DESARROLLO DE UN PROTOTIPO DE SISTEMA DE INFORMACIÓN PARA LA APLICACIÓN DEL MODELO DE CLUSTERIZACIÓN PARA EL RUTEO DE VEHÍCULOS PARA UNA RED DE DISTRIBUCIÓN DE MERCANCÍAS DE UN OPERADOR LOGÍSTICO EN COLOMBIA

ADRIAN CAMILO DÍAZ BARRETO - 625737

UNIVERSIDAD CATÓLICA DE COLOMBIA

FACULTAD DE INGENIERÍA

INGENIERÍA DE SISTEMAS

BOGOTÁ

2020

DESARROLLO DE UN PROTOTIPO DE SISTEMA DE INFORMACIÓN PARA LA APLICACIÓN DEL MODELO DE CLUSTERIZACIÓN PARA EL RUTEO DE VEHÍCULOS PARA UNA RED DE DISTRIBUCIÓN DE MERCANCÍAS DE UN OPERADOR LOGÍSTICO EN COLOMBIA.

ADRIAN CAMILO DÍAZ BARRETO - 625737

Asesor

ROGER GUZMÁN

M. Sc. (c) Ingeniería de Sistemas y Computación

UNIVERSIDAD CATÓLICA DE COLOMBIA

FACULTAD DE INGENIERÍA

BOGOTÁ

2020

Nota de aceptación

Aprobado por el comité de grado en cumplimiento de los requisitos exigidos por la Facultad de Ingeniería y la Universidad católica de Colombia para optar al título de Ingeniero de Sistemas.

_____

Juan Carlos Barreto

Jurado

Roger Enrique Guzmán Avendaño, Msc

Asesor

DEDICATORIA

Dedico este proyecto a mi familia, la cual me ha apoyado en todo momento de manera incondicional para sacar adelante este proyecto de vida, estando presente en todos los momentos importantes para el desarrollo de mi carrera, depositando toda su confianza en mí y con la plena convicción de que se cumplirían las metas propuestas, para convertirme en un ciudadano capaz de aportar a la sociedad y al país.

AGRADECIMIENTOS

Deseo expresar mis más sinceros agradecimientos a todas las personas que aportaron a mi crecimiento profesional y personal, y directa o indirectamente me brindaron su apoyo para contar con fortaleza, constancia y de esa manera cumplir las diferentes metas que surgieron durante el transcurso de la carrera. Sin duda, este impulso y las metas establecidas me permitieron desarrollar este documento, ya en la cúspide de mi carrera.

En primera medida, agradezco infinitamente a mi familia, la cual, de manera incondicional, siempre me ha brindado su apoyo, amor e inmensa comprensión. Agradezco a todos por permanecer presentes durante la evolución de este proyecto y durante todo el transcurso de la carrera, puesto que todo lo que soy es gracias a ellos.

Por último, y de manera muy especial, también quiero agradecer a la Universidad Católica de Colombia, a todos los profesores que me permitieron llegar a este punto, a través de sus enseñanzas fue posible concientizarme y enamorarme aún más de esta linda carrera, y me ofrecieron todas las herramientas pertinentes para llegar a la elaboración de este documento.

CONTENIDO

LISTA DE CUADROS

LISTA DE FIGURAS

LISTA DE ANEXOS

RESUMEN

El presente proyecto de investigación tuvo como objetivo principal desarrollar un prototipo de sistema de información, con el propósito de mitigar los problemas ocasionados al momento de gestionar la ruta más corta de un operador logístico. Lo anterior, mediante una de las metodologías más ágiles, esto es, la metodología Scrum, para el desarrollo de software.

Este prototipo surgió debido a la necesidad de desplegar una interfaz web para la aplicación de un modelo clusterización, y de esa manera encontrar el camino más corto para llevar a cabo la distribución de productos. Este modelo fue implementado en el sistema de información basado en una heurística desarrollada dentro del grupo de investigación científica y desarrollo tecnológico Jóvenes Investigadores de la Universidad Católica de Colombia. Dicha implementación fue llevada a cabo por el ingeniero Andrés Felipe León Villalba, integrante del grupo de investigación, quien, a través de su proyecto titulado “Modelo de ruteo de vehículos para la red de distribución de mercancías de un operador logístico en Colombia”, cumplió a cabalidad con su objetivo propuesto, esto es, gestionar la ruta más corta; sin embargo, este no tuvo en cuenta la creación de una interfaz amigable que permitiera la interacción con los diferentes usuarios.

Por tal motivo, una vez interpretado el modelo, se desarrolló el sistema de información que gestiona los datos suministrados este, a través del lenguaje de programación Python. Este último, apoyado, a su vez, en el marco de referencia de trabajo Django y aplicando modelos de bases de datos relacionales, basados en los módulos de administración ofrecidos por este *framework*. Así pues, fu posible obtener un prototipo con una interfaz eficiente, rápida y amigable para el usuario, permitiendo a los operadores visualizar un mapa con las rutas más cortas a través de un ambiente web.

Finalmente, se realizaron todas las pruebas respectivas al software desarrollado, validando cada una de las funcionalidades establecidas en los levantamientos de requerimientos. Esto, para posteriormente proveer al usuario final un manual que permita la plena manipulación del sistema y, al mismo tiempo, administrar los modelos de datos establecidos para dicho fin.

Palabras clave: logística, inteligencia artificial, sistemas de información, camino más corto.

ABSTRACT

This project looking for develop a prototype of the information system which seeks to solve the various problems presented with the logistics operators when making routing and delivery of commodities, from the use of algorithmic models to find the shortest routes to the accomplishment of the objectives

The information system will be developed in Python programming language supported by the Django framework, which provides many tools for user administration and management of database models.

The present project seeks to create an efficient and lightweight user-friendly interface with the ability to manage logistics operators that allows the display of routes through a map in a web interface, developing all stages of software development and in turn developed all the respective tests to verify its full functionality and corroborate all this the end user.

Keywords: logistics, artificial intelligence, information systems, shortest path.

INTRODUCCIÓN

El diseño de las pruebas que se aplicaron al software desarrollado para el proyecto “Enrutamiento operador logístico”, tuvo como objetivo la validación y correcta funcionalidad de cada uno de los componentes de código presentes en él. Asimismo, se pretendió avalar las interrelaciones de cada uno de los módulos alojados en este, con el propósito de obtener un software de calidad que cumpliera con lo establecido en la planeación del proyecto. Cabe señalar que dichas pruebas vienen sujetas a las debidas restricciones mencionadas en el documento base de trabajo.

La distribución de mercancías ha sido uno de los retos logísticos más grandes para el sector productivo del país. Adicional a ello, el continuo crecimiento de este sector ha significado para la empresas adentrarse en un proceso de reinvención, en aras de alcanzar soluciones óptimas que les ayuden a reducir costos y, en consecuencia, aumentar la productividad y sus capacidades.

De manera concreta, han sido los operadores logísticos quienes han dedicado esfuerzos para administrar sus cadenas de suministros, a partir de sistemas de información de una arquitectura tecnológica enfocada en modelos de datos ágiles y productivos. Lo anterior, con el objetivo de adquirir competitividad en este nicho de mercado, la cual puede ser utilizada para obtener una noción global del negocio y, con ello, brindar oportuna solución a los retos que emergen, debido a la constante demanda de los clientes y la disponibilidad de recursos para la producción, alcanzando así una óptima distribución.

Adicionalmente, los operadores logísticos apoyan su operación en diferentes modelos de ruteo de vehículos con ventanas de tiempo. Estas, básicamente, se centran en la aplicación de modelos matemáticos desarrollados en lenguaje de programación Python sobre grandes volúmenes de información obtenida a partir de diferentes métricas y estadísticas establecidas dentro de las redes de distribución, como es el caso del modelo de ruteo de vehículos para la red de distribución de mercancías para un operador logístico.

Dicho lo anterior, para el desarrollo de este proyecto de investigación, se planteó llevar a cabo el desarrollo de un prototipo de sistema de información. Para ello, se consultó un estado del arte actual, exponiendo más adelante todas las fases de desarrollo de software aplicadas, a saber: planeación, análisis, diseño, desarrollo, pruebas y despliegue; del modelo de ruteo de vehículos. Todo esto, haciendo uso de metodologías ágiles como Scrum, con el fin de generar el mayor impacto económico, debido al ahorro de costos generado a través de la aplicación de este sistema de información. Sin duda, esto motiva la implementación de estos modelos, en tanto que una interfaz de usuario amigable, eficiente, ligera y potente, con capacidad de realizar a cabalidad su objetivo final, es de suma importancia.

1. GENERALIDADES

1.1. LÍNEA DE INVESTIGACIÓN

El presente proyecto aborda el campo de arquitectura de software aplicable a procesos de minería de datos y de interpretación de modelos de clusterización.

1.2. DESCRIPCIÓN DEL PROBLEMA

La construcción del sistema de información es consecuencia directa de la necesidad de desplegar una interfaz web para la aplicación de un modelo de clusterización, para así hallar el camino más corto para llevar a cabo la distribución de productos. Este modelo fue implementado en el sistema de información basado en una heurística desarrollada dentro del grupo de investigación científica y desarrollo tecnológico Jóvenes Investigadores de la Universidad Católica de Colombia. Dicha implementación fue llevada a cabo por el ingeniero Andrés Felipe León Villalba, integrante del grupo de investigación, quien, a través de su proyecto titulado "Modelo de ruteo de vehículos para la red de distribución de mercancías de un operador logístico en Colombia", cumplió a cabalidad con su objetivo propuesto, esto es, gestionar la ruta más corta; sin embargo, este no tuvo en cuenta la creación de una interfaz amigable que permitiera la interacción con los diferentes usuarios.

En dicho orden de ideas, este modelo pretende llevar los asuntos logísticos a un nivel estratégico, táctico y operativo; garantizando la reducción de tiempo y costos, considerando las variables de demanda, puntos de entrega, capacidades, ventanas de tiempo, distancias, horarios y otras, que aporten a la taxonomía del modelo. Asimismo, fue posible identificar que el sector logístico de Colombia afronta una época de cambios, debido a los grandes retos de los mercados durante la segunda década del siglo XXI. A la par con la globalización y los factores socioambientales, la tecnología representa un factor determinante en la proyección y evolución de este importante sector.

Según el Banco Interamericano de desarrollo[1], en su guía de mejores prácticas internacionales como hoja de ruta para Latinoamérica Para encarar la transformación digital de las cadenas de suministro, pone a Colombia en un lugar en el cual se encuentra en un momento de coyuntura debido a los diferentes desafíos en cuanto a la capacidad de impulsar la transformación digital empresarial tanto en el ámbito público como privado, Ya que esta economía es la cuarta más grande de Latinoamérica (PBI 2017: USD $309,191 mil millones).

[1] CALATAYUD, Agustina. Cadena de suministro 4.0: mejores prácticas internacionales y hoja de ruta para América Latina [En línea]. [2019]. Disponible en: https://publications.iadb.org/publications/spanish/document/Cadena_de_suministro_4.0_Mejores_pr%C3%A1cticas_internacionales_y_hoja_de_ruta_para_Am%C3%A9rica_Latina_es.pdf

Año tras año la economía Colombiana Ha incrementado debido a la digitalización de los hogares, como lo es el uso masivo de telefonía celular y alta penetración del uso de banda ancha e internet, Asimismo inversiones gigantescas en infraestructura de telecomunicaciones y conectividad digital de consumo cómo se muestra a continuación:

CARACTERÍSTICA	RANGO TIEMPO	TASA DE CRECIMIENTO ANUAL COMPUESTO
Crecimiento uso telefonía Celular	2004-2019	16.1%
Penetración de banda ancha y uso de internet	2004-2018	15.8%
Infraestructura de las telecomunicaciones	2004-2019	12.46%
Conectividad Digital Empresarial	2004-2019	11.10%
Digitalización de la producción	2004-2019	5.11%

Fuente: elaboración propia

Basado en lo anterior, la digitalización empresarial colombiana ha tenido un crecimiento más lento respecto a las otras características evaluadas, debido a la inversión interior del PIB en tecnologías I+D (0.24% del PIB), respecto a otros países como los son Brasil (1.17% del PIB), Argentina (0.59% del PIB) y México (0.55% del PIB), De igual manera, el Banco Interamericano de desarrollo auspicia un gran aumento al porcentaje de empresas que realizan pedidos de productos o servicios por internet obteniendo una taza de crecimiento anual compuesto del 32.7% siendo evaluados los últimos 5 años anteriores al 2019.

Agregando a lo anteriormente mencionado, Colombia encuentra grandes rezagos comparándose con los países de la OCDE en factores de tecnología, innovación, comercio internacional, inversión, marco institucional, sostenibilidad, Presentando un bajo nivel de desarrollo y requiriendo altas preparaciones para una revolución digital tecnológica. esta preparación se acompaña de la gran adopción de tecnologías digitales en empresas en los últimos 5 años según lo establecido por el Observatorio de la economía digital de Colombia, así:

	2015	2016	2017	CAGR
Porcentaje de empresas que utilizan computadoras	65 %	96%	94%	20.3%
Porcentaje de Empresas que utilizan internet regularmente	62 %	96%	96%	24.4%
Porcentaje de Empresas que poseen Sitio Web	25 %	55%	55%	48.3%
Porcentaje de Empleados que acceden a Internet regularmente	25 %	61%	74%	72.0%
Porcentaje de empresas que acceden a la información sobre insumos por Internet	34 %	90%	82%	55.3%
Porcentaje de empresas que utilizan Internet para obtener información de Gobierno	23 %	79%	60%	61.5%
Porcentaje de empresas que utilizan Internet para la banca electrónica	24 %	59%	62%	60.7%
Porcentaje de empresas que utilizan Internet para interactuar con organizaciones gubernamentales	15 %	49%	37%	57.1%
Porcentaje de empresas que utilizan Internet para proporcionar atención al cliente	39 %	79%	77%	40.5%
Porcentaje de empresas que utilizan Internet para entrega de productos en línea	6 %	17%	28%	116.0%
Porcentaje de empresas que reciben pedidos por Internet	9 %	25%	44%	121.1%
Porcentaje de empresas que realizan pedidos de productos o servicios por Internet	21 %	35%	37%	32.7%
Porcentaje de empresas que utilizan Internet para contratar apoyo externo	10 %	30%	27%	64.3%
Porcentaje de ventas totales por vía electrónica	6 %	7%	12%	41.4%

Fuente: Calatayud, y Katz. Cadenas de suministro 4.0 mejores prácticas internacionales y hoja de ruta para Latinoamérica. Caracas: Banco Interamericano de Desarrollo. 2018

Ondeando la problemática del presente proyecto, basado en los datos suministrados por el Banco Interamericano de desarrollo, se evidencia un considerable crecimiento en el lapso de los últimos 5 años para las empresas que usan internet para la entrega de productos en línea (Crecimiento 116%) y recepción de pedidos (crecimiento 121%), así como el suministro de servicios y apoyo externó a través de internet (crecimiento 64.3%), poniendo el contexto de las carencias en temas de cadenas de suministro las cuales deben ser solventadas principalmente a optando soluciones digitales.

Los apalancamientos enfocados en tecnología dentro del sector logístico del país se encuentran en un proceso de lenta transformación, puesto que cada vez son más las empresas que tienen inmiscuido, dentro de su estrategia de negocio, el uso de las tecnologías de la información y de la comunicación (TIC) para el alcance y cumplimiento de los objetivos, dando lugar a reducciones considerables de costos de operatividad y distribución de mercancías.

Actualmente, uno de los principales problemas que afronta el sector logístico del país es contar con un concepto errado que le invita a centrar sus esfuerzos operativos en temas de infraestructura para el transporte y operatividad en la movilización de carga, dejando a un lado los planteamientos estratégicos que determinen un mejor rumbo en las decisiones tomadas en una cadena de suministros[2], considerando que son muchas empresas que cuentan con procesos desactualizados, no documentados y poco óptimos. Es por ello por lo que terminan llevando un control netamente manual de estos, sin un debido manejo de los tiempos de respuesta, creando así una gran incertidumbre a los clientes sobre los despachos de sus productos. En efecto, todo esto conduce a la carencia de una retroalimentación sobre las acciones realizadas para la gestión logística de una empresa.

En ese sentido, una de las mejores prácticas para tratar de mitigar todos los riesgos derivados de la utilización de métodos manuales que no cuentan con un adecuado control, es la implementación de diferentes sistemas de información, a través de los cuales se pueda obtener datos veraces sobre toda la operatividad dentro de una cadena de suministro que pertenece a una red logística. De esa manera, es posible contar con información de forma rápida y oportuna, así como datos almacenados de manera adecuada y precisa. Del mismo modo, ayuda a planificar y establecer objetivos, siendo estos medidos a través de diferentes métricas contempladas para tal fin. En últimas. todo esto posibilita una reducción considerable de costos de producción, operación y distribución dentro de una empresa logística.

Aquellas empresas que han sistematizado su operatividad han encontrado nuevas problemáticas[3], como lo son:

- Ventanas de tiempo de permanencia en las estaciones de carga y descarga, que no han sido reducidas al mínimo.
- Selección selectiva del equipo a transportar para reducir costos de transporte no ha sido debidamente organizada.
- Rutas de despecho no han sido debidamente optimizadas para la reducción de costos de transporte.

Así pues, se identificó que estas problemáticas pueden ser mitigadas mediante inversiones en TIC y haciendo uso de diferentes algoritmos. Una vez desarrollado esto, es posible hacer reducciones considerables en costos, y dependiendo de la cantidad de volumen de despachos de una empresa, se puede recuperar dicha inversión en un lapso corto, teniendo en cuenta los ahorros generados en la

---

[2] GUTIÉRREZ, Rogelio. ¿Cómo enfrenta hoy la logística colombiana las nuevas necesidades de los mercados? [En línea]. [2019]. Disponible en: https://revistadelogistica.com/logistica/como-enfrenta-hoy-la-logistica-colombiana-las-nuevas-necesidades-de-los-mercados/

[3] PORTAL, Carlos. Costos logísticos: qué son, cuáles son y cómo minimizarlos. [En línea]. [29 de junio de 2011]. Disponible en: https://www.gestiopolis.com/costos-logisticos-que-son-cuales-son-y-como-minimizarlos/

aplicación de estos métodos. En la actualidad, dichos algoritmos han avanzado tanto, que muchos de estos están apoyados en la inteligencia artificial (IA), con el fin de contrarrestar al máximo la materialización de las diferentes problemáticas que pueda tener una empresa logística al momento de realizar despachos de mercancía.

1.3. FORMULACIÓN DEL PROBLEMA

Para una adecuada orientación y desarrollo de este proyecto, se formuló la siguiente pregunta de investigación: ¿Cómo aplicar el modelo de ruteo de vehículos para la red de distribución de mercancías para un operador logístico en Colombia, haciendo uso de un sistema de información, que permita la interacción del usuario con este modelo a través de una interfaz de usuario amigable?

1.4. OBJETIVOS

1.4.1. Objetivo general

Implementar un prototipo funcional de un sistema de información que permita calcular la ruta más corta de red de distribución de mercancías, aplicado a un operador logístico en Colombia.

1.4.2. Objetivos específicos

- Diseñar un sistema de información para optimizar tiempos en las redes de distribución mediante el uso de arquitecturas web.
- Desarrollar un sistema de información para optimizar tiempos en las redes de distribución mediante el uso arquitecturas web y algoritmos.
- Validar el pleno funcionamiento del sistema de información para evaluar su diseño y posibles mejoras mediante pruebas funcionales y de validación de software.

1.5. JUSTIFICACIÓN

La distribución de mercancías ha sido uno de los retos logísticos más grandes para el sector productivo del país. Adicional a ello, el continuo crecimiento de este sector[4] ha significado para la empresas adentrarse en un proceso de reinvención, en aras de alcanzar soluciones óptimas que les ayuden a reducir costos y, en consecuencia, aumentar la productividad y sus capacidades.

---

[4] MONTERROSA, Heidy. Logística se lleva 13,5 % de los ingresos de las compañías en Colombia. [En línea]. [13 de diciembre de 2018]. Disponible en: https://www.larepublica.co/economia/logistica-se-lleva-135-de-los-ingresos-de-las-companias-en-colombia-2805319#:~:text=Una%20empresa%20en%20Colombia%20destina,la%20que%20participaron%202.738%20empresas

De manera específica, los operadores logísticos han dedicado esfuerzos para administrar sus cadenas de suministros y construir una arquitectura tecnológica enfocada en sistemas de información que les permita ganar competitividad. Eso último, en tanto que esta es una red abierta de bajo costo, que puede ser utilizada para tener una noción global del negocio y, de esa manera, brindar una solución rápida a los retos que surgen, como consecuencia de la constante demanda de los clientes y la disponibilidad de recursos. En ese sentido, todo esto obliga a pensar y ejecutar cambios estructurales a la logística de productos, dado que estos cumplen un papel esencial. Estos cambios contribuyen a elevar la competitividad y a ofrecer un valor agregado en términos de tiempo de respuesta al cliente; por ello, los mercados digitales han ampliado considerablemente la demanda, por lo que para el 2019, según cifras del Departamento Administrativo Nacional de Estadística (DANE), el comercio electrónico en Colombia representó el 8.5 % en el PIB[5].

Sumado a ello, los operadores logísticos apoyan su operación en diferentes modelos de ruteo de vehículos con ventanas de tiempo. Estas, básicamente, se centran en la aplicación de modelos matemáticos desarrollados sobre grandes volúmenes de información, obtenidos a través de diferentes métricas y estadísticas establecidas dentro de las redes de distribución. En el caso de las pequeñas y medianas empresas, en su mayoría, cuentan con una estructura logística netamente informal, donde su talento humano carece de muchos conocimientos técnicos que podrían ser aplicados a sus cadenas de distribución.

Considerando lo expuesto hasta el momento, en este proyecto se propuso crear un prototipo de sistema de información, de implementación económica y cómoda para este tipo de empresas, que, a su vez, gestione de manera eficaz las cadenas de suministro a partir de la aplicación de algoritmos de aprendizaje reforzado. Dicho prototipo de sistema de información está enfocado en la aplicación del modelo de ruteo de vehículos, con el fin de generar el mayor impacto económico que motiva la aplicación de este, teniendo en cuenta la importancia de contar con una interfaz de usuario amigable, eficiente, ligera y potente, que esté en la capacidad de realizar a cabalidad su objetivo final.

