Diseño de una aplicación web como apoyo para recuperación de documentos perdidos por los ciudadanos, que se encuentran en custodia por la policía nacional- metropolitana de Ibagué

Jonny Josué Lozano Pineda
Ruby Daniela Morales Rendón

Universidad Nacional Abierta y a Distancia - UNAD
Ingeniería de Sistemas
Escuela de Ciencias Básicas, Tecnologías e Ingenierías - ECBTI
2022

Diseño de una aplicación web como apoyo para recuperación de documentos perdidos por los ciudadanos, que se encuentran en custodia por la policía nacional- metropolitana de Ibagué

Jonny Josué Lozano Pineda
Ruby Daniela Morales Rendón

Directora del proyecto
Gloria Alejandra Rubio Vanegas
(Ingeniera de Sistemas)

Universidad Nacional Abierta y a Distancia - UNAD
Ingeniería de Sistemas
Escuela de Ciencias Básicas, Tecnologías e Ingenierías - ECBTI
2022

Dedicatoria 1

Agradezco a Dios primero que todo por permitirme llegar hasta este momento, rodearme de las personas correctas para lograr mis metas propuestas y poder culminar con éxito mi carrera.

Gracias a mi familia, mis hermanos (Yudi, Eliza y Sebastián) mi madre (Rubiela Rendón) que siempre me apoyaron en todas mis decisiones y me acompañaron en todo momento para seguir adelante con todos mis objetivos.

Sin ti no sé si hubiese logrado llegar donde estoy ahora y hacer lo que me gusta, gracias a tus motivaciones, a tu ayuda y a siempre impulsarme para seguir adelante a pesar de los momentos en los que pensé que no podría continuar. (Jonny Lozano)

Dedicatoria 2

Dedico este título a mis padres por el apoyo en este nuevo inicio y por el mismo apoyo que me brindaron en el primer estudio que realicé.

A Daniela Morales por su compañía y la colaboración en los momentos difíciles de la carrera adonde el tiempo era poco y el esfuerzo por realizar mucho.

A mi mascota Cielo, porque su compañía brindó mucha energía y ánimo para hacer esto posible.

Agradecimientos

Agradecemos a nuestros padres por su apoyo incondicional en medio del proceso de obtener este título.

Agradecemos a nuestras parejas por su colaboración tanto moral como en los temas universitarios, ya que al hacer junto este título pudimos avanzar juntos de manera más eficiente.

Gracias a nuestra docente Emilia y Alejandra Rubio por su atención y compartir sus conocimientos a lo largo de la carrera en las dudas que surgieron,

Al Teniente coronel Alex Venegas Ruiz por su valiosa colaboración y aportes de la información necesaria para realizar este proyecto.

Resumen

Las TIC´S realizan un avance continuo en el mundo, y permiten que todo individuo tenga una visión holística de los procesos de aprendizaje que puede tener a su disposición, de esta manera la tecnología se ha vuelto un apoyo, imponiendo cambios significativos en la forma como las personas pueden adquirir nuevos conocimientos. Hoy en día existen muchas herramientas que sirven de apoyo para diferentes áreas del conocimiento, este proyecto pretende ser un apoyo entre el ciudadano y la Policía metropolitana de Ibagué, haciendo un acercamiento más eficaz y efectivo en las funciones misionales de la Policía y su relación con el ciudadano en un mecanismo que si se realizara de forma física, el tiempo que gastaria para verificar si en algún lugar de la policía se encuentra el documento seria largo puesto que existen más de 20 estaciones y en cada una de ella se encuentra un repositorio de documentos, adicional sin garantizar que lo pueda o no encontrar, por lo cual será un herramienta que permita acortar distancia y mejorar tiempos de respuesta para la solución a una problemática

Palabras clave: Aplicación web, Documentos, Recuperación, Custodia, Policía

Abstract

TIC's make continuous progress in the world, and allow every individual to have a holistic view of the learning processes that may be available to them, in this way technology has become a support, imposing significant changes in the way How can people acquire new knowledge? Today there are many tools that serve as support for different areas of knowledge, this project aims to be a support between the citizen and the Metropolitan Police of Ibagué, making a more efficient and effective approach in the missionary functions of the Police and its relationship with the citizen in a mechanism that if it were done physically, the time it would take to verify if the document is somewhere in the police would be long since there are more than 20 stations and in each of them there is a repository of documents, additional without guaranteeing that you can find it or not, for which it will be a tool that allows you to shorten distance and improve response times for the solution to a problem.

Keywords: Web application, Documents, Recovery, Custody, Police

Tabla de Contenido

Listado de Tablas

Lista de Figuras

Título del Proyecto

Diseño de una aplicación web como apoyo para recuperación de documentos perdidos por los ciudadanos, que se encuentran en custodia por la Policía Nacional - Metropolitana de Ibagué

Introducción

Actualmente en la ciudad de Ibagué no existe ningún sistema que permita a la ciudadanía encontrar sus documentos perdidos que se encuentran en custodia de la policía nacional de Ibagué.

Este proyecto tiene como propósito facilitar a los ciudadanos encontrar sus documentos extraviados que se encuentran en los caí o subestaciones de la policía y a la vez ayudar a la policía con la gestión de esta información.

A través de una investigación realizada (encuestas y entrevistas) se valida la importancia, necesidad y viabilidad del proyecto el cual será aplicado en la ciudad de Ibagué pero que se puede llegar a implementar a nivel nacional.

Objetivos

Objetivo General

Diseñar una aplicación web que sirva como apoyo para recuperación de documentos perdidos por los ciudadanos, que se encuentran en custodia por la Policía Nacional- Metropolitana de Ibagué

Objetivos Específicos

Identificar la pertinencia de la aplicación móvil como apoyo para la recuperación de documentos perdidos por los ciudadanos, que se encuentran en Custodia por la Policía Nacional – Metropolitana de Ibagué

Reconocer cuales son los Tipos de documentos, la cantidad de documentos y la distribución de la ubicación de donde se encuentran en custodia por la Policía Nacional- Metropolitana de Ibagué

Determinar los elementos básicos para el desarrollo de la APP enfocada a identificar la localización de los documentos en custodia por la Policía Nacional- Metropolitana de Ibagué.

Diseñar la interfaz usuario de la APP de acuerdo a las Necesidad del proyecto.

Implementar la APP en el sistema operativo más adecuado de acuerdo a la interfaz de usuario desarrollada.

Justificación

El Ministerio de las TIC y las entidades territoriales están enfocadas en fortalecer y fomentar el desarrollo tecnológico e investigativo, mediante la formulación de planes, políticas y estrategias, en busca de una evolución positiva sobre diferentes situaciones que puedan afectar a la población colombiana; como resultado de estos desafíos se espera obtener mejoras en los elementos o entorno que lo integran los inconvenientes encontrados.

Uno de los puntos centrales del avance tecnológico es el desarrollo de software que puedan beneficiar a las comunidades o a grupos focales articuladas con las necesidades de cada territorio, ya que los conocimientos humanos se están digitalizando y son accesibles a través del internet, lo que implica ampliar las fronteras del diseño digital

En el momento actual la Información se ha considerado como materia prima para producir y generar nuevos conocimientos, mediante la innovación continua y permanente de amplios desarrollos tecnológicos, internet de las cosas, realidad aumentada entre otros. La sociedad de la Información enfrenta una revolución tecnológica que demandas nuevas dinámicas y nuevos retos en la forma como se interactúa con el ambiente que nos rodea, despertando en la conciencia humana la necesidad de contenidos dinámicos y llamativos. De tal manera que el auge de las TIC se ha perfeccionado gracias a la telefonía móvil permitiendo un salto tecnológico a las personas que cada día se adaptan a tecnologías más avanzadas.

Las TIC son tecnologías que constituyen nuevos canales, ofreciendo la posibilidad de crear nuevos entornos comunicativos que permiten desarrollar nuevas experiencias, pues a través de la innovación se apunta al correcto funcionamiento y evolución de la sociedad por medio de la focalización del usuario en la Web 2.0 para consumir y producir. En Colombia, al igual que en otros países, las TIC se han convertido en una herramienta predominante para

el desarrollo de la economía y la sociedad, pues son de la eficiencia y la competitividad de otros sectores; siendo un recurso indispensable para entrar para competir en un mercado mundial.

Las TIC permiten acceder a una gran cantidad de información de manera eficiente y eficaz, lo que ha facultado poner en práctica diferentes tipos de estrategias en diferentes áreas como por ejemplo en la educación. Esto ha dado múltiples posibilidades para las personas de utilizar diferentes herramientas tecnológicas para el desarrollo de sus habilidades cognitivas.

El mundo digital ha permitido que se cierre la brecha del acceso que se tiene para la educación, tanto así que los niños de hoy en día, que son llamados nativos digitales, porque utilizan las TIC mejor que muchos adultos. De esto nace su necesidad continua el uso de este tipo de herramientas digitales, por lo que se ha visto la necesidad de implementarlas en el momento de ver un impacto positivo en la educación que tienen en la actualidad.

En Colombia, de acuerdo a la verificación de la pertinencia del proyecto existen una herramienta que cumple con la función de la identificación y localización de documentos perdidos, sin embargo esta es una aplicación que para el ciudadano que está en la búsqueda de sus documentos tiene un costo, y que aun haciendo el pago del mismo, no le da la garantía de que los documentos perdidos o robados los pueda encontrar en esa base de datos, lo cual, para el ciudadano incurría en costos que los pudiera estar pagando para la generación de nuevos documentos y no para la recuperación de los mismos, de igual forma la Policía Metropolitana dentro de sus funciones, está la de mantener en custodia este tipo de documentos, sin embargo al verificar las herramientas tecnológicas que ellos tienen disponible, como A denunciar, o la app para conocer el Código nacional de seguridad y seguridad ciudadana, no existe una herramienta como la que se plantea este proyecto que busca beneficiar tanto la policía, para la entrega de los documentos, como para el ciudadano para evitar costos para su recuperación y ahorro en tiempo, para poder ser identificado en

cualquier entidad por este tipo de documentos válidos.

Planteamiento del Problema

Hoy en día, los ciudadanos por múltiples causas como descuido, robo entre otros aspectos, pierden sus documentos de identificación, lo cual genera traumatismo para su respectiva identificación ante diferentes entidades oficiales públicas y/o privadas, adicional que el costo para la recuperación de los mismo, es elevado, al igual que tiempo para poder recuperar los mismos puede oscilar entre 1 a 3 meses en el caso de la cedula, y demás documentos que entidades privadas genera costos, de acuerdo a Información brindada por la Policía Metropolitana de Ibagué ellos cuenta con 4 estaciones de policía, y en cada una de ella disponen de 26 comandos de Policía, donde en cada uno de ellos tienen en custodia múltiples documentos como cedulas, tarjetas de débito, tarjetas de crédito, carnet de EPS, entre otros documentos, los cuales muchos ciudadanos desconoce que ellos tienen estos documentos en custodia, sin embargo no hay una herramienta tecnológica que permita una interfaz entre el ciudadano y la Policía Metropolitana e Ibagué, para la recuperación de estos documentos a los dueños de los mismos.

Marcos de Referencia

Marco Legal

Ley 962 de 2005

Por la cual se dictan disposiciones sobre racionalización de trámites y procedimientos administrativos de los organismos y entidades del Estado y de los particulares que ejercen funciones públicas o prestan servicios públicos (G.C, 2005).

Esta ley tiene como objetivo incentivar el fortalecimiento tecnológico permitiendo a los ciudadanos acceder a los servicios públicos de manera más efectiva evitando generar costos y pérdida de tiempo innecesarios.

Decreto Anti trámites 019 de 2012

Por el cual se dictan normas para suprimir o reformar regulaciones, procedimientos y trámites innecesarios existentes en la Administración Pública (S/F, 2012).

Con este decreto el objetivo es agilizar los trámites a los ciudadanos garantizando la efectividad de los derechos de las personas ante la administración pública, suprimiendo o reformando los trámites que son innecesarios existentes en la administración.

Ley 1163 de 2007

Por la cual se regulan las tasas por la prestación de servicios de la Registraduría Nacional del Estado Civil y se dictan otras disposiciones (S/F, 2007).

Esta ley tiene como objetivo regular las tasas de prestación de servicios de expedición física del duplicado, ratificación, pérdida o deterioro de documentos de identidad de los ciudadanos, en la cual se mencionan los costos de los duplicados de documentos de identidad y los trámites que son exonerados de cobro.

Marco Teórico

Los sistemas de información documental: Consideraciones sobre sus características, concepto y funciones.

Menciona la importancia de los sistemas de información para el manejo y organización de la información, facilitando su acceso, solucionando problemas informáticos y la toma de decisiones.

Gracias a los ordenadores la información se ha convertido en el sector predominante, para los sistemas basados en la recuperación estos deben realizar diferentes funciones entre ellos es ofrecer información existente y pertinente a través de identificación de contenido haciendo accesible la información para el usuario, brindar la información necesaria, que sea comprensible y fácil de consumir.

Procedimiento para la estructuración y almacenamiento de documentos en el Sistema de Recuperación de Información Orión

En cuba se desarrolló una herramienta llamada Orión usada en la red universitaria con el objetivo de recuperar y visualizar información alojada en la web cubana, esta idea nació debido a la gran cantidad de información que se puede encontrar ahora en internet, (textos, imágenes, audios, videos, etc) lo cual puede dificultar la búsqueda de la misma.

Orión permite a los usuarios realizar búsquedas de manera eficiente, permitiendo el rápido acceso a la información publicada en la intranet de la universidad.

Zapata y otros (2020) *"Diseño de una App, para la Gestión de Documentos y Control de Actividades Enfocada a Trabajo Comunitario, para los Estudiantes de la Carrera de Pedagogía de las Ciencias Experimentales Informática, de la Facultad de Filosofía, Letras y Ciencias de la Educación, de la Universidad Central del Ecuador"*

En su proyecto presentan permite describir los problemas encontrados en el proceso de Trabajo Comunitario y a su vez el conocimiento que los estudiantes tienen sobre ello, además de los proyectos sociales existentes para esto se aplicó una encuesta a un total de 84 estudiantes de la carrera de Pedagogía de las Ciencias Experimentales Informática, de la Facultad de Filosofía, Letras y Ciencias de la Educación de la Universidad Central del Ecuador, dicha encuesta consta de 15 ítems entre los cuales están preguntas dicotómicas y de selección múltiple que permitió conocer sobre la realidad evidenciada por los estudiantes acerca de Trabajo Comunitario. La investigación se elaboró con un enfoque cuantitativo y un nivel descriptivo en el cual también se realizó una investigación documental previa a la realización de la propuesta tecnológica. La información obtenida permitió elaborar un análisis estadístico, que llevo a concluir que los estudiantes tienen un conocimiento y acceso a la información de Trabajo Comunitario deficiente además de no recibir una concientización acerca de su importancia para la formación académica y profesional, de no existir una solución a los problemas encontrados en la investigación la deficiencia seguirá latente y generando un efecto negativo tanto en estudiantes como en la sociedad.

Portocarrero (2018) ***"Sistema para la recuperación de documentos de identificación institucional a partir del reconocimiento de patrones"***

En su proyecto presenta un proyecto de investigación basada en el estudio de las técnicas necesarias para llevar a cabo la implementación de dichas APIs usando librerías de código abierto para la clasificación como transformación de imágenes y la extracción de texto. Los objetivos planteados buscan desarrollar un estudio comparado de los algoritmos más representativos, definir un modelo para la indexación de información no estructurada de Facebook, desarrollar un sistema automático basado en el diseño planteado y determinar los grados de precisión en las implementaciones de los algoritmos.

Cascon, López (2019) "**A*plicación de SIGs (georreferenciación y geolocalización) para mejorar la recuperación de la documentación histórica gráfica"***

En su proyecto presentan es la creación de un geo portal donde los potenciales usuarios puedan disfrutar, de forma fácil, cómoda e intuitiva, de parte del patrimonio histórico cartográfico y fotográfico de la ciudad de Gra-nada, centrándose principalmente en lo referente a urbanismo y paisaje. La recuperación de este tipo de documentación hoy en día no es efectiva en otras webs institucionales, en perfiles de redes sociales o en webs personales. A través de la metodología adecuada que incluye: localización, selección, digitalización, descripción, georreferenciación (cartografía) y geolocalización (fotografía), teselación y publicación web, se ha obtenido como resultado un geo portal o visualizador carto-gráfico donde poder hacer búsquedas textuales tradicionales mejoradas con filtros, búsquedas geográficas más intuitivas, comparar el pasado y el presente entre los documentos, y poner en relación unos documentos con otros.

Arbeláez. (2014). **"Las tecnologías de la información y la comunicación (TIC) un instrumento para la investigación. *Investigaciones"***

Sin duda alguna, las Tecnologías de la Información y la Comunicación (TIC) han transformado de manera vertiginosa la vida cotidiana y social de los seres humanos, algunos ejemplos están en el uso de los teléfonos móviles, los computadores, el internet y sus herramientas de comunicación, la televisión digital, aplicaciones como Google earth, Google maps, museos virtuales, entre otros, que nos permiten conocer un lugar sin haber estado físicamente en él.

Díaz, Pérez, & Florido. (2011). **"*Impacto de las tecnologías de la información y las comunicaciones (tic) para disminuir la brecha digital en la sociedad actual*".**

En los últimos años, la evolución del concepto de formación se ha visto deslumbrado por la aparición y consolidación de las tecnologías de la información y las comunicaciones

(TIC), Internet y su realización en la World Wide Web ha facilitado el acceso a todo tipo de información necesaria, provocando un aumento considerable de la interactividad entre personas de distintas continentes y países del mundo, brindando la posibilidad de desarrollar sus capacidades y habilidades para el tele trabajo, la interacción multicultural, el acceso a la información, al conocimiento y la educación con el objetivo de disminuir la brecha digital, siendo este el tema abordado en la presente reseña bibliográfica.

Marco Conceptual

Conservación de Archivos: Conjunto de medidas adoptadas para garantizar la integridad física de los documentos que alberga un archivo. (Guía Para La Conservación De Documentos En Soportes Físicos, 2021)

Conservación de documentos: Conjunto de medidas preventivas o correctivas adoptadas para asegurar la integridad física y funcional de los documentos de archivo. (Guía Para La Conservación De Documentos En Soportes Físicos, 2021)

Conservación preventiva de documentos: Conjunto de estrategias y medidas de orden técnico, político y administrativo orientadas a evitar o reducir el riesgo de deterioro de los documentos de archivo, preservando su integridad y estabilidad. (Guía Para La Conservación De Documentos En Soportes Físicos, 2021)

Custodia de documentos: Guarda o tenencia de documentos por parte de una institución o una persona, que implica responsabilidad jurídica en la administración y conservación de los mismos, cualquiera que sea su titularidad. (Guía Para La Conservación De Documentos En Soportes Físicos, 2021)

Depósito de archivo: Local especialmente equipado y adecuado para el almacenamiento y la conservación de los documentos de archivo. (Guía Para La Conservación De Documentos En Soportes Físicos, 2021)

Digitalización: Técnica que permite la reproducción de información que se encuentra guardada de manera analógica (Soportes: papel, video, casettes, cinta, película, microfilm y otros) en una que sólo puede leerse o interpretarse por computador. (DIGITALIZACIÓN, s. f.).

