</ng-template>
          <div class="buttons-container" fxLayoutGap="20px">
            <button mat-button matStepperPrevious type="button" mat-raised-button>
              Atras
            </button>
            <button
              mat-button
              matStepperNext
              type="submit"
              color="primary"
              mat-raised-button
              [disabled]="f.invalid"
            >
              Guardar
            </button>
          </div>
        </mat-step>
      </mat-horizontal-stepper>
    </ng-template>
  </form>
```

```
<mat-spinner
  *ngIf="(isLoading$ | async)"
  align="center"
  [diameter]="48"
  style="margin:0 auto;"
></mat-spinner>

<!-- </mat-dialog-content> -->
<!-- <mat-dialog-actions> </mat-dialog-actions> -->
```

iii. edit.component.ts

```
import { Component, OnInit } from "@angular/core";
import { NgForm, Validators } from "@angular/forms";
import { ReglaService, UIService, HelperService} from "../../../services";
import { Observable } from "rxjs";
import { Store } from "@ngrx/store";
import { Regla } from "../../../models";

import * as fromRoot from "../../../store/app.reducers";
import * as UI from "../../../store/ui/ui.actions";
import { Router, ActivatedRoute } from "@angular/router";

@Component({
  selector: "app-edit-regla",
  templateUrl: "./edit.component.html",
  styleUrls: ["./edit.component.css"]
})
export class EditComponent implements OnInit {
  reglaArray: any = [];
  regla: Regla;
  isLoading$: Observable<boolean>;

  proyectoSel: any[] = [];
  reglaData: any[] = [];
  mobileScreen = false;
  desktopScreen = true;
  mHeight: any;
  mWidth: any;
  public passedData: any;
  id: string;
  constructor(
    private reglaService: ReglaService,
    private uiService: UIService,
    private store: Store<fromRoot.State>,
    private route: ActivatedRoute,
    private router: Router,
    private helperService: HelperService
  ) {
    this.mHeight = window.screen.height;
    this.mWidth = window.screen.width;
    this.desktopScreen = false;
    if (this.mWidth <= 700 || (this.mWidth == 768 && this.mHeight == 1024)) {
      this.mobileScreen = true;
    }
    this.id = this.route.snapshot.paramMap.get("id");
    this.isLoading$ = this.store.select(fromRoot.getIsLoading);
  }

  ngOnInit() {
    this.store.dispatch(new UI.StartLoading());

    this.getDetails();
  }

  onSubmit(form: NgForm) {
```

```
      this.store.dispatch(new UI.StartLoading());
      this.reglaService.updateRegla(this.id, this.passedData).subscribe(
        data => {
          this.store.dispatch(new UI.StopLoading());
          this.uiService.showSnackbar(
            "Regla actualizada!",
            null,
            3000,
            "custom-snack-bar"
          );
          this.router.navigate(["/configuracion/listar"]);
        },
        error => {
          this.store.dispatch(new UI.StopLoading());
          this.uiService.showSnackbar(
            "Problemas actualizando regla",
            null,
            15000,
            "custom-snack-bar-error"
          );
        }
      );
    }
    onAddIndicador(index) {
      this.passedData.indicadores[index].valores.push({
        valor: null,
        min: null,
        max: null
      });
    }
    onRemoveIndicador(index, idx) {
      this.passedData.indicadores[index].valores.splice(idx, 1);
    }

    getDetails() {
      this.reglaService.getRegla(this.id).subscribe((reglaData: Regla[]) => {
        this.passedData = reglaData;
        this.store.dispatch(new UI.StopLoading());
      });
    }
    validateValue(event: KeyboardEvent) {
      this.helperService.validateValue(event);
    }
  }
```

c. list.component.css

```
  .form-container {
    display: flex;
    flex-direction: column;
  }

  ::ng-deep .mat-snack-bar-container {
    background-color: #36a5e6;
    color: #fff;
  }
```

d. list.component.html

```
  <h3>Listado de Reglas</h3>
  <mat-table [dataSource]="dataSource" matSort>
    <ng-container matColumnDef="nombre">
      <mat-header-cell mat-header-cell *matHeaderCellDef>
```

```
        Nombre
      </mat-header-cell>
      <mat-cell *matCellDef="let element"> {{ element.nombre }} </mat-cell>
    </ng-container>

    <ng-container matColumnDef="descripcion">
      <mat-header-cell mat-header-cell *matHeaderCellDef>
        Descripcion
      </mat-header-cell>
      <mat-cell *matCellDef="let element"> {{ element.descripcion }} </mat-cell>
    </ng-container>

    <ng-container matColumnDef="actions">
      <mat-header-cell *matHeaderCellDef mat-sort-header>Actions</mat-header-cell>
      <mat-cell *matCellDef="let element">
        <a routerLink="{{element.id}}"><mat-icon>edit</mat-icon></a>
        <button mat-icon-button (click)="onDelete(element.id)">
          <mat-icon>delete</mat-icon>
        </button>
      </mat-cell>
    </ng-container>
    <mat-header-row *matHeaderRowDef="displayedColumns"></mat-header-row>
    <mat-row *matRowDef="let row; columns: displayedColumns"></mat-row>
  </mat-table>
  <div *ngIf="notResults" align="center">No se encontraron resultados</div>

  <mat-paginator [pageSize]="5" [pageSizeOptions]="[5, 10, 20]"></mat-paginator>
  <mat-spinner
    *ngIf="(isLoading$ | async)"
    align="center"
    [diameter]="48"
    style="margin:0 auto;"
  ></mat-spinner>
```

e. list.component.ts

```
  import {
    Component,
    OnInit,
    ViewChild,
    AfterViewInit,
    ChangeDetectorRef
  } from "@angular/core";
  import { ReglaService } from "../../services";
  import {
    MatTableDataSource,
    MatSort,
    MatPaginator,
    MatDialog
  } from "@angular/material";
  import { Observable } from "rxjs";

  import { Regla } from "../../models";
  // import { EditComponent } from "./edit/edit.component";
  import { DeleteComponent } from "./delete/delete.component";
  import { Store } from "@ngrx/store";
  import * as fromRoot from "../../store/app.reducers";
  import * as UI from "../../store/ui/ui.actions";

  @Component({
    selector: "app-list",
    templateUrl: "./list.component.html",
    styleUrls: ["./list.component.css"]
  })
  export class ListComponent implements OnInit, AfterViewInit {
    regla: Regla;
```

```
    displayedColumns = ["nombre", "descripcion", "actions"];
    dataSource = new MatTableDataSource<Regla>();

    @ViewChild(MatSort) sort: MatSort;
    @ViewChild(MatPaginator) paginator: MatPaginator;
    notResults = false;
    isLoading$: Observable<boolean>;
    constructor(
      private reglaService: ReglaService,
      private dialog: MatDialog,
      private changeDetectorRefs: ChangeDetectorRef,
      private store: Store<fromRoot.State>
    ) {}

    ngOnInit() {
      this.isLoading$ = this.store.select(fromRoot.getIsLoading);
      this.onRefresh();
    }

    ngAfterViewInit() {
      this.dataSource.sort = this.sort;
      this.dataSource.paginator = this.paginator;
    }
    doFilter(filterValue: string) {
      this.dataSource.filter = filterValue.trim().toLowerCase();
    }

    onRefresh() {
      this.store.dispatch(new UI.StartLoading());
      this.reglaService.getAllReglas().subscribe((regla: Regla[]) => {
        this.dataSource.data = regla;
        if(this.dataSource.data.length == 0){
          this.notResults = true;
        }
        this.store.dispatch(new UI.StopLoading());

      });
    }
    onDelete(id) {
      this.reglaService.getRegla(id).subscribe((reglaData: Regla[]) => {
        const dialogRef = this.dialog
          .open(DeleteComponent, {
            data: {
              regla: reglaData,
              id: id
            }
          })
          .afterClosed()
          .subscribe(result => {
            this.onRefresh();
          });
      });
    }
  }
```

3. reglas.module.ts

```
  import { NgModule } from '@angular/core';
  import { CommonModule } from '@angular/common';
  import { RouterModule, Routes } from '@angular/router';

  import { ListComponent } from './list/list.component';
  import { CreateComponent } from './create/create.component';
  import { MaterialModule } from '../material.module';
  import { ReglaService } from '../services';
```

```
import { EditComponent } from './list/edit/edit.component';
import { DeleteComponent } from './list/delete/delete.component';
import { NgxMatSelectSearchModule } from 'ngx-mat-select-search';
import { ReactiveFormsModule, FormsModule } from '@angular/forms';
const routes: Routes = [
  { path: "crear", component: CreateComponent },
  { path: "listar", component: ListComponent },
  { path: "listar/:id", component: EditComponent }
];
@NgModule({
  imports: [
    CommonModule,
    MaterialModule,
    ReactiveFormsModule,
    FormsModule,
    NgxMatSelectSearchModule,
    RouterModule.forChild(routes)
  ],
  declarations: [
    ListComponent,
    CreateComponent,
    EditComponent,
    DeleteComponent
  ],
  entryComponents: [
    DeleteComponent
  ],
  providers: [ReglaService]
})
export class ReglasModule { }
```

x. services

1. alcaldia.service.ts

```
import { Injectable } from "@angular/core";
import { HttpClient } from "@angular/common/http";
import { Alcaldia } from "../models";
import { environment } from "../../environments/environment";

const API_URL = environment.apiUrl;

@Injectable()
export class AlcaldiaService {
  constructor(private http: HttpClient) {}

  crear(alcaldia: Alcaldia) {
    return this.http.post<Alcaldia[]>(API_URL + "Alcaldia/", alcaldia);
  }

  getAllAlcaldias() {
    let filter = JSON.stringify({ where: { status: true } });
    return this.http.get<Alcaldia[]>(API_URL + `Alcaldia/?filter=${filter}`);
  }
  getAllAlcaldias2(alcaldia_id) {
    if(alcaldia_id){
      let filter = JSON.stringify({ where: { id: alcaldia_id } });
      return this.http.get<Alcaldia[]>(API_URL + `Alcaldia/?filter=${filter}`);
    }
    return this.http.get<Alcaldia[]>(API_URL + `Alcaldia/`);

  }
  getAlcaldia(alcaldia_id) {
    return this.http.get<Alcaldia[]>(API_URL + `Alcaldia/${alcaldia_id}`);
  }
```

```
  updateAlcaldia(alcaldia_id, alcaldia: Alcaldia) {
    return this.http.patch<Alcaldia[]>(API_URL + `Alcaldia/${alcaldia_id}`, alcaldia);
  }

  deleteAlcaldia(alcaldia_id) {
    return this.http.delete<Alcaldia[]>(API_URL + `Alcaldia/${alcaldia_id}`);
  }
}
```

2. auth.service.ts

```
import { Injectable } from "@angular/core";
import { HttpClient, HttpHeaders } from "@angular/common/http";

import { BehaviorSubject, Observable } from "rxjs";
import { map } from "rxjs/operators";
import { AuthData, Auth } from "../models";
import { environment } from "../../environments/environment";

import { Store } from "@ngrx/store";

import * as fromRoot from "../store/app.reducers";
import * as AuthActions from "../store/auth/auth.actions";
import * as Users from "../store/users/users.actions";
const API_URL = environment.apiUrl;

@Injectable({ providedIn: "root" })
export class AuthService {
  private currentUserSubject: BehaviorSubject<Auth>;
  public currentUser: Observable<Auth>;

  constructor(private http: HttpClient, private store: Store<fromRoot.State>) {
    this.currentUserSubject = new BehaviorSubject<Auth>(
      JSON.parse(localStorage.getItem("currentUser"))
    );
    this.currentUser = this.currentUserSubject.asObservable();
  }

  public get currentUserValue(): Auth {
    return this.currentUserSubject.value;
  }
  initAuthListener() {
    const theUser: any = JSON.parse(localStorage.getItem("currentUser"));
    if (theUser) {
      this.store.dispatch(new AuthActions.SetAuthenticated());
      this.store.dispatch(
        new Users.SetCurrentUser(this.currentUserSubject.value)
      );
    } else {
      this.store.dispatch(new AuthActions.SetUnauthenticated());
    }
  }
  login(authData: AuthData) {
    const httpOptions = {
      headers: new HttpHeaders({
        "Content-Type": "application/json"
      })
    };
    return this.http
      .post(API_URL + "Users/signin", JSON.stringify(authData), httpOptions)
      .pipe(
        map((auth: Auth) => {
          if (auth.token && auth.user.id) {
            localStorage.setItem("currentUser", JSON.stringify(auth));
            this.currentUserSubject.next(auth);
```

```
              return auth;
            }
          })
        );
    }
    logout() {
      localStorage.removeItem("currentUser");
      this.store.dispatch(new AuthActions.SetUnauthenticated());
      this.currentUserSubject.next(null);
    }
  }
```

3. carpeta.service.ts

```
  import { Injectable } from "@angular/core";
  import { HttpClient } from "@angular/common/http";
  import { Carpeta } from "../models";

  import { environment } from "../../environments/environment";

  const API_URL = environment.apiUrl;

  @Injectable()
  export class CarpetaService {
    constructor(private http: HttpClient) {}

    crear(carpeta: Carpeta) {
      return this.http.post(API_URL + "Carpeta/", carpeta);
    }

    getAllCarpetas() {
      return this.http.get(API_URL + `Carpeta/`);
    }
    getCarpeta(carpeta_id) {
      return this.http.get(API_URL + `Carpeta/${carpeta_id}`);
    }

    updateCarpeta(carpeta_id, carpeta: Carpeta) {
      return this.http.patch(API_URL + `Carpeta/${carpeta_id}`, carpeta);
    }

    deleteCarpeta(carpeta_id) {
      return this.http.delete(API_URL + `Carpeta/${carpeta_id}`);
    }
    getEvaluacionesById(carpeta_id) {
        console.log('dsadsd',carpeta_id)
      let filter = JSON.stringify({ where: { carpetaId: carpeta_id } });
      return this.http.get(API_URL + `Evaluaciones/?filter=${filter}`);
    }
    evaluarCarpeta(carpeta_id) {
      return this.http.post(API_URL + "Calculos/evaluacion", {carpetaId: carpeta_id });
    }
  }
```

4. helper.service.ts

```
  import { Injectable } from "@angular/core";

  @Injectable({
      providedIn: 'root'
  })
  export class HelperService {
    allowedChars = new Set('0123456789'.split('').map(c => c.charCodeAt(0)));
```

```
  constructor() {}

  validateValue(event) {
    if (event.keyCode > 31 && !this.allowedChars.has(event.keyCode)) {
      event.preventDefault();
    }
  }
}
```

5. index.ts

```
export * from './auth.service';
export * from './users.service';
export * from './sidenav.service';
export * from './ui.service';
export * from './alcaldia.service';
export * from './carpeta.service';
export * from './proyecto.service';
export * from './regla.service';
export * from './helper.service';
```

6. proyecto.service.ts

```
import { Injectable } from "@angular/core";
import { HttpClient } from "@angular/common/http";
import { Proyecto, TipoProyecto } from "../models";
import { ReglaService } from './regla.service';
import { environment } from "../../environments/environment";

const API_URL = environment.apiUrl;

@Injectable()
export class ProyectoService {
  public tiposProyecto: TipoProyecto[] = [];
  public costos: any[] = [];
  public beneficios: any[] = [];
  public areas = this.reglaService.getAreasList();

  constructor(private http: HttpClient, private reglaService: ReglaService) {}

  crear(proyecto: Proyecto) {
    return this.http.post(API_URL + "Proyectos", proyecto);
  }

  getAllProyectos() {
    return this.http.get(API_URL + `Proyectos/`);
  }
  getProyecto(proyecto_id) {
    return this.http.get(API_URL + `Proyectos/${proyecto_id}`);
  }

  updateProyecto(proyecto_id, proyecto: Proyecto) {
    return this.http.patch(API_URL + `Proyectos/updateOne/${proyecto_id}`, proyecto);
  }

  deleteProyecto(proyecto_id) {
    return this.http.delete(API_URL + `Proyectos/${proyecto_id}`);
  }

  getTipoProyecto() {
    return this.http.get(API_URL + `Settings/getTipoProyectos`);
  }
```

```
    crearObjeto(form, flujoCaja) {
      let proyectoObject: Proyecto[] = [];
      let alcaldia = form.value.generales.idAlcaldia;
      proyectoObject= [{
        idAlcaldia: alcaldia,
        nombre: form.value.generales.nombre,
        descripcion: form.value.generales.descripcion,
        beneficiarios: form.value.detalles.beneficiarios,
        vidaUtil: form.value.detalles.vidautil,
        duracion: form.value.detalles.duracion,
        inversion: form.value.detalles.inversion,
        presupuesto: form.value.detalles.presupuesto,
        valorResidual: form.value.detalles.valorresidual,
        ubicacion: {
          lat: form.value.ubicacion.latitud,
          lng: form.value.ubicacion.longitud
        },
        tipoProyecto: this.tiposProyecto.filter(
          x => x.id == form.value.generales.tipoproyecto
        )[0],
        areaPrioritaria: this.areas.filter(
          x => x.id == form.value.generales.areasprioritarias
        )[0],
        // puntaje: form.value.generales.puntaje,
        costos: this.costos[0],
        beneficios: this.beneficios[0],
        responsable: {
          nombre: form.value.responsable.nombreresponsable,
          apellido: form.value.responsable.apellidoresponsable,
          cargo: form.value.responsable.cargo
        },
        costoTotal: 0,
        status: true,
        costoConstante: form.value.generales.costoconstante,
        porcentajeCostoVariable: form.value.generales.porcentaje
      }];
      if(flujoCaja != null){
          proyectoObject[0]['flujoCaja'] = flujoCaja;
      }
      return proyectoObject;
    }
    onSelectTipo(id) {
      this.costos = [];
      this.beneficios = [];
      this.costos.push(this.tiposProyecto.filter(x => x.id == id)[0].costos);
      this.beneficios.push(
          this.tiposProyecto.filter(x => x.id == id)[0].beneficios
      );
    }
    getTipoProyectos() {
      return this.getTipoProyecto()
        .subscribe((data: TipoProyecto[]) => {
          this.tiposProyecto = data;
        });
    }

    recalculoFlujoCaja(proyecto_id, proyecto: Proyecto) {
      return this.http.patch(API_URL + `Proyectos/updateOne/${proyecto_id}`, proyecto);
    }
  }
```

7. regla.service.ts

```
  import { Injectable } from "@angular/core";
  import { HttpClient } from "@angular/common/http";
```

```
import { Regla, Area } from "../models";

import { environment } from "../../environments/environment";

const API_URL = environment.apiUrl;