5 DEPARTAMENTO ADMINISTRATIVO NACIONAL DE ESTADÍSTICA (DANE). PIB por departamento. [En línea]. Disponible en: https://www.dane.gov.co/index.php/estadisticas-por-tema/cuentas-nacionales/cuentas-nacionales-epartamentales#:~:text=Para%20el%20a%C3%B1o%202019pr,de%20millones%20de%20pesos%2C%20respectivamente

Figura 1. Camiones de distribución logística

Fuente: PIXABAY. Camión transporte logística. [En línea]. [2018]. Disponible en: https://pixabay.com/es/photos/cami%C3%B3n-transporte-log%C3%ADstica-1030846/

1.6. DELIMITACIONES

1.6.1. Limitaciones

A través de este proyecto se diseñó una interfaz web, siento el mapa óptimo para una ruta de distribución en una empresa logística. Dicha ruta cuenta con la limitación de un uso de máximo 10 de puntos de la ruta (*waypoints*) desarrollados a partir del uso de la versión gratuita de API de Google Maps (Directions API)[6].

Por otro lado, el peso de las diferentes librerías utilizadas a partir del lenguaje de programación y el *framework* empleado para su desarrollo, esto es, Python y Django, respectivamente, hizo que este proyecto tuviera un peso total de 168 Mb, lo cual sobrepasa los límites permitidos en los diferentes servicios de *hosting* aplicables a la tecnología a desarrollar[7].

1.6.2. Alcance

El presente proyecto se llevó a cabo con base en un conjunto de datos suministrado por el semillero MAILab de la Universidad Católica de Colombia. Lo anterior, puesto que esta línea de investigación aborda diferentes proyectos de minería de datos, lo

---

[6] GOOGLE DEVELOPERS. Directions API Usage and Billing. [En línea]. developers.google.com/ [en línea]. Disponible en: https://developers.google.com/maps/documentation/directions/usage-and-billing#other-usage-limits

[7] PYTHONANYWHERE. Plans and pricing. [En línea]. [2020]. Disponible en: https://www.pythonanywhere.com/pricing/

cual se ajustó correctamente al desarrollo de un prototipo. Todo parámetro fuera de la caracterización de la presente base de datos queda fuera del alcance del presente proyecto.

Así las cosas, se desarrolló un prototipo de sistema de información enfocado en la visualización de la ruta más optima y en la construcción del modelo de datos adecuado de acuerdo con la base de datos suministrada.

2. MARCO REFERENCIAL

2.1. MARCO CONCEPTUAL

2.1.1. Cadenas de suministro

Las cadenas de suministro son todas aquellas actividades involucradas dentro de una operación, con el fin de que una mercancía llegue al cliente final en óptimas condiciones. Todo ello, en función de una estrategia y una logística empresarial esquematizados en un nivel jerárquico, viéndose involucrados desde los proveedores hasta el cliente final[8].

Los medios necesarios para la distribución de mercancías van desde las materias primas, transformación, fabricación, transporte, rutas de distribución y entrega al consumidor final. Esto, teniendo como objetivo primordial la entrega de bienes y servicios a tiempo, evitar pérdidas innecesarias, tener tiempo de distribución optimizado, mantener un adecuado uso y manejo de inventarios y almacenes, tener planes de contingencia antes imprevistos de la demanda u oferta, canales a Ecuador de comunicación y coordinación entre proveedores y clientes, entre muchas otras[9].

Asimismo, cada empresa puede adoptar distintas maneras de elaboración de cadenas de suministro, dependiendo el contexto del negocio. Sin embargo, estas actividades siempre deben estar orientadas a cumplir con la estrategia empresarial, viéndose involucradas en los procesos del negocio, manteniendo un oportuno seguimiento y control a las cadenas de distribución de mercancías, garantizando una administración lógica de existencias y garantías, servicio al cliente, y despacho de mercancías.

2.1.2. Logística de transporte

La logística de transporte es toda actividad que cuenta con una fácil movilización bienes o mercancías protegidas por un embalaje apropiado. Dicha logística se asocia con la estrategia de la empresa y esta se desarrolla en diferentes etapas de análisis, de acuerdo con su tipo o naturaleza. Lo anterior, en aras de mejorar el servicio al cliente, optimizar la fase de mercadeo y obtener un transporte al menor costo posible.

Las actividades dentro de una logística de transporte permiten mejorar la rentabilidad del negocio, incrementando la competitividad para hacer frente a los diversos retos de la industria, convirtiéndose en una metodología de actividades internas y externas empresariales. Dentro de los retos de la logística se encuentran

---

[8] ROLDÁN, Paula. Cadena de suministro. [En línea]. [2017]. Disponible en: https://economipedia.com/definiciones/cadena-de-suministro.html

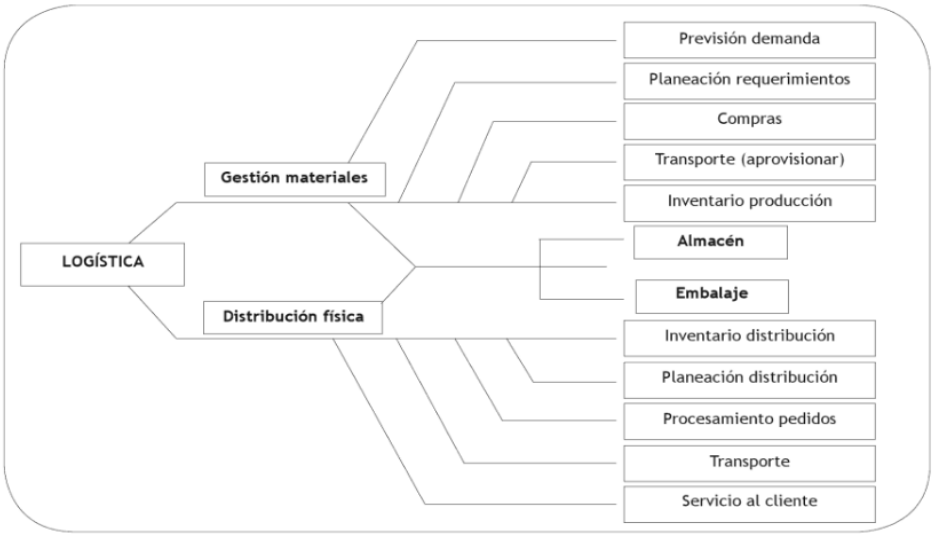
[9] Ibíd.

las flotas de distribuciones, las cuales hacen necesario un proceso de adaptación que considere el contexto de negocio, puesto que no todas las empresas requieren las mismas flotas de distribución; en contraste, muchas empresas apoyan su gestión logística en empresas tercerizadas, debido a que su modelo de negocio es netamente a través de internet[10].

Durante los últimos años se han implementado diferentes software de gestión logística para pequeñas y grandes empresas, lo que les posibilita tener gestión de los datos de despacho, obtener información y valor de este, y mantener comunicadas entre sí las diferentes dependencias de la compañía, editando papeleo y movilización de personal. Estos software trabajan con el objetivo de minimizar costos y aumentar la efectividad de las entregas mediante un uso óptimo de los recursos con los que cuenta la empresa, tratando de mitigar al máximo situaciones inesperadas.

La gestión logística se divide en dos grandes vertientes. La primera hace referencia a la gestión de materiales, la cual, en función de la planeación de requerimientos, garantiza el aprovisionamiento de materias primas para mantener un inventario de producción dentro de un almacén. La segunda consiste en la distribución física del producto final de la empresa, teniendo procesos estandarizados de distribución de inventarios y una debida planeación de distribución, coordinando pedidos y transporte, para así conseguir y conservar la satisfacción del servicio al cliente.

Figura 2. Gráfico gestión logística del transporte



Fuente: RAMÍREZ, Andrés. Manual de la gestión logística del transporte y distribución de mercancías. Barranquilla: Ediciones Uninorte, 2009.

[10] RAMÍREZ, Andrés. Manual de la gestión logística del transporte y distribución de mercancías. Barranquilla: Ediciones Uninorte, 2009.

2.1.3. Arquitectura de software

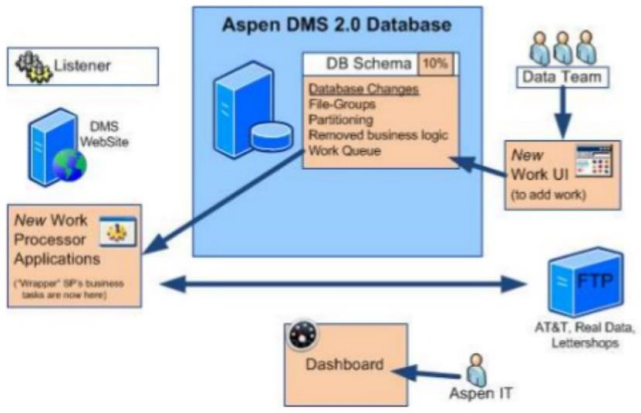
La arquitectura de software es un marco de referencia usado para la construcción de software, el cual sirve como guía para todo aquel que interactúe con un proyecto en específico. Dichas guías se ciñen a diferentes patrones de diseño, en los cuales se evidencian los componentes e interacciones entre ellos, siendo claros los niveles jerárquicos y los flujos de información.

La importancia de una adecuada implementación de una arquitectura de software se centra en que se vean plenamente reflejados los requerimientos funcionales y no funcionales requeridos en un sistema. Esto último, evidenciando de manera tácita los atributos de seguridad, disponibilidad y usabilidad; exponiendo, a su vez, las restricciones a las que pueda estar sujeta dicha arquitectura, tratando de minimizar las interpretaciones subjetivas y ofreciendo claridad sobre la visión del proyecto[11].

La arquitectura debe ser diagramada dependiendo de la necesidad y el contexto del proyecto, siguiendo estándares de modelamiento como lo son Unified Modeling Lenguaje (UML), el cual permite seguir los patrones de secuencia sobre los datos, las clases, los estados, las actividades, entre otras. Estos diagramas pueden estar divididos en diferentes tipos como, por ejemplo, los diagramas de alto nivel, a través de los cuales se visualizan la arquitectura de la empresa y su interacción en los sistemas de información; los diagramas de hardware, que permiten conocer los servidores y la jerarquía entre ellos, así como la aplicación donde se evidencian los componentes en sistemas de información dentro de una arquitectura, y la secuencia donde se observa la relación entre sistemas de información.

[11] NAVARRO, Mirta, *et al*. Integración de arquitectura de software en el ciclo de vida de las metodologías ágiles. Una perspectiva basada en requisitos. [En línea]. Disponible en: http://sedici.unlp.edu.ar/bitstream/handle/10915/62077/Documento_completo.pdf-PDFA.pdf?sequence=1&isAllowed=y

Figura 3. Ejemplo arquitectura de software, esquema general



Fuente: SCHREINER, Keith. A software architect's view on diagramming. [En línea]. Disponible en: https://meghantaylor.net/diagramming-30545905

2.2. MARCO TEÓRICO

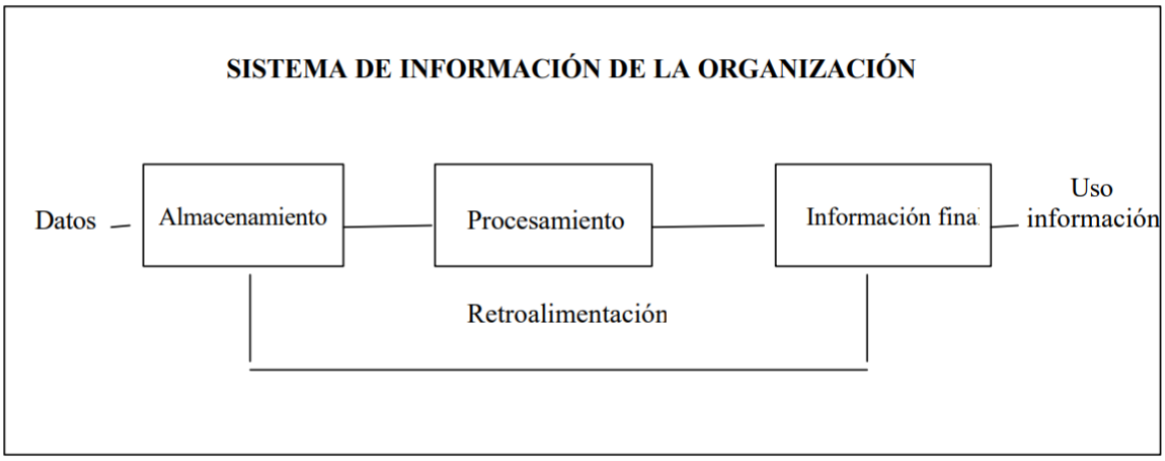
2.2.1. Sistema de información

De acuerdo con Hernández[12], un sistema de información representa un conjunto formal de procesos que operan sobre una estructura de datos, siendo esta definida de acuerdo con las necesidades de una empresa. Asimismo, tiene como objetivo principal la recopilación, elaboración y distribución selectiva de la información, desempeñando funciones de negocio en relación con la estrategia de la empresa.

Por otra parte, la información es el activo más importante dentro de un sistema de información, el cual se obtiene a partir de datos, los cuales son almacenados, procesados y transformados de tal forma que generen valor y una retroalimentación adecuada.

[12] HERNÁNDEZ, Alejandro. Los sistemas de información: evolución y desarrollo. En: Proyecto social: Revista de relaciones laborales, 2003, no 10, p. 149-165.

Figura 4. Flujos de datos de un sistema de información



Fuente: HERNÁNDEZ, Alejandro. Los sistemas de información: evolución y desarrollo. En: Proyecto social: Revista de relaciones laborales, 2003, no. 10, p. 149.

Bajo dicho contexto, resulta pertinente aclarar que el sistema de información no hace referencia a un programa informático para el desarrollo de tal fin. En contraste, el sistema de información abarca más que un tema netamente informático, aborda el enfoque en los modos de obtención de la información y las herramientas que utilizan para su propósito, teniendo en cuenta los conocimientos respectivos sobre TIC y plena apropiación de la estrategia del negocio. Esto, logrando que un sistema de información satisfaga las necesidades, dando lugar al cumplimiento de los objetivos y las estrategias de la empresa para brindar valor a la toma de decisiones, y de esa manera monitorear y controlar las diferentes métricas para la materialización de metas establecidas, entre otras.

2.2.2. Lenguaje de programación

El lenguaje de programación es definido como el conjunto de reglas predefinidas que interpretan una secuencia de palabras para la realización de una tarea en específico[13]. Dichas combinaciones han evolucionado a partir de la interpretación del lenguaje de máquina a lenguajes de programación de alto nivel, entendiendo el primero como aquel que interpreta los cambios de voltaje que ocurren dentro de los transistores del procesador de un equipo de cómputo, los cuales son interpretados por patrones binarios (0s y 1s).

Para nutrir la interpretación del código de máquina, fue necesario hacer un abordaje de este mediante instrucciones con símbolos y mnemónicos, conocidos

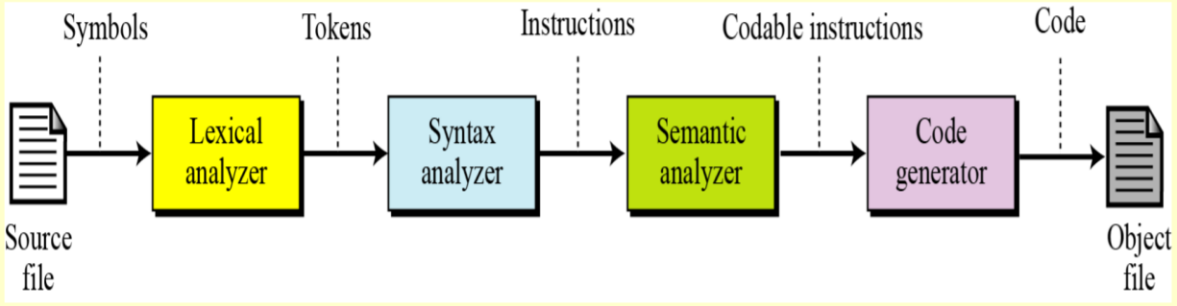
[13] SOLANO, J. Lenguajes de programación. Temario. Lima: Universidad Nacional de Ingeniería, Facultad de Ciencias, 2011.

actualmente como lenguajes ensambladores, que propician la interpretación del lenguaje de máquina y la adecuada interacción entre el hardware de los equipos de cómputo.

En búsqueda de eficiencia para los programadores surgieron los lenguajes de alto nivel, permitiendo mejorar la tediosa tarea de codificar individualmente cada instrucción a lenguaje de máquina. Esto, llevando su codificación a una estructura semejante al lenguaje natural humano, basado en expresiones sintácticas y con la capacidad de construir algoritmos similares a los obtenidos mediante pensamiento humano y no llevados a la ejecución de máquina, lo que comúnmente se conoce como código fuente.

Cabe señalar que los lenguajes de programación realizan un proceso de traducción, en el cual se interpreta y se compilan los diferentes códigos fuentes mediante una serie de instrucciones hasta llegar al punto de traducción a código de máquina, posibilitando la ejecución de la tarea en específico.

Figura 5. Estructura de gestión de lenguajes de programación



Fuente: SOLANO, J. Lenguajes de programación. Temario. Lima: Universidad Nacional de Ingeniería, Facultad de Ciencias, 2011.

2.2.3. Python

Python es un lenguaje de programación de propósito general creado por Guido van Rossum en el año de 1989, por medio de la filosofía del lenguaje de programación ABC. teniendo como objetivo principal establecer un primer acercamiento con las personas que no tenían contacto directo con las ciencias de la computación. Es comúnmente conocido como el primer lenguaje de programación, debido a su gran versatilidad en diferentes ámbitos como lo son aplicaciones web, administración de sistemas, ciencias de datos, computación científica, IA e internet de las cosas[14].

Hoy en día, es uno de los lenguajes de programación más conocidos, puesto que cuenta con alta legibilidad, entornos amigables de desarrollo, potentes librerías para

[14] GARCÍA, José. Python como primer lenguaje de programación textual en la Enseñanza Secundaria. En: Education in the Knowledge Society, 2017, vol. 18, no. 2, p. 147-162.

la interpretación de multiparadigmas y una comunidad entusiasta basada en software libre.

2.2.4. Django

Django es un *framework* escrito en Python, basado en el desarrollo de aplicaciones enfocadas a entornos web y de código abierto con componentes para el desarrollo web ágil fácil y rápido. Este *framework* permite manejar de manera óptima ciertos elementos tediosos para un desarrollador como lo son el manejo de autentificación de usuarios, administración de contenidos, manejo de formularios, carga de archivos servidores, entre otros[15].

Asimismo, posibilita reutilizar componentes ya dispuestos para enfocarlos como servicios dentro de un servidor *backend* con altos estándares de seguridad, permitiendo así mitigar los errores más comunes al momento de programación y evitar la materialización de incidentes de seguridad dentro de servidores *backend*, Adicional a ello, cuenta con una alta capacidad de escalamiento, lo que potencia el crecimiento de diferentes proyectos basados en las demandas de tráfico.

2.2.5. API

La interfaz de programación de aplicaciones (API) es un conjunto de técnicas y protocolos utilizados para integrar aplicaciones de software o sistemas de información. Muchos servicios están desarrollados de manera independiente y las API permiten la comunicación entre ellas sin que se presente alguna discriminación debido a su lenguaje, desarrollo y por su forma de interpretar los datos.

Sumado a lo anterior, las API otorgan flexibilidad gracias a su gran simplificación de diseño y administración, ahorrando tiempo y dinero entre la comunicación de sistemas de información y garantizando la colaboración entre los diferentes equipos de desarrollo. Esta comunicación puede darse entre infraestructuras dentro de una misma red o en aplicaciones nativas de la nube, propiciando comunicaciones a diferentes destinos finales como lo son aplicaciones, servicios de internet de las cosas o dispositivos móviles[16].

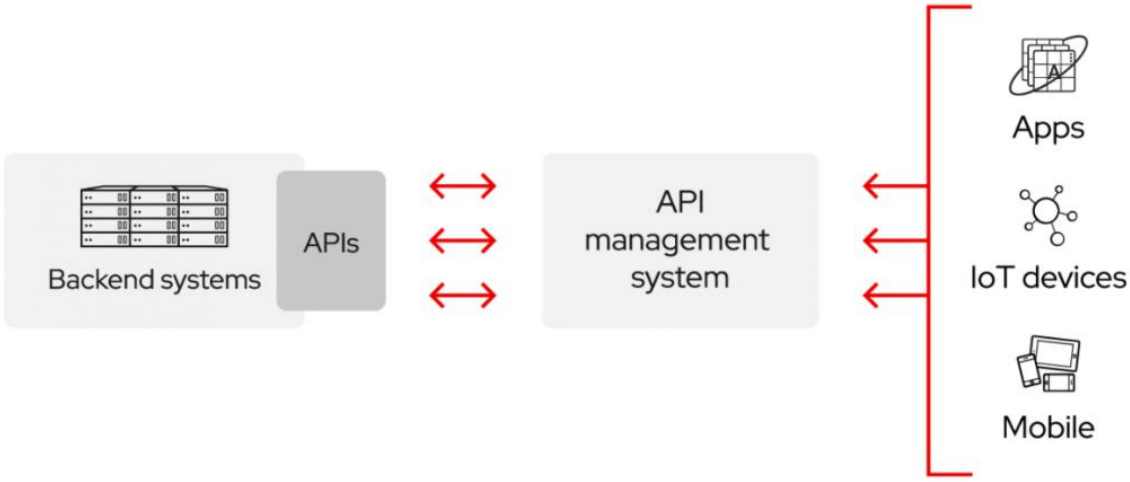
Las API pueden ser de dominio público o privado y se le pueden aplicar diferentes factores de seguridad, permitiendo tener un mayor control sobre estas. Cabe mencionar que muchos servicios empresariales son comunicados a través de API, manteniendo altos estándares de calidad y de seguridad sin comprometer factores de integridad en los datos.

---

[15] DJANGO. Why Django? [En línea]. [2020]. Disponible en: https://www.djangoproject.com/start/overview/

[16] RED HAT. Qué son las API y para qué sirven. [En línea]. Disponible en: https://www.redhat.com/es/topics/api/what-are-application-programming-interfaces

Figura 6. Diagrama modelo de API flujos de datos



Fuente: RED HAT. ¿Qué es una API? [En línea]. Disponible en: https://www.redhat.com/cms/managed-files/styles/wysiwyg_full_width/s3/API-page-graphic.png?itok=5zMemph9>

2.2.6. Google Cloud Platform

Google Cloud Platform es un conjunto de herramientas que abarca computación en la nube, networking, almacenamiento de datos, análisis de datos, machine learning, entre otros. Todos estos contenidos dentro de una *suite* de servicios que funcionan dentro de la infraestructura de Google.

