UNIVERSIDAD AUTÓNOMA DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería Informática y Matemáticas

TRABAJO FIN DE GRADO

Detección de patrones y trayectorias a partir de información temporal y geoespacial poco precisa

Autor: Carlos Li Hu
Tutor: Álvaro Ortigosa Juárez

junio 2019

Carlos Li Hu
Detección de patrones y trayectorias a partir de información temporal y geoespacial poco precisa

Carlos Li Hu
C\ Francisco Tomás y Valiente Nº 11

IMPRESO EN ESPAÑA – PRINTED IN SPAIN

Resumen

Las fuerzas y cuerpos de seguridad muchas veces se enfrentan a la tarea de encontrar sospechosos para un crimen ya cometido, o, en el caso de una desaparición, de encontrar a la víctima. Para resolver estos crímenes, muchas veces sólo disponen de una ubicación aproximada del crimen realizado o del teléfono de la víctima. Por tanto, accediendo a un registro de llamadas y antenas telefónicas, debería facilitarse el proceso de resolución del crimen.

Es en este contexto donde se desarrolla este trabajo, pues consiste en diseñar una aplicación que provea de información a un analista para poder identificar a los posibles sospechosos de un crimen conociendo la ubicación aproximada y el intervalo temporal de realización de dicho crimen. Además, también permite seguir la ubicación de determinados teléfonos y las trayectorias que siguen de cara a poder establecer sus paraderos y, en el caso de ser los sospechosos, de identificar a sus posibles cómplices en el crimen.

Para ello, se ha desarrollado un módulo de inserción en base de datos de ficheros con un formato configurable. También se ha desarrollado una interfaz web donde se pueden realizar, mediante formularios, las consultas en bases de datos. Los resultados se muestran mediante un mapa interactivo en Google Maps, donde se pueden visualizar los datos en detalle al clicar en las distintas ubicaciones del mapa.

Palabras clave

aplicación, base de datos, Google Maps, llamadas telefónicas, crimen, sospechosos

Abstract

The security forces and bodies usually deal with the task of finding crime suspects of a commited crime, or in other cases, to find the victim of a disappearance. To solve these crimes, they usually just start with an approximate location of the crime scene, or with the victim's phone number. Thus, having access to a phone call and antenna registry, it should be easier to solve these crimes.

This project is developed in this context, as it consists of designing an app that provides an analyst with information to identify possible suspects of a crime given the approximate location and the time interval in which that crime was committed. In addition, it allows the user to see the locations and path that a given phone number follows to establish its whereabouts and, in the case of the suspects, to identify possible accomplices to the crime.

To tackle this issue, a data insertion module has been developed to insert data in a database with a configurable format. Also a web interface has been developed. In this interface, through forms, queries in the database can be done. The results of the query are shown through a Google Maps interactive map, where data can be visualized in detail when clicking on the different map locations.

Keywords

app, database, Google Maps, phone calls, crime, suspects

Índice

Listas

Lista de figuras

Lista de tablas

1

Introducción

1.1. Motivación

A diario, se cometen crímenes en ubicaciones recónditas y pasan desapercibidos. Una vez que se descubren tales crímenes, ya es muy tarde para ubicar a los sospechosos, o en el caso de una desaparición o secuestro, a las víctimas de tales crímenes.

Para resolver dichos crímenes, las fuerzas y cuerpos de seguridad muchas veces disponen únicamente de una ubicación aproximada en donde se ha cometido el crimen para identificar a los posibles sospechosos. En otros casos, en los que interesa más localizar a una víctima, sólo disponen de su teléfono para comenzar la investigación de su paradero.

Por tanto, a petición de la Guardia Civil, se decide implementar una herramienta con los requisitos establecidos por ellos. La funcionalidad de esta herramienta consiste en, a partir de los datos provistos por distintas compañías de operadoras telefónicas, ser capaz de proveer a un analista de información y un cierto grado de reconocimiento automático para poder resolver estos casos.

1.2. Objetivos

A continuación listaremos los objetivos que debe cumplir la aplicación para ser satisfactoria:

O-1.– Permitir definir los formatos de los ficheros de datos a insertar en la Base de Datos.

O-2.– Permitir insertar los datos en la Base de Datos usando dichos formatos.

O-3.– Permitir conocer el comportamiento de todos los terminales en un área determinada y los instantes en que estuvieron allí. Además de permitir excluir de entre esos terminales los que sean menos relevantes.

O-4.– Permitir conocer las trayectorias de un terminal objetivo y los instantes en que estuvo allí. Además de permitir identificar posibles terminales que estuvieran con el objetivo en esos instantes.

O-5.– La aplicación debe tener una interfaz gráfica sobre la que se pueda realizar filtros de búsqueda y mostrar por pantalla los resultados de esos filtros.

1.3. Organización de la memoria

Este documento está organizado en 6 capítulos (siendo el primero esta introducción):

- El capítulo 2 contiene el *Estado del Arte*, donde se detallarán las herramientas de desarrollo que existen actualmente para implementar la aplicación, destacando las más útiles y analizando sus características.
- En el capítulo 3 de *Análisis y Diseño* se listarán los requisitos funcionales y no funcionales que tiene que cumplir la aplicación. También se decidirán razonadamente las herramientas (detalladas en el capítulo anterior) que se usarán en la implementación junto a los esquemas de la estructura que seguirá esta aplicación. Adicionalmente se comentarán las mayores dificultades encontradas durante el diseño.
- En el capítulo 4 de *Desarrollo*, se detallan las versiones de las librerías utilizadas en el momento, y se destacan los aspectos más interesantes que surgen de pasar del diseño a la implementación en código.
- En el capítulo 5 de *Integración, pruebas y resultados*, se mostrará la interfaz web de la aplicación y se detallará un ejemplo de uso en un posible caso realista e interesante.
- En el capítulo 6 de *Conclusiones y trabajo futuro*, se concluye el resultado de este proyecto, y se plantearán posibles mejoras que no ha habido tiempo para implementar.

2

Estado del Arte

En la actualidad, existen numerosas herramientas de desarrollo que facilitan la implementación de esta aplicación. A continuación se describirán los componentes esenciales que son relevantes para el diseño y sus variantes más útiles.

2.1. Framework

A la hora de considerar el desarrollo de esta aplicación, a petición del cliente, es necesario que la aplicación esté desarrollada en Python, y que posea una interfaz web. Por tanto, se ha decidido usar un framework, pues de esta manera se facilita la creación de la app, ya que se le dota de un esqueleto estandarizado en el que la funcionalidad está bien modularizada.

