SISTEMA DE ANÁLISIS SOBRE OCURRENCIAS DELICTIVAS USANDO MAPAS DE DATOS LIBRES PROVISTOS EN LA CIUDAD DE BOGOTÁ

Propuesta de proyecto de grado

Presentada a la facultad

de

Ingeniería

por

Camilo Pimienta

Aplicando al título

de

Especialista en ingeniería de software

Diciembre 2019

Universidad Antonio Nariño

Bogotá, Colombia

TABLA DE CONTENIDOS

LISTA DE TABLAS

LISTA DE FIGURAS

ABREVIACIONES

API	Application Programming Interface
GCP	Google Cloud Plataform
OSEMN	Obtain Scrub Explore Model iNterpret
UML	Unified Model Language
URL	Uniform Resource Locator
REST	REpresentational State Transfer

ABSTRACT

Pimienta, Camilo IT spc, Universidad Antonio Nariño, Diciembre 2019. Sistema de análisis sobre ocurrencias delictivas usando mapas de datos libres provistos en la ciudad de Bogotá.

One of the problems that cities face is the citizen insecurity. So, in order to improve this aspect, this project is software that processes maps information about delinquency, for predicting and undestanding behaviours in this context; through the use of machine learnning models in a distributed environment, achieving for this purpose scalability in the processing phase, as for future uses.

CAPÍTULO 1. INTRODUCCIÓN

La inseguridad ciudadana es un factor que ha tenido auge en los últimos tiempos en las principales ciudades del país colombiano, una de las ciudades más afectadas es la capital, Bogotá.

1.1 Situación actual

Diferentes tipos de delitos se presentan de forma frecuente, uno de éstos son los hurtos a personas, utilizando distintos de armas, del mismo modo, las víctimas podían movilizarse ya sea a pié en moto, u algún vehículo, al igual que su victimario. Un gran número de estas ocurrencias se reportan a la policía quien posee a su disposición una base de datos abierta, y con variables importantes, sobre los acontecimientos mencionados.

Las cifras de ocurrencias delictivas han ido aumentando desde el 2015, y en 2017 casi duplica los registros en solo cuestión de dos años (*Inseguridad en Bogotá: ¿aumentó el delito o cambió la forma de medirlo?*, n.d.).

1.2 Propuesta

Con el fin de aprovechar dicha información, se presenta este proyecto como una solución de software para recolección y análisis de ocurrencias delictivas, cuya finalidad será identificar patrones, entender comportamientos e identificar zonas de mayor riesgo; lo anterior, usando las variables propuestas por los datos libres registrados. Además, uno de los aspectos más importantes es la posibilidad de registrar una ocurrencia individual al sistema. De tal forma que a futuro pueda ser utilizado por

los mismos ciudadanos, para registrar y visualizar los resultados de los algoritmos que se aplicarán sobre los datos.

1.3 Estructura del documento

En el capítulo 4 se expondrá el estado actual del problema, limitaciones y objetivos. Luego, el capítulo 2 se presenta una breve descripción de terminologías importantes para el entendimiento del documento. Posteriormente, el estado del arte se presenta en el capítulo 3. Después, el capítulo 5 presenta el modelo de ciclo de vida utilizado para llevar a cabo este desarrollo. El capítulo 6 presentará desde la descripción de los objetivos en términos de requerimientos funcionales y no funcionales; además, distintas vistas de diseño, desde los casos de uso hasta diagramas detallados de estructura y comportamiento. Finalmente, en 7 se planteará discusiones acerca del proyecto y futuro trabajo.

CAPÍTULO 2. MARCO TEÓRICO

Dentro del presente capítulo se tratará dos temas relevantes que abarcan el contexto del problema. En principio, se introducirá brevemente al aprendizaje estadístico en la sección 2.1 como uno de los factores de gran relevancia en el caso de estudio presentado más adelante; del mismo modo, se utilizará diferentes terminologías sobre tecnologías de software, una breve descripción de éstas estará en 2.2.

2.1 Acerca del aprendizaje estadístico

Según James, Witten, Hastie, and Tibshirani (2013) es un conjunto de herramientas que sirven para entender la información. Puede ser clasificada en aprendizaje supervisado y no supervisado. En el primero encontramos variables de entrada y una variable objetivo que se busca predecir/estimar. A diferencia del aprendizaje no supervisado, dicha variable objetivo no existe, entonces solo se tiene las variables de entrada, que se busca agrupar por comportamientos; sin embargo, se puede aprender de las relaciones que encontremos, como de la estructura de dicha información. En este proyecto hay un mayor énfasis en el aprendizaje no supervisado dada la naturaleza de los datos.

Uno de los estadísticos usados para el aprendizaje no supervisado es el K-Means Cover and Hart (1967). Este modelo, dado un número dado de clústers, calcula centroides al azar dentro del espacio de las variables y a los registros más cercanos le asigna una etiqueta del centroide, este proceso se repite un determinado número de veces con el fin de encontrar puntos donde las distancias de los puntos de una etiqueta y el centroide sea la mínima.

Una de las cualidades en este estadístico es que no sugiere un número de clústers a utilizar, a diferencia del modelo jerárquico para el aprendizaje no supervisado. Por esta razón existen varios paramétricos que permiten identificar el número óptimo de clústers para aplicar el K-Means. En Kodinariya and Makwana (2013) se presentan varios de métodos que se puede utilizar para encontrar el número óptimo de clústers y éstos son:

- Regla del pulgar

  Es un cálculo simple que aproxima el número óptimo de clúster dependiendo al número de registros. Se sigue por la ecuación:

  $$k \approx \sqrt[2]{n/2} \tag{2.1}$$

  Donde n es el número de registros

- Método del codo

  Este es un método visual. Para calcularlo se debe entrenar los datos con un $k = 2$ e ir incrementando este número en una unidad. Calcular la función de costo en cada una de las pruebas, finalmente, se grafica una función que relaciona el valor del costo en función del número de clústers. La gráfica normalmente viene en decrecimiento, sin embargo, este decrecimiento tiene una pendiente bastante negativa en un inicio, luego, en algún punto de la figura, la pendiente se acerca más a cero, es decir, decrece más lento. Muchas veces de manera visual se puede observar el punto en que este decrecimiento se reduce de manera drástica, esa sería como una representación de un codo y el número de clústers que coincide con dicho punto debe ser el elegido.

- Mediante el cálculo de la silueta.

  Este paramétrico es basado en una distancia promedio entre clústers, el valor ronda entre 0-1 donde uno busca que el número sea lo más cercano a 1, dado que muestra una homogeneidad de los datos. Se sigue por la función:

  $$s(i) = \frac{b(i) - a(i)}{max(a(i), b(i))} \tag{2.2}$$

Este es un método que requiere considerablemente una gran capacidad de cómputo en comparación al método del codo.

2.1.1 Aprendizaje no supervisado

Dentro del aprendizaje no supervisado se cuenta con un conjunto de variables sobre un conjunto de observaciones. Para este contexto, predecir no pasa a ser uno de los fines directos, dado que, en términos de los datos, no hay una variable objetivo para hacerlo. Una de las preguntas que busca resolver el aprendizaje no supervisado es si existen subgrupos dentro de los datos.

2.2 Conceptos de software

Los capítulos posteriores contienen un componente fuerte de aspectos técnicos en la construcción del software, por consiguiente, se introducirán los más importantes, como lo son: la arquitectura, los atributos de calidad y los requerimientos

2.2.1 Arquitectura

En Bass, Clements, and Kazman (2003) definen la arquitectura del software como la estructura de estructuras del sistema, que comprende los elementos del software, como sus propiedades visibles de dichos elementos y las relaciones entre ellos. De esto, se resaltan dos características principales y son la estructura, como los elementos que componen el sistema, y el comportamiento, siendo éste la relación de estos últimos.

Además, Fowler (2002) destaca,dos aspectos a tener en cuenta y es la descomposición de un sistema en partes; y, las decisiones que son difíciles de tomar. Por tanto, cuando se habla de arquitectura, se refiere a aquel diseño de un sistema que

facilitará su desarrollo, éste será construido en función necesidades que debe solventar el sistema.

2.2.2 Atributos de calidad

En la ISO 25000 se definen los atributos de calidad como el grado en que unas características inherentes alcanzan los requerimientos establecidos (ISO, 2008). Esto es, alcanzar satisfacción del usuario en requerimientos específicos por medio de características estándares en el software.

2.2.3 Requerimientos funcionales y no funcionales

Las necesidades que busca solventar un producto de software son descritas en términos técnicos en requerimientos funcionales y no funcionales, entonces, es un contrato que describe los servicios que debe proporcionar un software.

Estos requisitos pueden ser categorizados en requerimientos funcionales y no funcionales; Sommerville (2005) los define los requerimientos funcionales como: "declaraciones de los servicios que debe proporcionar el sistema, de la manera en que éste debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares" (p.109).

Del mismo modo, define los requerimientos no funcionales de la siguiente manera: "Son restricciones de los servicios o funciones ofrecidas por el sistema. Incluyen restricciones de tiempo, sobre el proceso de desarrollo y estándares". (Sommerville, 2005, p.109).

CAPÍTULO 3. ESTADO DEL ARTE

Esta sección se presenta un proyecto similar que está relacionado con la problemática expuesta por este trabajo, principalmente, el por qué dicha solución no bastan para abarcar los objetivos expuestos al inicio. De igual forma, se presenta el impacto del llevar a cabo el proyecto. finalmente, se presenta el factor de innovación.

3.1 Revisión de literatura

Retomando los objetivos expuestos anteriormente, las principales características de este proyecto presentan una problemática enfocada hacia un dominio que es la ciudad de Bogotá, será un análisis de comportamiento utilizando herramientas de procesamiento de datos con la particularidad de ser escalable en todos sus aspectos (incluyendo la carga de datos por usuarios individuales). Se utilizará algoritmos de machine learnning con el fin de atribuir indicadores y comparaciones sobre los resultados. Finalmente, se buscará el algoritmo más apropiado para la tendencia presentada.

Por tanto, el dominio de proyectos asociados está relacionado con estudios espaciales sobre el comportamiento y herramientas que permiten visualizar datos de la misma naturaleza. Existen muchas investigaciones de carácter social, éstas investigaciones están por fuera del alcance dado que el área de conocimiento difiere, como sus metodologías.

3.1.1 ArcGIS

Es un framework que dispone de funcionalidades específicas para análisis de mapas, cuenta con un algoritmo que permite realizar operaciones de limpieza sobre

datos, sin embargo, dichas funcionalidades están disponibles en versiones pagas, además, solo soporta un tipo de análisis genérico para el procesamiento. Por tanto, podría haber mejores algoritmos que se adapten mejor al escenario tratado en este proyecto. Finalmente, no posee el componente de registro de ocurrencias.

