ANÁLISIS Y PROTOTIPADO DE UN COMPONENTE DE SOFTWARE DE EXPLORACIÓN DE DATOS, INTEGRADO A LA *"ARQUITECTURA DE VISUALIZACIÓN UTILIZANDO DASHBOARDS"*

BRIGITTE ANGELICA HINCAPIE ORTIZ
WILSON DANIEL PINTO RIOS

UNIVERSIDAD TECNÓLOGICA DE PEREIRA
FACULTAD DE INGENIERÍAS: ELÉCTRICA, ELECTRÓNICA, FISICA Y CIENCIAS DE LA COMPUTACIÓN
MAESTRÍA EN INGENIERÍA DE SISTEMAS Y COMPUTACIÓN
PEREIRA
FEBRERO 2019

ANÁLISIS Y PROTOTIPADO DE UN COMPONENTE DE SOFTWARE DE EXPLORACIÓN DE DATOS, INTEGRADO A LA *"ARQUITECTURA DE VISUALIZACIÓN UTILIZANDO DASHBOARDS"*

BRIGITTE ANGÉLICA HINCAPIÉ ORTIZ
WILSON DANIEL PINTO RÍOS

TRABAJO DE INVESTIGACIÓN PRESENTADO COMO REQUISITO PARA OPTAR AL TÍTULO DE: MAGISTER EN INGENIERÍA DE SISTEMAS Y COMPUTACIÓN

DIRECTOR: ALEJANDRO RODAS VÁQUEZ

UNIVERSIDAD TECNOLÓGICA DE PEREIRA
FACULTAD DE INGENIERÍAS: ELÉCTRICA, ELECTRÓNICA, FÍSICA Y CIENCIAS DE LA COMPUTACIÓN
MAESTRÍA EN INGENIERÍA DE SISTEMAS Y COMPUTACIÓN
PEREIRA
FEBRERO 2019

NOTA DE ACEPTACIÓN

_____

_____

_____

_____

_____

_____

_____

FIRMA DIRECTOR

_____

FIRMA JURADO

_____

FIRMA JURADO

Pereira, Risaralda, Febrero 2019

Agradecimientos

Queremos agradecer a Dios por darnos la posibilidad cada día de enfrentar nuevos retos y superarlos de la mejor forma, a nuestros padres y familias los cuales son el soporte de nuestra vida y siempre están allí para apoyarnos en cada paso, a la Universidad Tecnológica de Pereira la cual más allá de ser una institución educativa ha sido un hogar en la cual nos hemos formado y nos ha brindado todas las herramientas para alcanzar nuestros logros educativos y profesionales. A nuestro director de proyecto Alejandro Rodas Vásquez quien siempre estuvo allí guiándonos y apoyando todo este proceso de investigación y trabajo. A cada uno de nuestros profesores por sus enseñanzas y a los compañeros quienes apoyaron este proyecto de grado. Gracias a todos quienes acompañaron este proceso he hicieron posible el sueño de estar aquí el día de hoy a punto de recibir un título tan valioso como lo es el de magister en ingeniería de sistemas y computación.

Tabla de contenido

Índice de figuras

Índice de Anexos

Índice de tablas

Resumen

Los sistemas tecnológicos han experimentado cambios radicales en los últimos años, la transición de sistemas aislados e independientes, a un conjunto de sistemas interconectados entre si generando y almacenando información constantemente, ha dado paso a la analítica de datos. Anteriormente las empresas y los sistemas tecnológicos enfocaban sus esfuerzos e investigaciones a buscar formas de conseguir mayor almacenamiento, menor tiempo de respuesta o información disponible permanentemente, pero este paradigma ha ido cambiando gracias a la analítica de datos, la empresa actual no solo requiere sus datos apilados en una base de datos y disponibles para ser extraídos también busca que sus datos muestren información para ser analizada y usada con el fin de obtener ventajas competitivas sobre las demás empresas o para procesos de mejoramientos propios en procesos productivos, para esto la analítica de datos nos brinda en cada una de sus fases, formas de procesar los datos para extraer lo mejor de ellos, una de las fases más importantes es la exploración de datos, la cual se enfoca en brindar al usuario herramientas suficientes para que posterior al proceso de extracción y limpieza de datos, el usuario pueda manipular la información y obtener visualizaciones útiles, otorgando la posibilidad de analizar sus datos tanto de manera individual como también encontrar patrones de comportamiento y relaciones entre las distintas variables identificadas en los datos. El presente proyecto busca analizar y diseñar un componente de software partiendo de una arquitectura definida, brinde a los usuarios la posibilidad de realizar procesos de exploración de datos donde pueda obtener visualizaciones específicas y dar un valor agregado a su información con el fin de apoyar el proceso de toma de decisiones.

Abstract

Technological systems have undergone radical changes in recent years, the transition from isolated and independent systems, to a set of interconnected systems generating and storing information constantly, has given way to data analytics. Previously, companies and technological systems focused their efforts and research on finding ways to achieve greater storage, decrease response times or permanently available information, but this paradigm has been changing thanks to data analytics, the company of today does not only require that your data is stacked in a database and are available to be extracted, it also wants that your data show information that can be analyzed and used to obtain competitive advantages over other companies or for own improvement processes in production processes, to achieve this data analytics provides us in each of its phases, ways to treat the data to extract the best of them, one of the most important phases is the exploration of data, which focuses on providing the user with sufficient tools so that later to the data extraction and cleaning process, the user can manipulate the info and obtain useful visualizations that give you the possibility to analyze your data both individually and also find patterns of behavior and relationships between the different variables identified in the data. This project wants to analyze and design a software component that, based on a defined architecture, provides users with the possibility of performing data exploration processes in which they can obtain specific visualizations and add value to their information in order to support the decision-making process.

Parte I

Introducción

1. Generalidades

1.1. Introducción

Una de las principales características de los sistemas de información existentes, es su creciente capacidad para el almacenamiento de grandes volúmenes de datos. Donde la recolección, búsqueda e interpretación de estos, requiere de herramientas de visualización con el fin de ayudar en los procesos de toma de decisiones.

No obstante, antes de crear la visualización para el análisis de un conjunto de datos (*data set*) es imperativo ejecutar previamente la fase de exploración, tal y como se plantea en el proceso de análisis de datos *(Nelli, 2015*).

De tal forma, desde el punto de vista de la Arquitectura de Software, una aplicación orientada al análisis de datos está compuesta de varios módulos integrados y especializados en realizar procesos y funcionalidades específicas para cada una de las fases que corresponden a la analítica de datos.

En el presente proyecto se plantea a partir de una arquitectura ya existente construida por (Betancurt Obando & Abril Pérez, 2018) realizar el análisis, diseño y prototipado de un componente de software orientado a realizar funcionalidades de exploración de datos que brinde a la arquitectura, la posibilidad de realizar análisis univariado y bivariado de variables categóricas y numéricas, que permita a los usuarios del software tener herramientas suficientes para obtener de los datos obtenidos a partir de las fuentes de datos, información y conocimiento adicional, que sea útil para el proceso de toma de decisiones a partir del problema que se plantee en la fase inicial de definición del problema en el proceso de análisis de datos.

También se expone y se aplica la metodología C4 en cada uno de sus cuatro niveles con el fin de realizar una descripción desde una visión general del sistema hasta un nivel de especificidad alto en donde cada uno de los componentes construidos es descrito tanto a nivel estructural, como a nivel de la integración y colaboración que realiza con cada uno de los componentes de la arquitectura.

1.2. Planteamiento del problema

Siguiendo el enfoque Arquitectónico de Software, la presente investigación toma como base la propuesta planteada por (Betancurt Obando & Abril Pérez, 2018) denominada "Diseño de una Arquitectura por Componentes para la Visualización de Datos Utilizando Dashboards[1]".

En dicho proyecto se identifica en primera instancia dos tipos de usuarios en el ámbito de la visualización de datos. El primer usuario se caracteriza por ser un experto en el área con conocimientos en frameworks o lenguajes de programación para tal fin. Como lo es D3[2] o el lenguaje de programación R[3] y su librería ggplot[4], respectivamente. El segundo usuario se caracteriza por emplear herramientas de software de tipo genérico como Tableau[5] o Many Eyes[6]. Sin embargo, al ser herramientas estandarizadas se sacrifica la posibilidad de personalizar las visualizaciones resultantes. De esta forma, en el trabajo propuesto por (Betancurt Obando & Abril Pérez, 2018) se plantea una arquitectura basada en componentes (cada uno con una funcionalidad específica) permitiendo a los usuarios ya mencionados visualizar un conjunto de datos determinado, utilizando tanto las gráficas y herramientas ofrecidas por la propia arquitectura como también la creación de nuevas visualizaciones a través del componente denominado *Chart Component* empleando el framework AngularJS[7] y la notación JSON (JavaScript Object Notation)[8].

De tal manera, analizando la arquitectura desarrollada por (Betancurt Obando & Abril Pérez, 2018) se identificó la falta del componente responsable de ejecutar la tarea de exploración de datos, sin el cual el usuario podría tener una percepción errada de los datos, dificultándole un adecuado análisis de los mismos para la toma de decisiones.

[1] Son una de las técnicas más populares en visualización, donde hace referencia a que la información más importante debe estar consolidada en una pantalla, de manera que se pueda revisar solo con un vistazo
[2] D3 Data-Driven Documents. URL: https://d3js.org/
[3] https://www.r-project.org/
[4] https://cran.r-project.org/web/packages/ggplot2/
[5] http://www.tableausoftware.com/.
[6] http://services.alphaworks.ibm.com/manyeyes/home.
[7] https://angularjs.org/
[8] https://www.json.org/

1.3. Justificación

Dada la cantidad de datos requeridos por la industria en la actualidad, la información obtenida toma una gran importancia, partiendo de los datos recolectados, las empresas pueden construir conocimiento no solo de sus clientes a nivel comercial sino sobre sus procesos internos. De tal forma, se vuelve necesario el uso de técnicas de exploración de datos para caracterizar la información recolectada y correlacionar los eventos representados en ella, brindando herramientas encaminadas a proporcionar, según su naturaleza, una adecuada interpretación y visualización de los mismos, permitiendo llegar a conclusiones más acertadas para facilitar la toma de decisiones y ayudar a determinar el valor de los datos en el entorno donde están siendo analizados.

Por tal razón, se encuentra en la arquitectura propuesta por (Betancurt Obando & Abril Pérez, 2018) la necesidad de implementar un componente (prototipo de software) destinado a la exploración, el cual permita el análisis de las variables contenidas en un conjunto de datos (*data set*) donde se pueda obtener medidas de tendencia central para el análisis de comportamiento, correlación, desviaciones estándar y todas aquellas técnicas estadísticas relacionadas con esta fase propia del análisis de datos.

1.4. Objetivo

1.4.1. Objetivo General

Análisis e implementación de un componente de software a través de un prototipo orientado a la exploración de datos integrado a la arquitectura de visualización propuesta en el proyecto denominado “Diseño de una arquitectura por componentes para la visualización de datos utilizando dashboards”.

1.4.2. Objetivos Específicos

- Diseñar la arquitectura del prototipo a desarrollar.
- Implementar el prototipo para la exploración de datos.
- Diseñar y validar un caso base de pruebas empleando el prototipo, con el fin de registrar los resultados proporcionados por dicha técnica.

1.5. Metodología de la investigación

Fase 1 – Análisis de la arquitectura existente

Se realiza un análisis a partir de la arquitectura existente para la cual va a ser construido el componente de exploración de datos, se analizan los patrones arquitecturales y de diseño que posee, así como también sus colaboraciones e integraciones.

Fase 2-Estado del arte sobre las Técnicas de Exploración de Datos

Se realiza la investigación literaria en el contexto de la Exploración de Datos como una rama de la Ciencia de la Computación actual; su impacto y las técnicas empleadas.

Fase 3 – Estado del arte de la Arquitectura de Software aplicada a la Exploración de Datos

Se realiza la investigación literaria sobre el impacto y trabajos planteados que relacionen la Arquitectura de Software y la Exploración de Datos

Fase 4 – Lenguajes de programación empleados en el Análisis de Datos y en la construcción de aplicaciones orientadas a la Exploración de Datos

Se exploran los lenguajes de programación que permitan implementar tanto la fase de Exploración de Datos como de Visualización de los mismos. Se busca un lenguaje de programación preferiblemente que tenga un framework Web.

Fase 5 – Selección del lenguaje de programación y librerías pertinentes

Se realiza la selección del lenguaje de programación y se buscan librerías o complementos que permitan facilitar la tarea tanto de Exploración como de Visualización de Datos. Se realizan pequeños proyectos para probar estos dos artefactos.

Fase 6 – Selección de las técnicas de exploración a emplear en el proyecto y el tipo de dimensionalidad de datos a visualizar

Existen diversas técnicas de exploración de datos de modo que es necesario especificar cuál será la indicada para el proyecto. Así como definir los grados de dimensionalidad que el componente de software podrá manejar.

Fase 7 – Fase de pruebas inicial

Se realizan ejercicios de pruebas que incluyan el lenguaje de programación, librerías y las técnicas o técnica de exploración seleccionada.

Fase 8 – Selección de la Metodología para la construcción de una Arquitectura de Software

Existen diversas Metodologías para el desarrollo y construcción de arquitectura de software. Así como también estilos arquitectónicos, patrones arquitectónicos y patrones de diseño. Es necesario realizar una evaluación de estos para seleccionar el más apropiado para la plataforma en desarrollo.

Fase 9 – Diseño de la arquitectura de software empleado la metodología seleccionada

Se crea la arquitectura de software empleando los conceptos recopilados en la fase anterior

Fase 10 – Codificación de la arquitectura e implementación de los componentes

Empleando el lenguaje de programación, librerías y framework web seleccionado se inicia la codificación de la arquitectura de software aplicando las técnicas de exploración escogidas.

Fase 11 – Pruebas

Se realizan las pruebas que permitan comprobar la funcionalidad de la arquitectura de software y la efectivad de los componentes implementados.

2. Estado del arte

En esta sección se hace una introducción sobre los trabajos y proyectos actuales en el campo de la exploración de datos a nivel comercial y de investigación, los cuales sirven como referencia y muestran un panorama global sobre la utilidad de estas exploraciones para encontrar insights[9] dentro de un gran volumen de información heterogénea.

Dentro de los estudios desarrollados e implementados en diferentes áreas del conocimiento, aplicando análisis, visualización y exploración de grandes volúmenes de datos, se puede destacar la investigación realizada por parte de ingenieros de la Universidad de Carnegie Mellon (CMU), en Pittsburgh, EE.UU pertenecientes al centro de innovación CreateLab[10]. Estos desarrolladores implementaron una herramienta de exploración de datos de alta dimensión[11] llamada EVA (Explorable Visual Analytics) la cual, según ellos "pueda analizar grandes volúmenes de datos relacionados con la demografía o la educación y su relación con aspectos económicos como sector productivo o nivel de ingresos" (BBVAOPEN4U, 2016). Dicha herramienta está disponible para su uso en línea con un tipo de licenciamiento de código abierto; esta se desarrolló en el lenguaje de programación interpretado JavaScript, orientada a "visualizar y analizar grandes volúmenes de datos de alta dimensión que permite a los usuarios navegar intuitivamente por terabytes de datos con cientos de dimensiones".

También se destaca Microsoft, el cual provee procesos de exploración de datos en su servicio de computación en la nube Azure para analizar, entre otros, información proveniente de blogs, bases de datos SQLServer, máquinas virtuales y tablas de Hive[12] (Microsoft Azure, 2017). Con estas opciones los usuarios pueden almacenar grandes volúmenes de información y procesarlos de una forma sencilla usando diferentes lenguajes de programación, apoyados por ejemplos funcionales provistos por el sistema.

[9] Un insight es un hallazgo que tiene un valor para el usuario y proporciona utilidad en sus tareas.
[10] CreateLab es el laboratorio de la comunidad de robótica, educación y empoderamiento tecnológico de la CMU
[11] Datos que poseen un gran número de columnas en su estructura con una alta densidad de filas
[12] Hive es una infraestructura de almacenamiento de datos de proceso de datos estructurados en Hadoop

En la actualidad la exploración de datos se está convirtiendo en un punto importante tanto para las organizaciones como para las industrias, el IOT[13] está generando datos en tiempo real los cuales se convierten en un activo inmaterial, aportando nuevas oportunidades y valor a las mismas, pasando de la oferta de productos a una oferta integrada de "producto más servicio" como lo expresan los autores en el artículo "*Interactive Data Exploration as a Service for the smart factory"* (Ada Bagozi, Bianchini, Antonellis, Marini, & Ragazzi, 2017) donde proponen un marco de exploración de datos interactivo, el cual plantea una perspectiva orientada al servicio en la fábrica inteligente.

También se encuentran diferentes herramientas disponibles para explorar fuentes de datos tanto a nivel comercial como en el ámbito investigativo, se destacan por su poderío, facilidad de uso, rápido crecimiento y amplia trayectoria las siguientes herramientas:

- Tableau: es una potente herramienta para la exploración, visualización e inteligencia de negocios la cual ha tenido una evolución muy rápida. Posee una gran facilidad para su implementación, una curva de aprendizaje muy corta y una interfaz intuitiva en su usabilidad. Su interfaz permite la carga de datos provenientes de distintas fuentes y su análisis posterior permite explorar mediante asociación de variables y diferentes tipos de gráficas, la existencia de insights.
- RapidMiner: esta herramienta permite implementar las fases de un modelo de predicción, desde la preparación de los datos hasta la creación del modelo y su posterior validación empleando un entorno gráfico, en el cual se van encadenando los diferentes operadores a usar. Su uso principal está asociado a investigación, educación, capacitación, creación de prototipos y aplicaciones empresariales. Para el análisis de los datos se tiene a disposición más de 500 operadores, incluyendo entrada y salida, preprocesamiento de datos y visualización.
- SAS: es una suite orientada a la analítica avanzada desarrollada por el instituto SAS, en la cual también se pueden hacer análisis multivariados, inteligencia de negocios, administración de datos y análisis predictivos desde una amplia gama de fuentes de

[13] Internet de las cosas (Internet of things)

datos. Posee una poderosa interfaz gráfica la cual se puede complementar con el lenguaje de programación del mismo nombre. Dentro de un análisis de datos con SAS se deben definir una serie de pasos de datos (DATA steps) para cargar y manipular la información y una serie de pasos de procesamiento (PROC steps) para analizar los datos cargados.

- Orange: es un software para análisis de datos open source. Presenta una interfaz de programación visual para hacer análisis explorativo y visualización interactiva de datos, pudiendo ser usado también como una librería para Python. Es capaz de realizar de manera simple, histogramas, boxplots, heatmaps e incluso análisis avanzados como árboles de decisión o métodos de clustering. Su interfaz consiste en un lienzo interactivo (canvas) sobre el cual se insertan widgets creando un flujo de trabajo de análisis de datos. Los widgets proveen funcionalidades básicas como la lectura de los datos, creación de tablas, selección, exploración, comparación, algoritmos de aprendizaje, visualización de datos entre otras. El usuario puede explorar visualizaciones de forma interactiva o alimentar el subconjunto seleccionado en otros widgets.
- Scavis: es un software adecuado para centros de investigación y universidades, útil para trabajar en una amplia variedad de aplicaciones científicas y matemáticas incluyendo análisis y exploración de datos, computación numérica, estadística y análisis de big data. Es una plataforma que permite trabajar con muchos lenguajes de programación como Python/Jython, BeanShell, Groovy, Ruby, y también Java. Está escrita en Java y puede ser ejecutada en cualquier plataforma que tenga una instalación de Java. Puede ser usada para la creación de visualizaciones en 2D y 3D. Incluye más de 10.000 clases y métodos estadísticos, así como una extensa lista de librerías numéricas para cálculos analíticos, desarrollo de algoritmos y manipulación de datos.

3. Marco de Referencia

3.1. Marco Histórico

En los últimos años se han llevado a cabo, en diferentes partes del mundo, estudios sobre la exploración de datos, los cuales han guiado a los investigadores a implementar herramientas para la generación, adquisición, almacenamiento y análisis de datos, estimulados por aportar al desarrollo tecnológico y computacional. El crecimiento de los datos generado por los diferentes sistemas y actividades realizadas en la cotidianidad, conllevan a la creación de nuevos métodos y herramientas para el tratamiento de los datos, debido a la falta de capacidad requerida de los sistemas de gestión tradicionales para llevar a cabo dicho procesamiento.

Lo anterior se ve reflejado en el artículo “Big Data: una exploración de investigaciones, tecnologías y casos de aplicación” (Hernandez Leal, Duque Méndez, & Moreno Cadavid, 2013), en donde se realiza una exploración a las características, debilidades, fortalezas de las aplicaciones y modelos que incluyen Big Data. Así mismo, plantean una metodología empleada para la exploración mediante la aplicación de dos estrategias, una haciendo referencia al análisis cienciométrico y la otra a una categorización de documentos por medio de una herramienta web la cual brinda apoyo a los procesos de revisión literaria.