Estos servicios pueden ser conectados mediante API a diferentes programas en muchos lenguajes de programación, permitiendo beneficiarse de los servicios ofrecidos con altos estándares de seguridad y una gran escalabilidad. Gran parte de estos servicios son ofrecidos de manera gratuita hasta cierto tope de operaciones realizadas, una vez superado este tope, inicia un proceso de facturación basado en los servicios usados y en las diferentes regiones donde se operen estos[17].

Entre los principales servicios ofrecidos por Google Cloud Platform se encuentran los siguientes:

[17] GOOGLE CLOUD. Product Launch Stages. [En línea]. [2020]. Disponible en: https://cloud.google.com/products?hl=es#product-launch-stages

- Servicios de *computing*, donde se brindan productos de rango escalable relacionados con procesos de computación como, por ejemplo, el balance automático de máquinas virtuales, puesto en marcha a través de aplicaciones móviles y administración de recursos consumidos, teniendo gran compatibilidad con el lenguaje de programación como Java, Go, PHP o Python, entre otros.
- Servicios de *networking*, los cuales están relacionados con todo el funcionamiento de una red, permitiendo conectar las diferentes máquinas virtuales entre sí o aislarlas según las políticas establecidas por cada empresa. Adicionalmente, garantiza la administración de dominios de servicio y su correspondiente balanceo según el tráfico.
- Servicios de *storage*, que permiten el almacenamiento de los productos en la nube, ofreciendo administración de bases de datos relacionales y no relacionales, garantizando altos estándares de seguridad y rendimiento de la industria, facilitando las administraciones y configuraciones de los diferentes servicios empleados, y gran escalabilidad.
- Servicios de *big data*, posibilitando procesar y consultar altos volúmenes de datos en la nube, obteniendo respuestas de forma rápida y concisa, contando con toda la infraestructura necesaria para el procesamiento de grandes volúmenes de información sin tener que manejar una infraestructura robusta para tal fin. Lo anterior, teniendo una tarifa únicamente por los datos procesados a través de este servicio.
- Servicios de *machine learning*, permite desarrollar patrones y técnicas asociados a la ciencia de computación y a la rama de IA, brindando grandes infraestructuras de aceleración de GPU para el procesamiento de grandes conjuntos de datos, optimizando tiempos en procesos como entrenamiento predictivo, balanceado, escalable y automático.

2.2.7. Google Maps Platform API

Google Maps Platform es la *suite* de servicios ofrecidos por Google, por medio de la cual los desarrolladores interactúan con dos mapas de Google en el desarrollo de aplicaciones móviles y entorno web. Dentro de los servicios ofrecidos por esta *suite* se encuentran la asignación de marcadores dentro de un mapa, asignación de rutas, segmentación de sectores dentro de un mapa coma, entre otros[18].

Google cuenta con los datos de ubicación de más de 100 millones de sitios, a través de los cuales se puede conocer datos como números de teléfono, direcciones,

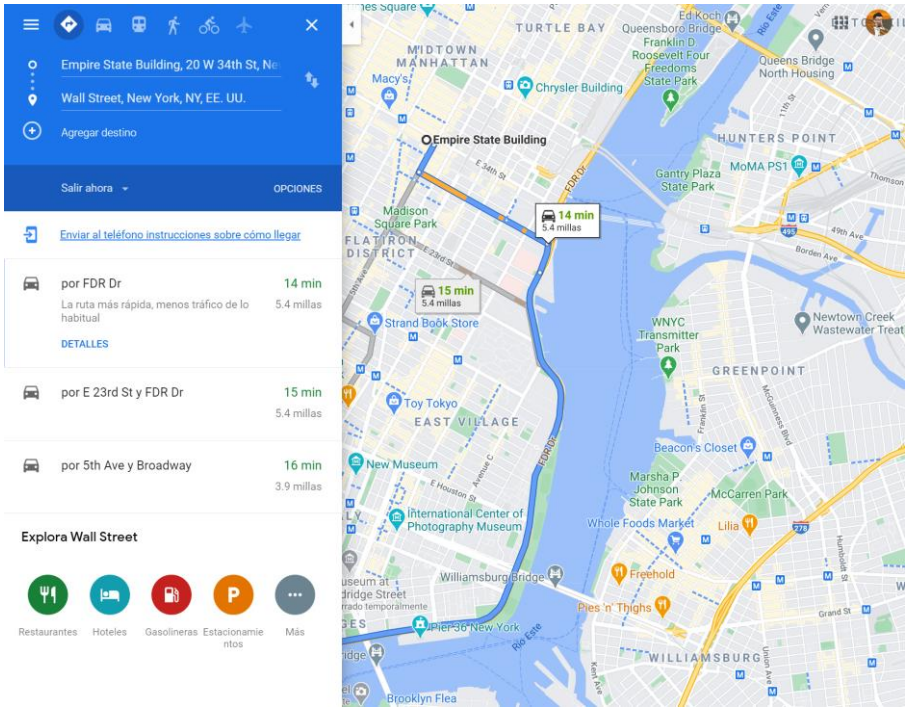
[18] GOOGLE CLOUD. Google Maps Platform. [En línea]. [2020]. Disponible en: https://cloud.google.com/maps-platform/products?hl=es-419

señales de tránsito, tráfico, lugares de interés, entre otros. Todos estos, compatibles con su API para el desarrollo de aplicaciones móviles y web.

2.2.8. API Directions – Google Maps

Este servicio calcula la distancia entre dos o más ubicaciones, por medio de una solicitud http. Adicionalmente, puede volver la ruta más eficiente calculando tiempos de viaje y otros factores como son distancia, número de giros y hora de llegada. Este servicio puede ser aplicable en transporte público, bicicleta, automóvil, o a pie.

Figura 7. Captura de pantalla Google Maps funcionalidad de direcciones

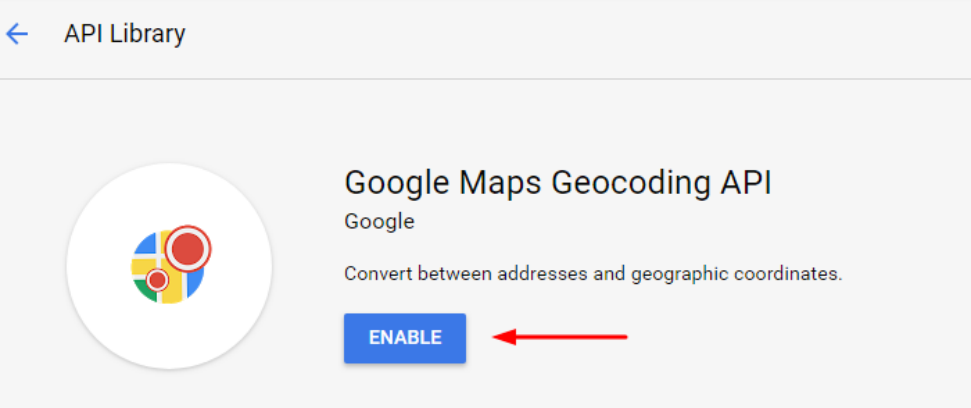


Fuente: elaboración propia

2.2.9. API Geocoding – Google Maps

Es un servicio de Google que posibilita la conversión de direcciones en coordenadas geográficas y viceversa. Resulta útil para visualizar coordenadas geográficas dentro de un mapa legibles para un humano. Se accede a sus servicios a través de una solicitud http y, posteriormente, es posible hacer la visualización a través de diferentes *scripts* de JavaScript, que pueden ser renderizados en una aplicación web o móvil mediante los mapas de Google.

Figura 8. Captura de pantalla activación Geocoding en consola Google



Fuente: elaboración propia

2.2.10. Librería PANDA

Python Data Analysis Library (PANDA) es una librería de lenguaje de programación Python. Es una poderosa librería que posibilita ejecutar un análisis de conjuntos de datos estructurados multidimensionales, de código abierto y de uso gratuito. Asimismo, es posible obtener datos desde diferentes sistemas de archivos como lo son CSV o bases de datos relacionales, creando objetos de filas y columnas llamados *data frames*. Se asemeja mucho a un software de análisis estadístico basado en tablas como lo es Excel, por lo que permite trabajar con listas o diccionarios a través de diferentes funciones para recorrer datos mediante bucles o listas[19].

Por otra parte, posee diversas funciones como, por ejemplo, cargar y guardar datos, principalmente en listas, diccionarios o matrices, para posteriormente ser procesadas como archivos planos o archivos formato JavaScript Object Notation (JSON). Por otro lado, permite una completa inspección de datos, devolviendo diferentes relaciones de medidas entre columnas y filas. También es posible obtener diferentes datos estadísticos como son mínimas, máximas, medianas, promedios entre otros; y filtrar, ordenar y agrupar colecciones de datos de distintas formas para su posterior tratamiento y procesamiento.

Es una librería de fácil uso, siendo compatible con Python 3.5.3 o superiores, lo que garantiza grandes interacciones con otras bibliotecas como lo son NumPy y Matplotlib. Su forma más fácil de instalación es a través de un paquete dentro de un entorno virtual sobre lenguaje de programación Python, compatible con diferentes sistemas operativos como lo son Windows Linux o OS X.

---

[19] PANDAS. User Guide Pandas. [En línea]. [2020]. Disponible en: https://pandas.pydata.org/docs/user_guide/index.html

2.2.11. Librería NumPy

NumPy es una librería desarrollada en Python, para el desarrollo de computación científica. Es utilizada como herramienta estadística para realizar diferentes cálculos avanzados, basados en cálculos de matrices N dimensionales, como lo son la transformada de Fourier, operaciones relacionadas con funciones de álgebra lineal, variaciones de aleatoriedad, entre otras[20]. Aunado a ello, proporciona cálculos con base en estructuras de datos a partir de matrices, las cuales ofrecen beneficios sobre las listas regulares de Python, y cuenta con altos estándares de eficiencia y altos tiempos de lectura de datos para cálculos eficientes en matrices.

2.3. ESTADO DEL ARTE

Para la estructuración y elaboración del estado del arte se consultaron diferentes documentos relacionados con temas de logística y cómo esto se aplica en sistemas de información, considerando las buenas prácticas realizadas por los diferentes desarrolladores y la manera en que se realiza el cálculo de las rutas más cortas en diferentes sistemas de ruteo reconocidos a nivel nacional y mundial.

En el informe del Banco Interamericano de Desarrollo (BID)[21], en el año 2013, se expuso sobre la importancia de la distribución urbana de mercancías, cómo las urbes deben adoptar estrategias para el despliegue de centros logísticos, identificando todos los eslabones de las cadenas de suministro, y cómo estas aportan al desarrollo en las ciudades. Por otro lado, es fundamental tener en cuenta todos los factores ambientales como la contaminación y consumo energético, exhortando a las diferentes empresas a adoptar tecnologías que garanticen la optimización de recursos al máximo, puesto que esto, en primer lugar, se refleja en mayores ganancias, en segundo lugar, permite mitigar los daños causados al medioambiente. Esto último, ya sea dentro de las cadenas de producción o dentro de las cadenas de suministro, las cuales adoptan diferentes medios de transporte, que contribuyen a la destrucción del medioambiente.

Otro aspecto importante que resaltó el BID[22], es que en la última década se ha acrecentado el auge del comercio electrónico, obligando a las indiferentes industrias a adoptar innovaciones en materia tecnológica para una mejor gestión de sus recursos logísticos, elevando los niveles de satisfacción de sus clientes. Todo esto,

---

[20] NUMPY COMMUNITY. User Guide NumPy. [En línea]. [2020]. Disponible en: https://numpy.org/doc/stable/numpy-user.pdf

[21] BANCO INTERAMERICANO DE DESARROLLO (BID). Distribución urbana de mercancías con centros logísticos. Nota técnica. [En línea]. [2013]. Disponible en: /Distribucion_Urbana_de_Mercancias__Estrategias_con_Centros_Logisticos._Nota_Tecnica<http://www.academia.edu/download/55566569/Distribucion_Urbana_de_Mercancias__Estrategias_con_Centros_Logisticos._Nota_Tecnica.pdf>

[22] Ibíd.

acompañado de políticas locales para la adopción de cadenas de suministro óptimas por parte de los gobiernos municipales, locales y nacionales.

Adicionalmente, el Grupo de investigación científica y desarrollo tecnológico[23] de la Universidad Católica de Colombia, manifestó, dentro de su proyecto del modelo de ruta de vehículos, la importancia de las ventanas de tiempo (Vehicle Routing Problem with Time Windows – VRPTW), como sustento teórico que garantiza satisfacer las necesidades del proyecto, mediante la aplicación de modelos en un entorno real. Esto permite reducir la brecha entre la academia y la industria, dado que por medio de la aplicación de un sistema de información se posibilita la gestión de grandes volúmenes de información, lo que facilita métricas en diferentes variables que dan lugar a una planeación táctica basada en esta solución computacional.

Por último, según datos del Departamento Nacional de Planeación (DNP)[24], para el año 202, con la creación del Compes 3982, la logística a nivel nacional transportó alrededor de siete toneladas por vehículo, equivalente a más de 38 millones de viajes para el año 2018. El documento cuenta con un estudio en el cual se estima que se han perdido alrededor de 16 millones de pesos, debido a los diferentes impactos generados por las congestiones de las grandes urbes y las diferentes aglomeraciones del país, especialmente en la ciudad de Bogotá, donde se hizo un especial énfasis en los grandes agentes contaminantes que ocasionan las cadenas de distribución. Lo anterior, debido a que estas basan su gestión en vehículos que expulsan grandes contaminantes al ambiente, en tanto que gran parte del parque automotor se encuentra obsoleto o con daños considerables de antigüedad.

[23] LEÓN, ANDRÉS. Modelo de ruteo de vehículos para la red de distribución de mercancías de un operador logístico en Colombia. Bogotá, D.C.: Universidad Católica de Colombia, 2020.
[24] DEPARTAMENTO NACIONAL DE PLANEACIÓN (DNP). Encuesta nacional logística 2018. [En línea]. [2019]. Disponible en: https://onl.dnp.gov.co/es/Publicaciones/SiteAssets/Paginas/Forms/AllItems/Informe%20de%20resultados%20Encuesta%20Nacional%20Log%C3%ADstica%202018.pdf

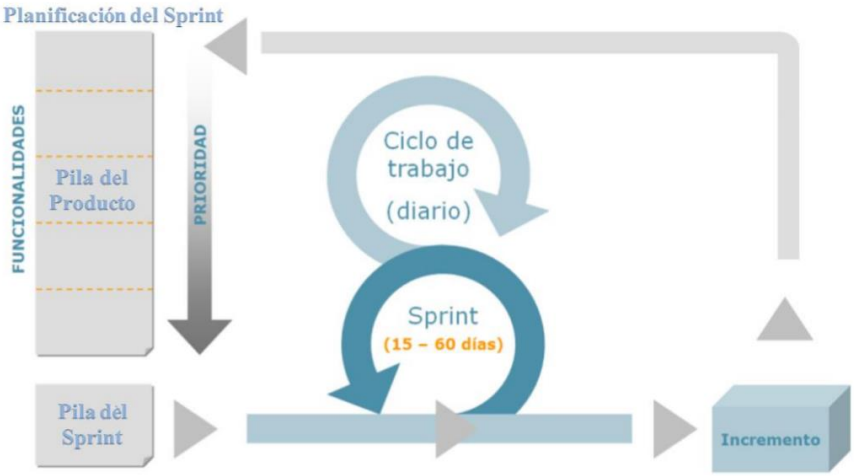
3. METODOLOGÍA

3.1. METODOLOGÍA PARA EL DESARROLLO DEL PROYECTO

Para llevar a cabo el desarrollo del presente proyecto de investigación, se escogió una metodologías ágil de desarrollo de software llamada Scrum, puesto que con esta se pueden tener interacciones entre los diferentes componentes sin que esto implique estar aferrado a cierto proceso o cierta herramienta. Por otra parte, dentro de esta no se contempla una documentación exhaustiva sobre el producto a crear, sino que se hace especial énfasis en el producto final del software. Cabe señalar que, una vez adoptada una metodología, se cuenta con capacidad de respuesta mucho más práctica ante el cambio durante el desarrollo del proyecto, contando así con una flexibilidad beneficiosa para la obtención de resultados finales.

La metodología Scrum se caracteriza por realizar, de forma interactiva, ciclos de construcción de software. En cada ciclo se incorporó una nueva funcionalidad que puede estar sujeta a cambios dentro de muy cortos lapsos. Esta capacidad de cambio permite satisfacer las necesidades del cliente en menor tiempo de construcción, por lo que Scrum basa su metodología en los principios de inspección continua, adaptación, autogestión e innovación.

Figura 9. Estructuración de flujo de datos dentro de la metodología Scrum



Fuente: FUENTES, Manuel y PEÑA, Junio. Desarrollo de un Sistema de Información con tecnologías Web y Móvil para tramitar la inspección de vehículos en una compañía de seguros. Caracas: Universidad Central de Venezuela, 2015.

En cuanto al ciclo de vida del proyecto, por medio de metodología Scrum se pueden identificar diferentes etapas como las que se exponen a continuación:

- La planificación del *sprint*, en donde se establecen las necesidades y el esfuerzo suficiente para lograr la ejecución del trabajo planificado. En esta etapa se evalúan requerimientos y estimaciones basados en las funcionalidades mismas del sistema, y el propietario del producto es responsable y es quien decide sobre este.
- Distribución, revisión y ajustes de los estándares del producto como, por ejemplo, la fase en donde los desarrolladores ajustan los requerimientos mínimos y verifican los estándares para comenzar la fase de *sprint*.
- *Sprint* es la fase donde se realiza el desarrollo de software y los ajustes basados en las reuniones enfocadas a elaborar, revisar y ajustar los diferentes componentes de software previstos para el proyecto. Por lo general, esta fase tiene una duración aproximada de 30 días.
- La revisión del *sprint* se lleva a cabo cuando se prueban los diferentes desarrollos basados en una codificación limpia y documentada, permitiendo una repetición iterativa hasta que el producto esté listo para la fase de cierre.
- En la fase de cierre se permite la depuración y corrección de errores hasta lograr la calidad idónea del producto. Asimismo, se revisan todos los objetivos estipulados en la fase de planeación, para la plena satisfacción del cliente final.

3.2. ROLES EN EL PROCESO SCRUM

Con respecto al desarrollo de software, en el marco de la metodología Scrum se definieron roles de comportamientos y actividades dentro del proyecto, los cuales fueron adoptados entre el estudiante y el tutor de la presente tesis. Lo anterior, validando la funcionalidad de los modelos de clusterización utilizados con el equipo de la heurística seleccionada. De acuerdo con ello, los roles se discriminaron de la siguiente manera:

- **Propietario del producto:** esta persona determina las prioridades dentro del proyecto y debe conocer muy bien el producto final. De igual manera, debe contar con todos los canales de comunicación entre las partes involucradas en el proyecto, con el fin de realizar todas las actividades establecidas para el cumplimiento de los objetivos. Quien estuvo encargado del acceso en el desarrollo del proyecto fue el tutor de la tesis, esto es, el profesor Roger Guzmán.
- **Scrum *manager*:** es la persona encargada de realizar la gestión necesaria para facilitar la ejecución del proyecto y velar por el cumplimiento de las metodologías trazadas. Esta función fue desempeñada por el tutor de la tesis, el profesor Roger Guzmán.

- **Equipo Scrum:** es el principio de la metodología, en donde se centran todos los esfuerzos basados en el desarrollo del software. Para este caso, la persona encargada fue el estudiante Adrián Díaz.
- **Interesados:** las personas interesadas dentro del proyecto son aquellas que tienen un rol de observadores y, a su vez, pueden brindar asesoría sobre el proceso. También pueden estar interesados en cómo promover el futuro del proyecto, en tanto que pueden tener intereses futuros. Para el cumplimiento de este rol se tuvo en cuenta el semillero de investigación MAILab, de la Universidad Católica de Colombia.
- **Usuarios:** dentro de este proyecto, se concibió este rol como uno de los menos considerados, dado que, finalmente, para este alcance solo se tuvo contemplada la elaboración de un prototipo, sin tener centrar la mirada en el usuario final, Solo se tomó un contexto de negocio, al cual pudiera ser aplicada la presente tecnología desarrollada. A pesar de lo anterior, en este rol fueron tomadas en cuenta las opiniones en relación con el funcionamiento del sistema del tutor de tesis, el profesor Roger Guzmán, y el equipo de trabajo del semillero de investigación MAILab de la Universidad Católica de Colombia.

3.3. REUNIONES DE SEGUIMIENTO

Uno de los elementos fundamentales de la metodología Scrum son las reuniones de seguimiento al equipo de trabajo del proyecto, lo que ayuda a generar resultados óptimos, principalmente durante la planificación del *sprint*, donde se contemplan las estimaciones de esfuerzo a nivel de desarrollo, tratando de contemplar posibles eventualidades. A pesar de la disponibilidad ilimitada de muchas de las personas involucradas en el presente estudio, se realizaron reuniones, visualizando el cumplimiento de los objetivos dentro del cronograma diseñado, exponiendo todas las apreciaciones de cambio que pudieran surgir durante estas reuniones.

Al incorporar la metodología Scrum dentro del desarrollo de software, se obtuvo una plena gestión de las expectativas finales del producto, así como del valor del producto final a un nicho de negocio específico. Adicionalmente, fue posible visualizar los resultados anticipados antes de tener por finalizado el proyecto, gracias a la flexibilidad de adaptación que ofrece, dando lugar al cumplimiento de los requisitos para garantizar una mejor comprensión del producto final.

Da igual forma, desde la primera reunión se tuvo en cuenta la mitigación de riesgos, con el fin de que no se materializara durante el desarrollo ni la entrega final del proyecto. Asimismo, se garantizó la calidad del producto, satisfaciendo las diferentes necesidades del cliente final.

4. ANÁLISIS Y DISEÑO DEL SISTEMA INFORMACIÓN

Dentro de la fase de análisis y diseño de un sistema información, fue necesario contemplar las diferentes fases del desarrollo de software.