3.1.2 Adenunciar

Es una aplicación móvil de la policía nacional que permite denunciar diferentes de delitos. Busca agilizar trámites relacionados a ciertos actos de delincuencia, esta información será validada ellos para tomar acción.

La principal característica por la que no se alcanza a abarcar los requerimientos de este proyecto es por la manera en que se utiliza esta información, a pesar de que permite registrar acontecimientos, no se hablar de algún análisis posterior en función a las necesidades de un ciudadano. Es decir, esta es una solución reactiva, antes que preventiva, como lo buscaría ser la implementación presentada en este proyecto.

3.2 Impacto

El principal impacto de este proyecto será brindar una herramienta de software para la comunidad que permita registrar acontecimientos, lo anterior, con el fin de clasificar, entender comportamientos delictivos en distintas zonas e identificar patrones en los modus operandi.

3.3 Factor de innovación

Este proyecto busca proveer un software con estadísticas espaciales sobre criminalidad en la ciudad de Bogotá. Esta aplicación es orientada hacia los ciudadanos, por dicha razón, se mantendrá como software libre, además, haciendo uso de una prueba de concepto de análisis de datos sobre los registros de la policía.

CAPÍTULO 4. PLANTEAMIENTO DEL PROBLEMA

En esta sección se presenta la descripción del problema y cuáles serían las principales contribuciones de este proyecto

4.1 Descripción del problema

La inseguridad en la ciudad de Bogotá tiene altas tasas de delincuencia. La policía nacional posee una base de datos libre con información sobre algunos de los actos delictivos ocurridos. Sin embargo, no hay una herramienta que haga uso adecuado de dicha información a la disposición de los ciudadanos, que le permita observar de manera libre y registrar acontecimientos.

La figura 4.1 podemos observar un resumen de los datos por departamentos, a pesar de la diferencia de tamaños de cada uno de éstos, se observa un punto que concentra una gran cantidad de las ocurrencias nacionales, la capital.

4.2 Objetivo principal

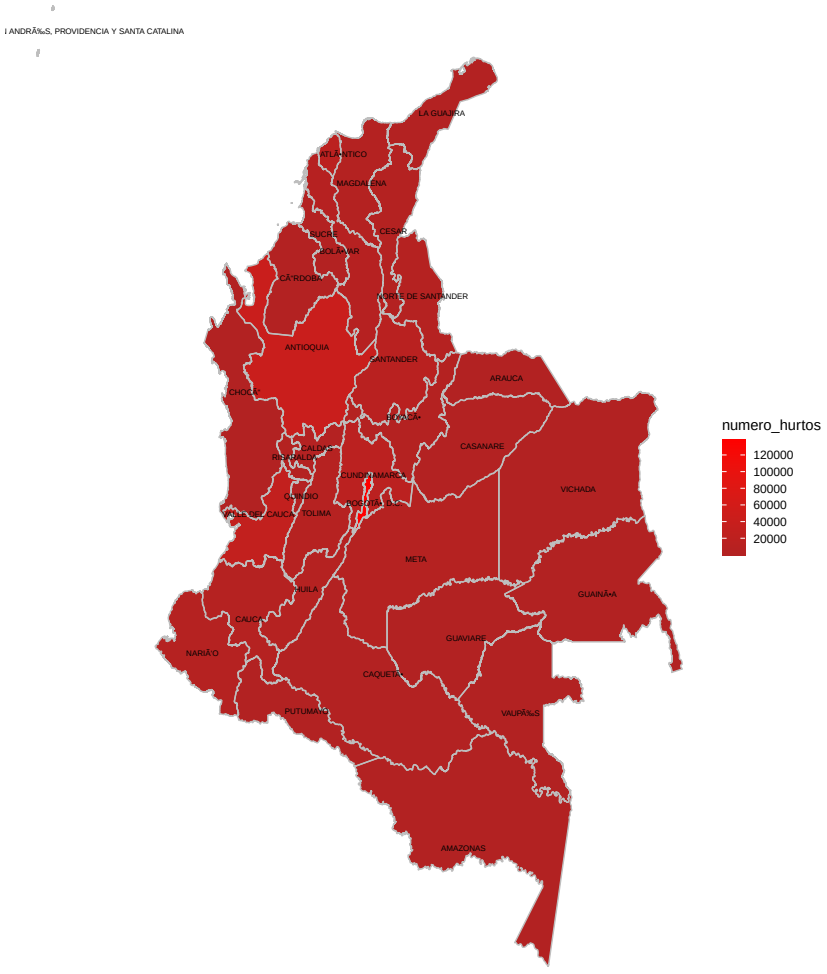
Implementar una solución de software que permita registrar, procesar y visualizar incidencias delictivas, este segundo por medio de modelos de análisis datos de geolocalización.

4.3 Objetivos específicos

- Construir una solución que permita registrar ocurrencias delictivas y almacenarlas.

- Permitir a un usuario procesar, mediante modelos existentes, los datos de geolocalización con tal de obtener patrones de comportamientos.
- Diseñar una solución que permita visualizar los resultados del ítem anterior.

Figure 4.1. Representación tomada de los registros de la policía nacional. Adaptación de autoría propia



CAPÍTULO 5. METODOLOGÍA

5.1 Ciclo de vida del proyecto

Para llevar a cabo este proyecto se ha adoptado un enfoque adoptando una metodología base, sin embargo, se rescatan conceptos de otras. Dado el contexto del problema, la programación extrema (Beck, 1999) se adapta más a este proyecto, dado que permite adaptarse sobre la marcha, además, existe un tiempo bastante limitado de entrega y que el equipo es mínimo.

Para realizar este proyecto se tuvo en cuenta un enfoque cualitativo presentado por Creswell (1994) de ciclo de vida de software, esto, con el fin de cumplir, implementar y evaluar los objetivos propuestos en esta tesis.

Las fases que conformaron esta tesis constan de: elicitación, diseño, implementación y pruebas; de esta forma brindando un prototipo funcional de software. A continuación se presenta los objetivos principales de cada una de las fases:

- **Análisis y elicitación**

    1. Identificación del problema y sus causas, esto con el fin de proponer soluciones hacia dicha problemática.
    2. Revisión sistemática del estado del arte, donde se busca las principales herramientas cuyas características sean más adecuadas a la problemática.
    3. Descripción de los objetivos, éstos planteados en términos de requerimientos funcionales y no funcionales.

- **Diseño**

1. Diseño utilizando de diagramas detallados, éstos seguidos por el estándar UML.
2. Modelos conceptuales siguiendo nomenclaturas no formales con el fin de familiarizar entidades utilizadas para el desarrollo.
3. Diseño de arquitectura de la solución.

- **Desarrollo y pruebas**

1. Implementación del ETL con el fin de extraer y organizar la información.
2. Implementación de los modelos de clasificación y predicción aplicado a la información recolectada.
3. Implementación de visualizadores de información con el fin de mostrar y analizar los resultados.

- **Pruebas**

1. Integración del código fuente a herramientas de código estático.
2. Verificación de pruebas funcionales del código bajo test unitarios.

5.2 Etapas para procesamiento de datos

Transversal a esto, y teniendo en cuenta el componente de análisis de grandes cantidades de datos, se plantean varias etapas relacionadas al alcance de procesamiento de datos, éstas basadas en el modelo presentado por Mason (2010) -comúnmente conocido como OSEMN por sus siglas en inglés-, quien discrimina unas etapas para un proyecto de analítica de datos y son:

- **Obtener**

1. En esta etapa se obtiene la información que va a ser procesada.

- **Limpiar**

1. El siguiente paso es limpiar y filtrar los datos dado que el no hacer esto deja el análisis propenso a resultados no equívocos. Aquí se debe consolidar la información en un repositorio único. También, eliminar o filtrar registros incompletos, inconsistentes, etc.

- **Explorar**

1. Analizar las variables que provee el conjunto de datos, identificar las propiedades, los tipos de datos, cómo se categoriza la información, es decir, elegir el tipo de dato estadístico.

- **Modelar los datos**

1. Esta fase será crucial, dado que en esta etapa se elige el estadístico a utilizar y el modelo. Ya sea una categorización, una regresión, etc.

- **Interpretar los datos**

1. Finalmente, entender los resultados y presentarlos, que para este caso será un mapa con diferentes variables. El objetivo de esta etapa es poder exponer los datos de una forma no técnica.

Éstas etapas básicamente resumen o engloban las fases de un proyecto de aprendizaje sobre datos.

La documentación se puede encontrar en el Anexo 1.

CAPÍTULO 6. PROCESO DE SOFTWARE

En este capítulo se presentará los principales modelos que integran la estructura de la implementación y su modelo. En principio se hablará de la arquitectura de alto nivel, luego, diagramas detallados que son: casos de uso, diagrama de clases, componentes, de despliegue y de secuencia; finalmente, se presentará la especificación de tecnología.

6.1 Restricciones de tecnología

En principio, se tiene la restricción del manejo de mapas, para el almacenamiento de datos se utiliza (Postgres para mapas o el de Spark). Del mismo modo, debe haber una transferencia de información, por tal motivo, existe un estándar llamado GeoJson para dicha tarea.

GeoJson permite una comunicación textual de datos espaciales entre aplicaciones remotas, dado que los descriptores XML y Json no son suficientes para alcanzar el objetivo necesario.

Para persistir la información se tenían dos opciones, los factores utilizados para elegir entre éstos son: facilidad de integración con la tecnología y de transformación y procesamiento por las librerías ofrecidas por Spark; del mismo modo, otro factor crucial e implícito es que sea de libre uso.

Cabe anotar que utilizar herramientas en la nube es muy buena opción para este tipo de proyectos, el conocimiento para llevar a cabo la integración no es inconveniente. Sin embargo, por restricciones monetarias, se optó por no acudir a servicios en la nube, debido a que las capas gratuitas de las nubes privadas más reconocidas (AWS, GCP y Azure) normalmente duran 1 año, además de tener servicios limitados. Dado que este proyecto tomó un poco más de tiempo del pronosticado y el

uso de las herramientas, como los cálculos asociados, podrían salirse de la capa gratuita, se optó por no utilizar recursos en la nube; siendo considerado un riesgo el poder quedar sin recursos de la plataforma en etapas avanzadas.

6.2 Requerimientos funcionales

Los requerimientos funcionales se presentan en las Tablas 6.1, 6.2, 6.3, 6.4 y 6.5

6.3 Requerimientos no funcionales

Los requerimientos no funcionales se describen en las tablas 6.6, 6.7 y 6.8.