En la actualidad se habla de datos no estructurados y en cualquier área o entorno donde se interactúe se generan datos constantemente, las investigaciones también se han guiado hacia la revisión de estrategias para la visualización y exploración de textos. En el artículo “Cómo dibujamos textos. Revisión de propuestas de visualización y exploración textual” (Nualart Vilaplana, Pérez Montoro, & Whitelaw, 2014), se da una revisión a las herramientas encargadas de realizar dicha actividad, en donde se enumeran, clasifican y analizan los trabajos más relevantes desde el año 1995 hasta el año 2013, con el fin de identificar aquellas técnicas orientadas a detectar patrones, conductas y evidencias en la representación de la realidad, mostrando con claridad los

hechos ocultos en los datos. Un estudio similar a este fue realizado en “Infometría e Ingeniería del Conocimiento: Exploración de Datos y Análisis de la Información en vista del Descubrimiento de Conocimientos” (Polanco, 1997), en donde se destaca la ingeniería del conocimiento capaz de realizar procesamiento estadístico de la información científica y técnica, la cual tiene por objetivo la creación de indicadores, métodos e instrumentos de naturaleza lingüística, informática y matemática para analizar y representar dicha información.

Otra de las áreas en donde se genera una cantidad considerable de datos es en la industria, siendo de gran importancia la identificación de la información proporcionada por los datos y de esta manera tomar decisiones pertinentes en aspectos como innovación, ubicación, intereses de sus clientes, entre otros. Los autores Miguel Alejandro Flores Segovia y Amado Villarreal González del Tecnológico de Monterrey, realizaron una investigación sobre la distribución espacial de las industrias en el sector de la innovación en México (Flores Segovia & Villarreal González, 2014), en donde se lleva a cabo un análisis exploratorio de datos espaciales con el fin de identificar, cuantificar y localizar clústeres de empresas, en donde se muestra la existencia de zonas con una fuerte actividad de innovación distribuidas por todo el país. Esto permitirá la formulación de políticas orientadas a promover el desarrollo de las economías locales e identificar en dónde es posible desarrollar y financiar sistemas de innovación.

Por su parte, la arquitectura orientada a servicios SOA (**Service Oriented Architecture**) brinda la posibilidad de entender la composición y comunicación de los componentes existentes dentro del actual proyecto.

Como primer concepto para entender SOA debemos comprender el significado de una arquitectura dentro del ámbito de la informática y el desarrollo de software. En la versión 3.0 del SWEBOK se define el concepto de arquitectura de software como “el conjunto de estructuras necesarias para razonar sobre el sistema, que comprende elementos de software, relaciones entre ellas y propiedades de ambos” (IEEE Computer Society, 2014). Simon Brown la define de dos maneras: la arquitectura como un sustantivo donde se descompone un producto en una colección de componentes o módulos e interacciones y también es definida como verbo donde se debe comprender lo que se

necesita construir, crear una visión para construirlo y tomar las decisiones de diseño adecuadas. (Brown, 2018).

Después de exponer el concepto sobre una arquitectura en el ámbito del desarrollo de software, lo siguiente a tener en cuenta es el término "orientado al servicio". Este término se ha utilizado en diferentes contextos y con diferentes propósitos con el fin de encontrar la lógica necesaria para dar solución a un problema definido. Se debe realizar una descomposición en piezas más pequeñas relacionadas, cada una de las cuales aborda una parte específica del problema. Este enfoque trasciende las soluciones de tecnología, siendo una teoría establecida y genérica para abordar una variedad de problemas (Erl, 2016).

El concepto fundamental a la hora de resolver problemas de gran escala siempre ha sido la división del problema en unidades más pequeñas, logrando al darle solución a cada una de estas unidades, solucionar el problema inicial. SOA implementa este modelo creando un conjunto de unidades para conformar la solución de la lógica principal del negocio (Erl, 2016).

En SOA las unidades individuales de lógica existen de manera autónoma pero no aislada una de otra. Las unidades de lógica aún deben ajustarse a un conjunto de principios para evolucionar de manera independiente, al mismo tiempo que mantienen una cantidad suficiente de elementos comunes y de estandarización. Estas unidades de lógica se conocen como servicios (Erl, 2016), esto permite la escalabilidad y flexibilidad a la hora de extender o versionar los servicios, a su vez definir estándares para lograr una correcta comunicación y un correcto funcionamiento, definiendo unos estándares para los llamados y las respuestas de estos.

La arquitectura aplicada a los proyectos de software busca dar una visión global de cada uno de sus componentes, su comunicación e interacción. En el diseño del proyecto actual se ve reflejada la arquitectura orientada a servicios usada para describir el comportamiento del componente de exploración de datos integrado a la arquitectura ya existente.

A continuación, se muestran los conceptos básicos de la exploración de datos, así como las técnicas y tecnologías empleadas en el desarrollo del prototipo del proyecto actual. Primero se hará un recorrido por los diferentes conceptos estadísticos participes del proceso de análisis de los datos para posteriormente, afrontar conceptos de Big data y las tecnologías a implementar.

En la estadística se emplean múltiples técnicas y conceptos, siendo importante comprenderlos para tener un entendimiento claro de los pasos a desarrollar para lograr los objetivos propuestos. Lo primero a definir son los tipos de variables empleados en el análisis, siendo estos de naturaleza cualitativa o cuantitativa. Las **variables cualitativas** definen atributos de los elementos y no permiten una representación numérica. Sin embargo, algunas cualidades pueden ser codificadas de forma numérica para generar categorías de orden cualitativo. Entre las variables cualitativas están: el estrato socioeconómico, el estado civil, la profesión, el color de una flor, entre otras (Posada Hernandez, 2016). Respecto a las **variables cuantitativas,** estas permiten una escala numérica y las características de los elementos son observados cuantitativamente a través de una medida y una escala definida. Entre las variables cuantitativas se encuentran: el salario de los empleados, la talla de una persona, el peso, el número de hijos en una familia, el número de artículos vendidos en un almacén, entre otros (Posada Hernandez, 2016).

Una vez definidos los tipos de variables empleados es importante entrar a describir el **análisis de datos** (Data Analysis o DA), donde la información es en realidad el resultado del procesamiento, en el cual se tiene en cuenta un determinado conjunto de datos para extraer algunas conclusiones para ser utilizadas de varias formas. Este proceso de extracción de información sin procesar es precisamente un análisis de datos. Su propósito es extraer información la cual no es fácilmente deducible (insight), pero una vez entendida permite hacer pronósticos de posibles escenarios (Nelli, 2015).

El análisis de datos se puede describir como un proceso el cual consta de varios pasos, en donde los datos sin procesar se transforman y procesan para producir visualizaciones y pueden hacer predicciones gracias a un modelo matemático basado en los datos

recopilados. Está esquematizado como una cadena de procesos la cual consta de la siguiente secuencia de etapas:

- Definición del problema
- Extracción de datos
- Limpieza de datos
- Transformación de datos
- Exploración de datos
- Modelización predictiva
- Modelo de validación / prueba
- Visualización e interpretación de resultados.
- Despliegue de la solución.(Nelli, 2015).

Dentro del análisis de datos, según la cantidad de variables a examinar, este puede ser de tipo univariado o bivariado, describiendo el comportamiento de los datos y observando la relación existente entre las mismas. En el **análisis univariado** se exploran las variables (atributos) una por una, siendo estas de origen categórica o numérica. Existen diferentes técnicas estadísticas y de visualización para cada tipo de variable. Las variables numéricas se pueden transformar en contrapartes categóricas mediante un proceso denominado agrupación o discretización. También es posible transformar una variable categórica en su contraparte numérica mediante un proceso llamado codificación. Finalmente, el manejo adecuado de los valores perdidos es un tema importante en la extracción de los datos (Dr. Saed Sayad, 2018).

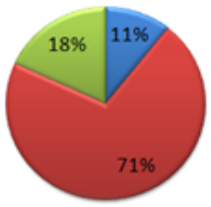

Figura 1. Ejemplo diagrama de pie para el análisis univariado (Dr. Saed Sayad, 2018).

Cuando el análisis se realiza sobre dos variables es llamado **análisis bivariado** donde se explora el concepto de relación entre ellas, si existe una asociación y la fuerza de esta asociación, o si hay diferencias entre dos variables y la importancia de estas diferencias (Dr. Saed Sayad, 2018).

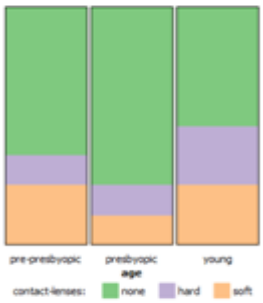

Figura 2. Ejemplo diagrama de barras apiladas para análisis bivariado (*Dr. Saed Sayad, 2018*).

Estos análisis pueden estar enfocados a resolver preguntas planteadas por los usuarios o propietarios de los datos (analítica descriptiva), orientados a la confirmación de una hipótesis (análisis confirmatorio) o inducir al usuario a descubrir conocimiento nuevo (análisis exploratorio). En la **analítica descriptiva** se emplean reportes o tableros de control (Dashboards) especializados para responder preguntas sobre eventos ocurridos. Por lo general, los reportes son de naturaleza estática y muestran datos históricos presentados en forma de tablas de datos o gráficos. Está basada en el trabajo sobre sistemas de almacenamiento donde se encuentren concentrados los datos. El sistema de almacenamiento puede ser tanto una base de datos estructurada como un sistema de archivos no estructurados, dependiendo de la cantidad y complejidad de los datos a manejar (Tan, Michael, & Kumar, 2014). Por otro lado, en el **análisis confirmatorio de datos** se emplea un enfoque deductivo en el cual se propone previamente la causa del fenómeno investigado. Se emplean estadísticos de resumen numéricos generados por medio de un modelo previamente definido, con el objetivo de confirmar o negar una hipótesis. Para este fin se emplean diferentes medidas como la media, la varianza, el coeficiente de correlación y la regresión, así como las pruebas de hipótesis (Tan et al., 2014). Finalmente, el **análisis exploratorio de datos** define un enfoque inductivo

relacionado estrechamente con la minería de datos (Data Mining). Este se caracteriza por usar herramientas o técnicas con mucha carga visual o gráfica, enfatizada en mostrar información vital sobre los datos examinados. Un buen análisis exploratorio de datos debe ser objetivo y extenso en la búsqueda, además de minucioso en lo respectivo a la indagación del comportamiento de las variables (Tan et al., 2014).

La **exploración de datos** por su parte consiste en un examen preliminar de los datos, lo cual es importante para comprender el tipo de información recopilada y su significado. En combinación con la información adquirida durante la definición del problema, esta categorización determinará el método de análisis de datos más adecuado para llegar a una definición de modelo. Esta fase, además de un estudio detallado de gráficos a través de los datos de visualización, puede consistir en una o más de las siguientes actividades:

- Resumir datos
- Agrupar datos
- Exploración de la relación entre los distintos atributos.
- Identificación de patrones y tendencias.
- Construcción de modelos de regresión.
- Construcción de modelos de clasificación (Nelli, 2015).

La información se puede entonces analizar de manera sistemática (análisis estadístico) o de manera gráfica (análisis visual) empleando los conceptos previos. En el **análisis visual** los datos son representados gráficamente para facilitar o mejorar la percepción visual del usuario, ayudando a identificar y señalar patrones, correlaciones y anomalías ocultas. Contrario a este, el **análisis estadístico** emplea métodos basados en fórmulas matemáticas como medio para analizar los datos, se utiliza para describir datasets de forma resumida mediante los siguientes tipos de análisis:

- El test A/B también conocidos como split o bucket testing, compara dos versiones de un elemento para determinar cuál versión es superior basado en métricas predefinidas. El elemento puede ser una variedad de cosas, por ejemplo, puede ser el contenido de un sitio web, una oferta para un producto o servicio, como ofertas en artículos electrónicos. La versión actual del elemento se conoce como versión de control, mientras la versión modificada se conoce como tratamiento. Ambas versiones

están sujetas simultáneamente a un experimento. Las observaciones son registradas para determinar cuál versión es más exitosa.

- La correlación es una técnica de análisis utilizada para determinar si dos variables están relacionadas entre sí. Ayuda a entender un dataset y a encontrar las relaciones para explicar un fenómeno. Cuando se estima la correlación de dos variables, se consideran ajustadas de acuerdo con una relación lineal. Esto quiere decir si una variable cambia, la otra variable también lo hace de manera constante y proporcional.
- La regresión investiga la relación existente entre una variable independiente y una variable dependiente dentro de un dataset. Como escenario de muestreo, la regresión podría ayudar a determinar el tipo de relación existente entre la temperatura (variable independiente) y el desempeño de los cultivos (variable dependiente).

El componente estadístico es el pilar fundamental del proceso de análisis sobre los datos, teniendo estos en los últimos años un crecimiento sin mesura dando origen al **Big data**, donde se estima una creación diaria de 2.5 quintillones de bytes de datos; de hecho, el 90 por ciento de los datos en el mundo actual se han creado solo en los últimos dos años. Estos datos provienen de una amplia variedad de fuentes como sensores utilizados para recopilar información sobre el clima, publicaciones en sitios de redes sociales, fotos y videos digitales, compra de registros de transacciones y señales de GPS de teléfonos celulares, entre otros. Big Data abarca cuatro dimensiones:

- Volumen: las empresas están inundadas de todo tipo de datos, acumulando fácilmente terabytes e incluso peta bytes de información.
- Velocidad: para procesos sensibles al tiempo, como la detección de fraudes, los datos grandes deben analizarse a medida que se introducen en la empresa para maximizar su valor empresarial.
- Variedad: se extiende más allá de los datos estructurados para incluir datos no estructurados de todas las variedades: texto, datos de sensores, audio, video, secuencias de clics, archivos de registro y más.
- Veracidad: al crecer la complejidad de la información, las organizaciones deben mejorar el nivel de confianza de los usuarios sobre la información, garantizar la coherencia en toda la organización y salvaguardar la información. Establecer la

confianza es esencial para impulsar mejores resultados comerciales (International Business Machines, 2013).

Todos estos conceptos están soportados sobre una base tecnológica desarrollada en Python, haciendo uso de diferentes frameworks, paquetes y librerías para realizar los procesos de análisis y visualización previa de los datos y sus resultados. Uno de los paquetes bases para la manipulación numérica descrito en el libro “Python for Data Analysis” (McKinney, 2018) es **NumPy (Numerical Python)** el cual proporciona las estructuras de datos, los algoritmos y las librerías necesarias en la mayoría de las aplicaciones científicas las cuales involucran datos numéricos en Python. Dentro de sus características encontramos:

- Emplea arreglos multidimensionales de forma rápida y eficiente.
- Posee funciones para realizar operaciones y cálculos matemáticos matriciales.
- Implementa herramientas para leer y escribir conjuntos de datos basados en matrices en disco.
- Provee operaciones de álgebra lineal, transformada de Fourier y generación de números aleatorios.

El alto rendimiento en cálculo numérico de NumPy la convirtió en el núcleo de la librería **Pandas,** el cual potencializó de forma rápida el uso y la implementación de la librería en distintos proyectos. También ayudó en la compatibilidad con otros módulos y disminuyó los problemas de integración (Nelli, 2015). Pandas es descrito en el libro “Python Data Analytics” (Nelli, 2015) como “*una librería de código abierto de Python para el análisis de datos altamente especializados. Actualmente es el punto de referencia de todos los profesionales que usan Python para estudiar y analizar conjuntos de datos con fines estadísticos de análisis y toma de decisiones*”. Wes McKinney fue su principal desarrollador en el año 2008, seguido por Sien Chang quien en el año 2012 se incorporó al equipo de desarrollo, creando una de las librerías más empleadas en el análisis de datos con Python, proporcionando de una forma simple herramientas para procesar, extraer y manipular datos.

Los conceptos definidos previamente se integran en una arquitectura basada en componentes, la cual es descrita en “Application Architecture Guide” (Microsoft

Corporation, 2009) donde es definida como "*una aproximación de ingeniería de software al diseño y desarrollo de un sistema, enfocándose en la descomposición del diseño en componentes funcionales o lógicos que expongan interfaces de comunicación bien definidas"*. Esta descomposición brinda un nivel de abstracción mayor a los modelos orientados a objeto sin la necesidad de especificar protocolos de comunicación ni la forma como se comparte el estado. Esta arquitectura posee las siguientes características (Microsoft Corporation, 2009):

- Es un estilo de diseño para aplicaciones compuestas de componentes individuales.
- Se enfatiza en la descomposición del sistema en componentes lógicos o funcionales con interfaces bien definidas.
- Define una aproximación de diseño basada en componentes discretos, comunicándose por medio de interfaces las cuales contienen métodos, eventos y propiedades.

En su libro (Microsoft Corporation, 2009), el autor define un **componente** como "*un objeto de software específicamente diseñado para cumplir con cierto propósito*" cumpliendo los siguientes principios:

- Reusable: son usualmente diseñados para ser utilizados en diferentes escenarios y aplicaciones, sin embargo, algunos componentes pueden ser diseñados para tareas específicas.
- Sin contexto específico: son diseñados para operar en diferentes ambientes y contextos. Información específica como el estado de los datos deben ser pasadas al componente en vez de incluirlos o permitirle acceder a ellos.
- Extensible: puede ser extendido desde un componente existente para crear un nuevo comportamiento.
- Encapsulado: exponen interfaces, las cuales permiten al programa usar su funcionalidad, sin revelar detalles internos del proceso o su estado.
- Independiente: están diseñados para tener una dependencia mínima de otros componentes, por lo tanto, pueden ser instalados en el ambiente adecuado sin afectar otros componentes o sistemas.

Estos principios proporcionan los siguientes beneficios:

- Facilidad de Instalación: cuando una nueva versión esté disponible, se podrá reemplazar la versión existente sin impacto en otros componentes o el sistema como un todo.
- Costos reducidos: el uso de componentes de terceros permite distribuir el costo del desarrollo y del mantenimiento.
- Facilidad de desarrollo: los componentes implementan una interfaz bien definida permitiendo el desarrollo sin impactar otras partes del sistema.
- Mitigación de complejidad técnica: mitigan la complejidad por medio del uso de contenedores de componentes y sus servicios.

Para tener un panorama global de la implementación se emplea el **modelo C4 para arquitecturas de software**, el cual surge ante la necesidad de diagramar modelos arquitectónicos de una manera amigable, entendible para el usuario y cada uno de los miembros del equipo de desarrollo. Este modelo arquitectónico (Brown, 2018) está compuesto por cuatro niveles:

Nivel 1. Diagrama de contexto del sistema: es un buen punto de partida para diagramar y documentar un sistema de software, permitiendo ver el panorama general. El detalle no es importante, el foco debe estar en los actores, roles, personajes, los sistemas de software, etc.

Nivel 2. Diagrama de contenedor: muestra una vista de alto nivel de la arquitectura del software y cómo se distribuyen las responsabilidades a través de ella. También muestra las principales opciones de tecnología y cómo los contenedores se comunican entre sí.

Nivel 3. Diagrama de componente: permite ver la lógica existente entre los componentes y sus relaciones, identificando los principales bloques estructurales con sus interacciones, responsabilidades y los detalles de la tecnología aplicada en la implementación.

Nivel 4. Diagrama de clases o código: es un diagrama opcional donde puede acercarse a cada componente para mostrar cómo se implementa el código; usando diagramas de clases UML, diagramas de relaciones de entidades o similares.

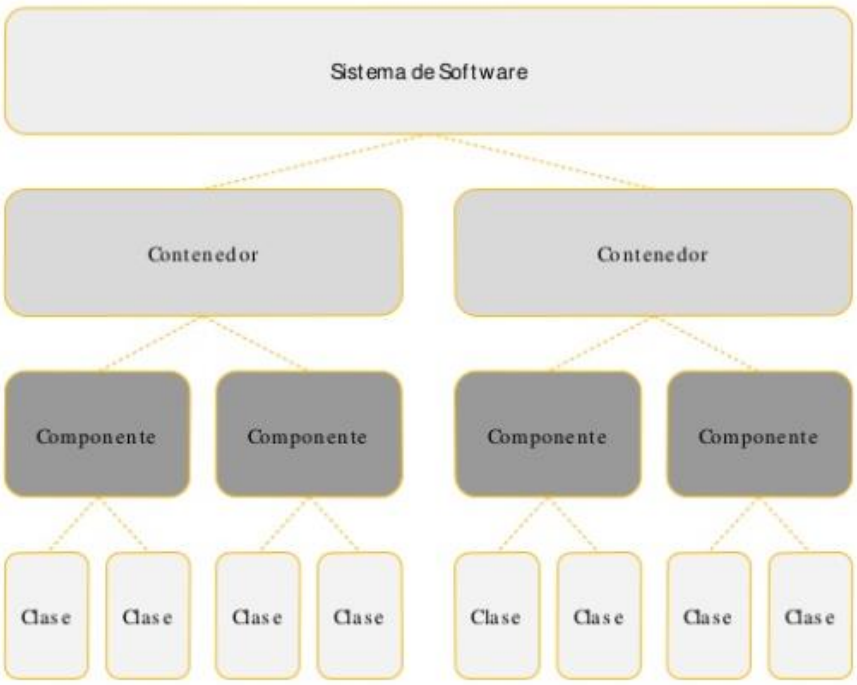

Figura 3. Descripción gráfica de los niveles del modelo C4 (Brown, 2018)

Paralelo al modelo C4, en la mayoría de los desarrollos de software se emplea **UML (Unified Modeling Language)** para representar cada uno de los diagramas, permitiendo tener una visión más detallada del sistema. En el manual de referencia del lenguaje de modelado unificado se define al UML así:

(Rumbaugh, Jacobson, & Booch, 2004)

El Lenguaje de Modelado Unificado (UML) es un lenguaje visual de propósito general utilizado para especificar, visualizar, construir y documentar los artefactos de un sistema de software. Captura las decisiones y la comprensión de los sistemas que deben construirse. Se utiliza para comprender, diseñar, explorar, configurar, mantener y

controlar información sobre dichos sistemas. Está diseñado para usarse con todos los métodos de desarrollo, etapas del ciclo de vida, dominios de aplicaciones y medios. Está diseñado para ser soportado por herramientas de modelado visual interactivo que tienen generadores de código y escritores de informes. La especificación UML no define un proceso estándar, pero pretende ser útil en un proceso de desarrollo iterativo. Está destinado a soportar la mayoría de los procesos de desarrollo orientados a objetos existentes.