Los dos frameworks más populares del momento para aplicaciones web desarrolladas en Python son Django [1] y Flask [2].

Ambos proporcionan una estructura a la aplicación, siendo la de Django más restrictiva y estandarizada. A pesar de ello, Django proporciona muchas herramientas de desarrollo integradas.

Mientras tanto, Flask se caracteriza por su flexibilidad y simpleza. Proporciona menos herramientas de desarrollo, pero se puede solucionar fácilmente mediante la inclusión de plugins.

2.2. API de mapas

Existen muchas APIs disponibles para implementar mapas en Javascript, la más usada y completa en el mercado actualmente es la de Google Maps [3]. Al ser tan utilizada, se ha convertido prácticamente en el estándar para mapas en la web. Y por tanto, contiene muchas funcionalidades, mapas muy actualizados, una documentación muy cuidada y un buen soporte por parte de los desarrolladores de la API.

El problema principal con esta API es que fue gratuita hasta el 11 de junio de 2018, pero desde

entonces se ha vuelto de pago, y esto conlleva un gasto mensual dependiendo del número de transacciones con ella. Actualmente la API permite su uso con un coste máximo de 200$ mensuales de forma gratuita. Además, tiene la desventaja de que no permite su uso offline, por lo que una conexión a Internet es necesaria.

Por tanto, se han investigado las alternativas opensource y gratuitas existentes. Las dos APIs principales en este sector son OpenLayers [4] y Leaflet [5].

OpenLayers es la alternativa más clásica de implementación de mapas opensource. Posee una gran cantidad de funcionalidades integradas como, por ejemplo, visualización de mapas 3D y permite mostrar rápidamente grandes conjuntos de datos vectoriales.

Leaflet es una librería Javascript más moderna que OpenLayers. Por tanto, posee una mejor arquitectura y diseño interno. La idea tras Leaflet es la de una librería más simple y sencilla de utilizar. Por tanto, posee una documentación de la API más cuidada y actualizada. No tiene tantas funcionalidades integradas como OpenLayers, pero lo suple con la gran variedad de plugins que se le pueden añadir.

A diferencia de Google Maps, estas alternativas, no poseen sus propios mapas, por tanto, tienen que hacer uso de un mapa de terceros. En este caso, el proveedor de mapas gratuitos por excelencia es OpenStreetMap [6]. Además, la funcionalidad que incluyen es más limitada, por ejemplo, Google Maps es la única que permite consultar carreteras cercanas a ubicaciones y trazar sus trayectorias fácilmente (Véase Snap to Roads [7]).

2.3. Bases de Datos

Se pueden enumerar numerosos tipos de Bases de Datos (BBDD) dependiendo de la estructura que utilizan para almacenar sus datos. Los dos principales a considerar para esta aplicación son las Bases de Datos Relacionales (BDR) y las Bases de Datos Orientadas a Grafos (BDOG) .

Las BDR siguen el modelo relacional, que consiste en guardar los datos en tablas y relaciones. Mientras que las BDOG guardan los datos en nodos (también llamados vértices) relacionados con otros nodos mediante aristas (también llamadas relaciones). Las BDOG llevan mucho tiempo inventadas, pero fueron eclipsadas en su momento por las BDR , las cuáles son más sencillas y por tanto más populares. Por ese mismo motivo, actualmente existen escasas herramientas de BDOG .

De entre todas las alternativas para ambos tipos, en este apartado se analizarán las dos más populares en cada campo. En este caso resultan ser PostgreSQL [8] como BDR y Neo4J [9] como BDOG .

PostgreSQL es una BDR que fue creada en 1989 e implementada en C. Entre sus características más útiles se destaca su alta concurrencia, es decir, permite que mientras un proceso modifica una tabla, otros accedan a la misma sin necesidad de bloqueos, y así permitiendo a cada usuario obtener

una visión consistente. Además permite la ejecución de funciones, procedimientos almacenados y triggers (también llamados disparadores). Por estas razones, y por su longevidad en el mercado, es una herramienta más clásica, con muchas funcionalidades y que tiene muchos más años de soporte, lo que la hace más robusta y versátil.

Por otra parte, Neo4J es una BDOG que fue creada en 2007 usando Java y se basa en el uso de "grafos de propiedad". Estos son un tipo de grafos dirigidos, con peso en las relaciones entre nodos, y que poseen etiquetas y propiedades en los nodos y relaciones entre ellos. Neo4J tiene mejor rendimiento que la mayoría de BBDD relacionales, pues aunque las consultas de datos aumenten exponencialmente, el rendimiento de Neo4j no desciende (frente a lo que sí sucede con las BDR). Esto es debido a que las BDOG responden a las consultas actualizando los nodos y las relaciones de esa búsqueda y no las de todo el grafo completo y eso optimiza mucho sus consultas. Además, cabe destacar su flexibilidad y escalabilidad, pues cuando aumentan las necesidades, es muy sencillo añadir más nodos y relaciones a un grafo ya existente. En general, por su naturaleza estas BBDD destacan en eficiencia si lo que se busca es optimizar las búsquedas basadas en relaciones entre nodos.

En general, ambas BBDD permiten realizar consultas relacionadas con objetos geográficos. PostgreSQL lo puede hacer mediante una extensión espacial llamada PostGIS [10]. Esta herramienta permite crear objetos geográficos, y a su vez realizar consultas SQL sobre estos objetos. En este trabajo, lo interesante es la funcionalidad de distancia entre coordenadas geográficas. Por otra parte, Neo4J tiene ya integrada esta funcionalidad geoespacial, permitiéndole realizar la misma funcionalidad, pero de forma más natural y optimizada, pues la BD fue creada teniendo estas funcionalidades en mente.

2.4. Datos

Para diseñar la aplicación se necesitan dos fuentes de datos principales.

Primeramente, se requieren los datos de las antenas o repetidores existentes y sus ubicaciones. Actualmente la mayor fuente de información de antenas opensource es la que se encuentra en OpenCellid [11]. Esta BD es mantenida por una comunidad de forma voluntaria, y como tal, no contiene todas las antenas existentes.

Por otra parte se necesitan datos sobre las llamadas realizadas por números de teléfono, la antena a la que se conectaron y el momento en que se realizó la llamada. Por supuesto, estos datos son privados y sólo se pueden conseguir contactando directamente con las distintas operadoras. Además existe una complicación añadida, pues cada operadora tiene sus respectivos datos y que pueden no ser los mismos o ni siquiera compartir los mismos formatos que los que poseen otras compañías.