@Injectable()
export class ReglaService {
  constructor(private http: HttpClient) {}

  crear(regla: Regla) {
    return this.http.post<Regla[]>(API_URL + "Reglas/", regla);
  }

  getAllReglas() {
    return this.http.get(API_URL + `Reglas/`);
  }
  getRegla(regla_id) {
    return this.http.get(API_URL + `Reglas/${regla_id}`);
  }

  getAreasPrioritarias() {
    return this.http.get(API_URL + `Settings/getAreasPrioritarias`);
  }

  getIndicadores() {
    return this.http.get(API_URL + `Settings/getIndicadores`);
  }

  updateRegla(regla_id, regla: Regla) {
    return this.http.patch(API_URL + `Reglas/${regla_id}`, regla);
  }

  deleteRegla(regla_id) {
    return this.http.delete(API_URL + `Reglas/${regla_id}`);
  }

  getAreasList() {
    let areas: any[] = [];
    this.getAreasPrioritarias().subscribe((data: Area[]) => {
      data.forEach((element, x) => {
        areas.push({
          id: element.id,
          nombre: element.nombre,
          puntaje: null
        });
      });
    });
    return areas;
  }
}
```

8. sidenav.service.ts

```
import {EventEmitter, Injectable} from '@angular/core';

@Injectable()
export class NavService {
  public sidenav: any;

  constructor() {
  }

  public closeNav() {
    this.sidenav.close();
  }
```

```
  public openNav() {
    this.sidenav.open();
  }
}
```

9. ui.service.ts

```
import { Injectable } from '@angular/core';
import { MatSnackBar } from '@angular/material';

@Injectable()
export class UIService {

  constructor(private snackbar: MatSnackBar) {}

  showSnackbar(message, action, duration, customClass) {
    this.snackbar.open(message, action, {
      panelClass: [customClass],
      duration: duration
    });
  }
}
```

10. users.service.ts

```
import { Injectable } from "@angular/core";
import { HttpClient } from "@angular/common/http";
import { User } from "../models";
import { environment } from "../../environments/environment";

const API_URL = environment.apiUrl;

@Injectable()
export class UsersService {
  constructor(private http: HttpClient) {}

  register(user: User) {
    return this.http.post(API_URL + `users/`, user);
  }

  getAllUsers() {
    return this.http.get(API_URL + `users/`);
  }
  getUser(userid) {
    return this.http.get(API_URL + `users/${userid}`);
  }
  getUsersByAlcaldia(alcaldia_id) {
    let filter = JSON.stringify({ where: { idAlcaldia: alcaldia_id } });
    return this.http.get(API_URL + `users?filter=${filter}`);
  }
  updateUser(userid, user: User) {
    return this.http.patch(API_URL + `users/${userid}`, user);
  }

  deleteUser(userid) {
    return this.http.delete(API_URL + `users/${userid}`);
  }
}
```

xi. store

1. auth

a. auth.actions.ts

```
import { Action } from '@ngrx/store';

export const SET_AUTHENTICATED = '[Auth] Set Authenticated';
export const SET_UNAUTHENTICATED = '[Auth] Set Unauthenticated';

export class SetAuthenticated implements Action {
  readonly type = SET_AUTHENTICATED;
}

export class SetUnauthenticated implements Action {
  readonly type = SET_UNAUTHENTICATED;
}
export type AuthActions =  SetAuthenticated | SetUnauthenticated;
```

b. auth.reducers.ts

```
import { AuthActions, SET_AUTHENTICATED, SET_UNAUTHENTICATED } from './auth.actions';

export interface State {
  isAuthenticated: boolean;
}

const initialState: State = {
  isAuthenticated: false
};

export function authReducer(state = initialState, action: AuthActions) {
  switch (action.type) {
    case SET_AUTHENTICATED:
      return {
        isAuthenticated: true
      };
    case SET_UNAUTHENTICATED:
      return {
        isAuthenticated: false
      };
    default: {
      return state;
    }
  }
}
export const getIsAuth = (state: State) => state.isAuthenticated;
```

2. ui

a. ui.actions.ts

```
import { Action } from '@ngrx/store';

export const START_LOADING = '[UI] Start Loading';
export const STOP_LOADING = '[UI] Stop Loading';

export class StartLoading implements Action {
```

```
  readonly type = START_LOADING;
}

export class StopLoading implements Action {
  readonly type = STOP_LOADING;
}

export type UIActions = StartLoading | StopLoading;
```

b. ui.reducer.ts

```
import { UIActions, START_LOADING, STOP_LOADING } from './ui.actions';

export interface State {
  isLoading: boolean;
}

const initialState: State = {
  isLoading: false
};

export function uiReducer(state = initialState, action: UIActions) {
  switch (action.type) {
    case START_LOADING:
      return {
        isLoading: true
      };
    case STOP_LOADING:
      return {
        isLoading: false
      };
    default: {
      return state;
    }
  }
}

export const getIsLoading = (state: State) => state.isLoading;
```

3. users

a. users.actions.ts

```
import { Action } from '@ngrx/store';
import { Auth } from '../../models';
export const SET_CURRENT_USER = '[User] Set Current User';

export class SetCurrentUser implements Action {
  readonly type = SET_CURRENT_USER;

  constructor(public payload: Auth) { }
}

export type AuthActions = SetCurrentUser;
```

b. users.reducers.ts

```
import { Action } from "@ngrx/store";
```

```
import { AuthActions, SET_CURRENT_USER } from "./users.actions";
import { Auth } from "../../models";

export interface State {
  userData: Auth;
}
const initialState: State = {
  userData: null
};
export function userReducer(state = initialState, action: AuthActions) {
  switch (action.type) {
    case SET_CURRENT_USER:
      return {
        userData: action.payload
      };
    default: {
      return state;
    }
  }
}
export const getCurrentUser = (state: State) => state.userData;
```

4. app.reducers.ts

```
import { ActionReducerMap, createFeatureSelector, createSelector } from '@ngrx/store';

import * as fromUi from './ui/ui.reducer';
import * as fromAuth from './auth/auth.reducers';
import * as fromUser from './users/users.reducers';

export interface State {
  ui: fromUi.State;
  auth: fromAuth.State;
  user: fromUser.State;
}

export const reducers: ActionReducerMap<State> = {
  ui: fromUi.uiReducer,
  auth: fromAuth.authReducer,
  user: fromUser.userReducer
};

export const getUiState = createFeatureSelector<fromUi.State>('ui');
export const getIsLoading = createSelector(getUiState, fromUi.getIsLoading);

export const getAuthState = createFeatureSelector<fromAuth.State>('auth');
export const getIsAuth = createSelector(getAuthState, fromAuth.getIsAuth);

export const getUserState = createFeatureSelector<fromUser.State>('user');
export const getCurrentUser = createSelector(getUserState, fromUser.getCurrentUser);
```

xii. users

1. create

a. create.component.css

```
.form-container {
    display: flex;
    flex-direction: column;
    align-items: center;
```

```
}

.mat-form-field {
  width: 50%;
}

::ng-deep .mat-snack-bar-container {
  background-color: #36a5e6;
  color: #fff;
}
h2{
  color: #3f51b5;
  text-align: center;
}
/*
  ##Device = Tablets, Ipads (portrait)
  ##Screen = B/w 768px to 1024px
*/

@media (min-width: 768px) and (max-width: 1024px) {

  .mat-form-field {
    width: 80%;
  }

}

/*
  ##Device = Tablets, Ipads (landscape)
  ##Screen = B/w 768px to 1024px
*/

@media (min-width: 768px) and (max-width: 1024px) and (orientation: landscape) {

  .mat-form-field {
    width: 80%;
  }

}

/*
  ##Device = Low Resolution Tablets, Mobiles (Landscape)
  ##Screen = B/w 481px to 767px
*/

@media (min-width: 481px) and (max-width: 767px) {

  .mat-form-field {
    width: 80%;
  }

}

/*
  ##Device = Most of the Smartphones Mobiles (Portrait)
  ##Screen = B/w 320px to 479px
*/

@media (min-width: 320px) and (max-width: 480px) {

  .mat-form-field {
    width: 80%;
  }

}
```

b. create.component.html

```
   <h2>Crear Usuarios</h2>
   <form class="form-container" fxLayout="column" fxLayoutAlign="center center" fxLayoutGap="10px"
#f="ngForm" (ngSubmit)="onSubmit(f)">
     <mat-form-field *ngIf="((currentUser$ | async)?.user.rol == 'root')">
       <mat-select matInput ngModel placeholder="Alcaldia" idAlcaldia name="idAlcaldia" required
#idAlcaldiaInput="ngModel">
         <mat-option>Seleccione una opcion</mat-option>
         <mat-option *ngFor="let alcaldia of alcaldias; let in=index" [value]="alcaldia.id">
           {{alcaldia.nombre}}
         </mat-option>
       </mat-select>
       <mat-error>Este campo es requerido</mat-error>
     </mat-form-field>
     <mat-form-field>
       <input type="text" matInput ngModel placeholder="Nombre" name="nombre" nombre required
#nombreInput="ngModel">
       <mat-error *ngIf="nombreInput.hasError('required')">Field must not be empty.</mat-error>
     </mat-form-field>
     <mat-form-field>
       <input type="text" matInput ngModel placeholder="Apellido" name="apellido" apellido required
#apellidoInput="ngModel">
       <mat-error *ngIf="apellidoInput.hasError('required')">Field must not be empty.</mat-error>
     </mat-form-field>
     <mat-form-field>
       <input type="text" matInput ngModel placeholder="Username" name="username" username required
#usernameInput="ngModel">
       <mat-error *ngIf="usernameInput.hasError('required')">Field must not be empty.</mat-error>
     </mat-form-field>
     <mat-form-field>
       <input type="email" matInput ngModel placeholder="email" name="email" email required
#emailInput="ngModel">
       <mat-error *ngIf="emailInput.hasError('required')">Field must not be empty.</mat-error>
       <mat-error *ngIf="!emailInput.hasError('required')">E-Mail is invalid.</mat-error>
     </mat-form-field>
     <mat-form-field>
       <input type="text" matInput ngModel placeholder="Cargo" name="cargo" username required
#cargoInput="ngModel">
     </mat-form-field>
     <mat-form-field>
       <mat-select matInput ngModel placeholder="Rol" rol name="rol" required #rolInput="ngModel">
         <mat-option>Seleccionar Rol</mat-option>
         <mat-option *ngIf="((currentUser$ | async)?.rol != 'user')" value="admin">Admin</mat-option>
         <mat-option value="user">User</mat-option>
         <mat-option *ngIf="((currentUser$ | async)?.rol == 'root')" value="root">Root</mat-option>
       </mat-select>
     </mat-form-field>
     <mat-form-field hintLabel="Longitud de al menos 6 caracteres.">
       <input type="password" matInput placeholder="Contraseña" ngModel name="password" required
minlength="6" #pwInput="ngModel">
       <mat-hint align="end">{{ pwInput.value?.length }} / 6</mat-hint>
       <mat-error>Has to be at least 6 characters long.</mat-error>
     </mat-form-field>
     <mat-form-field hintLabel="Introduzca nuevamente la contraseña.">
       <input type="password" matInput placeholder="Confirmar contraseña" ngModel name="password2"
required minlength="6"
         #pw2Input="ngModel">
       <mat-error *ngIf="pw2Input.value !==  pwInput.value">Las contraseñas no coinciden.</mat-error>
     </mat-form-field>
     <br>
     <button *ngIf="!(isLoading$ | async)" type="submit" mat-raised-button color="primary"
[disabled]="f.invalid">Crear</button>
     <mat-spinner *ngIf="isLoading$ | async"></mat-spinner>
   </form>
```

c. create.component.ts

```
import { Component, OnInit } from "@angular/core";
import { NgForm, FormGroup, FormControl, Validators } from "@angular/forms";
import {
  UsersService,
  UIService,
  AlcaldiaService,
  AuthService
} from "../../services";
import { Observable } from "rxjs";
import { Store } from "@ngrx/store";
import { User, Auth, Alcaldia } from "../../models";
import * as fromRoot from "../../store/app.reducers";
import * as UI from "../../store/ui/ui.actions";
import { Router } from "@angular/router";

@Component({
  selector: "app-create",
  templateUrl: "./create.component.html",
  styleUrls: ["./create.component.css"]
})
export class CreateComponent implements OnInit {
  isLoading$: Observable<boolean>;
  currentUser$: Observable<Auth>;
  public alcaldias: any;
  constructor(
    private alcaldiaService: AlcaldiaService,
    private userService: UsersService,
    private authService: AuthService,
    private uiService: UIService,
    private store: Store<fromRoot.State>,
    private router: Router
  ) {}

  ngOnInit() {
    this.getAlcaldias();
    this.isLoading$ = this.store.select(fromRoot.getIsLoading);
    this.currentUser$ = this.store.select(fromRoot.getCurrentUser);
  }
  onSubmit(form: NgForm) {
    this.store.dispatch(new UI.StartLoading());
    form.value.status = true;
    if (this.authService.currentUserValue.user.rol == "admin") {
      form.value.idAlcaldia = this.authService.currentUserValue.user.idAlcaldia;
    }
    return this.userService
      .register(form.value)
      .pipe()
      .subscribe(
        data => {
          this.store.dispatch(new UI.StopLoading());
          this.uiService.showSnackbar(
            "Usuario creado",
            null,
            3000,
            "custom-snack-bar"
          );
          this.router.navigate(["/usuarios/listar"]);
        },
        error => {
          console.log("error: ", error);
          this.store.dispatch(new UI.StopLoading());
          this.uiService.showSnackbar(
            error,
            null,
            3000,
            "custom-snack-bar-error"
```

```
            );
          }
        );
    }
    getAlcaldias() {
      return this.alcaldiaService
        .getAllAlcaldias()
        .pipe()
        .subscribe(
          data => {
            this.store.dispatch(new UI.StartLoading());
            this.alcaldias = data;
            this.store.dispatch(new UI.StopLoading());
          },
          error => {
            this.uiService.showSnackbar(
              "Problemas obteniendo alcaldías",
              null,
              3000,
              "custom-snack-bar-error"
            );
            console.log("error: ", error);
          }
        );
    }
  }
```

2. list

a. delete

i. delete.component.css

ii. delete.component.html

```
  <h1 mat-dialog-title>Usuario: {{ passedData.user.nombre }} {{ passedData.user.apellido }}</h1>
  <mat-dialog-content>
    <mat-list>
      <mat-list-item> Esta seguro que quiere eliminarlo? </mat-list-item>
    </mat-list>
  </mat-dialog-content>
  <mat-dialog-actions  align="center">
    <button   *ngIf="!(isLoading$   |   async)"   type="button"   mat-raised-button   color="primary"
(click)="onSubmit()">Eliminar</button>
    <button mat-raised-button [mat-dialog-close]="false">Cancelar</button>
    <mat-spinner *ngIf="isLoading$ | async"></mat-spinner>
  </mat-dialog-actions>
```

iii. delete.component.ts

```
  import { Component, Inject, OnInit } from "@angular/core";
  import { MAT_DIALOG_DATA, MatDialogRef } from "@angular/material";
  import { UsersService, UIService } from "../../../services";
  import { Observable } from "rxjs";
  import { Store } from "@ngrx/store";

  import * as fromRoot from "../../../store/app.reducers";
  import * as UI from "../../../store/ui/ui.actions";

  @Component({
```

```
    selector: "app-delete-user",
    templateUrl: "delete.component.html",
    styleUrls: ["../list.component.css"]
  })
  export class DeleteComponent implements OnInit {
    isLoading$: Observable<boolean>;
    cargoSel: any;
    rolSel: any;
    constructor(
      public dialogRef: MatDialogRef<DeleteComponent>,
      @Inject(MAT_DIALOG_DATA) public passedData: any,
      private userService: UsersService,
      private uiService: UIService,
      private store: Store<fromRoot.State>
    ) {}

    ngOnInit() {
      this.isLoading$ = this.store.select(fromRoot.getIsLoading);
    }

    onSubmit() {
      this.userService.deleteUser(this.passedData.id).subscribe(
        data => {
          this.store.dispatch(new UI.StopLoading());
          this.uiService.showSnackbar(
            "Usuario eliminado!",
            null,
            3000,
            "custom-snack-bar"
          );
          this.dialogRef.close();
        },
        error => {
          this.store.dispatch(new UI.StopLoading());
          this.uiService.showSnackbar(
            "Problemas eliminando usuario!",
            null,
            3000,
            "custom-snack-bar-error"
          );
          this.dialogRef.close();
        }
      );
    }
  }
```

b. edit

i. edit.component.css

ii. edit.component.html

```
  <h2 mat-dialog-title>Editando Usuario</h2>
  <form #f="ngForm" (ngSubmit)="onSubmit(f)">
    <mat-dialog-content>
      <div class="form-container">
        <mat-form-field *ngIf="(rol == 'root')">
          <mat-select    matInput    [ngModel]="idAlcaldia"    placeholder="Alcaldia"    idAlcaldia
name="idAlcaldia" required
            #idAlcaldiaInput="ngModel">
            <mat-option>Seleccione una opcion</mat-option>
            <mat-option *ngFor="let alcaldia of alcaldias; let in=index" [value]="alcaldia.id">
              {{alcaldia.nombre}}
            </mat-option>
```

```
            </mat-select>
            <mat-error>Este campo es requerido</mat-error>
          </mat-form-field>
          <mat-form-field>
            <input type="text" matInput placeholder="Nombre" [ngModel]="passedData.user.nombre"
name="nombre" nombre
              required #nombreInput="ngModel">
            <mat-error *ngIf="nombreInput.hasError('required')">Field must not be empty.</mat-error>
          </mat-form-field>
          <mat-form-field>
            <input type="text" matInput placeholder="Apellido" [ngModel]="passedData.user.apellido"
name="apellido"
              apellido required #apellidoInput="ngModel">
            <mat-error *ngIf="apellidoInput.hasError('required')">Field must not be empty.</mat-error>
          </mat-form-field>
          <mat-form-field>
            <input type="text" matInput placeholder="Username" [ngModel]="passedData.user.username"
name="username"
              username required #usernameInput="ngModel">
            <mat-error *ngIf="usernameInput.hasError('required')">Field must not be empty.</mat-error>
          </mat-form-field>
          <mat-form-field>
            <input type="email" matInput placeholder="email" [ngModel]="passedData.user.email"
name="email" email required
              #emailInput="ngModel">
            <mat-error *ngIf="emailInput.hasError('required')">Field must not be empty.</mat-error>
            <mat-error *ngIf="!emailInput.hasError('required')">E-Mail is invalid.</mat-error>
          </mat-form-field>
          <mat-form-field>
            <input type="text" matInput placeholder="Cargo" [ngModel]="passedData.user.cargo"
name="cargo" cargo required
              #cargoInput="ngModel">
          </mat-form-field>
          <mat-form-field>
            <mat-select matInput rol [ngModel]="rolSel" name="rol" required #rolInput="ngModel">
              <mat-option>Seleccionar Rol</mat-option>
              <mat-option *ngIf="(rol != 'user')" value="admin">Admin</mat-option>
              <mat-option value="user">User</mat-option>
              <mat-option *ngIf="(rol == 'root')" value="root">Root</mat-option>
            </mat-select>
          </mat-form-field>
        </div>
      </mat-dialog-content>
      <mat-dialog-actions align="end">
        <button *ngIf="!(isLoading$ | async)" type="submit" mat-raised-button color="primary"
[disabled]="f.invalid">Guardar</button>
        <button mat-raised-button [mat-dialog-close]="false">Cancelar</button>
        <mat-spinner *ngIf="isLoading$ | async"></mat-spinner>
      </mat-dialog-actions>
    </form>
```

iii. edit.component.ts