4.1. REQUERIMIENTOS FUNCIONALES

4.1.1. Descripción general

Este sistema desarrollado tiene la capacidad de realizar la gestión administrativa y operativa del funcionamiento interno y externo de una empresa. Esto, teniendo en cuenta la operabilidad general, esto es, desde la creación de usuarios hasta la visualización de las rutas de acceso de desplazamiento por medio de las API de Google Cloud Platform.

4.2. PERSPECTIVA DEL PRODUCTO

El aplicativo web tiene la capacidad de suministrar la interacción empresa-cliente, la gestión administrativa y trabajador de planta, así como las rutas de desplazamiento. Estas características fueron aplicadas a través de un núcleo principal, desarrollado e implementado en el lenguaje de programación Python y el *frameworks* Django, haciendo uso de interfaces gráficas amigables y de fácil manejo.

4.3. CARACTERÍSTICAS DE LOS USUARIOS

Cuadro 1. Características de los usuarios

Tipo de usuario	Actividad
Administrador	Administrar, crear, monitorear operaciones internas de la empresa.

Tipo de usuario	Actividad
Estándar	Interacción con la plataforma.

Fuente: elaboración propia

5. REQUERIMIENTOS DEL SOFTWARE

5.1. REQUERIMIENTOS FUNCIONALES

5.1.1. Requerimientos funcionales N.° 1

Cuadro 2. Requerimientos funcionales 1

Inicio de sesión			
Identificador:	RF01	Nombre:	Inicio de sesión
Tipo:	Funcional	Requerimiento que lo asocia	Ninguno
Prioridad del desarrollo		Alta	
Entrada:	Correo Contraseña	Salida	Ingreso a la plataforma
Descripción			
Es necesario que el usuario administrador o usuario estándar inicie sesión para poder interactuar en la plataforma.			
Manejo de situaciones			
Si el usuario no se encuentra registrado en la plataforma tendrá que dirigirse con la gestión administrativa para poder realizar el registro correspondiente.			
Responsables	Equipo de administradores.		

Fuente: elaboración propia

5.1.2. Requerimientos funcionales N.° 2

Cuadro 3. Requerimientos funcionales 2

Módulo de registro			
Identificador:	RF02	Nombre:	Registro de usuarios estándar
Tipo:	Funcional	Requerimiento que lo asocia	RF01
Prioridad del desarrollo		Alta	

Entrada:	Nombre, documento de identificación, correo, celular, dirección.	Salida	Registro de usuario exitoso.
Descripción			
Para realizar el registro correcto de usuarios estándar, es necesario contar con los datos de identificación básicos para poseer el control interno del aplicativo.			
Manejo de situaciones			
Si el usuario no suministra la información correctamente o pierde acceso a su cuenta, tendrá que comunicarse con la gestión administrativa o proceder con la recuperación de contraseña por medio de token.			
Responsables	Equipo de administradores.		

Fuente: elaboración propia

5.1.3. Requerimientos funcionales N.° 3

Cuadro 4. Requerimientos funcionales 3

Módulo de registro			
Identificador:	RF03	Nombre:	Registro de usuarios administradores
Tipo:	Funcional	Requerimiento que lo asocia	RF01
Prioridad del desarrollo		Alta	
Entrada:	Nombre, documento de identificación, correo, celular.	Salida	Registro de usuario exitoso
Descripción			
Para realizar el registro correcto de usuarios administradores, es necesario contar con los datos de identificación básicos para poseer el control interno del aplicativo.			
Manejo de situaciones			

Si el usuario no suministra la información correctamente o pierde acceso a su cuenta, tendrá que comunicarse con la gestión administrativa o proceder con la recuperación de contraseña por medio de token.	
Responsables	Equipo de administradores.

Fuente: elaboración propia

5.1.4. Requerimientos funcionales N.° 4

Cuadro 5. Requerimientos funcionales 4

Módulo de registro			
Identificador:	RF04	Nombre:	Registro vehicular
Tipo:	Funcional	Requerimiento que lo asocia	RF01
Prioridad del desarrollo		Alta	
Entrada:	Placa, tipo de vehículo, modelo, año.	Salida	Registro de vehículo exitoso
Descripción			
Para la administración correcta de la operabilidad de la plataforma y para el control general del funcionamiento de este, en donde se enmarcó el rutero operacional, Se requirió realizar el despliegue funcional del registro vehicular.			
Manejo de situaciones			
Si los registros vehiculares no se realizan, no se podrá realizar el cálculo correspondiente de desplazamiento a nivel nacional.			
Responsables	Equipo de administradores.		

Fuente: elaboración propia

5.1.5. Requerimientos funcionales N.° 5

Cuadro 6. Requerimientos funcionales 5

Módulo de Google Cloud Platform			
Identificador:	RF05	Nombre:	Implementación de API de Google

Tipo:	Funcional	Requerimiento que lo asocia	RF01
Prioridad del desarrollo		Alta	
Entrada:	Api Google Cloud	Salida	Vista de Google Maps
Descripción			
Mostrará las rutas de desplazamiento correspondiente.			
Manejo de situaciones			
Si no se cuenta con los accesos a las API de Google, el servicio de navegación y de optimización no será posible.			
Responsables	Equipo de administradores.		

Fuente: elaboración propia

5.1.6. Requerimientos funcionales N.° 6

Cuadro 7. Requerimientos funcionales 6

Módulo de Google Cloud Platform			
Identificador:	RF06	Nombre:	Navegación por medio de API
Tipo:	Funcional	Requerimiento que lo asocia	RF01
Prioridad del desarrollo		Alta	
Entrada:	Dirección de proveedores, dirección usuario estándar.	Salida	Latitud, longitud y rutas de desplazamiento.
Descripción			
Para que los pedidos solicitados por clientes sean correcta y eficientemente entregados, se debe tener acceso a la ruta de desplazamiento, guiándose por Google Cloud Platform.			
Manejo de situaciones			

Si el usuario o el vehículo no se encuentra registrado, no será posible la reasignación de rutas de entrega nivel nacional.	
Responsables	Equipo de administradores.

Fuente: elaboración propia

5.1.7. Requerimientos funcionales N.° 7

Cuadro 8. Requerimientos funcionales 7

Base de datos			
Identificador:	RF07	Nombre:	Implementación DB
Tipo:	Funcional	Requerimiento que lo asocia	Ninguno
Prioridad del desarrollo		Alta	
Entrada:	Parametrización de base de datos.	Salida	DB orientada al manejo de datos JSON.
Descripción			
Se creará una base de datos (DB) relacional para el manejo de datos geoespaciales.			
Manejo de situaciones			
Si la base de datos no relacional cuenta con problemas de conexión, no será posible el almacenamiento de datos.			
Responsables	Equipo de administradores.		

Fuente: elaboración propia

5.1.8. Requerimientos funcionales N.° 8

Cuadro 9. Requerimientos funcionales 8

Base de datos			
Identificador:	RF08	Nombre:	Almacenamiento de datos

Tipo:	Funcional	Requerimiento que lo asocia	Ninguno
Prioridad del desarrollo		Alta	
Entrada:	Latitud, longitud datos de usuarios estándar, datos de usuarios administradores y datos vehiculares.	Salida	Información almacenada correctamente.
Descripción			
Para poder almacenar la información en la base de datos correctamente se tiene que contar con la estructura adecuada para su implementación.			
Manejo de situaciones			
Si no se cuenta con la estructura correcta, no será posible el almacenamiento de la información.			
Responsables	Equipo de administradores.		

Fuente: elaboración propia

5.2. REQUERIMIENTOS NO FUNCIONALES

5.2.1. Características de los usuarios

Cuadro 10. Características usuarios no funcionales

Tipo de usuario	Actividad
Administrador	Administrar, crear, monitorear operaciones internas de la empresa.

Tipo de usuario	Actividad
Estándar	Interacción con la plataforma.

Fuente: elaboración propia

5.2.2. Requerimientos no funcionales N.° 1

Cuadro 11. Requerimientos no funcionales 1

Inicio de sesión

Identificador:	RNF01	Nombre:	Inicio de sesión
Tipo:	No funcional	Requerimiento que lo asocia	Ninguno
Descripción			
El sistema tendrá que denegar el acceso al aplicativo web al personal que no se encuentre registrado o a personal externo.			
Manejo de situaciones			
El acceso permitido solo será posible al personal registrado; en caso contrario, este no tendrá acceso previo.			
Responsables	Equipo de administradores.		

Fuente: elaboración propia

5.2.3. Requerimientos no funcionales N.° 2

Cuadro 12. Requerimientos no funcionales 2

Inicio de sesión			
Identificador:	RNF02	Nombre:	Restricción de privilegios
Tipo:	No funcional	Requerimiento que lo asocia	Ninguno
Descripción			
El sistema tiene que tener la capacidad de gestionar los usuarios por privilegios de acceso a ciertas funciones internas del sistema.			
Manejo de situaciones			
Si no se establecen las configuraciones de derechos de acceso el sistema, el mismo aplicativo toma una configuración básica de acceso por usuario *staff*.			
Responsables	Equipo de administradores.		

Fuente: elaboración propia

5.2.4. Requerimientos no funcionales N.° 3

Cuadro 13. Requerimientos no funcionales 3

Copias de seguridad			
Identificador:	RNF03	Nombre:	Copias de seguridad
Tipo:	No funcional	Requerimiento que lo asocia	Ninguno
Descripción			
Se requiere que el sistema gestione sus correspondientes copias de seguridad en el sistema.			
Manejo de situaciones			
Ninguna.			
Responsables	Equipo de administradores.		

Fuente: elaboración propia

5.2.5. Requerimientos no funcionales N.° 4

Cuadro 14. Requerimientos no funcionales 4

Privilegios			
Identificador:	RNF04	Nombre:	Interoperabilidad
Tipo:	No funcional	Requerimiento que lo asocia	Ninguno
Descripción			
El sistema tiene que ser adaptativo para consultas desde cualquier sistema móvil como tabletas, celulares, etc.			
Manejo de situaciones			
El sistema sin importar desde qué dispositivo se consulte, este siempre será adaptativo por su operación con *framework* Django.			
Responsables	Equipo de administradores.		

Fuente: elaboración propia

5.2.6. Requerimientos no funcionales N.° 5

Cuadro 15. Requerimientos no funcionales 5

Funcionamiento			
Identificador:	RNF05	Nombre:	Facilidad de uso
Tipo:	No funcional	Requerimiento que lo asocia	Ninguno
Descripción			
Las interfaces del usuario de la operabilidad del sistema tienen que ser amigables.			
Manejo de situaciones			
Ninguna.			
Responsables	Equipo de administradores.		

Fuente: elaboración propia

5.3. DESCRIPCIÓN ESPECÍFICA DEL SOFTWARE

5.3.1. Supuestos del proyecto

El Sistema de información para el ruteo en operación logística es una aplicación en que hizo posible el cálculo de la ruta más corta por medio de la web, lo cual se expresó como requerimientos funcionales (RF) y los requerimientos no funcionales (RNF). Por tal motivo, se especificaron estrategias en los que se realizó análisis con el comité de investigación de la Universidad Católica de Colombia. En ella se establecieron los distintos modelos, diagramas y tiempos para los cumplimientos de los objetivos propuestos anteriormente.

5.3.2. Vulnerabilidad de acceso

En toda aplicación web la seguridad es un factor fundamental. Por tal razón, si no se establece un estado inicial de seguridad con sus correspondientes credenciales y las llaves de acceso, como lo es la seguridad a la base de datos, las infiltraciones de seguridad podrían causar la pérdida de información importante para el desarrollo del proyecto.

5.3.3. Vulnerabilidad de código fuente

Por otro lado, si no se hubiera tenido en cuenta la seguridad principal del servidor, la accesibilidad de este podría haber causado la pérdida total del proyecto en su código de fuente.

5.3.4. Acceso a mapas de Google

En caso en que no se realice la implementación adecuada para la comunicación y la integración de Google Cloud Platform en la plataforma de interacción y en la comunicación con las redes neuronales implementadas, puede existir un fallo interno en el que suspenda la utilización del mapa para el cálculo del ruteo correspondiente.

5.3.5. Comunicación de acceso

Si los protocolos de configuración de la red neuronal con la base de datos y la comunicación correspondiente de la aplicación web a la base de datos no se realiza correctamente, puede causar una deficiencia de los datos, funcionamiento o hasta anulación completa de los procesos internos dentro de la aplicación y sus servicios.

5.3.6. Dependencias

Se realizaron las especificaciones de las dependencias encontradas para el funcionamiento principal del proyecto. Con estas, es posible la realización del acople y establecer un funcionamiento específico de la aplicación web “Sistema de información para el ruteo en operación logística”.

5.3.7. Dependencia obligatoria

Inicialmente, se realizó el planteamiento del problema para el desarrollo del proyecto, en el centro investigativo de la Universidad Católica de Colombia. Con apoyo de algunos integrantes de las facultades de ingeniería de sistemas e ingeniería industrial, tras análisis de las tecnologías a implementar y sus soluciones en el mercado, se tomó la decisión de realizar el proyecto bajo tecnologías de la industria 4.0.

Para dicho propósito, se estableció una base principal, en la que se estipuló la creación del proyecto con las siguientes características: implementación de bases de datos geoespaciales, capaces de almacenar la información en tiempo real; utilización de lenguajes de desarrollo, por medio de los cuales se realiza la interacción con el usuario en tiempo real; y, finalmente, capacidad para llevar a cabo la implementación de IA. Para tal fin, teniendo en cuenta estas dependencias, se planteó el proyecto “sistema de información para el ruteo en operación logística”.

5.3.8. Restricciones del proyecto

Se realizó la descripción de las restricciones con las que contó el proyecto a la hora de su implementación, las cuales estuvieron dirigidas a los componentes principales con los que se estructuró el proyecto, a saber: las instancias externas e internas de comunicación.

5.3.9. Restricciones de lectura y escritura

Las lecturas correspondientes se llevaron a cabo de dos maneras: en primer lugar, a través de las funcionalidades implementadas en Python para la lectura de archivos csv y xls, y al mismo tiempo, se hizo la escritura en la base de datos de estos documentos; en segundo lugar, se realizó la lectura de datos con consulta a la base de datos para realizar el mapeo de las rutas establecidas.

5.3.10. Restricciones de acceso

Para las restricciones de acceso se contó con una API KEY capaz de realizar la lectura de información existente en la base de datos en tiempo real sobre los usuarios. Esta permitió la recepción de las solicitudes de acceso de la aplicación, las cuales, por medio de validaciones realizadas, dieron acceso a la utilización del mapa a todos los usuarios estándar.

5.3.11. Riesgos

Cuadro 16. Tabla de riesgos asociados

Nombre de módulo	Riesgos relacionados
Módulo inicio de sesión	Si el usuario correspondiente no cuenta con las credenciales de acceso, o si no son posibles recordarlas, no tendrá acceso al aplicativo. En caso de difundir esta información, el aplicativo será expuesto a personas no autorizadas para su utilización.
Módulo modelo de ruteo	El sistema de información cuenta con un módulo interno codificado, el cual permite realizar el cálculo de las correspondientes rutas de acceso y de desplazamiento, siento esto reflejado en el mapa de Google.
Módulo de acceso administrador	El módulo administrador cuenta con el acceso total a la aplicación, en la que le permitirá crear usuarios internos, los cuales podrán consultar la funcionalidad del mapa.

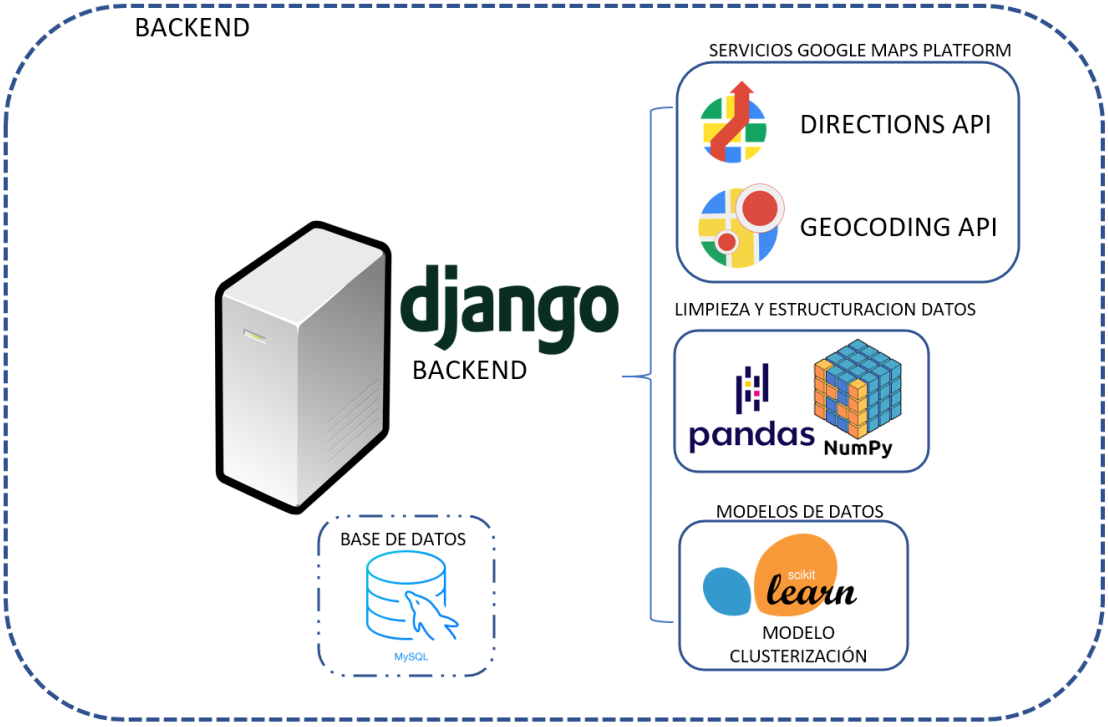
Fuente: elaboración propia

5.3.12. Diseño detallado

A continuación, en el siguiente apartado, se hace una descripción global y detallada de las características generales de la arquitectura de software. Esta se realizó a nivel de detalle, donde es posible observar cuatro niveles específicos y divididos a nivel de usuario.

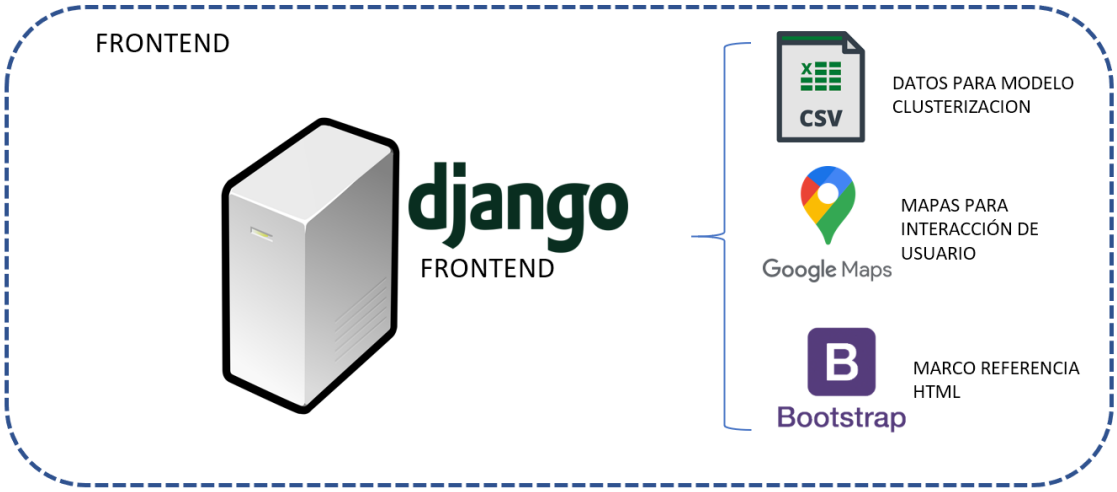
5.3.13. Diseño de arquitectura

Figura 10. Diseño *backend* arquitectura



Fuente: elaboración propia

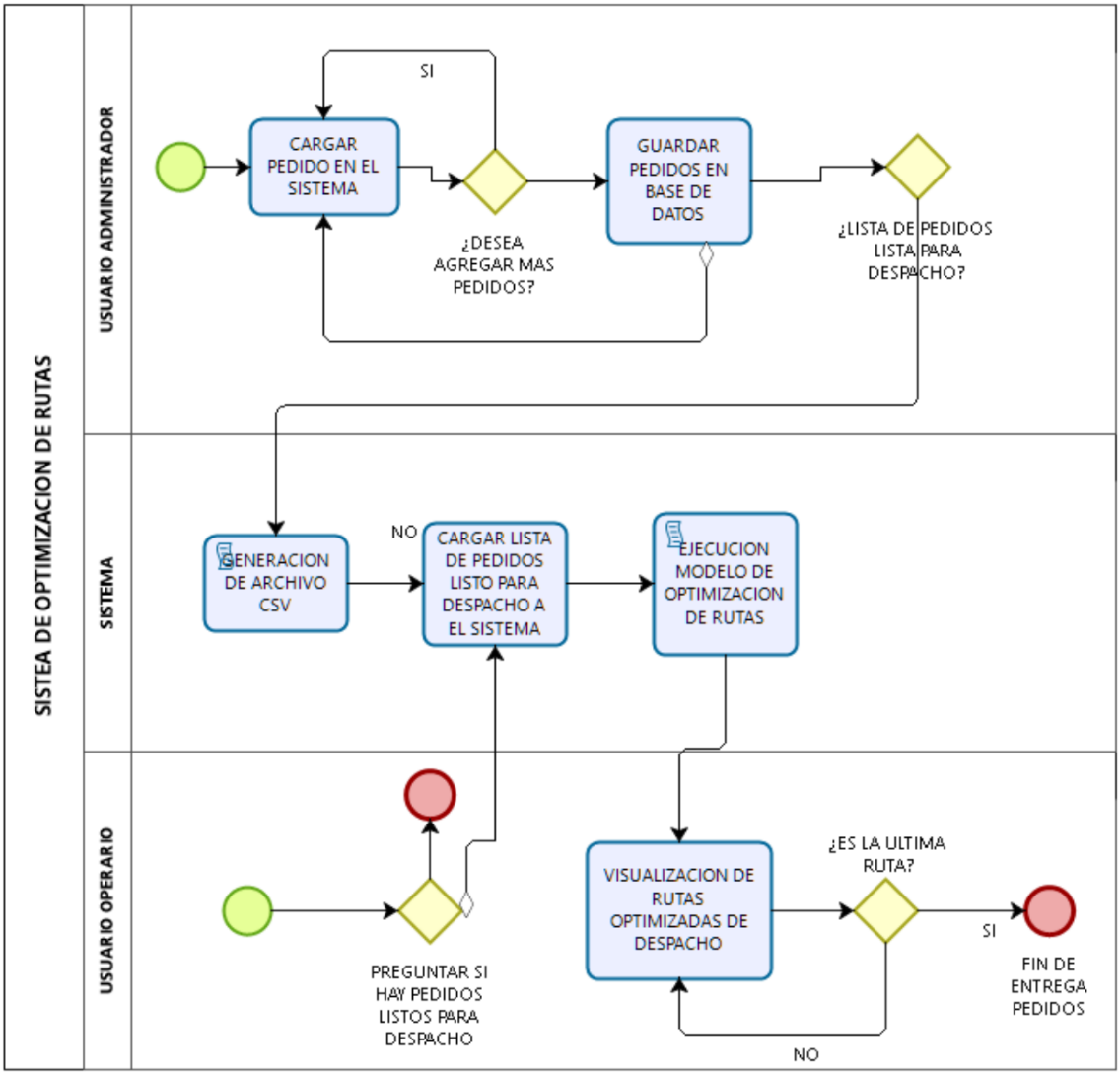
Figura 11. Diseño *frontend* arquitectura