3

Análisis y Diseño

3.1. Análisis de Requisitos

A continuación se detallarán los requisitos funcionales y no funcionales que tiene que cumplir la aplicación según lo especificado por el cliente, la Guardia Civil.

Requisitos funcionales

RF-1.– Permitir definir el formato de los datos mediante un fichero JSON.

RF-2.– Permitir insertar los datos desde un fichero excel con el formato definido en el fichero JSON.

RF-3.– Permitir filtrar los terminales en una ubicación del mapa y los momentos en los que han estado allí, dados:

- **RF-3.1.–** Sus coordenadas en latitud y longitud.
- **RF-3.2.–** Un radio de búsqueda.
- **RF-3.3.–** Un intervalo de tiempo entre 2 fechas.

RF-4.– Permitir filtrar las ubicaciones, los instantes de tiempo y la trayectoria seguida por un teléfono objetivo dados:

- **RF-4.1.–** El número de teléfono objetivo.
- **RF-4.2.–** Un intervalo de tiempo entre 2 fechas.

RF-5.– Tener una interfaz gráfica compuesta por:

- **RF-5.1.–** Un mapa interactivo para visualizar los resultados
- **RF-5.2.–** Un formulario web para realizar los distintos filtros de búsqueda.

Requisitos no funcionales

RNF-1.– La lógica de la aplicación debe estar programada en Python 3.

RNF-2.– Se mostrará en una interfaz web programada en Javascript.

RNF-3.– La aplicación está aislada de Internet.

- **RNF-3.1.–** La aplicación funciona correctamente en local.
- **RNF-3.2.–** Pero permite realizar consultas en Internet, siempre y cuando estas no impliquen el envío de datos policiales sensibles.

3.2. Diseño

Tras la evaluación de las diferentes herramientas existentes en el capítulo 2, aquí se asignan las que se usarán finalmente junto al criterio que se ha seguido a la hora de decidirlo. Además se detallará un esquema de la estructura que seguirán los distintos componentes.

3.2.1. Framework

La envergadura del proyecto no es muy grande y con respecto a las herramientas necesarias, ambas proporcionan las mismas funcionalidades. Por tanto, el framework que se utilizará será Flask, pues su arquitectura es más sencilla que la de Django, por lo que en este caso primaremos la simpleza, para que el proyecto no sea innecesariamente complicado de desarrollar.

El esquema que sigue es la estándar de una arquitectura MVC en Flask. Por tanto, en el capítulo 4 se realizará un análisis más detallado del esquema.

3.2.2. API de mapas e Interfaz Web

Finalmente se ha decidido usar la API de Google Maps [3] a pesar de que su uso conlleve un gasto mensual. Esto es porque, tras probar las diferentes alternativas, se ha comprobado que esta API proporciona muchas más funcionalidades interesantes que las demás no tienen. Como por ejemplo, un módulo de Roads para detectar carreteras cercanas. Además su documentación y mapas son regularmente actualizados, y por tanto, todo esto la hace idónea de cara a su escalabilidad y mantenimiento futuros.

Este módulo forma parte de la interfaz web de la aplicación, pues se usa la API mediante Javascript. Su utilidad consiste principalmente en representar gráficamente los resultados de las consultas realizadas mediante un formulario web.

Principalmente, se tendrán dos formularios distintos:

- En el primero, dados un intervalo de tiempo entre una fecha inicial y una fecha final, y dado un teléfono objetivo, representará en el mapa todas las ubicaciones donde se ha realizado una llamada que involucre a ese teléfono.
- En el segundo, dado un intervalo de tiempo entre una fecha inicial y una fecha final, una ubicación, y un rango de búsqueda, mostrará todos las llamadas de teléfono que se han realizado en ese área.

3.2.3. Base de Datos

Ambas BBDD descritas en el capítulo anterior proporcionan la misma funcionalidad de consultas geoespaciales. Por tanto, la BD que se usará finalmente es PostgreSQL (BDR) con la extesión PostGIS para consultas geoespaciales. Esto es debido a que para esta aplicación, se ha observado que

no existe mucha complejidad a la hora de almacenar los datos y por tanto, no es necesario recurrir a BDOG , las cuáles tienen una estructura más complicada.

Los formatos de los datos de antenas se han obtenido basándose en los formatos de datos encontrados en OpenCellid [11]. Mientras que para crear los formatos de los ficheros de llamadas, se han tenido de referencia datos reales proporcionados por distintas compañías telefónicas privadas. Finalmente se ha definido un modelo básico que contiene todos los campos comunes y necesarios para la aplicación como vemos en el diagrama Entidad-Relación de 3.1.

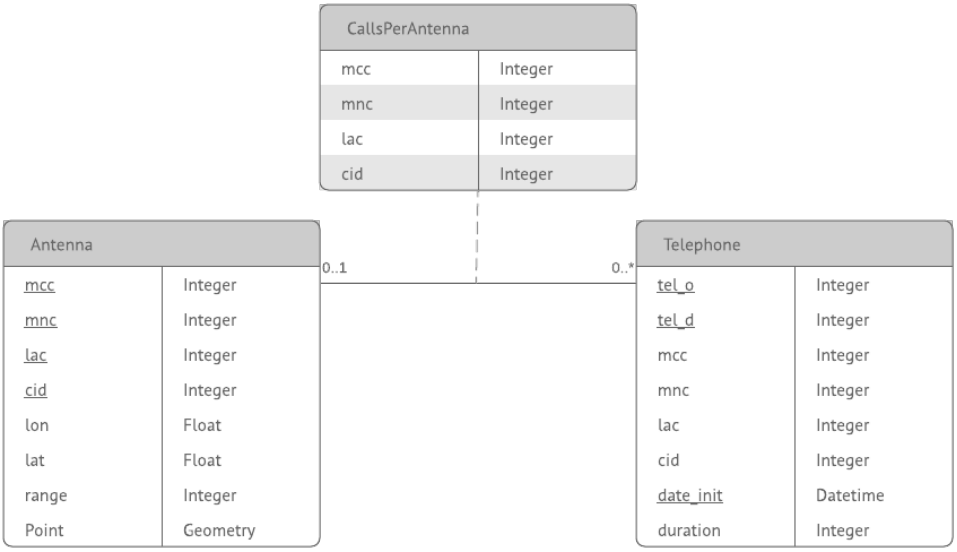


Figura 3.1: Diagrama Entidad-Relación

Como se ve en la imagen, la entidad *Antenna* se identifica con la tupla (mcc, mnc, lac, cid) que forman un identificador único para la antena. Además, da información sobre su ubicación con su longitud y latitud y su rango de alcance. Para realizar las consultas geoespaciales, se añade también un objeto Point que es de tipo Geometry de la librería PostGIS, y se crea usando la longitud y latitud.