```
    import { Component, Inject, OnInit } from "@angular/core";
    import { MAT_DIALOG_DATA, MatDialogRef } from "@angular/material";
    import { NgForm, FormGroup, FormControl, Validators } from "@angular/forms";
    import {
      UsersService,
      AlcaldiaService,
      UIService,
      AuthService
    } from "../../../services";
    import { Observable } from "rxjs";
    import { Store } from "@ngrx/store";
    import { Alcaldia } from "../../../models";
    import * as fromRoot from "../../../store/app.reducers";
```

```
import * as UI from "../../../store/ui/ui.actions";
import { Router } from "@angular/router";
@Component({
  selector: "app-edit",
  templateUrl: "edit.component.html",
  styleUrls: ["../list.component.css"]
})
export class EditComponent implements OnInit {
  isLoading$: Observable<boolean>;
  cargoSel: any;
  rolSel: any;
  rol: string;
  idAlcaldia: string;
  public alcaldias: Alcaldia[] = [];

  constructor(
    public dialogRef: MatDialogRef<EditComponent>,
    @Inject(MAT_DIALOG_DATA) public passedData: any,
    private userService: UsersService,
    private authService: AuthService,
    private alcaldiaService: AlcaldiaService,
    private uiService: UIService,
    private store: Store<fromRoot.State>,
    private router: Router
  ) {}

  ngOnInit() {
    this.isLoading$ = this.store.select(fromRoot.getIsLoading);
    this.cargoSel = this.passedData.user.cargo;
    this.rolSel = this.passedData.user.rol;
    this.rol = this.authService.currentUserValue.user.rol;
    this.idAlcaldia = this.passedData.user.idAlcaldia;
    this.getAlcaldias();
  }

  onSubmit(form: NgForm) {
    this.userService
      .updateUser(this.passedData.id, form.value)
      .pipe()
      .subscribe(
        data => {
          this.store.dispatch(new UI.StopLoading());
          this.uiService.showSnackbar(
            "Usuario actualizado!",
            null,
            3000,
            "custom-snack-bar"
          );
          this.dialogRef.close();
          this.router.navigate(["/usuarios/listar"]);
        },
        error => {
          this.store.dispatch(new UI.StopLoading());
          this.uiService.showSnackbar(
            "Problemas actualizando usuario",
            null,
            15000,
            "custom-snack-bar-error"
          );
          this.dialogRef.close();
        }
      );
  }
  getAlcaldias() {
    return this.alcaldiaService.getAllAlcaldias().subscribe(data => {
      this.alcaldias = data;
    });
  }
}
```

c. list.component.css

```
.form-container {
    display: flex;
    flex-direction: column;
  }

  ::ng-deep .mat-snack-bar-container {
    background-color: #36a5e6;
    color: #fff;
  }
  .mat-dialog-title, .mat-dialog-content{
    text-align: center;
  }
  h2{
    color: #3f51b5;
    font-weight: bold;
    text-align: center;
  }
  .filter{
    margin-left: 15px;
  }
  /*
    ##Device = Tablets, Ipads (portrait)
    ##Screen = B/w 768px to 1024px
  */

  @media (min-width: 768px) and (max-width: 1024px) {

   .mat-icon-button {
     width: 30px;
   }

  }

  /*
    ##Device = Tablets, Ipads (landscape)
    ##Screen = B/w 768px to 1024px
  */

  @media (min-width: 768px) and (max-width: 1024px) and (orientation: landscape) {

   .mat-icon-button {
     width: 30px;
   }

  }

  /*
    ##Device = Low Resolution Tablets, Mobiles (Landscape)
    ##Screen = B/w 481px to 767px
  */

  @media (min-width: 481px) and (max-width: 767px) {

   .mat-icon-button {
     width: 30px;
   }

  }

  /*
    ##Device = Most of the Smartphones Mobiles (Portrait)
    ##Screen = B/w 320px to 479px
  */
```

```
  @media (min-width: 320px) and (max-width: 480px) {

   .mat-icon-button {
     width: 30px;
   }

  }
```

d. list.component.html

```
<h2>Lista de Usuarios</h2>
<div fxLayoutAlign="center center" class="filter">
  <mat-form-field fxFlex="50%">
    <input
      matInput
      type="text"
      (keyup)="doFilter($event.target.value)"
      placeholder="Filter"
    />
  </mat-form-field>
</div>
<mat-table [dataSource]="dataSource" matSort>
  <ng-container matColumnDef="nombre">
    <mat-header-cell *matHeaderCellDef mat-sort-header>Nombre</mat-header-cell>
    <mat-cell *matCellDef="let element">{{ element.nombre }}</mat-cell>
  </ng-container>

  <ng-container matColumnDef="apellido">
    <mat-header-cell *matHeaderCellDef mat-sort-header
      >Apellido</mat-header-cell
    >
    <mat-cell *matCellDef="let element">{{ element.apellido }}</mat-cell>
  </ng-container>

  <ng-container matColumnDef="username">
    <mat-header-cell *matHeaderCellDef mat-sort-header
      >Username</mat-header-cell
    >
    <mat-cell *matCellDef="let element">{{ element.username }}</mat-cell>
  </ng-container>
  <ng-container matColumnDef="email">
    <mat-header-cell *matHeaderCellDef mat-sort-header>Email</mat-header-cell>
    <mat-cell *matCellDef="let element">{{ element.email }}</mat-cell>
  </ng-container>
  <ng-container matColumnDef="rol">
    <mat-header-cell *matHeaderCellDef mat-sort-header>Rol</mat-header-cell>
    <mat-cell *matCellDef="let element">{{ element.rol }}</mat-cell>
  </ng-container>

  <ng-container matColumnDef="actions">
    <mat-header-cell *matHeaderCellDef mat-sort-header>Actions</mat-header-cell>
    <mat-cell *matCellDef="let element">
      <button
        *ngIf="canEditDelete(element.rol)"
        mat-icon-button
        (click)="onEdit(element.id)"
      >
        <mat-icon>edit</mat-icon>
      </button>
      <button
        *ngIf="canEditDelete(element.rol)"
        mat-icon-button
        (click)="onDelete(element.id)"
      >
        <mat-icon>delete</mat-icon>
```

```
        </button>
      </mat-cell>
    </ng-container>

    <mat-header-row *matHeaderRowDef="displayedColumns"></mat-header-row>
    <mat-row *matRowDef="let row; columns: displayedColumns"></mat-row>
  </mat-table>
  <div *ngIf="notResults" align="center">No se encontraron resultados</div>
  <mat-paginator [pageSize]="5" [pageSizeOptions]="[5, 10, 20]"> </mat-paginator>
  <mat-spinner
    *ngIf="(isLoading$ | async)"
    align="center"
    [diameter]="48"
    style="margin:0 auto;"
  ></mat-spinner>
```

e. list.component.ts

```
  import {
    Component,
    OnInit,
    AfterViewInit,
    ViewChild,
    ChangeDetectorRef
  } from "@angular/core";
  import { Observable } from "rxjs";
  import { UsersService, AuthService, UIService } from "../../services";
  import {
    MatTableDataSource,
    MatSort,
    MatPaginator,
    MatDialog
  } from "@angular/material";
  import { EditComponent } from "./edit/edit.component";
  import { DeleteComponent } from "./delete/delete.component";
  import { User } from "../../models";
  import { Store } from "@ngrx/store";
  import * as fromRoot from "../../store/app.reducers";
  import * as UI from "../../store/ui/ui.actions";
  @Component({
    selector: "app-list",
    templateUrl: "./list.component.html",
    styleUrls: ["./list.component.css"]
  })
  export class ListComponent implements OnInit, AfterViewInit {
    mHeight: any;
    mWidth: any;
    isLoading$: Observable<boolean>;

    displayedColumns = [];
    dataSource = new MatTableDataSource<User>();
    user: User;
    res: boolean;
    currentUser: Observable<User>;
    idAlcaldia: String;
    rol: String;
    @ViewChild(MatSort)
    sort: MatSort;
    @ViewChild(MatPaginator)
    paginator: MatPaginator;
    notResults = false;

    constructor(
      private userService: UsersService,
      private authService: AuthService,
      private uiService: UIService,
```

```
    private dialog: MatDialog,
    private changeDetectorRefs: ChangeDetectorRef,
    private store: Store<fromRoot.State>
  ) {
    this.idAlcaldia = this.authService.currentUserValue.user.idAlcaldia;
    this.rol = this.authService.currentUserValue.user.rol;
    this.mHeight = window.screen.height;
    this.mWidth = window.screen.width;
    if (this.mWidth <= 700 || (this.mWidth == 768 && this.mHeight == 1024)) {
      this.displayedColumns = [
        "username",
        //"email",
        "rol",
        "actions"
      ];
    } else {
      this.displayedColumns = [
        "nombre",
        "apellido",
        "username",
        "email",
        "rol",
        "actions"
      ];
    }
  }

  ngOnInit() {
    this.isLoading$ = this.store.select(fromRoot.getIsLoading);
    this.onRefresh();
  }

  ngAfterViewInit() {
    this.dataSource.sort = this.sort;
    this.dataSource.paginator = this.paginator;
  }
  doFilter(filterValue: string) {
    this.dataSource.filter = filterValue.trim().toLowerCase();
  }
  onEdit(id) {
    return this.userService
      .getUser(id)
      .pipe()
      .subscribe(
        (userData: User[]) => {
          const dialogRef = this.dialog
            .open(EditComponent, {
              height: "600px",
              width: "500px",
              data: {
                user: userData,
                id: id
              }
            })
            .afterClosed()
            .subscribe(result => {
              this.onRefresh();
            });
        },
        error => {
          console.log("error: ", error);
          this.store.dispatch(new UI.StopLoading());
          this.uiService.showSnackbar(error, null, 3000, "custom-snack-bar-error");
        }
      );
  }
  onDelete(id) {
    return this.userService
      .getUser(id)
```

```
        .pipe()
        .subscribe(
          (userData: User[]) => {
            const dialogRef = this.dialog
              .open(DeleteComponent, {
                width: "250px",
                data: {
                  user: userData,
                  id: id
                }
              })
              .afterClosed()
              .subscribe(result => {
                this.onRefresh();
              });
          },
          error => {
            console.log("error: ", error);
            this.store.dispatch(new UI.StopLoading());
            this.uiService.showSnackbar(error, null, 3000, "custom-snack-bar-error");
          }
        );
    }
    onRefresh() {
      this.store.dispatch(new UI.StartLoading());
      if (this.rol == "root") {
        return this.userService
          .getAllUsers()
          .pipe()
          .subscribe(
            (data: User[]) => {
              this.dataSource.data = data;
              if(this.dataSource.data.length == 0){
                this.notResults = true;
              }
              this.store.dispatch(new UI.StopLoading());
            },
            error => {
              console.log("error: ", error);
              this.store.dispatch(new UI.StopLoading());
              this.uiService.showSnackbar(error, null, 3000, "custom-snack-bar-error");
            }
          );
      } else {
        this.userService
          .getUsersByAlcaldia(this.idAlcaldia)
          .pipe()
          .subscribe(
            (data: User[]) => {
              this.dataSource.data = data;
              this.store.dispatch(new UI.StopLoading());
            },
            error => {
              console.log("error: ", error);
              this.store.dispatch(new UI.StopLoading());
              this.uiService.showSnackbar(error, null, 3000, "custom-snack-bar-error");
            }
          );
      }
    }
    canEditDelete(userRol) {
      this.res = true;
      if (this.rol == "admin" && userRol == "admin") {
        this.res = false;
      }
      return this.res;
    }
  }
```

3. users.module.ts

```
import { NgModule } from "@angular/core";
import { CommonModule } from "@angular/common";
import { CreateComponent } from "./create/create.component";
import { RouterModule, Routes } from "@angular/router";
import { ListComponent } from "./list/list.component";
import { FormsModule } from "@angular/forms";
import { MaterialModule } from "../material.module";
import { EditComponent } from "./list/edit/edit.component";
import { DeleteComponent } from "./list/delete/delete.component";

const userRoutes: Routes = [
  { path: "crear", component: CreateComponent },
  { path: "listar", component: ListComponent }
];
@NgModule({
  declarations: [
    CreateComponent,
    ListComponent,
    EditComponent,
    DeleteComponent
  ],
  imports: [
    CommonModule,
    RouterModule.forChild(userRoutes),
    FormsModule,
    MaterialModule
  ],
  entryComponents: [
    EditComponent,
    DeleteComponent
  ],
})
export class UsersModule {}
```

xiii. app.component.css

```
/* mat-sidenav-container, mat-sidenav-content, mat-sidenav {
  height: 100%;
} */

mat-sidenav {
  width: 250px;
}

.main-container {
  position: absolute;
  top: 60px;
  bottom: 0;
  left: 0;
  right: 0;
}

.main-sidenav {
  display: flex;
  align-items: center;
  justify-content: center;
  /* width: 200px; */
  /* background: rgba(255, 0, 0, 0.5); */
  /* background: rgb(164, 165, 170); */
  background: #fff;
```

```
}
.user-profile {
  position: relative;
  background-size: cover;
}
.user-profile .profile-img > img {
  width: 60px;
  margin-left: 90px;
  padding: 20px 0;
}
.user-profile .profile-text {
  padding: 10px 0 10px 20px;
  position: relative;
  cursor: pointer;
  white-space: nowrap;
  color: rgb(105, 103, 103);
}

.user-profile .profile-img > img {
  border-radius: 50%;
}

::ng-deep .mat-nav-list {
  padding-top: 0px !important;
}

::ng-deep .mat-step-header .mat-step-icon-selected {
  background-color: #f44336;
}
::ng-deep .custom-snack-bar-error {
  background-color: #f44336 !important;
}

::ng-deep .custom-snack-bar-warning {
  background-color: #de9367 !important;
}
```

xiv. app.component.html

```
<app-header (sidenavToggle)="sidenav.toggle()" *ngIf="isAuth$ | async"></app-header>
<mat-sidenav-container autosize class="main-container">
    <mat-sidenav
      #sidenav
      mode="side"
      class="main-sidenav"
      role="navigation"
      [fixedTopGap]="0"
      [fixedBottomGap]="0"
      opened="{{isAuth$ | async}}"
    >
      <mat-nav-list>
        <div
          class="user-profile"
        >
          <div class="profile-img">
            <img alt="user" src="../../../assets/img/user.jpg" />
          </div>
          <div class="profile-text">
            <!-- <a aria-haspopup="true" class=""> John Doe - Nombre alcaldía</a> -->
          </div>
        </div>
        <mat-divider></mat-divider>
        <app-sidenav-list
          (closeSidenav)="sidenav.close()"
          *ngFor="let item of navItems"
          [item]="item"
```

```
          ></app-sidenav-list>
          <mat-divider></mat-divider>
          <mat-list-item>
              <button mat-icon-button (click)="onLogout()" *ngIf="isAuth$ | async">
              <mat-icon>exit_to_app</mat-icon>
              <span class="nav-caption">Logout</span>
            </button>
          </mat-list-item>
        </mat-nav-list>
      </mat-sidenav>
      <mat-sidenav-content>
        <main><router-outlet></router-outlet></main>
      </mat-sidenav-content>
    </mat-sidenav-container>
```

xv. app.component.ts

```
import {
  Component,
  ViewChild,
  ElementRef,
  AfterViewInit,
  OnInit,
  OnDestroy,
  HostListener
} from "@angular/core";
import { Observable } from "rxjs";
import { NavItem, Auth } from "./models";
import { NavService, AuthService } from "./services";
import { Store } from "@ngrx/store";
import * as fromRoot from "./store/app.reducers";
import { Router } from "@angular/router";

@Component({
  selector: "app-root",
  templateUrl: "./app.component.html",
  styleUrls: ["./app.component.css"]
})
export class AppComponent implements OnInit, AfterViewInit, OnDestroy {
  @ViewChild("sidenav") sidenav: ElementRef;
  title = "tesis-frontend";
  isAuth$: Observable<boolean>;
  rol = JSON.parse(localStorage.getItem("currentUser"))
    ? JSON.parse(localStorage.getItem("currentUser")).user.rol
    : null;
  currentUser$: Observable<Auth>;

  navItems: NavItem[] = [
    {
      displayName: "Home",
      iconName: "home",
      route: "home",
      code: "home"
    },
    {
      displayName: "Usuarios",
      iconName: "group_add",
      disabled: this.rol == "user" ? true : false,
      children: [
        {
          displayName: "Crear usuarios",
          iconName: "add",
          route: "usuarios/crear"
        },
        {
          displayName: "Listar usuarios",
```

```
            iconName: "list",
            route: "usuarios/listar"
          }
        ]
      },
      {
        displayName: "Alcaldias",
        iconName: "store",
        disabled: this.rol == "user" ? true : false,
        children: [
          {
            displayName: "Crear Alcaldía",
            iconName: "add",
            route: "alcaldias/crear",
            disabled: this.rol == "root" ? false : true
          },
          {
            displayName: "Administrar Alcaldía",
            iconName: "list",
            route: "alcaldias/administrar",
            disabled: this.rol == "user" ? true : false
          }
        ]
      },
      {
        displayName: "Carpetas",
        iconName: "folder",
        children: [
          {
            displayName: "Crear Carpetas",
            iconName: "add",
            route: "carpetas/crear"
          },
          {
            displayName: "Administrar Carpetas",
            iconName: "list",
            route: "carpetas/administrar"
          }
        ]
      },
      {
        displayName: "Proyectos",
        iconName: "folder",
        children: [
          {
            displayName: "Crear Proyectos",
            iconName: "add",
            route: "proyectos/crear"
          },
          {
            displayName: "Listar Proyectos",
            iconName: "list",
            route: "proyectos/listar"
          }
        ]
      },
      {
        displayName: "Reglas",
        iconName: "settings",
        disabled: this.rol == "user" ? true : false,
        children: [
          {
            displayName: "Crear Reglas",
            iconName: "add",
            route: "configuracion/crear"
          },
          {
            displayName: "Listar Configuraciones",
            iconName: "list",
```