Fuente: elaboración propia

5.3.14. Diagrama de proceso

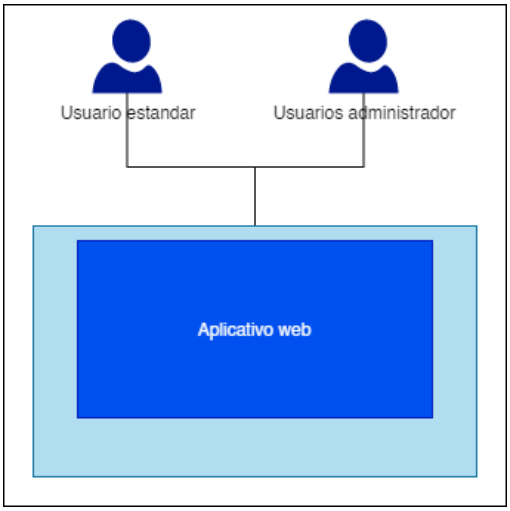
Figura 12. Diagrama de proceso de sistema de Información



Fuente: elaboración propia

5.3.15. Diagrama del contexto del sistema

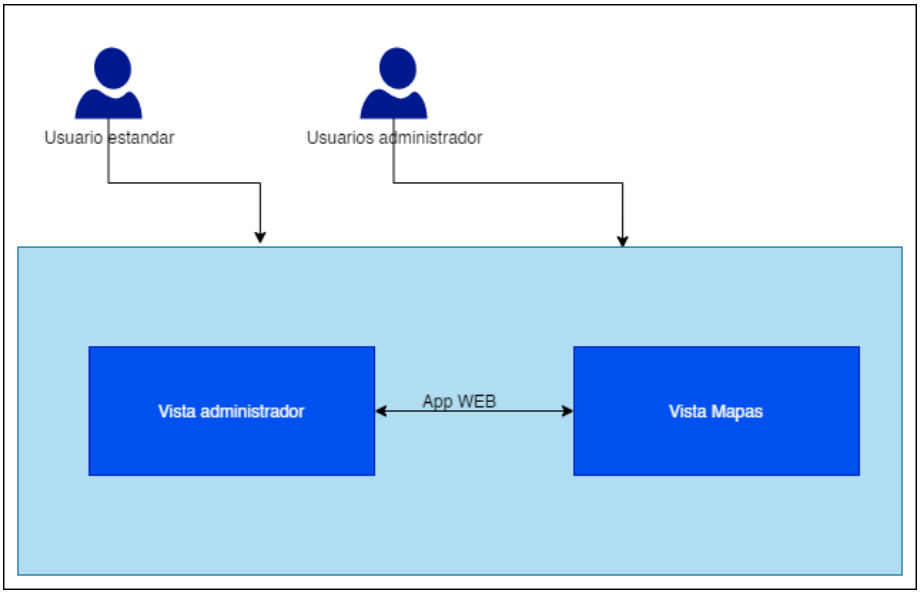
Figura 13. Diagrama del contexto



Fuente: elaboración propia

5.3.16. Diagrama de contenedores

Figura 14. Diagrama de contenedores



Fuente: elaboración propia

5.4. DESARROLLO DEL SOFTWARE

Ya en la fase de desarrollo de software, dentro del editor de código Visual Studio Code, se realizó la estructuración y la codificación de las siguientes estructuras.

5.4.1. Modelos de datos

Al estructurar los modelos de datos, los cuales van a soportar todos los flujos de información dentro del sistema de información, estos fueron acoplados al levantamiento de requerimientos realizados.

- Modelo ciudad (Django)

Figura 15. Modelo usuario

```python
# Ciudades designadas para registros
class Ciudad(models.Model):

    nombre = models.CharField(
        verbose_name='Nombre de la ciudad',
        help_text='Indique el nombre de la ciudad a registrar',
        max_length=50,
        blank=True
    )

    REQUIRED_FIELDS = [
        'nombre'
    ]

    class Meta:
        verbose_name = 'Ciudad'
        verbose_name_plural = 'Ciudades'

    def __str__(self):
        return self.nombre
```

Fuente: elaboración propia

- Modelo usuario (Django)

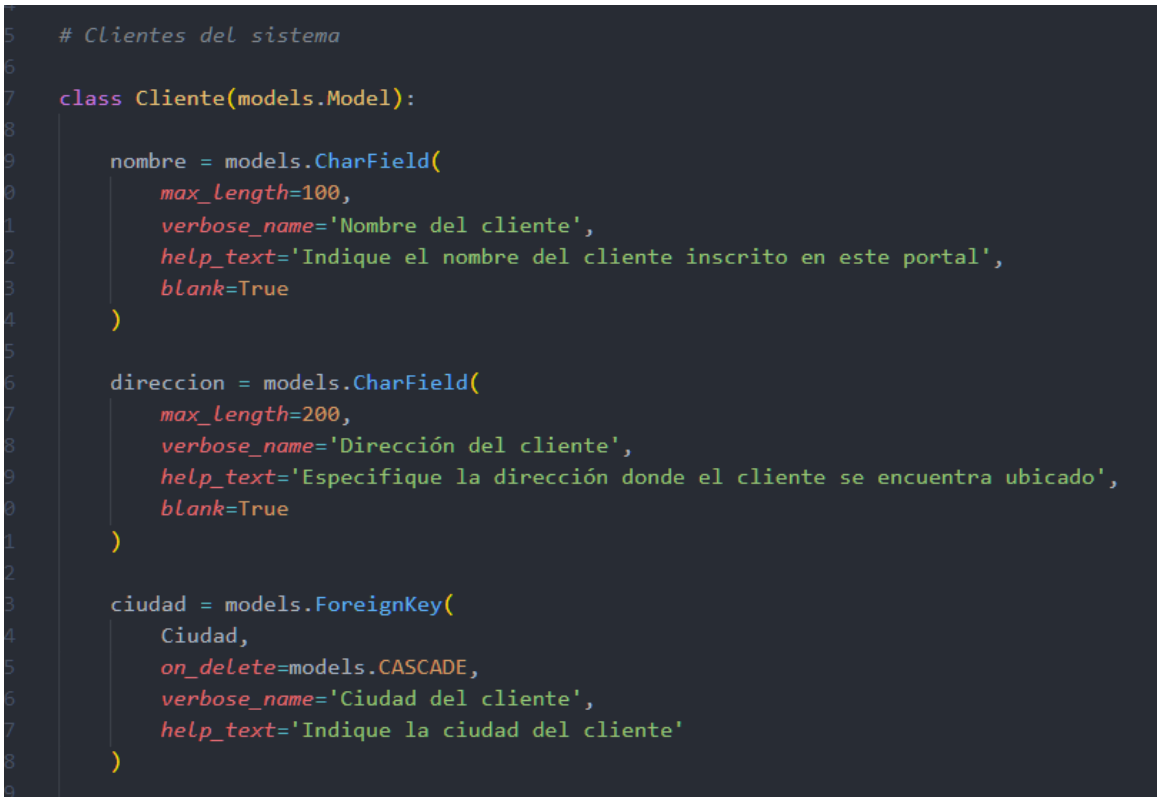
Figura 16. Modelo usuario

```python
class Users(AbstractBaseUser):

    email = models.EmailField(
        verbose_name='Correo electrónico',
        help_text='Ingrese el correo electrónico de su cuenta',
        unique=True
    )
    first_name = models.CharField(
        verbose_name='Nombre',
        help_text='Indique su nombre',
        max_length=100
    )
    last_name = models.CharField(
        verbose_name='Apellidos',
        help_text='Indique sus apellidos',
        max_length=100
    )
    cedula = models.CharField(
        verbose_name='Número de cédula',
        help_text='Indique su número de cédula o licencia de conducir para su identificación en el sis
        max_length=10,
        unique=True,
        validators=[only_digits],
        default=random_values(1111111111, 9999999999)
    )
    licencia = models.CharField(
        verbose_name='Número de licencia de conducción',
        help_text='Indique su número de licencia de conducir para su identificación en el sistema',
        max_length=15,
        unique=True,
        validators=[only_digits],
        default=random_values(111111111111111, 999999999999999)
    )

    # atributos adicionales para el usuario

    active = models.BooleanField(
        verbose_name='¿Cuenta activa?',
        help_text='Especifique el estado activo de este usuario en el portal',
        default=True
    )
    staff = models.BooleanField(
        verbose_name='¿Personal del sitio?',
        help_text='Clarifique si este usuario hace parte del staff del portal',
        default=True
    )
    admin = models.BooleanField(
        verbose_name='¿Administrador del portal?',
        help_text='Indique si este usuario tiene privilegios de superusuario en el portal',
        default=False
    )
```

Fuente: elaboración propia

- Modelo clientes (Django)

Figura 17. Modelo clientes



```python
# Clientes del sistema

class Cliente(models.Model):

    nombre = models.CharField(
        max_length=100,
        verbose_name='Nombre del cliente',
        help_text='Indique el nombre del cliente inscrito en este portal',
        blank=True
    )

    direccion = models.CharField(
        max_length=200,
        verbose_name='Dirección del cliente',
        help_text='Especifique la dirección donde el cliente se encuentra ubicado',
        blank=True
    )

    ciudad = models.ForeignKey(
        Ciudad,
        on_delete=models.CASCADE,
        verbose_name='Ciudad del cliente',
        help_text='Indique la ciudad del cliente'
    )
```

Fuente: elaboración propia

- Modelo pedido cliente (Django)

Figura 18. Modelo pedido cliente

```python
# Pedido del cliente
class Pedido(models.Model):

    cliente = models.ForeignKey(
        Cliente,
        on_delete=models.CASCADE,
        verbose_name='Cliente dueño del pedido',
        help_text='Indique quien es el cliente propietario de este pedido'
    )

    cantidad_dummies = models.IntegerField(
        verbose_name='Cantidad de dummies',
        help_text='Indique la cantidad de dummies para este pedido',
        default=0,
        null=True
    )

    cantidad_giftcards = models.IntegerField(
        verbose_name='Cantidad de gift cards',
        help_text='Indique la cantidad de gift cards para este pedido',
        default=0,
        null=True
    )

    cantidad_handsets = models.IntegerField(
        verbose_name='Cantidad de handset',
        help_text='Indique la cantidad de handsets cards para este pedido',
        default=0,
        null=True
    )

    cantidad_modems = models.IntegerField(
        verbose_name='Cantidad de modems',
        help_text='Indique la cantidad de modems para este pedido',
        default=0,
        null=True
    )

    cantidad_packs = models.IntegerField(
        verbose_name='Cantidad de packs',
        help_text='Indique la cantidad de packs para este pedido',
        default=0,
        null=True
    )

    cantidad_prepaid = models.IntegerField(
        verbose_name='Cantidad de prepaid cards',
        help_text='Indique la cantidad de prepaid cards para este pedido',
        default=0,
        null=True
    )

    cantidad_simcards = models.IntegerField(
        verbose_name='Cantidad de sim cards',
        help_text='Indique la cantidad de sim cards para este pedido',
        default=0,
        null=True
    )

    cantidad_simcards_prep = models.IntegerField(
        verbose_name='Cantidad de sim cards prepagas',
        help_text='Indique la cantidad de sim cards prepagas para este pedido',
        default=0,
        null=True
    )
```

Fuente: elaboración propia

- Modelo despacho del producto (Django)

Figura 19. Modelo Despacho del Producto

```python
# Despacho del producto

class Despacho(models.Model):

    pedido = models.ForeignKey(
        Pedido,
        on_delete=models.SET_NULL,
        verbose_name='Pedido a despachar',
        help_text='Asigne el pedido que será despachado',
        null=True
    )

    vehiculo = models.ForeignKey(
        Vehiculo,
        on_delete=models.SET_NULL,
        verbose_name='Vehículo asignado',
        help_text='Indique cual será el vehículo destinado para realizar el despacho asignado',
        null=True
    )

    si_min = models.IntegerField(
        verbose_name='Tiempo de servicio',
        help_text='Es decir, vehículo llega a la dirección, se estaciona, entonces es el tiempo que se demora el '
                  'conductor o el ayudante en buscar el pedido en el vehículo y entregárselo al cliente',
        default=0
    )

    ai_min = models.IntegerField(
        verbose_name='Ventana de inicio',
        help_text='Es decir, el vehículo puede llegar  después del minuto x a entregar el pedido, en ese caso '
                  'particular fueron tomadas según la normativa y restricciones según las zonas de la ciudad para el '
                  'movimiento de los vehículos de carga, siendo las horas valle',
        default=0
    )

    bi_min = models.IntegerField(
        verbose_name='Ventana de fin',
        help_text='Es decir, es el tiempo máximo en el que puede llegar el vehículo para entregar el pedido',
        default=0
    )

    fecha_despacho = models.DateField(
        auto_now_add=True,
        verbose_name='Fecha de registro del despacho',
        help_text='Fecha autogenerada para el despacho el día que se registre'
    )
```

Fuente: realización propia

- Modelo vehículos (Django)

Figura 20. Modelo vehículos

```python
# Vehículos a usar en el sistema

class Vehiculo(models.Model):

    conductor = models.OneToOneField(
        Users,
        on_delete=models.SET_NULL,
        verbose_name='Conductor del vehículo',
        help_text='Indique el conductor asignado para este vehículo',
        null=True,
    )

    ciudad = models.ForeignKey(
        Ciudad,
        on_delete=models.SET_NULL,
        verbose_name='Ciudad donde se ubica',
        help_text='Indique en que ciudad se encuentra situado el vehículo a usar en el sistema.',
        null=True
    )

    direccion = models.CharField(
        max_length=200,
        verbose_name='Dirección de la ubicación del vehículo',
        help_text='Especifique la dirección donde el vehículo se encuentra ubicado',
        default='Calle 22 # 56-24'
    )

    capacidad = models.IntegerField(
        verbose_name='Capacidad de carga (en kg)',
        help_text='Estipule la maxima capacidad de carga en kilogramos para este vehiculo',
        default=5000
    )

    placa = models.CharField(
        verbose_name='Placa del vehículo',
        help_text='Ingrese la placa del vehículo a registrar para su identificación correcta',
        max_length=6,
        unique=True
    )
```

Fuente: elaboración propia

5.4.2. Integración modelo de optimización

Para la ejecución de este proyecto, se optó por un modelo que implementado en el sistema de información basado en una heurística desarrollada dentro del grupo de investigación científica y desarrollo tecnológico Jóvenes Investigadores de la Universidad Católica de Colombia. Dicha implementación fue llevada a cabo por el ingeniero Andrés Felipe León Villalba, integrante del grupo de investigación, a través de su proyecto titulado “Modelo de ruteo de vehículos para la red de distribución de mercancías de un operador logístico en Colombia”.

Este fue ajustado dentro del archivo views.py, el cual permite la interacción entre la interfaz web y los diferentes modelos de datos gestionados directamente por el *framework* Django.

Figura 21. Fragmento archivo views.py

views.py

mainpage_routes > views.py > ...

```python
import math
import time
from datetime import date

import numpy as np
import pandas as pd
from django.contrib import messages
from django.contrib.auth.decorators import login_required
from django.shortcuts import render, redirect
from sklearn.cluster import KMeans, OPTICS
from sklearn.neighbors import DistanceMetric
from sklearn.preprocessing import normalize, StandardScaler

from .models import *

# Create your views here.

csv_filename = None
xlx_filename = None
ruta_waypts = []


def index(request):
    if request.user.is_authenticated:
        return redirect('routes/')
    else:
        return redirect('login/')

```

Fuente: elaboración propia

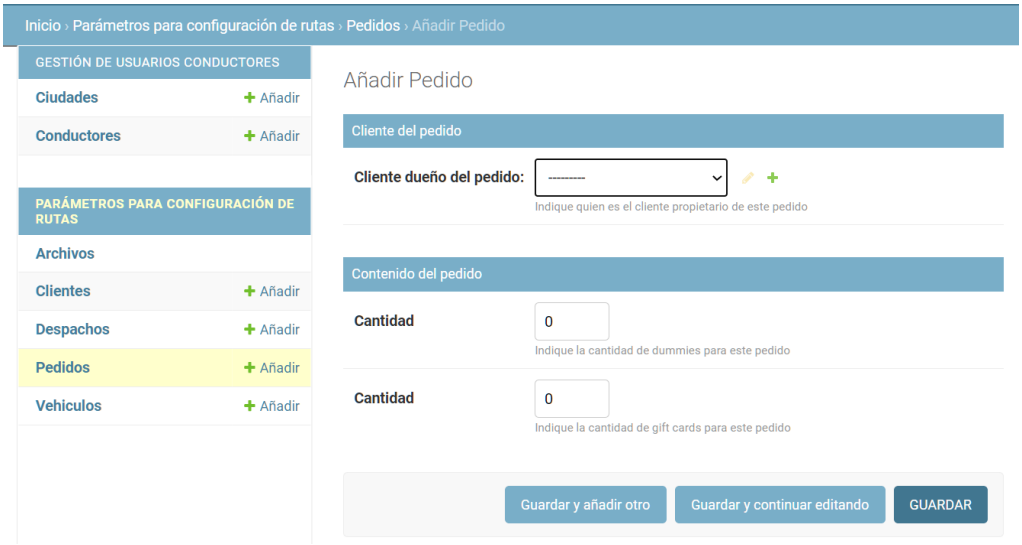
5.4.3. Desarrollo de flujo de datos

Los datos fueron gestionados de acuerdo con la estructura de datos anteriormente expuesta, dentro del administrador de Django. Dichos datos se alimentan de dos maneras.

Por una parte, se puede realizar la gestión de datos directamente desde el administrador del sistema, puesto que en este se pueden obtener todos los pedidos a partir de un formulario dentro de este módulo; y una vez consolidados todos los

pedidos, se pueden programar despachos considerando la cantidad de pedidos con la que se cuente.

Figura 22. Captura de pantalla de la gestión de pedidos desde administración



Fuente: elaboración propia

Por otro lado, los datos se pueden obtener a partir de un archivo CSV que cuente con toda la estructura predefinida para el modelo de datos establecido dentro del administrador. Este archivo puede ser cargado a través del módulo de despachos a través de un archivo CSV separado por comas. En ese sentido, si el archivo cumple con todos los requisitos de estructura y de datos, será cargado satisfactoriamente a la lista de despachos.

Figura 23. Captura de pantalla del módulo importación de archivo CSV

Fuente: elaboración propia

5.4.4. Integración con Google Maps API

La integración con la API de Google Maps se realizó con lenguaje de programación JavaScript, el cual tiene comunicación directa con el navegador y permite el despliegue de los mapas en este. Para ello, se gestionaron las diferentes funciones para la interacción, posibilitando así la geocodificación de los puntos, los cuales han sido interpretados como rutas cortas a partir del flujo de datos ofrecido. Estos puntos fueron suministrados con coordenadas de localización, los cual se representan como puntos en el mapa para su completa visualización por el usuario final.