6.4 Diseño detallado

En esta sección se presentará los aspectos más relevantes de la etapa de diseño, desde la especificación de requerimientos funcionales y no funcionales, pasando por los diagramas de casos de uso, UML de clases y secuencia.

6.4.1 Arquitectura

En términos de arquitectura se ha seleccionado una arquitectura de microservicios, lo anterior, con el fin de alcanzar un mejor rendimiento, esto en el momento que se produzca un crecimiento, en las peticiones, causada por alta demanda.

Para el caso del proyecto, la Figura 6.1 evidencia el escenario particular del diseño de alto nivel de la solución, donde existe un servicio que expone un API REST (capa 1), posterior a esto, en la capa 2, lo que vendría a ser los nodos de procesamiento o servidores, cuya cantidad dependerá del uso del servicio. Finalmente, un motor de base de datos optimizado para registros espaciales PostGis, persistiendo la información de los datos (capa 3).

Table 6.1

Requerimiento funcional 1

UAN UNIVERSIDAD ANTONIO NARIÑO	**Formatos** **Requerimientos - casos de uso**	
	Arquitectura de software I	**2019**
Requerimiento funcional		
Nombre	Cargar información espacial	**1**
Descripción: El sistema debe permitir a un usuario administrador cargar archivos de extensión CSV al sistema con un peso máximo de 1Gb por archivo		
Actores: Sistema		
Precondiciones: Hay información espacial en un archivo CSV		
Flujo Normal		
Acción del actor	**Respuesta**	
1. El sistema recibe un Json	2. Validar los datos. 3. Almacenar los datos 4. Emitir mensaje de éxito	
5. El sistema carga el archivo por lotes para procesarlo posteriormente	6. Archivo cargado y listo para procesar	
Flujo Alternativo		
1. El archivo pesa más de 1Gb	2. Se emite un mensaje explicando que el archivo pesa más del máximo permitido	
3. Existen archivos de otra extensión	4. El sistema ignorará este tipo de archivos para no tenerlos en cuenta	
Poscondiciones Una parte del archivo es cargada por lotes en memoria		

Table 6.2

Requerimiento funcional 2

Requerimiento funcional		
Nombre	Transformar información espacial cargada en datos espaciales para ser persistidos	**2**
Descripción: El sistema debe permitir interpretar el archivo con información espacial en un nuevo tipo de dato espacial para su posterior información		
Actores: Sistema		
Precondiciones: Hay un archivo CSV cargado en memoria.		
Flujo Normal		
Acción del actor	**Respuesta**	
7. El sistema debe iterar sobre los campos existentes en el archivo cargado y transformar esta información en un tipo de dato procesable.	8. El tipo de dato requerido es encontrado	
Flujo Alternativo		
5. El archivo no posee la información en términos de columnas reconocidas obligatorias	6. Se emite un mensaje de error donde se especifica que no existe la información suficiente para procesar	
Poscondiciones Ninguna.		

Table 6.3

Requerimiento funcional 3

Requerimiento funcional		
Nombre	Almacenar los datos espaciales persistidos en un repositorio listo para ser procesado	**3**
Descripción: El sistema debe permitir almacenar los nuevos datos espaciales en un repositorio entendible por el sistema		
Actores: Sistema		
Precondiciones: Existen objetos del tipo de datos requerido cargado en memoria.		
Flujo Normal		
Acción del actor	**Respuesta**	
9. El sistema debe permitir persistir los datos ingresados en su repositorio.	10. Los datos son almacenados de forma correcta.	
Flujo Alternativo		
5. El archivo no posee la información en términos de columnas reconocidas obligatorias	6. Se emite un mensaje de error donde se especifica que no existe la información suficiente para procesar	
Poscondiciones Repositorio con datos nuevos.		

Table 6.4

Requerimiento funcional 4

Requerimiento funcional		
Nombre	Procesamiento de los datos del repositorio	**4**
Descripción: El sistema debe permitir aplicar algoritmos de clasificación a los datos almacenados en su repositorio con el fin identificar comportamientos		
Actores: Sistema		
Precondiciones: Existe un repositorio con datos espaciales para ser procesados.		
Flujo Normal		
Acción del actor	**Respuesta**	
11. El sistema debe permitir pre-procesar la información en busca de datos corruptos para que no sean tenidos en cuenta.	12. Se realiza la pertinente limpieza de datos para su posterior procesamiento.	
Flujo Alternativo		
13. El sistema debe permitir aplicar un algoritmo de clasificación sobre los datos limpios	14. Datos espaciales clasificados de forma correcta.	
Poscondiciones Repositorio con datos nuevos.		

Table 6.5

Requerimiento funcional 5

Requerimiento funcional		
Nombre	Visualización de resultados	**5**
Descripción: El sistema debe permitir visualizar gráficamente los resultados de los algoritmos utilizados.		
Actores: Sistema		
Precondiciones: Se tienen resultados espaciales producto de aplicar algoritmos de diferentes familias sobre un conjunto de datos.		
Flujo Normal		
Acción del actor	**Respuesta**	
15. El sistema debe permitir graficar los resultados de los análisis y exportarlos como PDF.	16. Un archivo de extensión PDF disponible para la descarga.	
17. El sistema debe permitir la descarga de los archivos de visualización.	18. Los bytes de un archivo PDF con el diagrama correspondiente.	
Flujo Alternativo		
11. El resultado no es suficiente para realizar un diagrama	12. Se emite una excepción de negocio donde se puede visualizar el error correspondiente.	
13. Se intenta descargar un archivo inexistente.	14. Se emite una excepción de recurso no encontrado.	
Poscondiciones Repositorio con datos nuevos.		

Table 6.6

Requerimiento no funcional 1

Requerimiento no funcional		
Nombre	Volumen de datos	**1**
Tipo	Necesario	
Descripción	La aplicación debe poder ser escalable cuando la cantidad de información aumente. Por tanto, el sistema debe obtener mejores tiempos de respuesta al procesar la misma cantidad de información a medida que aumente el número de dispositivos de hardware.	
Criterios de aceptación	El sistema debe permitir al menos 50 usuarios concurrentes para la subida de información.	

Table 6.7

Requerimiento no funcional 2

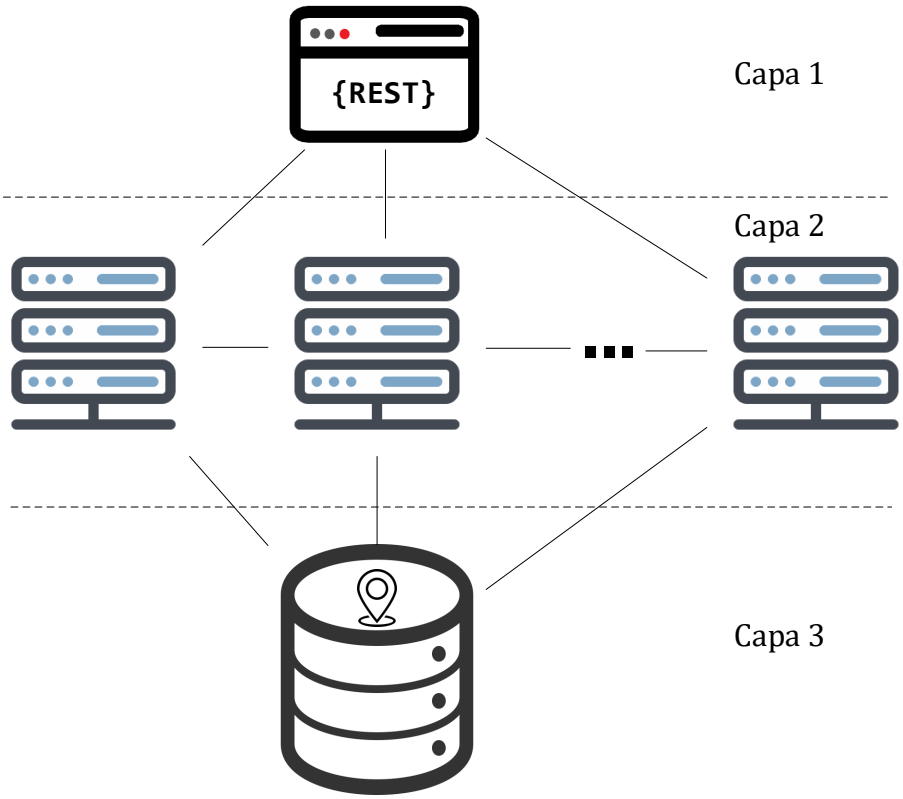
Requerimiento no funcional		
Nombre	Autenticación	**2**
Tipo	Necesario	
Descripción	El API debe permitir la autenticación a un usuario invitado que exista en la base de datos local.	
Criterios de aceptación	Como usuario invitado quiero poder loguearme al sistema para poder poblar los registros de la base de datos.	

Table 6.8

Requerimiento no funcional 3

Requerimiento no funcional		
Nombre	Mantenibilidad	**3**
Tipo	Necesario	
Descripción	El software debe poder ser mantenido y cambiado si el escenario de uso (criminalidad) varía.	
Criterios de aceptación	Para este caso se utilizarán reglas de desacoplamiento de la solución que permitirá la fácil adaptación del software a nuevos casos, pero manteniendo su funcionalidad descrita en los requerimientos funcionales.	