Disposición final de documentos: Decisión resultante de la valoración hecha en cualquier etapa del ciclo vital de los documentos, registrada en las tablas de retención y/o tablas de valoración documental, con miras a su conservación total, eliminación, selección y/o reproducción. (DISPOSICIÓN FINAL DE DOCUMENTOS, s. f.)

Documento Electrónico: Es la información generada, enviada, recibida, almacenada y comunicada por medios electrónicos, ópticos o similares. (Gonzáles C, 2021)

Documento

Un documento es una evidencia sobre un acontecimiento o situación. Puede tratarse de un texto redactado en un papel o un archivo guardado en un soporte electrónico.

Un documento puede servir para dejar constancia de un hecho relevante. Por ejemplo, si una persona finaliza sus estudios universitarios puede solicitar a la entidad educativa un escrito con todas las materias cursadas y las notas obtenidas. (Westreicher G, 2020)

Pasaporte

Este documento sirve como garantía para la acreditación de la identidad y nacionalidad de las personas de cada país, permitiendo viajar de manera internacional.

El pasaporte tiene un aspecto físico similar en todos los Estados y siempre contiene datos personales como nombre, apellidos, número individualizado o fotografía.

Cuando se viaja a otros países, en el documento del pasaporte se pone un sello donde queda acreditada la visita de ese país, con una finalidad de control fronterizo. (Trujillo E, 2020)

Tarjeta de identidad

La tarjeta de identidad es el documento oficial que hace las veces de identificación para los menores de edad entre los 7 y los 18 años. Tiene un número único y será el que otorgue la distinción y servirá para adelantar diferentes procedimientos diarios. (¿Qué es la tarjeta de identidad?, 2016)

Metodología

La presente investigación es de enfoque cualitativo y cuantitativo donde un estudio cuantitativo se basa en otras investigaciones previas y el estudio cualitativo se fundamenta primordialmente en sí mismos, Profundiza en las mismas, combina diferentes técnicas, mejorando la comprensión del problema, mejora la creatividad. Hernández, R; Fernández, C; Baptista, M, (2010) "El primero se utiliza para consolidar las creencias, formuladas de manera lógica en una teoría o un esquema teórico, y establecer con exactitud patrones de comportamiento en una población y el segundo para construir creencias propias sobre el fenómeno estudiado". Es decir que el enfoque cualitativo tendrá en cuenta cuales son las temáticas más relevantes sobre ciudades inteligentes y los elementos necesarios para su desarrollo.

El alcance del presente proyecto es de tipo descriptiva, ya que se desarrollará basado en un conjunto de procesos y procedimiento lógicos que permitirán identificar las características de la población. Los estudios descriptivos buscan desarrollar una imagen o fiel representación (descripción) del fenómeno estudiado a partir de sus características. Describir en este caso es sinónimo de medir. Miden variables o conceptos con el fin de especificar las propiedades importantes de comunidades, personas, grupos o fenómeno bajo análisis. El énfasis está en el estudio independiente de cada característica, es posible que de alguna manera se integren las mediciones de dos o más características con el fin de determinar cómo es o cómo se manifiesta el fenómeno. Pero en ningún momento se pretende establecer la forma de relación entre estas características. En algunos casos los resultados pueden ser usados para predecir.

En cuanto al diseño es una investigación de tipo no experimental porque no manipula intencionalmente variables, ya que no se tiene control directo sobre ellas. Este tipo de proyecto es un diseño transeccional descriptivo que tiene como objeto indagar la incidencia

de las modalidades, que consiste en ubicar un grupo de personas y unas situaciones y de esta manera describir sus sucesos.

Procedimiento que se llevó a cabo:

1. Análisis de los factores
2. Evaluación Tecnológica
3. Socialización de Resultados.
4. Entrega de producto final.

Para el desarrollo del producto final se utilizó la metodología Agile tipo **Scrum** para el desarrollo de software que permitió la asignación de tareas diarias basado en reuniones rápidas de manera presencial y virtual a través de Skype para llevar un control de la evolución de los procesos, mediante la herramienta del Drive. De esta manera además de las tareas asignadas mediante la metodología de **SCRUM** se pudieron incrementar de acuerdo a las necesidades que se iban presentando, y esto coincide exactamente con el devenir normal del desarrollo de software.

Hipótesis

Si diseñamos y desarrollamos una aplicación web para la recuperación y administración de documentos perdidos, entonces la población podrá informarse de manera más sencilla donde ubicar su documento y la policía nacional podrá administrar la información de manera más eficiente logrando optimizar los procesos.

Técnicas e Instrumentos de Recolección de Información

Para la recolección de la información de este proyecto el instrumento que se utiliza es la Encuesta a policías pertenecientes a las diferentes estaciones de policía de la ciudad de Ibagué, los cuales son quienes se encargan del proceso de recibir los documentos extraviados

y entregarlos a las personas que los reclaman en los caís, lo cual nos permite tener mejor claridad sobre las necesidades de la población objeto de estudio.

Población

La población de la aplicación web como apoyo para recuperación de documentos perdidos por los ciudadanos, que se encuentran en custodia por la policía metropolitana de Ibagué son los diferentes estaciones que se encuentran en las 13 comunas de la ciudad de Ibagué

Tipo De Investigación

Línea de Investigación

Programación y desarrollo de aplicaciones: El objetivo de este proyecto es

diseñar una aplicación web que sirva como apoyo para recuperación de documentos perdidos por los ciudadanos, que se encuentran en custodia por la Policía Nacional- Metropolitana de Ibagué

Alternativa de Trabajo de Grado

Proyecto Aplicado en la modalidad de proyecto de desarrollo tecnológico: El diseño y desarrollo de la aplicación web como apoyo para recuperación de documentos perdidos por los ciudadanos, que se encuentran en custodia por la Policía Nacional- Metropolitana de Ibagué, se realizara soportado en la opción de grado proyecto aplicado modalidad proyecto de desarrollo tecnológico dando solución a una problemática que se presenta con los documentos extraviados en la ciudad de Ibagué mediante el desarrollo de una aplicación web.

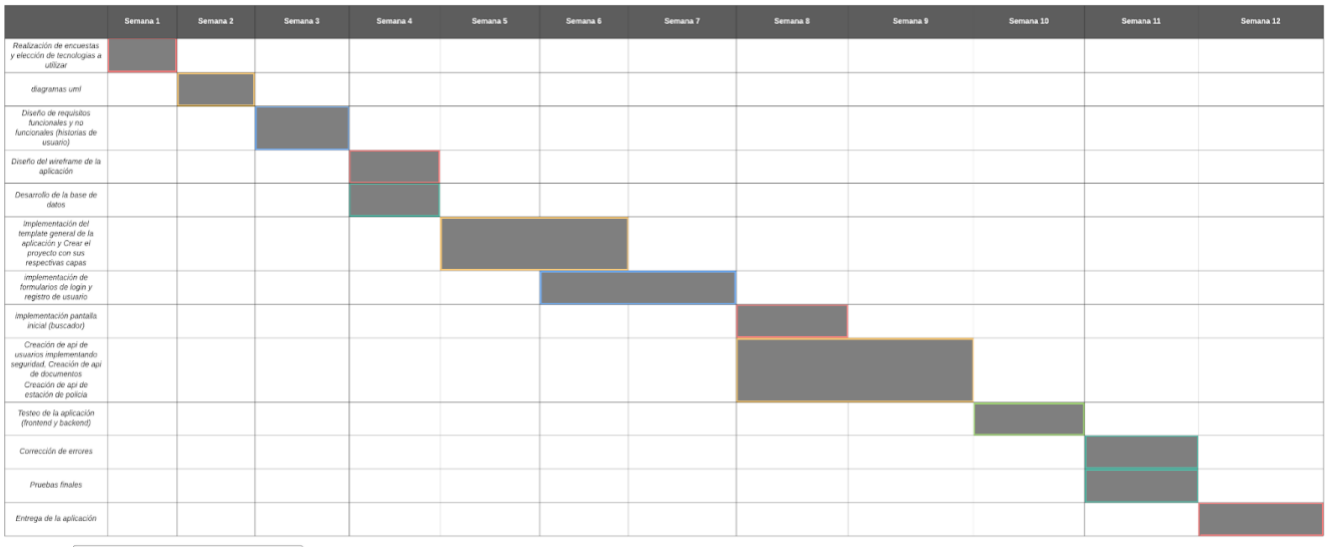
Presupuesto

Tabla 1

Presupuesto para el desarrollo del proyecto

RECURSO	DESCRIPCIÓN	PRESUPUESTO
Equipo Humano	Estudiantes, Policía metropolitana de Ibagué, Docente	$200.000
Equipos y Software	Licencia Software para la creación de la Aplicación, Espacio en la Nube para BD	$500.000
Servicios públicos	Agua, energía, internet	$175.000
Bibliografía	Libros temáticos para la implementación de la solución	$100.000
TOTAL		**$975.000**

Cronograma de Actividades



Fases del Proyecto

Fase de Análisis

De acuerdo a la problemática identificada en el proceso para la administración de los documentos extraviados de la policía nacional – metropolitana de Ibagué, se procedió a realizar la recolección de información a través de una encuesta la cual se a las diferentes estaciones de la policía en Ibagué.

Gracias a la información recolectada se procedió a realizar un análisis de todos los resultados obtenidos en la encuesta con el fin de determinar la importancia y viabilidad de la propuesta en la solución de la problemática

A continuación, se muestran los resultados de la información recolectada en la encuesta realizada:

Encuestas

Se realizaron encuestas a los policías para conocer su experiencia en el proceso de recolección y entrega de documentos extraviados.

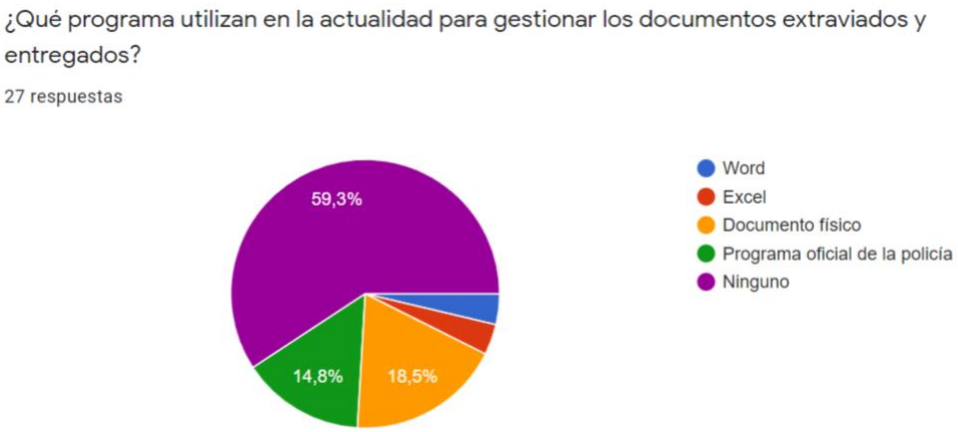
Figura 1
Resultado de Encuestas 1

Análisis Figura 1.

Las consultas por parte de los ciudadanos sobre documentos extraviados se encuentran en los diferentes rangos de temporalidad entre diario y semanal con un porcentaje del 74% de necesidades de ciudadanos en la búsqueda de documentos extraviados.

Figura 2

Resultado de Encuestas 2

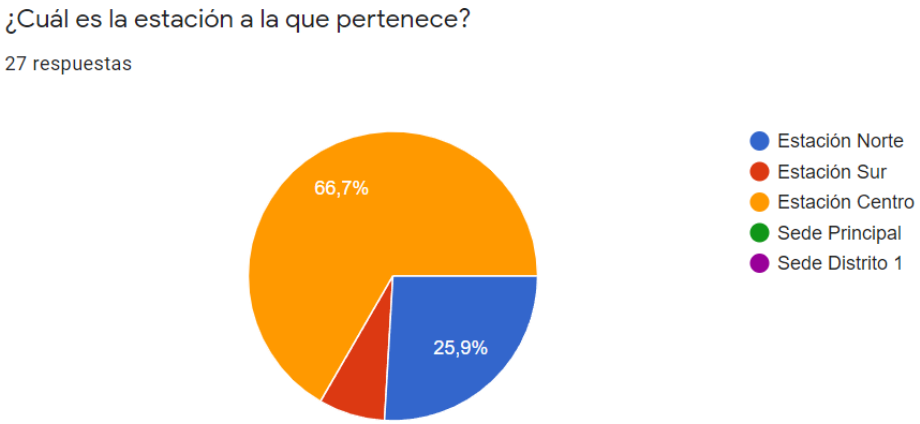


Análisis Figura 2.

El 59.3% de los encuestados no usan ningún programa para gestionar las entregas de documentos extraviados, otro 18.5% realiza la gestión con documentos físicos
Y otro 7.4% usa Word o Excel.

Figura 3

Resultado de Encuestas 3

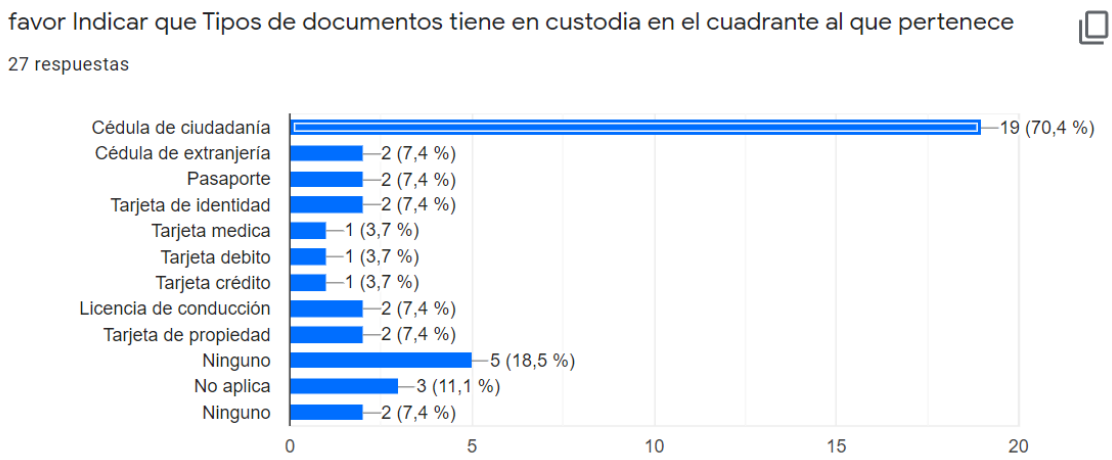


Análisis Figura 3.

La mayoría de documentos extraviados se encuentran en custodia de la estación centro con un porcentaje del 66.7%, el 25.9% en la estación Norte y el otro 7.4% a la estación sur.

Figura 4

Resultado de Encuestas 4

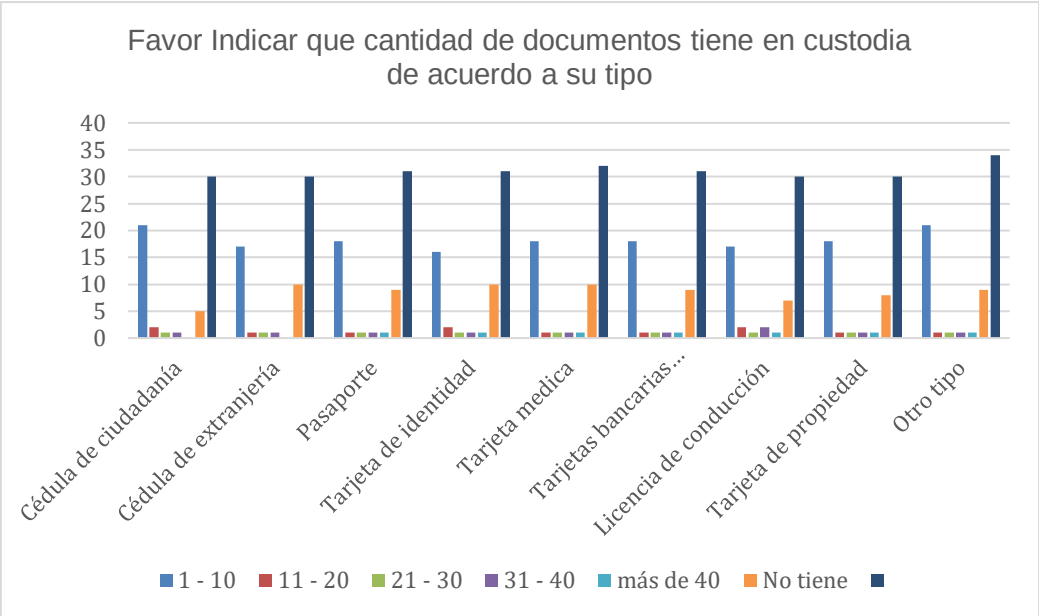


Análisis Figura 4.

De acuerdo a los resultados obtenidos el 70.4%, es decir, la mayoría de los documentos extraviados son cédulas de ciudadanía y en menor cantidad encontramos, cedula de extranjería, pasaportes, tarjetas de identidad, licencias de conducción y tarjetas de propiedad con un 7.4% cada uno.

Figura 5

Resultado de Encuestas 5



Análisis Figura 5.

De acuerdo a los resultados obtenidos el 70.4%, es decir, la mayoría de los documentos extraviados en custodia de la policía son cédulas de ciudadanía.

Fase de Diseño

Descripción del Sistema Propuesto

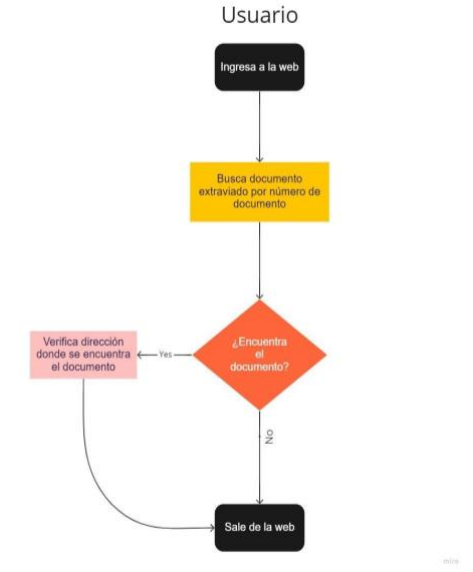
Es un sistema que permite a la población verificar si en algún Caí o sub estación se encuentran sus documentos extraviados, en caso de que se encuentre en dichos lugares, el ciudadano podrá ver la ubicación del mismo. A la policía le permite tener el apoyo tecnológico para poder exponer dicha información a los ciudadanos y obtener estadísticas respecto a los documentos reclamados.