```
            route: "configuracion/listar"
          }
        ]
      }
    ];
//   @HostListener('window:unload', ['$event'])
//   unloadHandler(event) {
//     this.authService.logout();
//   }
  constructor(
    private navService: NavService,
    private authService: AuthService,
    private store: Store<fromRoot.State>,
    private router: Router
  ) {}
  ngOnInit() {
    this.authService.initAuthListener();
    this.isAuth$ = this.store.select(fromRoot.getIsAuth);
    this.currentUser$ = this.store.select(fromRoot.getCurrentUser);
  }
  ngAfterViewInit() {
    this.navService.sidenav = this.sidenav;
  }
  onLogout() {
    this.authService.logout();
    this.router.navigate(["/login"]);
  }
  ngOnDestroy() {
    this.authService.logout();
  }
}
```

xvi. app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';

import { HttpClientModule, HTTP_INTERCEPTORS } from '@angular/common/http';
import { BrowserAnimationsModule } from '@angular/platform-browser/animations';
import { routing } from './app-routing.module';

import { ReactiveFormsModule, FormsModule } from '@angular/forms';

import { MaterialModule } from './material.module';
import { FlexLayoutModule } from '@angular/flex-layout';
import { StoreModule } from '@ngrx/store';
import { reducers } from './store/app.reducers';

import { AppComponent } from './app.component';
import { LoginComponent } from './login/login.component';
import { HomeComponent, HeaderComponent, SidenavListComponent } from './home';

import { JwtInterceptor, ErrorInterceptor } from './helpers';
import { NavService, UIService, UsersService, AlcaldiaService } from './services';

import { maxMinValidator } from "./helpers"
@NgModule({
  declarations: [
    AppComponent,
    LoginComponent,
    HomeComponent,
    HeaderComponent,
    SidenavListComponent,
    maxMinValidator
  ],
  imports: [
```

```
      BrowserModule,
      routing,
      HttpClientModule,
      ReactiveFormsModule,
      BrowserAnimationsModule,
      MaterialModule,
      FlexLayoutModule,
      StoreModule.forRoot(reducers)
    ],
    providers: [
      { provide: HTTP_INTERCEPTORS, useClass: JwtInterceptor, multi: true },
      { provide: HTTP_INTERCEPTORS, useClass: ErrorInterceptor, multi: true },
      NavService,
      UIService,
      UsersService,
      AlcaldiaService
    ],
    bootstrap: [AppComponent]
  })
  export class AppModule { }
```

xvii. app-routing.module.ts

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';
import { LoginComponent } from './login';
import { HomeComponent } from './home';
import { AuthGuard } from './guards';
const appRoutes: Routes = [
  { path: '', component: HomeComponent, canActivate: [AuthGuard] },
  { path: 'login', component: LoginComponent },
  {
    path: 'usuarios',
    loadChildren: './users/users.module#UsersModule'
  },
  {
    path: 'alcaldias',
    loadChildren: './alcaldia/alcaldia.module#AlcaldiaModule'
  },
  {
    path: 'carpetas',
    loadChildren: './carpeta/carpeta.module#CarpetaModule'
  },
  {
    path: 'proyectos',
    loadChildren: './proyectos/proyectos.module#ProyectosModule'
  },
  {
    path: 'configuracion',
    loadChildren: './reglas/reglas.module#ReglasModule'
  },
  { path: '**', redirectTo: '' }
];

export const routing = RouterModule.forRoot(appRoutes);
```

xviii. material.module.ts

```
import { NgModule } from '@angular/core';
import {
  MatButtonModule, MatIconModule, MatFormFieldModule, MatInputModule, MatDatepickerModule,
  MatNativeDateModule, MatCheckboxModule, MatSidenavModule, MatToolbarModule, MatListModule,
  MatTabsModule, MatCardModule, MatSelectModule, MatProgressSpinnerModule, MatDialogModule,
```

```
    MatTableModule, MatSortModule, MatPaginatorModule, MatSnackBarModule, MatStepperModule,
    MatGridListModule, MatMenuModule
} from '@angular/material';

@NgModule({
  imports: [
    MatButtonModule,
    MatIconModule,
    MatFormFieldModule,
    MatInputModule,
    MatDatepickerModule,
    MatNativeDateModule,
    MatCheckboxModule,
    MatSidenavModule,
    MatToolbarModule,
    MatListModule,
    MatTabsModule,
    MatCardModule,
    MatSelectModule,
    MatProgressSpinnerModule,
    MatDialogModule,
    MatTableModule,
    MatSortModule,
    MatPaginatorModule,
    MatSnackBarModule,
    MatStepperModule,
    MatGridListModule,
    MatMenuModule
  ],
  exports: [
    MatButtonModule,
    MatIconModule,
    MatFormFieldModule,
    MatInputModule,
    MatDatepickerModule,
    MatNativeDateModule,
    MatCheckboxModule,
    MatSidenavModule,
    MatToolbarModule,
    MatListModule,
    MatTabsModule,
    MatCardModule,
    MatSelectModule,
    MatProgressSpinnerModule,
    MatDialogModule,
    MatTableModule,
    MatSortModule,
    MatPaginatorModule,
    MatSnackBarModule,
    MatStepperModule,
    MatGridListModule,
    MatMenuModule
  ]
})
export class MaterialModule { }
```

b. assets

i. img

1. background.jpeg

2. circled_user.png

3. logo.png
4. logo-white.png
5. user.jpg

c. environments

i. environment.prod.ts

```
export const environment = {
  production: true,
  apiUrl: 'https://tesis-api-testing.herokuapp.com/api/'
};
```

ii. evironment.ts

```
// The file contents for the current environment will overwrite these during build.
// The build system defaults to the dev environment which uses `environment.ts`, but if you do
// `ng build --env=prod` then `environment.prod.ts` will be used instead.
// The list of which env maps to which file can be found in `.angular-cli.json`.

export const environment = {
  production: false,
  apiUrl: 'https://tesis-api-testing.herokuapp.com/api/'
};
```

d. browserslist

```
# This file is currently used by autoprefixer to adjust CSS to support the below specified browsers
# For additional information regarding the format and rule options, please see:
# https://github.com/browserslist/browserslist#queries
#
# For IE 9-11 support, please remove 'not' from the last line of the file and adjust as needed

> 0.5%
last 2 versions
Firefox ESR
not dead
not IE 9-11
```

e. favicon.ico

f. index.html

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>TesisFrontend</title>
  <base href="/">

  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
```

```
    <link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
    <link href="https://fonts.googleapis.com/css?family=Roboto" rel="stylesheet">
  </head>
  <body>
    <app-root></app-root>
  </body>
  </html>
```

g. karma.conf

```
// Karma configuration file, see link for more information
// https://karma-runner.github.io/1.0/config/configuration-file.html

module.exports = function (config) {
  config.set({
    basePath: '',
    frameworks: ['jasmine', '@angular-devkit/build-angular'],
    plugins: [
      require('karma-jasmine'),
      require('karma-chrome-launcher'),
      require('karma-jasmine-html-reporter'),
      require('karma-coverage-istanbul-reporter'),
      require('@angular-devkit/build-angular/plugins/karma')
    ],
    client: {
      clearContext: false // leave Jasmine Spec Runner output visible in browser
    },
    coverageIstanbulReporter: {
      dir: require('path').join(__dirname, '../coverage'),
      reports: ['html', 'lcovonly'],
      fixWebpackSourcePaths: true
    },
    reporters: ['progress', 'kjhtml'],
    port: 9876,
    colors: true,
    logLevel: config.LOG_INFO,
    autoWatch: true,
    browsers: ['Chrome'],
    singleRun: false
  });
};
```

h. main.ts

```
import { enableProdMode } from '@angular/core';
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';

import { AppModule } from './app/app.module';
import { environment } from './environments/environment';

if (environment.production) {
  enableProdMode();
}

platformBrowserDynamic().bootstrapModule(AppModule)
  .catch(err => console.error(err));
```

i. polyfills.ts

```
/**
```

```
    * This file includes polyfills needed by Angular and is loaded before the app.
    * You can add your own extra polyfills to this file.
    *
    * This file is divided into 2 sections:
    *   1. Browser polyfills. These are applied before loading ZoneJS and are sorted by browsers.
    *   2. Application imports. Files imported after ZoneJS that should be loaded before your main
    *      file.
    *
    * The current setup is for so-called "evergreen" browsers; the last versions of browsers that
    * automatically update themselves. This includes Safari >= 10, Chrome >= 55 (including Opera),
    * Edge >= 13 on the desktop, and iOS 10 and Chrome on mobile.
    *
    * Learn more in https://angular.io/guide/browser-support
    */

   /***************************************************************************************************
**
    * BROWSER POLYFILLS
    */

   /** IE9, IE10 and IE11 requires all of the following polyfills. **/
   // import 'core-js/es6/symbol';
   // import 'core-js/es6/object';
   // import 'core-js/es6/function';
   // import 'core-js/es6/parse-int';
   // import 'core-js/es6/parse-float';
   // import 'core-js/es6/number';
   // import 'core-js/es6/math';
   // import 'core-js/es6/string';
   // import 'core-js/es6/date';
   // import 'core-js/es6/array';
   // import 'core-js/es6/regexp';
   // import 'core-js/es6/map';
   // import 'core-js/es6/weak-map';
   // import 'core-js/es6/set';

   /**
    * If the application will be indexed by Google Search, the following is required.
    * Googlebot uses a renderer based on Chrome 41.
    * https://developers.google.com/search/docs/guides/rendering
    **/
   // import 'core-js/es6/array';

   /** IE10 and IE11 requires the following for NgClass support on SVG elements */
   // import 'classlist.js';  // Run `npm install --save classlist.js`.

   /** IE10 and IE11 requires the following for the Reflect API. */
   // import 'core-js/es6/reflect';

   /**
    * Web Animations `@angular/platform-browser/animations`
    * Only required if AnimationBuilder is used within the application and using IE/Edge or Safari.
    * Standard animation support in Angular DOES NOT require any polyfills (as of Angular 6.0).
    **/
   // import 'web-animations-js';  // Run `npm install --save web-animations-js`.

   /**
    * By default, zone.js will patch all possible macroTask and DomEvents
    * user can disable parts of macroTask/DomEvents patch by setting following flags
    */

    //   (window   as   any).__Zone_disable_requestAnimationFrame   =   true;   //   disable   patch
requestAnimationFrame
    // (window as any).__Zone_disable_on_property = true; // disable patch onProperty such as onclick
    // (window as any).__zone_symbol__BLACK_LISTED_EVENTS = ['scroll', 'mousemove']; // disable patch
specified eventNames

    /*
    * in IE/Edge developer tools, the addEventListener will also be wrapped by zone.js
```

```
     * with the following flag, it will bypass `zone.js` patch for IE/Edge
     */
    // (window as any).__Zone_enable_cross_context_check = true;

    /***************************************************************************************************
**
     * Zone JS is required by default for Angular itself.
     */
    import 'zone.js/dist/zone';  // Included with Angular CLI.


    /***************************************************************************************************
**
     * APPLICATION IMPORTS
     */
```

j. styles.css

```
    /* You can add global styles to this file, and also import other style files */
    @import "@angular/material/prebuilt-themes/indigo-pink.css";

    html, body {
      font-family: 'Roboto', sans-serif;
      height: 100%;
    }

    body {
      margin: 0;
    }
```

k. tsconfig.app

```
    {
      "extends": "../tsconfig.json",
      "compilerOptions": {
        "outDir": "../out-tsc/app",
        "types": []
      },
      "exclude": [
        "test.ts",
        "**/*.spec.ts"
      ]
    }
```

l. tsling

```
    {
        "extends": "../tslint.json",
        "rules": {
            "directive-selector": [
                true,
                "attribute",
                "app",
                "camelCase"
            ],
            "component-selector": [
                true,
                "element",
                "app",
                "kebab-case"
```

```
          ]
      }
  }
```

2. angular.json

```
{
  "$schema": "./node_modules/@angular/cli/lib/config/schema.json",
  "version": 1,
  "newProjectRoot": "projects",
  "projects": {
    "tesis-frontend": {
      "root": "",
      "sourceRoot": "src",
      "projectType": "application",
      "prefix": "app",
      "schematics": {},
      "architect": {
        "build": {
          "builder": "@angular-devkit/build-angular:browser",
          "options": {
            "outputPath": "dist/tesis-frontend",
            "index": "src/index.html",
            "main": "src/main.ts",
            "polyfills": "src/polyfills.ts",
            "tsConfig": "src/tsconfig.app.json",
            "assets": [
              "src/favicon.ico",
              "src/assets"
            ],
            "styles": [
              "src/styles.css"
            ],
            "scripts": []
          },
          "configurations": {
            "production": {
              "fileReplacements": [
                {
                  "replace": "src/environments/environment.ts",
                  "with": "src/environments/environment.prod.ts"
                }
              ],
              "optimization": true,
              "outputHashing": "all",
              "sourceMap": false,
              "extractCss": true,
              "namedChunks": false,
              "aot": true,
              "extractLicenses": true,
              "vendorChunk": false,
              "buildOptimizer": true,
              "budgets": [
                {
                  "type": "initial",
                  "maximumWarning": "2mb",
                  "maximumError": "5mb"
                }
              ]
            }
          }
        },
        "serve": {
          "builder": "@angular-devkit/build-angular:dev-server",
          "options": {
            "browserTarget": "tesis-frontend:build"
```

```
            },
            "configurations": {
              "production": {
                "browserTarget": "tesis-frontend:build:production"
              }
            }
          },
          "extract-i18n": {
            "builder": "@angular-devkit/build-angular:extract-i18n",
            "options": {
              "browserTarget": "tesis-frontend:build"
            }
          },
          "test": {
            "builder": "@angular-devkit/build-angular:karma",
            "options": {
              "main": "src/test.ts",
              "polyfills": "src/polyfills.ts",
              "tsConfig": "src/tsconfig.spec.json",
              "karmaConfig": "src/karma.conf.js",
              "styles": [
                "src/styles.css"
              ],
              "scripts": [],
              "assets": [
                "src/favicon.ico",
                "src/assets"
              ]
            }
          },
          "lint": {
            "builder": "@angular-devkit/build-angular:tslint",
            "options": {
              "tsConfig": [
                "src/tsconfig.app.json",
                "src/tsconfig.spec.json"
              ],
              "exclude": [
                "**/node_modules/**"
              ]
            }
          }
        }
      },
      "tesis-frontend-e2e": {
        "root": "e2e/",
        "projectType": "application",
        "prefix": "",
        "architect": {
          "e2e": {
            "builder": "@angular-devkit/build-angular:protractor",
            "options": {
              "protractorConfig": "e2e/protractor.conf.js",
              "devServerTarget": "tesis-frontend:serve"
            },
            "configurations": {
              "production": {
                "devServerTarget": "tesis-frontend:serve:production"
              }
            }
          },
          "lint": {
            "builder": "@angular-devkit/build-angular:tslint",
            "options": {
              "tsConfig": "e2e/tsconfig.e2e.json",
              "exclude": [
                "**/node_modules/**"
              ]
            }
```

```
          }
        }
      }
    },
    "defaultProject": "tesis-frontend"
  }
```

3. package.json

```
  {
    "name": "tesis-frontend",
    "version": "0.0.0",
    "scripts": {
      "ng": "ng",
      "start": "node server.js",
      "build": "ng build",
      "test": "ng test",
      "lint": "ng lint",
      "e2e": "ng e2e",
      "postinstall":"ng build --prod --aot"
    },
    "private": true,
    "dependencies": {
      "@agm/core": "^1.0.0-beta.5",
      "@angular/animations": "^7.0.4",
      "@angular/cdk": "^7.0.4",
      "@angular/common": "~7.0.0",
      "@angular/compiler": "~7.0.0",
      "@angular/core": "~7.0.0",
      "@angular/flex-layout": "^7.0.0-beta.19",
      "@angular/forms": "~7.0.0",
      "@angular/http": "~7.0.0",
      "@angular/material": "^7.0.4",
      "@angular/platform-browser": "~7.0.0",
      "@angular/platform-browser-dynamic": "~7.0.0",
      "@angular/router": "~7.0.0",
      "@ngrx/store": "^6.1.2",
      "@types/googlemaps": "^3.30.16",
      "core-js": "^2.5.4",
      "express": "^4.16.4",
      "ng-multiselect-dropdown": "^0.2.3",
      "ng2-currency-mask": "^5.3.1",
      "ngx-mat-select-search": "^1.4.2",
      "path": "^0.12.7",
      "rxjs": "~6.3.3",
      "zone.js": "~0.8.26",
      "@angular/cli": "~7.0.3",
      "@angular/compiler-cli": "~7.0.0",
      "typescript": "~3.1.1"
    },
    "devDependencies": {
      "@angular-devkit/build-angular": "~0.10.0",
      "@angular/language-service": "~7.0.0",
      "@types/jasmine": "~2.8.8",
      "@types/jasminewd2": "~2.0.3",
      "@types/node": "~8.9.4",
      "codelyzer": "~4.5.0",
      "enhanced-resolve": "^3.3.0",
      "jasmine-core": "~2.99.1",
      "jasmine-spec-reporter": "~4.2.1",
      "karma": "~3.0.0",
      "karma-chrome-launcher": "~2.2.0",
      "karma-coverage-istanbul-reporter": "~2.0.1",
      "karma-jasmine": "~1.1.2",
      "karma-jasmine-html-reporter": "^0.2.2",
      "protractor": "~5.4.0",
```

```
    "ts-node": "~7.0.0",
    "tslint": "~5.11.0"
  },
  "engines": {
    "node": "8.12.0",
    "npm": "6.4.1"
  }
}
```

4. server.js

```
//Install express server
const express = require('express');
const path = require('path');

const app = express();

// Serve only the static files form the dist directory
app.use(express.static(__dirname + '/dist/tesis-frontend'));

app.get('/*', function(req,res) {

res.sendFile(path.join(__dirname+'/dist/tesis-frontend/index.html'));
});

// Start the app by listening on the default Heroku port
app.listen(process.env.PORT || 4200);
```

5. tsconfig.json

```
{
  "compileOnSave": false,
  "compilerOptions": {
    "baseUrl": "./",
    "outDir": "./dist/out-tsc",
    "sourceMap": true,
    "declaration": false,
    "module": "es2015",
    "moduleResolution": "node",
    "emitDecoratorMetadata": true,
    "experimentalDecorators": true,
    "target": "es5",
    "typeRoots": [
      "node_modules/@types"
    ],
    "lib": [
      "es2018",
      "dom"
    ]
  }
}
```

6. tsling.json

```
{
  "rulesDirectory": [
    "node_modules/codelyzer"
  ],
  "rules": {
    "arrow-return-shorthand": true,
```

```
        "callable-types": true,
        "class-name": true,
        "comment-format": [
          true,
          "check-space"
        ],
        "curly": true,
        "deprecation": {
          "severity": "warn"
        },
        "eofline": true,
        "forin": true,
        "import-blacklist": [
          true,
          "rxjs/Rx"
        ],
        "import-spacing": true,
        "indent": [
          true,
          "spaces"
        ],
        "interface-over-type-literal": true,
        "label-position": true,
        "max-line-length": [
          true,
          140
        ],
        "member-access": false,
        "member-ordering": [
          true,
          {
            "order": [
              "static-field",
              "instance-field",
              "static-method",
              "instance-method"
            ]
          }
        ],
        "no-arg": true,
        "no-bitwise": true,
        "no-console": [
          true,
          "debug",
          "info",
          "time",
          "timeEnd",
          "trace"
        ],
        "no-construct": true,
        "no-debugger": true,
        "no-duplicate-super": true,
        "no-empty": false,
        "no-empty-interface": true,
        "no-eval": true,
        "no-inferrable-types": [
          true,
          "ignore-params"
        ],
        "no-misused-new": true,
        "no-non-null-assertion": true,
        "no-redundant-jsdoc": true,
        "no-shadowed-variable": true,
        "no-string-literal": false,
        "no-string-throw": true,
        "no-switch-case-fall-through": true,
        "no-trailing-whitespace": true,
        "no-unnecessary-initializer": true,
        "no-unused-expression": true,
```

```
      "no-use-before-declare": true,
      "no-var-keyword": true,
      "object-literal-sort-keys": false,
      "one-line": [
        true,
        "check-open-brace",
        "check-catch",
        "check-else",
        "check-whitespace"
      ],
      "prefer-const": true,
      "quotemark": [
        true,
        "single"
      ],
      "radix": true,
      "semicolon": [
        true,
        "always"
      ],
      "triple-equals": [
        true,
        "allow-null-check"
      ],
      "typedef-whitespace": [
        true,
        {
          "call-signature": "nospace",
          "index-signature": "nospace",
          "parameter": "nospace",
          "property-declaration": "nospace",
          "variable-declaration": "nospace"
        }
      ],
      "unified-signatures": true,
      "variable-name": false,
      "whitespace": [
        true,
        "check-branch",
        "check-decl",
        "check-operator",
        "check-separator",
        "check-type"
      ],
      "no-output-on-prefix": true,
      "use-input-property-decorator": true,
      "use-output-property-decorator": true,
      "use-host-property-decorator": true,
      "no-input-rename": true,
      "no-output-rename": true,
      "use-life-cycle-interface": true,
      "use-pipe-transform-interface": true,
      "component-class-suffix": true,
      "directive-class-suffix": true
    }
  }
```

4.5 Calendarización

En esta sección se proporciona la estructura de desglose del trabajo para el proceso de desarrollo del sistema; es decir, describe el orden en que se entregarán los distintos componentes de la solución.