La entidad *Telephone* que representa las llamadas telefónicas, se compone de un teléfono de origen, un teléfono destino, una fecha de inicio de llamada y su duración en segundos. Como se ve, se relaciona con la entidad Antenna, en tanto que una llamada se realiza conectándose a una antena concreta, por lo que se tiene de clave foránea la tupla de identificación de la antena.

3.2.4. Datos

Debido a la sensibilidad y privacidad de los datos que se manejan, ha sido imposible conseguir datos reales de llamadas telefónicas para probar la aplicación. Por tanto, la Guardia Civil ha proporcionado datos simulados (mediante un algoritmo secreto) de antenas y llamadas telefónicas de forma que se garantice la consistencia de estos, es decir, las antenas, sus ubicaciones, y los teléfonos que realizan llamadas en esas antenas son falsos, pero tienen sentido entre sí. Véase la sección 3.3 para más información.

Debido a que los datos tienen formatos completamente distintos dependiendo de la compañía que las provea, para insertar los datos en la BD, se ha de programar un módulo en Python.

Los datos proporcionados vendrán principalmente en 3 tipos de ficheros distintos:

- Un fichero JSON donde se definen los formatos de los otros 2 ficheros, es decir, se detallarán los nombres de las columnas correspondientes para cada modelo de datos.
- Un fichero excel que contiene los datos de las antenas telefónicas.
- Un fichero excel que contiene los datos de las llamadas realizadas.

En el capítulo 4, se realizará un análisis más detallado del formato de estos ficheros y del algoritmo de inserción de datos.

3.3. Dificultades Encontradas

A continuación se detallarán las mayores dificultades que han surgido al tratar de diseñar e implementar el código. Se incluye en esta sección, pues esencialmente son complicaciones surgidas del diseño.

3.3.1. API de mapas

Durante la implementación del módulo de mapas, se optó en un principio por tratar de usar librerías y mapas opensource. Más concretamente, se usó la librería de Leaflet junto a los mapas proporcionados por OpenStreetMap.

El problema surgió cuando se comprobó que los mapas estaban poco actualizados y contenían pocos datos útiles para consultas, entre ellos, datos de carreteras cercanas. Además, esta librería tenía funcionalidades mucho más limitadas que las de Google Maps.

Por ese motivo, y para mejorar la escalabilidad en caso de querer ampliar la aplicación en un futuro, se pasó a la API de Google Maps como se detalló previamente. Aunque se mencionó en el capítulo 2 que esta API es de pago tras un número de transacciones y que no permite su uso offline, también es

la más completa y usada del mercado y se comprobó que el uso que se le daba no llegaba a superar el umbral de transacciones, por lo que a este nivel, se puede usar sin gastos económicos.

3.3.2. Simulado de datos

Para empezar, como se mencionó en el capítulo 2, los datos que poseen las distintas compañías sobre antenas y teléfonos tienen formatos completamente distintos, siendo muy difícil unificar la lectura de ficheros en las mismas tablas de la BD. Además, debido a la sensibilidad de los datos que se manejan, las compañías no han podido proveer datos reales para utilizar la aplicación. Por ese motivo, sin una fuente de datos reales de la que partir, ha resultado inviable simular casos realistas para comprobar que la aplicación funciona correctamente.

Por tanto, al final se han pedido datos simulados al personal de la Guardia Civil. Estos datos contienen unos formatos muy básicos y son de un muestreo bastante reducido, del orden de sólo 5 ubicaciones distintas para las antenas y sólo 50 llamadas realizadas entre esas 5 ubicaciones.

Pero a pesar de que ya se tenga un muestreo del que partir, se desconoce el algoritmo que han utilizado para simular estos datos (pues en su momento, consideraron que desvelarlo podría desembocar en que se pudieran deducir los datos originales). Por este motivo, ampliar el muestreo por nuestra parte podría contaminarlo y volverlo menos realista. Por tanto, se ha decidido utilizar los datos proporcionados por la Guardia Civil sin alterarlos.

4

Desarrollo

4.1. Versiones de Librerías

Al desarrollar esta aplicación, se ha optado por usar las versiones más recientes de cada librería en el momento. A continuación se muestra una tabla con esa información:

Librería	Versión
Python	3.7.2
PostgreSQL	11.2
Flask	1.0.2
Jinja2	2.10
psycopg2	2.7.7
SQLAlchemy	1.3.0
Flask-SQLAlchemy	2.3.2
PostGIS	2.5
GeoAlchemy2	0.6.2
Pandas	0.24.2

Tabla 4.1: Esta es una tabla donde detallamos las versiones de las librerías usadas.

4.2. Implementación

En el capítulo 3 se eligieron las herramientas que se usarán, y un esquema del diseño que se seguirá. A continuación se destacarán las características más interesantes que surgen de traducir del diseño al desarrollo.

4.2.1. Flask

Para implementar toda la arquitectura de la aplicación en Flask se ha seguido el tutorial provisto por la documentación oficial de Flask [2] y se ha modulado como dicta el estándar de diseño. Esencialmente sigue una arquitectura de diseño MVC tal y como se ve en la figura 4.1.

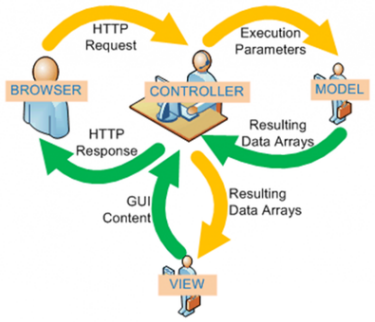


Figura 4.1: Diagrama Modelo-Vista-Controlador

El esquema de los ficheros que lo componen ha quedado así:

```
/
├── static
│   └── css
│       └── style.css
├── templates
│   └── base.html
├── app.py
├── forms.py
├── models.py
├── manage.py
└── populate.py
```

En el directorio principal se encuentra una carpeta *static*, donde se guarda el css de la página, y una carpeta *templates*, donde se encuentra *base.html* donde se tiene toda la interfaz gráfica de la aplicación con Javascript. Estos componentes conforman la vista.

En el directorio principal también se tiene el resto de los módulos en Python. Entre ellos se encuentra el *models.py*, que conforma el modelo del sistema, pues almacena los modelos de la base de datos.