Diagrama de Flujo

Nos permite identificar el flujo desde que el usuario ingresa a la aplicación web para consultar un documento extraviado hasta salir de la aplicación, teniendo dos posibles resultados: Si el documento se encuentra en algún cai o sub estación, el sistema muestra la ubicación de este, y si no se encuentra sale de la aplicación

Figura 6

Diagrama de flujo usuario

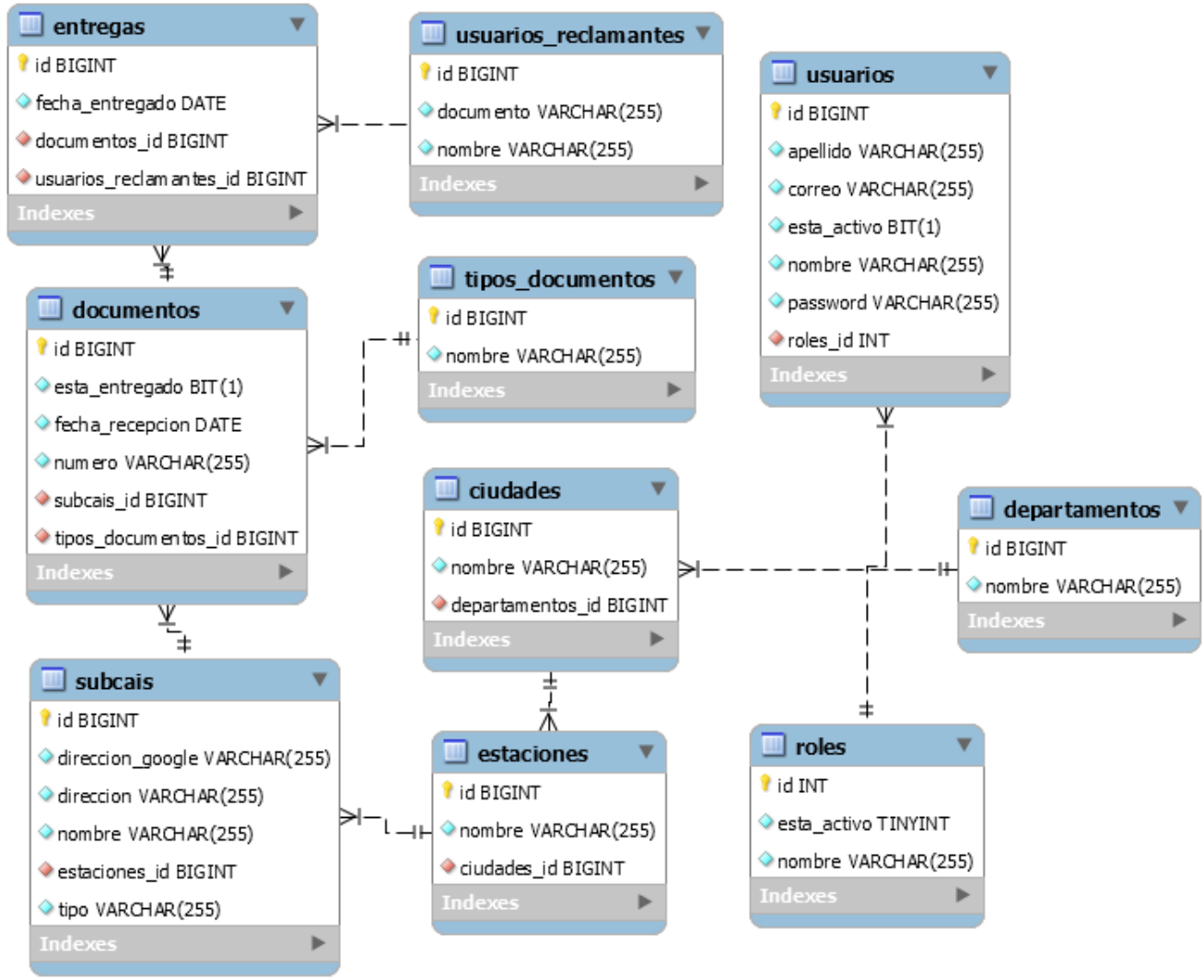


Modelo Entidad Relación

Este diagrama representa las tablas, sus atributos y relaciones existentes en el sistema para la persistencia de la información.

Figura 7

Diagrama Entidad – Relación



Diccionario de Datos

Tabla 2

Para el registro de usuarios

USUARIOS

Campos	Diccionario de datos	Restricción	Tipo de dato	Tipo de llave
id	Identificador de registro	Not null	Int	Primary key
nombre	Nombre de usuario	Not null	Varchar (200)	
apellido	Apellido usuario	Not null	Varchar (200)	
correo	Correo de usuario	Not null	Varchar (250)	
password	Contraseña usuario	Not null	Varchar (200)	
esta_activo	Estado del usuario (activo o inactivo)	Not null	Boolean - Tinyint	
Roles_id	Rol del usuario	Not null		Foreign Key

Tabla 3

Para el registro de roles

ROLES

Campos	Diccionario de datos	Restricción	Tipo de dato	Tipo de llave
id	Identificador de registro	Not null	Int	Primary key
nombre	Nombre del rol	Not null	Varchar (15)	
esta_activo	Estado del rol (activo o inactivo)	Not null	Boolean - Tinyint	

Tabla 4

Para el registro de departamentos

DEPARTAMENTOS

Campos	Diccionario de datos	Restricción	Tipo de dato	Tipo de llave

id	Identificador de registro	Not null	Int	Primary key
nombre	Nombre del departamento	Not null	Varchar (100)	

Tabla 5

Para el registro de ciudades

CIUDADES

Campos	Diccionario de datos	Restricción	Tipo de dato	Tipo de llave
id	Identificador de registro	Not null	Int	Primary key
nombre	Nombre de la ciudad	Not null	Varchar (100)	
Departamentos_id	Correo de usuario	Not null	Int	Foreign Key

Tabla 6

Para el registro de estaciones

ESTACIONES

Campos	Diccionario de datos	Restricción	Tipo de dato	Tipo de llave
id	Identificador de registro	Not null	Int	Primary key
nombre	Nombre de usuario	Not null	Varchar (100)	
Ciudades_id	Correo de usuario	Not null	Int	Foreign Key

Tabla 7

Para el registro de cai-subcais

SUBCAIS

Campos	Diccionario de datos	Restricción	Tipo de dato	Tipo de llave
id	Identificador de registro	Not null	Int	Primary key
nombre	Nombre del cai	Not null	Varchar (255)	
Direccion_google	Dirección de google del cai	Not null	Varchar (255)	
direccion	Dirección del cai o subcai	Not null	Varchar (255)	

tipo	Cai o subcai	Not null	Varchar (255)	
Estaciones_id	Rol del usuario	Not null	Int	Foreign Key

Tabla 8

Para el registro de tipos de documento

TIPO_DOCUMENTOS

Campos	Diccionario de datos	Restricción	Tipo de dato	Tipo de llave
id	Identificador de registro	Not null	Int	Primary key
nombre	Nombre del tipo de documento	Not null	Varchar (60)	

Tabla 9

Para el registro de documentos

DOCUMENTOS

Campos	Diccionario de datos	Restricción	Tipo de dato	Tipo de llave
id	Identificador de registro	Not null	Int	Primary key
numero	Número del documento	Not null	Varchar (255)	
fecha_recepcion	Fecha en la que se recibió el documento extraviado	Not null	Date	
esta_entregado	Estado del documento (activo o inactivo)	Not null	Boolean - Tinyint	
Subcais_id	Cai donde se encuentra el documento	Not null	Int	Foreign Key
Tipos_documentos_id	Tipo de documento	Not null	Int	Foreign Key

Tabla 10

Para el registro de usuarios reclamantes

USUARIOS_RECLAMANTES

Campos	Diccionario de datos	Restricción	Tipo de dato	Tipo de llave

id	Identificador de registro	Not null	Int	Primary key
documento	documento de usuario que reclama	Not null	Varchar (255)	
nombre	Nombre de usuario que reclama	Not null	Varchar (255)	

Tabla 11

Para el registro de entregas

ENTREGAS				
Campos	**Diccionario de datos**	**Restricción**	**Tipo de dato**	**Tipo de llave**
id	Identificador de registro	Not null	Int	Primary key
Fecha_entregado	Fecha de entrega del documento	Not null	Date	
Usuario_reclamantes_id	Usuario que reclama documento	Not null	Int	Foreign Key
Documentos_id	Documento extraviado	Not null	Int	Foreign Key

Fase de Diseño

Una vez establecida la recolección de información pertinente para el desarrollo de este proyecto se hace necesario construir el mockup de cada una de las plantillas y forma de trabajar de la aplicación como solución a la problemática presentada tanto para la policía como para el ciudadano.

Figura 8

Prototipo de inicio de sesión

Figura 9

Prototipo de lista usuarios (Rol administrador)

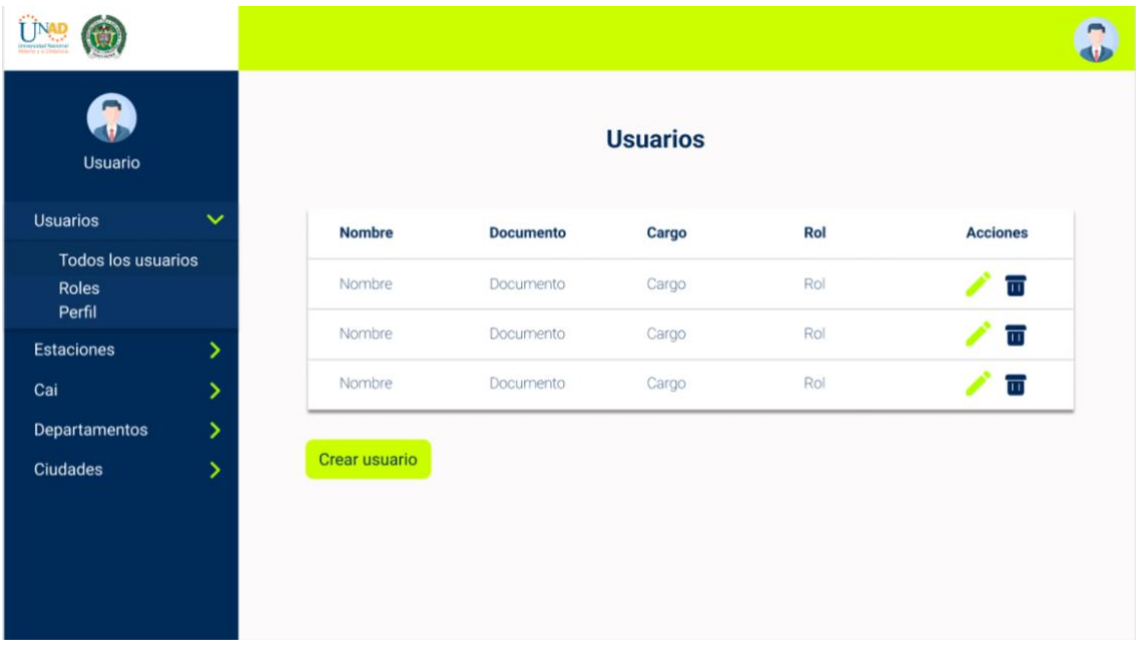


Figura 10

Prototipo de crear usuario (Rol administrador)

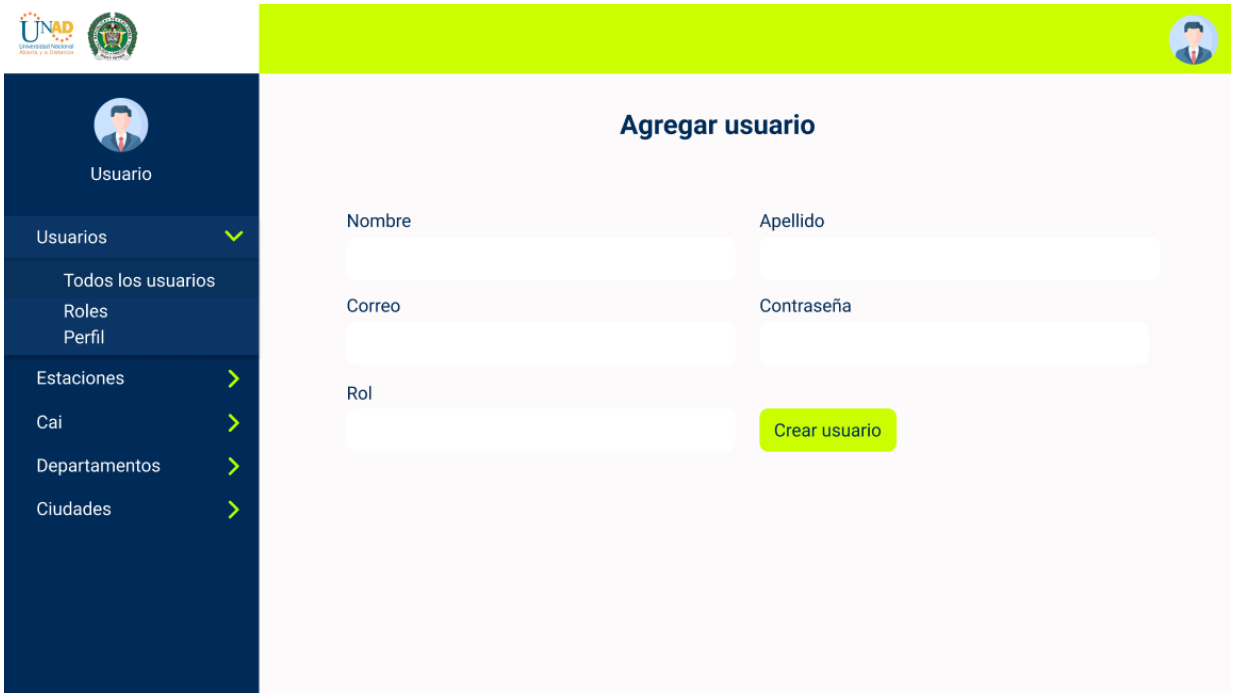


Figura 11

Prototipo inicio (Usuario admin)

Figura 12
Prototipo búsqueda documento (Usuario reclamante)

Requerimientos del Sistema

Para el desarrollo de la aplicación como apoyo para la recuperación de documentos extraviados, robados, y/o recuperados por los ciudadanos, el lenguaje de programación seleccionado es Java, el cual es uno de los lenguajes de programación más usado por grandes empresas de todo el mundo, algunas de las aplicaciones que fueron creadas en este lenguaje son: Twitter, Netflix, Uber las cuales son muy usadas hoy en día.

Java es un lenguaje con muchas ventajas las cuales se aprovecharán para el desarrollo de la aplicación, entre ellas están:

Multiplataforma

Java funciona en cualquier dispositivo o sistema operativo lo cual permite que muchas personas lo puedan usar sin problema.

Orientada a objetos

Gracias a que es orientado a objetos se podrá crear una aplicación modular, reutilizar código y hacer que la aplicación sea escalable, además ayuda a que las modificaciones sean fáciles y sencillas de realizar sin afectar el funcionamiento del sistema y permite utilizar patrones de diseño de acuerdo a lo que se requiera.

Fuertemente tipado

Ya que java es un lenguaje de tipado fuerte es más seguro ya que no se permite que una variable realice operaciones con otra de distinto tipo evitando que se comentan errores al desarrollar

Versátil

Con Java se pueden hacer muchas aplicaciones ya que permite la creación de aplicaciones web, de escritorio, móviles (Con Android Studio) y juegos (Minecraft).

Además, se utilizarán diferentes herramientas, librerías y framework que son tecnologías muy usadas para realizar una aplicación que sea simple, fácil de modificar y escalar, entre ellos están: Spring boot, hibérnate, JPA, Junit, Mockito, MySql, Rest, Log4j, swagger, Git, entre otros.

MySQL

Es un sistema para la gestión de bases de datos relacional lo que significa que los datos almacenados en el conjunto de datos son organizados en forma de tablas.

Angular

Para el desarrollo del Frontend (interface de usuario) se usará Angular, el cual es un lenguaje basado en la programación orientada a objetos lo cual permite que el sistema se pueda modularizar y sea fácil de modificar y escalar ya que usa componentes para desarrollar cada una de las funcionalidades del sistema.

Bootstrap

Es un framework de código abierto para el frontend el cual permite desarrollar aplicaciones responsivas y que le da estilos a la interface, es compatible con diferentes navegadores y es fácil de usar.

1.1.1. Historias de Usuario

En lugar de realizar requerimientos funcionales, de acuerdo a la metodología ágil nos basamos en las historias de usuario lo cual es una herramienta que tiene muchas fortalezas para expresar con un lenguaje más sencillo con el cual cualquier persona puede entender los requisitos del sistema y lo que se quiere lograr.

Como usuario administrador quiero poder ingresar al sistema para poder administrar las diferentes funcionalidades de este.

Como usuario administrador quiero poder registrar otros usuarios para que puedan ingresar y realizar determinadas tareas de acuerdo a su rol.

Como usuario administrador quiero poder editar la información como estaciones, sub estaciones y centros de atención al ciudadano en caso de que la información este incorrecta.

Como usuario quiero poder crear documentos extraviados para registrar la información y los ciudadanos puedan consultarla.

Como usuario quiero poder editar el documento para ser entregado al usuario reclamante.

Como usuario reclamante quiero poder acceder al sistema de manera fácil para poder buscar mi documento extraviado y verificar en qué lugar lo puedo reclamar.

Fase de Implementación

Seguridad

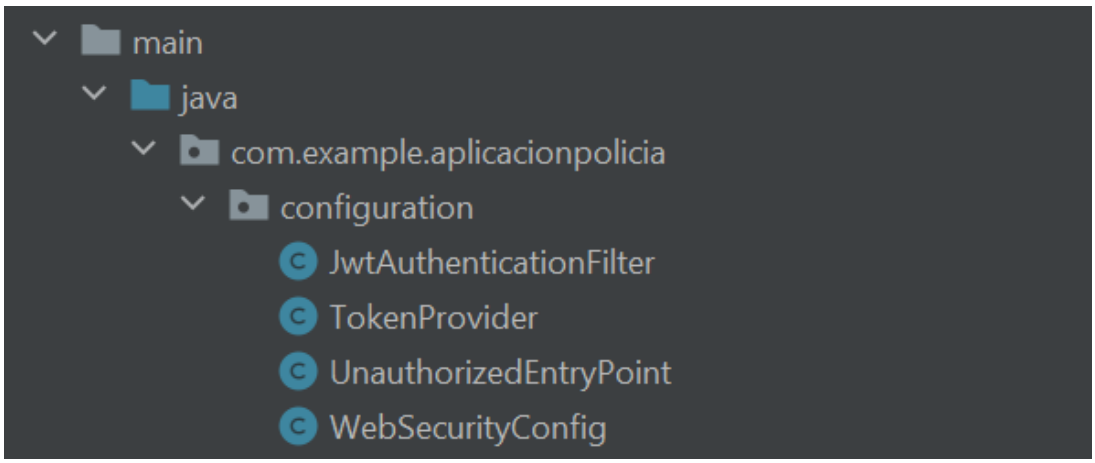
La seguridad para la primera versión que se entregará está basada en Spring Security utilizando JWT y respectivos roles para controlar los accesos de los usuarios a las acciones del sistema.

Este JWT está firmado por el backend para su autenticidad y validación.

Se aconseja que una vez la policía tome el desarrollo del mismo, se implemente un sistema de seguridad Oauth 2.0 que garantice una mayor escalabilidad y mayor seguridad en general con soporte para MFA.

En el frontend se utiliza el JWT para obtener la información del usuario y con el guard se restringe el acceso del usuario dependiendo su rol.