Fase	Tarea	Duracion(dias)	Inicio	Fin
Fase 1: Preparacion de ambiente de desarrollo	Configuración de plataforma de control de versiones(GitHub)	3	01-02-19	04-02-19
	Configuración de entorno de Despliegue(Heroku)	2	05-02-19	07-02-19
	Configuración de plataforma de seguimiento de desarrollo(Trello)	2	08-02-19	10-02-19
	Instalacion de tecnologias	2	11-02-19	13-02-19
	Configuracion NodeJs	2	14-02-19	16-02-19
	Configuracion Express/LoopBack	2	17-02-19	19-02-19
	Configuracion Angular Material	2	20-02-19	22-02-19
	Configuracion MongoDB	2	23-02-19	25-02-19
	Organización de Equipo BackEnd/FrontEnd	2	26-02-19	28-02-19
Fase 2: Desarrollo	Creacion de BD	30	01-03-19	31-03-19
	Desarrollo BackEnd	29	01-04-19	30-04-19
	Desarrollo FrontEnd	30	01-05-19	31-05-19
	Desarrollo Seguridad	29	01-06-19	30-06-19
Fase 3: Pruebas	Testing de componentes	3	01-07-19	04-07-19
	Inicio de sesion	3	05-07-19	08-07-19
	Reporte de Errores	2	09-07-19	11-07-19
	Solucion de Errores de Inicio de Sesion	3	12-07-19	15-07-19
	Usuarios	2	16-07-19	18-07-19
	Reporte de Errores	3	19-07-19	22-07-19
	Solucion de Errores de Usuarios	3	23-07-19	26-07-19
	Alcaldias	2	27-07-19	29-07-19
	Reporte de Errores	3	30-07-19	02-08-19
	Solucion de Errores de Alcaldias	2	03-08-19	05-08-19
	Carpetas	3	06-08-19	09-08-19
	Reporte de Errores	2	10-08-19	12-08-19
	Solucion de Errores de Carpetas	3	13-08-19	16-08-19
	Alcaldias	2	17-08-19	19-08-19
	Reporte de Errores	2	20-08-19	22-08-19

	Solucion de Errores de Alcaldias	3	23-08-19	26-08-19
	Proyectos	2	27-08-19	29-08-19
	Reporte de Errores	3	30-08-19	02-09-19
	Solucion de Errores de Proyectos	3	03-09-19	06-09-19
	Evaluaciones	3	07-09-19	10-09-19
	Reporte de Errores	3	11-09-19	14-09-19
	Solucion de Errores de Evaluaciones	3	15-09-19	18-09-19
	Reglas	3	19-09-19	22-09-19
	Reporte de Errores	3	23-09-19	26-09-19
	Solucion de Errores de Reglas	3	27-09-19	30-09-19
Fase 4 Documentacion	Documentacion de Software	30	01-10-19	31-10-19

4.6 Plan de Seguridad

4.6.1 Proceso para cuentas de usuario

En base a los requerimientos del diseño, se ha determinado distintos roles dentro del sistema:

- Root: Este usuario tiene permisos sobre todas las funcionalidades del sistema.
- Admin: Este usuario tiene permisos sobre todas las funcionalidades del sistema a excepción de la creación, edición y eliminación de alcaldías.
- User: Este usuario tiene permisos limitados sobre las funcionalidades del sistema. Solamente tiene permisos sobre Carpetas, Proyectos y Reportes.

4.6.2 Identificación y autenticación

Para el manejo de identificación y autenticación de usuarios se utilizó JSON Web Token como mecanismo de control. El cual es un estándar web para crear tokens basados en JSON.

4.7 Plan de Pruebas

El plan de prueba describe la estrategia y el enfoque utilizados para planificar, organizar y administrar las actividades de prueba del sistema.

4.7.1 Procedimientos de prueba

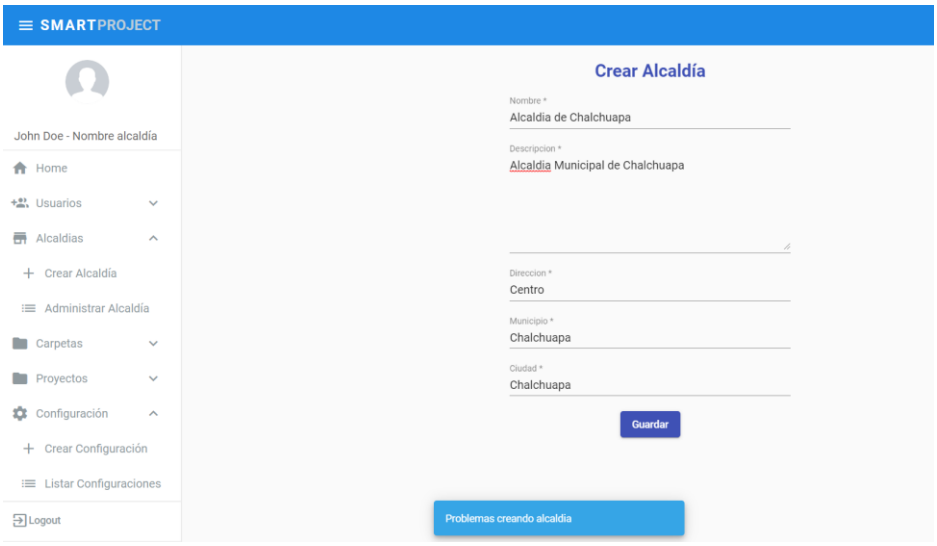
1. Prueba de integración: Primeramente, se verifica que la nueva porción de código (o la modificación) fue implementada correctamente en el todo del sistema. Se asegura, además, que la interacción con los demás componentes se realice adecuadamente.
2. Prueba funcional: Se verifica que la nueva poción de código (o la modificación) realiza el proceso que se esperaba. Es decir que procesa correctamente la entrada de datos que recibe y se obtiene el resultado esperado.
3. Prueba de Usabilidad: Se verifica que la nueva porción de código (o la modificación) es adecuada para el usuario y no representa una dificultad lógica y funcional en el uso.
4. Prueba de Rendimiento: Se verifica que la nueva porción de código (o la modificación) responde en el tiempo adecuado para la tarea que está realizando.
5. Realizar informe de pruebas: Se lista los resultados obtenidos de las pruebas anteriores.

4.7.3 Seguimiento e informes de estado

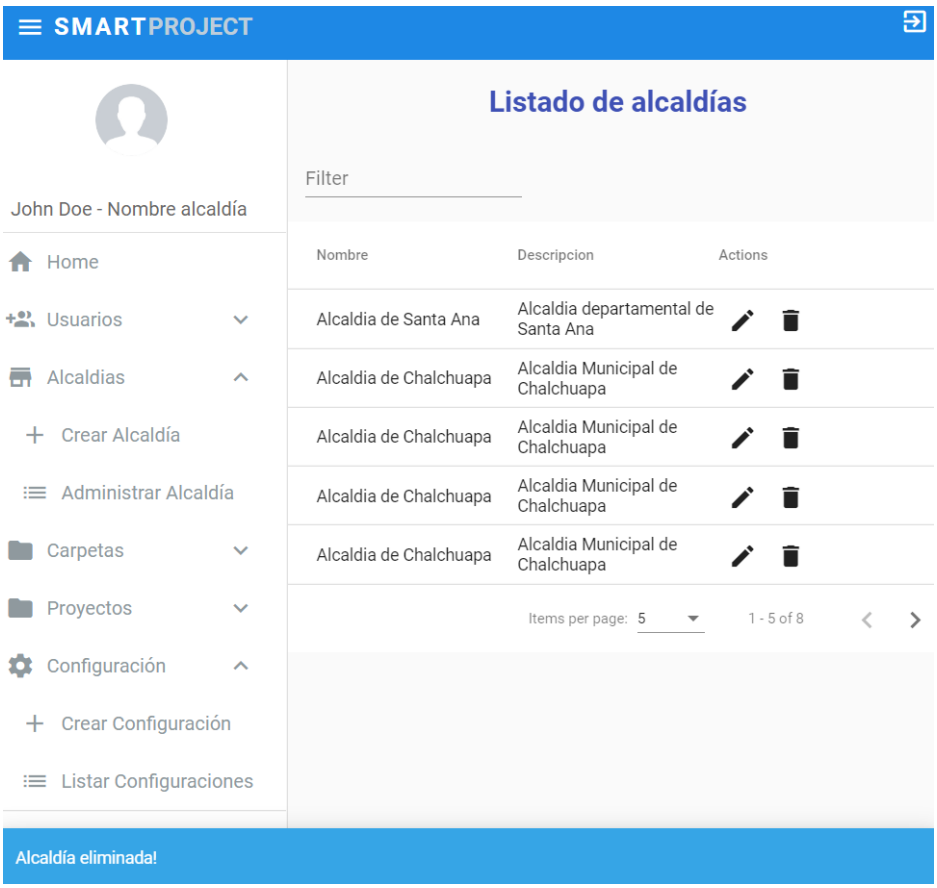
Luego de realizar las pruebas necesarias al sistema se realiza un informe de estado. Esto normalmente incluye información de estado en cada caso de prueba (porcentaje planificado, desarrollado, ejecutado, aprobado, fallido y bloqueado) y la probabilidad de completar el ciclo de prueba según lo programado. Se clasifican los resultados para definir las acciones a realizar. En caso de ser errores, son atendidos de forma inmediata y con prioridad alta; en caso de ser adecuaciones, solo se realizan modificaciones no prioritarias pero necesarias.

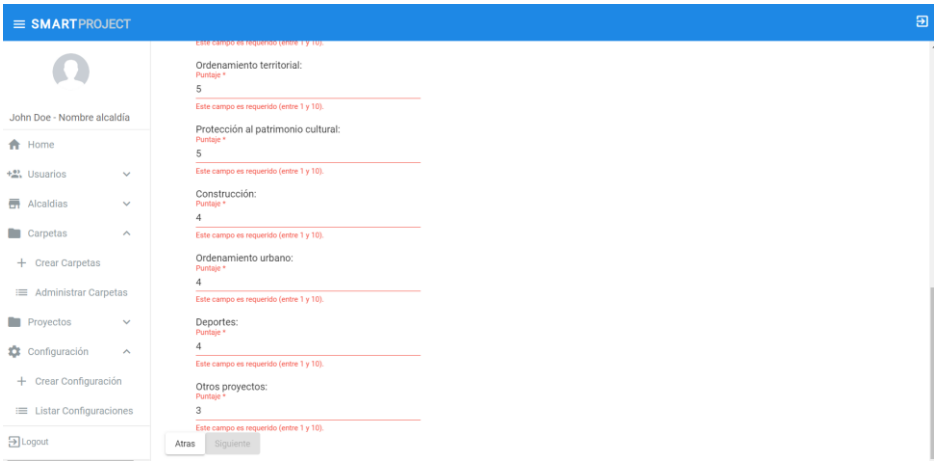
4.7.3 Listado de bugs encontrados en el sistema

B001	Inicio de sesión
Descripción	La sesión se mantiene activa aun cuando el navegador se ha cerrado.
Proceso esperado	La sesión debe finalizar después de 15 minutos de inactividad o al cerrar la pestaña.

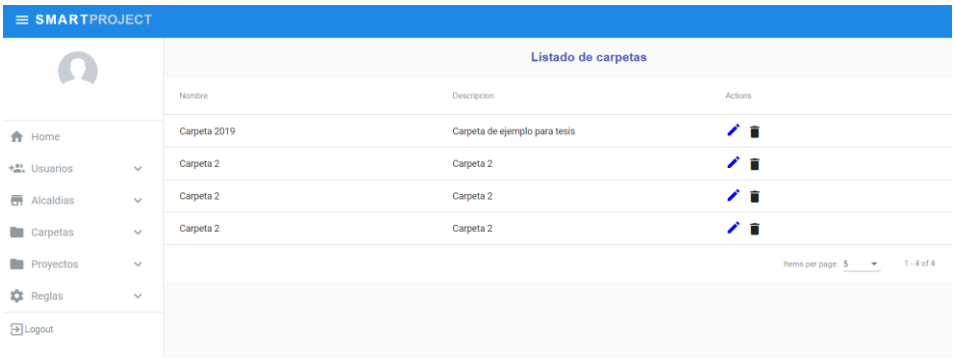
B002	Creación de Alcaldía
Descripción	Al crear una nueva alcaldía y dar clic en guardar, se muestra un mensaje “Problemas creando alcaldía”. Al listar las alcaldías, si se encuentran creadas.
Proceso correcto	1. Usuario ingresa información. 2. Clic en guardar 3. Mostrar mensaje de “Alcaldía creada exitosamente” (Si hay un error mostrar el mensaje de error específico). 4. Regresar a ventana home
Captura	

B003	Eliminación de Alcaldía
Descripción	Aunque se muestra un mensaje sobre la eliminación de la alcaldía, esta sigue siendo visible.

Captura	

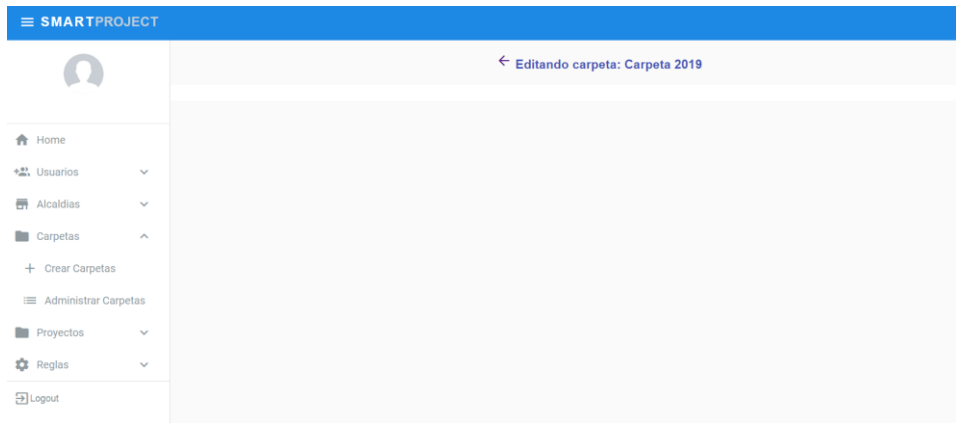
B004	Creación de reglas
Descripción	1. Los puntajes ingresados aparecen en rojo a pesar que son válidos. 2. No se habilita el botón siguiente luego de ingresar la toda la información.
	

B005	Nombre de Usuario
Descripción	1. No se ve el nombre de la persona logueada en el sistema. 2. El nombre debe poder verse abajo de la imagen de usuario en el menú lateral.

B006	Error en botones de acción
Descripción	1. Los botones de guardar, eliminar y editar, al finalizar su actividad no realizan ninguna actualización en el front-end. Esto genera casos de duplicidad. 2. Al finalizar su actividad debe realizar alguna clase de actualización o redireccionamiento.
	

B007	Guardar proyecto
Descripción	1. Al guardarse un proyecto no se muestra mensaje de confirmación. 2. Debería mostrarse un mensaje que confirme el guardado o el error en el guardado del proyecto.

B008	Editar carpeta
Descripción	1. Luego de crearse una carpeta no se permite editar los valores guardados.

B009	Listado de Proyectos
Descripción	1. El padding de la tabla de proyectos está muy pequeño. 2. padding-left y padding-right 24px y padding-top ; padding-bottom 6px.

CONCLUSIONES

La presente tesis tuvo como propósito diseñar y desarrollar un sistema informático para la evaluación de proyectos. Durante el proceso, se ha llegado a las siguientes conclusiones:

El sistema informático desarrollado responde a la necesidad de cada alcaldía de asegurar la confidencialidad de los datos de proyectos ingresados a ella.

El uso de este sistema informático en el sector gubernamental, en específico, el área de planificación mejora la priorización de proyectos al proporcionar una herramienta de evaluación multicriterio y personalizable. Lo anterior incide directamente en una mejor gestión del recurso económico disponible para una alcaldía.

El sistema informático desarrollado en esta tesis responde a los estándares modernos de desarrollo web, proporcionando una base de datos en línea, así como la capacidad de acceder desde cualquier dispositivo que posea conexión a internet. Esta es la respuesta a las exigencias actuales de acceso inmediato a la información que requieren los cuerpos gubernamentales como lo son las alcaldías. Es entonces natural pensar que los sistemas informáticos formaran parte clave en los procesos de toma de decisiones dentro las políticas de gobierno de cualquier lugar.

RECOMENDACIONES

Durante el proceso del diseño y desarrollo del sistema, se han considerado las siguientes recomendaciones:

Actualmente el sistema cuenta con cinco indicadores definidos a nivel de código, es decir que no son editables desde el nivel de usuario. A pesar de que la definición de evaluaciones multi criterios representan una muy útil funcionalidad del sistema, se recomienda para próximas versiones o desarrollos el habilitar la posibilidad de crear más indicadores para las evaluaciones de proyectos. Esto permitiría que el sistema tuviera una longevidad más alargada.

En consideración a las funcionalidades y utilidades que proporciona el sistema, se recomienda diseñar e implementar un proyecto de mejora y actualización tecnológica en las áreas de planificación de proyectos en el sector gubernamental, con el fin de implementar el sistema desarrollado en esta tesis.