Para la implementación del sistema se emplean los diagramas de componentes y de clases incluidos en UML, los cuales por su utilidad facilitan el análisis del sistema, ayudando a representar la arquitectura con un mayor nivel de detalle. **El diagrama de componentes** muestra las unidades de software a partir de las cuales se construye la aplicación, así como las dependencias entre los componentes para poder evaluar el impacto de un cambio propuesto (Rumbaugh, Jacobson, & Grady, 2004). A su vez **el diagrama de clases** representa los aspectos estáticos del sistema, definiendo las clases de objetos, sus asociaciones, atributos y operaciones (o métodos). Este diagrama permite indicar la estructura y el comportamiento de cada uno de los objetos del sistema, mostrando las relaciones con los demás (Rumbaugh, Jacobson, & Grady, 2004).

PARTE II

Arquitectura propuesta

4. Diagramas modulares del sistema

Partiendo de la arquitectura definida en la tesis “Diseño de una arquitectura por componentes para la visualización de datos utilizando dashboards” se plantea diseñar componentes adicionales y extender los desarrollados con anterioridad, con el fin de agregar la funcionalidad de la exploración de datos.

En la figura 4 se puede apreciar la definición de los diferentes usuarios del sistema y las acciones realizadas según su rol (Betancurt Obando & Abril Pérez, 2018). A continuación, se hará una breve descripción de cada uno de los objetos que lo componen.

4.1. Diagrama de contexto

Este diagrama muestra la interacción de los diferentes usuarios del sistema con el mismo.

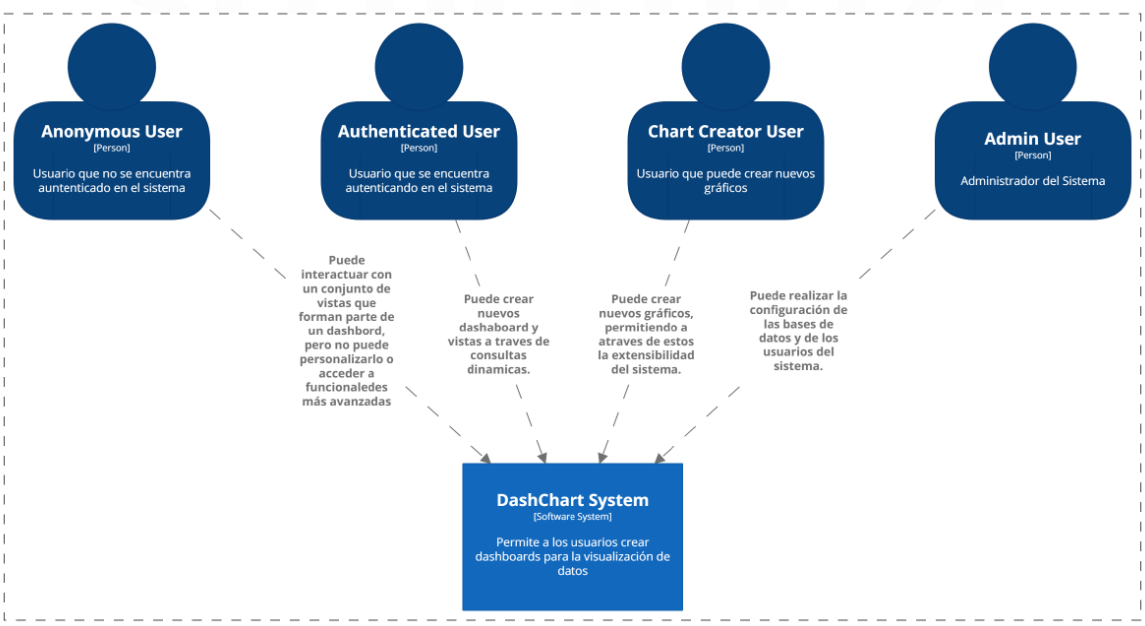

Figura 4. Diagrama de contexto del sistema de visualización y exploración.

Usuario Anónimo: es un usuario no autenticado, el cual puede interactuar de una forma limitada con el sistema mediante widgets y dashboards públicos.

Usuarios autenticados: estos usuarios pueden crear vistas, dashboards y explorar datos mediante una interfaz más personalizada.

Usuario creador de gráficos: es un usuario más avanzado, el cual podrá crear y personalizar gráficas del sistema.

Usuario administrador: es un usuario con la capacidad de configurar el ambiente, así como la administración de usuarios.

4.2. Diagrama de Contenedores

Este diagrama muestra las conexiones, en alto nivel, entre los diferentes componentes del sistema, así como la selección de tecnologías empleadas.

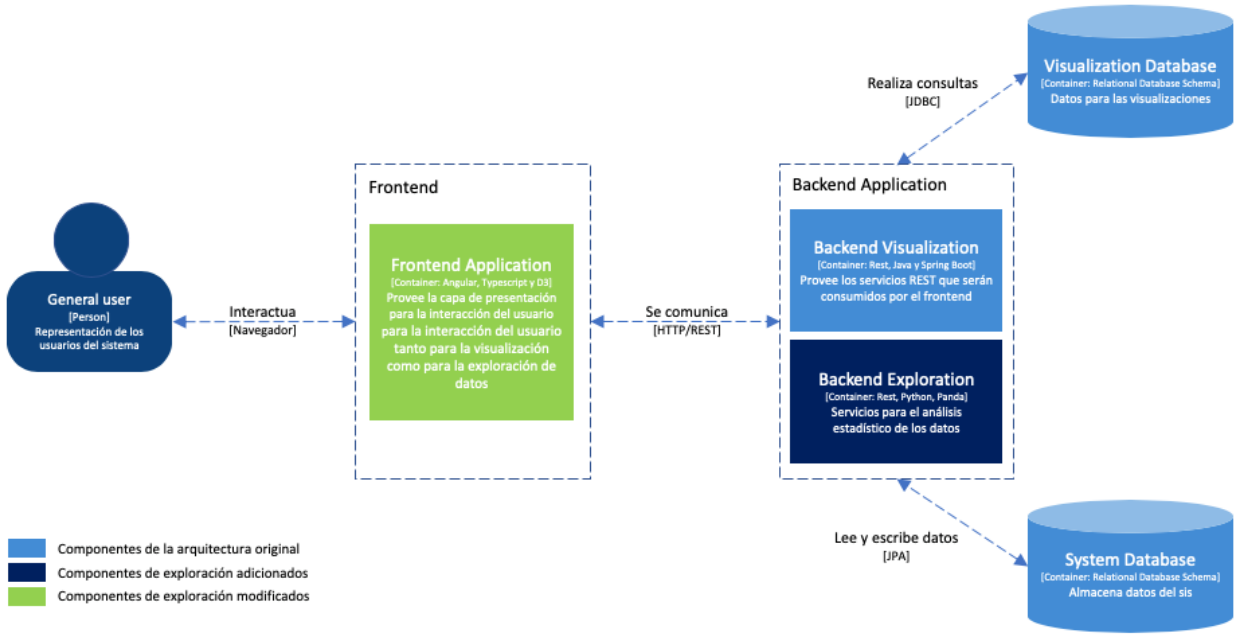

Figura 5. Diagrama de contenedores compuesto por frontend, backend y databases.

Usuario general: son los diferentes tipos de usuarios los cuales pueden realizar operaciones con el sistema.

Frontend: es el responsable de las actividades de visualización e intercambio de información entre el usuario y la capa de procesamiento. Se usan las diferentes plantillas provistas por AngularJS para adicionar la funcionalidad de exploración de datos, permitiendo al usuario autenticado analizar los datos almacenados en las diferentes fuentes de información configuradas en el sistema.

Backend Application, está compuesto de:

Backend Visualization: provee servicios de visualización y de conectividad con las bases de datos configuradas en el sistema.

Backend Exploration: provee servicios de análisis estadístico sobre datos cualitativos y cuantitativos almacenados en las bases de datos configuradas en el sistema. Este contenedor emplea las librerías Pandas[14], NumPy[15] y Matplotlib[16] de Python, por su robustez y madurez tanto en el análisis de datos complejos como simples.

Databases: son bases de datos donde se almacena la configuración del sistema y las diferentes fuentes de información a ser empleadas para la visualización y la exploración.

[14] Pandas es una librería de código abierto para el manejo de estructuras de datos con un alto rendimiento y fácil de usar, siendo una herramienta de análisis de datos, fuente http://pandas.pydata.org/
[15] NumPy es un paquete para computación científica y matemática, fuente http://www.numpy.org/
[16] Matplotlib es una librería de gráficos 2D que produce gráficas de alta calidad en una variedad de formatos y entornos interactivos, fuente https://matplotlib.org/

4.3. Diagrama de Componentes

Este diagrama muestra los bloques de un contenedor y la interacción existente entre estos.

4.3.1. Diagrama de Componentes del Frontend

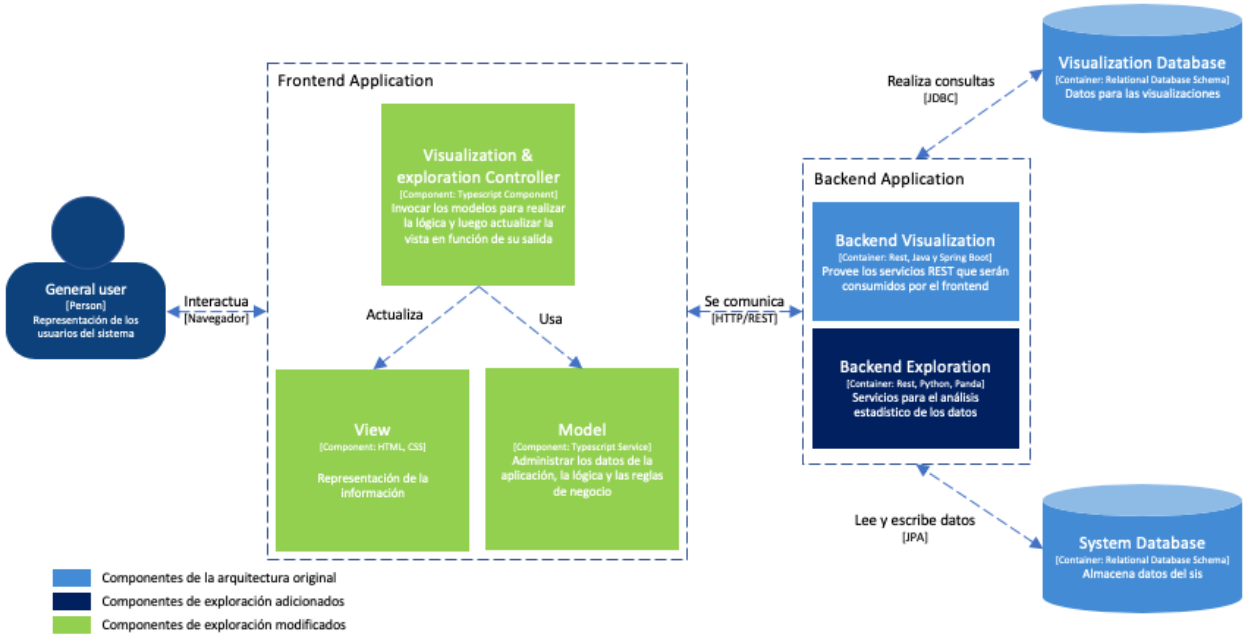

Figura 6. Diagrama de los componentes del frontend.

Frontend Application, está compuesto de:

Model: es el responsable de recuperar la información para ser mostrada al usuario, así como la lógica del negocio. Se adiciona la funcionalidad para recuperar el resultado del análisis de la exploración de datos, para luego ser formateada y presentada al usuario.

View: es la encargada de representar la información proveniente del controlador hacia el usuario. Se adicionan los formularios de exploración de datos, así como las interfaces para visualizar el resultado.

Visualization & exploration controller: es el encargado de recibir las solicitudes desde la vista hacia el modelo, actuando como enlace entre ellos y administrando el flujo de información.

4.3.2. Diagrama de Componentes Backend

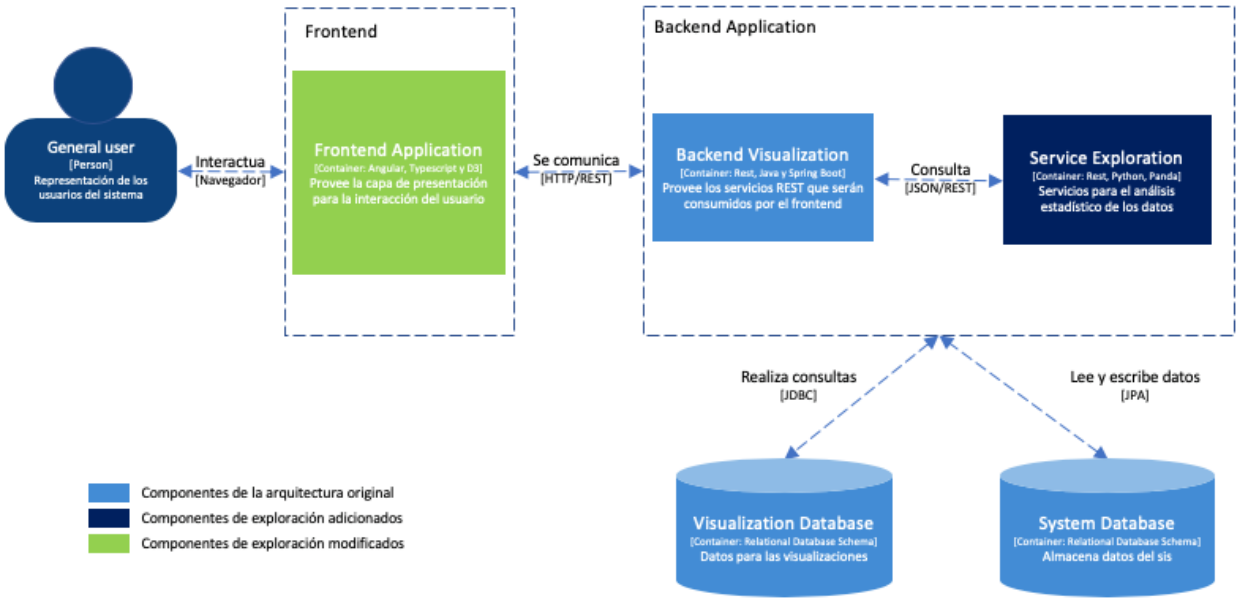

Figura 7. Diagrama de componentes del Backend simplificado.

Backend Application, el componente del backend tiene como objetivo realizar el procesamiento de las solicitudes o peticiones provenientes del frontend, está compuesto por:

Backend Visualization: este componente provee la lógica necesaria para atender las peticiones de visualización solicitadas desde el frontend. Usa servicios REST[17] para la integración de las diferentes tecnologías implementadas.

Service Exploration: es el encargado de resolver las peticiones sobre exploración de datos. Permite a través de sus servicios, realizar los cálculos estadísticos necesarios o dar formato a los datos, entregando una respuesta a las peticiones realizadas por el usuario y posteriormente mostrar las visualizaciones.

[17] REST (REpresentational State Transfer) es un estilo de arquitectura de software usado para describir interfaces entre sistemas para la manipulación de datos. (Li & Chou, 2011)(Li & Chou, 2011)(Li & Chou, 2011)

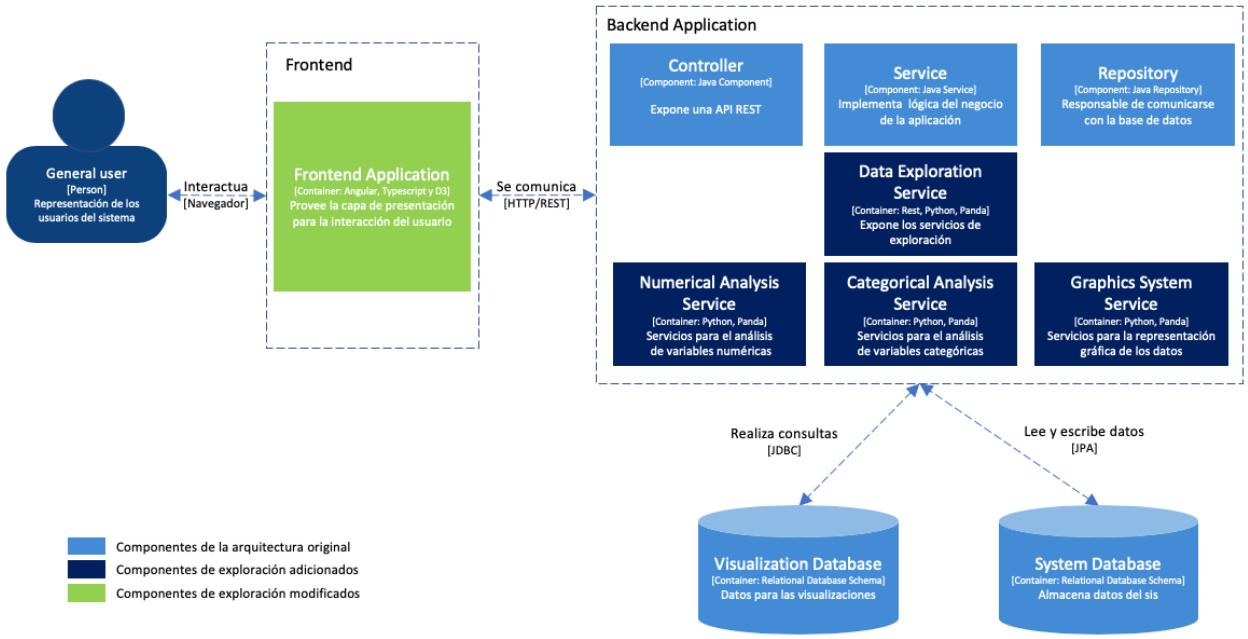

Figura 8. Diagrama de componentes del Backend ampliado.

En la figura 8 se muestra los detalles de los componentes del Backend de la aplicación, en los cuales encontramos:

Controller: recibe las peticiones en formato JSON[18] desde el frontend hacia las capas internas del backend, traduciéndolas a objetos Java para su posterior procesamiento.

Service (Java): procesa las solicitudes provenientes del controller, dirigiéndolas hacia la capa correspondiente según la lógica del negocio. Sirve de intermediario entre las capas controller y repository.

Repository: administra las conexiones a las bases de datos del sistema, teniendo la lógica para recuperar la información desde las diferentes fuentes configuradas por el usuario administrador, devolviéndola a la capa correspondiente.

Data Exploration Service (Python): recibe las solicitudes del usuario en formato JSON desde el frontend, donde se especifican los campos a analizar para luego ser enviados

[18] JSON (JavaScript Object Notation) es un formato ligero de intercambio de datos independiente del lenguaje, fuente https://www.json.org/json-es.html

a la capa correspondiente, devolviendo al usuario el resultado del análisis mediante la interfaz de exploración.

Numerical Analysis Service: aplica los diferentes análisis estadísticos a los datos de tipo numérico. Recibe la solicitud desde el frontend con los campos a analizar, pasando la solicitud al servicio de Python y posteriormente al servicio de análisis numérico; el resultado del análisis es retornado al frontend para ser presentado al usuario.

Categorical Analysis Service: es el responsable de realizar los diferentes análisis estadísticos a los datos de tipo cualitativo, realizando el mismo proceso definido en el componente “Numerical Analysis Service”.

Graphics System Service: sirve como complemento de apoyo al análisis estadístico, mostrando una representación gráfica de los datos analizados para una mejor comprensión y un análisis más acertado por parte del usuario.