Figura 13
Seguridad implementada



Backend

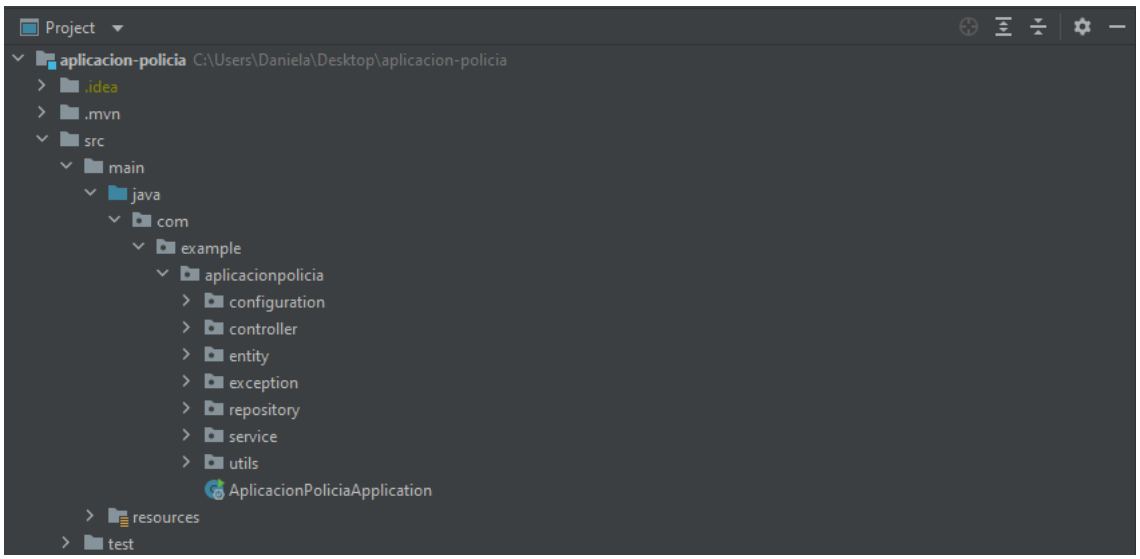
Para el backend utilizamos la arquitectura MVC (Modelo Vista Controlador) la cual permite separar cada capa del sistema con un rol especifico:

Modelo: Es donde se encuentra lo lógica de negocio y persistencia de los datos.

Vista: Se encuentra externalizada en angular

Controlador: Es el punto de entrada de la información que envía el usuario y se comunica con el servicio.

Figura 14
Estructura de la aplicación



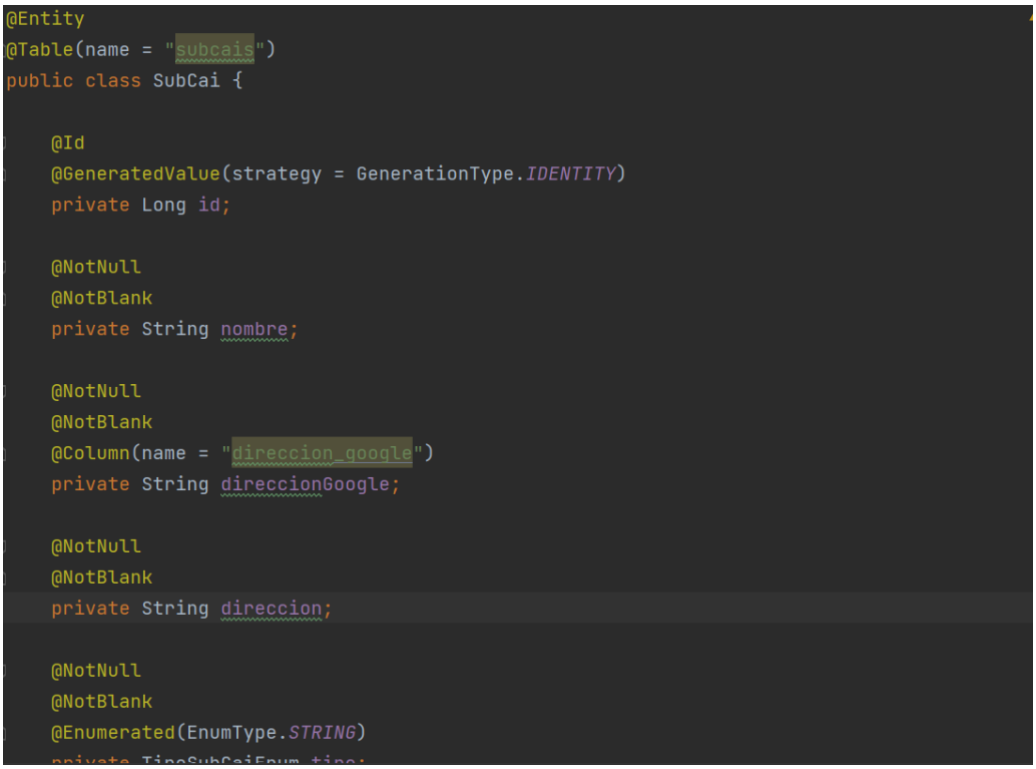
Modelo (Entidades)

Las entidades son la representación de las tablas en la base de datos los cuales se asocian a través de las anotaciones (@Table, @Colum @Entity, etc) utilizando las especificaciones JPA con el Orm Hibérnate.

Figura 15
Entidad Documento

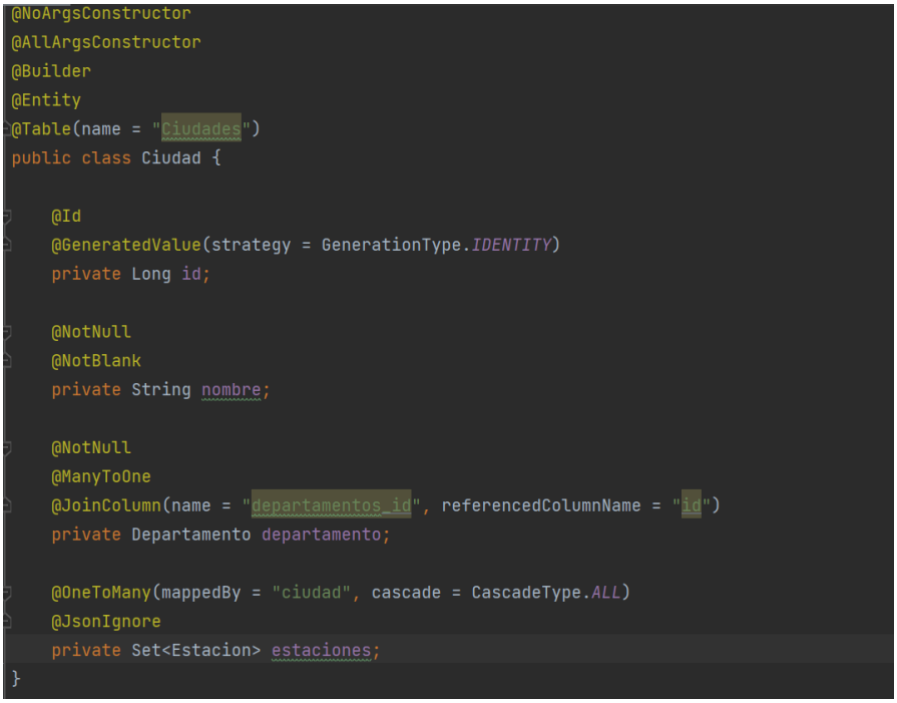
```java
@Entity
@Table(name = "Documentos")
public class Documento {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @NotNull
    @NotBlank
    @Column(unique = true)
    private String numero;

    @NotNull
    @Column(name = "fecha_recepcion")
    private LocalDate fechaRecepcion;

    @NotNull
    @Column(name = "esta_entregado")
    private Boolean estaEntregado;

    @NotNull
    @ManyToOne
    @JoinColumn(name = "subcais_id", referencedColumnName = "id")
    private SubCai subcai;
```

Figura 16
Entidad SubCai



```java
@Entity
@Table(name = "subcais")
public class SubCai {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @NotNull
    @NotBlank
    private String nombre;

    @NotNull
    @NotBlank
    @Column(name = "direccion_google")
    private String direccionGoogle;

    @NotNull
    @NotBlank
    private String direccion;

    @NotNull
    @NotBlank
    @Enumerated(EnumType.STRING)
```

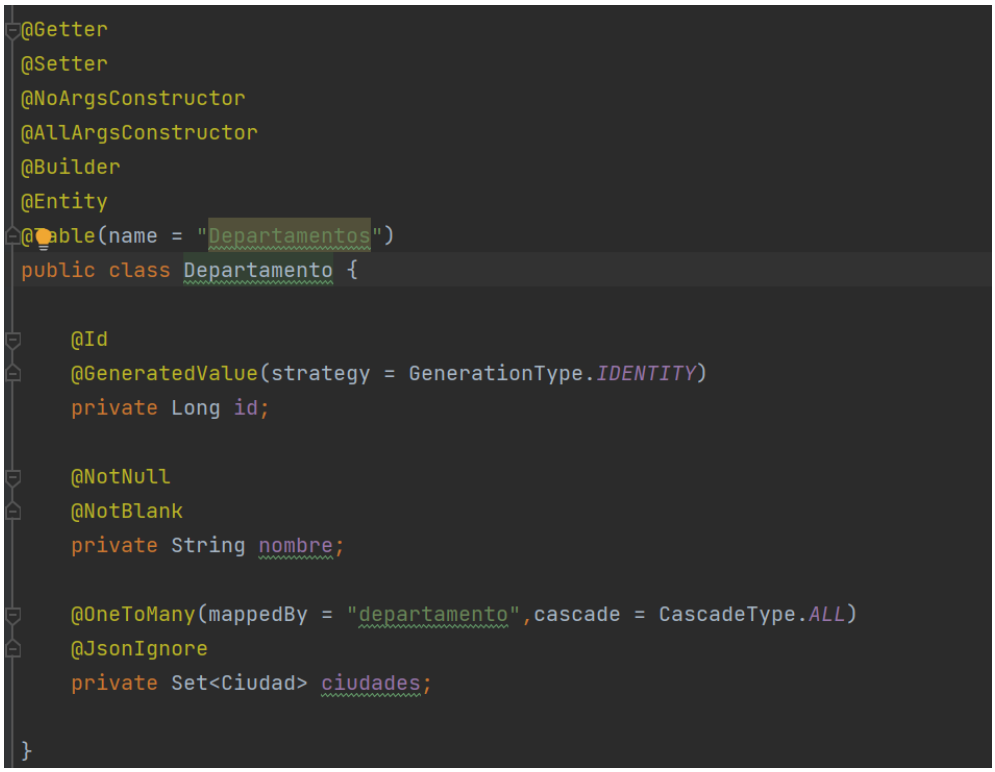
Figura 17
Entidad Ciudad



```java
@NoArgsConstructor
@AllArgsConstructor
@Builder
@Entity
@Table(name = "Ciudades")
public class Ciudad {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @NotNull
    @NotBlank
    private String nombre;

    @NotNull
    @ManyToOne
    @JoinColumn(name = "departamentos_id", referencedColumnName = "id")
    private Departamento departamento;

    @OneToMany(mappedBy = "ciudad", cascade = CascadeType.ALL)
    @JsonIgnore
    private Set<Estacion> estaciones;
}
```

Figura 18

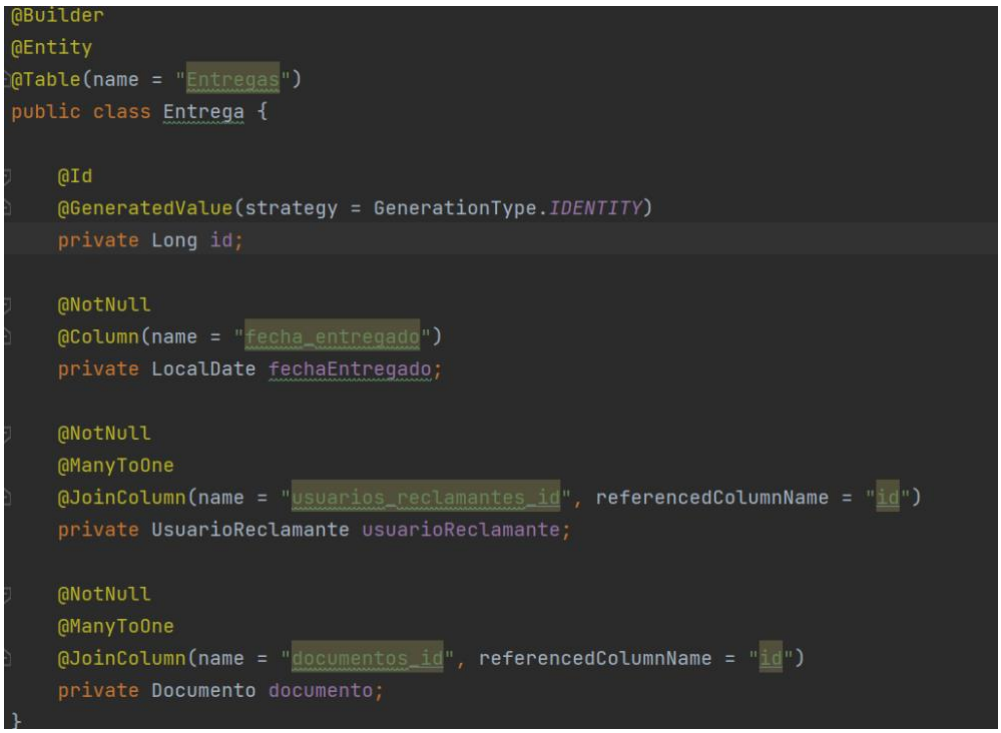
Entidad Departamento



```java
@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
@Builder
@Entity
@Table(name = "Departamentos")
public class Departamento {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @NotNull
    @NotBlank
    private String nombre;

    @OneToMany(mappedBy = "departamento",cascade = CascadeType.ALL)
    @JsonIgnore
    private Set<Ciudad> ciudades;

}
```

Figura 19

Entidad Entrega



```java
@Builder
@Entity
@Table(name = "Entregas")
public class Entrega {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @NotNull
    @Column(name = "fecha_entregado")
    private LocalDate fechaEntregado;

    @NotNull
    @ManyToOne
    @JoinColumn(name = "usuarios_reclamantes_id", referencedColumnName = "id")
    private UsuarioReclamante usuarioReclamante;

    @NotNull
    @ManyToOne
    @JoinColumn(name = "documentos_id", referencedColumnName = "id")
    private Documento documento;
}
```

Figura 20

Entidad Estación

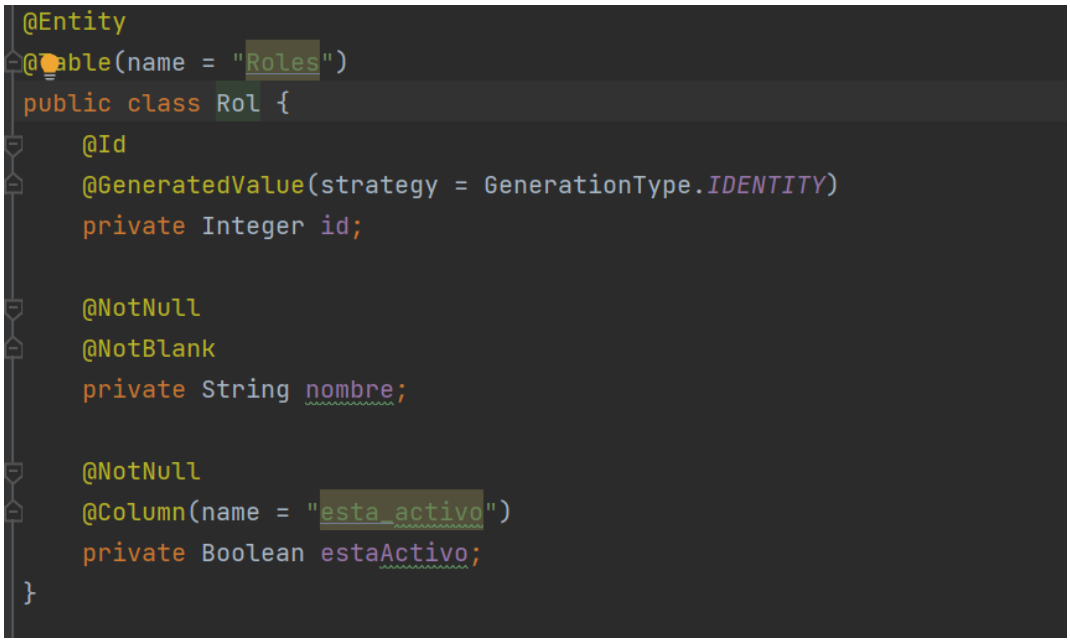
```java
@Table(name = "Estaciones")
public class Estacion {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @NotNull
    @NotBlank
    private String nombre;

    @NotNull
    @ManyToOne
    @JoinColumn(name = "ciudades_id", referencedColumnName = "id")
    private Ciudad ciudad;

    @OneToMany(mappedBy = "estacion", cascade = CascadeType.ALL)
    @JsonIgnore
    private Set<SubCai> subCais;
}
```

Figura 21

Entidad Usuario

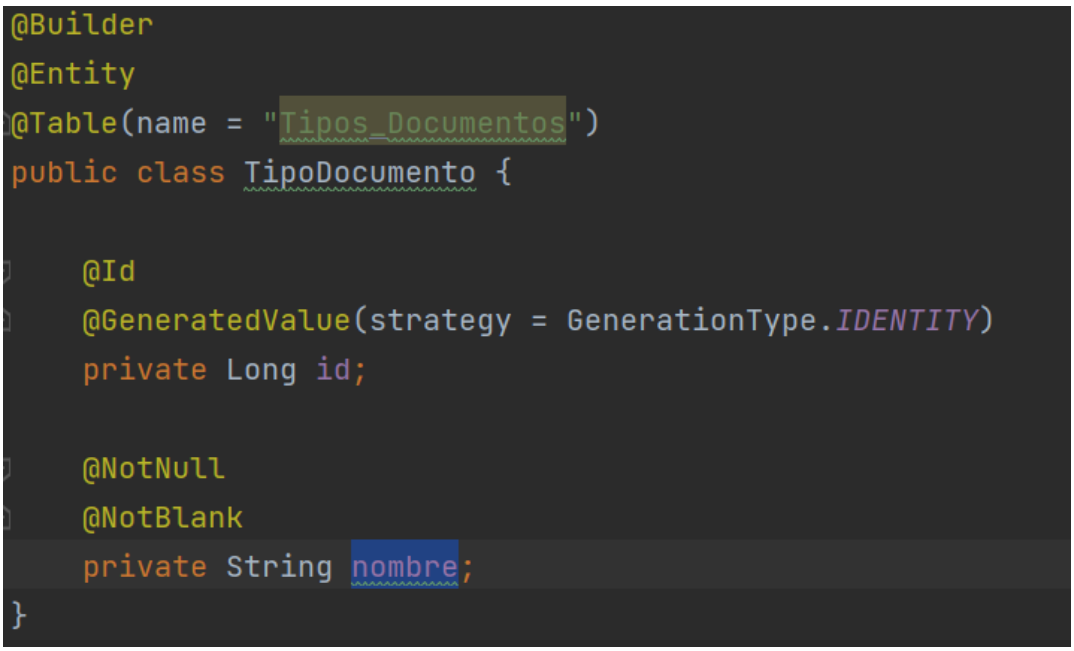
```java
public class Usuario {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @NotNull
    @NotBlank
    private String nombre;

    @NotNull
    @NotBlank
    private String apellido;

    @NotNull
    @NotBlank
    @Email
    @Column(unique = true)
    private String correo;

    @ToString.Exclude
    @NotNull
    @NotBlank
    private String password;
```

Figura 22
Entidad Rol



```java
@Entity
@Table(name = "Roles")
public class Rol {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;

    @NotNull
    @NotBlank
    private String nombre;

    @NotNull
    @Column(name = "esta_activo")
    private Boolean estaActivo;
}
```

Figura 23
Entidad TipoDocumento



```java
@Builder
@Entity
@Table(name = "Tipos_Documentos")
public class TipoDocumento {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @NotNull
    @NotBlank
    private String nombre;
}
```

Figura 24
Entidad UsuarioReclamante

```java
@Entity
@Table(name = "Usuarios_Reclamantes")
public class UsuarioReclamante {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;

    @NotNull
    @NotBlank
    private String documento;

    @NotNull
    @NotBlank
    private String nombre;
}
```

Repositorios:

Es la capa de persistencia la cual se encarga de comunicarse con la respectiva base de datos para obtener la información que el usuario solicita y serializarla a las clases Java configuradas.