Por la parte de controlador, se tiene el *forms.py*, que hace de intermediario entre las peticiones del usuario y la base de datos, el *manage.py* para realizar migraciones de base de datos en caso de que sea necesario, y finalmente el *app.py*, el programa principal que se ejecuta para activar el servidor.

Aparte del modelo MVC, hay un fichero *populate.py*, un script que se encarga de la inserción de datos en la BD, tal y como se explica en la subsección 4.2.4.

4.2.2. Google Maps API e Interfaz Web

Toda la interfaz web se puede encontrar principalmente en el fichero *base.html* y con el estilo definido en el fichero *style.css*. El código consiste esencialmente en código HTML y Javascript, por tanto no hay mucho código que destacar. Lo único menos convencional es el uso de la librería Javascript de la API de Google Maps, pero sigue los estándares de implementación de la documentación oficial en [3], por lo que no se profundizará más en ello.

El aspecto y la funcionalidad de la interfaz web se examinarán detalladamente en el capítulo 5.

4.2.3. Base de Datos PostgreSQL

Para implementar este módulo se ha decidido usar la librería SQLAlchemy [12] de Python. De esta manera se puede abstraer el lenguaje SQL utilizado y realizar todas las operaciones con su ORM. Para poder usar la funcionalidad geoespacial asociada a la extensión PostGIS, SQLAlchemy cuenta con su propia extensión de herramienta llamada GeoAlchemy 2 [13]. Esta herramienta, permite abstraer el uso de PostGIS al igual que SQLAlchemy lo hace con PostgreSQL y extendiendo de forma natural a SQLAlchemy.

El modelo de datos, que se encuentra en el fichero models.py, es esencialmente el descrito en la sección 3.2 de diseño, en la figura 3.1.

4.2.4. Inserción de datos en BD

La inserción de datos se hace con el código que se encuentra en el fichero *populate.py*. A continuación se describirá el algoritmo, pues la metodología no es trivial, y requiere de explicaciones.

Primero se leen los formatos de los datos a insertar usando el mapeo definido en un fichero JSON. El mapeo ha sido diseñado tal y como aparece en la figura 4.2.

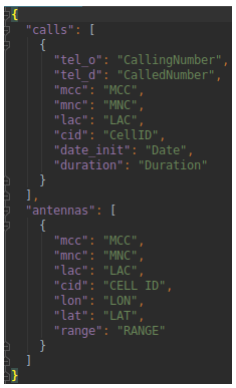


Figura 4.2: Contenido del fichero models.json

Esencialmente hay dos listas de objetos, una para los distintos modelos de fichero para la tabla de llamadas y otra para la de la tabla de antenas. Cada objeto está formado por pares clave-valor, en este caso la clave se refiere a la columna de la tabla en el modelo en PostgreSQL, y el valor es el nombre que tiene asociada la columna correspondiente en el fichero de datos a insertar.

A continuación se leen los datos de los ficheros excel con la librería de Pandas [14] y usando el mapeo designado en el fichero JSON.

Cabe destacar, que antes de insertar los datos se hace una comprobación por si hay claves primarias duplicadas. En tal caso, se imprimen por pantalla las filas conflictivas y se insertan el resto de datos. Sabiendo que los datos a insertar no tienen claves duplicadas, se realiza un bulk insert pues es más rápido y eficiente que la inserción fila a fila.

5

Integración, pruebas y resultados

En este apartado se procede a mostrar la interfaz gráfica de la aplicación web y a enseñar su funcionalidad principal mediante un ejemplo de uso en el que se usan los datos simulados de los que se disponen.

5.1. Interfaz Web

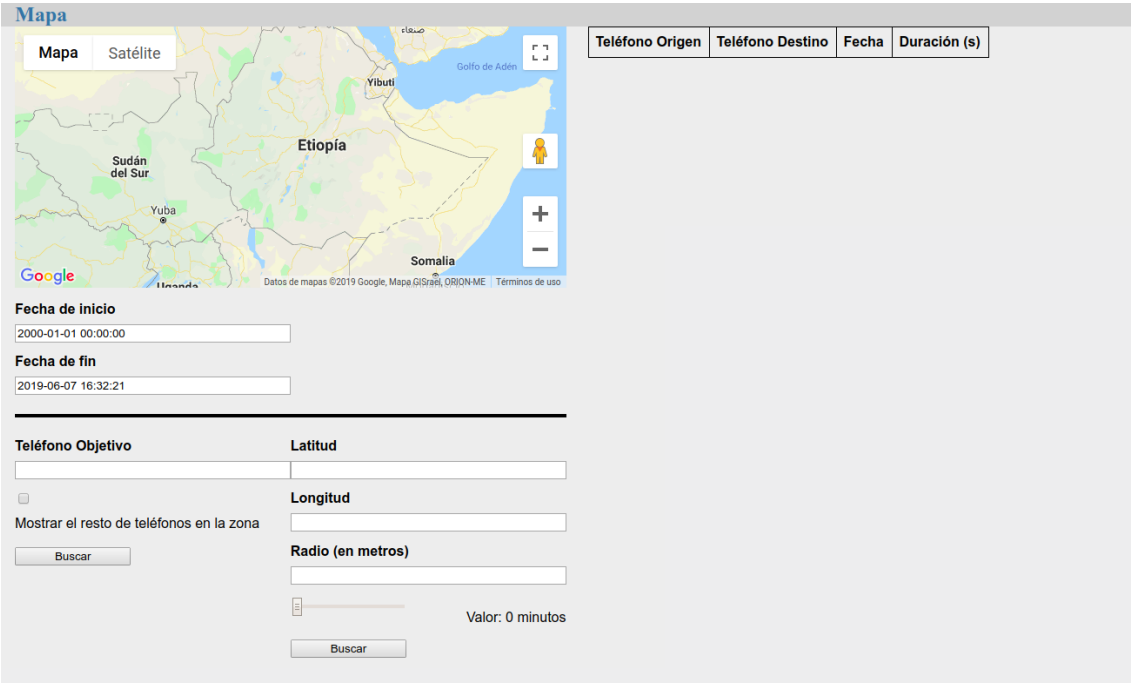


Figura 5.1: Interfaz Web de la aplicación

En 5.1 se puede ver el aspecto que tiene la página principal al acceder sin aplicar ningún filtro, y como se puede observar, contiene 2 paneles. En el panel izquierdo se tiene el mapa y los 2 formularios para aplicar los filtros. Ambos formularios comparten los input de *Fecha de inicio* y *Fecha de fin* y sirven para filtrar las llamadas en un intervalo temporal de interés.