4.4. Funcionamiento de REST

La base de la comunicación entre los componentes de la arquitectura desarrollada son los mensajes REST (Representational State Transfer), los cuales permiten interconectar la capa visual implementada en AngularJS con las de procesamiento y análisis en Java y Python de una manera sencilla apoyándose en mensajes de tipo JSON. Se hace pertinente describir el uso de REST para tener una mejor comprensión de la interacción entre las partes que conforman la arquitectura.

En el caso del desempeño, las mejoras son logradas debido a la simpleza y eficiencia de su diseño, características heredadas a los sistemas donde se implementa, lo cual permite tener una escalabilidad en la interacción de componentes en ambientes distribuidos. Además de contribuir a la interoperabilidad de sistemas distribuidos, su característica de interactuar con diferentes tecnologías y lenguajes de programación, hacen de REST un sistema portable, robusto y ágil, destaca el autor (Doglio, 2018).

Otro aspecto descrito por el autor es su capacidad de procesar las solicitudes sin la necesidad de almacenar información (stateless - sin estado), lo cual facilita el monitoreo

de las peticiones entrantes, agiliza la liberación de recursos, acelera la recuperación ante alguna falla del sistema y simplifica las tareas de desarrollo ya que no se requiere guardar información persistente. Este aspecto se apoya en la posibilidad de ofrecer un sistema de cache para brindar una respuesta más eficiente y disminuir la carga hacia las bases de datos u otros componentes de procesamiento de información, todo esto se soporta en un modelo de capas para simplificar tareas complejas. En la figura 9 se puede apreciar un esquema donde se ilustra la integración de diferentes tecnologías usando REST como intermediario.

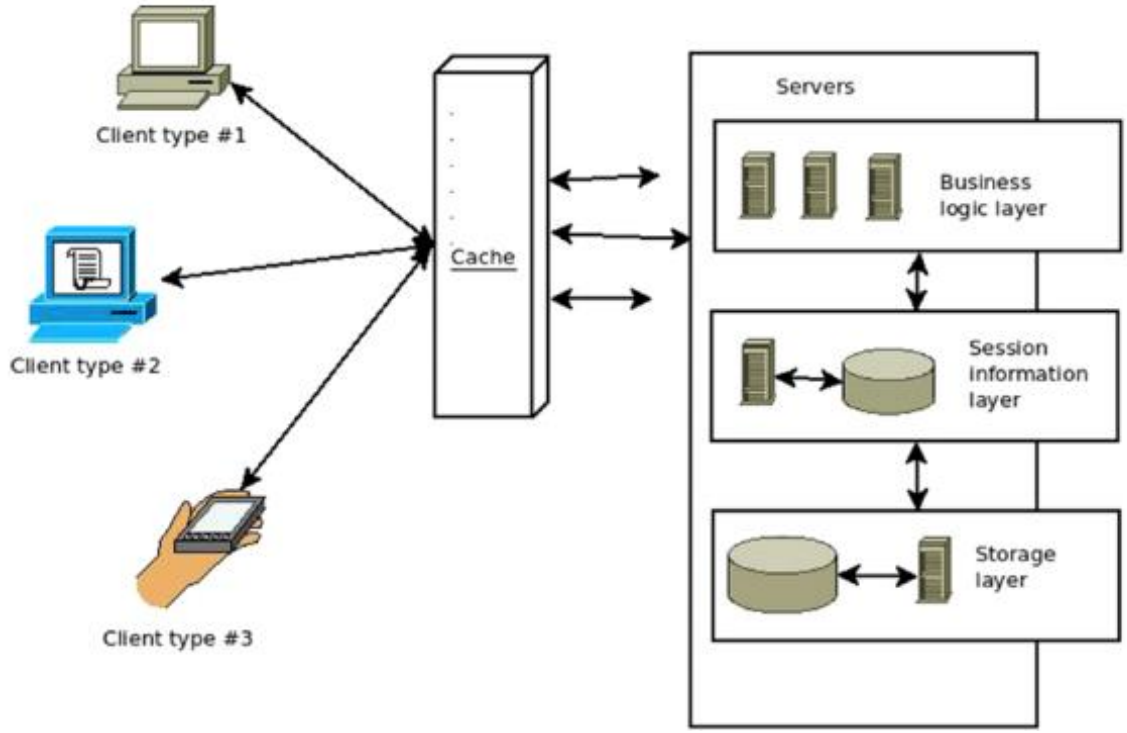

Figura 9. Ejemplo de una arquitectura multicapas (Doglio, 2018).

El objetivo principal de REST es brindar recursos a las aplicaciones solicitantes sin preocuparse por el tipo de dato a retornar, empleando principalmente el protocolo http[19]

[19] http (Hypertext Transfer Protocol) es un protocolo usado en Internet para tener acceso a contenido almacenado en un servidor

o https[20]. Este recurso puede ser representado en diversas formas o formatos como pueden ser datos binarios, objetos JSON, XML entre otros. La forma de llegar a ellos es por medio del uso de una URL[21] la cual los identifique de forma inequívoca, adicionando la metadata necesaria para permitirle al navegador o aplicación hacer una adecuada interpretación de los datos transmitidos por el servicio REST.

Las llamadas a estos recursos y el paso de parámetros se pueden realizar con diferentes métodos como lo son GET[22] y POST[23]. Para el desarrollo del proyecto se selecciona el primero de ellos debido a su simplicidad en la implementación, su aporte al correcto funcionamiento de la solución y como continuación a las metodologías empleadas en la tesis (Betancurt Obando & Abril Pérez, 2018).

Para la construcción de servicios web, Python proporciona diferentes frameworks los cuales permiten la configuración y su posterior publicación. Entre ellos encontramos a Django[24] como uno de los frameworks completos más usados por su robustez, facilidad y uso masivo por parte de desarrolladores tanto para sitios pequeños como para grandes aplicaciones; de los microframeworks, Flask[25] es la solución más liviana soportando de forma nativa el manejo de REST, siendo la opción seleccionada para la implementación del proyecto por su tamaño reducido, su bajo consumo de recursos y facilidad en el manejo de proyectos tipo REST.

4.5. Rutas del servicio REST

El desarrollo del sistema cuenta con los siguientes métodos, los cuales son invocados usando servicios REST desde angular:

- Análisis de una variable numérica (estadColum)
- Análisis de una variable categórica (estadColumCategorica)

[20] HTTPS (Hypertext Transfer Protocol Secure) permite de una manera segura tener acceso desde un navegador a el contenido que almacena un servidor
[21] URL (Uniform Resource Locator) representa la ubicación de un recurso dentro de un servicio o servidor
[22] GET es un método de HTTP usado para solicitar datos de una fuente específica
[23] POST es un método de HTTP usado para enviar datos hacia un servidor
[24] https://www.djangoproject.com
[25] http://flask.pocoo.org/

- Análisis de datos numéricos bivariados (estadNumericaBV)
- Análisis de datos categóricos bivariados (estadCategoricaBV)

En la figura 10 se puede ver la creación de la clase ExplorationService, la cual permite hacer el llamado a los servicios implementados en Python.

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { Observable } from 'rxjs/Observable';
import { TableListResponse, TableRequest, ColumnListResponse, ColumnsSQLResponse } from '../model/table.model';

@Injectable()
export class ExplorationService {

  constructor(private http:HttpClient) { }

  estadColum(sourceId: number, tableId: string, colum: string, min:number, max:number, groups:number): Observable<TableListResponse[]> {
    return  this.http.get<TableListResponse[]>('exploration/'+sourceId+'/'+tableId+'/'+colum+'/'+min+'/'+max+'/'+groups );
  }

  estadColumCategorica(sourceId: number, tableId: number, colum: string): Observable<TableListResponse[]> {
    return  this.http.get<TableListResponse[]>('exploration/categorica/' + sourceId + '/' + tableId + '/' + colum);
  }

  estadNumericaBV(source:number, tabla:string, column1:string, column2:string): Observable<TableListResponse[]>{
    return this.http.get<TableListResponse[]>('exploration/nbv/'+source+'/'+tabla+'/'+column1+'/'+column2);
  }

  estadCategoricaBV(source:number, tabla:string, column1:string, column2:string):Observable<TableListResponse[]> {
    return this.http.get<TableListResponse[]>('exploration/cbv/'+source+'/'+tabla+'/'+column1+'/'+column2);
  }

  imagen(method: string): Observable<ColumnListResponse[]> {
    return  this.http.get<ColumnListResponse[]>('exploration/imagen/' + method);
  }