Figure 6.1. Arquitectura en vista de alto nivel. Autoría propia

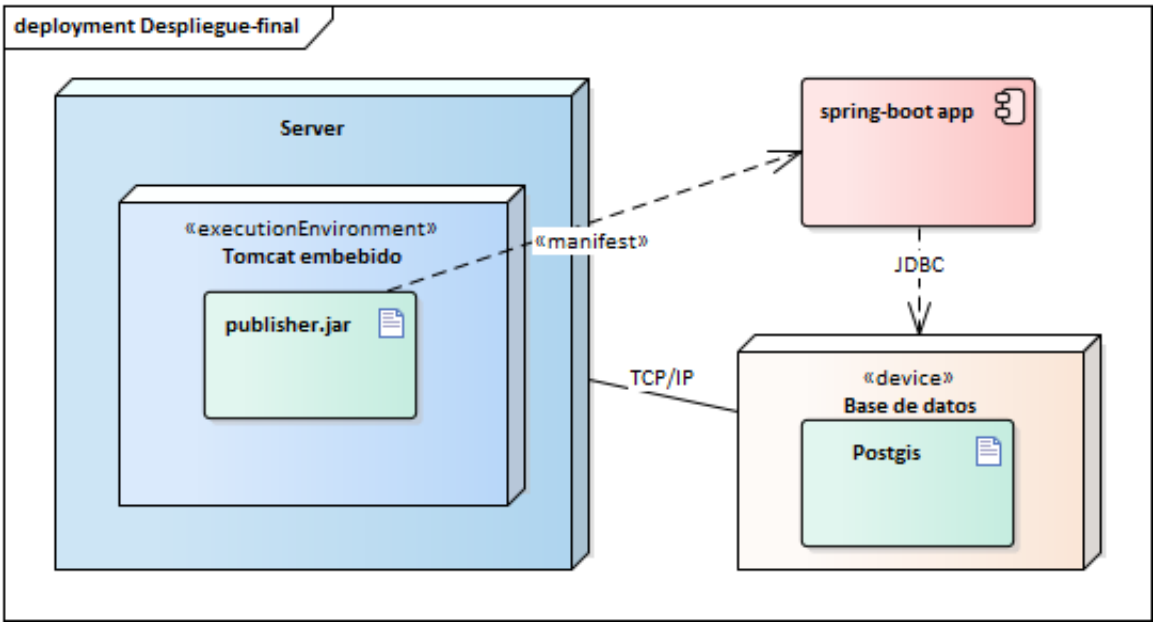


6.4.1.1. Despliegue

La figura 6.2 es el diagrama de despliegue propuesto para la solución. Como se puede ver, a un alto nivel, se requieren mínimo dos nodos de procesamiento, el primero contendrá el servidor de base de datos; el otro nodo contendrá la aplicación en Spring boot, que se despliega dentro de un entorno de ejecución de Tomcat embedido, para exponer el ejecutable del software: *publisher.jar*. Como bien muestra el diagrama, este ejecutable expone los servicios descritos en la figura 6.14 por medio de un API REST, que pueden ser utilizados bajo el protocolo HTTP desde cualquier dispositivo con soporte de peticiones de esta naturaleza.

La peculiaridad de esta arquitectura es que es posible replicar el nodo del servidor principal a demanda, lo anterior, para garantizar un mejor desempeño a medida que haya mayor cantidad de peticiones.

Figure 6.2. Diagrama de despliegue. Autoría propia

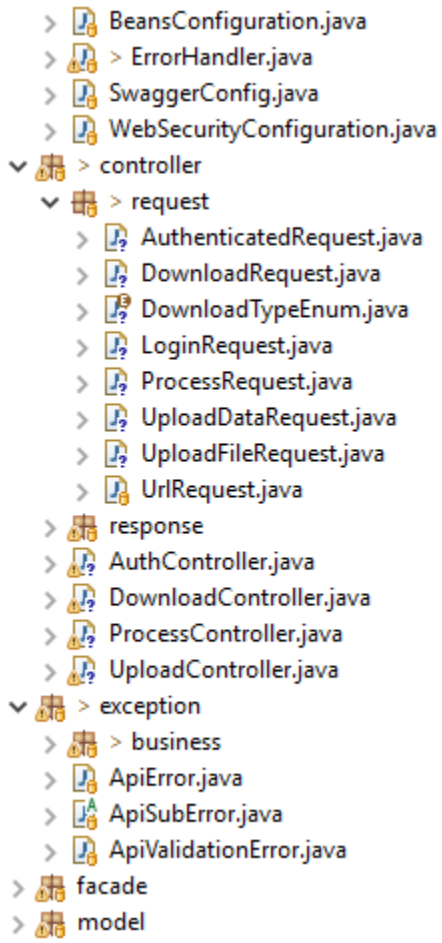


6.4.2 Construcción y documentación del API

Dentro de esta sección se detallará los entregables más significativos por medio de capturas de pantalla de la solución, éste será la documentación del API por la cual se interactúa con el software; buscando de esta manera, alcanzar los requerimientos funcionales descritos:

Con el fin de mantener estándares básicos de seguridad, se maneja autenticación vía REST por medio de HEADER's, para dificultar la lectura de las credenciales por la petición, dado que si viaja por la URL es demasiado insegura y en el *body* es preferible evitarlo. Esto se evidencia en la figura 6.5. Este servicio responde con un Token que tiene vigencia temporal, de esta forma se puede disminuir el riesgo del

Figure 6.3. Paquetes del aplicativo. Autoría propia



robo de sesión por descuido por parte del usuario. Dicho Token se encarga de autenticar al usuario para usar los demás recursos provistos por el API.

De este punto en adelante, todos y cada uno de los servicios utiliza el Token de autenticación para acceder a los recursos, de esta forma se alcanza una sesión sin estado (Stateless). Es importante tener en cuenta este comportamiento, dado que permite que cada artefacto desplegado sea independiente, y distintas peticiones de una misma transacción -siendo una transacción, por ejemplo, la subida de un archivo junto con la consulta de su estado- puedan realizarse en diferentes nodos, transparente al usuario.

Figure 6.4. Documentación del API REST. Autoría propia

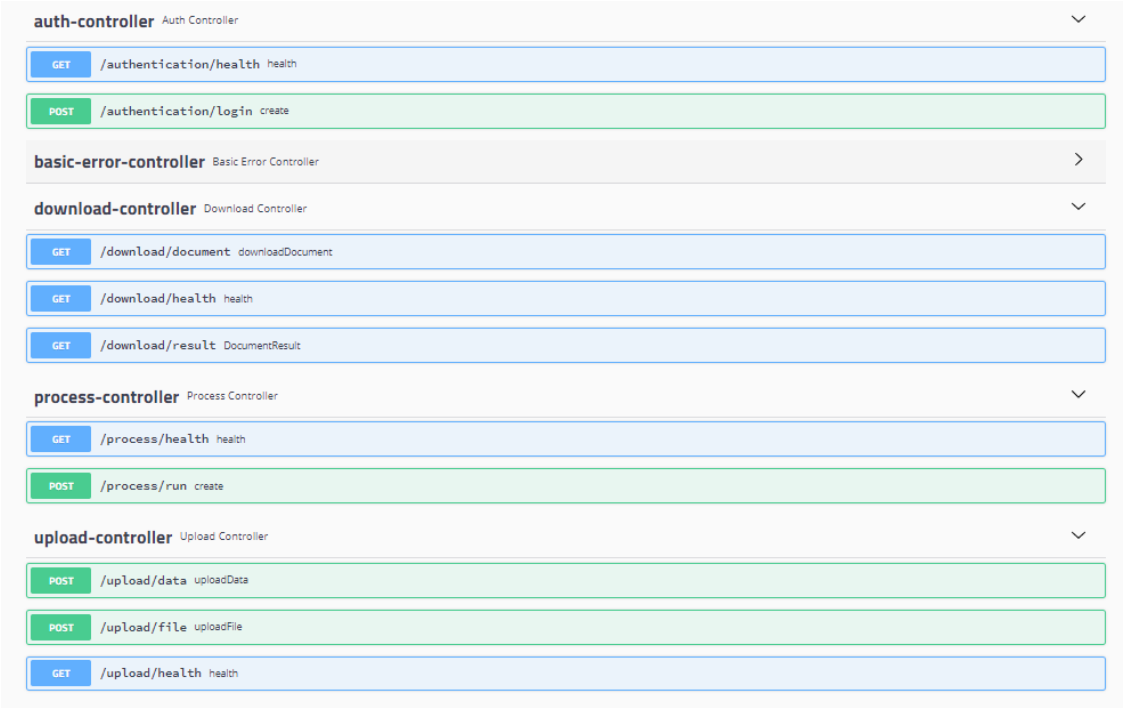
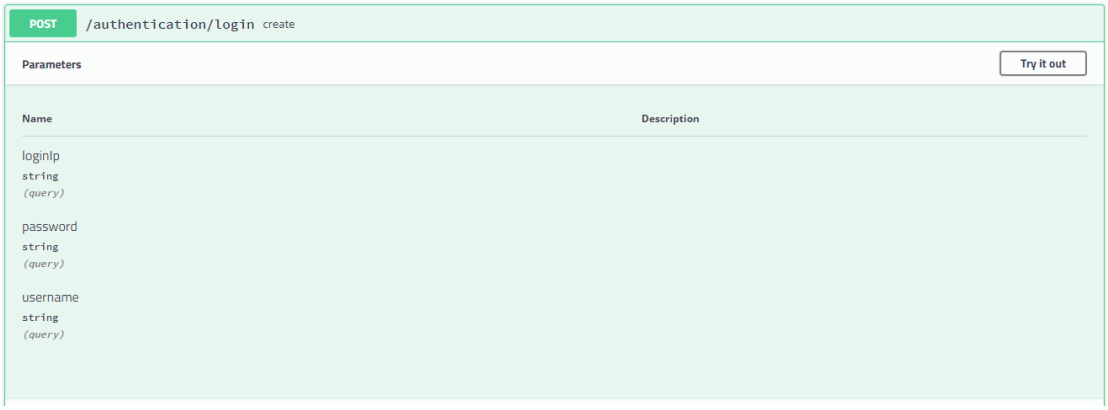
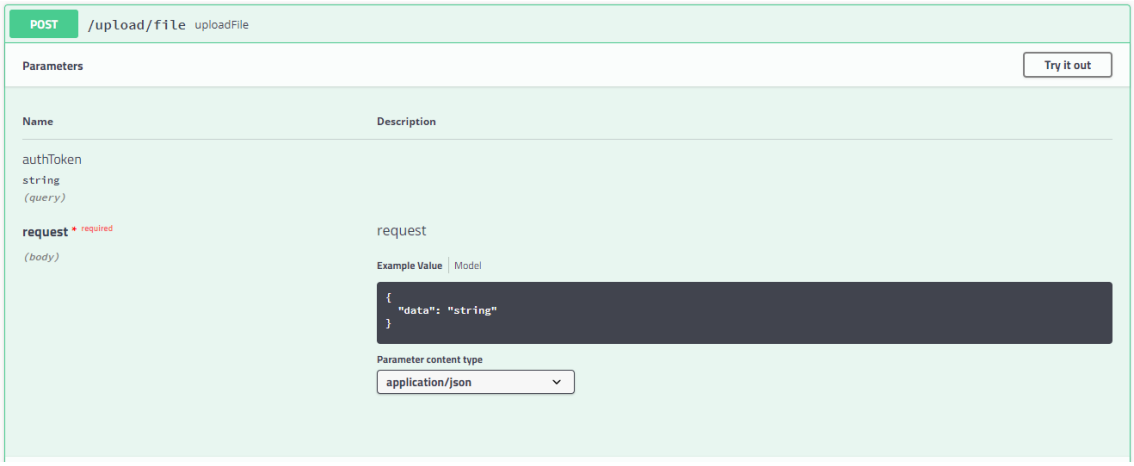