El primer formulario consiste en una búsqueda de las ubicaciones donde ha estado un *Teléfono*

Objetivo, o sospechoso, realizando llamadas. Se ha añadido un filtro que si se marca, muestra también los teléfonos que han estado en el mismo área que el *Teléfono Objetivo*. De esta manera se podrían establecer posibles cómplices del sospechoso.

El segundo formulario consiste en la búsqueda, en una ubicación mediante *Latitud* y *Longitud* y un *Radio* en metros, de todas las llamadas realizadas en ese área. Además admite un slider temporal para filtrar para cada ubicación, los dispositivos que hayan realizado llamadas con una distancia temporal de más de X minutos (siendo X el número de minutos fijados en el slider). Lo que en este caso, se interpreta como los sospechosos que han permanecido en ese área más de X minutos cometiendo el crimen. De esta manera se pueden descartar posibles transeúntes inocentes.

En el panel derecho se tiene la tabla donde se visualizarán en detalle los datos de las llamadas en las ubicaciones pintadas en el mapa. Los datos consisten en un *Teléfono Origen* que llama a un *Teléfono Destino* en una cierta *Fecha* y con una *Duración* de llamada en segundos.

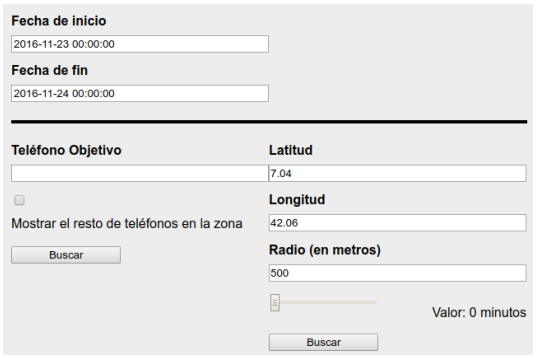
5.2. Ejemplo de uso

Primero, es importante fijar un contexto habitual para el uso de la aplicación. En este caso, se va a asumir la situación en que un crimen se ha cometido en una cierta ubicación en un intervalo de tiempo de menos de 24 horas. Por tanto, el objetivo de la aplicación es establecer posibles sospechosos a partir de su número de teléfono. A continuación se muestra una secuencia completa de uso de la aplicación para este problema.

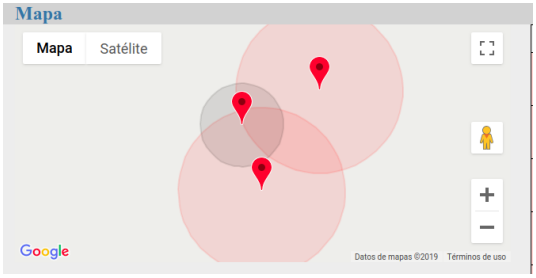
En primer lugar, se empieza fijando las fechas entre las que se ha cometido el crimen. A continuación, se usa el segundo formulario, pues se sabe una ubicación en coordenadas de latitud y longitud donde ha ocurrido el crimen, y se establece un radio de búsqueda. De momento no se realiza el filtro por tiempo de estancia, pues no interesa descartar casos aún. Por tanto, el formulario queda rellenado tal y como aparece en 5.2(a).

Como se ve en el mapa 5.2(b), con estos filtros, aparecen 2 antenas cuyos rangos de alcance (circunferencias en rojo) solapan con el área de búsqueda (circunferencias en negro). Por tanto, es muy probable que el sospechoso haya realizado llamadas conectándose a alguna de esas dos antenas.

Como se observa, hay muchos resultados en la tabla 5.2(c), por lo que ahora es cuando resulta interesante utilizar el filtro por tiempo de estancia, es decir, descartar los teléfonos que no han realizado varias llamadas en un intervalo pequeño de tiempo en el mismo lugar, pues es posible que simplemente hayan pasado por la zona en un vehículo y sin pararse. Por tanto aplicando el filtro para un tiempo de estancia mínimo, se consigue eliminar sospechosos en la zona, obteniendo los resultados en la figura 5.3. Como se puede apreciar, se ha conseguido reducir el número de sospechosos a solamente dos, cada uno en una ubicación distinta y en momentos distintos del día.



(a) Formulario rellenado



(b) Resultado en el mapa

Teléfono Origen	Teléfono Destino	Fecha	Duración (s)
285290001134867	34652896527	2016-11-23 15:23:58	21
285290001134867	34652896527	2016-11-23 15:24:32	22
285290001134867	34652896527	2016-11-23 15:25:09	8
285290001134867	34652896527	2016-11-23 15:25:46	42
285290001134867	34652896527	2016-11-23 15:26:50	17
235290041984001	34652896527	2016-11-23 15:27:31	28
285290001134867	34652896527	2016-11-23 15:33:18	35
285290001134867	34652896527	2016-11-23 15:35:23	172
285219215401382	34652896527	2016-11-23 15:57:52	25
285219215401382	34652896527	2016-11-23 15:58:30	173
285219215401382	34652896527	2016-11-23 18:50:28	60

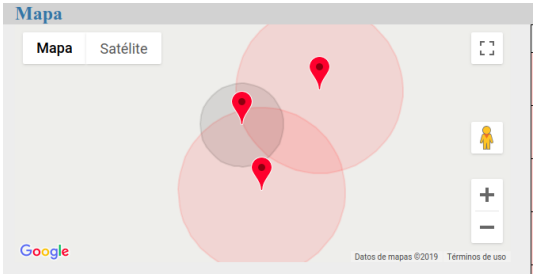
(c) Tabla de datos de ubicación superior

Teléfono Origen	Teléfono Destino	Fecha	Duración (s)
285289572740407	34652896527	2016-11-23 06:51:57	28
285289572740407	34652896527	2016-11-23 06:53:47	13
285289572740407	34652896527	2016-11-23 06:54:55	11
285289572740407	34652896527	2016-11-23 07:06:49	3
285289572740407	34652896527	2016-11-23 07:22:10	6
285289572740407	34652896527	2016-11-23 07:27:32	14
285289572740407	34652896527	2016-11-23 07:28:10	16
285289572740407	34652896527	2016-11-23 07:29:38	55

(d) Tabla de datos de ubicación inferior

Figura 5.2: Consulta por filtro de ubicación. En la figura 5.2(a) tenemos el segundo formulario rellenado, y en 5.2(b) mostramos el resultado en el mapa. Pinchando en la ubicación superior del mapa se muestran los datos en la tabla 5.2(c). En cambio, si se pincha en la ubicación inferior se obtiene la tabla 5.2(d)