}
```

Figura 10. Clase de exploración de datos donde se definen los métodos del sistema.

A continuación, se describen los métodos expuestos desde la capa de análisis de Python a la capa de presentación.

Análisis de una variable numérica: este método es el encargado de realizar los diferentes análisis para las variables numéricas, retornando el resultado en un formato JSON y posteriormente es representado de una forma adecuada al usuario.

Ruta: <ip>/exploration/<source>/<tabla>/<column>/<minV>/<maxV>/<grpH>

Tabla 1. *Parámetros del servicio análisis de una variable numérica*

Nombre	Tipo	Descripción
source	entero	Fuente de datos donde se encuentra la variable a analizar
tabla	cadena	Tabla dentro de la fuente donde está la variable solicitada
column	cadena	Nombre de la columna (variable) a analizar
minV	entero	Límite inferior del análisis, si el valor es cero se toma el valor mínimo encontrado en los datos
maxV	entero	Límite superior del análisis, si el valor es cero se toma el valor máximo encontrado en los datos
grpH	entero	Cantidad de grupos a dividir los datos (aplica para histograma)
Retorno	JSON	Resultado del análisis realizado

Ejemplo de aplicación usando Postman[26] para invocar el servicio y ver cómo se comporta con datos reales:

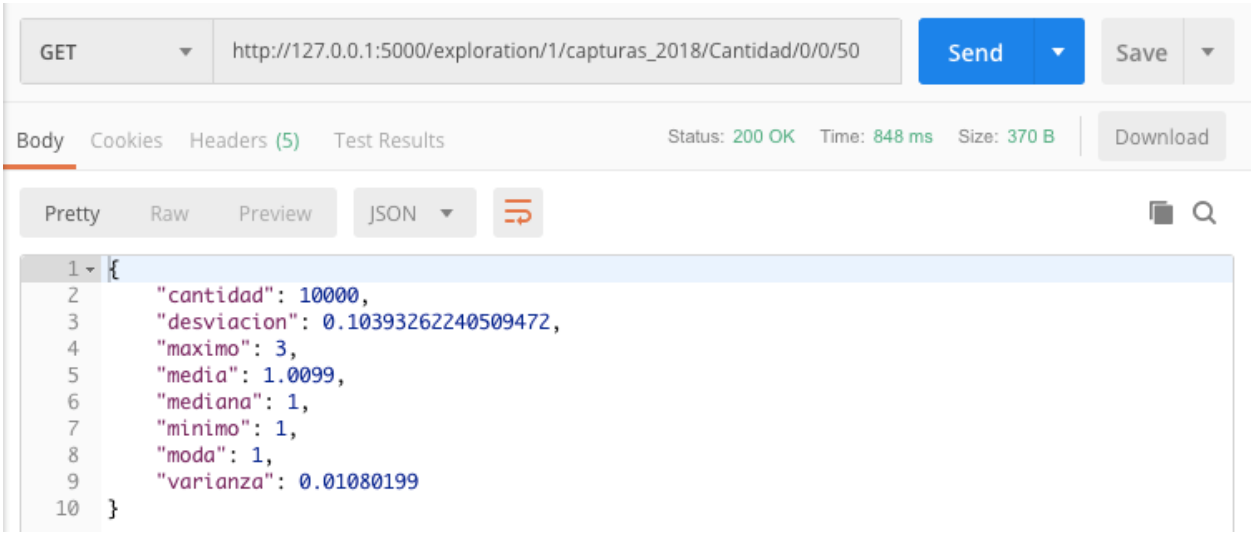

Figura 11. Solicitud y respuesta de análisis de una variable numérica

[26] https://www.getpostman.com/

Análisis de una variable categórica: este método analiza una variable categórica, retornando el resultado en un formato JSON y posteriormente es representado de una forma adecuada al usuario

Ruta: <ip>/exploration/categorica/<source>/<tabla>/<column>

Tabla 2. *Parámetros del servicio análisis de una variable categórica*

Nombre	Tipo	Descripción
source	entero	Fuente de datos donde se encuentra la variable a analizar
tabla	entero	Tabla dentro de la fuente donde está la variable solicitada
column	cadena	Nombre de la columna (variable) a analizar
Retorno	JSON	Los datos agrupados, así como su conteo y porcentaje

Ejemplo de aplicación usando Postman

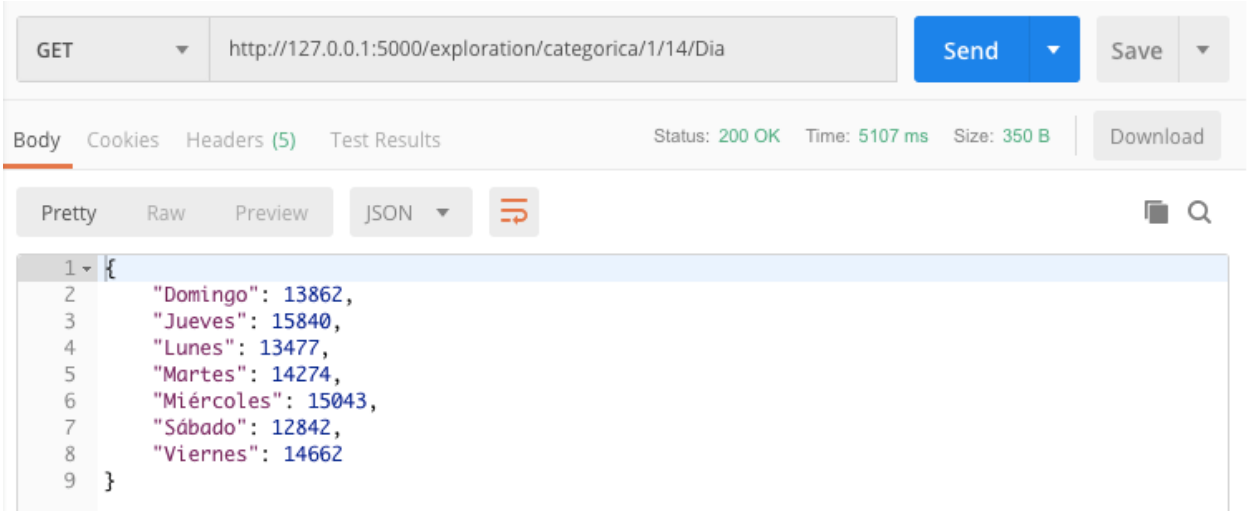

Figura 12. Solicitud y respuesta de análisis de una variable categórica

Análisis de datos numéricos bivariados: este método toma dos variables (columnas) de la misma fuente para hacer el análisis correspondiente y retornar un resultado unificado con la información de ambas.

Ruta: <ip>/exploration/nbv/<source>/<tabla>/<column1>/<column2>

Tabla 3. *Parámetros del servicio análisis de datos numéricos bivariados*

Nombre	Tipo	Descripción
source	entero	Fuente de datos donde se encuentra la variable a analizar
tabla	cadena	Tabla dentro de la fuente donde está la variable solicitada
column1	cadena	Nombre de la primera columna (variable) a analizar
column2	cadena	Nombre de la segunda columna (variable) a analizar
Retorno	JSON	Resultado del análisis realizado a las dos variables

Ejemplo de aplicación usando Postman

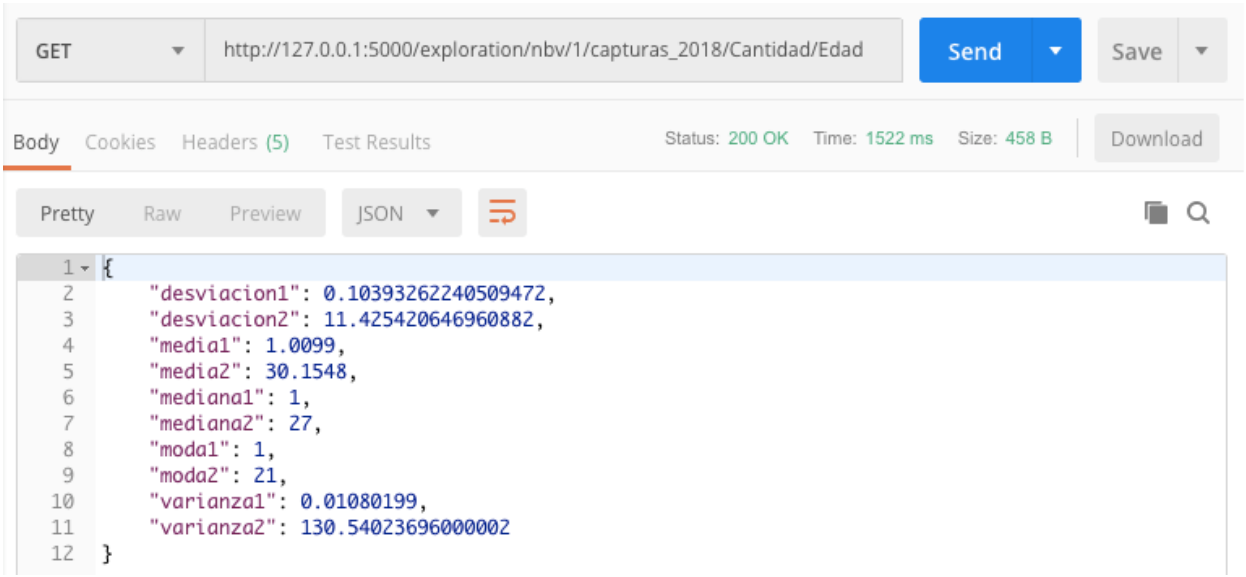

Figura 13. Solicitud y respuesta de análisis de datos numéricos bivariados

Análisis de datos categóricos bivariados: este método recibe dos variables categóricas y analiza la relación ejercida por una variable respecto a la otra, retornando el resultado de dicho análisis dentro de un objeto JSON para luego ser presentado al usuario en una forma adecuada.

Ruta: <ip>/exploration/cbv/<source>/<tabla>/<column1>/<column2>

Tabla 4. *Parámetros del servicio análisis de categóricos bivariados*

Nombre	Tipo	Descripción
source	entero	Fuente de datos donde se encuentra la variable a analizar
tabla	cadena	Tabla dentro de la fuente donde está la variable solicitada
column1	cadena	Nombre de la primera columna (variable) a analizar
column2	cadena	Nombre de la segunda columna (variable) a analizar
Retorno	JSON	Dos arreglos conteniendo la información agrupada de cada variable, así como su conteo y porcentajes respectivos

Ejemplo de aplicación usando Postman

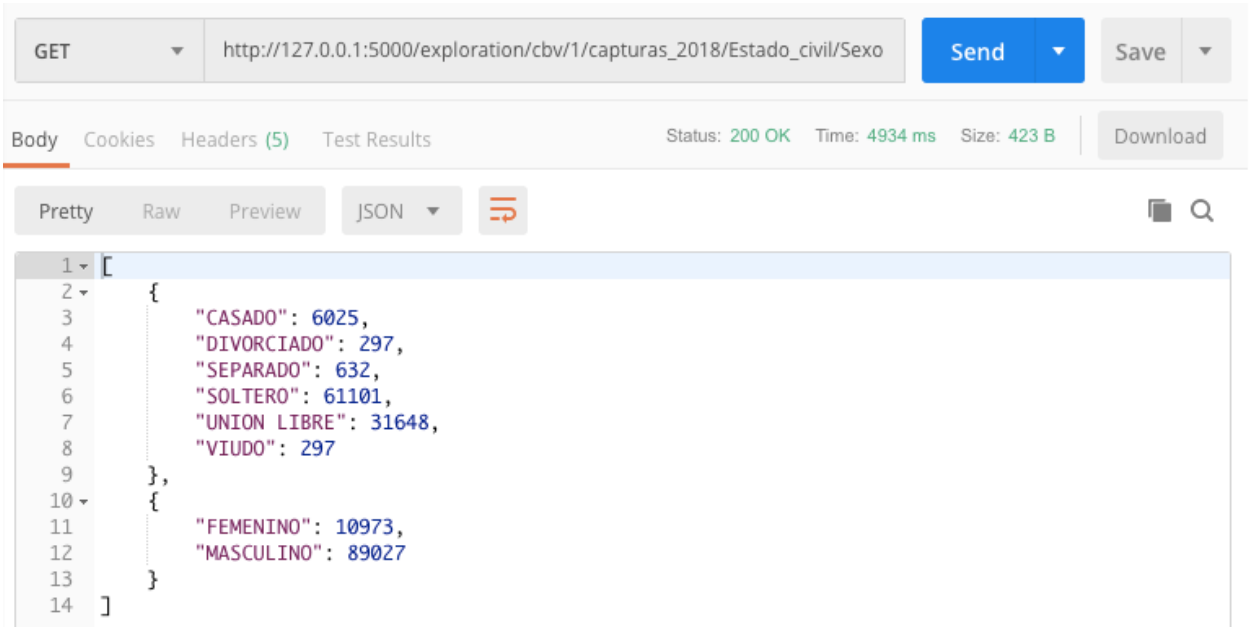

Figura 14. Solicitud y respuesta de análisis de datos categóricos bivariados

Representación gráfica de los datos según selección del usuario: este método es el encargado de generar la gráfica solicitada para el conjunto de datos recién analizados. Debe llamarse después de alguno de los métodos anteriores.

Ruta: <ip>/exploration/imagen/<metodo>

Tabla 5. *Parámetros del servicio representación gráfica de los datos*

Nombre	Tipo	Descripción
metodo	cadena	Representa el método gráfico a realizar según el tipo dado
	histo	histograma
	boxpl	diagrama de cajas o bigotes
	lines	diagrama de líneas
	pie	diagrama de torta
	bar	diagrama de barras
	scatter	diagrama de dispersión
	correl	diagrama de correlación
	stacked	diagrama de columnas apiladas
Retorno	cadena	la imagen codificada en base64[27] para ser embebida dentro de una etiqueta <img> de la siguiente manera: <img src="{{base64data}}">

[27] Base64 es un grupo de esquemas de codificación de binario a texto que representa los datos binarios mediante una cadena ASCII (tomado de https://developer.mozilla.org/es/docs/Web/API/WindowBase64/Base64_codificando_y_decodificando)

Ejemplo de aplicación usando Postman

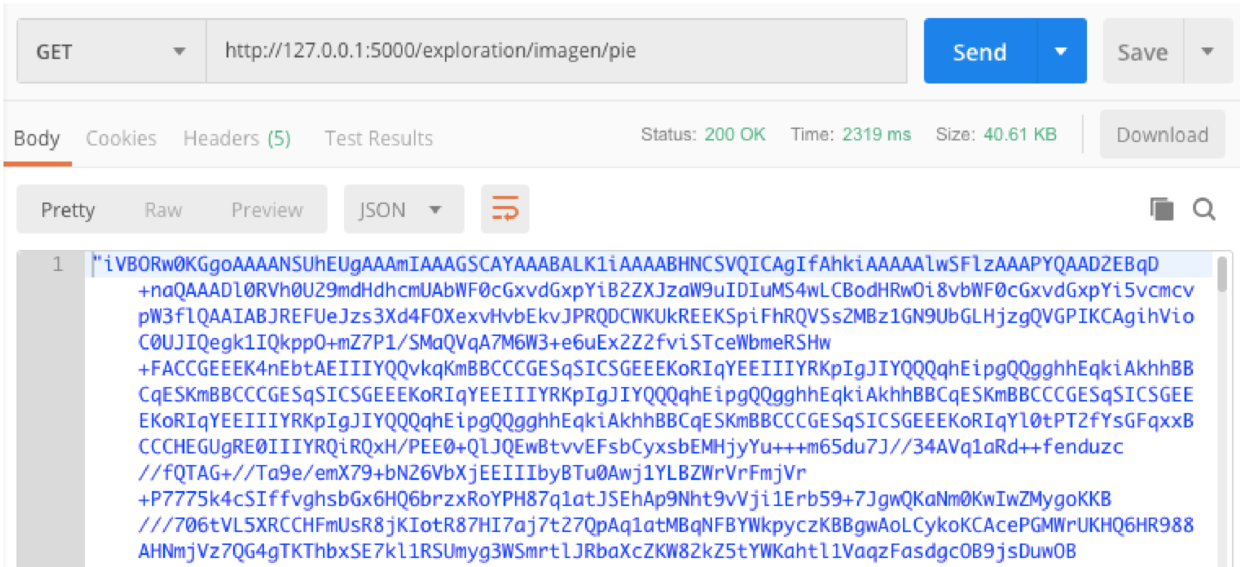

Figura 15. Solicitud y respuesta de representación gráfica

4.6. Diagramas UML

Para la construcción del actual proyecto se han seleccionado del diverso número de diagramas existentes en UML los siguientes:

- Diagrama de secuencia
- Diagrama de clases

Estos diagramas permiten describir un sistema a través de los elementos que lo componen, la comunicación entre ellos y su comportamiento ante las entradas y salidas. Se elige un diagrama estructural (Clases) y un diagrama comportamental (Secuencia). A continuación, se presentan los diagramas definidos para el actual proyecto:

Diagrama de secuencia:

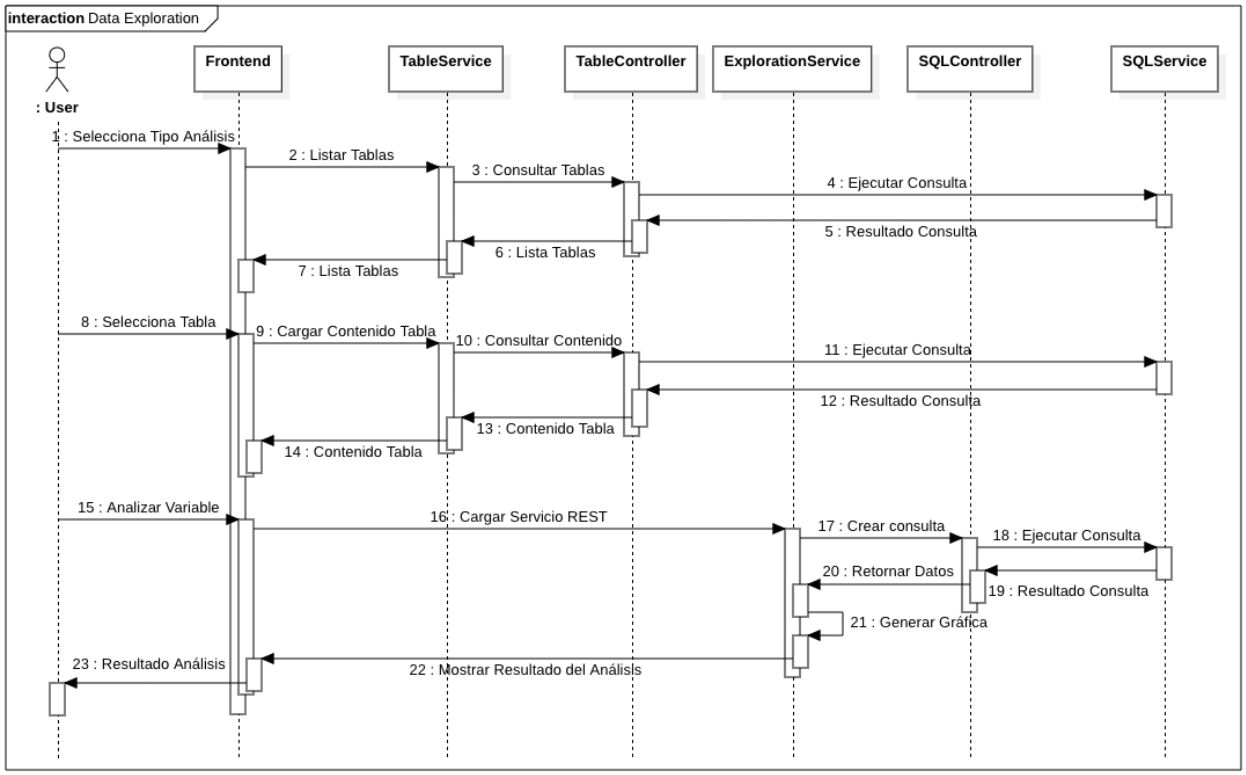

Figura 16*. Diagrama de secuencia para exploración de datos.*

El proceso de exploración de datos inicia por parte del usuario seleccionando en el Frontend el tipo de análisis a realizar, este puede ser univariado o bivariado. Una vez seleccionado el tipo de análisis, el sistema procede a cargar la lista de las tablas configuradas previamente, para esto el Frontend instancia la clase TableService la cual realiza un llamado REST al servicio expuesto en java por la clase TableController. El TableController recibe la solicitud de listar las tablas disponibles, creando una petición a la base de datos mediante el SQLService quien consulta en el sistema para retornar dicha lista. La lista de tablas es devuelta por el SQLService al TableController quien la retorna al TableService y por último entregarlo al usuario por medio del Frontend.

Cuando el usuario tiene la lista de las tablas en el Frontend, selecciona cuál desea analizar, generando de nuevo una solicitud a la instancia de TableService quien pasa la petición mediante REST al TableController para su procesamiento. La solicitud es

recibida y crea una petición a la base datos mediante el SQLService mostrando las diferentes columnas de la tabla. Para ello el SQLService debe consultar la fuente de datos donde está la tabla y generar la consulta adecuada para obtener el listado de las columnas junto con sus características como el tipo de dato que almacena. Este listado es retornado al TableController, luego al TableService hasta ser renderizado por el Frontend mostrando al usuario la información de las columnas.

Dependiendo del tipo de análisis seleccionado por el usuario, el Frontend crea una solicitud REST a la instancia de ExplorationService la cual determina el tipo de exploración a realizar (univariado, bivariado, numérico o categórico) y genera una solicitud a la base de datos por medio del SQLController. El SQLController pasa esta solicitud al SQLService con el propósito de recuperar los datos para hacer el análisis deseado, siendo este resultado retornado al SQLController y posteriormente al ExplorationService, quien se encarga de realizar las operaciones estadísticas acordes a la naturaleza de las variables y el análisis requerido. Una vez generados los resultados estadísticos se crea la gráfica respectiva la cual representa los datos analizados y tanto el resultado del análisis como la gráfica son retornados al Frontend para ser presentados al usuario para su posterior interpretación.

Diagrama de clases:

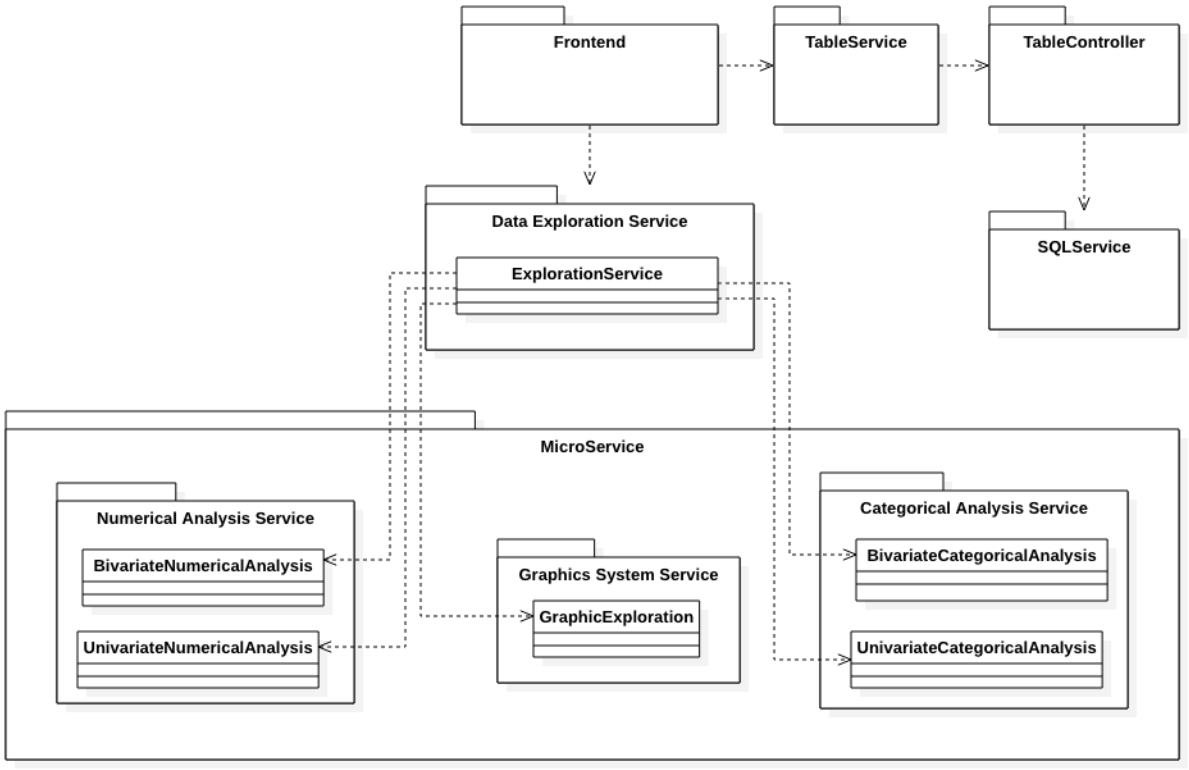

Figura 17. Diagrama de clases arquitectura proyecto de grado.

En el diagrama de clases se puede observar la relación de los componentes y las clases contenidas en ellos. Cada componente provee funcionalidades específicas a la arquitectura mediante servicios REST los cuales pueden ser consumidos desde cualquier componente en la arquitectura o pueden recibir solicitudes de módulos externos a ella. En el diagrama de clases se adicionan algunos de los componentes correspondientes a la arquitectura definida por Paola Betancurt y Abril Pérez en su proyecto de grado (Betancurt Obando & Abril Pérez, 2018).

Dentro de la arquitectura de Betancurt y Pérez se define el componente SQLService, encargado de ejecutar las consultas en las diferentes bases de datos, obtener la información de las tablas requeridas y devolver la información al componente correspondiente. También definen los componentes TableController y TableServices encargados de mapear las solicitudes de los usuarios hacia el SQLService y exponer los

servicios respectivamente. Por su lado el Frontend contiene los componentes visuales requeridos para tener acceso a las funcionalidades de la arquitectura, su estructura interna está definida por el framework angularJS y desde estos componentes visuales se realizan todas las peticiones a los servicios requeridos tanto para la visualización de los dashboards como de la visualización y los análisis en la exploración de datos.

El componente Data Exploration Service brinda a la arquitectura la posibilidad de ser escalable, encargándose de gestionar las peticiones REST realizadas por los demás componentes para el análisis exploratorio de datos, recibe las solicitudes y las dirige al componente indicado mediante microservicios, enviando los parámetros necesarios como son las fuentes de información a consultar, los tablas a cargar y las columnas a analizar.

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { Observable } from 'rxjs/Observable';
import { TableListResponse, TableRequest, ColumnListResponse, ColumnsSQLResponse } from '../model/table.model';

@Injectable()
export class ExplorationService {

  constructor(private http:HttpClient) { }

  estadColum(sourceId: number, tableId: string, colum: string, min:number, max:number, groups:number): Observable<TableListResponse[]> {
    return  this.http.get<TableListResponse[]>('exploration/'+sourceId+'/'+tableId+'/'+colum+'/'+min+'/'+max+'/'+groups );
  }

  estadColumCategorica(sourceId:number, tableId:string, colum:string): Observable<TableListResponse[]> {
    return  this.http.get<TableListResponse[]>('exploration/categorical/' + sourceId + '/' + tableId + '/' + colum);
  }

  estadNumericaBV(source:number, table:string, column1:string, column2:string): Observable<TableListResponse[]>{
    return this.http.get<TableListResponse[]>('exploration/nbv/'+source+'/'+table+'/'+column1+'/'+column2);
  }

  estadCategoricaBV(source:number, table:string, column1:string, column2:string):Observable<TableListResponse[]> {
    return this.http.get<TableListResponse[]>('exploration/cbv/'+source+'/'+table+'/'+column1+'/'+column2);
  }

  picture(type: string): Observable<ColumnListResponse[]> {
    return  this.http.get<ColumnListResponse[]>('exploration/picture/' + type);
  }

}
```

Figura 18. Definición de ExplorationService

Por otro lado, las clases UnivariateNumericalAnalysis y BivariateNumericalAnalysis se encargan de realizar el análisis de las variables de naturaleza numérica. Ellas reciben la información proveniente del ExplorationService, recupera la información de la fuente de datos seleccionada empleando solicitudes REST a la clase SQLService mediante la ruta

/api/sql, para posteriormente almacenar esta información y retornar el resultado del análisis realizado en un tipo de dato JSON. Los análisis por realizar sobre los datos son su promedio, la media, la desviación estándar, la varianza, los valores máximos y mínimos, la cantidad de datos analizados y la moda.

```
class UnivariateNumericalAnalysis(Resource):
    def get(self, dataSource, table, column, minValue, maxValue, histogramGroups):
        global dataTable1, histogram
        dataTable1 = []
        histogram  = histogramGroups
        if minValue != maxValue:
            sql = 'sql=select '+column+' from '+table+' where '+column+' between '+str(minValue)+' and '+str(maxValue)
        else:
            sql = 'sql=select '+column+' from '+table
        response = requests.get( 'http://127.0.0.1:9090/api/sql/'+str(dataSource)+'/query?' + sql )
        jData = json.loads(response.content)
        for i in jData:
            dataTable1.append( i[column])
        moda = stats.mode(dataTable1)
        datos = {
            "media": np.mean(dataTable1),
            "mediana": np.median(dataTable1),
            "desviacion": np.std(dataTable1),
            "varianza": np.var(dataTable1),
            "minimo": np.min(dataTable1),
            "maximo": np.max(dataTable1),
            "cantidad": np.size(dataTable1),
            "moda": moda[0][0]
        }
        return datos, 200
```

Figura 19. Definición de UnivariateNumericalAnalysis

```python
class BivariateNumericalAnalysis(Resource):
    def get(self, dataSource, table, column1, column2):
        global dataTable1, dataTable2, variableName1, variableName2
        variableName1 = column1
        variableName2 = column2
        dataTable1  = []
        dataTable2 = []
        # dataTable1
        response = requests.get('http://127.0.0.1:9090/api/sql/'+str(dataSource)+'/query?sql=select '+column1+' from '+table)
        jData = json.loads(response.content)
        for i in jData:
            dataTable1.append( i[column1])
        # dataTable2
        response = requests.get('http://127.0.0.1:9090/api/sql/'+str(dataSource)+'/query?sql=select '+column2+' from '+table)
        jData = json.loads(response.content)
        for i in jData:
            dataTable2.append( i[column2])
        moda1 = stats.mode(dataTable1)
        moda2 = stats.mode(dataTable2)
        datos = {
            "media1": np.mean(dataTable1),
            "mediana1": np.median(dataTable1),
            "desviacion1": np.std(dataTable1),
            "varianza1": np.var(dataTable1),
            "moda1": moda1[0][0],
            "media2": np.mean(dataTable2),
            "mediana2": np.median(dataTable2),
            "desviacion2": np.std(dataTable2),
            "varianza2": np.var(dataTable2),
            "moda2": moda2[0][0]
        }
        return datos, 200
```

Figura 20. Definición de BivariateNumericalAnalysis

Cuando la exploración es sobre una variable categórica, las clases UnivariateCategoricalAnalysis y BivariateCategoricalAnalysis se encargan de atender la solicitud de exploración. Igual a las clases anteriores, estas cargan la información mediante una llamada REST a la clase SQLService para luego realizar el análisis de los datos empleando la función counter, la cual resume y realiza un conteo de los diferentes valores (categorías) contenidos en la variable analizada retornando un arreglo con este resumen para luego ser mostrada en el Frontend.

```
class UnivariateCategoricalAnalysis (Resource):
    def get(self, dataSource, table, column):
        global dataTable1, categories1, categorieValue1
        dataTable1 = []
        categories1 = []
        categorieValue1   = []
        response = requests.get('http://127.0.0.1:9090/api/sql/'+str(dataSource)+'/query?sql=select '+column+' from '+table)
        jData = json.loads(response.content)
        for i in jData:
            dataTable1.append( i[column])
        datos = Counter(dataTable1)
        for key, val in datos.items():
            categories1.append(key)
            categorieValue1.append(val)
        return datos, 200
```

Figura 21. Definición de UnivariateCategoricalAnalysis

```
class BivariateCategoricalAnalysis (Resource):
    def get(self, dataSource, table, column1, column2):
        global dataTable1, dataTable2, categories1, categories2, categorieValue1, categorieValue2, dataP
        dataTable1   = []
        dataTable2  = []
        categories1  = []
        categories2 = []
        categorieValue1 = []
        categorieValue2= []
        response = requests.get('http://127.0.0.1:9090/api/sql/'+str(dataSource)+'/query?sql=select '+column1+','+column2+' from '+table)
        jData = json.loads(response.content)
        dataP = pandas.DataFrame(jData)
        for i in dataP[column1]:
            dataTable1.append( i )
        datos = Counter(dataTable1)
        for key, val in datos.items():
            categories1.append(key)
            categorieValue1.append(val)
        for i in dataP[column2]:
            dataTable2.append( i )
        datos2 = Counter(dataTable2)
        for key, val in datos2.items():
            categories2.append(key)
            categorieValue2.append(val)
        return [datos, datos2], 200
```

Figura 22. Definición de BivariateCategoricalAnalysis

Por medio de la clase GraphicExploration se genera una representación gráfica de los datos previamente cargados y analizados por las anteriores clases según sea el tipo de variable y la cantidad de columnas para analizar. Por ejemplo, para una gráfica donde se tengan dos variables numéricas se utilizan las variables dataTable1 y dataTable2, pero si las variables a analizar son categóricas, se toma como fuente de información los datos registrado en las variables categories1 para mostrar las categorías encontradas y categorieValue1 para asociar su valor.

```
class GraphicExploration(Resource):
    def get(self, method):
        global dataTable1, dataTable2, categorieValue1, categories1, variableName1, variableName2, histogram, dataP
        plt.figure()
        if method=="histo" or method=="":
            pandas.DataFrame(dataTable1).plot(kind='hist', bins=histogram, grid=True, rwidth=0.8, legend=False, fontsize=10)
            plt.title('Histogram')
        elif method=="boxpl":
            pandas.DataFrame(dataTable1).plot(kind='box')
            plt.title('BoxPlot')
        elif method == "lines":
            pandas.DataFrame(dataTable1).plot(legend=False, grid=True)
            plt.title("Linear")
        elif method == "pie":
            pandas.DataFrame(categorieValue1).plot(kind='pie', subplots=True, labels=categories1, fontsize=6, legend=True, autopct='%.1f%%', startangle=90)
            plt.legend(bbox_to_anchor=(0.9,0.5), loc="center right", bbox_transform=plt.gcf().transFigure, prop={'size': 8})
            plt.subplots_adjust(left=0.0, bottom=0.1, right=0.6)
            plt.axis('equal')
        elif method == "bar":
            datos = pandas.DataFrame({'lab':categories1, 'val':categorieValue1}).plot.bar(x='lab', y='val', grid=True, legend=False, fontsize=8)
            plt.title("BarChar")
        elif method == 'scatter':
            pandas.DataFrame({variableName1:dataTable1,variableName2:dataTable2}).plot(kind='scatter',x=variableName1, y=variableName2)
            plt.title('Scatter')
        elif method == "correl":
            datos = pandas.DataFrame({variableName1:dataTable1, variableName2:dataTable2})
            fig = plt.figure()
            ax = fig.add_subplot(111)
            print datos.corr()
            cax = ax.matshow(datos.corr(), vmin=-1, vmax=1)
            fig.colorbar(cax)
            ticks = np.arange(0,2,1)
            ax.set_xticks(ticks)
            ax.set_yticks(ticks)
            ax.set_xticklabels([variableName1, variableName2])
            ax.set_yticklabels([variableName1, variableName2])
            plt.title('Correlacion')
        elif method == 'stacked':
            dataP.groupby([dataP.columns[0],dataP.columns[1]])[dataP.columns[0]].count().unstack(dataP.columns[1]).fillna(0).plot(kind='bar',stacked=True,fontsize=8)
            plt.legend(prop={'size': 8})
            plt.title('Stacked')
        figfile = BytesIO()
        plt.savefig(figfile, format='png', bbox_inches="tight")
        figfile.seek(0)  # rewind to beginning of file
        figdata_png = base64.b64encode(figfile.getvalue())
        return figdata_png, 200
```

Figura 23. Definición de GraphicExploration

5. Análisis de resultados

A continuación, se presentan los resultados obtenidos al utilizar un caso básico de prueba en el prototipo implementado, donde el usuario puede acceder a diferentes funcionalidades, permitiéndole interactuar con las opciones que ofrece el sistema para la exploración de los datos.