Figure 6.5. Autenticación por medio de servicios REST. Autoría propia



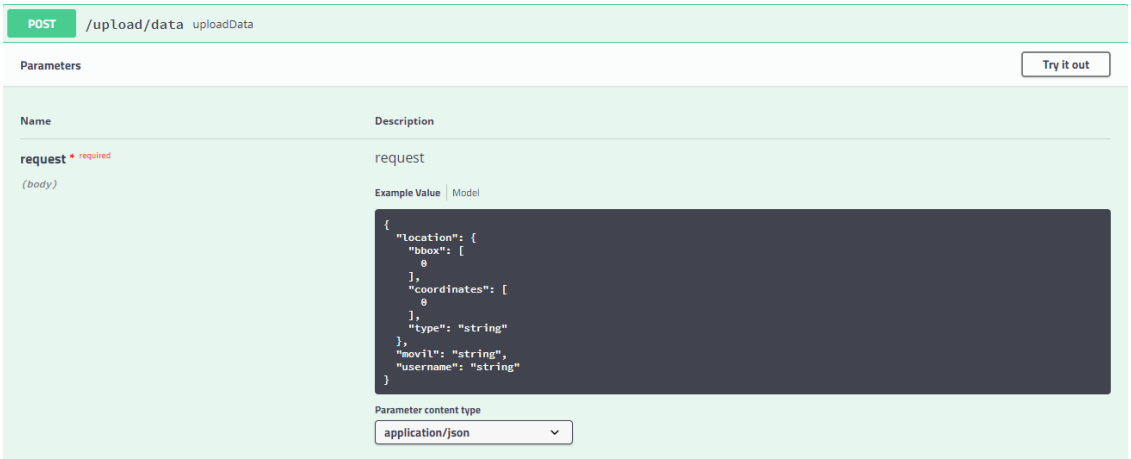
Entrando en contexto, la figura 6.6 evidencia la representación del requerimiento referido en la Tabla 6.1, la transformación y posterior procesamiento -Tablas 6.2 y 6.3- ocurren en la misma interacción.

Figure 6.6. Carga de información por medio de servicios REST. Autoría propia



Ahora, en la Figura 6.7 evidencia la funcionalidad de carga de un dato específico de localización, en el request el tipo de dato deberá ser un punto con una longitud y una latitud asociada, también, dentro del GeoJSon se evidencia la captura de otra información opcional para la persistencia.

Figure 6.7. Captura de pantalla de subida de información geográfica. Autoría propia



Del mismo modo, en la Figura 6.8 se relaciona con los requerimientos funcionales descritos en las Tablas 6.4 y 6.5; donde se permite la descarga de un resultado en particular.

Figure 6.8. Descarga de documento por medio de servicios REST. Autoría propia

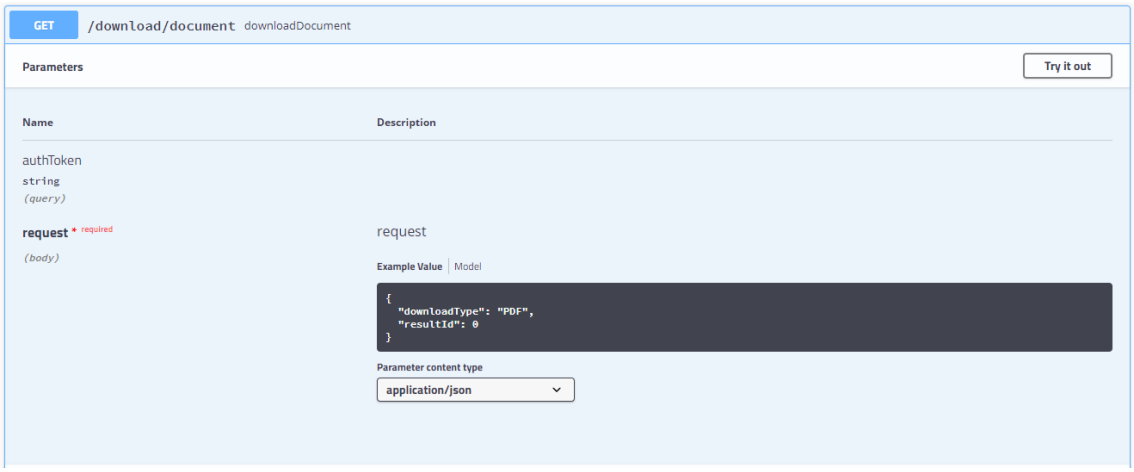
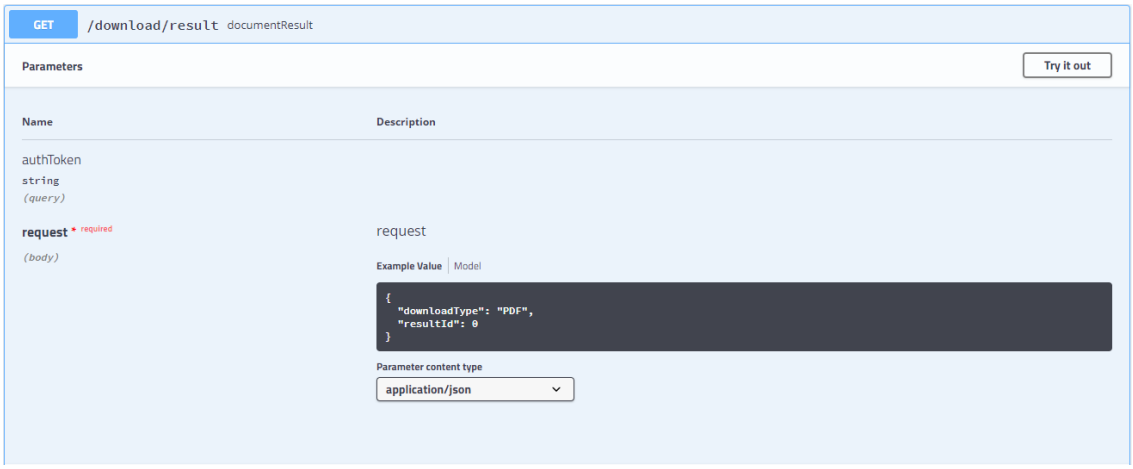


Figure 6.9. Descarga de un resultado particular por medio de servicios REST. Autoría propia

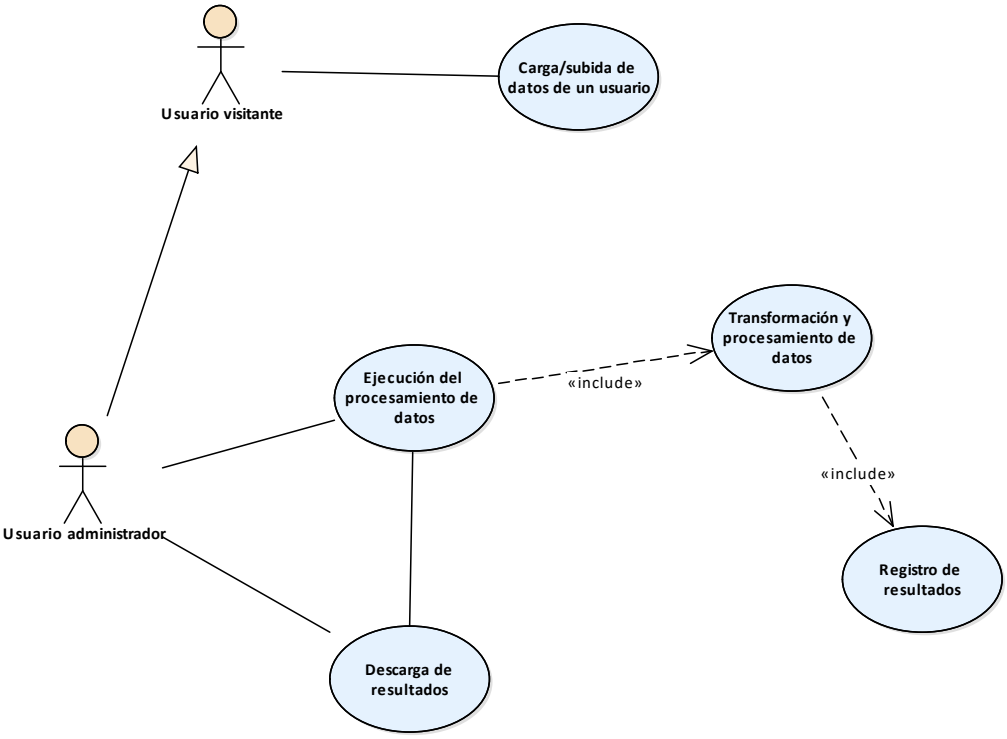


6.4.3 Vistas 4+1

6.4.3.1. Diagrama de casos de uso

Dentro de la definición del problema se plantea los requerimientos de casos de uso, esta vista se muestra en la Figura 6.10. En el modelo de 4+1 esta vista sería la del caso de uso.

Figure 6.10. Diagrama de casos de uso. Autoría propia



6.4.3.2. Diagrama de clases

En la Figura 6.11 se muestra el diagrama de clases de la solución, compuesta por tres capas que son: Capa de controladores, capa de servicios y capa de modelo de datos. En el modelo 4+1 esta vista representa

La figura 6.12 presenta el modelo de datos usado en particular para el módulo de persistencia.

6.4.3.3. Diagrama de secuencia

La Figura 6.13 presenta el diagrama de secuencia para el caso de uso de subida de datos

6.4.3.4. Diagrama de componentes

La Figura 6.14 presenta el diagrama de componentes para la solución

6.4.4 Procesamiento de datos

Como se mencionó en 5.2, se llevó a cabo el proceso del procesamiento de datos que consta en los pasos descritos en dicha sección.

En primer lugar, se obtuvo los datos correspondientes a los registros de la policía nacional, éstos, almacenados en formato csv.

Luego, en la limpieza, se utilizó un Jupiter Notebook, cuya evidencia está en el adjunto 1. En síntesis, esta etapa constó de un filtro de las ocurrencias registradas en la ciudad de Bogotá. Por cada variable, se realizó un análisis de las filas, para identificar aquellas vacías o inconsistentes. Dependiendo de la naturaleza de la variable (cualitativa o cuantitativa) se escogió herramientas para llevar a cabo este filtro: dentro de las cualitativas se hizo por conteos de registros vacíos; las cualitativas, por medio de análisis de tendencias central, identificando tanto valores máximos como mínimos. Donde, por ejemplo, se puede identificar edades de personas de un año como mayores

de 100 años o menores de 6 años reportando estos casos, suponiendo errores de tipado, por lo cual, son descartados. Una vez realizado la limpieza, se descartó más o menos el 4% de los datos, dado que no es tan representativa esta cantidad, se decide eliminar estas filas.