(a) Formulario rellenado con filtro de tiempos de estancia



(b) Resultado en el mapa

Teléfono Origen	Teléfono Destino	Fecha	Duración (s)
285219215401382	34652896527	2016-11-23 15:57:52	25
285219215401382	34652896527	2016-11-23 15:58:30	173
285219215401382	34652896527	2016-11-23 18:50:28	60
285219215401382	34652896527	2016-11-23 21:17:10	12
285219215401382	34652896527	2016-11-23 21:17:50	168

(c) Tabla de datos de ubicación superior

Teléfono Origen	Teléfono Destino	Fecha	Duración (s)
285289572740407	34652896527	2016-11-23 06:51:57	28
285289572740407	34652896527	2016-11-23 06:53:47	13
285289572740407	34652896527	2016-11-23 06:54:55	11
285289572740407	34652896527	2016-11-23 07:06:49	3
285289572740407	34652896527	2016-11-23 07:22:10	6
285289572740407	34652896527	2016-11-23 07:27:32	14
285289572740407	34652896527	2016-11-23 07:28:10	16
285289572740407	34652896527	2016-11-23 07:29:38	55

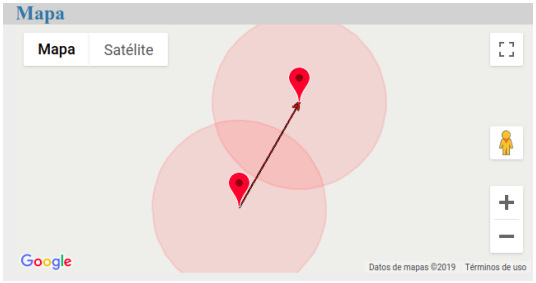
(d) Tabla de datos de ubicación inferior

Figura 5.3: Consulta por filtro de ubicación con tiempo de estancia. En la figura 5.3(a) tenemos el segundo formulario rellenado tal y como en 5.2(a), pero añadiendo un filtro extra de tiempo de estancia de más de 20 minutos. Enviando el formulario, se obtiene el mapa 5.3(b), que coincide con el de 5.2(b). Pinchando en cada ubicación, se obtienen las tablas 5.3(c) y 5.3(d), que son versiones filtradas de las tablas 5.2(c) y 5.2(d) respectivamente

A continuación, lo interesante es examinar las llamadas individuales de cada teléfono y las ubicaciones en las que las ha realizado. Por tanto, se puede rellenar el primer formulario con el *Teléfono Objetivo*, o equivalentemente, se puede clicar en las tablas de datos anteriores el teléfono a investigar.

Supongamos en este caso, que se sabe que el crimen se realizó por la mañana y no por la tarde o noche. En tal caso, el *Teléfono Objetivo* que se investigará será el que aparece en la tabla 5.3(d). Por lo que se procede a realizar ese filtro con el primer formulario, obteniendo los resultados en la figura 5.4.

(a) Formulario rellenado



(b) Resultado en el mapa

Teléfono Origen	Teléfono Destino	Fecha	Duración (s)
285289572740407	34652896527	2016-11-23 06:51:57	28
285289572740407	34652896527	2016-11-23 06:53:47	13
285289572740407	34652896527	2016-11-23 06:54:55	11
285289572740407	34652896527	2016-11-23 07:06:49	3
285289572740407	34652896527	2016-11-23 07:22:10	6
285289572740407	34652896527	2016-11-23 07:27:32	14
285289572740407	34652896527	2016-11-23 07:28:10	16
285289572740407	34652896527	2016-11-23 07:29:38	55

(c) Tabla de datos de ubicación inferior

Teléfono Origen	Teléfono Destino	Fecha	Duración (s)
285289572740407	34652896527	2016-11-23 22:09:35	160

(d) Tabla de datos de ubicación superior

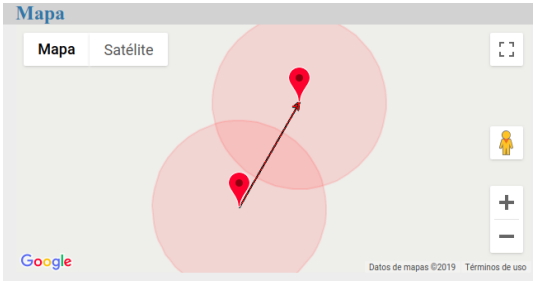
Figura 5.4: Consulta por filtro de *Teléfono Objetivo*. En la figura 5.4(a) tenemos el primer formulario rellenado, y en 5.4(b) mostramos el resultado en el mapa. Siguiendo el orden de la flecha, pinchando en la ubicación inferior del mapa se muestran los datos en la tabla 5.2(c). En cambio, si se pincha en la ubicación superior se obtiene la tabla 5.2(d)

Como se ve en la figura, se pinta la trayectoria seguida por el teléfono objetivo, y además se mantiene la funcionalidad de mostrar los datos al clicar en una posición. Podemos concluir que el sospechoso ha estado toda la mañana en la misma ubicación, y que en algún momento de la noche ha estado de nuevo en una ubicación cercana. No se puede determinar si ha permanecido allí todo el

día, o si ha vuelto posteriormente.

Ahora, supongamos que sabemos que el sospechoso ha tenido ayuda de un cómplice para realizar el crimen. Por tanto, en esta consulta es interesante poder ver los teléfonos que han realizado llamadas en la misma ubicación que el objetivo en instantes de tiempo cercanos a los que ha realizado el objetivo, por tanto se puede activar el checkbox y volver a enviar la consulta, obteniendo los resultados de la figura 5.5.