Es una interface que define las acciones que se van a realizar sobre la base de datos para la respectiva entidad.

Figura 25
Repositorio DocumentoRepository

```java
import ...

public interface DocumentoRepository extends JpaRepository<Documento, Long>, JpaSpecificationExecutor<Documento> {

    int countByEstaEntregadoTrue();

    int countByEstaEntregadoFalse();

    @Query(value = "select count(tipos_documentos_id) as cantidad, td.nombre as nombre\n" +
            "from documentos\n" +
            "         right outer join tipos_documentos td on documentos.tipos_documentos_id = td.id\n" +
            "group by td.nombre", nativeQuery = true)
    List<DocumentosCantidadPorTipoProjection> getAllByDocumentoCount();

    List<Documento> findAllByNumeroAndEstaEntregadoFalse(String numero);
}
```

Figura 26
Repositorio CiudadRepository

```
import ...

public interface CiudadRepository extends JpaRepository<Ciudad, Long> {

    List<Ciudad> findAllByDepartamentoId(Long idDepartamento);
}
```

Figura 27
Repositorio RolRepository

```
import ...

public interface RolRepository extends JpaRepository<Rol, Integer> {

    Rol findRolByNombre(String nombre);

}
```

Figura 28
Repositorio UsuarioReclamanteRepository

```
import ...

public interface UsuarioReclamanteRepository extends JpaRepository<UsuarioReclamante, Long> {
    Optional<UsuarioReclamante> findUsuarioReclamanteByDocumento(String documento);
}
```

Figura 29
Repositorio UsuarioRepository

```
import ...

@Repository
public interface UsuarioRepository extends JpaRepository<Usuario, Long> {

    Optional<Usuario> findUsuarioByCorreo(String email);
}
```

Servicio e implementación:

El servicio es la capa intermedia en donde se ejecuta la lógica y el tratamiento de la información la cual es llamada por el controlador y a su vez si es necesario comunicarse con la base de datos es esta quien llama al repositorio manteniendo así una separación de capas en el sistema respetando la responsabilidad única.

Figura 30
Servicio CiudadService

```java
package com.example.aplicacionpolicia.service;

import ...

public interface CiudadService {

    List<CiudadDto> findAllByDepartamentoId(Long idDepartamento);
    CiudadDto save(CiudadRequest request);
    Page<CiudadDto> getAll(Pageable page);
    CiudadDto getById(Long id);
    CiudadDto update(Long id, CiudadRequest request);
    void deleteById(Long id);
}
```

Figura 31
Servicio DepartamentoService

```java
import ...

public interface DepartamentoService {

    List<DepartamentoDto> getAllDepartamentos();
    DepartamentoDto save(DepartamentoRequest departamentoRequest);
    Page<DepartamentoDto> getAll(Pageable page);
    DepartamentoDto getById(Long id);
    DepartamentoDto update(Long id, DepartamentoRequest departamentoRequest);
    void deleteById(Long id);
}
```

Figura 32

Servicio DocumentoService

```java
public interface DocumentoService {

    DocumentoDto save(DocumentoRequest request);
    List<DocumentoDto> getAllDocumentos();
    Page<DocumentoDto> getAll(Pageable page);
    DocumentoDto getById(Long id);
    List<DocumentoDto> findAllByNumeroAndEstaEntregadoFalse(String numero);
    DocumentoDto update(Long id, DocumentoRequest request);
    Page<DocumentoDto> filterDocumento(String numero, Boolean estado , String tipo, Pageable page);
    void deleteById(Long id);
    int countByEstaEntregadoTrue();
    int countByEstaEntregadoFalse();
    List<DocumentosCantidadPorTipoProjection> getAllByDocumentoCount();
}
```

Figura 33

Servicio EstacionService

```java
import ...

public interface EstacionService {

    EstacionDto save(EstacionRequest request);
    List<EstacionDto> getAllEstaciones();
    Page<EstacionDto> getAll(Pageable page);
    EstacionDto getById(Long id);
    EstacionDto update(Long id, EstacionRequest request);
    void deleteById(Long id);
}
```

Figura 34

Servicio EntregaService

```java
import ...

public interface EntregaService {

    EntregaDto save(EntregaRequest request);
    List<EntregaDto> getAllEntregas();
    Page<EntregaDto> getAll(Pageable page);
    EntregaDto getById(Long id);
    EntregaDto update(Long id, EntregaRequest request);
    void deleteById(Long id);
}
```

Figura 35
Servicio RolService

```java
public interface RolService {

    RolDto save(RolRequest request);
    List<Rol> getAll();
    RolDto getById(Integer id);
    RolDto update(Integer id, RolRequest request);
    void deleteById(Integer id);
    Rol findRolByNombre(String nombre);
```

Figura 36
Servicio SubCaiService

```java
public interface SubCaiService {

    SubCaiDto save(SubCaiRequest request);
    List<SubCaiDto> getAllCais();
    Page<SubCaiDto> getAll(Pageable page);
    SubCaiDto getById(Long id);
    SubCaiDto update(Long id, SubCaiRequest request);
    void deleteById(Long id);
}
```

Figura 37
Servicio TipoDocumentoService

```java
import ...

public interface TipoDocumentoService {

    TipoDocumentoDto save(TipoDocumentoRequest request);
    Page<TipoDocumentoDto> getAll(Pageable page);
    List<TipoDocumentoDto> getAllTiposDocumentos();
    TipoDocumentoDto getById(Long id);
    TipoDocumentoDto update(Long id, TipoDocumentoRequest request);
    void deleteById(Long id);
}
```

Figura 38

Servicio UsuarioService

```java
public interface UsuarioService {

    UsuarioDto save(UsuarioRequest request);
    Page<UsuarioDto> getAll(Pageable page);
    UsuarioDto getById(Long id);
    UsuarioDto update(Long id, UsuarioRequest request);
    void deleteById(Long id);
    Usuario findUsuarioByCorreo(String email);
    long countAllUsers();
```

Figura 39

Servicio UsuarioReclamanteService

```java
import ...

public interface UsuarioReclamanteService {
    UsuarioReclamanteDto save(UsuarioReclamanteRequest request);
    Page<UsuarioReclamanteDto> getAll(Pageable page);
    UsuarioReclamanteDto getById(Long id);
    UsuarioReclamanteDto update(Long id, UsuarioReclamanteRequest request);
    void deleteById(Long id);
    UsuarioReclamanteDto findUsuarioReclamanteByDocumento(String documento);
}
```

Figura 40

Implementación CiudadServiceImpl

```java
@Service
public class CiudadServiceImpl implements CiudadService {

    private final CiudadRepository ciudadRepository;
    public CiudadServiceImpl(CiudadRepository ciudadRepository) { this.ciudadRepository = ciudadRepository

    @Override
    public List<CiudadDto> findAllByDepartamentoId(Long idDepartamento) {
        return ciudadRepository.findAllByDepartamentoId(idDepartamento).stream() Stream<Ciudad>
                .map(ciudad -> CiudadDto.builder()
                        .id(ciudad.getId())
                        .nombre(ciudad.getNombre())
                        .build()) Stream<CiudadDto>
                .collect(Collectors.toList());
    }

    @Override
    public CiudadDto save(CiudadRequest request) {
        Ciudad savedCiudad = ciudadRepository.save(Ciudad.builder()
                .id(request.getId())
                .nombre(request.getNombre())
```

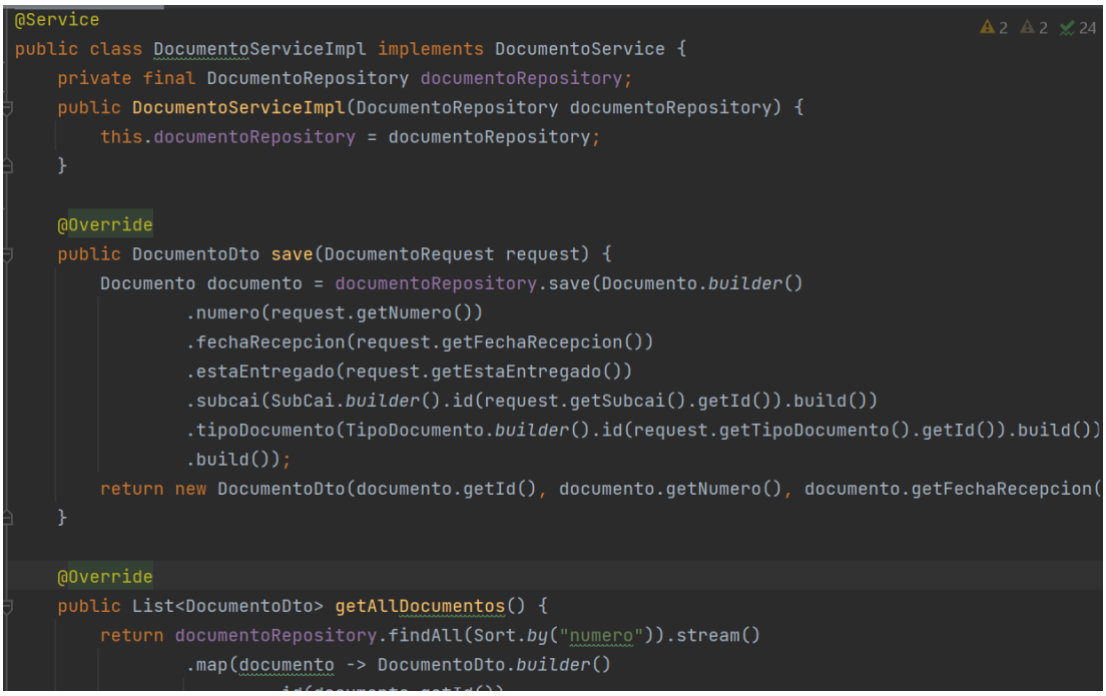
Figura 41

Implementación DepartamentoServiceImpl

```
@Service
public class DepartamentoServiceImpl implements DepartamentoService {
    private final DepartamentoRepository departamentoRepository;
    public DepartamentoServiceImpl(DepartamentoRepository departamentoRepository) {
        this.departamentoRepository = departamentoRepository;
    }

    @Override
    public List<DepartamentoDto> getAllDepartamentos() {
        return departamentoRepository.findAll(Sort.by("nombre")).stream()
                .map(departamento -> DepartamentoDto.builder()
                        .id(departamento.getId())
                        .nombre(departamento.getNombre())
                        .build()
                ).collect(Collectors.toList());
    }

    @Override
    public DepartamentoDto save(DepartamentoRequest departamentoRequest) {
        Departamento savedDepartamento = departamentoRepository.save(Departamento.builder()
                .id(departamentoRequest.getId())
                .nombre(departamentoRequest.getNombre())
                .build());
        return new DepartamentoDto(savedDepartamento.getId(), savedDepartamento.getNombre(),
    }
```

Figura 42

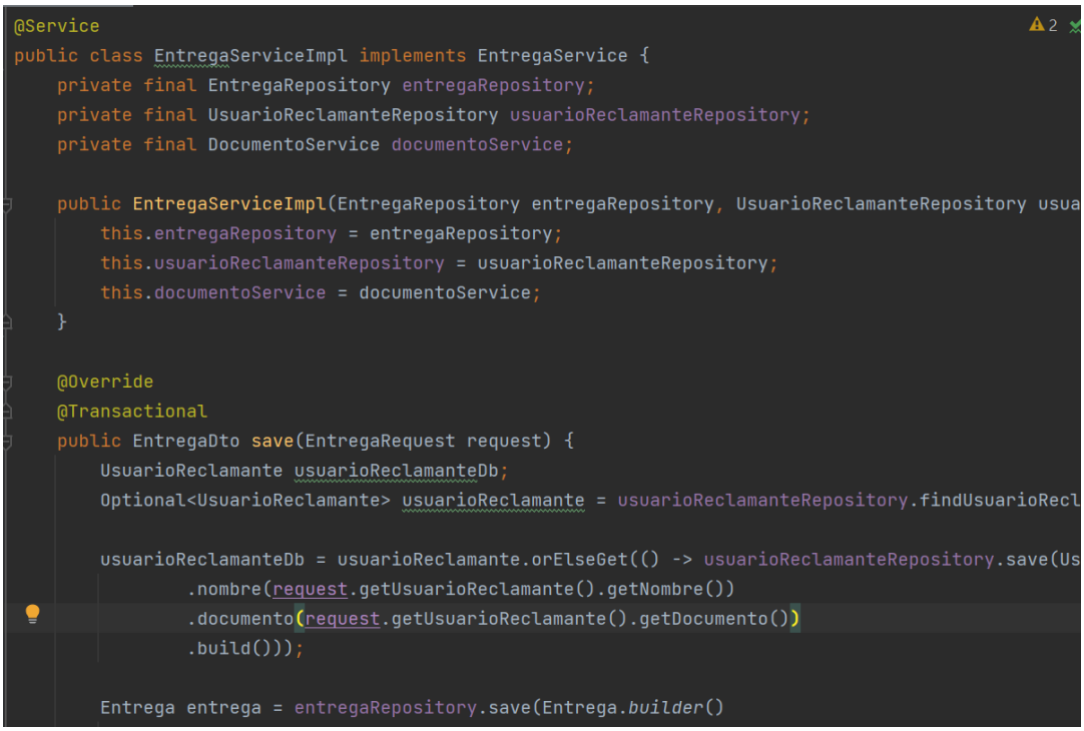
Implementación DocumentoServiceImpl



```
@Service
public class DocumentoServiceImpl implements DocumentoService {
    private final DocumentoRepository documentoRepository;
    public DocumentoServiceImpl(DocumentoRepository documentoRepository) {
        this.documentoRepository = documentoRepository;
    }

    @Override
    public DocumentoDto save(DocumentoRequest request) {
        Documento documento = documentoRepository.save(Documento.builder()
                .numero(request.getNumero())
                .fechaRecepcion(request.getFechaRecepcion())
                .estaEntregado(request.getEstaEntregado())
                .subcai(SubCai.builder().id(request.getSubcai().getId()).build())
                .tipoDocumento(TipoDocumento.builder().id(request.getTipoDocumento().getId()).build())
                .build());
        return new DocumentoDto(documento.getId(), documento.getNumero(), documento.getFechaRecepcion(
    }

    @Override
    public List<DocumentoDto> getAllDocumentos() {
        return documentoRepository.findAll(Sort.by("numero")).stream()
                .map(documento -> DocumentoDto.builder()
```

Figura 43

Implementación EntregaServiceImpl



```
@Service
public class EntregaServiceImpl implements EntregaService {
    private final EntregaRepository entregaRepository;
    private final UsuarioReclamanteRepository usuarioReclamanteRepository;
    private final DocumentoService documentoService;

    public EntregaServiceImpl(EntregaRepository entregaRepository, UsuarioReclamanteRepository usua
        this.entregaRepository = entregaRepository;
        this.usuarioReclamanteRepository = usuarioReclamanteRepository;
        this.documentoService = documentoService;
    }

    @Override
    @Transactional
    public EntregaDto save(EntregaRequest request) {
        UsuarioReclamante usuarioReclamanteDb;
        Optional<UsuarioReclamante> usuarioReclamante = usuarioReclamanteRepository.findUsuarioRecl

        usuarioReclamanteDb = usuarioReclamante.orElseGet(() -> usuarioReclamanteRepository.save(Us
                .nombre(request.getUsuarioReclamante().getNombre())
                .documento(request.getUsuarioReclamante().getDocumento())
                .build()));

        Entrega entrega = entregaRepository.save(Entrega.builder()
```

Figura 44

Implementación EstacionServiceImpl

```
@Service
public class EstacionServiceImpl implements EstacionService {
    private final EstacionRepository estacionRepository;
    public EstacionServiceImpl(EstacionRepository estacionRepository) { this.estacionRepository = estaci

    @Override
    public EstacionDto save(EstacionRequest request) {
        Estacion estacion = estacionRepository.save(Estacion.builder()
                .nombre(request.getNombre())
                .ciudad(Ciudad.builder()
                        .id(request.getCiudad().getId())
                        .build())
                .build());
        return new EstacionDto(estacion.getId(), estacion.getNombre(), estacion.getCiudad(), cais: null);
    }

    @Override
    public List<EstacionDto> getAllEstaciones() {
        return estacionRepository.findAll(Sort.by("nombre")).stream()
                .map(estacion -> EstacionDto.builder()
                        .id(estacion.getId())
                        .nombre(estacion.getNombre())
                        .build()
                ).collect(Collectors.toList());
    }
```

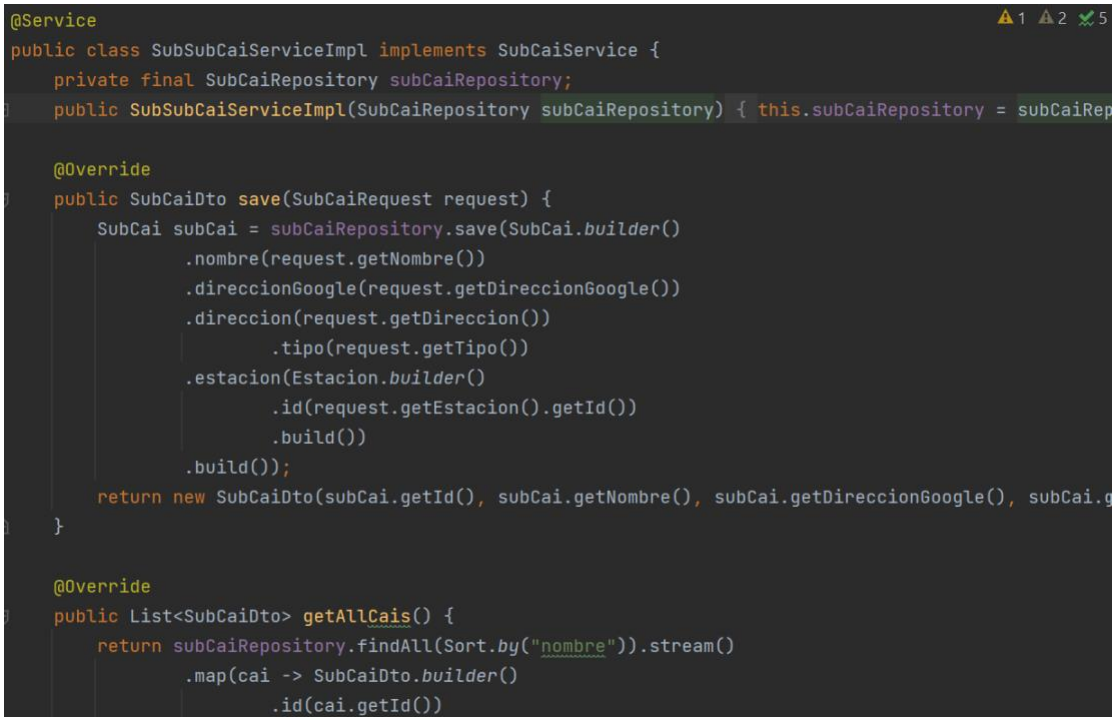
Figura 45

Implementación RolServiceImpl

```
@Service
public class RolServiceImpl implements RolService {
    private final RolRepository rolRepository;
    public RolServiceImpl(RolRepository rolRepository) { this.rolRepository = rolRepository; }

    @Override
    public RolDto save(RolRequest request) {
        Rol rol = rolRepository.save(Rol.builder()
                .id(request.getId())
                .nombre(request.getNombre())
                .estaActivo(request.getEstaActivo())
                .build());
        return new RolDto(rol.getId(),rol.getNombre(), rol.getEstaActivo());
    }

    @Override
    public List<Rol> getAll() {
       return rolRepository.findAll();
      }

    @Override
    public RolDto getById(Integer id) {
        Rol rol = rolRepository.findById(id).orElseThrow(()->{
            throw new CustomBaseException("Rol no encontrado, por favor revise", HttpStatus.BAD_
```