REFERENCIAS BIBLIOGRAFICAS

Ing. Quevedo, Ing. Calderon, E (2014). *Diseño de un sistema de Evaluacion de proyectos de inversion, impulsados por las Alcaldias Municipales del departamento de Santa Ana, en la zona oocidental de El Salvador (Tesis de posgrado).* Universidad de El Salvador, Santa Ana, El Salvador, Página

Investigacion Holistica. (2 de marzo de 2014). Obtenido de http://investigacionholistica.blogspot.com

Asamblea Legislativa El Salvador. (1986). *Codigo Municipal.* San Salvador: Imprenta Nacional.

Asamblea Legislativa El Salvador. (1988). *Ley de Creación del Fondo para el Desarrollo Económico.* San Salvador: Imprenta Nacional.

Baker, J. L. (2010). *Evaluacion del Impacto de los Proyectos de Desarrollo en la Pobreza. Manual para profesionales.* Washington: Naciones Unidas.

Bretton Woods Projects. (22 de Febrero de 2008). *Bretton Woods Projects*. Recuperado el 2014 de Marzo de 13, de Bretton Woods Projects: http://www.brettonwoodsproject.org/es/2008/02/art-560746/

Castaneda, F. H. (12 de Agosto de 2013). Evaluacion de Proyectos Municipales. (M. R. Quevedo, Entrevistador)

Cohen, E. M. (2010). *Formulación, Evaluación y Monitoreo de proyectos sociales.* Santiago: Cepal.

Ehrhardt, M. C. (2007). Finanzas Corporativas. En M. C. Ehrhardt, *Finanzas Corporativas* (pág. 672). DF: Cengage Learning Editores.

Ernesto Cohen, R. F. (1995). *Evaluación de Proyectos Sociales.* Mexico: Siglo XXI.

FODES. (5 de Julio de 2012). *Instituto Salvadoreño de Desarrollo Municipal*. Recuperado el 11 de Octubre de 2013, de Instituto Salvadoreño de Desarrollo Municipal: http://www.isdem.gob.sv/index.php?view=items&cid=1%3APreguntas+Frecuentes+ASTCA&id=4%3Aque-es-el-fodes&option=com_quickfaq

Fontaine, E. (2007). *La evaluación privada y social de proyectos, el rol del estado.* Bolivia: Cepal.

Fontaine, E. (2010). *Evaluacion Social de Proyectos.* Santiago: Alfayomega.

Hernández Sampieri, R. (1999). *Metodología de la Investigación.* México: McGraw Hill.

Hernandez, A. (2005). *Formulación y Evaluación de Proyectos de Inversión.* Madrid: Paraninfo.

Hugo Navarro, K. K. (2006). *Pauta metodológica de evaluación de impacto ex-ante y ex-post de programas sociales de lucha contra la pobreza. Aplicación metodológica.* Mexico: Cepal.

Hurtado de Barrera, J. (2008). *Metodología de la investigación, una comprensión holística.* Venezuela: Payes Quirón.

Kendall, K., & Kendall, J. (2005). *Análisis y Diseño de Sistemas.* Mexico: Pearson.

Mideplan. (5 de Abril de 2013). *Sistema Nacional de Inversiones.* Recuperado el 4 de Agosto de 2013, de Sistema Nacional de Inversiones: http://sni.ministeriodesarrollosocial.gob.cl/fotos/Agua%20Potable%20Rural%202013.pdf

Mideplan. (5 de Abril de 2013). *Sistema Nacional de Inversiones.* Recuperado el 15 de Agosto de 2013, de Sistema Nacional de Inversiones: http://sni.ministeriodesarrollosocial.gob.cl/fotos/Agua%20Potable%20Rural%202013.pdf

Trading Economics. (4 de Marzo de 2014). *Trading Economics*. Recuperado el 18 de Septiembre de 2013, de Trading Economics: http://es.tradingeconomics.com/el-salvador/inflation-cpi

Valquiro, J. D. (2010). El Valor Presente Neto. *Pymes Futuro*, 22.

ANEXO

ANEXO 1: MANUAL DE USUARIO

Con el objetivo de otorgar soporte al sistema se ha descrito este manual el cual esta organizo de acuerdo con la secuencia de ingreso a las pantallas de la siguiente manera:

- Acceso a URL
- Ingreso al sistema.
- Ventana de Home.
- Creación y administración de alcaldías.
- Creación y administración de reglas
- Creación y administración de proyectos
- Creación y administración de carpetas
- Evaluaciones.

A. Acceso a URL

B. Ingreso al sistema.

Para el ingreso al sistema, se deberá contar con las credenciales de acceso (Usuario/contraseña), las cuales se ingresaran en los campos correspondientes.

1. Ingresar un usuario.
2. Ingresar la contraseña asignada.
3. Clic en Ingresar

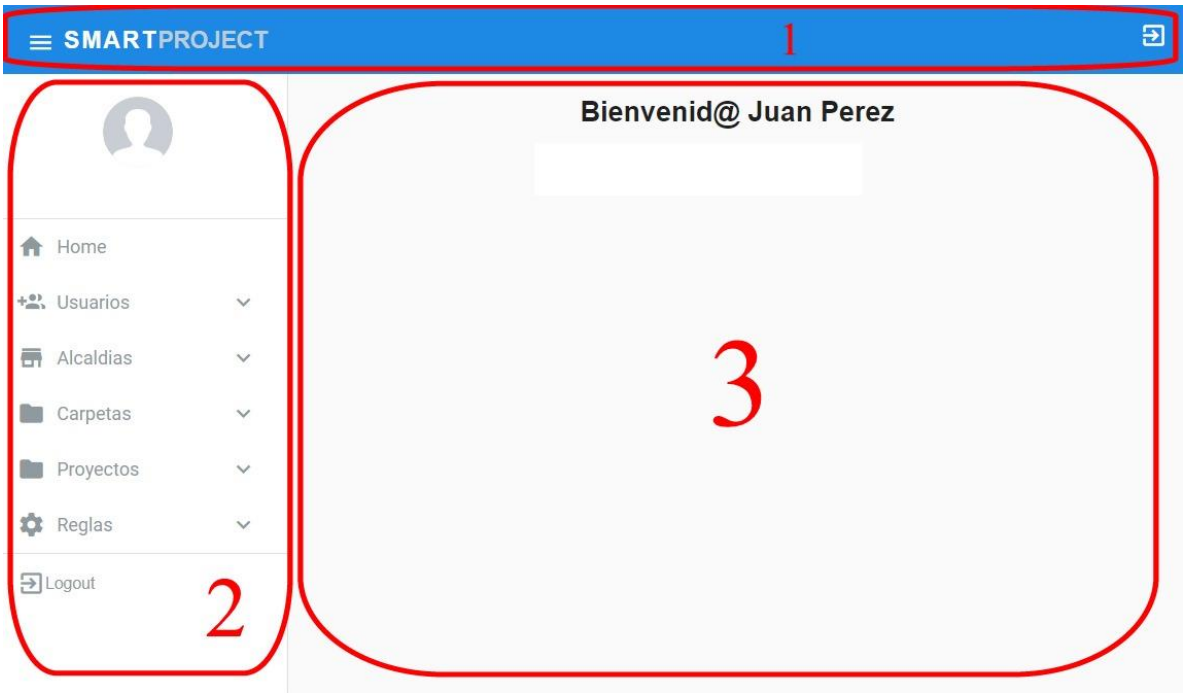
Ventana Home

Esta ventana de inicio y las partes que la constituyen se describen de la siguiente manera:

1. Barra de titulo
    a. Se muestra un botón para acoplar la barra de menús y ampliar el espacio de trabajo.
    b. El titulo de la aplicación.
    c. Botón para salir de la aplicación.

2. Barra de menús: Lista todas las opciones disponibles para cada usuario.
    a. Home
    b. Usuarios
    c. Alcaldías
    d. Carpetas
    e. Proyectos
    f. Reglas
    g. LogOut

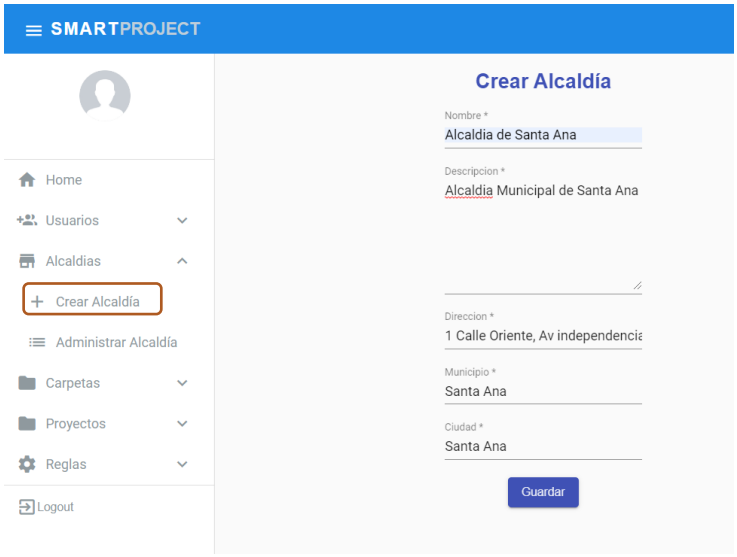
3. Espacio de trabajo: en este espacio es donde se trabajaran todo tipo de ingreso y verificación de información.



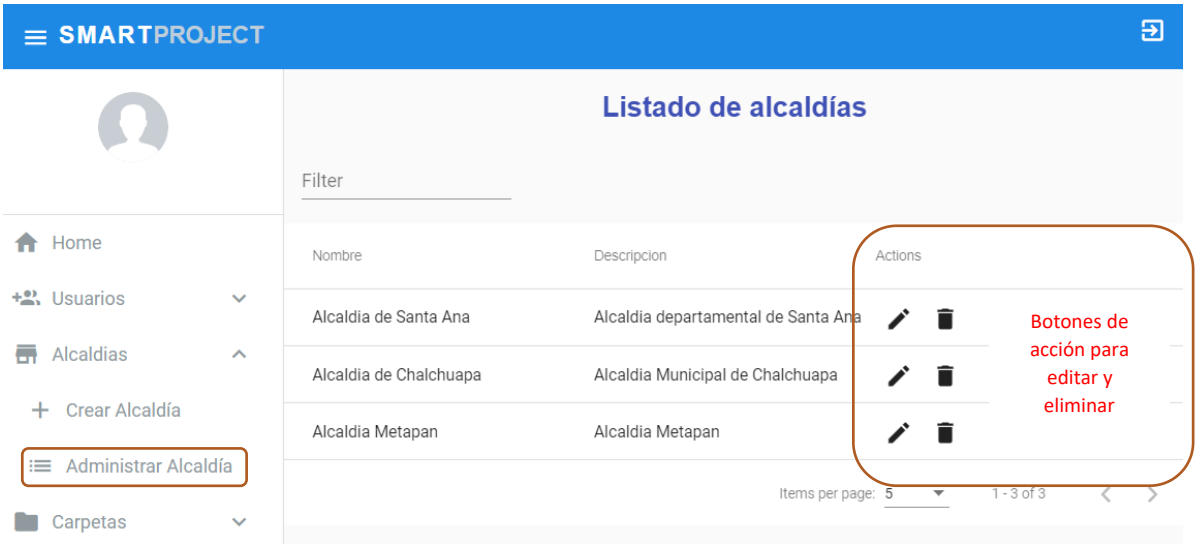
Creación y administración de alcaldías

Como primer punto de configuración es necesario crear una alcaldía, puesto que las siguientes funciones de este sistema lo solicitaran. Esta función solamente está disponible para usuarios con el rol ***ROOT.*** Para lo cual se necesitara realizar los siguientes pasos:

1. Clic en Menú ***Alcaldías***
2. Clic en ***Crear Alcaldía***
3. Aparecera en el área de trabajo un formulario donde se deberá especificar la información correspondiente para la creación.
    a. Nombre de Alcaldía
    b. Descripción
    c. Dirección
    d. Municipio
    e. Ciudad
4. Clic en ***Guardar.***



Luego de guardar los datos la vista se dirigirá a la pantalla ***Administrar Alcaldias.*** La cual nos mostrara el listado de alcaldías creadas.



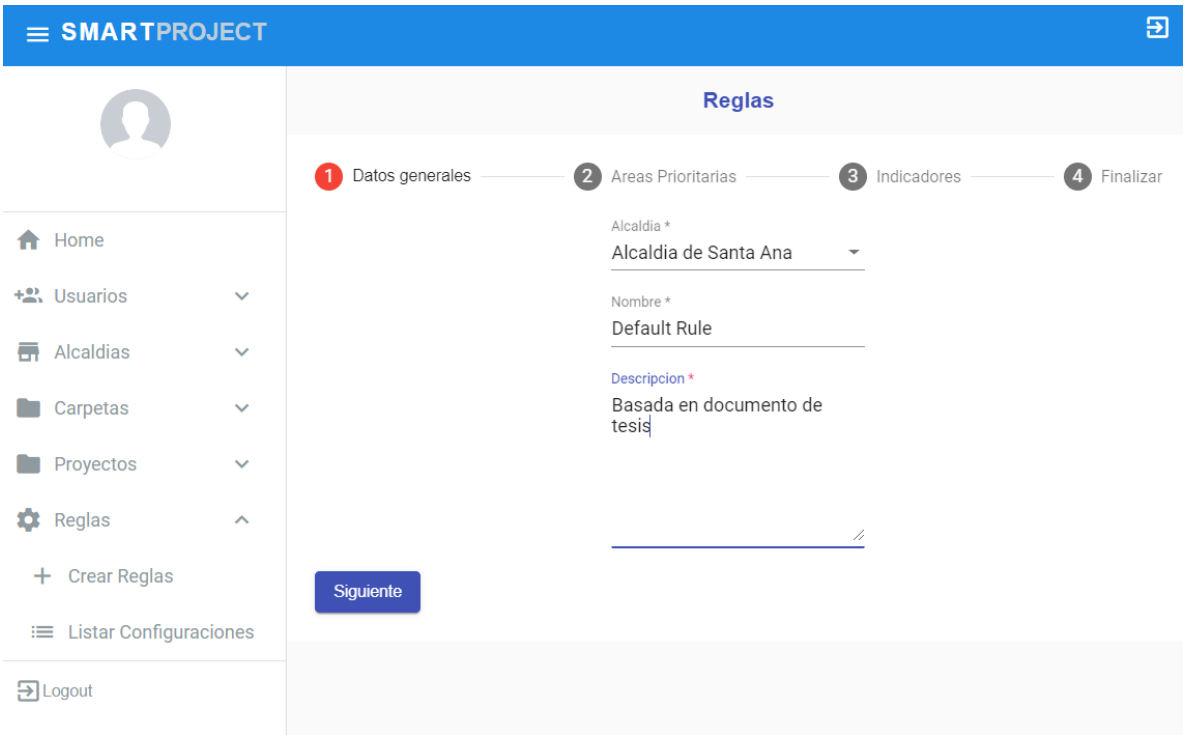
Creación y administración de Reglas

La próxima configuración necesaria es la creación de Reglas, las reglas son las configuraciones que darán un puntaje a los proyectos evaluados.

Esta configuración se hará a través de un workflow que mostrara paso a paso la información agrupada por rasgos comunes.

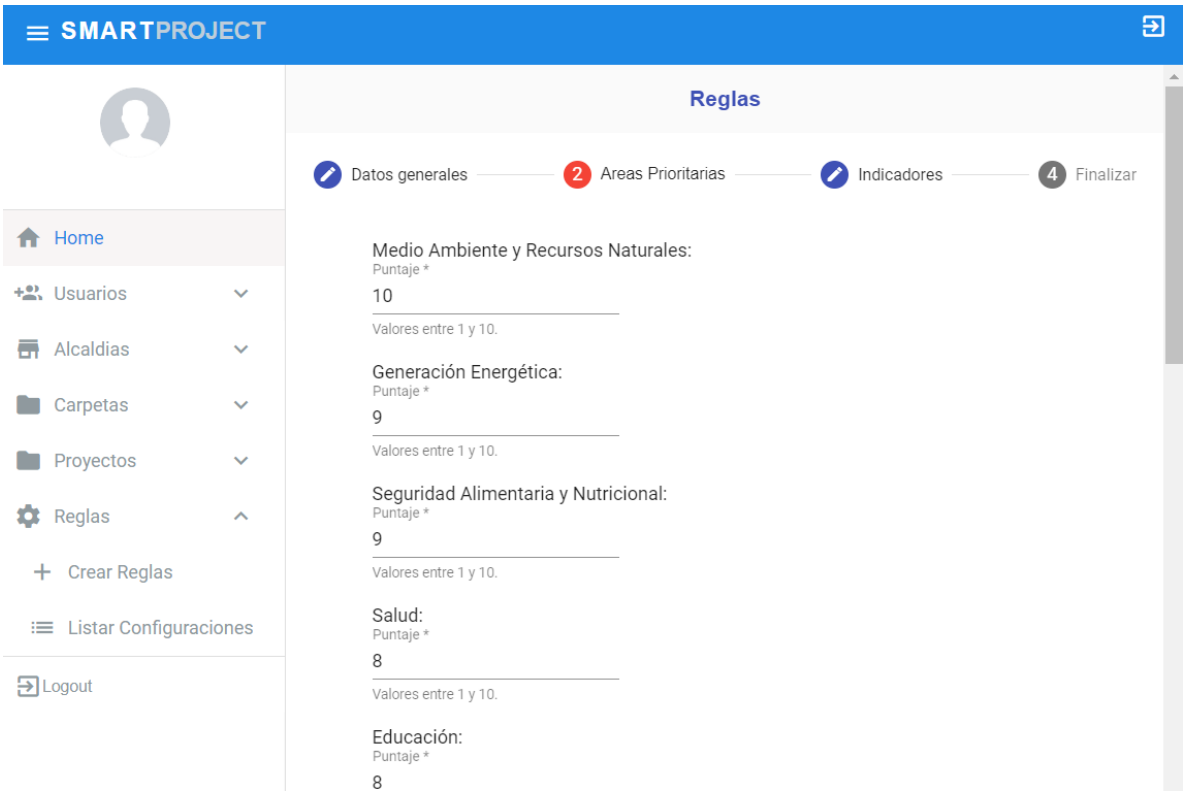
Paso 1. Datos Generales

1. Clic en ***Reglas***
2. Clic en ***Crear Reglas***
3. Ingresar datos generales
    a. Elegir Alcaldía
    b. Ingresar un nombre a la regla
    c. Agregar una descripción de referencia.
    d. Clic en ***Siguiente***



Paso 2. Areas Prioritarias

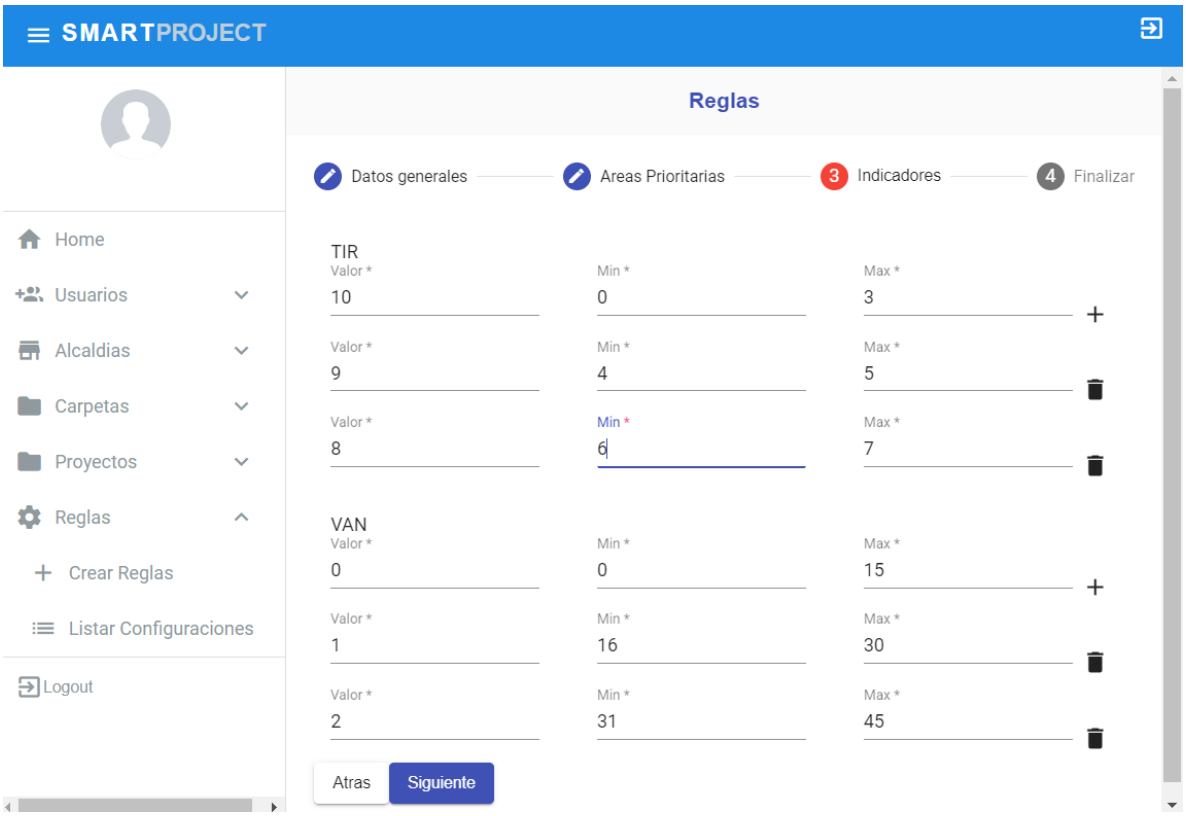
Se mostrara el siguiente paso que implica en darle puntaje a las ***Areas Prioritarias.*** Estos puntajes pueden colocarse a gusto de quien realice las evaluaciones.