(a) Formulario rellenado con filtro activado de teléfonos en la misma zona



(b) Resultado en el mapa

Teléfono Origen	Teléfono Destino	Fecha	Duración (s)
285289572740407	34652896527	2016-11-23 06:51:57	28
285289572740407	34652896527	2016-11-23 06:53:47	13
285289572740407	34652896527	2016-11-23 06:54:55	11
285289572740407	34652896527	2016-11-23 07:06:49	3
285289572740407	34652896527	2016-11-23 07:22:10	6
285289572740407	34652896527	2016-11-23 07:27:32	14
285289572740407	34652896527	2016-11-23 07:28:10	16
285289572740407	34652896527	2016-11-23 07:29:38	55

(c) Tabla de datos de ubicación inferior

285219215401382	34652896527	2016-11-23 15:57:52	25
285219215401382	34652896527	2016-11-23 15:58:30	173
285219215401382	34652896527	2016-11-23 18:50:28	60
285219215401382	34652896527	2016-11-23 21:17:10	12
285219215401382	34652896527	2016-11-23 21:17:50	168
285289572740407	34652896527	2016-11-23 22:09:35	160

(d) Tabla de datos de ubicación superior

Figura 5.5: Consulta por filtro de *Teléfono Objetivo* con teléfonos en la misma zona. En la figura 5.5(a) tenemos el primer formulario rellenado tal y como en 5.4(a), pero activando el checkbox para mostrar el resto de teléfonos en la misma zona. Enviando el formulario, se obtiene el mapa 5.5(b), que coincide con el de 5.4(b). Pinchando en cada ubicación, se obtienen las tablas 5.5(c) y 5.5(d), que son equivalentes a las 5.4(c) y 5.4(d) respectivamente, pero mostrando el resto de teléfonos en la misma zona

Como se observa, el resultado tiene, además de las filas de la consulta anterior, filas de llamadas de otros teléfonos. Al estar ordenadas las filas por fecha de llamada, es muy sencillo ver teléfonos que hayan realizado llamadas en el mismo lugar y a tiempos cercanos a los del objetivo, al menos en la

imagen 5.5(d). Por tanto, se pueden asumir otros sospechosos que hayan realizado el crimen junto al sospechoso principal, y en tal caso, se procedería a investigar las llamadas realizadas por esos nuevos posibles cómplices. Así se procedería hasta que se tuvieran identificados todos los posibles sospechosos partícipes del crimen.

6

Conclusiones y trabajo futuro

6.1. Conclusiones

En este trabajo, se ha diseñado una aplicación que se encarga de proveer a un analista de información para resolver casos en los que sólo se conoce la ubicación aproximada de un crimen o el número de teléfono de la víctima o sospechoso de dicho crimen.

La aplicación permite insertar datos de un fichero en una base de datos con un formato que se puede definir dependiendo del fichero. Aunque, como los datos usados en este trabajo son simulados, y son de un muestreo bastante reducido, los casos que se pueden probar pueden resultar poco realistas.

También ofrece una interfaz web, donde se permite realizar consultas sobre las distintas llamadas telefónicas realizadas en un mismo área y permitiendo descartar las llamadas realizadas por posibles transeúntes que simplemente pasaran por la zona sin participar en el crimen. Además permite filtrar las llamadas telefónicas realizadas por un mismo teléfono objetivo y visualizar otros posibles terminales que estuvieran con dicho teléfono objetivo (posibles cómplices). Los resultados de estas consultas son mostrados en una mapa de Google Maps, que permite visualizar los datos de esas llamadas más detalladamente al clicar en las distintas ubicaciones del mapa.

6.2. Trabajo futuro

A continuación se detallarán las distintas consideraciones que se podrían tener en cuenta de cara a mejorar el trabajo ya hecho y ampliar su funcionalidad y alcance:

- **Reconocimiento automático de formatos de ficheros de datos.**
  Actualmente para definir los formatos de los ficheros de datos de antenas y llamadas telefónicas hay que escribir un fichero JSON manualmente. Debido a que los datos siguen formatos distintos, pero muy parecidos, sería interesante realizar un algoritmo que identificara automáticamente columnas de los ficheros relevantes en base a la cabecera de las columnas o al formato de los datos.
- **Filtrar teléfonos en base a las carreteras cercanas.**
  La API de Maps tiene funcionalidades que permiten identificar carreteras cercanas (Véase Snap to Roads [7]).

De esta manera se podrían excluir en los filtros llamadas que se realicen a alta velocidad siguiendo una carretera. Y por tanto, excluir transeúntes que simplemente viajaran en un vehículo por la carretera y que no hayan tomado parte en ningún crimen.

- **Mejorar la precisión de las trayectorias de los terminales.**
  Con los datos simulados actuales, sólo se puede estimar la ubicación de un terminal por áreas alrededor de las antenas a las que se conecta. Pero con un muestreo de datos reales suficiente sería posible estimar la ubicación de un terminal de forma más exacta mirando sus instantes de tiempo y las intersecciones entre los rangos de las distintas antenas. De esta manera, se podría simular una trayectoria más precisa para un sospechoso.

Bibliografía

[1] "Django: The web framework for perfectionists with deadlines." https://www.djangoproject.com/. Accessed: 2019-01-01.

[2] "Flask (a python microframework)." http://flask.pocoo.org/. Accessed: 2019-01-01.

[3] "Google maps platform documentación." https://developers.google.com/maps/documentation/?hl=es. Accessed: 2019-01-03.

[4] "Openlayers: A high-performance, feature-packed library for all your mapping needs." https://openlayers.org/. Accessed: 2019-01-03.

[5] "Leaflet webpage." https://leafletjs.com/. Accessed: 2019-01-03.

[6] "Openstreetmap." https://www.openstreetmap.org/. Accessed: 2019-01-03.

[7] "Snap to roads." https://developers.google.com/maps/documentation/roads/snap. Accessed: 2019-01-03.

[8] "Postgresql: Relational database." https://www.postgresql.org/. Accessed: 2019-01-05.

[9] "Neo4j graph platform." https://neo4j.com/. Accessed: 2019-01-05.

[10] "Postgis: Spatial and geographic objects for postgresql." https://postgis.net/. Accessed: 2019-01-10.

[11] "Opencellid: The world's largest open database of cell towers." https://opencellid.org/. Accessed: 2019-01-07.

[12] "Sqlalchemy: The python sql toolkit and object relational mapper." https://www.sqlalchemy.org/. Accessed: 2019-01-06.

[13] "Geoalchemy 2: Using sqlalchemy with spatial databases." https://geoalchemy-2.readthedocs.io/en/latest/. Accessed: 2019-01-10.

[14] "Python data analysis library." https://pandas.pydata.org/. Accessed: 2019-01-12.

Definiciones

API La interfaz de programación de aplicaciones es un conjunto de subrutinas, funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

bulk insert Proceso provisto por el sistema de manejo de base de datos para cargar multiples filas de datos en una tabla.

framework Estructura conceptual y tecnológica de asistencia definida que sirve de base para la organización y desarrollo de software.

MVC Modelo-Vista-Controlador es un patrón de arquitectura de software, que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos.

ORM O Object-Relational mapping es una técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y la utilización de una base de datos relacional.

Acrónimos

BBDD Bases de Datos.

BDOG Bases de Datos Orientadas a Grafos.

BDR Bases de Datos Relacionales.