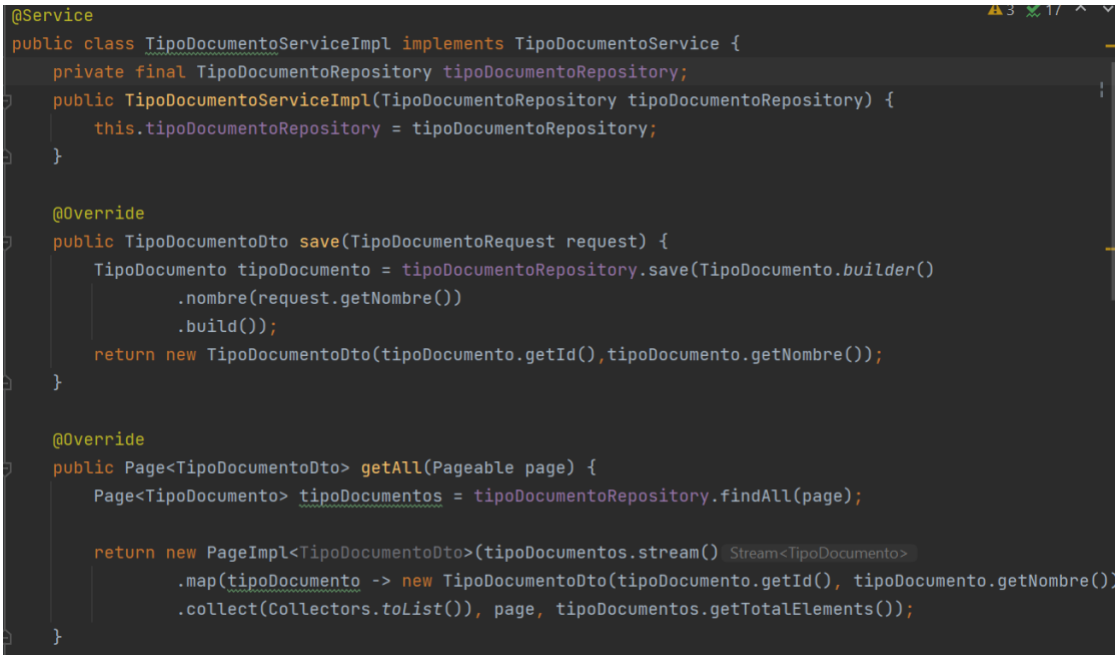
Figura 46

Implementación SubCaiServiceImpl



```
@Service                                                                    1  2  5
public class SubSubCaiServiceImpl implements SubCaiService {
    private final SubCaiRepository subCaiRepository;
    public SubSubCaiServiceImpl(SubCaiRepository subCaiRepository) { this.subCaiRepository = subCaiRep

    @Override
    public SubCaiDto save(SubCaiRequest request) {
        SubCai subCai = subCaiRepository.save(SubCai.builder()
                .nombre(request.getNombre())
                .direccionGoogle(request.getDireccionGoogle())
                .direccion(request.getDireccion())
                        .tipo(request.getTipo())
                .estacion(Estacion.builder()
                        .id(request.getEstacion().getId())
                        .build())
                .build());
        return new SubCaiDto(subCai.getId(), subCai.getNombre(), subCai.getDireccionGoogle(), subCai.g
    }

    @Override
    public List<SubCaiDto> getAllCais() {
        return subCaiRepository.findAll(Sort.by("nombre")).stream()
                .map(cai -> SubCaiDto.builder()
                        .id(cai.getId())
```

Figura 47
Implementación TipoDocumentoServiceImpl



```java
@Service
public class TipoDocumentoServiceImpl implements TipoDocumentoService {
    private final TipoDocumentoRepository tipoDocumentoRepository;
    public TipoDocumentoServiceImpl(TipoDocumentoRepository tipoDocumentoRepository) {
        this.tipoDocumentoRepository = tipoDocumentoRepository;
    }

    @Override
    public TipoDocumentoDto save(TipoDocumentoRequest request) {
        TipoDocumento tipoDocumento = tipoDocumentoRepository.save(TipoDocumento.builder()
                .nombre(request.getNombre())
                .build());
        return new TipoDocumentoDto(tipoDocumento.getId(),tipoDocumento.getNombre());
    }

    @Override
    public Page<TipoDocumentoDto> getAll(Pageable page) {
        Page<TipoDocumento> tipoDocumentos = tipoDocumentoRepository.findAll(page);

        return new PageImpl<TipoDocumentoDto>(tipoDocumentos.stream()
                .map(tipoDocumento -> new TipoDocumentoDto(tipoDocumento.getId(), tipoDocumento.getNombre())
                .collect(Collectors.toList()), page, tipoDocumentos.getTotalElements());
    }
```

Controladores:

Es el punto de acceso donde llega la información enviada por el usuario quien a su vez se comunica con el servicio para obtener dicha información, siendo el controlador quien retorna la información al usuario.

El usuario se comunica con el controlador a través de los verbos Http, en este sistema implementamos principalmente los siguientes verbos:

GET: Se utiliza para obtener información recibiendo o no parámetros a través de la url.

POST: Se utiliza para que el usuario envié información a través del body en este caso los formularios.

PUT: Se utiliza para actualizar información la cual se envía también a través del body, de acuerdo con los parámetros recibidos.

DELETE: Se utiliza para eliminar información de acuerdo con parámetros recibidos.

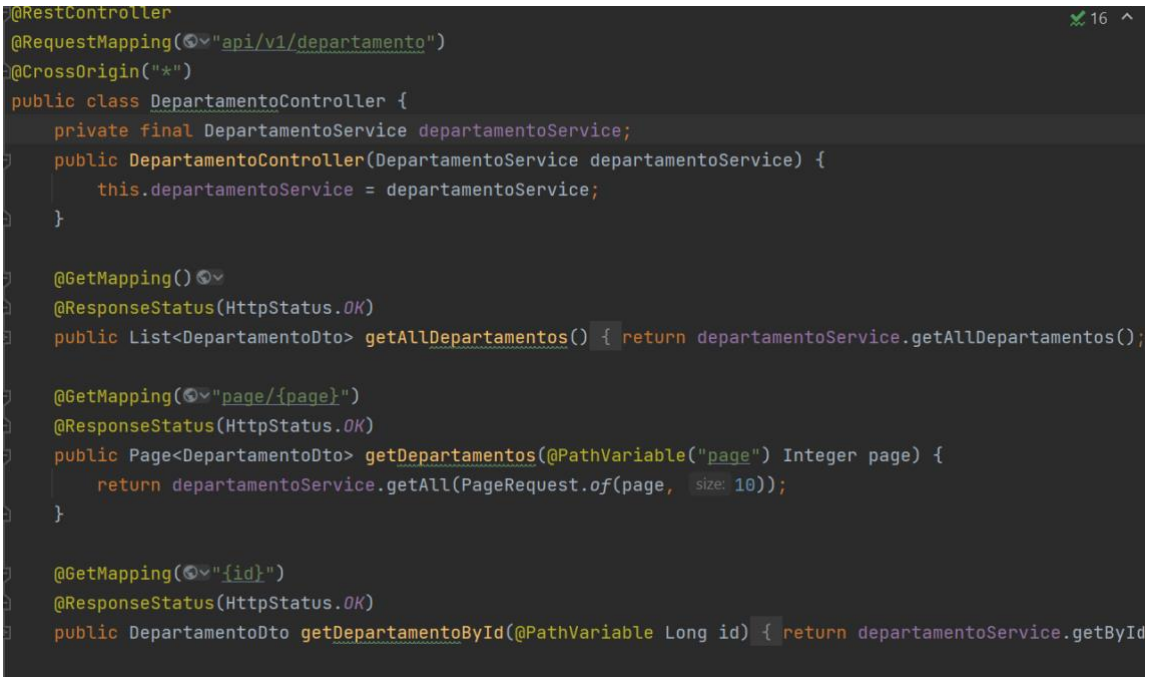
Figura 48

Controlador CiudadController

```
@RestController
@RequestMapping("api/v1/ciudad")
@CrossOrigin("*")
public class CiudadController {
    private final CiudadService ciudadService;
    public CiudadController(CiudadService ciudadService) { this.ciudadService = ciudadService; }

    @GetMapping("ciudades/{id}")
    @ResponseStatus(HttpStatus.OK)
    @RolesAllowed({"ADMIN"})
    public List<CiudadDto> findAllByDepartamentoId(@PathVariable Long id){
        return ciudadService.findAllByDepartamentoId(id);
    }

    @GetMapping("page/{page}")
    @ResponseStatus(HttpStatus.OK)
    @RolesAllowed({"POPI"})
    public Page<CiudadDto> getCiudades(@PathVariable("page") Integer page) {
        return ciudadService.getAll(PageRequest.of(page, size: 10));
    }

    @GetMapping("{id}")
    @ResponseStatus(HttpStatus.OK)
    public CiudadDto getCiudadById(@PathVariable Long id) { return ciudadService.getById(id); }
```

Figura 49

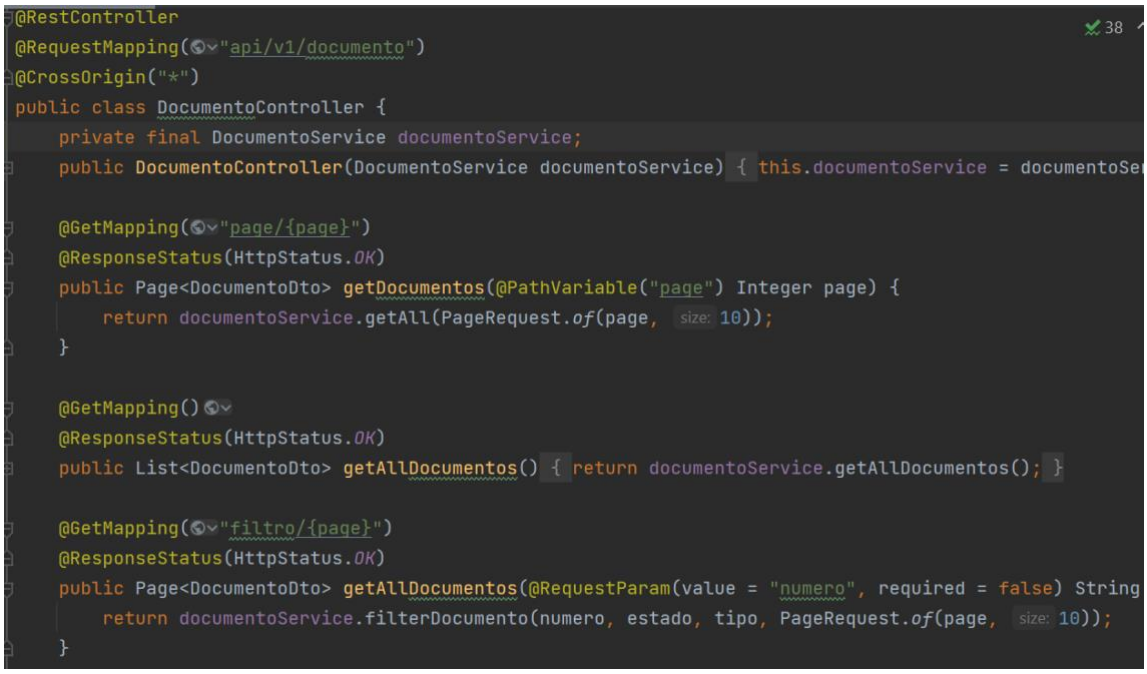
Controlador DepartamentoController



```
@RestController
@RequestMapping("api/v1/departamento")
@CrossOrigin("*")
public class DepartamentoController {
    private final DepartamentoService departamentoService;
    public DepartamentoController(DepartamentoService departamentoService) {
        this.departamentoService = departamentoService;
    }

    @GetMapping()
    @ResponseStatus(HttpStatus.OK)
    public List<DepartamentoDto> getAllDepartamentos() { return departamentoService.getAllDepartamentos();

    @GetMapping("page/{page}")
    @ResponseStatus(HttpStatus.OK)
    public Page<DepartamentoDto> getDepartamentos(@PathVariable("page") Integer page) {
        return departamentoService.getAll(PageRequest.of(page, size: 10));
    }

    @GetMapping("{id}")
    @ResponseStatus(HttpStatus.OK)
    public DepartamentoDto getDepartamentoById(@PathVariable Long id) { return departamentoService.getById
```

Figura 50

Controlador DocumentoController



```
@RestController
@RequestMapping("api/v1/documento")
@CrossOrigin("*")
public class DocumentoController {
    private final DocumentoService documentoService;
    public DocumentoController(DocumentoService documentoService) { this.documentoService = documentoSe

    @GetMapping("page/{page}")
    @ResponseStatus(HttpStatus.OK)
    public Page<DocumentoDto> getDocumentos(@PathVariable("page") Integer page) {
        return documentoService.getAll(PageRequest.of(page, size: 10));
    }

    @GetMapping()
    @ResponseStatus(HttpStatus.OK)
    public List<DocumentoDto> getAllDocumentos() { return documentoService.getAllDocumentos(); }

    @GetMapping("filtro/{page}")
    @ResponseStatus(HttpStatus.OK)
    public Page<DocumentoDto> getAllDocumentos(@RequestParam(value = "numero", required = false) String
        return documentoService.filterDocumento(numero, estado, tipo, PageRequest.of(page, size: 10));
    }
```

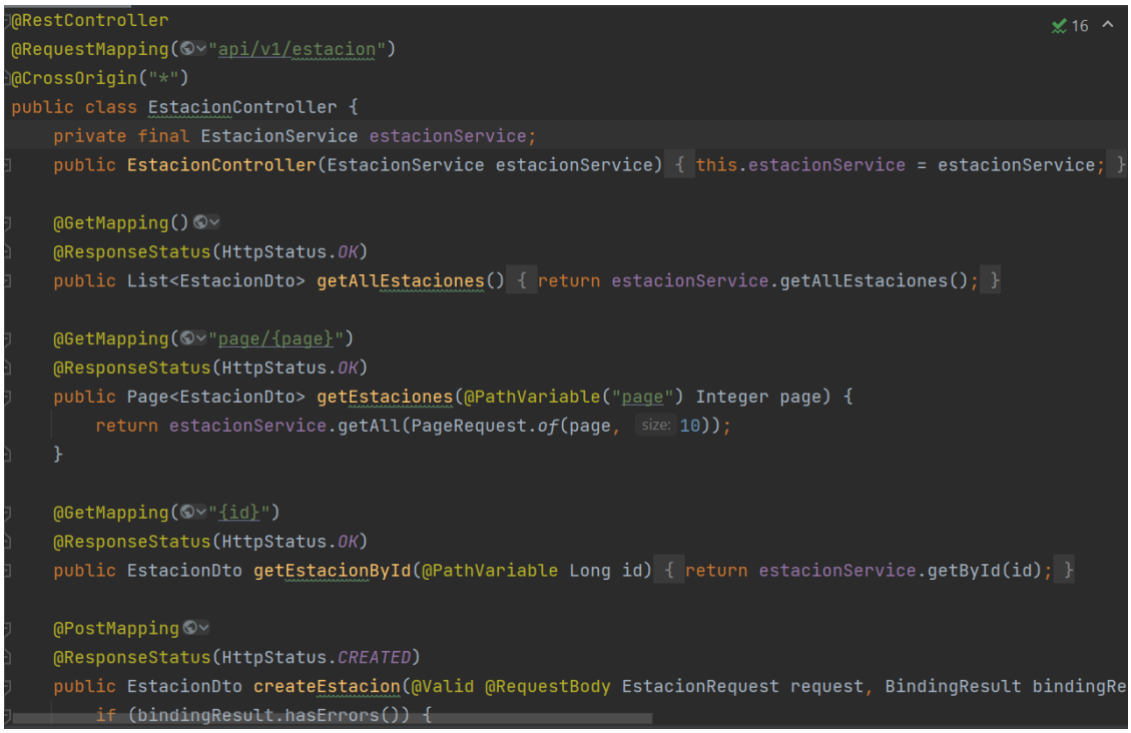
Figura 51

Controlador EntregaController

```
@RestController
@RequestMapping("api/v1/entrega")
@CrossOrigin("*")
public class EntregaController {
    private final EntregaService entregaService;
    public EntregaController(EntregaService entregaService) { this.entregaService = entregaService; }

    @GetMapping()
    @ResponseStatus(HttpStatus.OK)
    public List<EntregaDto> getAllEntregas() { return entregaService.getAllEntregas(); }

    @GetMapping("page/{page}")
    @ResponseStatus(HttpStatus.OK)
    public Page<EntregaDto> getEntregas(@PathVariable("page") Integer page) {
        return entregaService.getAll(PageRequest.of(page, size: 10));
    }

    @GetMapping("{id}")
    @ResponseStatus(HttpStatus.OK)
    public EntregaDto getEntregaById(@PathVariable Long id) { return entregaService.getById(id); }

    @PostMapping
    @ResponseStatus(HttpStatus.CREATED)
    public EntregaDto createEntrega(@Valid @RequestBody EntregaRequest request, BindingResult binding
```