La figura 6.15 evidencia los datos agrupados por localidad, de esta manera, evidenciando puntos críticos en la distribución de las ocurrencias en un alto nivel.

Una gráfica más detallada es presentada en 6.16, donde se percibe, por barrio, las ocurrencias registradas en un mapa de calor, donde los entre más grande el área de los círculos representa una mayor cantidad de las mismas en dicho lugar.

Nótese que ambas gráficas parecen diferir en su área de la ciudad, esto es debido a que una gran parte de las localidades presentadas en 6.15 incluyen zonas rurales amplias, pero en la figura segmentada por barrios se omiten estas zonas, por dicha razón la ciudad parece diferente. De igual forma, se observa que en las localidades de Chapinero y Kennedy es donde más concentración de hurtos registrados hay; a pesar de que la segunda gráfica no parece corresponder a los datos por donde se evidencia la mayor cantidad, esto es debido a que dicha área logra cruzarse con varias localidades a la vez, entonces, es repartido por varias localidades como lo son Santa fé, Chapinero, Teusaquillo, Candelaria y los Martires.

Para la exploración, se identificó el tipo de dato que constaba cada columna. De esta información observamos, en su mayoría, variables categóricas. Esta información es importante tener en cuenta a la hora de proponer un modelo, dado que no todos los algoritmos de aprendizaje trabajan con datos categóricos sino con variables numéricas. Se tienen dos variables que podrían interpretarse como numéricas que son edad y hora; el primero es discreto y el segundo (el tiempo) es continuo. Para este caso deberá realizarse una transformación de ambas variables en la etapa final con el fin de tener el mayor número de variables categóricas posibles para entrenar.

Dado que se trata temas geográficos, se optó por un algoritmo como el de k vecinos más cercanos Cover and Hart (1967), el cual calcula una probabilidad de una clasificación por medio de los elementos más cercanos. Este algoritmo trabaja con

variables cuantitativas, mas no cualitativas, por esta razón, se procedió a utilizar una adaptación llamada k-prototype Huang (1998). Se realizará una categorización (clusters) con el fin de identificar o clasificar comportamientos en base a los datos.

6.4.5 Pruebas

Las pruebas serán separadas de dos maneras: En principio, una prueba relacionada con el número de clústers elegidos en la implementación y pruebas acerca de los requerimientos de software presentados.

6.4.5.1. Elección del número de clústers

Como se mencionaba anteriormente, existen varias pruebas que se pueden realizar para validar el número de clústers idóneo. En términos de recursos y tiempos se optó únicamente por el método del codo dado que no requería una capacidad de cómputo tan elevada.

La Figura 6.17 se presenta una gráfica que relaciona el número de clústers (eje x) con la suma cuadrada entre clústers (eje y) como dicta el método del codo. Como bien muestra la gráfica, por esta razón se escogió 5 como el número de clústers

6.4.5.2. Pruebas de los requerimientos

Según el requerimiento no funcional de la Tabla 6.8, la autenticación se provee el mecanismo de autenticación por Token de seguridad, éste tendrá su vigencia con el fin de evitar una fuga de sesión. Este componente, según la ISO 25000 hace parte de un requerimiento de calidad dentro de la compleja definición de seguridad, esta es la razón por la que se eligió dicho requerimiento. La Figura 6.18 presenta una autenticación errónea.

Además, en términos de seguridad, se hace un prueba de inyección de código sobre uno de los servicios que inserta datos directamente a la base de datos, esto con el fin de evitar este tipo de ataques cibernéticos.

El ataque de inyección de código es que busca explotar una vulnerabilidad. Y es que muchas veces hay servicios expuestos que realizan inserciones directas a la base de datos, dependiendo del motor de base de datos se puede aprovechar esto para realizar daños en los datos almacenados.

La Figura 6.19 evidencia un ataque de inyección de código donde se intenta eliminar la base de datos principal del software, gracias a la implementación y las verificaciones de seguridad, en la Figura 6.20 se puede ver que se realiza una validación para evitar este tipo de eventos malintencionados e inserta el código limpio, de tal manera, que se evita el ataque y queda el registro en base de datos como si fuese una cadena de caracteres cualquiera.

Para el requerimiento no funcional descrito en la Tabla 6.8, se tiene planteado un modelo de datos modular que permite la abstracción de los datos a ser almacenados, como muestra la Figura 6.12. A su vez el esquema lógico de la solución es de dos tablas, una para los datos geográficos, otra para los del usuario. El principal componente de mantenibilidad es cuando se busca otro escenario de procesamiento ajeno al de criminalidad, dado que en ningún lugar hay una referencia directa hacia el dato que extiende, la lógica de negocio requiere ser modificada; el único punto de cambio (principio de diseño) es la creación de una clase extienda el dato *GeoLocalization* y su mapeo correspondiente no agregará una nueva tabla a la base de datos, de esta forma ésta última no aumenta el número de tablas, sino que bajo la misma se reutiliza el mismo esquema y se le agrega más columnas. Lo anterior se logra observar en el diagrama de clases de la Figura 6.11, donde no hay relaciones hacia la clase *DeliquencyGeoLocalization*, dado que esta clase concreta no es considerada en la solución sino únicamente cuando será persistida.

Según Page-Jones (Page-Jones, 1988), el nivel de acoplamiento de la clase en particular (*DeliquencyGeoLocalization*) es categorizado como acoplamiento común, la

clase *GeoLocalization* se consideraría acoplamiento de datos, pero como esta clase no requiere modificación alguna, a la hora de validar una clase que extienda su comportamiento, es baja con respecto al sistema.

6.4.6 Instalación y configuraciones

Con el fin de desplegar el artefacto de software se realiza teniendo en cuenta requisitos mínimos, tanto de software como de hardware:

6.4.6.1. Requisitos de hardware

Para alcanzar requerimientos mínimos de ejecución se requieren tres máquinas de 2 CPU's y 8 Gb de memoria RAM. Naturalmente, la máquina que disponga del entorno Hadoop debe tener más capacidad de almacenamiento para soportar una carga pesada de datos. Capacidad de red de 100 Mb, que asegura una disponibilidad y el intercambio de información más eficiente.

6.4.6.2. Requisitos de software

Las librerías necesarias para correr el software son las siguientes:

- JDK 1.8+
- R 2.5+
- Python 3.6+
- Jupyter
- Anaconda (recomendado)

6.4.6.3. Configuraciones necesarias

Una vez cumplidos los requisitos mencionados anteriormente, se debe proceder a las siguientes configuraciones:

1. Instalar los binarios del JDK
2. En un servidor donde se expone el API, ejecutar el jar con el comando:

```
java -jar data-publisher.jar
```