5.1. Caso básico de prueba:

Se seleccionaron las siguientes bases de datos disponibles en la página de “Datos Abiertos Colombia” (https://www.datos.gov.co) por su contenido y variabilidad.

- Resultados Saber Pro Competencias Genéricas 2018-1: estos son los resultados generales de las pruebas Saber Pro realizadas en el primer semestre del año 2018 reportadas por el ICFES. Su estructura se describe en el anexo 1.

  https://www.datos.gov.co/Educaci-n/Resultados-Saber-Pro-Competencias-Genericas-2018-1/5j9v-vwmd

- Capturas 2018: son las capturas reportadas por la Dijin y la policía nacional entre el 1 de enero y el 30 de septiembre del año 2018. Su estructura se define en el anexo 2.

  https://www.datos.gov.co/Seguridad-y-Defensa/Capturas-2018/wiw2-jmf9/data

Sobre esta información cargada en el sistema se inicia el proceso de exploración de algunas variables encontradas en las tablas, las cuales podrían contener información valiosa y no obvia, posteriormente el sistema muestra un análisis según el tipo de variable seleccionada y presenta al usuario una serie de opciones, dónde puede observar diferentes formas de interpretar los datos de manera gráfica, así como el resultado obtenido.

5.2. Ejecución del prototipo:

5.2.1. Análisis univariado

El primer tipo de análisis realizado corresponde a una variable numérica, donde seleccionamos de la tabla “capturas” la variable “edad” mostrando los siguientes resultados:

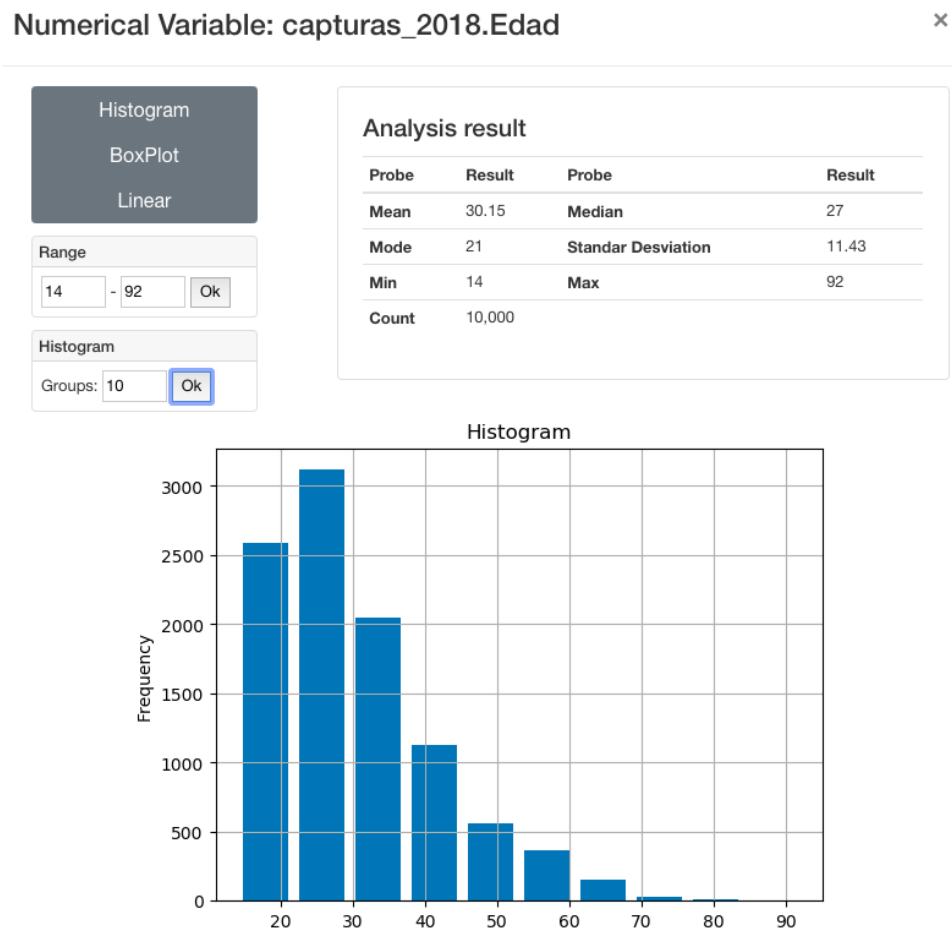

Figura 24. Resultados de la variable "edad" de la tabla "capturas_2018" mediante un histograma

Como se observa en la figura 19, el sistema luego de cargar y analizar la información presenta una tabla con los resultados de las principales medidas obtenidas: la media, la

mediana, la moda, la desviación estándar, el valor mínimo y el valor máximo, estos dos últimos valores (rango), así como la cantidad de grupos a crear en el histograma, se pueden modificar para ajustar el resultado según el criterio del usuario quien realiza la exploración.

En el gráfico de cajas mostrado en la figura 20, se puede observar otro tipo de información relevante para el análisis de esta variable. En él se identifican como datos atípicos todos aquellos valores superiores a 60, siendo este dato importante para realizar una debida interpretación de los datos y de ser necesario iniciar un proceso de limpieza de estos.

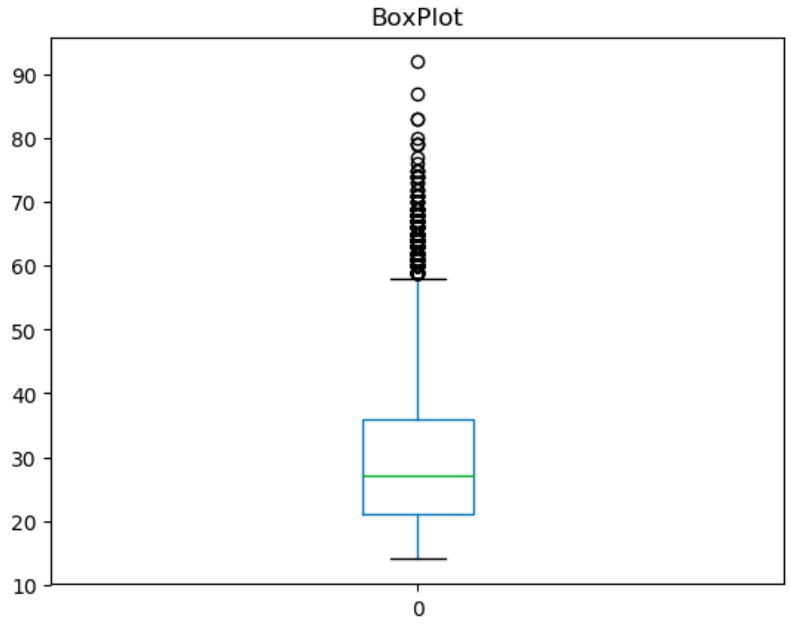

Figura 25. Diagrama de cajas de la variable "edad" de la tabla "capturas_2018".

En el caso de variables categóricas, al analizarlas se despliega la ventana mostrada en la figura 21, donde el usuario puede experimentar con los diferentes gráficos generados por el sistema:

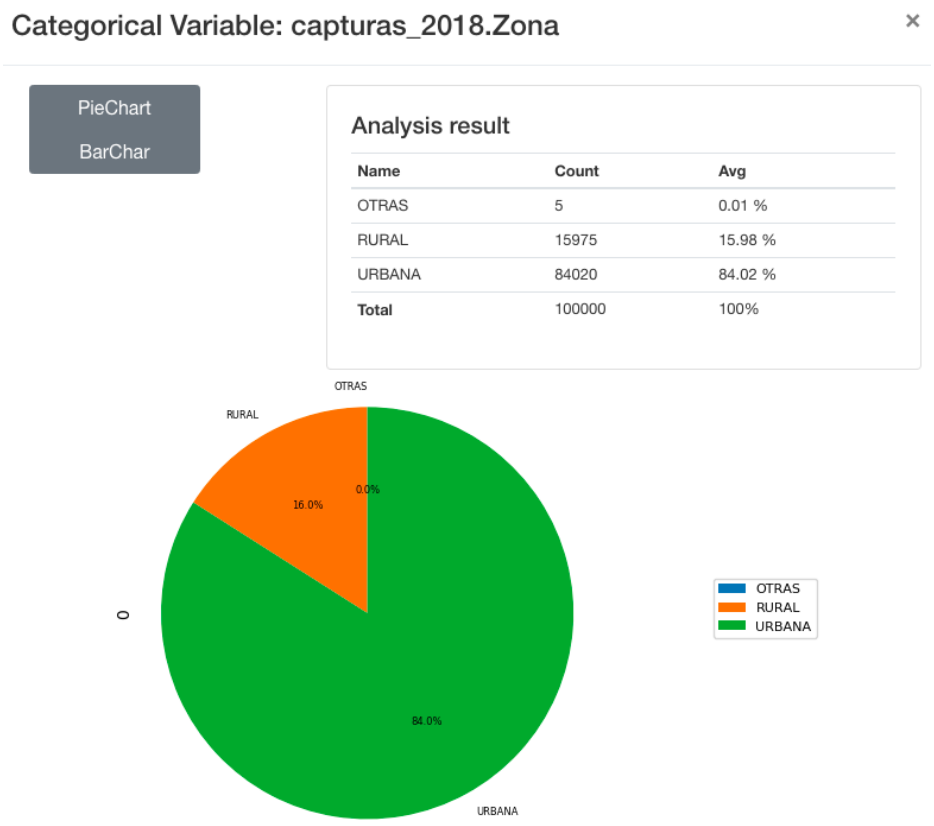

Figura 26. Resultados de la variable "Zona" de la tabla "capturas_2018" mediante un “Pie chart”

Como se puede observar en la gráfica anterior, en este tipo de análisis se cuantifican cada uno de los valores de la variable, determinando su frecuencia y el porcentaje de aparición acorde al valor total de datos. Otra opción que ofrece el sistema es el gráfico de barras, el cual facilita la comparación de las frecuencias de los diferentes datos que compone la variable.

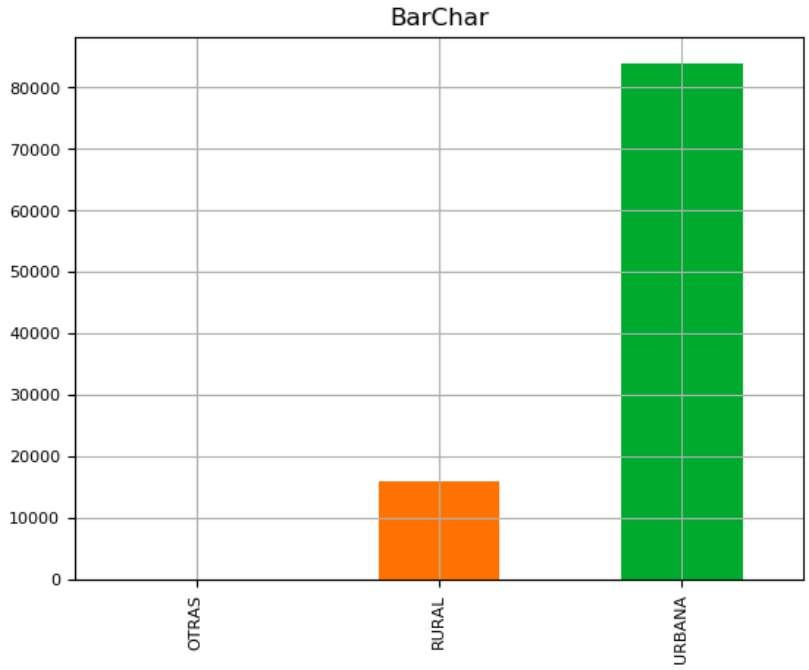

Figura 27. Diagrama de barras de la variable "Zona".

5.2.2. Análisis bivariado:

Se inicia con la selección de la tabla a analizar, en este caso “*Capturas_2018*”, posteriormente se eligen dos de sus variables para observar su comportamiento, como "*Edad*” y “*Delito_Captura*”

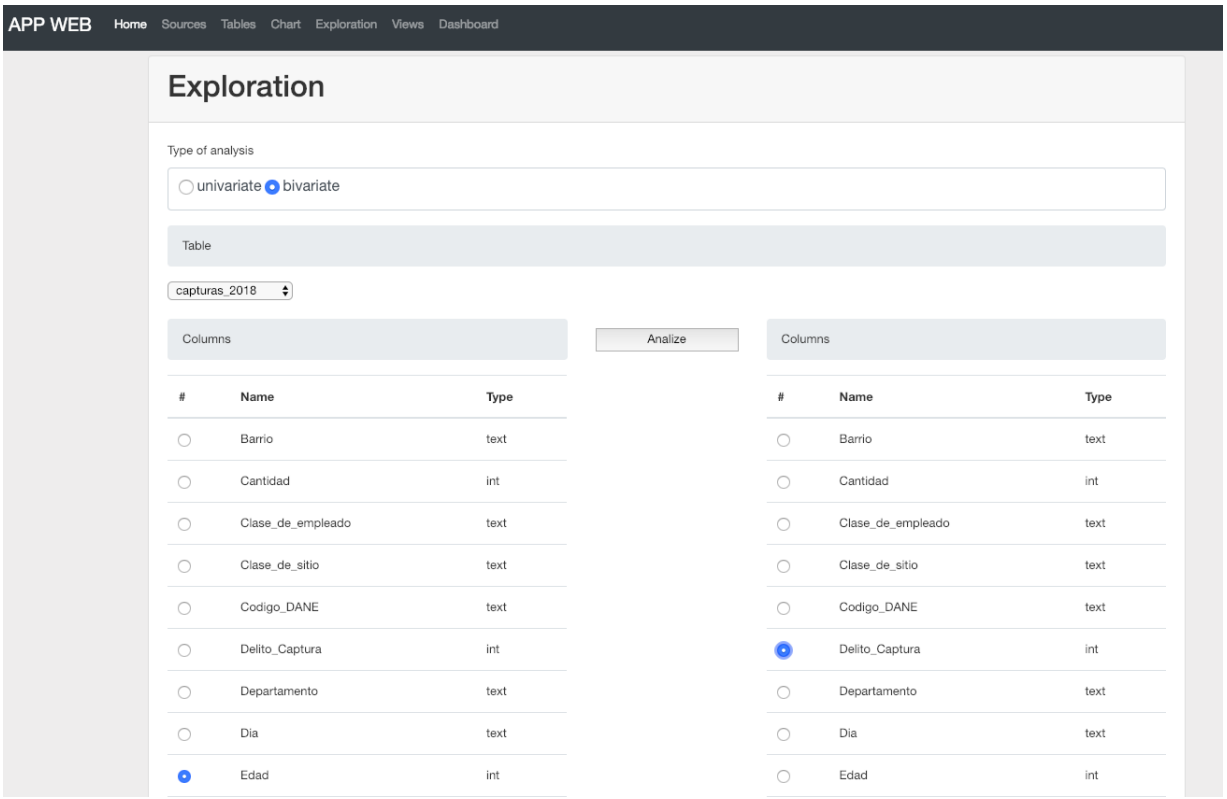

Figura 28. Ventana de selección de campos para el análisis bivariado.

Si ambas variables son numéricas se procede a hacer un análisis de dispersión y correlación de las mismas, mostrando el comportamiento según el gráfico seleccionado. En la figura 24 se pueden apreciar las medidas de tendencia central de cada variable junto con el gráfico de dispersión y en la figura 25 se puede apreciar el gráfico de correlación.

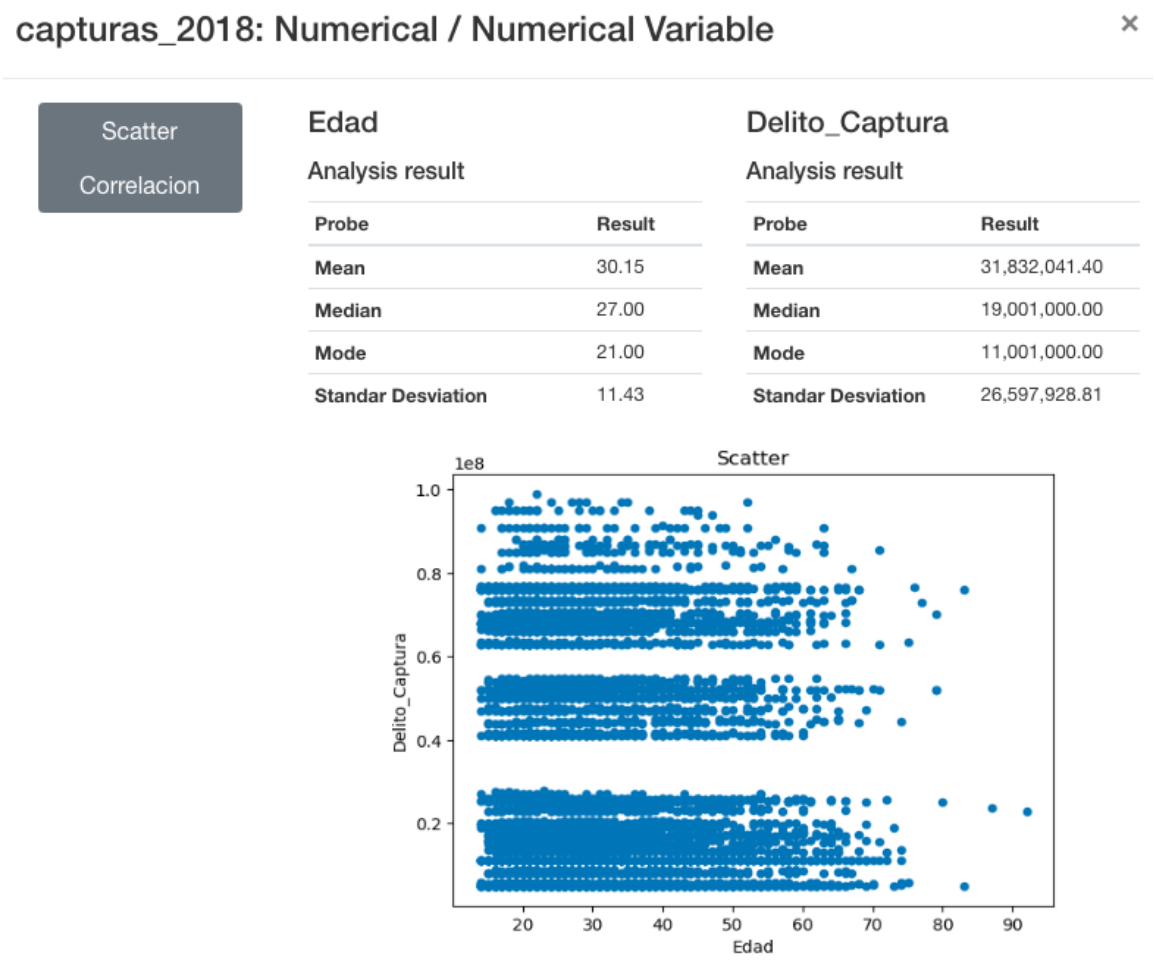

Figura 29. Resultado del análisis bivariado con variables numéricas.