Luego de indicar los valores para cada Area, dar clic en ***Siguiente.***

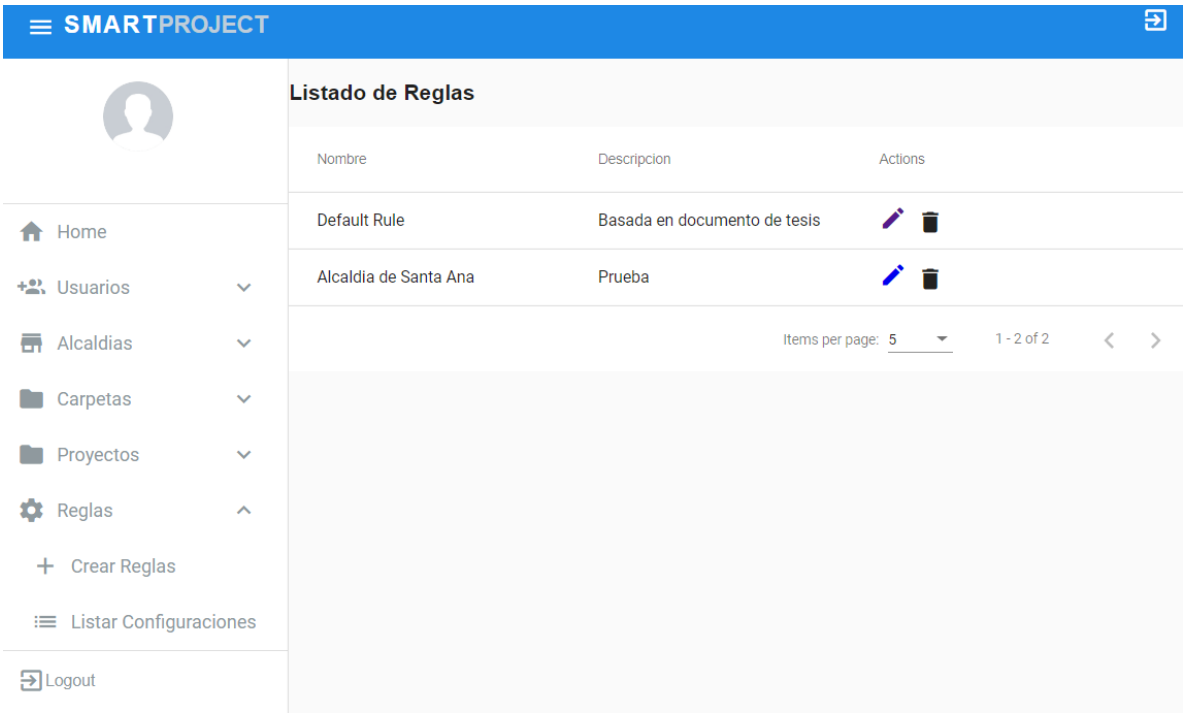
Paso 3. Indicadores

La siguiente parte es ingresar la configuración de los indicadores TIR y VAN.



Luego dar clic en ***Siguiente*** y por último dar clic en ***Guardar.***

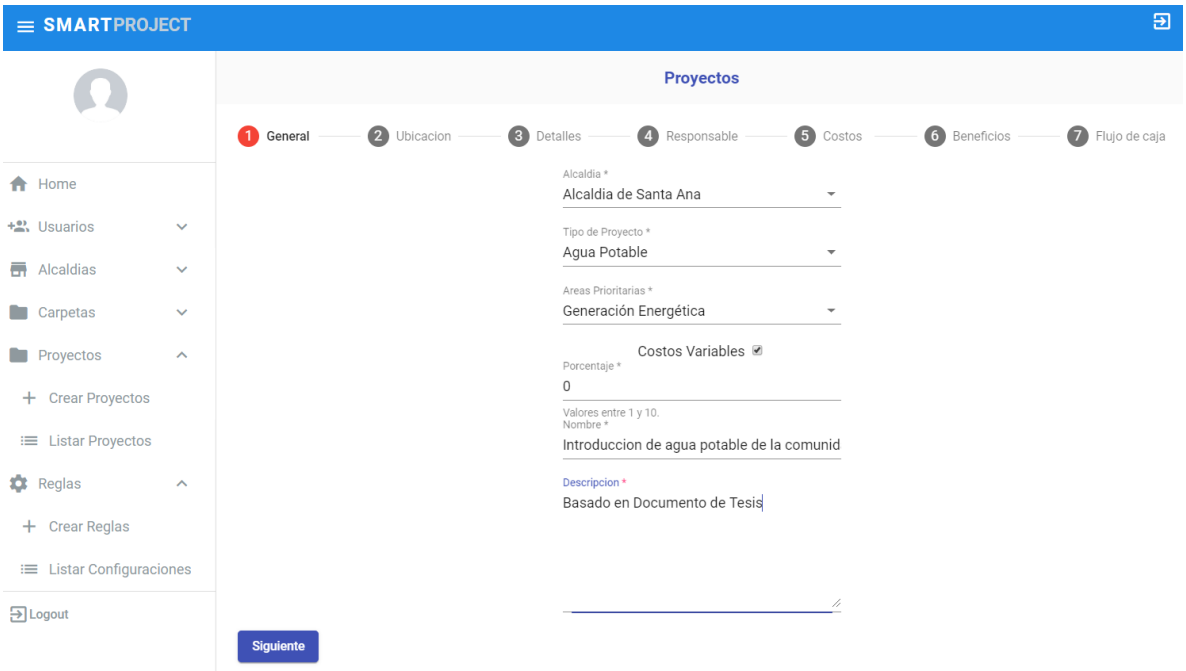
Con esto hemos creado nuestra regla y el sistema nos llevara a ver las Reglas creadas en ***Listar Configuraciones.*** Esta ventana nos permitirá acceder a nuestras reglas ya creadas para eliminar y editar.



Creación y administración de proyectos

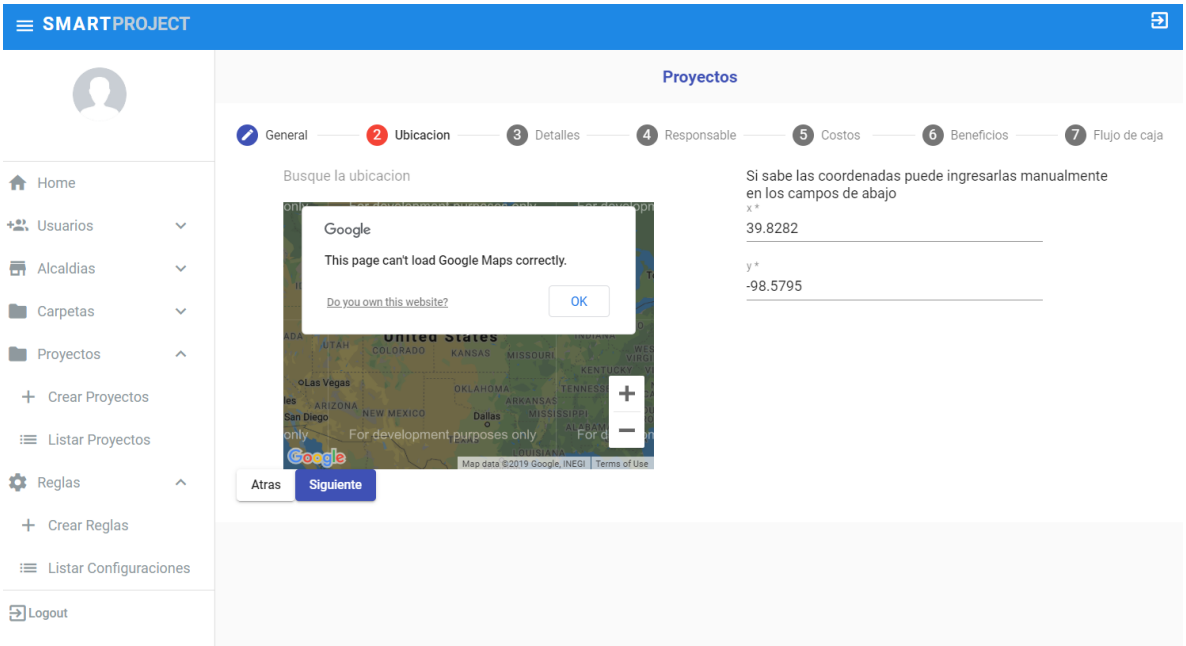
Continuando con la lógica del software, corresponde ingresar información de proyectos, para ellos llenaremos el workflow correspondiente.

1. Clic en Proyectos
2. Clic en Crear Proyectos
3. Ingresar la información requerida
    a. Alcaldía
    b. Tipo de Proyectos: es esencial indicar el tipo de proyectos puesto que de esto depende la generación de los costos y beneficios en las siguientes pasos.
    c. Seleccionar el Area Prioritaria
    d. Elegir si el proyecto es de Costos Variables.
    e. Ingresar un nombre.
    f. Agregar una descripción del mismo.
    g. Clic en Siguiente.



Paso 2. Ubicación.

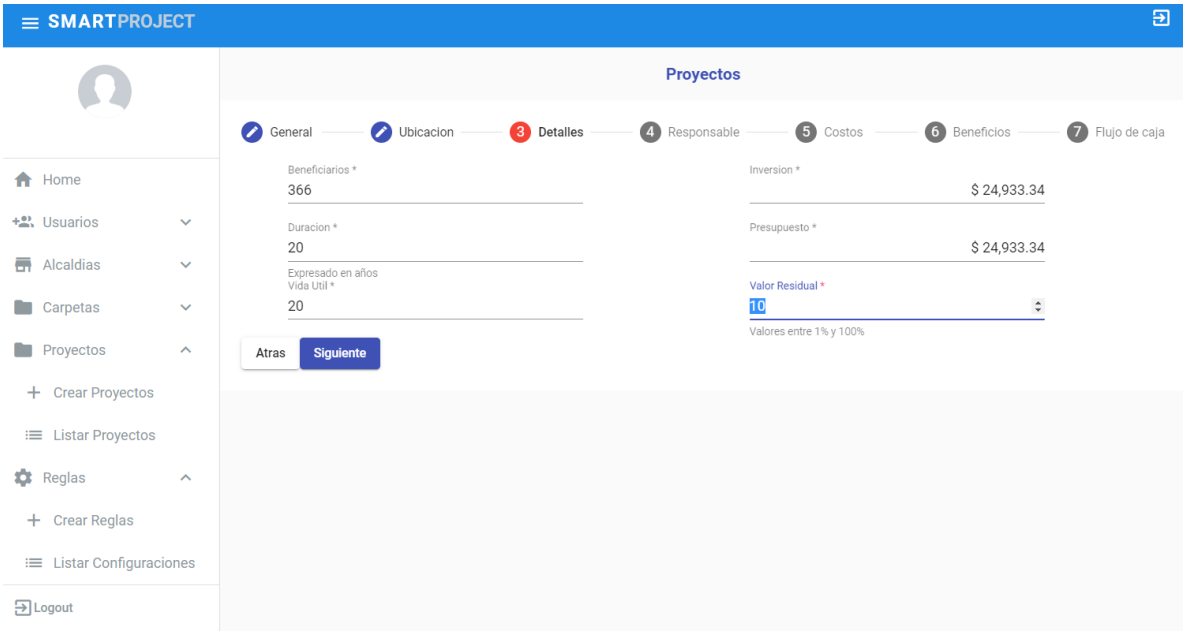
El sistema provee de un servicio para poder ubicar el lugar donde el proyecto será ejecutado. Para eso solo debemos indicar las coordenadas o en el mapa ubicarlo con un clic.



Paso 3. Detalles

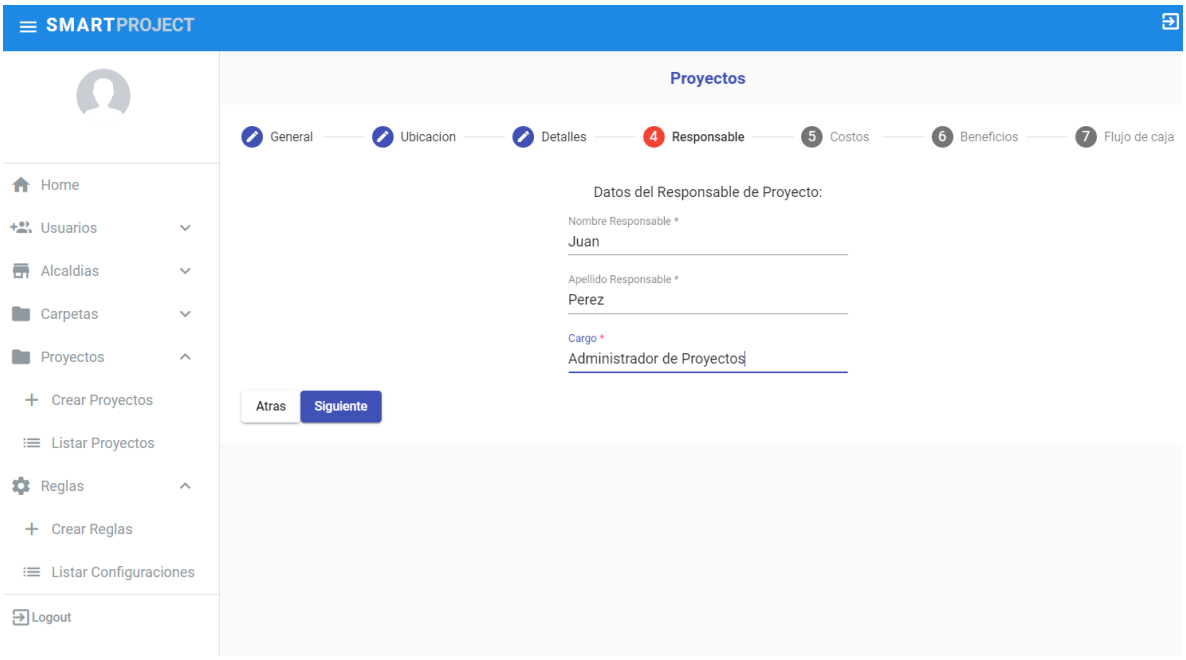
En este paso indicaremos los valores monetarios generales.

1. Beneficiarios: cantidad de personas que serán alcanzadas con el proyecto.
2. Duracion: el tiempo en años en los que será alcanzado el proyecto.
3. Vida útil: tiempo en el cual el proyecto mantendrá la inversión.
4. Inversion:
5. Presupuesto: Para el calculo será el mismo valor que el dato de inversión.
6. Valor residual: colocaremos un valor residual de un 10%
7. Luego damos clic en ***Siguiente.***



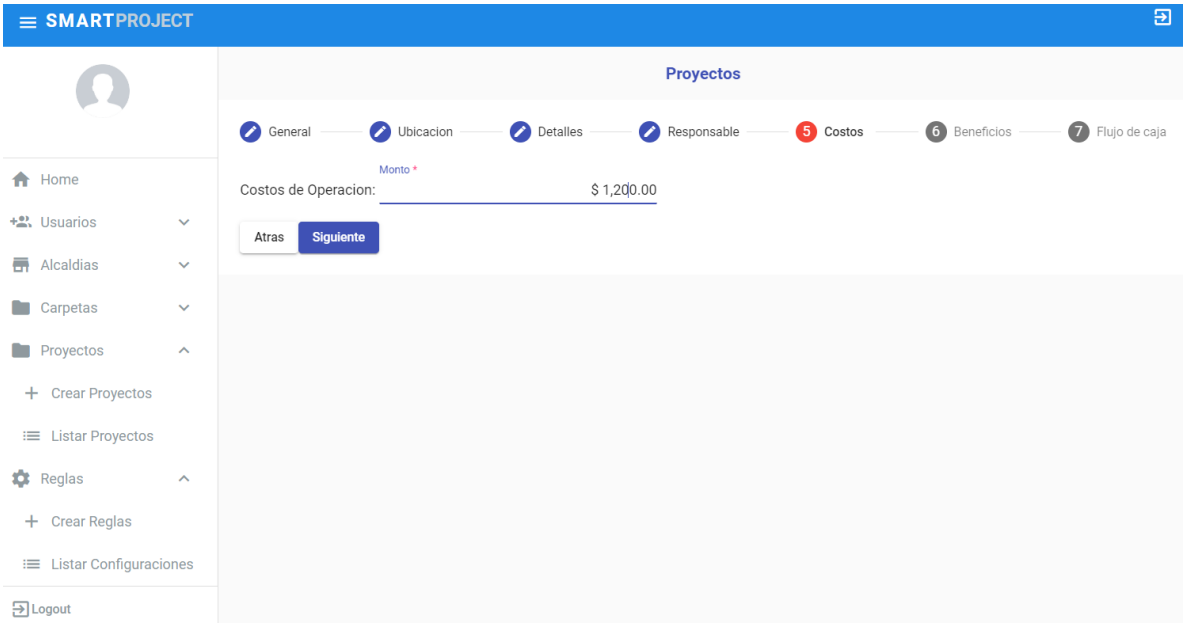
Paso 4. Responsable.