Figura 52

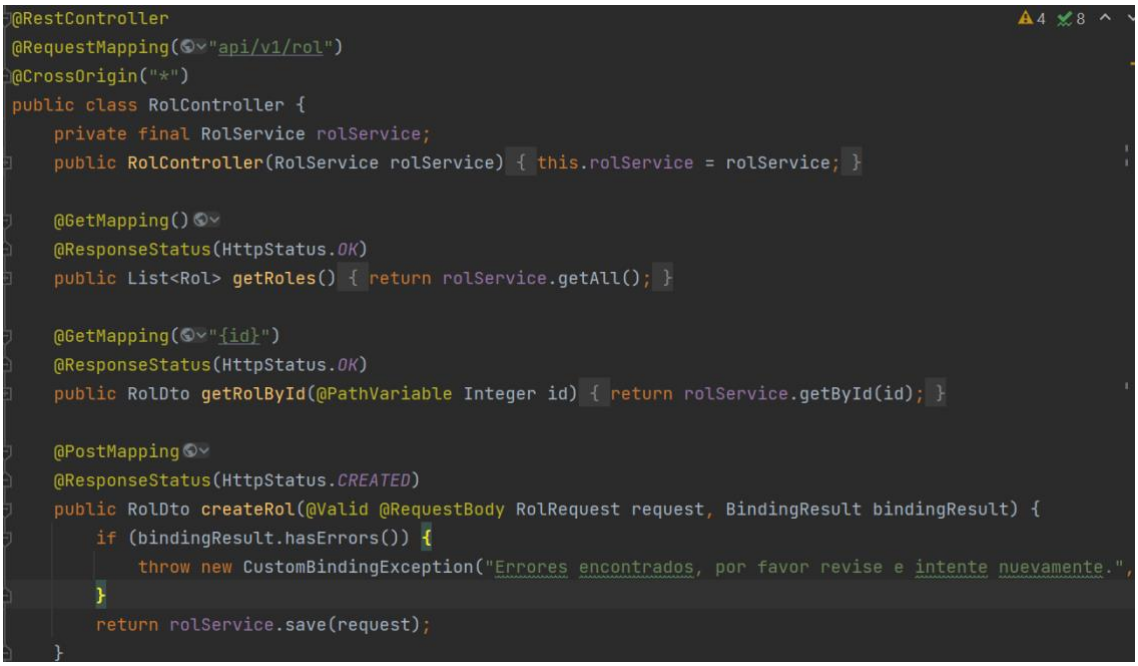
Controlador EstacionController



```
@RestController
@RequestMapping("api/v1/estacion")
@CrossOrigin("*")
public class EstacionController {
    private final EstacionService estacionService;
    public EstacionController(EstacionService estacionService) { this.estacionService = estacionService; }

    @GetMapping()
    @ResponseStatus(HttpStatus.OK)
    public List<EstacionDto> getAllEstaciones() { return estacionService.getAllEstaciones(); }

    @GetMapping("page/{page}")
    @ResponseStatus(HttpStatus.OK)
    public Page<EstacionDto> getEstaciones(@PathVariable("page") Integer page) {
        return estacionService.getAll(PageRequest.of(page, size: 10));
    }

    @GetMapping("{id}")
    @ResponseStatus(HttpStatus.OK)
    public EstacionDto getEstacionById(@PathVariable Long id) { return estacionService.getById(id); }

    @PostMapping
    @ResponseStatus(HttpStatus.CREATED)
    public EstacionDto createEstacion(@Valid @RequestBody EstacionRequest request, BindingResult bindingRe
        if (bindingResult.hasErrors()) {
```

Figura 53

Controlador RolController



```
@RestController
@RequestMapping("api/v1/rol")
@CrossOrigin("*")
public class RolController {
    private final RolService rolService;
    public RolController(RolService rolService) { this.rolService = rolService; }

    @GetMapping()
    @ResponseStatus(HttpStatus.OK)
    public List<Rol> getRoles() { return rolService.getAll(); }

    @GetMapping("{id}")
    @ResponseStatus(HttpStatus.OK)
    public RolDto getRolById(@PathVariable Integer id) { return rolService.getById(id); }

    @PostMapping
    @ResponseStatus(HttpStatus.CREATED)
    public RolDto createRol(@Valid @RequestBody RolRequest request, BindingResult bindingResult) {
        if (bindingResult.hasErrors()) {
            throw new CustomBindingException("Errores encontrados, por favor revise e intente nuevamente.",
        }
        return rolService.save(request);
    }
```

Figura 54

Controlador SubCaiController

```java
@RestController
@RequestMapping("api/v1/cai")
@CrossOrigin("*")
public class SubCaiController {
    private final SubCaiService subCaiService;
    public SubCaiController(SubCaiService subCaiService) {
        this.subCaiService = subCaiService;
    }

    @GetMapping("page/{page}")
    @ResponseStatus(HttpStatus.OK)
    public Page<SubCaiDto> getCais(@PathVariable("page") Integer page) {
        return subCaiService.getAll(PageRequest.of(page, size: 10));
    }

    @GetMapping()
    @ResponseStatus(HttpStatus.OK)
    public List<SubCaiDto> getCais() {
        return subCaiService.getAllCais();}

    @GetMapping("{id}")
    @ResponseStatus(HttpStatus.OK)
    public SubCaiDto getCaiById(@PathVariable Long id){
        return subCaiService.getById(id);
```

Figura 55

Controlador TipoDocumentoController

```java
@RestController
@RequestMapping("api/v1/tipodocumento")
@CrossOrigin("*")
public class TipoDocumentoController {
    private final TipoDocumentoService tipoDocumentoService;
    public TipoDocumentoController(TipoDocumentoService tipoDocumentoService) {
        this.tipoDocumentoService = tipoDocumentoService;
    }

    @GetMapping("page/{page}")
    @ResponseStatus(HttpStatus.OK)
    public Page<TipoDocumentoDto> getTiposDocumentos(@PathVariable("page") Integer page) {
        return tipoDocumentoService.getAll(PageRequest.of(page, size: 10));
    }

    @GetMapping()
    @ResponseStatus(HttpStatus.OK)
    public List<TipoDocumentoDto> getAllTiposDocumentos() { return tipoDocumentoService.getAllTiposDocumento

    @GetMapping("{id}")
    @ResponseStatus(HttpStatus.OK)
    public TipoDocumentoDto getTipoDocumentoById(@PathVariable Long id) { return tipoDocumentoService.getByI
```

Figura 56

Controlador UsuarioController

```java
@RestController
@RequestMapping("api/v1/usuario")
@CrossOrigin("*")
public class UsuarioController {
    @Autowired
    private AuthenticationManager authenticationManager;
    @Autowired
    private TokenProvider jwtTokenUtil;
    private final UsuarioService usuarioService;
    public UsuarioController(UsuarioService usuarioService) { this.usuarioService = usuarioService; }

    @GetMapping("page/{page}")
    @ResponseStatus(HttpStatus.OK)
    public Page<UsuarioDto> getUsuarios(@PathVariable("page") Integer page) {
        return usuarioService.getAll(PageRequest.of(page, size: 10));    }

    @GetMapping("{id}")
    @ResponseStatus(HttpStatus.OK)
    public UsuarioDto getUsuarioById(@PathVariable Long id) { return usuarioService.getById(id); }

    @GetMapping("email/{email}")
    @ResponseStatus(HttpStatus.OK)
    public Usuario getUsuarioByEmail(@PathVariable("email") String email) {
        return usuarioService.findUsuarioByCorreo(email);    }
```

Figura 57

Controlador UsuarioReclamanteController

```java
@RestController
@RequestMapping("api/v1/usuarioreclamante")
@CrossOrigin("*")
public class UsuarioReclamanteController {
    private final UsuarioReclamanteService usuarioReclamanteService;
    public UsuarioReclamanteController(UsuarioReclamanteService usuarioReclamanteService) {
        this.usuarioReclamanteService = usuarioReclamanteService;    }

    @GetMapping("page/{page}")
    @ResponseStatus(HttpStatus.OK)
    public Page<UsuarioReclamanteDto> getUsuariosReclamantes(@PathVariable("page") Integer page) {
        return usuarioReclamanteService.getAll(PageRequest.of(page, size: 10));    }

    @GetMapping("{id}")
    @ResponseStatus(HttpStatus.OK)
    public UsuarioReclamanteDto getUsuarioReclamanteById(@PathVariable Long id) {
        return usuarioReclamanteService.getById(id);    }

    @GetMapping("documento/{documento}")
    @ResponseStatus(HttpStatus.OK)
    public UsuarioReclamanteDto getUsuarioReclamanteByDocumento(@PathVariable String documento) {
        return usuarioReclamanteService.findUsuarioReclamanteByDocumento(documento);
    }
```

Manejo de excepciones:

Las excepciones son errores del sistema que interrumpen el flujo normal de ejecución, por esto capturamos las excepciones y las lanzamos de manera controlada para avisar al usuario lo que sucede con el sistema.

Figura 58
Excepciones ExceptionGlobalHandler

Figura 59
Excepciones CustomBaseException

Frontend

Componentes:

Los componentes son funcionalidades del sistema que cumplen con una tarea especifica y están compuestos por archivos Html, css y typescript. En el Html se especifica la estructura la interface, el Css nos permite agregarle estilos al documento y el typescript la funcionalidad.

En el sistema se crean componentes de tipo formulario para que los usuarios guarden información en el sistema y componentes para listar la información almacenada en la base de datos con su respectiva paginación para que sea más fácil la navegación para el usuario.

Figura 60

Componente DocumentosformComponent

```
export class DocumentosformComponent implements OnInit {
  documentoForm: FormGroup;
  cais: SubCai[] = [];
  documento: Documento;
  tipoDocumentos: TipoDocumento[] = [];

  constructor(private caiService: CaisService,
              private documentoService: DocumentosService, private router: Router,
              private activatedRoute: ActivatedRoute, private tipoDocumentoService: TipoDocumentosService)
  ngOnInit() {
    this.activatedRoute.paramMap.subscribe(
      next: (param : ParamMap ) => { const id: any = param.get('id');
      if (id) {
        this.documentoService.getDocumentoById(id).subscribe(
          next: (json) => {
            this.documento = json;
            this.documentoForm.setValue( value: {
              numero: this.documento.numero,
              fechaRecepcion: this.documento.fechaRecepcion,
              tipoDocumento: this.documento.tipoDocumento.id,
              subcai: this.documento.subcai.id
            });
```

Figura 61

Componente DocumentoslistComponent

```
export class DocumentoslistComponent implements OnInit {
  documentosList: Documento[] = [];
  page = 1;
  pageSize: number;
  collectionSize = 0;
  activeTipo = '';
  activeEstado = -1;
  idTipo: number;
  idEstado: number;
  estados: number[] = [0, 1];
  tipoDocumentos: TipoDocumento[] = [];
  arrayDatosFiltro = new Map();
  filtro = false;
  dataSource;
  constructor(public documentoService: DocumentosService, private router: Router, public tipoDocumentoService

  ngOnInit() {
    this.getAll();
    this.tipoDocumentoService.getAllTipoDocumentos().subscribe(
      next: (json) => {
        this.tipoDocumentos = json;
        console.log(json);
```

Figura 62

Componente EntregasformComponent

```
export class EntregasformComponent implements OnInit {
  entregaForm: FormGroup;
  documentos: Documento[] = [];
  documentoAEntregar: Documento = new Documento();
  resultado = false;

  constructor(private entregaService: EntregasService, private router: Router,
              private modalService: NgbModal, private activatedRoute: ActivatedRoute,
              private usuarioReclamanteService: UsuariosreclamantesService, private documentoService: Documento

  ngOnInit() {
    this.activatedRoute.paramMap.subscribe(
      next: (param : ParamMap ) => {
        const id: any = param.get('id');
        if (id) {
          this.documentoService.getDocumentoById(id).subscribe(
            next: (json) => {
              console.log('documento a entregar', json);
              this.documentoAEntregar = json;
            },
            error: (e) => {
              console.log(e);
            }
```

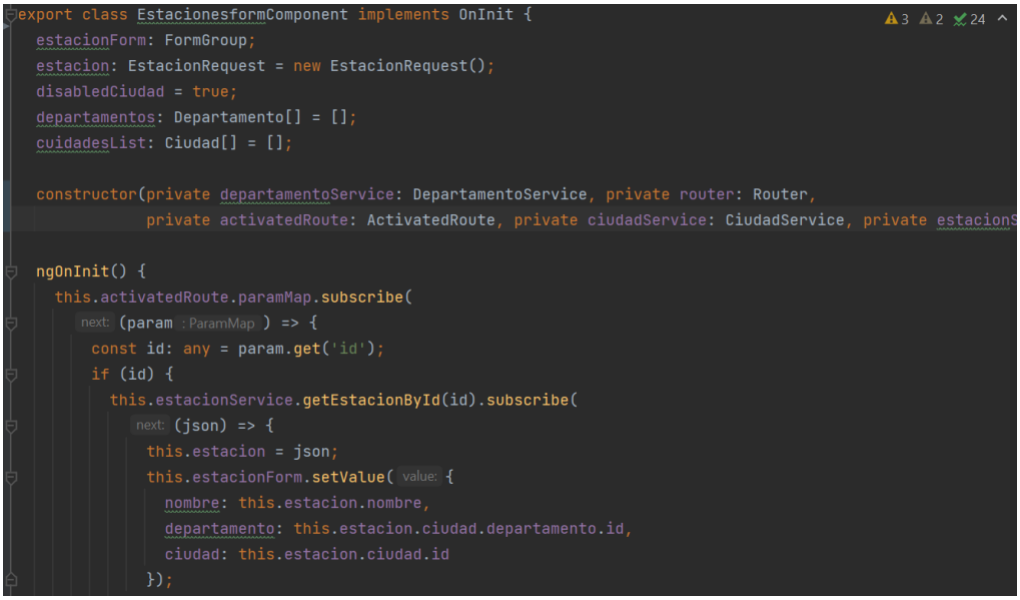
Figura 63

Componente EntregaslistComponent

```
export class EntregaslistComponent implements OnInit {
  entregasList: Entrega[] = [];
  page = 0;
  pageSize: number;
  constructor(private entregaService: EntregasService, private modalService: NgbModal) { }

  ngOnInit() {
    this.entregaService.getEntregas(this.page).subscribe( next: value => {
        this.entregasList = value.content;
        this.pageSize = value.size;
        this.page = value.pageable.pageNumber;
      },
      error: error => {
        console.log(error);
      }
    );
  }
  deleteEntrega(entrega: Entrega) {
    this.modalService.open(ModalPoliceComponent).result.then((result) => {
      this.entregaService.deleteEntrega(entrega.id).subscribe(
        next: success => {
          this.entregasList = this.entregasList.filter(item => item.id !== entrega.id);
        },
```

Figura 64

Componente EstacionesformComponent



```
export class EstacionesformComponent implements OnInit {                    ⚠3 ⚠2 ✓24
  estacionForm: FormGroup;
  estacion: EstacionRequest = new EstacionRequest();
  disabledCiudad = true;
  departamentos: Departamento[] = [];
  cuidadesList: Ciudad[] = [];

  constructor(private departamentoService: DepartamentoService, private router: Router,
              private activatedRoute: ActivatedRoute, private ciudadService: CiudadService, private estacionS

  ngOnInit() {
    this.activatedRoute.paramMap.subscribe(
      next: (param : ParamMap ) => {
        const id: any = param.get('id');
        if (id) {
          this.estacionService.getEstacionById(id).subscribe(
            next: (json) => {
              this.estacion = json;
              this.estacionForm.setValue( value: {
                nombre: this.estacion.nombre,
                departamento: this.estacion.ciudad.departamento.id,
                ciudad: this.estacion.ciudad.id
              });
```

Figura 65

Componente EstacioneslistComponent

```
export class EstacioneslistComponent implements OnInit {
  estacionesList: Estacion[] = [];
  page = 1;
  pageSize: number;
  collectionSize = 0;
  constructor(public estacionService: EstacionesService, private modalService: NgbModal) {  }

  ngOnInit() {
    this.estacionService.getEstaciones( page: 0).subscribe( next: value => {
        this.estacionesList = value.content;
        this.pageSize = value.size;
        this.page = value.number + 1;
        this.collectionSize = value.totalElements;
      },
      error: error => {
        console.log(error);
      }
    );

  }
```

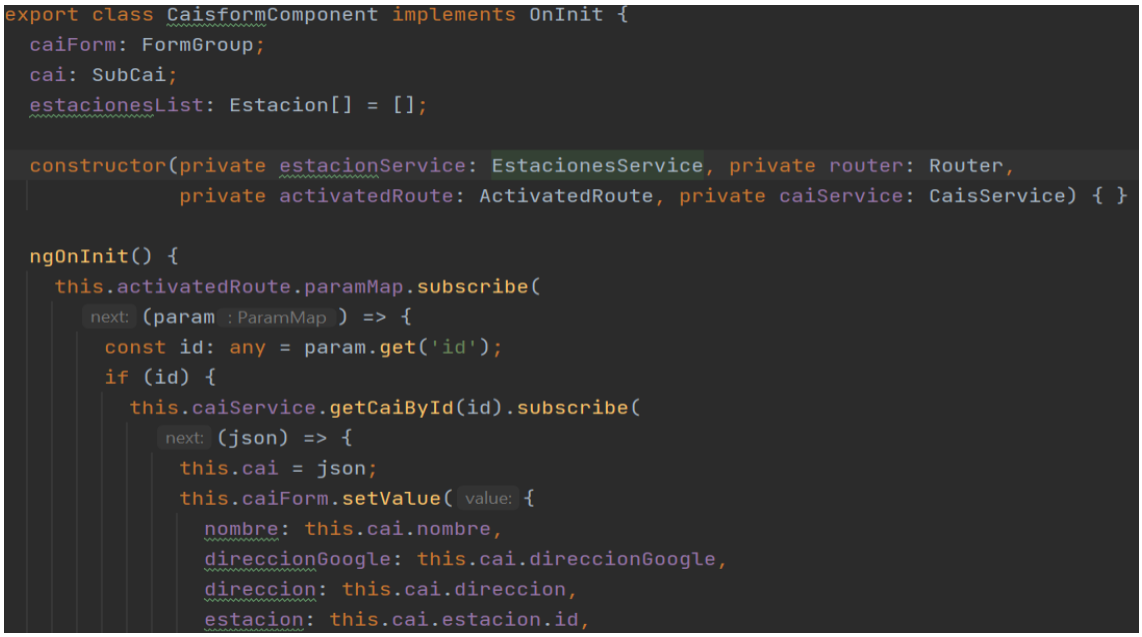
Figura 66

Componente UsuarioformComponent

```
export class UsuarioformComponent implements OnInit {                    ⚠4 ⚠4 ✓31
  usuarioForm: FormGroup;
  roles: Rol[] = [];
  usuario: Usuario;
  estados: number[] = [0, 1];

  constructor(private rolService: RolesService, private usuarioService: UsuariosService,
              private router: Router, private activatedRoute: ActivatedRoute) { }

  ngOnInit() {
    this.activatedRoute.paramMap.subscribe(
      next: (param : ParamMap ) => {
        const id: any = param.get('id');
        if (id) {
          this.usuarioService.getUsuarioById(id).subscribe(
            next: (json) => {
              this.usuario = json;
              this.usuarioForm.controls['nombre'].setValue(this.usuario.nombre);
              this.usuarioForm.controls['apellido'].setValue(this.usuario.apellido);
              this.usuarioForm.controls['email'].setValue(this.usuario.correo);
              this.usuarioForm.controls['rol'].setValue(this.usuario.rol.id);
              this.usuarioForm.controls['password'].setValue(this.usuario.password);
              this.usuarioForm.controls['estado'].setValue( value: this.usuario.estaActivo === true ? 1 : 0);
```