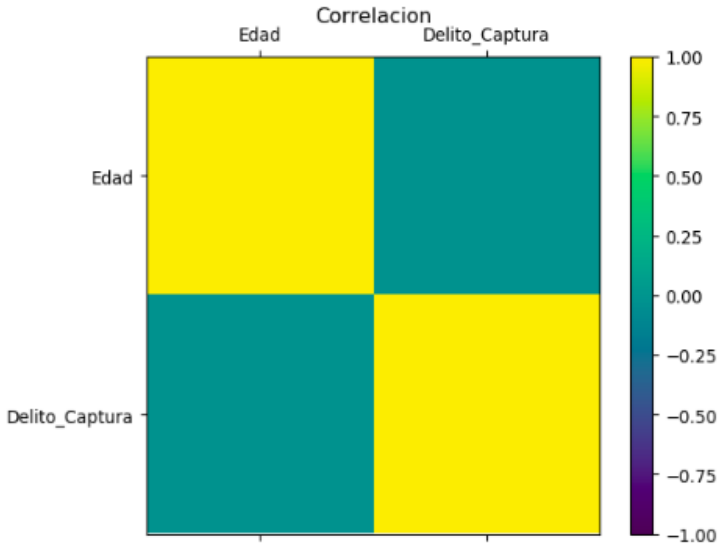

Figura 30. Resultado del análisis de correlación de las variables.

Si las variables seleccionadas son categóricas, se procede a realizar un análisis de frecuencias donde se combinan para observar la relación existente entre ellas (Figura 26).

Figura 31. Diagrama de barras apiladas con la información de las variables analizadas.

6. Conclusiones

La integración del componente de exploración de datos ayudó a la arquitectura, a ofrecer al usuario a obtener un conocimiento más amplio sobre los datos dando así un aporte para soportar de una mejor manera el futuro proceso de toma de decisiones.

Al ejecutar un caso de prueba con información real, se pudo evidenciar que las visualizaciones obtenidas como resultado de los diferentes tipos de análisis se adecuaron de buena forma a la naturaleza de los datos y las variables analizadas.

Se analizaron las técnicas de exploración de datos destacando la importancia para entender la información almacenada, la naturaleza y emplear el potencial en beneficio del usuario.

Las arquitecturas construidas por componentes ofrecen a los desarrolladores o diseñadores de software posibilidades infinitas a la hora de extender sus funcionalidades e integrar nuevos componentes.

Se considera viable la integración de nuevos componentes orientados a enriquecer la arquitectura actual para profundizar en alguna de las etapas que conforman un proceso de analítica de datos.

7. Recomendaciones y trabajo futuro

Teniendo en cuenta el diseño de la arquitectura tanto del componente de exploración de datos como la del proyecto de grado (Betancurt Obando & Abril Pérez, 2018), se plantea como trabajo futuro la integración de un nuevo componente enfocado a la implementación de la exploración visual de datos del análisis multivariado, como también componentes enfocados a las demás fases de la analítica de datos como lo son la fase de extracción de datos, preparación de datos, modelado predictivo y modelado de validación.

Bibliografía

BBVAOPEN4U. (2016). La nueva herramienta de exploración de datos de alta dimensión se llama EVA. Recuperado el 16 de julio de 2018, a partir de https://bbvaopen4u.com/es/actualidad/la-nueva-herramienta-de-exploracion-de-datos-de-alta-dimension-se-llama-eva

Betancurt Obando, P. A., & Abril Pérez, J. D. (2018). *Diseño de una arquitectura por componentes para la visualización de datos utilizando Dashboards*. Universidad Tecnológica de Pereira.

Brown, S. (2018). The C4 model for software architecture. Recuperado el 22 de octubre de 2018, a partir de https://c4model.com/

Doglio, F. (2018). *REST API Development with Node . js* (Second Edi). La Paz, Canelones, Uruguay: Apress. https://doi.org/https://doi.org/10.1007/978-1-4842-3715-1 Library

Dr. Saed Sayad. (2018). An Introduction to Data Science.

Erl, T. (2016). Service-Oriented Architecture Concepts, Technology, and Design. *Prentice Hall*, *1*.

Flores Segovia, M. A., & Villarreal González, A. (2014). Exploración de la geografía de la innovación en México por medio del análisis de datos espaciales. *El Trimestre Económico*, *LXXXI*(2), 517–544.

Hernandez Leal, E. J., Duque Méndez, N. D., & Moreno Cadavid, J. (2013). Big Data: una exploración de investigaciones, tecnologías y casos de aplicación. *Tecno Lógicas*, *20*(Vlsid), 25.

IEEE Computer Society. (2014). *Swebok V3.0*. *Guide to the Software Engineering Body of Knowledge*. https://doi.org/10.1234/12345678

International Business Machines. (2013). The IBM big data platform. En *International Business Machines* (2013a ed., p. 4). IBM.

Li, L., & Chou, W. (2011). Design and Describe REST API without Violating REST: A Petri Net Based Approach. En *2011 IEEE International Conference on Web Services* (pp. 508–515). https://doi.org/10.1109/ICWS.2011.54

McKinney, W. (2018). *Python for Data Analysis. Data Wrangling with Pandas, NumPy, and Ipython*. *Transplantation* (2nd Editio, Vol. 71). Sebastopol, CA: O'Reilly. https://doi.org/10.1097/00007890-200105270-00005

Microsoft Azure. (2017). Exploración de datos en el proceso de ciencia de datos en equipos. Recuperado el 1 de septiembre de 2018, a partir de https://docs.microsoft.com/es-es/azure/machine-learning/team-data-science-process/explore-data

Microsoft Corporation. (2009). *Microsoft Application Architecture Guide* (2nd Edition). Patterns & Practices. Recuperado a partir de https://www.intertech.com/Downloads/eBook/ApplicationArchitectureGuide.pdf

Nelli, F. (2015). *Python Data Analytics*. New York: Apress.

Nualart Vilaplana, J., Pérez Montoro, M., & Whitelaw, M. (2014). Cómo dibujamos textos. Revisión de propuestas de visualización y exploración textual. *El profesional de la información*, *23*, 15. https://doi.org/10.3145/epi.2014.may.02

Polanco, X. (1997). Infometría e Ingeniería del Conocimiento: Exploración de Datos y Análisis de la Información en vista del Descubrimiento de Conocimientos. En Tercer Mundo Editores (Ed.), *El universo de la medición : La perspectiva de la Ciencia y la Tecnología* (pp. 335–350). Bogotá: Colciencias, Cyted, Ricyt. Recuperado a partir de https://www.oei.es/historico/salactsi/polanco4.htm

Posada Hernandez, G. (2016). *ELEMENTOS BÁSICOS DE ESTADÍSTICA DESCRIPTIVA para*

el análisis de datos.

Rumbaugh, J., Jacobson, I., & Booch, G. (2004). *The Unified Modeling Language Reference Manual* (Second Edition). Boston: Addison-Wesley.

Rumbaugh, J., Jacobson, I., & Grady, B. (2004). *Advanced Praise for The Unified Modeling Language Reference Manual, Second Edition*. *Zentralblatt für Bakteriologie, Parasitenkunde, Infektionskrankheiten und Hygiene. Erste Abteilung Originale. Reihe A: Medizinische Mikrobiologie und Parasitologie* (Vol. 240).

Tan, P.-N., Michael, S., & Kumar, V. (2014). *Introduction To Data Mining* (Vol. 58). New York: Pearson.

ANEXOS

Anexo 1. Estructura de datos del archivo "Resultados_Saber_Pro_Competencias_Genericas_2018-1.csv" referente a los resultados de las pruebas Saber Pro del primer semestre del año 2018

ESTU_TIPODOCUMENTO	Texto simple
ESTU_NACIONALIDAD	Texto simple
ESTU_GENERO	Texto simple
ESTU_FECHANACIMIENTO	Fecha y hora
PERIODO	Número
ESTU_CONSECUTIVO	Texto simple
ESTU_ESTUDIANTE	Texto simple
ESTU_PAIS_RESIDE	Texto simple
ESTU_TIENEETNIA	Texto simple
ESTU_ETNIA	Texto simple
ESTU_LIMITA_MOTRIZ	Texto simple
ESTU_LIMITA_INVIDENTE	Texto simple
ESTU_LIMITA_CONDICIONESPECIAL	Texto simple
ESTU_LIMITA_SORDO	Texto simple
ESTU_LIMITA_AUTISMO	Texto simple
ESTU_DEPTO_RESIDE	Texto simple
ESTU_COD_RESIDE_DEPTO	Texto simple
ESTU_MCPIO_RESIDE	Texto simple
ESTU_COD_RESIDE_MCPIO	Texto simple
ESTU_AREARESIDE	Texto simple
ESTU_ESTADOCIVIL	Texto simple
ESTU_COLE_TERMINO	Texto simple
ESTU_CODDANE_COLE_TERMINO	Texto simple
ESTU_COD_COLE_MCPIO_TERMINO	Texto simple
ESTU_OTROCOLE_TERMINO	Texto simple
ESTU_TITULOOBTENIDOBACHILLER	Texto simple
ESTU_VALORMATRICULAUNIVERSIDAD	Texto simple
ESTU_PAGOMATRICULABECA	Texto simple
ESTU_PAGOMATRICULACREDITO	Texto simple
ESTU_PAGOMATRICULAPADRES	Texto simple
ESTU_PAGOMATRICULAPROPIO	Texto simple
ESTU_COMOCAPACITOEXAMENSB11	Texto simple
ESTU_SIMULACROTIPOICFES	Texto simple
ESTU_ACTIVIDADREFUERZOAREAS	Texto simple
ESTU_ACTIVIDADREFUERZOGENERIC	Texto simple
ESTU_TIPODOCUMENTOSB11	Texto simple
ESTU_SEMESTRECURSA	Texto simple
FAMI_HOGARACTUAL	Texto simple
FAMI_CABEZAFAMILIA	Texto simple
FAMI_NUMPERSONASACARGO	Texto simple

FAMI_EDUCACIONPADRE	Texto simple
FAMI_EDUCACIONMADRE	Texto simple
FAMI_OCUPACIONPADRE	Texto simple
FAMI_OCUPACIONMADRE	Texto simple
FAMI_ESTRATOVIVIENDA	Texto simple
FAMI_PERSONASHOGAR	Texto simple
FAMI_CUARTOSHOGAR	Texto simple
FAMI_TIENEINTERNET	Texto simple
FAMI_TIENESERVICIOTV	Texto simple
FAMI_TIENECOMPUTADOR	Texto simple
FAMI_TIENELAVADORA	Texto simple
FAMI_TIENEHORNOMICROOGAS	Texto simple
FAMI_TIENEAUTOMOVIL	Texto simple
FAMI_TIENEMOTOCICLETA	Texto simple
FAMI_TIENECONSOLAVIDEOJUEGOS	Texto simple
FAMI_CUANTOSCOMPARTEBAÑO	Texto simple
FAMI_NUMLIBROS	Texto simple
ESTU_DEDICACIONLECTURADIARIA	Texto simple
ESTU_DEDICACIONINTERNET	Texto simple
ESTU_HORASSEMANATRABAJA	Texto simple
ESTU_TIPOREMUNERACION	Texto simple
INST_COD_INSTITUCION	Texto simple
INST_NOMBRE_INSTITUCION	Texto simple
ESTU_PRGM_ACADEMICO	Texto simple
ESTU_SNIES_PRGMACADEMICO	Texto simple
GRUPOREFERENCIA	Texto simple
ESTU_PRGM_CODMUNICIPIO	Texto simple
ESTU_PRGM_MUNICIPIO	Texto simple
ESTU_PRGM_DEPARTAMENTO	Texto simple
ESTU_NIVEL_PRGM_ACADEMICO	Texto simple
ESTU_METODO_PRGM	Texto simple
ESTU_NUCLEO_PREGRADO	Texto simple
ESTU_INST_CODMUNICIPIO	Texto simple
ESTU_INST_MUNICIPIO	Texto simple
ESTU_INST_DEPARTAMENTO	Texto simple
INST_CARACTER_ACADEMICO	Texto simple
INST_ORIGEN	Texto simple
ESTU_PRIVADO_LIBERTAD	Texto simple
ESTU_COD_MCPIO_PRESENTACION	Texto simple
ESTU_MCPIO_PRESENTACION	Texto simple
ESTU_DEPTO_PRESENTACION	Texto simple
ESTU_COD_DEPTO_PRESENTACION	Texto simple
MOD_RAZONA_CUANTITAT_PUNT	Número
MOD_RAZONA_CUANTITATIVO_PNAL	Número
MOD_RAZONA_CUANTITATIVO_PGREF	Número
MOD_LECTURA_CRITICA_PUNT	Número

MOD_LECTURA_CRITICA_PNAL	Número
MOD_LECTURA_CRITICA_PGREF	Número
MOD_COMPETEN_CIUDADA_PUNT	Número
MOD_COMPETEN_CIUDADA_PNAL	Número
MOD_COMPETEN_CIUDADA_PGREF	Número
MOD_INGLES_PUNT	Número
MOD_INGLES_DESEM	Texto simple
MOD_INGLES_PNAL	Número
MOD_INGLES_PGREF	Número
MOD_COMUNI_ESCRITA_PUNT	Número
MOD_COMUNI_ESCRITA_DESEM	Número
MOD_COMUNI_ESCRITA_PNAL	Número
MOD_COMUNI_ESCRITA_PGREF	Número
PUNT_GLOBAL	Número
PERCENTIL_GLOBAL	Número
ESTU_ESTADOINVESTIGACION	Texto simple

Anexo 2. Estructura de datos del archivo "Capturas_2018.csv" referente a las capturas reportadas por la DIJIN y la policía nacional entre el 1 de enero al 30 de septiembre del año 2018.

Fecha	Fecha y hora
Departamento	Texto simple
Municipio	Texto simple
Día	Texto simple
Hora	Fecha y hora
Barrio	Texto simple
Zona	Texto simple
Clase de sitio	Texto simple
Edad	Número
Sexo	Texto simple
Estado civil	Texto simple
Clase de empleado	Texto simple
Profesión	Texto simple
Escolaridad	Texto simple
Delito Captura	Número
Código DANE	Texto simple
Cantidad	Número

Anexo 3. Código fuente en Python del servicio de exploración.

Servicio.py

```
from flask import Flask
from flask_restful import Api, Resource, reqparse
from flask_cors import CORS
from metodos import UnivariateNumericalAnalysis
from metodos import UnivariateCategoricalAnalysis
from metodos import BivariateNumericalAnalysis
from metodos import BivariateCategoricalAnalysis
from metodos import GraphicExploration

app = Flask(__name__)
cors = CORS(app, resources={r"/exploration/*": {"origins": "*"}})
api = Api(app)
api.add_resource(UnivariateNumericalAnalysis,
'/exploration/<int:dataSource>/<string:table>/<string:column>/<int:minValue>/<int:maxValue>/<int:histogramGroups>')
api.add_resource(UnivariateCategoricalAnalysis,'/exploration/categorical/<int:dataSource>/<string:table>/<string:column>')
api.add_resource(BivariateNumericalAnalysis,
'/exploration/nbv/<int:dataSource>/<string:table>/<string:column1>/<string:column2>')
api.add_resource(BivariateCategoricalAnalysis,
'/exploration/cbv/<int:dataSource>/<string:table>/<string:column1>/<string:column2>')
api.add_resource(GraphicExploration, '/exploration/picture/<string:method>')
app.run(debug=True)
```

metodos.py

```
from flask_restful import Resource
from collections import Counter
from numpy import linspace, exp, cos
from scipy import stats
from io import BytesIO
import matplotlib.pyplot as plt
import requests, json
import numpy as np
import pandas
import base64


dataTable1 = []
dataTable2 = []
dataP = []
categories1 = []
categories2 = []
categorieValue1 = []
categorieValue2 = []
variableName1 = ""
variableName2 = ""
histogram = 50
```

```
class UnivariateNumericalAnalysis(Resource):
  def get(self, dataSource, table, column, minValue, maxValue, histogramGroups):
    global dataTable1, histogram
    dataTable1 = []
    histogram = histogramGroups
    if minValue != maxValue:
      sql = 'sql=select '+column+' from '+table+' where '+column+' between '+str(minValue)+' and '+str(maxValue)
    else:
      sql = 'sql=select '+column+' from '+table
    response = requests.get( 'http://127.0.0.1:9090/api/sql/'+str(dataSource)+'/query?'+sql)
    jData = json.loads(response.content)
    for i in jData:
      dataTable1.append( i[column])
    moda = stats.mode(dataTable1)
    datos = {
      "media": np.mean(dataTable1),
      "mediana": np.median(dataTable1),
      "desviacion": np.std(dataTable1),
      "varianza": np.var(dataTable1),
      "minimo": np.min(dataTable1),
      "maximo": np.max(dataTable1),
      "cantidad": np.size(dataTable1),
      "moda": moda[0][0]
    }
    return datos, 200

class UnivariateCategoricalAnalysis (Resource):
  def get(self, dataSource, table, column):
    global dataTable1, categories1, categorieValue1
    url = 'http://127.0.0.1:9090/api/sql/'
    dataTable1 = []
    categories1 = []
    categorieValue1 = []
    response = requests.get(url+str(dataSource)+'/query?sql=select '+column+' from '+table)
    jData = json.loads(response.content)
    for i in jData:
      dataTable1.append(i[column])
    datos = Counter(dataTable1)
    for key, val in datos.items():
      categories1.append(key)
      categorieValue1.append(val)
    return datos, 200

class BivariateCategoricalAnalysis (Resource):
  def get(self, dataSource,table,column1,column2):
    global dataTable1,dataTable2,categories1,categories2,categorieValue1,categorieValue2,dataP url =
'http://127.0.0.1:9090/api/sql/'
    dataTable1 = []
```

```
        dataTable2 = []
        categories1 = []
        categories2 = []
        categorieValue1 = []
        categorieValue2= []
        response = requests.get(url+str(dataSource)+'/query?sql=select '+column1+','+column2+' from '+table)
        jData = json.loads(response.content)
        dataP = pandas.DataFrame(jData)
        for i in dataP[column1]:
          dataTable1.append(i)
        datos = Counter(dataTable1)
        for key, val in datos.items():
          categories1.append(key)
          categorieValue1.append(val)
        for i in dataP[column2]:
          dataTable2.append(i)
        datos2 = Counter(dataTable2)
        for key, val in datos2.items():
          categories2.append(key)
          categorieValue2.append(val)
        return [datos, datos2], 200

class BivariateNumericalAnalysis(Resource):
  def get(self, dataSource, table, column1, column2):
        global dataTable1, dataTable2, variableName1, variableName2
        url = 'http://127.0.0.1:9090/api/sql/'
        variableName1 = column1
        variableName2 = column2
        dataTable1 = []
        dataTable2 = []
        # dataTable1
        response = requests.get(url+str(dataSource)+'/query?sql=select '+column1+' from '+table)
        jData = json.loads(response.content)
        for i in jData:
          dataTable1.append(i[column1])
        # dataTable2
        response = requests.get(url+str(dataSource)+'/query?sql=select '+column2+' from '+table)
        jData = json.loads(response.content)
        for i in jData:
          dataTable2.append(i[column2])
        moda1 = stats.mode(dataTable1)
        moda2 = stats.mode(dataTable2)
        datos = {
          "media1": np.mean(dataTable1),
          "mediana1": np.median(dataTable1),
          "desviacion1": np.std(dataTable1),
          "varianza1": np.var(dataTable1),
          "moda1": moda1[0][0],
```

```
        "media2": np.mean(dataTable2),
        "mediana2": np.median(dataTable2),
        "desviacion2": np.std(dataTable2),
        "varianza2": np.var(dataTable2),
        "moda2": moda2[0][0]
    }
    return datos, 200

class GraphicExploration(Resource):
  def get(self, method):
    global dataTable1, dataTable2, categorieValue1, dataP
    global categories1, variableName1, variableName2, histogram
    plt.figure()
    if method=="histo" or method=="":
      pandas.DataFrame(dataTable1).plot(kind='hist',bins=histogram,grid=True, rwidth=0.8, legend=False, fontsize=10)
      plt.title('Histogram')
    elif method=="boxpl":
      pandas.DataFrame(dataTable1).plot(kind='box')
      plt.title('BoxPlot')
    elif method == "lines":
      pandas.DataFrame(dataTable1).plot(legend=False, grid=True)
      plt.title("Linear")
    elif method == "pie":
      pandas.DataFrame(categorieValue1).plot(kind='pie', subplots=True, labels=categories1, fontsize=6, legend=True,
autopct='%.1f%%', startangle=90)
      plt.legend(bbox_to_anchor=(0.9,0.5), loc="center right", bbox_transform=plt.gcf().transFigure, prop={'size': 8})
      plt.subplots_adjust(left=0.0, bottom=0.1, right=0.6)
      plt.axis('equal')
    elif method == "bar":
      datos = pandas.DataFrame({'lab':categories1, 'val':categorieValue1}).plot.bar(x='lab', y='val', grid=True, legend=False,
fontsize=8)
      plt.title("BarChar")
    elif method == 'scatter':
      pandas.DataFrame({ variableName1:dataTable1, variableName2: dataTable2} ).plot(kind='scatter', x=variableName1,
y=variableName2)
      plt.title('Scatter')
    elif method == "correl":
      datos = pandas.DataFrame({variableName1:dataTable1, variableName2:dataTable2})
      fig = plt.figure()
      ax = fig.add_subplot(111)
      print datos.corr()
      cax = ax.matshow(datos.corr(), vmin=-1, vmax=1)
      fig.colorbar(cax)
      ticks = np.arange(0,2,1)
      ax.set_xticks(ticks)
      ax.set_yticks(ticks)
      ax.set_xticklabels([variableName1, variableName2])
      ax.set_yticklabels([variableName1, variableName2])
```

```
        plt.title('Correlacion')
    elif method == 'stacked':

dataP.groupby([dataP.columns[0],dataP.columns[1]])[dataP.columns[0]].count().unstack(dataP.columns[1]).fillna(0).plot(kind='bar',st
acked=True,fontsize=8)
        plt.legend(prop={'size': 8})
        plt.title('Stacked')
        figfile = BytesIO()
        plt.savefig(figfile, format='png', bbox_inches="tight")
        figfile.seek(0) # rewind to beginning of file
        figdata_png = base64.b64encode(figfile.getvalue())
        return figdata_png, 200
```

Anexo 4. Contexto del componente de exploración.