La sección 6.4.2 muestra la documentación del API

Figure 6.11. Diagrama de clases de la solución. Autoría propia

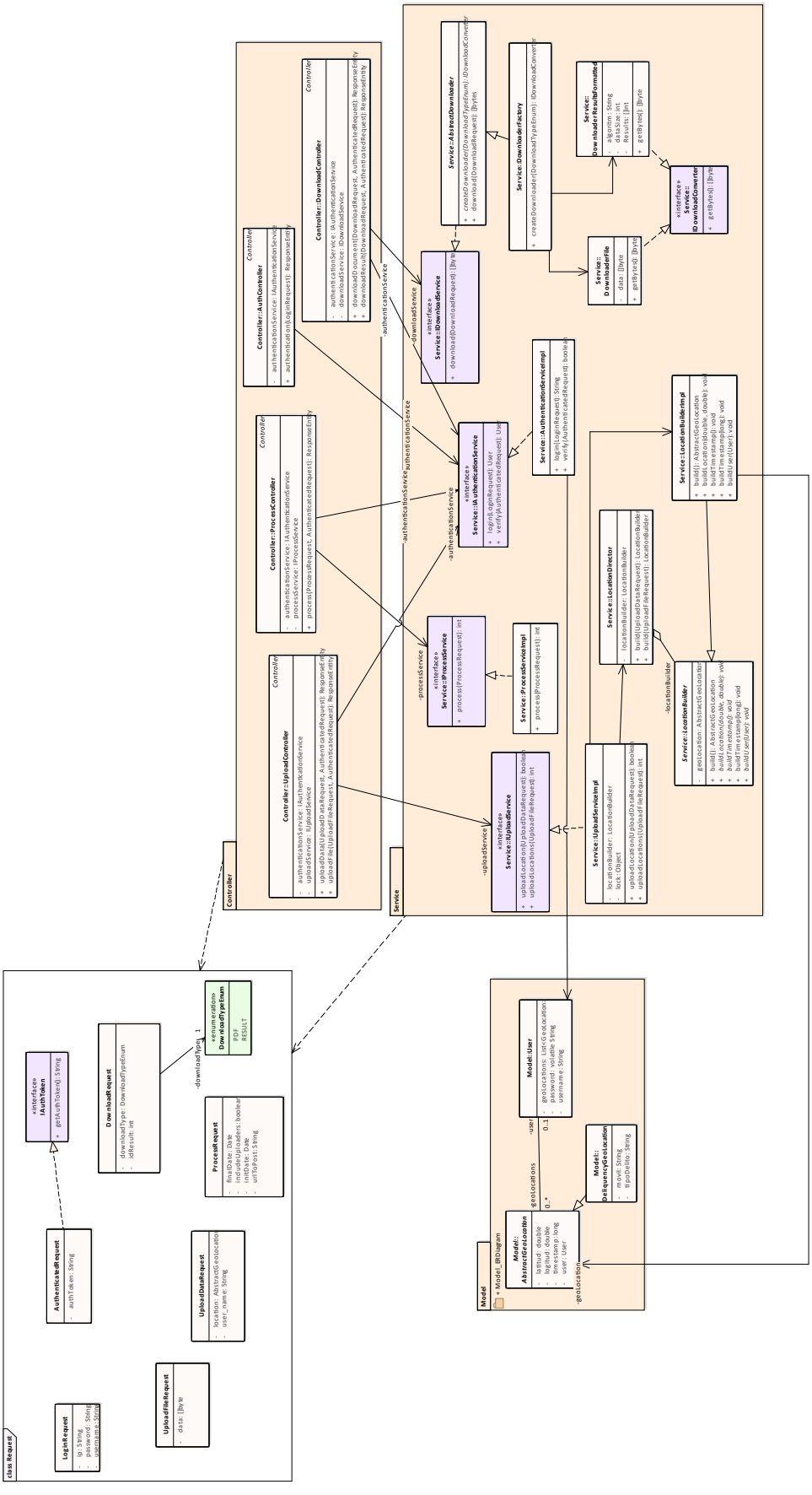


Figure 6.12. Modelo de datos extensible. Autoría propia

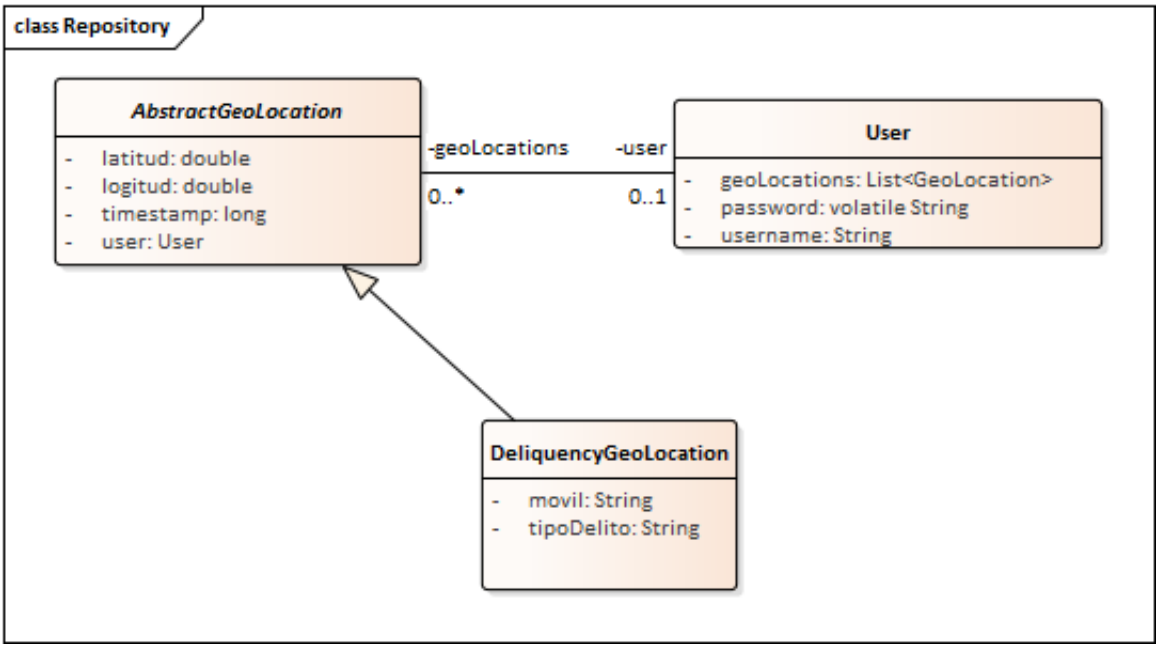


Figure 6.13. Diagrama de secuencia para el caso de uso de subida de datos. Autoría propia

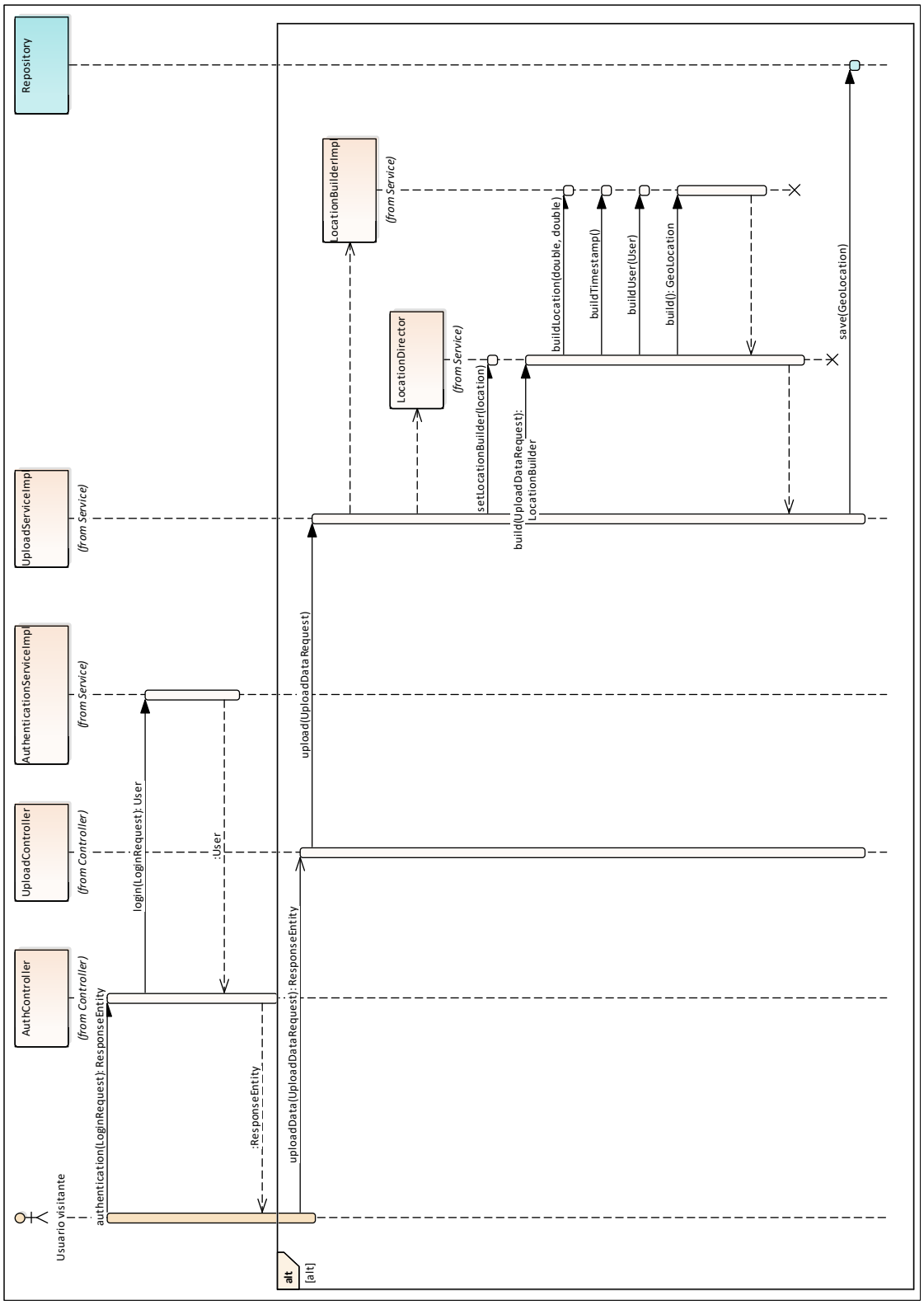


Figure 6.14. Diagrama de componentes de la solución. Autoría propia

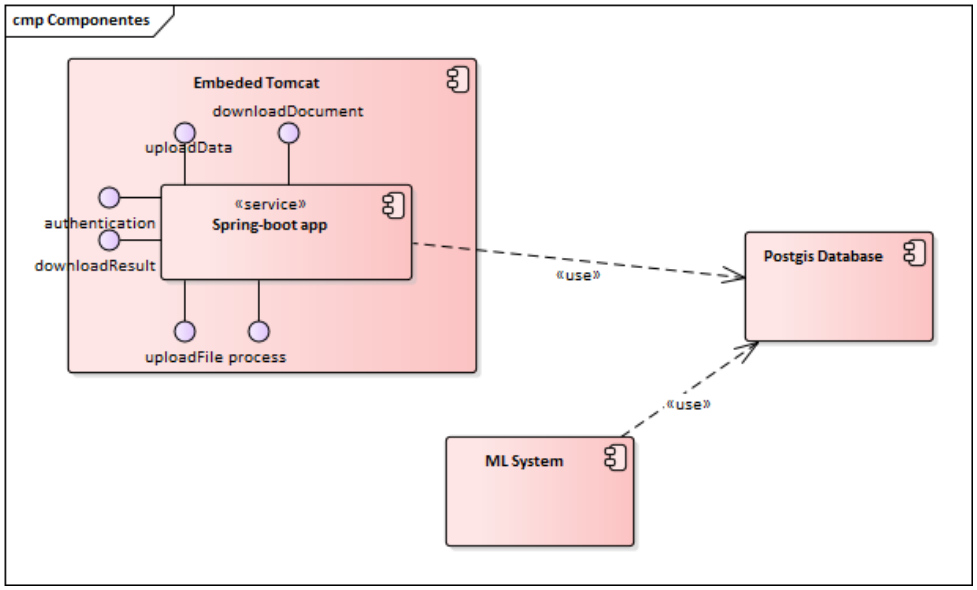


Figure 6.15. Representación tomada de los registros de la policía nacional, ciudad Bogotá por localidades. Adaptación de autoría propia

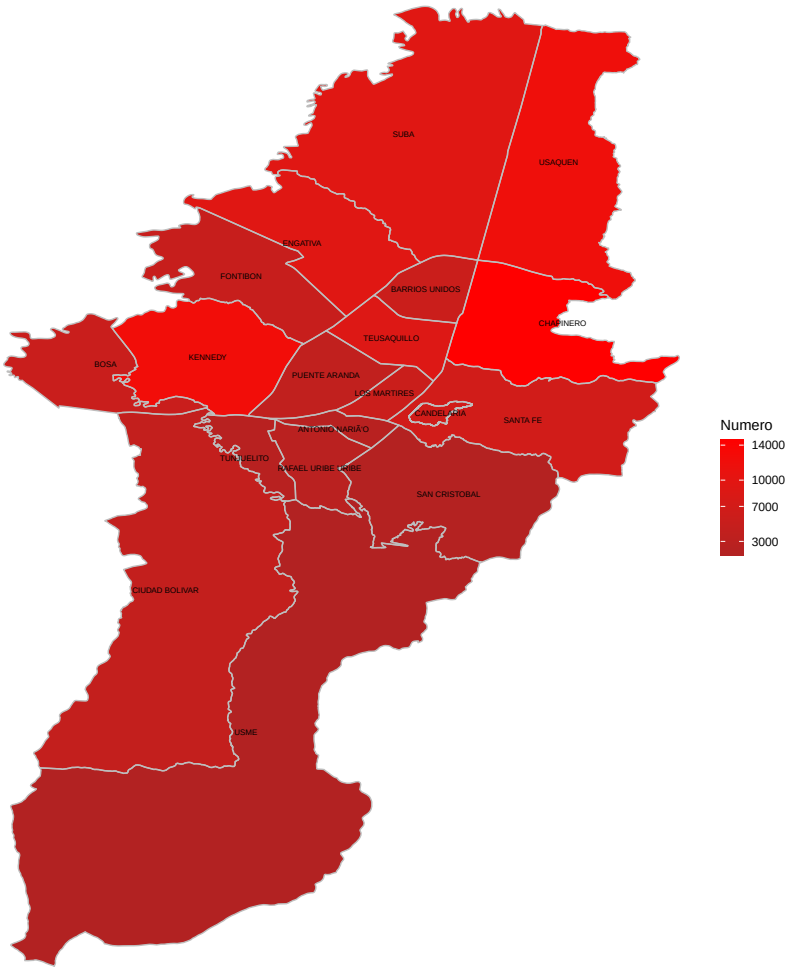


Figure 6.16. Representación tomada de los registros de la policía nacional, ciudad Bogotá por barrios. Adaptación de autoría propia