Se deberá indicar un responsable, un encargado de proyecto, un director de proyecto, lo que la unidad municipal decida como encargado del cual.



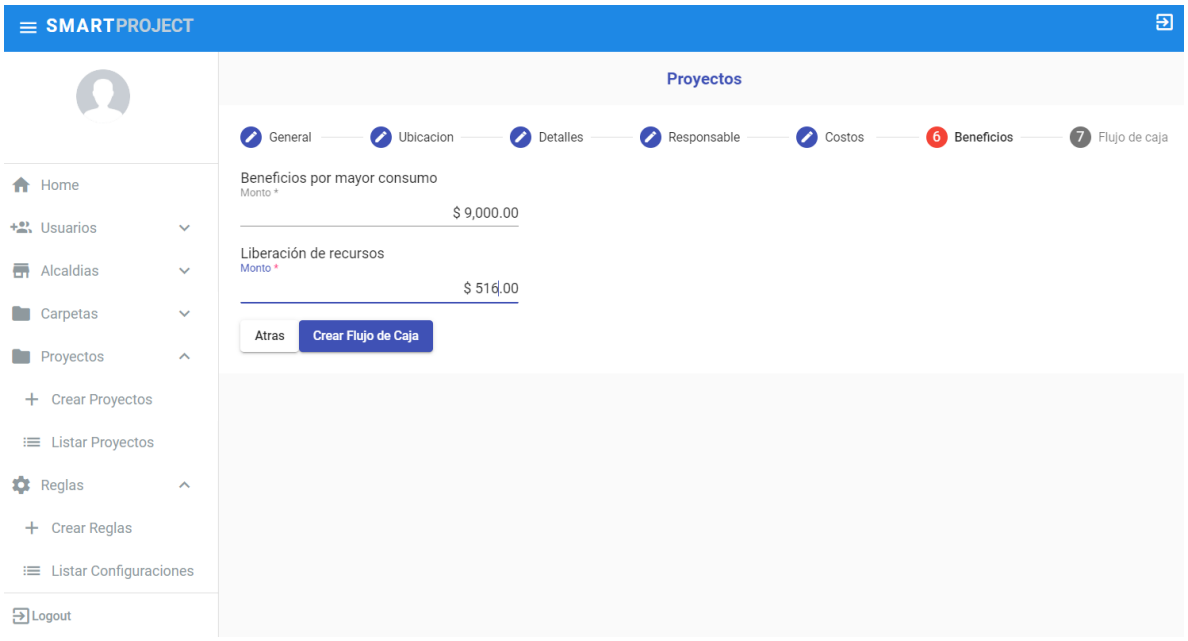
Paso 5. Costos

Se deberá ingresar la cantidad económica de los diferentes costos, Estos costos están relacionados al tipo de proyecto de proyecto selección. Para mayor detalle ver la sección 2.XXXXX

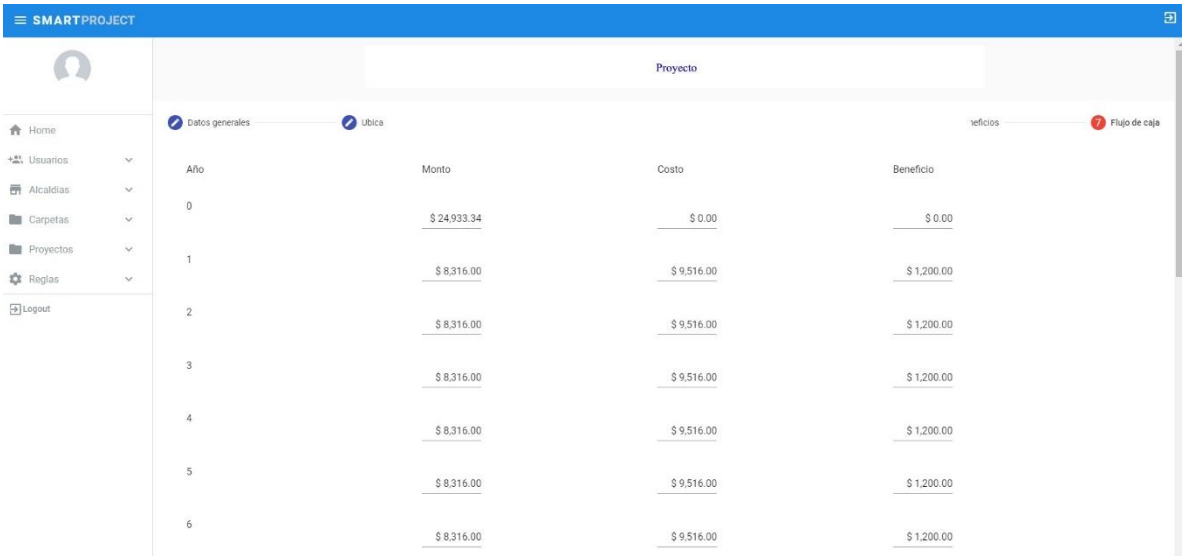


Paso 6. Beneficios

Se deberá ingresar los beneficios económicos relacionados al tipo de proyecto.

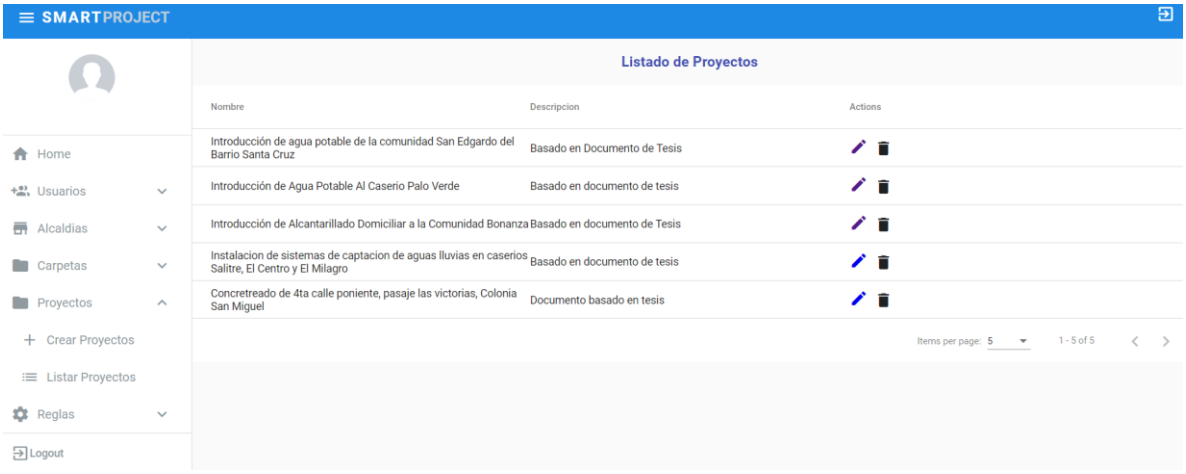


Luego dar clic en ***Crear Flujo de Caja.*** Esto nos mostrara una matriz editable para el usuario, en dado caso, el costo o beneficio de algún año en específico tenga un cambio. De lo contrario deberá de dejarse intacto.



El sistema permitirá al usuario listar los proyectos creados como también la edición de ellos. Para esto:

Clic en ***Listar Proyectos,*** se mostrara la siguiente ventana:

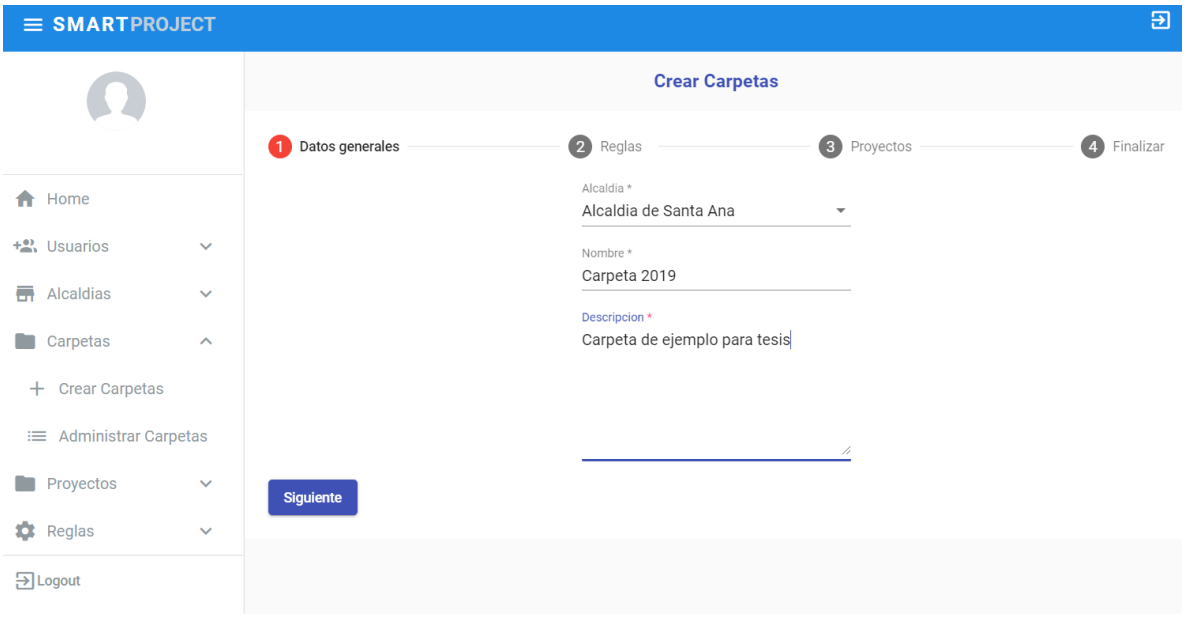


Creación y administración de carpetas

Los proyectos solo podrán evaluarse si están contenidos en ***Carpetas.*** Para ello es necesario crearlas siguiendo estos pasos.

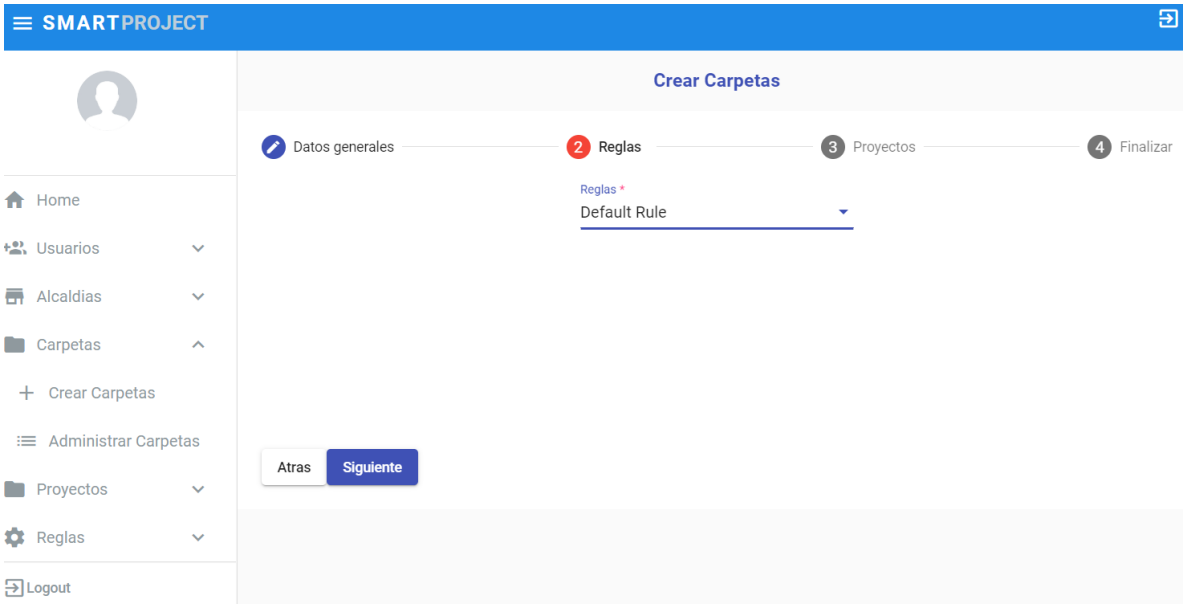
Paso 1. Datos Generales

1. Dar clic en ***Carpetas***
2. Dar clic en ***Crear Carpetas***
3. Ingresar datos generales
    a. Nombre de Alcaldia
    b. Nombre de Carpeta
    c. Descipcion.
4. Dar clic en ***Siguiente***



Paso 2. Reglas

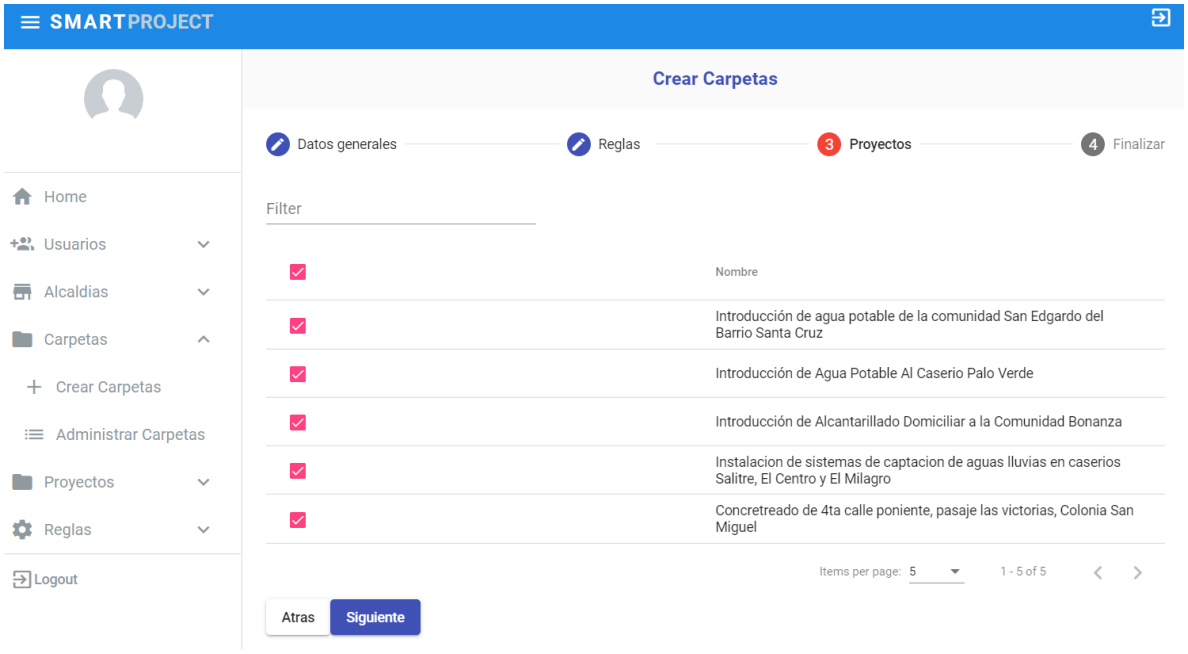
Es necesario agregar la regla con la que se evaluaran los proyectos en las carpetas. Para ello debemos elegir una de las reglas anteriormente creadas. Seleccionamos la creada en este manual: ***Default Rule.***



Paso 3. Proyectos

Es necesario agregar 2 proyectos o mas, estos no cambiaran de ubicación, solamente se agregaran a la carpeta y estarán disponibles para agregarse a otras carpetas para realizar diferentes evaluaciones.

Para ello, seleccionamos los proyectos deseados. Y luego en ***Siguiente***



Evaluaciones

El sistema proveerá la evaluación para ello se deberá acceder los siguietes pasos:

1. Clic en ***Administrar Carpetas.***
2. Clic en botón de acción ***Evaluar.***


SMARTPROJECT
Evaluacion: Carpeta 2019
Home
Usuarios
Alcaldias
Carpetas
Proyectos
Reglas
Logout
proyecto
puntaje
Introducción de agua potable de la comunidad San Edgardo del Barrio Santa Cruz
43
Introducción de Agua Potable Al Caserio Palo Verde
34
Introducción de Alcantarillado Domiciliar a la Comunidad Bonanza
6
Instalacion de sistemas de captacion de aguas lluvias en caserios Salitre, El Centro y El Milagro
18
Concretreado de 4ta calle poniente, pasaje las victorias, Colonia San Miguel
7
Atras
Guardar

ANEXO 2: REPORTE

SMARTPROJECT	**Reporte de Evaluación Carpeta 2019**

A continuación, se muestran la puntuación de los proyectos evaluados en **Carpeta 2019.**

Información General	
Fecha de Elaboración de Informe	25/10/2019
Hora de Elaboración de Informe	2:35 pm
Organización	Alcaldía Municipal de Santa Ana
Encargado	Juan Pérez
Proyectos Evaluados	5 Proyectos
Regla Utilizada	Default Rule

	Indicadores					
Nombre de proyecto	**Rentabilidad con Enfoque Social**	**Valor Actual Neto**	**Tiempo de Retorno de Inversión**	**Beneficiarios**	**Área Prioritaria**	**Puntaje**
Introducción de agua potable de la comunidad San Edgardo del Barrio Santa Cruz	10	10	8.5	10	4	**43**
Introducción de Agua Potable Al Caserío Palo Verde	6	9	7	8	4	**34**
Instalación de sistemas de captación de aguas lluvias en caseríos Salitre, El Centro y El Milagro	1	2	2	7	6	**18**
Concretreado de 4ta calle poniente, pasaje las victorias, Colonia San Miguel	1	1	1	0	4	**7**
Introducción de Alcantarillado Domiciliar a la Comunidad Bonanza	0	0	0	2	4	**6**

ANEXO 3: ENTREVISTA:

¿Cuántos proyectos son ejecutados por la municipalidad en un año?

¿Qué tipo de proyectos son ejecutados por la Municipalidad?

¿Cuál es el proceso de evaluación de proyectos que realizan?

¿Cuáles criterios son tomados en cuenta al momento de decidir ejecutar un proyecto?

¿De todos los criterios mencionados, existe una cuantificación formal sobre ellos?

¿Quién toma la decisión final para la ejecución de un proyecto?

¿Existen proyectos municipales que hayan fracasado?

¿Considera importante el establecer un procedimiento formal para la evaluación de proyectos?

¿Qué características desearía que tenga la metodología de evaluación de proyectos municipales?