Para realizar la adición de los componentes arquitectónicos al módulo de exploración de datos, se agregan las interfaces visuales que garanticen la entrada de los datos, con el fin de brinda al usuario la posibilidad de elegir el tipo de análisis que desea realizar, para este caso puede ser análisis univariado o bivariado.

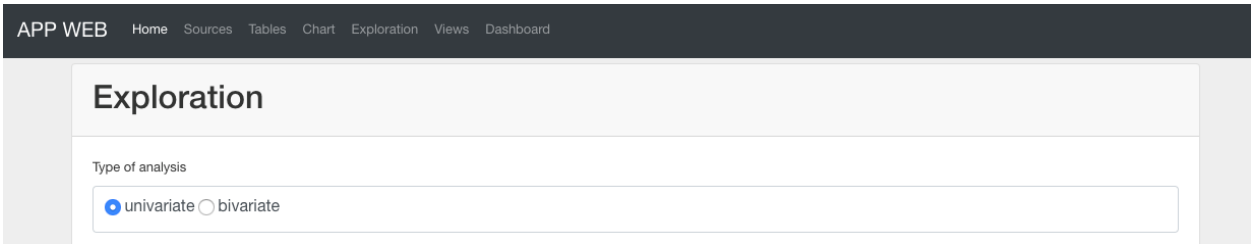

Figura 32. Interfaz de exploración para elección del tipo de análisis.

El análisis univariado da al usuario la posibilidad de explorar las columnas contenidas en las tablas de manera individual y obtener visualizaciones sobre el comportamiento de cada variable por separado. Para esto el usuario debe seleccionar la tabla de la lista desplegable que se le presenta en la interfaz.

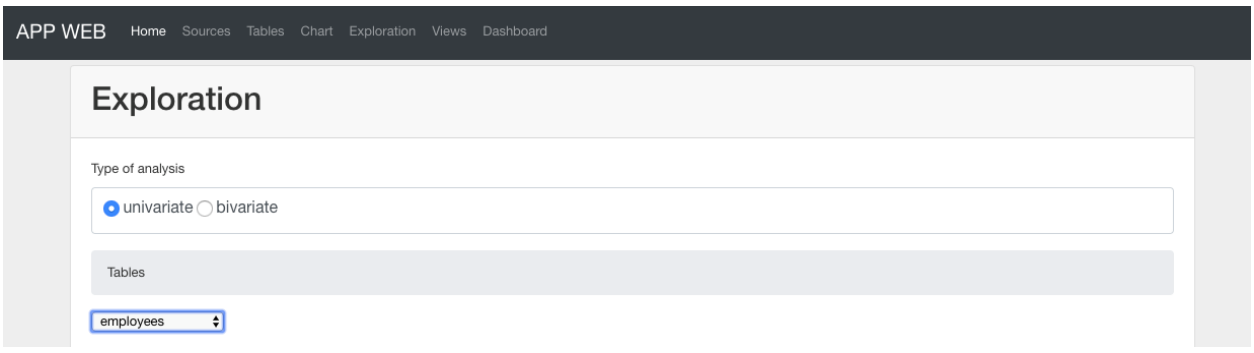

Figura 33. Interfaz para selección de tablas.

Luego de seleccionar la tabla de la lista, se cargan automáticamente todas las columnas que posee, dando al usuario la posibilidad de elegir qué columna desea analizar mediante la opción “*Analize*”.

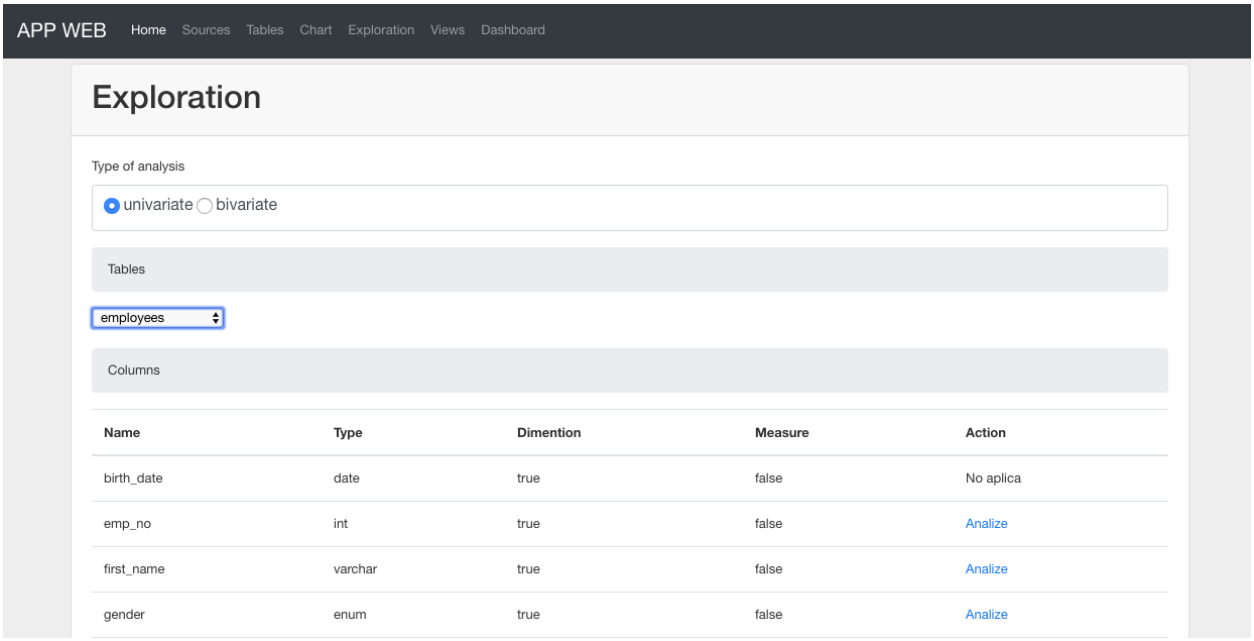

Figura 34. Interfaz para análisis univariado de variables.

Al elegir la opción de analizar, el sistema despliega una ventana modal con varias opciones para visualizar el comportamiento de su variable. Dependiendo de la clasificación de la variable (categórica o numérica), la ventana modal presenta al usuario diferente información. Para las categóricas presenta el conteo y las frecuencias porcentuales de cada categoría y para las numéricas un resumen de las medidas de tendencia central.

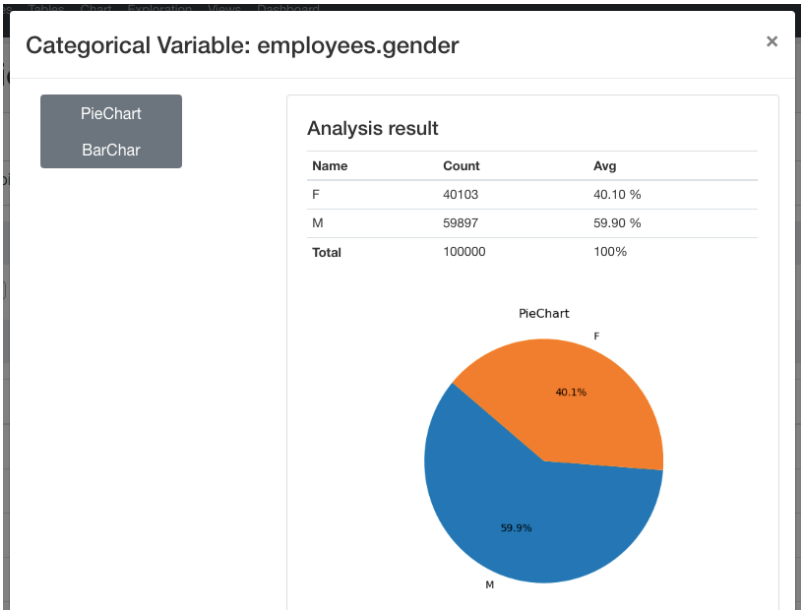

Figura 35. Ventana modal para variables categóricas.

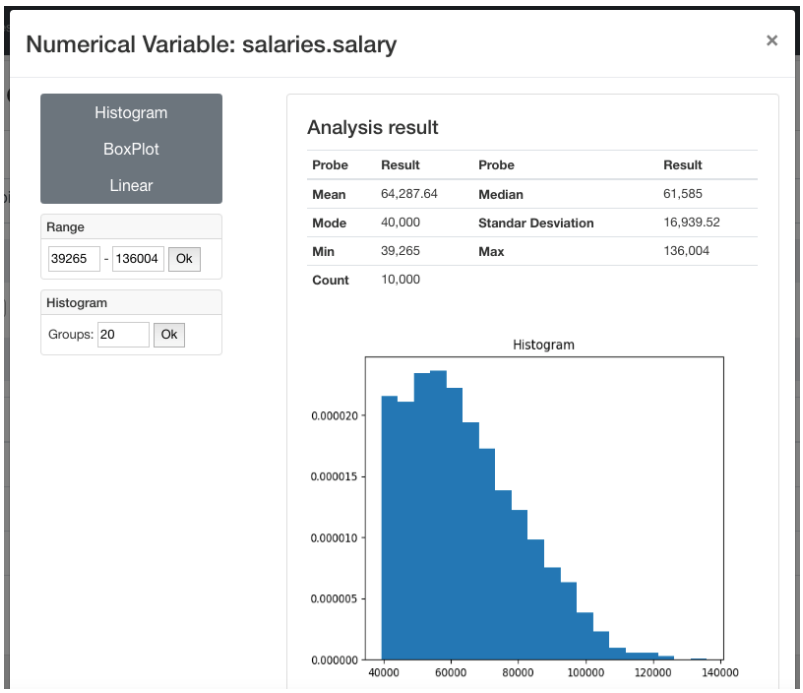

Figura 36. Ventana modal para variables numéricas.

Para el análisis bivariado se cuenta con una interfaz, en la cual el usuario debe seleccionar dos columnas de las tablas previamente definidas para generar el análisis correspondiente.

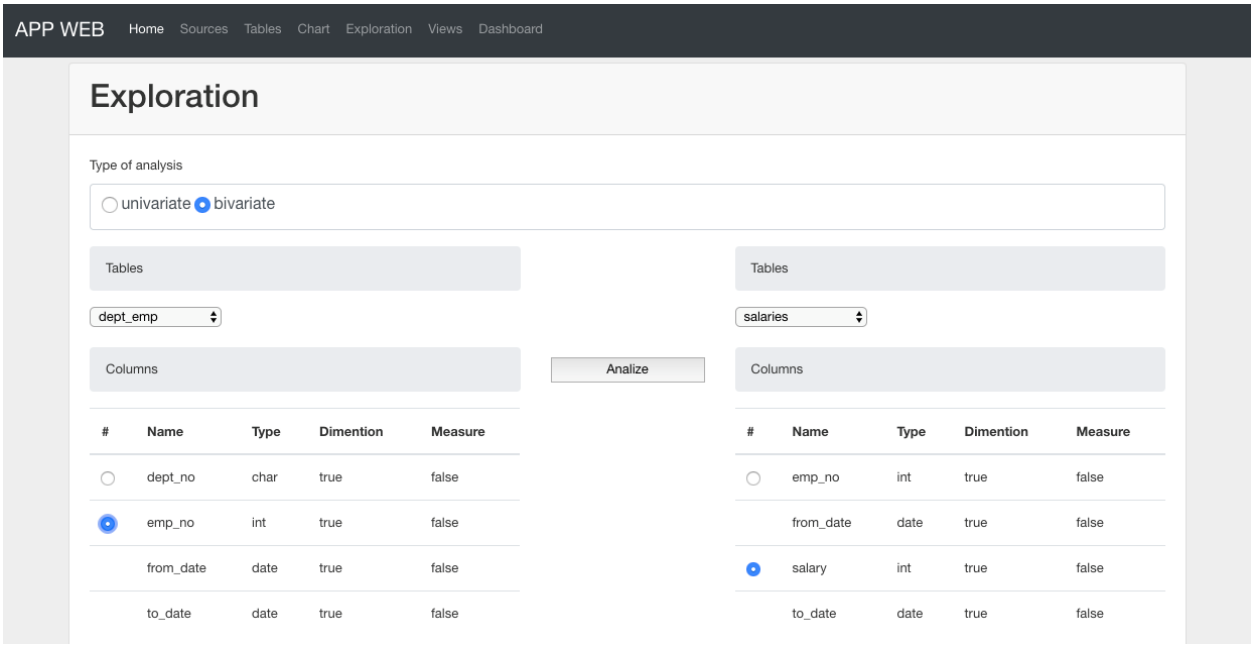

Figura 37. Interfaz de elección de variables para el análisis bivariado.

Tras la elección de las columnas mediante la opción analizar se exhibe al usuario una ventana modal, la cual resume información de cada variable y las representa en un conjunto de gráficas, según su naturaleza. Vale la pena resaltar que la información suministrada en la ventana varía en función de las características de la variable: categórica o numérica.

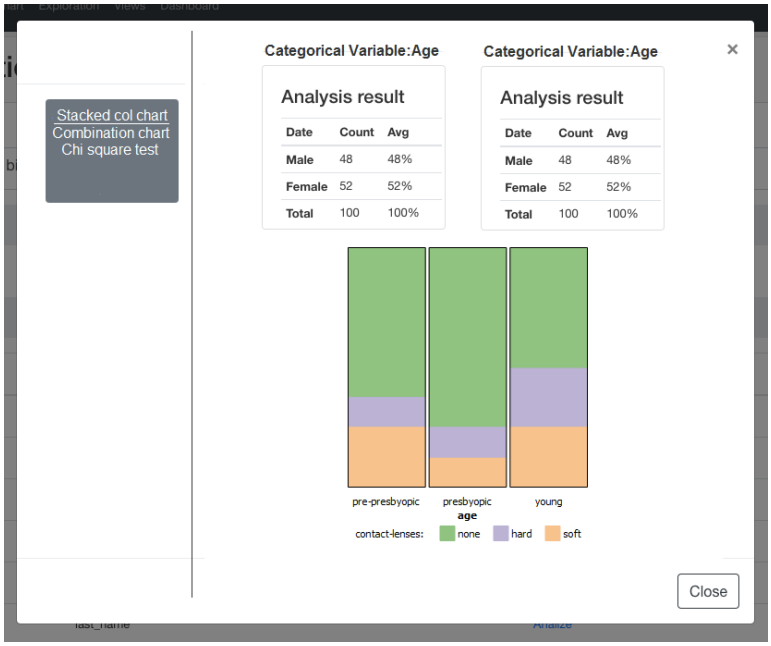

Figura 38. Análisis bivariado, variable categórica vs categórica.

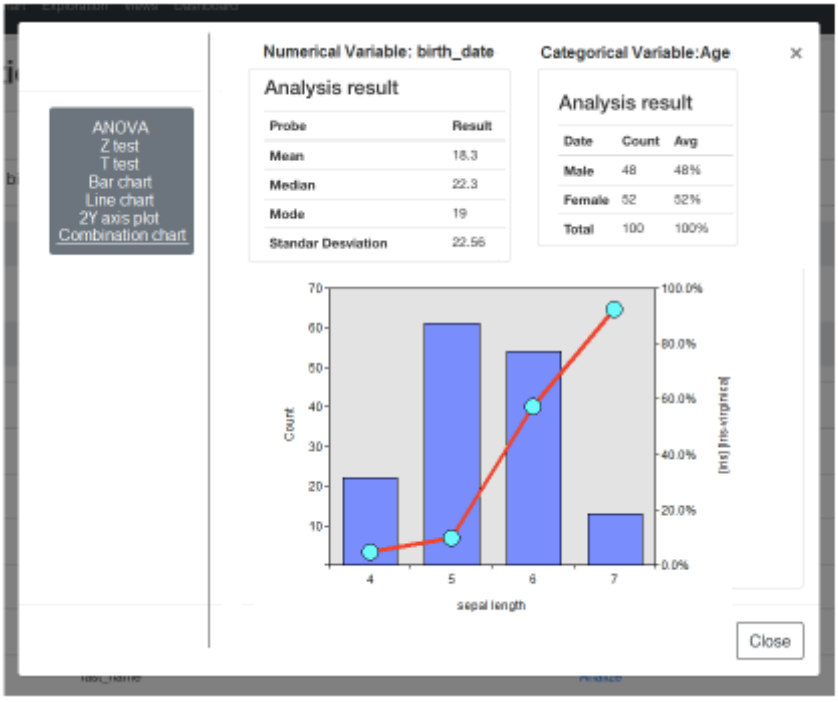

Figura 39. Análisis bivariado, variable numérica vs categórica.

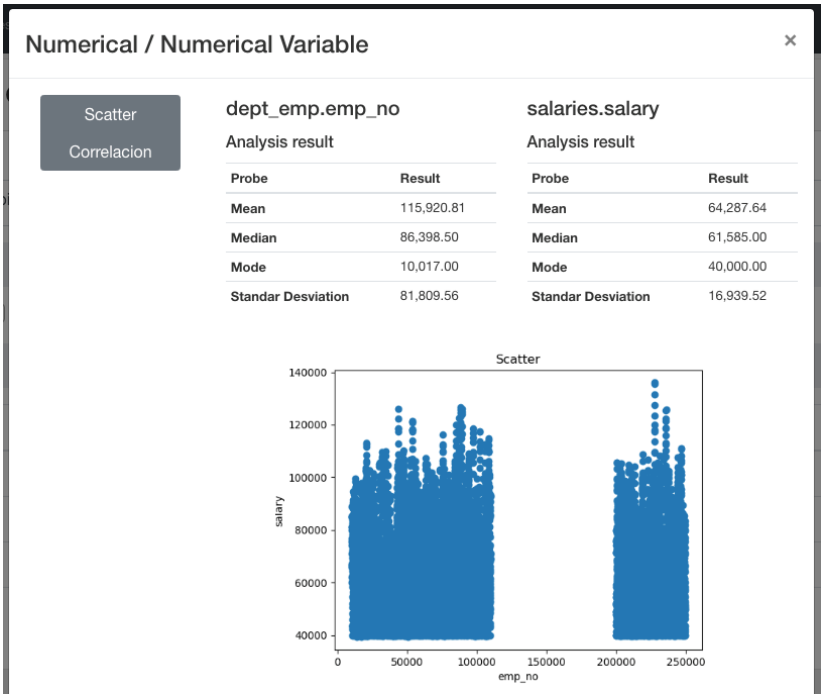

Figura 40. Análisis bivariado, variable numérica vs numérica.