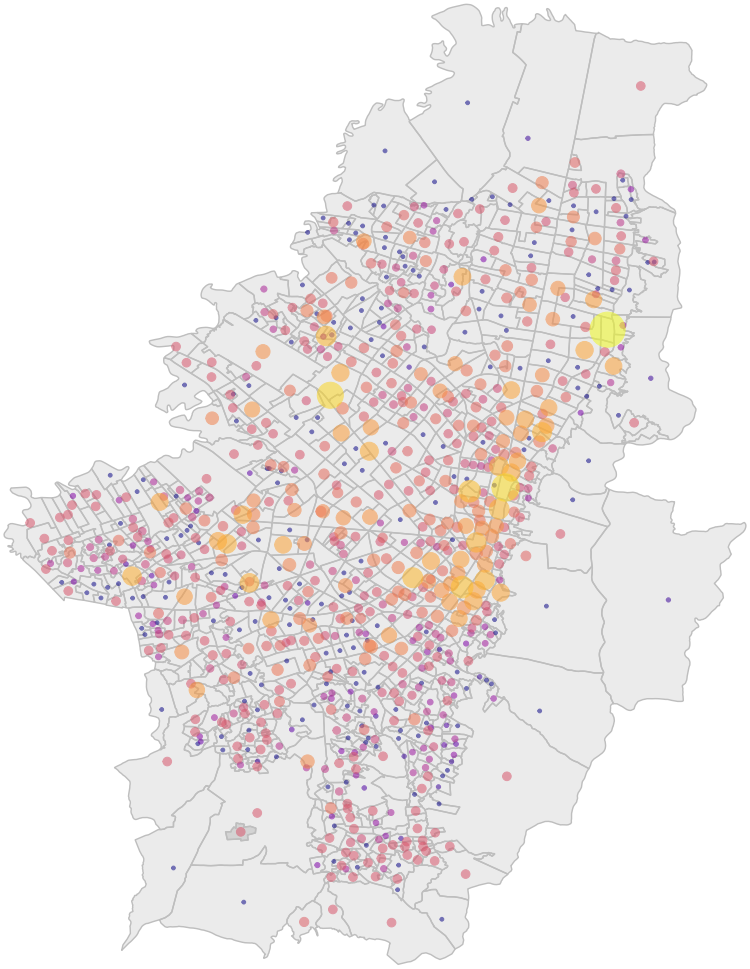


Figure 6.17. Método del codo. Autoría propia

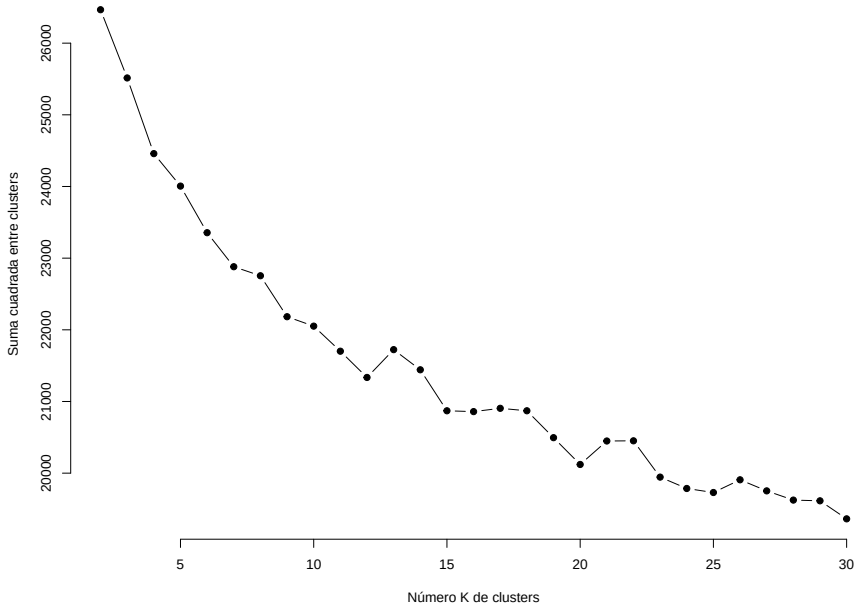


Figure 6.18. Captura de pantalla error de autorización. Autoría propia

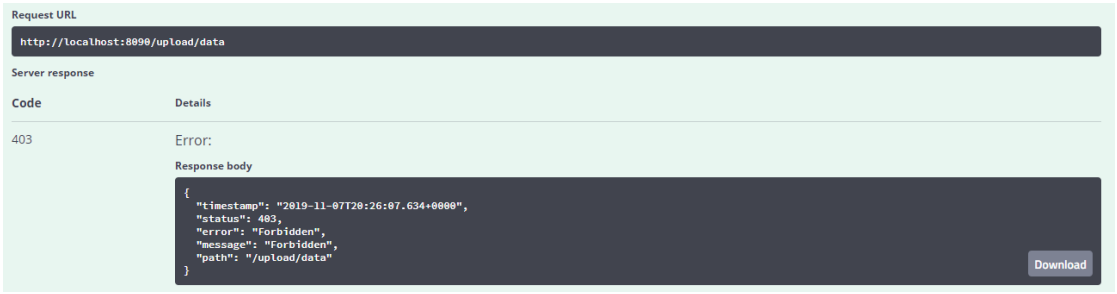
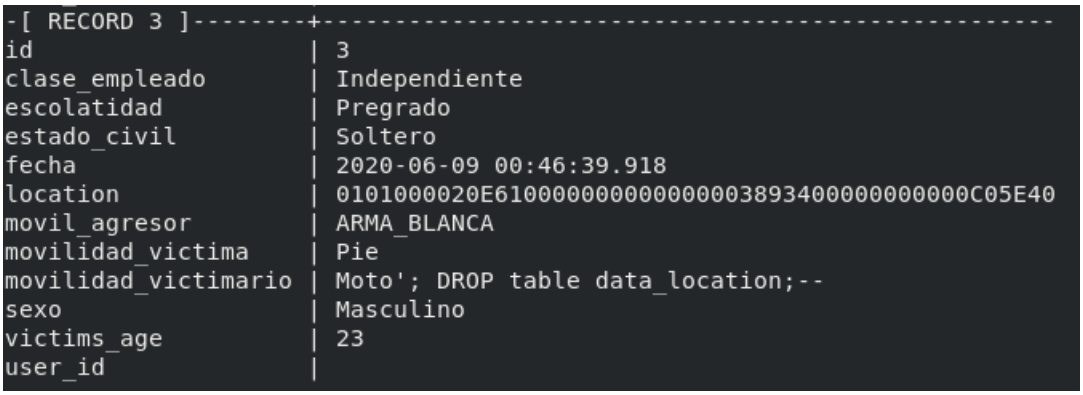


Figure 6.19. Captura de pantalla de ataque de inyección de código. Autoría propia

Figure 6.20. Captura de pantalla de consulta en la base de datos. Autoría propia



CAPÍTULO 7. CONCLUSIONES Y FUTURO TRABAJO

Para terminar, este proyecto deja una una solución de software a la disposición de la ciudadanía de la cual puedan interactuar, por medio del registro de datos y la visualización de resultados espaciales, y esta información puede ser almacenada para posteriormente ser procesada. Además, se evidencia un proceso de aprendizaje sobre los datos, por medio de un modelos pre-existente, que se ajusta a la estructura de los datos. Y finalmente, se ha obtenido una visualización de los resultados del aprendizaje que permite caracterizar los comportamientos clasificados por los modelos.

Del mismo modo, el desarrollo de este software deja abierta la opción de una solución móvil o desktop con el cual sea posible brindarle información a un usuario, para procurar identificar y disminuir las ocurrencias de dichos actos delictivos. También, gracias a la información que puede ser recuperada de la implementación se puede extender en variables, como en modelos para aplicar. Lo anterior, con tal de evolucionar la aplicación y sobrellevar una de las mayores complicaciones del proyecto, que fue los datos proporcionados, tanto en tamaño, como en variables que disponía.

APPENDICES

REFERENCIAS

REFERENCIAS

Bass, L., Clements, P., & Kazman, R. (2003). *Software architecture in practice*. Addison-Wesley Professional.

Beck, K. (1999). Extreme programming explained: embrace change.

Cover, T., & Hart, P. (1967). Nearest neighbor pattern classification. *IEEE transactions on information theory*, *13*(1), 21–27.

Creswell, J. W. (1994). *Research design: Qualitative & quantitative approaches*. Sage Publications, Inc.

Fowler, M. (2002). *Patterns of enterprise application architecture*. Addison-Wesley Longman Publishing Co., Inc.

Huang, Z. (1998). Extensions to the k-means algorithm for clustering large data sets with categorical values. *Data mining and knowledge discovery*, *2*(3), 283–304.

Inseguridad en bogotá: ¿aumentó el delito o cambió la forma de medirlo? (n.d.). `https://www.semana.com/nacion/articulo/cifras-del-delito-en-bogota-2018/564737`. (Accessed: 2019-05-10)

ISO, I. E. C. (2008). ISO/IEC 25020: Software product Quality Requeriments and Evaluations(SQuaRE) - Measurement reference model and guide. *International Organization for Standardization*.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An introduction to statistical learning* (Vol. 112). Springer.

Kodinariya, T. M., & Makwana, P. R. (2013). Review on determining number of cluster in k-means clustering. *International Journal*, *1*(6), 90–95.

Mason, H. (2010). *Dataists: A taxonomy of data science*. Retrieved 2019-05-10, from `http://www.dataists.com/tag/osemn/`

Page-Jones, M. (1988). *The practical guide structured systems design*. Prentice-Hall,.

Sommerville, I. (2005). *Ingeniería del software*. Pearson Educación.