- Función de geolocalización JavaScript

Figura 24. Captura de pantalla función de geolocalización JavaScript

```
function getPosition() {
  if(navigator.geolocation) {
    navigator.geolocation.getCurrentPosition(function(position) {
      originMode = true;
      pos = {
        lat: position.coords.latitude,
        lng: position.coords.longitude
      };
      var result;
      geocoder.geocode({'location': pos}, function(results, status) {
        if (status === 'OK') {
          if (results[0]) {   // with this conditional we can do get actual direction through the coordinates obtained by geolocating
            dir = results[0].formatted_address;
            infoWindow.setPosition(pos);
            infoWindow.setContent(dir);
            infoWindow.open(map);
            map.setCenter(pos);          // later we print that direction in our "infoWindow" and scopes the map to the geolocated place
            map.setZoom(15);
          } else {
            window.alert('Dirección desconocida');
          }
        } else {
          window.alert('No pudimos tomar tu ubicación, trata de ingresar una dirección en su lugar');
        }
      });
      start = pos;  // we save the geolocated start point value into a temp variable to prevent errors in placeRouteParameters()
      temp = start;
      document.getElementById('start').value = 'Tu ubicación actual';
    });
  } else {
    originMode = false; // if geolocation request is denied we just set start point by text
    setByText();
  }
}
```

Fuente: elaboración propia

- Función de rutas en Google Maps JavaScript

Figura 25. Función de rutas en Google Maps JavaScript

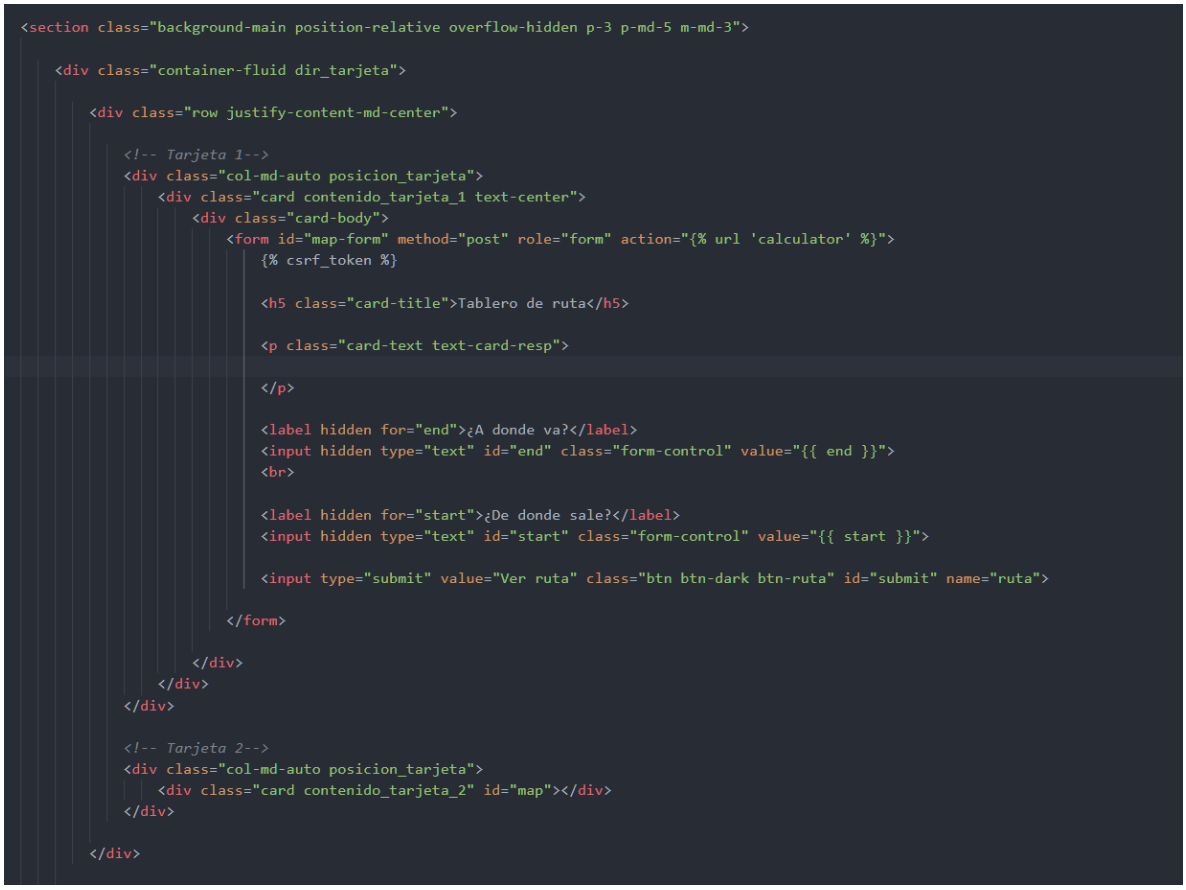
```
function placeRouteParameters(){
  console.log("Estamos aqui")
    console.log(originMode)

  if (originMode === true){
    if ('{{start}}'==='Tu ubicación actual'){
      end = '{{end}}';
      calculateAndDisplayRoute(directionsService, directionsRenderer);
      originMode = false;
    }
  }else{
    setByText();
    if (originMode === false){
      console.log('{{start}}')
      if ('{{start}}'!== 'Tu ubicación actual'){

        console.log("estamos aqui parte 3")
        calculateAndDisplayRoute(directionsService, directionsRenderer);
        originMode = false;
      }else{
        console.log('aqui estamos parte 4')
        console.log('{{end}}')
        start = temp;
        end = '{{end}}';
        calculateAndDisplayRoute(directionsService, directionsRenderer);
        originMode = false;
      }
    }
  }

}
```

Fuente: elaboración propia

5.4.5. Interfaz gráfica (HTML)

Para el desarrollo de la interfaz gráfica se tuvo en cuenta el uso del marco de trabajo Bootstrap, el cual, a partir de simples clases de HTML implementadas, facilitó el desarrollo de los diferentes formularios, así como la interacción responsiva de la interfaz web de usuario.

Figura 26. Captura de pantalla fragmento código HTML Bootstrap



```
<section class="background-main position-relative overflow-hidden p-3 p-md-5 m-md-3">

    <div class="container-fluid dir_tarjeta">

        <div class="row justify-content-md-center">

            <!-- Tarjeta 1-->
            <div class="col-md-auto posicion_tarjeta">
                <div class="card contenido_tarjeta_1 text-center">
                    <div class="card-body">
                        <form id="map-form" method="post" role="form" action="{% url 'calculator' %}">
                            {% csrf_token %}

                            <h5 class="card-title">Tablero de ruta</h5>

                            <p class="card-text text-card-resp">

                            </p>

                            <label hidden for="end">¿A donde va?</label>
                            <input hidden type="text" id="end" class="form-control" value="{{ end }}">
                            <br>

                            <label hidden for="start">¿De donde sale?</label>
                            <input hidden type="text" id="start" class="form-control" value="{{ start }}">

                            <input type="submit" value="Ver ruta" class="btn btn-dark btn-ruta" id="submit" name="ruta">

                        </form>

                    </div>
                </div>
            </div>

            <!-- Tarjeta 2-->
            <div class="col-md-auto posicion_tarjeta">
                <div class="card contenido_tarjeta_2" id="map"></div>
            </div>

        </div>
```

Fuente: elaboración propia

6. PRUEBAS DE SOFTWARE

6.1. ALCANCE DE LAS PRUEBAS

6.1.1. Pruebas unitarias

El objetivo de las pruebas consistió en validar los diferentes módulos presentes en el proyecto.

- Módulo de inteligencia artificial.
- Módulo de bases de datos
- Módulo de interfaz web.

6.1.2. Módulo de inteligencia artificial

Herramienta usada: Python, NumPy, Matplotlib, Sklearn.

Cuadro 17. Listado de pruebas módulo inteligencia artificial

N.°	Nombre de la prueba	Descripción
1	Lectura de los datos SQLite3	Procedimiento para la obtención de datos para el modelo de clusterización.
2	Procesamiento de datos	Procesamiento de datos dentro del modelo de clusterización.
3	Envío de datos procesados	Los datos fueron entregados a la interfaz web para ser visualizados.

Fuente: elaboración propia

6.1.3. Módulo de base de datos

Herramienta usada: MySQL (base de datos)

Cuadro 18. Listado de pruebas bases de datos

N.°	Nombre de la prueba	Descripción
1	Creación de datos de prueba	Creación de datos de prueba para el posterior procesamiento.
2	Validación de integridad de los datos	Realización de la validación de la integridad

		de datos, con el fin de que no se presenten errores al momento de la lectura de este.
3	Verificación de servicio para lectura de datos	Obtener satisfactoriamente los datos a partir del servicio emitido por Firebase.

Fuente: elaboración propia

6.1.4. Módulo de interfaz web

Herramienta usada: Django (*framework* de aplicaciones)

Cuadro 19. Listado de pruebas de interfaz web

N.°	Nombre de la prueba	Descripción
1	Validación de formulario de inicio de sesión	Lograr tener un inicio de sesión al sistema de ruteo cargando datos de usuario desde la base de datos.
2	Integración Django - API Google Maps	Validar la integración de los mapas de Google Maps directamente en la interfaz de Django.
3	Validación de trazado de ruta ideal	Verificar el trazado de la ruta ideal, de acuerdo con la arquitectura fijada para el proyecto y el modelo de clusterización.

Fuente: elaboración propia

Las pruebas se aplicaron para la validación de integración de los diferentes módulos presentes en este proyecto.

- Módulo de clusterización.
- Módulo de bases de datos.
- Módulo de interfaz web.

6.1.5. Integraciones

Cuadro 20. Listado de pruebas de integración

N.°	Nombre de la prueba	Descripción
1	Integración en Web Service: base de datos -> modelo de clusterización	Verificación de la integración.
2	Integración en Web Service: modelo de clusterización -> front web	Verificación de la integración.

Fuente: elaboración propia

6.2. ENFOQUE DE PRUEBAS (ESTRATEGIA)

Por otra parte, se optó por las funcionales, las cuales permiten la verificación de los propósitos específicos para cada módulo, teniendo en cuenta el flujo de datos y los valores esperados por este. Asimismo, a través de estas, es posible identificar factores no tenidos en cuenta durante el proceso de diseño y desarrollo del proyecto.

6.3. CRITERIOS DE ACEPTACIÓN O RECHAZO

6.3.1. Modelo de clusterización

Cuadro 21. Criterios de aceptación para el modelo de clusterización

N.°	Nombre de la prueba	Criterio de aceptación	Criterio de rechazo	%
1	Lectura de los datos a partir de archivo plano.	Lectura del 100 % de los datos suministrados a partir del archivo plano.	Lectura del 99 % de los datos suministrados a partir del archivo plano.	20 %
2	Procesamiento de datos.	Procesamiento del 100 % de los datos.	Procesamiento de menos del 99 % de los datos.	40 %
3	Envío de datos procesados.	Envío del 100 % de los datos.	Envío de menos del 99 % de los datos.	40 %

Fuente: elaboración propia

6.3.2. Módulo de base de datos

Cuadro 22. Criterios de aceptación para el modelo de base de datos

N.°	Nombre de la prueba	Criterio de aceptación	Criterio de rechazo	%
1	Creación de datos de prueba.	Contar la misma cantidad de datos que los creados.	No concordar la cantidad de datos creados con los alojados.	25 %
2	Validación de integridad de los datos.	Cumplimiento de los estándares requeridos para el proyecto.	No cumplimiento de los estándares requeridos para el proyecto.	25 %
3	Verificación de servicio para lectura de datos.	Conexión satisfactoria con el servicio de lectura de datos.	Conexión fallida con el servicio de lectura de datos.	50 %

Fuente: elaboración propia

6.3.3. Módulo de interfaz web

Cuadro 23. Criterios de aceptación del modelo de interfaz web

N.°	Nombre de la prueba	Criterio de aceptación	Criterio de rechazo	%
1	Validación de formulario de inicio de sesión.	Ingreso al sistema satisfactorio.	Ingreso al sistema fallido.	20 %
2	Integración Django - API Google Maps.	Visualización de la interfaz de Google Maps dentro de un *frame* en Django.	No lograr visualizar la interfaz de Google Maps dentro de un *frame* de Django	40 %
3	Validación de trazado de ruta ideal.	Visualizar el trazado de la ruta ideal de acuerdo con la arquitectura fijada para el proyecto basado en modelo de clusterización.	No visualizar el trazado de la ruta ideal según la arquitectura fijada para el proyecto basado en modelo de clusterización.	40 %

Fuente: elaboración propia

6.3.4. Pruebas de integración

Cuadro 24. Criterios de aceptación para el modelo de integración

N.°	Nombre de la prueba	Criterio de aceptación	Criterio de rechazo	%
1	Integración en Web Service: base de datos -> modelo de clusterización.	Completa integración de la base de datos y con el modelo de clusterización.	Incompleta integración de la base de datos con el modelo de clusterización.	50 %
2	Integración en Web Service: modelo de clusterización -> front web.	Completa integración del modelo de clusterización con la interfaz web.	Incompleta integración del modelo de clusterización con la interfaz web.	50 %

Fuente: elaboración propia

6.4. RESULTADOS DE LAS PRUEBAS

6.4.1. Módulo de clusterización

- Lectura de base de datos (MySQL)

Cuadro 25. Ejecución prueba lectura base de datos

Nombre del módulo	Módulo de clusterización.
Nombre de la prueba	Lectura de datos de MySQL.
Objetivo de la prueba	Obtener datos de la base de datos MySQL para ser procesada.
Complejidad	Media.
Perfil/Rol/Usuario	Administrador.
Tipo de navegador	No aplica.
Estado de la prueba	Exitosa.

Fuente: elaboración propia

Cuadro 26. Descripción prueba base de datos

Descripción de la prueba		
Flujo de prueba	**Descripción**	**Observaciones**
Conexión a la base de datos.	Declaración de la función llamada a la base de datos MySQL.	Se realiza conexión compatible con Django.
Validación de datos existentes.	Guardar en memoria los valores arrojados por base de datos.	Los datos son almacenados.

Total, casos de prueba exitosos	1
Total, casos de prueba no exitosos	1
Total, casos de prueba cancelados	0
Total, casos de prueba ejecutados	2

N.°	Nombre de la prueba	Criterio de aceptación	Aprobado
1	Lectura de los datos de MySQL.	Lectura del 100 % de los datos alojados en la base de datos MySQL.	Sí

Fuente: elaboración propia

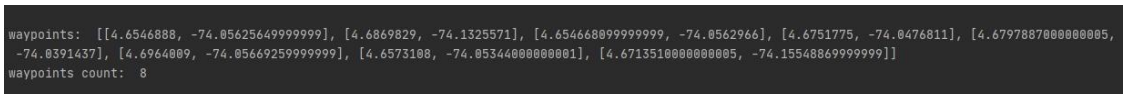
Procesamiento de datos dentro del algoritmo de clusterización

Cuadro 27. Ejecución prueba modelo clusterización

Nombre de módulo	Módulo de clusterización.
Nombre de la prueba	Procesamiento de datos dentro de algoritmo de clusterización.
Objetivo de la prueba	Procesamiento de datos dentro del algoritmo.
Complejidad	Media.
Perfil/Rol/Usuario	Administrador.
Tipo de navegador	No aplica.
Estado de la prueba	Exitosa.

Fuente: elaboración propia

Cuadro 28. Descripción prueba modelo clusterización

Descripción de la Prueba		
Flujo de prueba	Descripción	Observaciones
Lectura de datos	El algoritmo de Clusterización hace lectura de los datos.	Una vez obtenidos los datos, el algoritmo está listo para procesar los datos
Procesamiento de datos	El algoritmo de Clusterización hace el procesamiento de los datos.	Se usan librerías Python para el procesamiento de datos.
Resultado de datos	Luego del procesado el algoritmo arroja los resultados esperados.	Los resultados esperados son los puntos de coordenadas esperadas.
Captura de pantalla  Generación de datos por módulo de clusterización		

Total, casos de prueba exitosos	5
Total, casos de prueba no exitosos	1
Total, casos de prueba cancelados	0
Total, casos de prueba ejecutados	6

N.º	Nombre de la prueba	Criterio de aceptación	Aprobado
1	Procesamiento de datos.	Procesamiento del 100 % de los datos .	Sí

Fuente: elaboración propia

- Entrega de los datos procesados

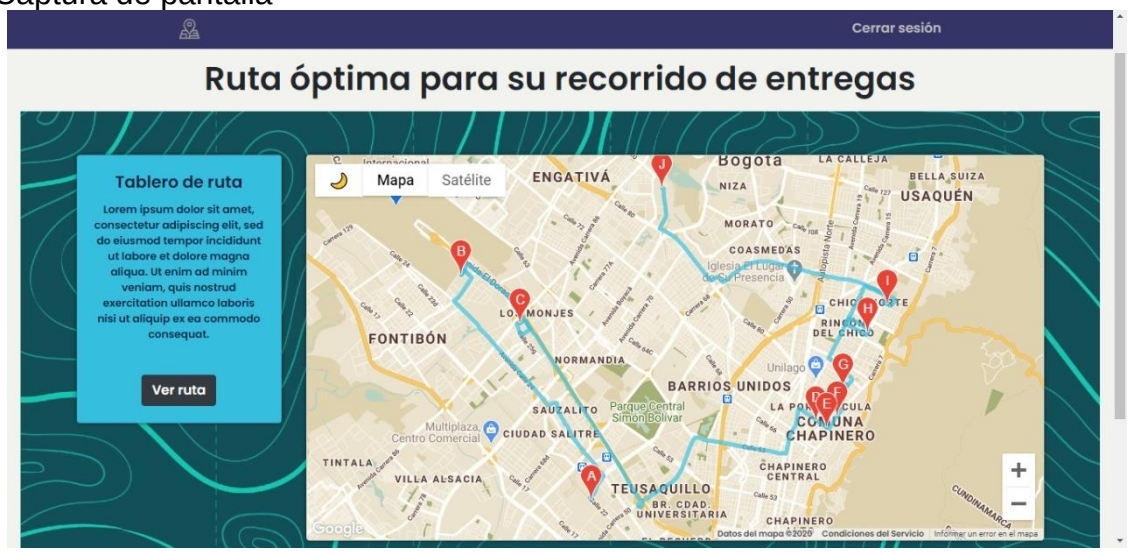
Cuadro 29. Ejecución prueba entrega de datos

Nombre de módulo	Módulo de clusterización.
Nombre de la prueba	Entrega de los datos procesados.
Objetivo de la prueba	Los datos serán entregados a la interfaz web para ser visualizados.

Complejidad	Media.
Perfil/Rol/Usuario	Administrador.
Tipo de navegador	No aplica.
Estado de la prueba	Exitosa.

Fuente: elaboración propia

Cuadro 30. Descripción prueba entrega de datos

Descripción de la prueba		
Flujo de prueba	**Descripción**	**Observaciones**
Acoplamiento de datos.	Los datos fueron agrupados en un array y entregados a la interfaz web (Django).	Los datos fueron entregados y procesados con librerías.
Recepción de datos por parte de la Interfaz web.	Los datos fueron entregados a Django y JavaScript para ser visualizados.	Los datos fueron entregados a Django para luego ser procesados por API de Google Maps.
Captura de pantalla 		
Ejecución exitosa de procesamiento de datos		

Total, casos de prueba exitosos	5
Total, casos de prueba no exitosos	4
Total, casos de prueba cancelados	0
Total, casos de prueba ejecutados	9

Fuente: elaboración propia

6.4.2. Módulo de interfaz web

- Validación de formulario de inicio de sesión

Cuadro 31. Ejecución prueba interfaz web

Nombre de módulo	Módulo de interfaz web (Django).
Nombre de la prueba	Validación de formulario de Inicio de sesión.
Objetivo de la prueba	Lograr tener un inicio de sesión al sistema de ruteo, cargando datos de usuario de la base de datos.
Complejidad	Media.
Perfil/Rol/Usuario	Administrador.
Tipo de navegador	Chrome.
Estado de la prueba	Exitosa.

Fuente: elaboración propia

Cuadro 32. Descripción prueba interfaz web

Descripción de la prueba		
Flujo de prueba	**Descripción**	**Observaciones**
Validación del formulario de Inicio de sesión.	Datos para validar dentro del formulario de Django.	Los datos fueron cargados desde la base de datos de MySQL y alojados localmente con compatibilidad de Django.
CRUD de usuarios dentro del sistema.	Ingreso al sistema mediante la autenticación Django.	Basados en los datos alojados de MySQL, el sistema permitió autenticación.

Captura de pantalla

Ventana de inicio de sesión

Total, casos de prueba exitosos	10
Total, casos de prueba no exitosos	2
Total, casos de prueba cancelados	0
Total, de casos de prueba ejecutados	12

N.°	Nombre de la prueba	Criterio de aceptación	Aprobado
1	Validación de formulario de inicio de sesión.	Ingreso satisfactorio al sistema.	Sí

Fuente: elaboración propia

Integración Django - API Google Maps

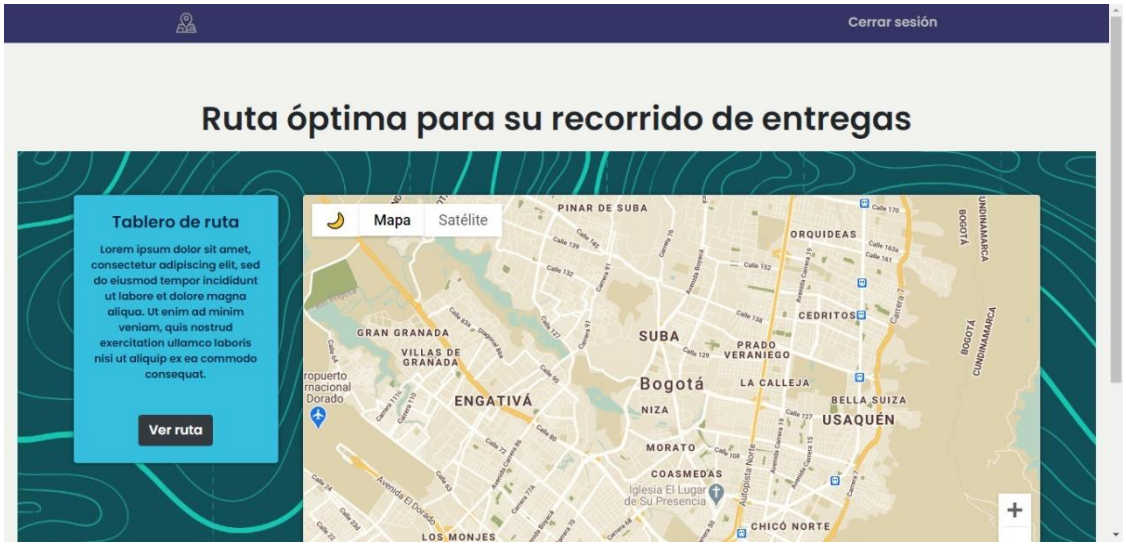
Cuadro 33. Ejecución prueba integración Google Maps

Nombre de módulo	Módulo de interfaz web (Django)
Nombre de la prueba	Integración Django - API Google Maps
Objetivo de la prueba	Validar la integración de los mapas de Google maps directamente en la interfaz de Django.
Complejidad	Media.

Perfil/Rol/Usuario	Administrador.
Tipo de navegador	Chrome.
Estado de la prueba	Exitosa.

Fuente: elaboración propia

Cuadro 34. Descripción prueba integración Google Maps

Descripción de la prueba		
Flujo de prueba	**Descripción**	**Observaciones**
Validación de Visualización del mapa.	Verificación de la integración del mapa de Google Maps con la interfaz de usuario de Django.	El mapa se desplegó sin errores de código.
Validación de elección de parámetros.	el mapa debe permitir la elección de parámetros dentro de su visualización	Estas interacciones no interfirieron con la ejecución de Django.
Captura de pantalla 		
Visualización mapa Google en módulo de pruebas		

Total, casos de prueba exitosos	2
Total, casos de prueba no exitosos	1

Total, casos de prueba cancelados	0
Total, casos de prueba ejecutados	3

N.°	Nombre de la prueba	Criterio de aceptación	Aprobado
1	Integración Django - API Google Maps.	Visualización de la interfaz de Google Maps dentro de un *frame* en Django.	Sí

Fuente: elaboración propia

- Validación de trazado de ruta ideal

Cuadro 35. Ejecución prueba de validación trazado de ruta ideal

Nombre de módulo	Módulo de interfaz web (Django).
Nombre de la prueba	Validación de trazado de ruta ideal.
Objetivo de la prueba	Verificar el trazado de la ruta ideal según la arquitectura fijada para el proyecto basado en clusterización.
Complejidad	Media.
Perfil/Rol/Usuario	Administrador.
Tipo de navegador	Chrome.
Estado de la prueba	Exitosa.