Figura 67

Componente CaisformComponent



```
export class CaisformComponent implements OnInit {
  caiForm: FormGroup;
  cai: SubCai;
  estacionesList: Estacion[] = [];

  constructor(private estacionService: EstacionesService, private router: Router,
              private activatedRoute: ActivatedRoute, private caiService: CaisService) { }

  ngOnInit() {
    this.activatedRoute.paramMap.subscribe(
      next: (param : ParamMap ) => {
        const id: any = param.get('id');
        if (id) {
          this.caiService.getCaiById(id).subscribe(
            next: (json) => {
              this.cai = json;
              this.caiForm.setValue( value: {
                nombre: this.cai.nombre,
                direccionGoogle: this.cai.direccionGoogle,
                direccion: this.cai.direccion,
                estacion: this.cai.estacion.id,
```

Figura 68

Componente CaislistComponent

```
export class CaislistComponent implements OnInit {
  caisList: SubCai[] = [];
  page = 1;
  pageSize: number;
  collectionSize = 0;
  constructor(public caisService: CaisService, private modalService: NgbModal) {  }

  ngOnInit() {
    this.caisService.getCais( page: 0).subscribe( next: value => {
        this.caisList = value.content;
        this.pageSize = value.size;
        this.page = value.number + 1;
        this.collectionSize = value.totalElements;
      },
      error: error => {
        console.log(error);
      }
    );
  }
  pageChange(event: number) {
    this.caisService.getCais( page: event - 1).subscribe( next: value => {
        this.caisList = value.content;
```

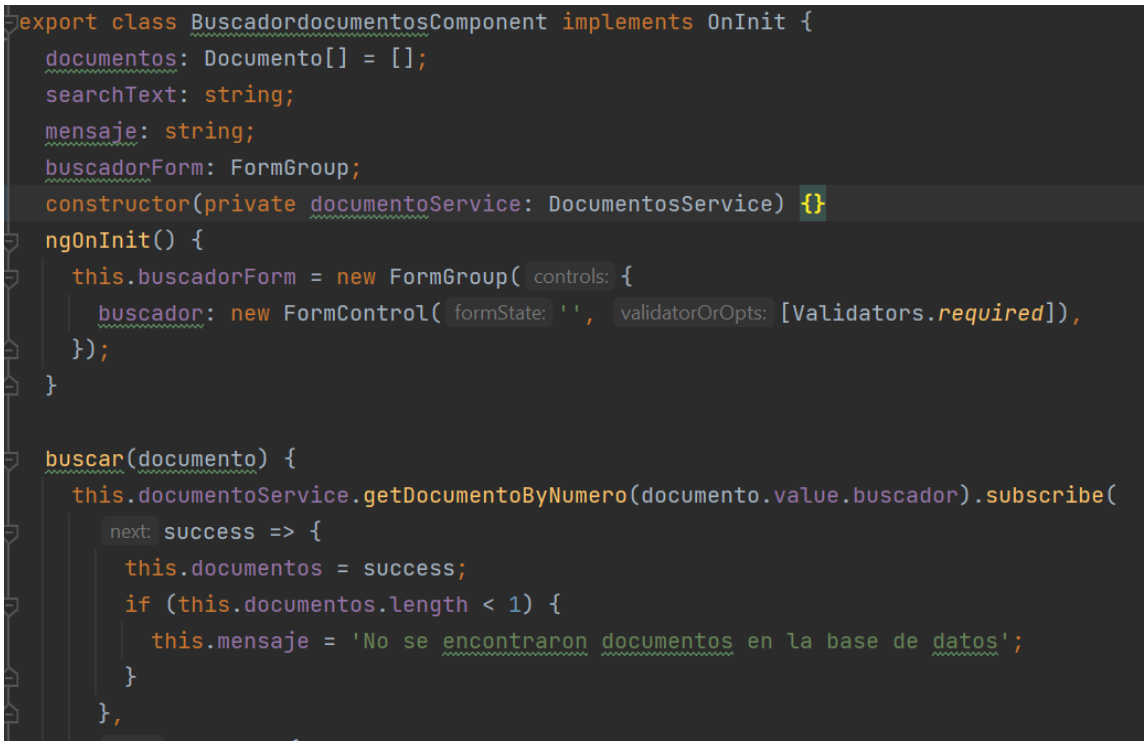
Figura 69

Componente UsuariolistComponent

```
export class UsuariolistComponent implements OnInit {
  usuarioList: Usuario[] = [];
  page = 1;
  pageSize: number;
  collectionSize = 0;
  roles: Rol[] = [];
  usuarioForm: FormGroup;
  dataSource;

  constructor(private usuarioService: UsuariosService, private rolService: RolesService,
  ngOnInit() {
    this.getAll();
    this.rolService.getRoles().subscribe(
      next: success => {
        console.log('roles', success);
        this.roles = success;
      },
      error: error => {
        console.log(error);
      }
    );
```

Figura 70

Componente BuscadorDocumentosComponent



```
export class BuscadordocumentosComponent implements OnInit {
  documentos: Documento[] = [];
  searchText: string;
  mensaje: string;
  buscadorForm: FormGroup;
  constructor(private documentoService: DocumentosService) {}
  ngOnInit() {
    this.buscadorForm = new FormGroup( controls: {
      buscador: new FormControl( formState: '',  validatorOrOpts: [Validators.required]),
    });
  }

  buscar(documento) {
    this.documentoService.getDocumentoByNumero(documento.value.buscador).subscribe(
      next: success => {
        this.documentos = success;
        if (this.documentos.length < 1) {
          this.mensaje = 'No se encontraron documentos en la base de datos';
        }
      },
```

Servicios:

Son los que se comunican con el backend para enviar y recibir la información que el usuario requiere a través de los mismos verbos http mencionados anteriormente.

Figura 71

Servicio DocumentosService

```
export class DocumentosService {
  private urlEndpoint: string = URL_BACKEND + 'documento/';
  private url: string;
  constructor(private http: HttpClient, private router: Router, private headerService: HeadersService) {}

  getDocumentos(page: number): Observable<any> {
    return this.http.get( url: this.urlEndpoint + 'page/' + page, options: {headers: this.headerService.agregarAut
  }

  createDocumento(documento: DocumentoRequest): Observable<any> {
    return this.http.post(this.urlEndpoint, documento, options: {headers: this.headerService.agregarAuthHeader()
  }

  updateDocumento(id: number, documento: DocumentoRequest): Observable<any> {
    return this.http.put( url: this.urlEndpoint + id, documento, options: {headers: this.headerService.agregarAuth
  }

  getAllDocumentos(): Observable<any> {
    return this.http.get(this.urlEndpoint , options: {headers: this.headerService.agregarAuthHeader()});
  }

  countDocumentoEntregadoTrue(): Observable<any> {
    return this.http.get( url: this.urlEndpoint + 'entregados/true', options: {headers: this.headerService.agregar
  }
```

Figura 72

Servicio EntregasService

```
@Injectable({
  providedIn: 'root'
})
export class EntregasService {
  private urlEndpoint: string = URL_BACKEND + 'entrega/';
  constructor(private http: HttpClient, private router: Router, private headerService: HeadersService) {}

  createEntrega(entrega: EntregaRequest): Observable<any> {
    return this.http.post(this.urlEndpoint, entrega, options: {headers: this.headerService.agregarAuthHeader()});
  }

  getAllEntregas(): Observable<any> {
    return this.http.get(this.urlEndpoint , options: {headers: this.headerService.agregarAuthHeader()});
  }

  getEntregas(page: number): Observable<any> {
    return this.http.get( url: this.urlEndpoint + 'page/' + page, options: {headers: this.headerService.agregarAuth
  }

  deleteEntrega(id: number): Observable<any> {
    return this.http.delete( url: this.urlEndpoint + id, options: {headers: this.headerService.agregarAuthHeader()}
  }
}
```

Figura 73

Servicio EstacionesService

```
export class EstacionesService {
  private urlEndpoint: string = URL_BACKEND + 'estacion/';
  constructor(private http: HttpClient, private router: Router, private headerService: HeadersService) {}

  createEstacion(estacion: EstacionRequest): Observable<any> {
    return this.http.post(this.urlEndpoint, estacion, options: {headers: this.headerService.agregarAuthHeader()})
  }

  updateEstacion(id: number, estacion: EstacionRequest): Observable<any> {
    return this.http.put( url: this.urlEndpoint + id, estacion, options: {headers: this.headerService.agregarAuthH
  }

  getAllEstaciones(): Observable<any> {
    return this.http.get(this.urlEndpoint , options: {headers: this.headerService.agregarAuthHeader()});
  }

  getEstaciones(page: number): Observable<any> {
    return this.http.get( url: this.urlEndpoint + 'page/' + page, options: {headers: this.headerService.agregarAut
  }

  deleteEstacion(id: number): Observable<any> {
    return this.http.delete( url: this.urlEndpoint + id, options: {headers: this.headerService.agregarAuthHeader()
  }
```

Figura 74

Servicio SigninService

```
export class SigninService {
  private url: string = URL_BACKEND + 'usuario/';
  private _usuario: Usuario;
  private _token: string;
  constructor(private httpClient: HttpClient) {}

  public get usuario(): Usuario {
    if (this._usuario != null) {
      return this._usuario;
    } else if (this._usuario == null && sessionStorage.getItem( key: 'usuario') != null) {
      this._usuario = JSON.parse(sessionStorage.getItem( key: 'usuario')) as Usuario;
      return this._usuario;
    }
    return new Usuario();}

  getUsuario(): Usuario {
    this._usuario = JSON.parse(sessionStorage.getItem( key: 'usuario'));
    return this._usuario;
  }

  public get token(): string {
    if (this._token != null) {
      return this._token;
```

Figura 75
Servicio UsuariosService

```
export class UsuariosService {
  private urlEndpoint: string = URL_BACKEND + 'usuario/';
  constructor(private http: HttpClient, private router: Router, private headerService: HeadersService) {}
  countAllUsuarios(): Observable<any> {
    return this.http.get( url: this.urlEndpoint + 'countAll', options: {headers: this.headerService.agregarAuthHea
  }

  getUsuarios(page: number): Observable<any> {
    return this.http.get( url: this.urlEndpoint + 'page/' + page, options: {headers: this.headerService.agregarAut
  }

  getUsuarioByEmail(email: string): Observable<any> {
    return this.http.get( url: this.urlEndpoint + 'email/' + email, options: {headers: this.headerService.agregarA
  }

  getUsuarioById(id: number): Observable<any> {
    return this.http.get( url: this.urlEndpoint  + id, options: {headers: this.headerService.agregarAuthHeader()})
  }

  createUsuario(usuario: UsuarioRequest): Observable<any> {
    return this.http.post( url: this.urlEndpoint + 'registro/', usuario, options: {headers: this.headerService.agr
  }
```

Figura 76
Servicio CaisService

```
export class CaisService {
  private urlEndpoint: string = URL_BACKEND + 'cai/';
  constructor(private http: HttpClient, private router: Router, private headerService: HeadersService) {}

  createCai(cai: SubCaiRequest): Observable<any> {
    return this.http.post(this.urlEndpoint, cai, options: {headers: this.headerService.agregarAuthHeader()});
  }

  updateCai(id: number, cai: SubCaiRequest): Observable<any> {
    return this.http.put( url: this.urlEndpoint + id, cai, options: {headers: this.headerService.agregarAuthHeader
  }

  getAllCais(): Observable<any> {
    return this.http.get(this.urlEndpoint, options: {headers: this.headerService.agregarAuthHeader()});
  }
  getCais(page: number): Observable<any> {
    return this.http.get( url: this.urlEndpoint + 'page/' + page, options: {headers: this.headerService.agregarAut
  }

  deleteCai(id: number): Observable<any> {
    return this.http.delete( url: this.urlEndpoint + id, options: {headers: this.headerService.agregarAuthHeader()
  }
```

Recomendaciones

Para la primera versión, se recomienda un hosting con 8GB de RAM y TLS para comunicación segura.

La aplicación tiene un sistema de monitoreo de su actividad utilizando micrometer con prometheus y grafana. Para poder utilizar este monitoreo se ejecutan en docker tanto prometheus como grafana. Si no se desea usar docker, se puede instalar grafana localmente.

De igual manera se recomienda cambiar la seguridad e incluir Oauth 2.0 con MFA para hacer el sistema de seguridad aún más robusto.

Para que los usuarios que utilizaran el sistema para el registro y entrega de documentos extraviados se recomiendan los siguientes requerimientos técnicos mínimos:

- Computador portátil o de escritorio
- Conexión a internet de por lo menos 5MB
- Procesador dual core
- RAM 4GB
- Monitor resolución mínima 720p
- Teclado y mouse

Conclusiones

No existía un sistema On-line, para apoyar a las estaciones de Policía que recopile los documentos en custodia que se encuentran en cada uno de ellos y que los ciudadanos puedan consultar en caso que se tuvieran en protección por parte de la Policía Nacional- Metropolitana de Ibagué.

El sistema implementado permitirá a la Ciudadanía generar ahorro en dinero y en tiempo, ya que la herramienta tecnológica que permitirá si los documentos están en custodia por parte de alguna Estación de la policía poder recuperarlos sin tener que volver a solicitar la generación de los mismos, la cual puede durar entre 1 y 3 meses aproximadamente.

El sistema puede ser aplicado tanto en plataforma web, como en plataforma APP, para que se pueda implementar por parte de la Policía Nacional- Metropolitana de Ibagué que disponga la institución para su aplicabilidad y uso para la ciudadanía.

El diseño de la herramienta se realizó teniendo en cuenta los diferentes usuarios que pudieran utilizar, así como la seguridad para el mismo de acuerdo al rol asignado.

El sistema permite generar estadísticas de acuerdo al tipo de Documento, y estado de los documentos en custodia en que se encuentra por las estaciones de policía.

Referencias Bibliográficas

Arbeláez. (2014). Las tecnologías de la información y la comunicación (TIC) un instrumento para la investigación. *Investigaciones Andina*, *16*(29), 997-1000. http://www.scielo.org.co/scielo.php?script=sci_arttext&pid=S0124-81462014000200001&lng=en&tlng=es.

Briones (2015). *Análisis retrospectivo del cambio tecnológico en la agricultura, el modelo productivo y la economía ecológica. Revista Universidad y Sociedad*, *7*(3), 126-132. http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S2218-36202015000300019&lng=es&tlng=es

B, G. (2020, 3 diciembre). *MySQL Glosario Dic 03, 2020 Gustavo B. 5min de lectura ¿Qué es MySQL? Explicación detallada para principiantes*. www.hostinger.es. https://www.hostinger.es/tutoriales/que-es-mysql

Digitalización. (s. f.). https://www.innova.com.co/. https://www.innova.com.co/digitalizacion/#:~:text=T%C3%A9cnica%20que%20permite%20la%20reproducci%C3%B3n,le%C3%ADda%20o%20interpretada%20por%20computador.

Disposición final de documentos. (s. f.). https://minciencias.gov.co. https://minciencias.gov.co/glosario/disposicion-final-documentos#:~:text=Decisi%C3%B3n%20resultante%20de%20la%20valoraci%C3%B3n,%2C%20selecci%C3%B3n%20y%2Fo%20reproducci%C3%B3n.

Díaz Lazo, Juliet, Pérez Gutiérrez, Adriana, & Florido Bacallao, René. (2011). Impacto de las tecnologías de la información y las comunicaciones (tic) para disminuir la brecha digital en la sociedad actual. Cultivos Tropicales, 32(1), 81-90. http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S0258-59362011000100009&lng=es&tlng=es.

G, C. (2005, 6 septiembre). *LEY 962 DE 2005*. secretariasenado.gov.co. http://www.secretariasenado.gov.co/senado/basedoc/ley_0962_2005.html

Guía Para La Conservación De Documentos En Soportes Físicos. (2021, agosto). https://dapre.presidencia.gov.co/. https://dapre.presidencia.gov.co/dapre/DocumentosSIGEPRE/G-GD-01-conservacion-documentos.pdf

Gonzáles, C. (2021, 14 enero). El documento electrónico como documento electrónico de archivo: componentes y características. https://soaint.com. https://soaint.com/el-

documento-electronico-como-documento-electronico-de-archivo-componentes-y-caracteristicas/

I, E. (2013, 7 febrero). *¿Qué es Java Hibernate?* blog.educacionit.com. https://blog.educacionit.com/2013/02/07/que-es-java-hibernate/

Leyes desde 1992 - Vigencia expresa y control de constitucionalidad [DECRETO_0019_2012]. (s/f). Senado de la República de Colombia. http://www.secretariasenado.gov.co/senado/basedoc/decreto_0019_2012.html

Ley 1163 de 2007. (s/f). Gov.co. https://www.suin-juriscol.gov.co/viewDocument.asp?ruta=Leyes/1674994

M, J. (2006). Los sistemas de información documental: Consideraciones sobre sus características, concepto y funciones. En *Los sistemas de información documental: Consideraciones sobre sus características, concepto y funciones.* (pp. 138–150). Universidad Carlos III de Madrid.

Procedimiento para la estructuración y almacenamiento de documentos en el Sistema de Recuperación de Información Orión (N.º 123456789/7914). (2018). Universidad de las Ciencias Informáticas. https://repositorio.uci.cu/handle/123456789/7914

¿Qué es Angular y para qué sirve? (2019, 16 septiembre). www.qualitydevs.com. https://www.qualitydevs.com/2019/09/16/que-es-angular-y-para-que-sirve/

Qué es Git. (s. f.). atlassian.com. https://www.atlassian.com/es/git/tutorials/what-is-git

¿Qué es la tarjeta de identidad? (2016, 10 octubre). misabogados.com.co. https://www.misabogados.com.co/blog/que-es-la-tarjeta-de-identidad

¿Qué es la tarjeta de identidad? (2016, 10 octubre). https://www.misabogados.com.co. https://www.misabogados.com.co/blog/que-es-la-tarjeta-de-identidad
Sánchez, Collado, Martín Casas, Cano de la Cuerda. (2016). *Apps en neurorrehabilitación. Una revisión sistemática de aplicaciones móviles*. *Neurología (English Edition)*. https://doi.org/10.1016/j.nrleng.2015.10.002

Sánchez-Soto, A. (2016). Necesidades de información y comportamiento in- formativo de los agricultores de agave azul de Tequi- la, Jalisco: un estudio de caso. Investigación Bibliotecológica: Archivonomía, Bibliotecología e Información. Volume 30, Issue 70, Pages 11-286. http://ac.els-cdn.com/S0187358X16300648/1-s2.0-S0187358X16300648-main.pdf?_tid=8fbb2ade-55d6-11e7-90be-00000aab0f26&acdnat=1497976748_0da53531ac1be140d7c9b4572191ff6a

T, E. (s. f.). *Pasaporte*. economipedia.com. Disponible en 4 de diciembre de 2021, de https://economipedia.com/definiciones/pasaporte.html

Trujillo, E. (2020, 10 mayo). Pasaporte. https://economipedia.com. https://economipedia.com/definiciones/pasaporte.html

Westreicher, G. (2020, 6 agosto). Documento. https://economipedia.com. https://economipedia.com/definiciones/documento.html#:~:text=Es%20decir%2C%20un%20documento%20es,constancia%20de%20un%20hecho%20relevante.

W, G. (s. f.). *Documento*. economipedia.com. https://economipedia.com/definiciones/documento.html