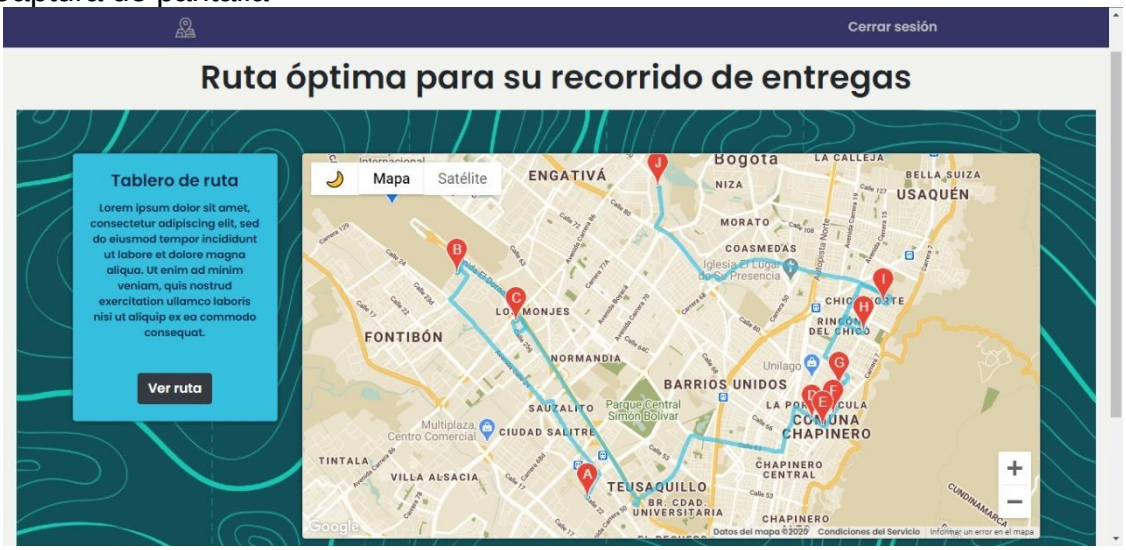
Fuente: elaboración propia

Cuadro 36. Descripción prueba de validación trazado de ruta ideal

Descripción de la prueba		
Flujo de prueba	**Descripción**	**Observaciones**
1. Validación del punto de inicio.	El formulario de Django y Google Maps da el punto de inicio	El formulario ofreció parámetros de entrada para iniciar una ruta.
2. Validación de los *waypoints.*	El algoritmo de clusterización da los *waypoints.*	Basado en el *array* que contiene las coordenadas de los puntos intermedios de la ruta.

3. Validación del punto de destino.	El formulario de Django y Google Maps ofreció el punto de fin.	El formulario brindó los parámetros de entrada para finalizar una ruta.

Captura de pantalla



Despliegue de *waypoints* basado en los datos registrados

Total, casos de prueba exitosos	3
Total, casos de prueba no exitosos	0
Total, casos de prueba cancelados	0
Total, casos de prueba ejecutados	3

N.°	Nombre de la prueba	Criterio de aceptación	Aprobado
1	Visualizar el trazado de la ruta ideal según la arquitectura fijada para el proyecto basado en clusterización.	No visualizar el trazado de la ruta ideal de acuerdo con la arquitectura fijada para el proyecto basado en clusterización.	Sí

Fuente: elaboración propia

6.5. RECURSOS DE LAS PRUEBAS

6.5.1. Requerimientos de entornos – software

- Módulo de inteligencia artificial (clusterización)

  - **Google Colaboratory:** este módulo fue desarrollado y probado en Google Colaboratory. Este permitió tener grandes recursos de hardware en préstamo para el alto procesamiento de datos requerido para este proyecto. Asimismo, por medio de este entorno se dispuso de tarjetas gráficas de alto poder, las cuales ayudaron a optimizar dicho procesado.

  - **Anaconda Python:** una vez puesto en producción, se adjuntaron las librerías necesarias sobre un entorno virtual, garantizando la perfecta compatibilidad, tanto del lenguaje de programación como de las librerías requeridas para tal fin.

- Módulo de base de datos

MySQL: es un sistema de gestión de bases de datos relacionales de código abierto (RDBMS) y de libre acceso que utiliza el lenguaje de consulta estructurado (SQL). SQL es el lenguaje más popular para agregar, acceder y administrar contenido en una base de datos. Es más conocido por su procesamiento rápido, confiabilidad probada, facilidad y flexibilidad de uso. MySQL es una parte esencial de casi todas las aplicaciones PHP de código abierto.

- Módulo de interfaz web

Python Django: este módulo fue desarrollado usando este *framework* de aplicaciones web gratuito y de código abierto, los cuales ofrecieron un conjunto de componentes para desarrollar sitios web más fácil y rápidamente. Además es completamente compatible con el JavaScript necesario para la visualización de mapas de Google Maps.

6.5.2. Requerimientos de entornos – hardware

- Módulo de clusterización

Para el desarrollo de estos módulos el hardware requerido, se recurrió a las bondades que ofrece de manera gratuita Google con sus servicios de Google Colaboratory, los cuales van disponiendo herramientas según las necesidades del proyecto.

- Módulo de inteligencia de interfaz web

Computador de escritorio con las siguientes características:

  - Procesador AMD Ryzen 5 3400G.

- RAM 8,00 GB.
- Sistema operativo Windows 10.

7. INSTALACIÓN Y CONFIGURACIÓN

7.1. COMPONENTES FUNDAMENTALES PARA LA INSTALACIÓN

Posterior al análisis de la información sobre el sistema, fue necesaria la instalación de los distintos módulos, los cuales contienen el funcionamiento interno y externo del aplicativo. Lo anterior, con el fin de generar una adecuada optimización de las rutas de acceso para las entregas estimadas y desarrolladas.

7.2. RECURSOS DE SOFTWARE

7.2.1. Matriz de certificación

En la matriz de certificación se encuentra una descripción de la compatibilidad de los módulos correspondientes con las tecnologías actuales y utilizadas para la realización de estos. Entre estas tecnologías con las que fueron desarrolladas las distintas aplicaciones internas del proyecto, se garantizó un estado de funcionalidad y de compatibilidad con aplicaciones externas como lo son la interacción con MySQL, puesto que la arquitectura del proyecto está desarrollada en Python en su versión 3.8 y la administración en Django 3.1.0, siendo esta la versión más reciente y más actualizada de este *framework*.

Se debe tener en cuenta que para el desarrollo principal del proyecto se hizo una emulación de máquinas virtuales para facilitar el desarrollo de este, hasta que se encontrara en producción o publicación directa de dominio. Una vez se siguieron los parámetros de configuración y despliegue de la aplicación debidamente configurada en dominio, esta funcionó sin ningún inconveniente. Esto incluyó la utilización de *smartphone*, computadores portátiles o de mesa, puesto que el diseño es completamente adaptativo a la aplicación que se empleó.

7.2.2. Restricciones técnicas del sistema

Para una correspondiente implementación del proyecto y para que fuera visible en distintas plataformas, considerando la tecnología de desarrollo, se decidió que el despliegue fuera de producción, lo cual se puede realizar en servidores web como Linux o Windows Server.

7.2.3. Instalación y configuración del software base

Software base	
Descripción	Es la correspondiente implementación del software, dado que este está configurado para ejecución no separada de componentes, permitiendo una sola configuración global para su funcionamiento.

Procedimiento de instalación	
Paso 1	Establezca un dominio principal para la publicación de la aplicación web.
Paso 2	Si usa Windows Server realice instalación de Anaconda, esto le permite tener todas las librerías correspondientes a Python para su ejecución.
Paso 4	En caso de que su servidor sea de distribución Linux Server, se recomienda realizar la actualización de los repositorios a Python 3.8.
Paso 5	Realice ubicación del aplicativo en los archivos internos del servidor de aplicaciones web con los que cuenta el servidor, ya sea distribución Linux Server o Windows Server.
Paso 6	Asegúrese de que la seguridad de acceso al servidor sea estable y sin fallas.
Paso 7	Realice la configuración correspondiente de los puertos del servidor, justo donde quiere realizar la publicación del sitio web para que sea accesible desde cualquier lugar o plataforma.
Parámetros para configurar	
Paso 1	Una vez realizada la configuración de todos los parámetros del servidor, realice la configuración del proyecto en el archivo settings.py, de la carpeta principal. En esta sección realice la búsqueda de la línea número 26. En esta sección encontrará el DEBUG = True, realice cambio de esta condición a “False”, puesto que “True” solo es para la utilización de entorno de desarrollo.
Paso 2	Una vez puesta en marcha toda la configuración, realice pruebas internas para confirmar que todo haya salido bien y realice el despliegue total del proyecto a publicación.

Fuente: elaboración propia

8. MANUAL DE USUARIO

8.1. DEFINICIONES PARA MANUAL DE USUARIO

A través de toda esta documentación se encuentran términos clave para el procedimiento de uso y desarrollo. A continuación, se describen los más importantes:

Nombre	Descripción
Usuario	Persona que usará el aplicativo web para la interacción.
Amin	Usuario con privilegios de modificación y manipulación de características del documento.
RF	Requerimiento funcional.
Cloud Platform	Consola de herramientas de programación y desarrollo provisionada por Google.
MAPA	Aplicación de interacción con el usuario en las rutas.
DB	Bases de datos.
URL	Código fuente responsable del enlace de las distintas secciones del aplicativo web.
Views	Código fuente del *backend* del motor del aplicativo web.
Conda	Término utilizado para referirse a los aspectos involucrados con Miniconda y la ejecución de entornos virtuales.
Bloque	Describe la extensión de plantillas HTML por medio del código de programación del proyecto (Python).
Backend	El *backend* es la parte del desarrollo que se encarga de que toda la lógica de una página web o programa funcione.
Frontend	Parte funcional y visual del programa o aplicativo web.
IDE	Integrated Development Environment, son los aplicativos utilizados para el desarrollo de código.
Plantilla	Las plantillas son una virtud de Django para el manejo de estructuras (o bloques) HTML mediante Python.

8.2. ESTRUCTURA DE LA DOCUMENTACIÓN DEL USUARIO DEL SOFTWARE

A continuación, se expone la estructura que implica toda la documentación de usuario para la manipulación del software.

8.3. ESTRUCTURA GENERAL DE LA DOCUMENTACIÓN

La estructura de la documentación para usuario de software posee segmentos respecto a su orden y jerarquía de acuerdo con los componentes que se quieren abarcar para el software tratado. El objetivo de esta estructura consiste en ayudar al usuario para poder localizar y comprender el contenido de la información.

Dependiendo del público al que sea dirigido este documento, se seguirán los lineamientos de estructura pertinentes para poder focalizar bien la información que sea de valor para cada uno de sus lectores:

- Secciones separadas dedicadas a las necesidades de la audiencia, de forma específica.
- Información de documentos o herramientas específicos para las necesidades de la audiencia.

Asimismo, en esta lista se pueden observar puntos clave de este manual que pueden ser de utilidad para usuarios concretos.

Cuadro 37. Glosario de palabras para manual de usuario

Componente	¿Requerido?
Información general para el uso de la documentación.	Requerido para la comprensión global de este documento y su contenido.
Conceptos de las operaciones	Requerido para entrar en materia respecto a todo el paso a paso vital para el proceso de uso del proyecto.
Procedimientos detallados.	Requerido para obtener mayor conocimiento sobre procedimiento de uso y manipulación de ruteo y todos sus componentes.
Información específica sobre comandos de software.	Necesario para conocimiento de procedimientos y ejecuciones internas del proyecto.
Todos los mensajes de error del sistema .	Vital para conocimiento previo de posibles inconvenientes que se puedan presentar

	durante los intentos de ejecución del proyecto.
Resolución de posibles problemas que presente el sistema.	Junto a todos los mensajes de error del sistema, en dicha sección se encuentran detalles sobre cómo mitigarlos.
Diccionario de datos.	Toda la terminología interna del proyecto.

Fuente: elaboración propia

8.4. COMPONENTES INICIALES

A continuación, se describen etapa por etapa según su tipo, los distintos componentes, lenguajes y herramientas utilizados y requeridos para la ejecución, instalación y manipulación del software:

8.4.1. Herramientas e IDE

Cuadro 38. Herramientas y entornos de desarrollo para el usuario

Miniconda	
En su versión 3 o posterior, Miniconda es un conjunto de herramientas para implementar, junto al lenguaje de programación Python, la ejecución de entornos virtuales de desarrollo. Mayor información sobre Miniconda (O Conda en general) consultar el sitio web propietario. Nota: se recomienda que, al momento de instalar Miniconda, se agregue al PATH la ruta de Python/Conda, acción que se podrá realizar durante dicho proceso de instalación a recomendación del instalador.	
Python	En su versión 3.8 o posterior, Python es el principal lenguaje de programación utilizado en el desarrollo del software. Junto con los módulos especificados más adelante, se desarrollan todas las actividades de programación para este sistema.
PIP	En su versión: 20.1.1 o posterior, PIP permite la instalación de paquetes o módulos para el lenguaje de programación Python, con el fin de emplearlos en el desarrollo de los componentes del software.

Conda virtual environments	Con Conda se tiene la posibilidad de utilizar ENV para el desarrollo de las herramientas del software (Ya es función nativa de Miniconda).

PyCharm
Es el IDE utilizado para desarrollar todo el código del software, su versatilidad con múltiples lenguajes de programación, herramientas de apoyo y estabilidad hicieron de este la selección apropiada para el desarrollo.

Fuente: elaboración propia

8.4.2. Plataformas web

Cuadro 39. Plataformas web usadas por el usuario

Google Cloud Platform	
Google Cloud Platform es una plataforma de Google para desarrollo con base en las soluciones propias de Google, en este caso específico, Google Maps. Por medio de esta se obtiene acceso a diversas API que permiten emplear herramientas de Google Maps como se deseen programar.	
Geocoding API	Esta API permite al usuario manipular todas las herramientas de Google Maps, en relación con la codificación y obtención de puntos de coordenadas en un plano determinado del mapa.
Maps JavaScript API	Permite la manipulación de mapas de Google programables bajo JavaScript. Es la API más importante que se debe implementar para poder intervenir el mapa.
Directions API	Ofrece la posibilidad y todas las herramientas bajo el poder de Google Maps para el trazado de rutas.

Geolocation API	Ofrece todas las herramientas requeridas para utilizar geolocalización en el proyecto.

Fuente: elaboración propia

8.4.3. Lenguajes, *frameworks*, módulos y librerías

Cuadro 40. Plataformas web usadas por el usuario

Python	
En su versión 3.8 o posterior, Python es el lenguaje con el que se desarrolló todo el *backend* de ruteo. Incluido al instalar Miniconda, el lenguaje Python viene en versión 3.8. Junto con Python se utilizaron múltiples módulos y librerías para complementar el desarrollo del sistema de información. Nota: todos los módulos, *frameworks*, Python y librerías usadas se instalan mediante comandos PIP (instalados por medio de Miniconda).	
Django	En su versión 3.1.3 o posterior, Django es el *framework* de desarrollo web escrito en Python, utilizado para todo el *backend* del sitio web de ruteo. Comando: *pip install django==3.1.3*
Google Maps	Es una librería de Python para uso de las herramientas de Google Maps dentro del proyecto. Es usado para temas de geocodificación de direcciones a coordenadas y para procesos matemáticos de cálculo de distancias con base en coordenadas. Cabe mencionar que dichos cálculos no son nativos de esta librería, se hacen según el resultado de geocodificación. Nota: es necesario indicar la credencial API obtenida en Google Cloud Platform para esta labor, activando las API necesarias para uso de geocodificación y direcciones.

	Comando: *pip install googlemaps*
Pandas	Es una librería que se usa para la manipulación de datos y análisis. Comando: *pip install pandas*
Matplotlib	Librería usada para manipulación gráfica de listas y arreglos. Comando: *pip install matplotlib*

JavaScript
Para el desarrollo de este proyecto se empleó lenguaje de programación JavaScript, dado que las integraciones de Google Maps corren bajo este lenguaje de programación web.

HTML
Siguiendo el camino de la programación web utilizada para este proyecto, se tuvo presente el manejo de HTML y en especial HTML5, en caso de que fuera requerido hacer modificaciones al *frontend* Web.

Bootstrap
Framework CSS utilizado para el desarrollo del *frontend* web del proyecto. Se recomienda tener presente utilizar la versión 4.0 de Bootstrap en caso de ser requeridas modificaciones al sitio web.

Fuente: elaboración propia

8.5. INFORMACIÓN CRÍTICA

Respecto a puntos críticos de interés en el programa, a continuación se exponen unas indicaciones para tener en cuenta al momento de hacer uso o de llevar a cabo intervenciones al software del proyecto:

Cuadro 41. Información crítica

Herramientas e IDE:
Las herramientas seleccionadas para el trabajo y programación del código de ruteo fueron seleccionadas en temas de rendimiento y eficacia en conjunto con su sencillez de uso. Se recomienda seguir el uso de dichas herramientas para una programación y edición sencilla de cualquier módulo o módulos del software. Las herramientas e IDE indicadas aquí son compatibles tanto con Windows como con Linux.

Plataformas Web:
Las plataformas web empleadas en el estudio fueron provistas por Google, por medio de Cloud Platform. Por ello, es necesaria la creación de un proyecto para poder utilizar todas estas herramientas, para las cuales se requiere el ingreso de un método de pago válido para su uso total y sin limitaciones.

Lenguajes, *frameworks*, módulos y librerías:
Python y JavaScript fueron los lenguajes de programación utilizados para el desarrollo de todo el software de ruteo, y junto con el lenguaje HTML, forman todo el sistema de información. Se debe tener presente que, por ello, hay momentos en los que se requiere interacción mutua entre ellos por medio de *forms* y bloques.

Fuente: elaboración propia

8.6. CONTENIDO INFORMATIVO DE LA DOCUMENTACIÓN DEL USUARIO DE SOFTWARE

En esta sección se abordan todos los temas requeridos respecto al contenido informativo para toda la documentación expuesta en el documento.

8.6.1. Información para el uso de la documentación

Teniendo conocimiento previo de todos los componentes de desarrollo del sistema de información, es viable comenzar a analizar los puntos necesarios del uso de toda esta herramienta y, sobre todo, de la documentación.

- Este documento es de carácter informativo e instructivo para todo el proceso de uso, manipulación e implementación del proyecto de ruteo. Se encontrarán

características y procedimientos sobre componentes y modos de operatividad de todo el software relacionado.

- Es necesario llevar a cabo toda la instalación y configuración de componentes, lenguajes, códigos, características, *frameworks*, entre otros, listados en todo el documento para el funcionamiento correcto del sistema de información.
- Respetar siempre las versiones mínimas indicadas en este documento para componentes, lenguajes, códigos, características, *frameworks*, o en su caso, versiones más recientes de las indicadas. Siempre los elementos utilizados en el desarrollo de este software garantizarán compatibilidad total con sus versiones predecesoras; sin embargo, siempre se debe revisar la documentación pertinente sobre nuevas versiones, sus cambios, mejoras o añadidos.
- Es recomendable que todo el equipo de trabajo use las mismas herramientas de desarrollo con las mismas versiones de cada una, para así garantizar compatibilidad total y evitar problemas de funcionamiento.

8.6.2. Concepto de operaciones

Respecto a las operaciones dentro del proyecto, se pueden destacar las siguientes acciones u operaciones que se deben realizar para que de esta forma la ejecución del proyecto sea exitosa:

- Adecuación de todas las herramientas de trabajo requeridas para la manipulación del código del proyecto, recordando que cada partícipe del desarrollo del software utilice las mismas herramientas para evitar problemas de compatibilidad o errores al usar IDE distintos. Para ello, es necesario instalar Miniconda y PyCharm.
- Compilación de los códigos de fuente partícipes del proyecto para su futura ejecución y sin inconvenientes. Es necesario instalar todos los módulos listados por medio de PIP para poder cubrir todas las demandas que el programa requiere y necesita operar para su ejecución.
- Ejecución del programa mediante el servidor de Django, con esta alternativa se permite correr el software en modo de pruebas, como un sitio web común y corriente; pero usando un servidor virtual y local, para así poder trabajar sin inconvenientes.

Todos estos pasos descritos pueden encontrarse de manera detallada en la sección 4.3, donde se especifican técnicas, métodos y actividades para la correcta ejecución (paso a paso) del software del proyecto.

Nota: respecto a compilación se refiere al acoplamiento de múltiples módulos, librerías o *frameworks* para satisfacer una tarea en común y no a la ejecución de código Python, debido a que Python es lenguaje intérprete. Evitar confundir estos términos.

8.6.3. Información para el uso general del software

En este apartado se describen los antecedentes conceptuales para el uso del software de ruteo, así como todos los procesos vitales que conllevan al uso de software. Para ello, se puede observar el procedimiento paso a paso de tres etapas: adecuación, compilación y ejecución del software.

Nota: estos pasos tienen en cuenta que el usuario ya posee el código de fuente y archivos del proyecto, así como una cuenta con acceso a Cloud Platform, por lo que el instructivo se centra en el protocolo paso a paso de ejecución directamente.

Cuadro 42. Uso general del software para el usuario

Adecuación del software		
Número	**Tipo de paso**	**Descripción**
1	Instalación de Miniconda	Se necesita realizar la instalación de Miniconda para obtener Python 3.8 o de mayor versión junto a las otras herramientas como los VENV y PIP, y así configurar el punto de partida del software. Es importante recordar agregar la variable de entorno de Python al PATH. Se puede instalar Miniconda ingresando al enlace de descarga de la herramienta.
2	Instalación de PyCharm	Se requiere instalar y configurar el IDE PyCharm en el dispositivo para poder acceder, modificar y trabajar todo el código del proyecto.

Fuente: elaboración propia

8.7. INTERFACES GRÁFICAS

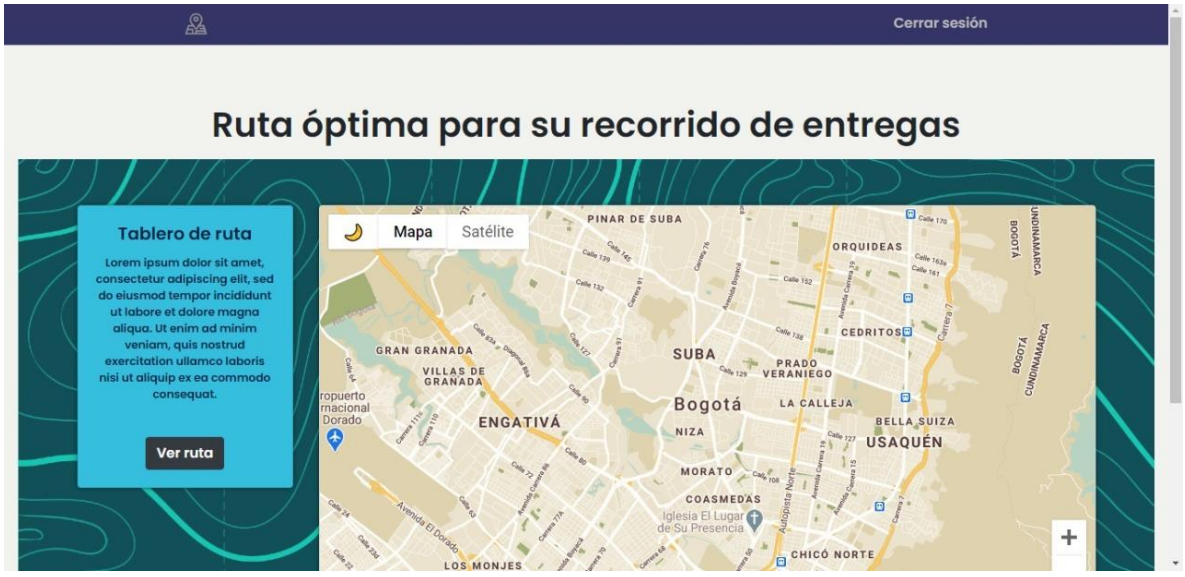
8.7.1. Interfaz de inicio de sesión

Figura 27. Captura interfaz inicio de sesión

Fuente: elaboración propia

8.7.2. Mapa de rutas óptimas

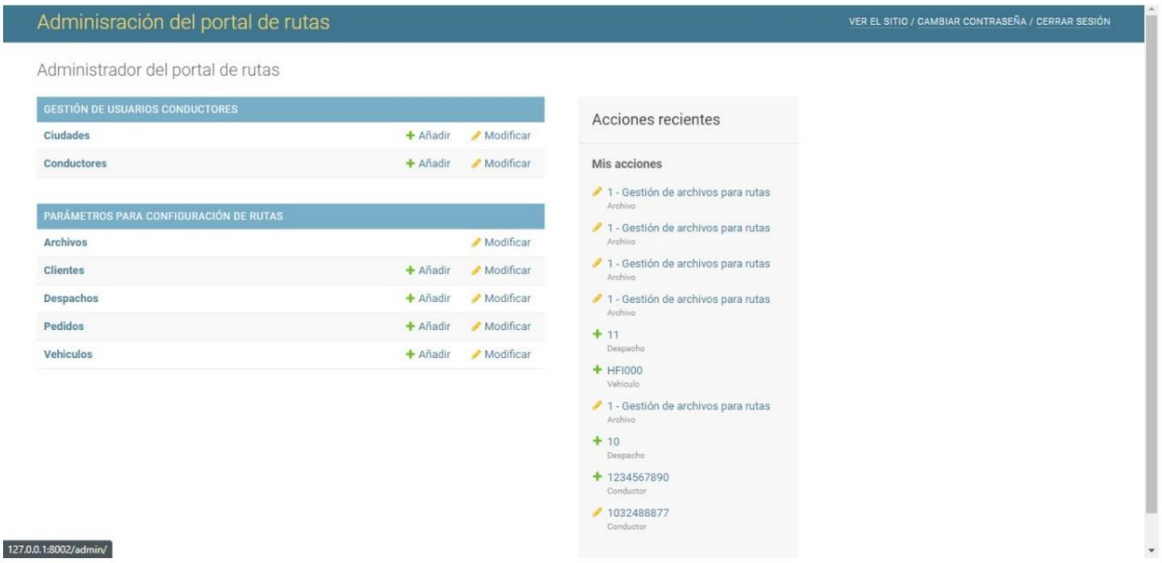
Figura 28. Captura mapa interfaz



Fuente: elaboración propia

8.7.3. Portal administrador de rutas

Figura 29. Captura modelo de datos por administración



Fuente: elaboración propia

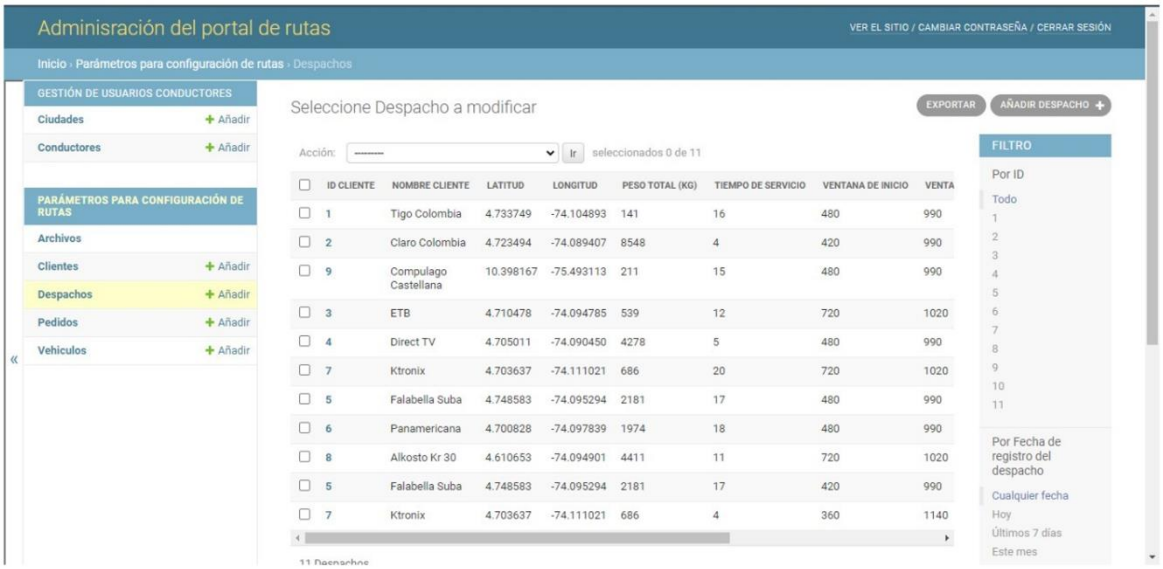
8.7.4. Vista ruta establecida

Figura 30. Captura visualización ruta

Fuente: elaboración propia

8.7.5. Vista de despachos

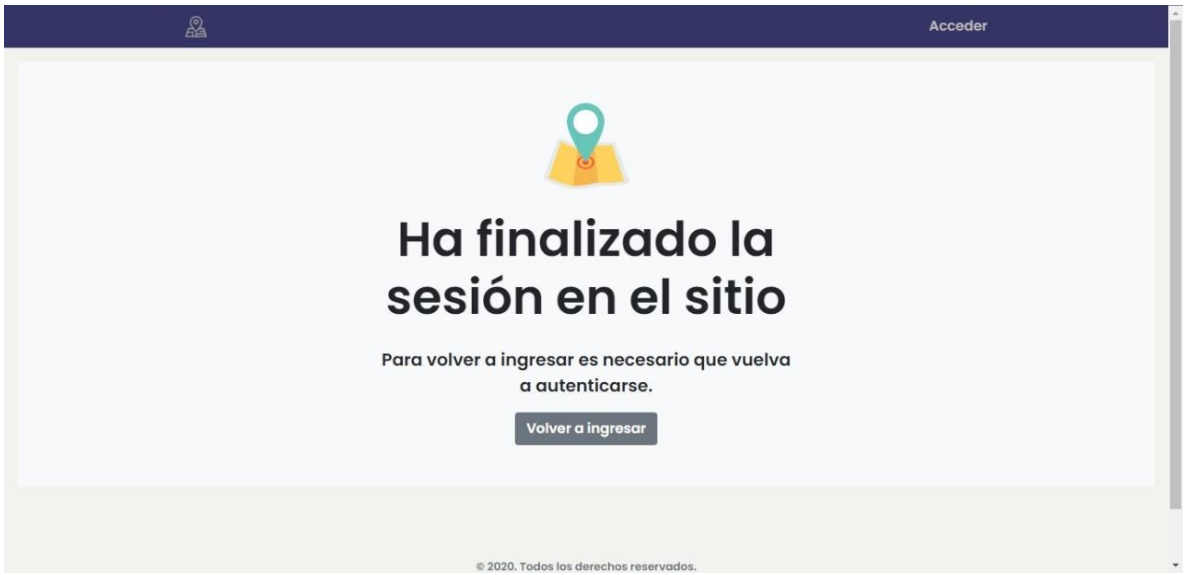
Figura 31. Lista de despachos



Fuente: elaboración propia

8.7.6. Vista fin de sesión

Figura 32. Cierre de sesión



Fuente: elaboración propia

9. RESULTADOS

9.1. DESARROLLO DE LOS OBJETIVOS

Como se mencionó en un principio, para el desarrollo del presente proyecto se buscaron los mejores referentes respecto a herramientas de desarrollo de software, que tuvieran plena compatibilidad con arquitecturas de fácil implementación y, a su vez, que contara con plena compatibilidad con el lenguaje de programación Python. En ese sentido, se optó por utilizar el *framework* Django, el cual permite la estructuración de modelos de datos de fácil implementación, y en un futuro posibilitará la expansión de estos modelos. Por otra parte, se utilizó el lenguaje de programación JavaScript, haciendo uso de las librerías de mapas de Google para la visualización de rutas óptimas en un navegador web. Este lenguaje se encuentra plenamente destinado para la realización de esta tarea y cuenta con plena compatibilidad con las API de Google referenciadas en este trabajo de investigación.

Respecto al primer objetivo, una vez identificadas las mejores herramientas para el cumplimiento este proyecto, se diseñó un prototipo de arquitectura con tecnologías *frontend* y *backend* orientadas al uso del *framework* Django, el cual es elaborado en Python. Esto permitió una óptima comunicación entre bases de datos y para su respectivo despliegue se utilizó la base de datos MySQL. Asimismo, se hizo la gestión de datos a través de Pandas y NumPy, sin dejar a un lado los servicios de Google a través de las APIS y la dirección ligera de coordenadas. El *frontend* se desarrolló según el marco referencial Bootstrap, el cual cuenta con muchas facilidades de diseño de sitios responsivos, haciendo posible la compatibilidad entre los mapas de Google Maps y la descarga de datos para el modelo de clusterización.

Una vez realizado todo el diseño del prototipo de arquitectura, se ejecutó el segundo objetivo relacionado con todo el desarrollo de software sobre una estrategia en la metodología Scrum, con el propósito de realizar iteraciones para el pleno cumplimiento del diseño de la arquitectura. A su vez, se estructuró el modelo de datos para la carga de los datos correspondientes para la plaza de rutas óptimas de una empresa logística, puesto que, a través del módulo de administración, fue posible la gestión de estos modelos de datos para continuar el sistema, brindando la información para la gestión de rutas.

El tercer objetivo se orientó a la fase de pruebas, la cual fue culmina satisfactoriamente bajo los estándares establecidos para este cumplimiento, tales como pruebas de datos, pruebas de modelos de clusterización, de interfaz de usuario; permitiendo garantizar la plena usabilidad del prototipo del sistema de información del presente proyecto. Todas las pruebas cumplieron con los umbrales mínimos exigidos para garantizar una calidad del software óptima para el cumplimiento de los objetivos.

Figura 33. Gráfica de consolidación de las pruebas realizadas

Fuente: elaboración propia

Cuadro 43. Número total de pruebas realizadas

Detalle	Número
Pruebas exitosas	25
Pruebas no Exitosas	9
Pruebas canceladas	1

Fuente: elaboración propia

10. CONCLUSIONES

Por medio del presente proyecto investigativo, se logró diseñar un sistema de información con una tecnología de punta, en aras de establecer una interacción de usuarios, a través de una interfaz amigable, haciendo posible la visualización de un mapa interactivo en cual se plasman las redes de distribución Dicho mapa interactivo se construyó sobre una sólida arquitectura del software, dividida en *backend* y *frontend*, posibilitando una gestión óptima de datos, así como una adecuada administración de usuarios. Adicional a ello, se llevó a cabo una correcta implementación del módulo de clusterización, lo que permitió encontrar las rutas más cortas y óptimas para un operador logístico de Colombia.

En ese orden de ideas, el desarrollo de software estuvo sujeto a una metodología ágil como Scrum, Implementando un modelo de datos adecuado a la problemática expuesta. Lo anterior, soportado por un lenguaje de programación sólido y con gran reconocimiento en la industria. Adicionalmente, la óptima gestión del marco de referencia Django, adecuado para el desarrollo de modelos de datos, estuvo basado en estructuras relacionales.

Finalmente, cabe mencionar que a través de este prototipo se puede visualizar la manera en que, por medio de la aplicación de un modelo, se puede llegar a la optimización de recursos empresariales, principalmente aquellos que tengan centrada su estrategia de negocio en cadenas de suministros y gestión logística; todo esto aplicado a un sistema de información teniendo la plena gestión de los datos necesarios para el cumplimiento de los retos logísticos.

11. RECOMENDACIONES

Para el desarrollo adecuado de futuros proyectos en estrecha relación con el expuesto en este documento, a continuación, se presentan unas cuantas recomendaciones a tener en cuenta:

- Prever una evolución tecnológica que optimice el modelo de datos y permita una mejor evaluación de las rutas óptimas dentro de una cadena de suministro y gestión logística.
- Hacer una validación de datos en diferentes ciudades, las cuales difieren a las características propias de la malla vial de la ciudad de Bogotá, a través de la cual se realizaron las validaciones propias para el presente proyecto.
- Se recomienda realizar la implementación del presente sistema de información con una empresa logística del mercado actual, la cual pueda verse beneficiada con la optimización de los recursos para la gestión de cadenas de suministro y gestión logística.
- Realizar el correspondiente despliegue del presente software en un servidor dedicado para tal fin, para tener plena accesibilidad desde cualquier punto en internet.

12. ANEXOS

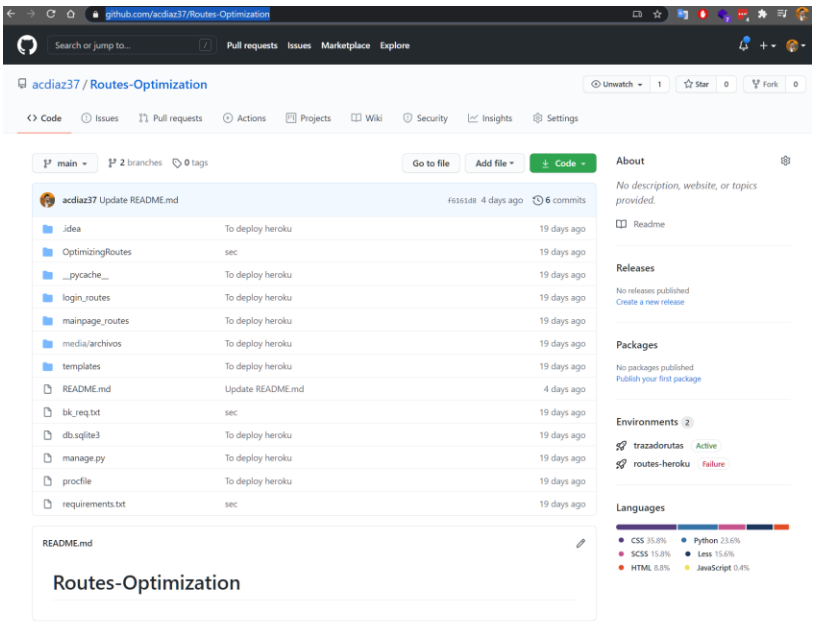
Anexo 1. Repositorio de código

A continuación, se entrega la ruta del código basado en las pruebas realizadas al prototipo de sistema de información para la instalación de dependencias y librerías y para su posterior ejecución.

Link: https://github.com/acdiaz37/Routes-Optimization

- Estructura datos repositorio

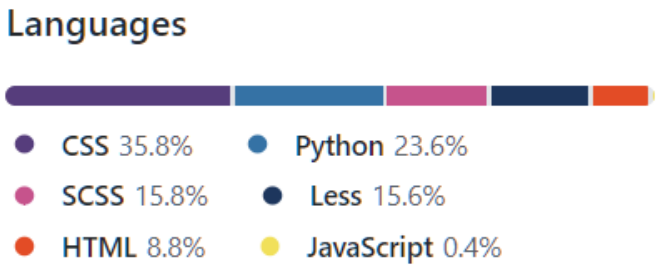
Figura 34. Gráfica de consolidación de las pruebas realizadas



Fuente: elaboración propia

- Lenguajes utilizados

Figura 35. Lenguajes usados



Fuente: elaboración propia

13. BLIBLIOGRAFÍA

BANCO INTERAMERICANO DE DESARROLLO (BID). Distribución urbana de mercancías con centros logísticos. Nota técnica. [En línea]. [2013]. Disponible en: /Distribucion_Urbana_de_Mercancias__Estrategias_con_Centros_Logisticos._Nota_Tecnica<http://www.academia.edu/download/55566569/Distribucion_Urbana_de_Mercancias__Estrategias_con_Centros_Logisticos._Nota_Tecnica.pdf>

DEPARTAMENTO ADMINISTRATIVO NACIONAL DE ESTADÍSTICA (DANE). PIB por departamento. [En línea]. Disponible en: https://www.dane.gov.co/index.php/estadisticas-por-tema/cuentas-nacionales/cuentas-nacionales-epartamentales#:~:text=Para%20el%20a%C3%B1o%202019pr,de%20millones%20de%20pesos%2C%20respectivamente

DEPARTAMENTO NACIONAL DE PLANEACIÓN (DNP). Encuesta nacional logística 2018. [En línea]. [2019]. Disponible en: https://onl.dnp.gov.co/es/Publicaciones/SiteAssets/Paginas/Forms/AllItems/Informe%20de%20resultados%20Encuesta%20Nacional%20Log%C3%ADstica%202018.pdf

DJANGO. Why Django? [En línea]. [2020]. Disponible en: https://www.djangoproject.com/start/overview/

FUENTES, Manuel y PEÑA, Junio. Desarrollo de un Sistema de Información con tecnologías Web y Móvil para tramitar la inspección de vehículos en una compañía de seguros. Caracas: Universidad Central de Venezuela, 2015.

GARCÍA, José. Python como primer lenguaje de programación textual en la Enseñanza Secundaria. En: Education in the Knowledge Society, 2017, vol. 18, no. 2, p. 147-162.

GOOGLE CLOUD. Product Launch Stages. [En línea]. [2020]. Disponible en: https://cloud.google.com/products?hl=es#product-launch-stages

GOOGLE CLOUD. Google Maps Platform. [En línea]. [2020]. Disponible en: https://cloud.google.com/maps-platform/products?hl=es-419

GOOGLE DEVELOPERS. Directions API Usage and Billing. [En línea]. developers.google.com/ [en línea]. Disponible en: https://developers.google.com/maps/documentation/directions/usage-and-billing#other-usage-limits

GUTIÉRREZ, Rogelio. ¿Cómo enfrenta hoy la logística colombiana las nuevas necesidades de los mercados? [En línea]. [2019]. Disponible en: https://revistadelogistica.com/logistica/como-enfrenta-hoy-la-logistica-colombiana-las-nuevas-necesidades-de-los-mercados/

HERNÁNDEZ, Alejandro. Los sistemas de información: evolución y desarrollo. En: Proyecto social: Revista de relaciones laborales, 2003, no 10, p. 149-165.

LEÓN, ANDRÉS. Modelo de ruteo de vehículos para la red de distribución de mercancías de un operador logístico en Colombia. Bogotá, D.C.: Universidad Católica de Colombia, 2020.

MONTERROSA, Heidy. Logística se lleva 13,5% de los ingresos de las compañías en Colombia. [En línea]. [13 de diciembre de 2018]. Disponible en: https://www.larepublica.co/economia/logistica-se-lleva-135-de-los-ingresos-de-las-companias-en-colombia-2805319#:~:text=Una%20empresa%20en%20Colombia%20destina,la%20que%20participaron%202.738%20empresas

NAVARRO, Mirta, *et al*. Integración de arquitectura de software en el ciclo de vida de las metodologías ágiles. Una perspectiva basada en requisitos. [En línea]. Disponible en: http://sedici.unlp.edu.ar/bitstream/handle/10915/62077/Documento_completo.pdf-PDFA.pdf?sequence=1&isAllowed=y

NUMPY COMMUNITY. User Guide NumPy. [En línea]. [2020]. Disponible en: https://numpy.org/doc/stable/numpy-user.pdf

PANDAS. User Guide Pandas. [En línea]. [2020]. Disponible en: https://pandas.pydata.org/docs/user_guide/index.html

PIXABAY. Camión transporte logística. [En línea]. [2018]. Disponible en: https://pixabay.com/es/photos/cami%C3%B3n-transporte-log%C3%ADstica-1030846/

PYTHONANYWHERE. Plans and pricing. [En línea]. [2020]. Disponible en: https://www.pythonanywhere.com/pricing/

PORTAL, Carlos. Costos logísticos: qué son, cuáles son y cómo minimizarlos. [En línea]. [29 de junio de 2011]. Disponible en: https://www.gestiopolis.com/costos-logisticos-que-son-cuales-son-y-como-minimizarlos/

RAMÍREZ, Andrés. Manual de la gestión logística del transporte y distribución de mercancías. Barranquilla: Ediciones Uninorte, 2009.

RED HAT. Qué son las API y para qué sirven. [En línea]. Disponible en: https://www.redhat.com/es/topics/api/what-are-application-programming-interfaces

RED HAT. ¿Qué es una API? [En línea]. Disponible en: https://www.redhat.com/cms/managed-files/styles/wysiwyg_full_width/s3/API-page-graphic.png?itok=5zMemph9>

ROLDÁN, Paula. Cadena de suministro. [En línea]. [2017]. Disponible en: https://economipedia.com/definiciones/cadena-de-suministro.html

SCHREINER, Keith. A software architect's view on diagramming. [En línea]. Disponible en: https://meghantaylor.net/diagramming-30545905

SOLANO, J. Lenguajes de programación. Temario. Lima: Universidad Nacional de Ingeniería, Facultad de Ciencias, 2011.