Smart City Overseer

Javier Villarreal Rodríguez
Christian González Jiménez

Director: Jose Luis Vázquez Poletti
Director: Jose Manuel Velasco Cabo

Trabajo de fin de grado del Grado en Ingeniería Informática
Facultad de informática
Universidad Complutense de Madrid

Curso 2016/2017

Autorización

Los abajo firmantes, Christian González Jiménez y Javier Villarreal Rodríguez , dan su consentimiento a la Universidad Complutense de Madrid (UCM) a difundir y utilizar con fines académicos, no comerciales y reconociendo la autoría mediante una mención de los mismos, tanto de la presente memoria como de los contenidos audiovisuales mostrados en la misma, así como de la documentación que se entrega junto con este trabajo desarrollados durante el curso 2016-2017 bajo la dirección de los profesores Jose Luis Vázquez Poletti y Jose Manuel Velasco Cabo en el Departamento de Arquitectura de Computadores y Automática, y a la Biblioteca UCM a depositarlo en el Archivo Institucional E-Prints Complutense con el fin de incrementar la difusión, uso e impacto del trabajo en Internet y garantizar su preservación y acceso a largo plazo.

Christian González Jiménez

Javier Villarreal Rodríguez

Madrid, 16 de Junio de 2017

A Jose Luis Vázquez Poletti y a Jose Manuel Velasco Cabo que sin su ayuda y apoyo no hubiera sido posible.

Agradecimientos

A todos nuestros familiares y amigos que nos han apoyado desde que comenzamos y nos han dado ánimos y entusiasmo durante todo el proyecto.

Gracias en especial a Jose Luis Vázquez Poletti y a Jose Manuel Velasco Cabo por ayudarnos y guiarnos durante todo el trabajo, siempre con entusiasmo y una dedicación asombrosa.

Gracias también a nuestros compañeros Rafael Gómez Bermejo y Alvaro Urda Díaz por acompañarnos al Hackaton de Accenture "Hackathon Smart & Connected Cities".

Índice

Índice de figuras

Resumen

"Smart City Overseer " es un panel de control que permite gestionar los servicios de la ciudad de una manera eficiente por medio de la obtención de la información en tiempo real del estado, tanto real como virtual, de la ciudad y sus incidencias para ayudar al personal de estos servicios a tomar decisiones y determinar la magnitud de la afluencia de eventos a través del analisis big data de tweets correspondientes a esta geolocalización.

El panel de control proporciona una interfaz de usuario montada sobre un bot de telegram, el cual responde a cualquier pregunta relacionada con el estado de la ciudad tales como el estado del aire de la ciudad, el ruido ambiental de la ciudad, las farmacias de guardia y las incidencias de la ciudad.

Todo esto está sustentado sobre una infraestructura de red montada en el Cloud mediante Amazon Web Services, que pobla de información tanto la aplicación como el bot de Telegram de manera periódica.

Figura 1: Logo de la Aplicación

Palabras clave: Smart Cities, Big data, Cloud, NoSQL, Geolocalización, Estado de la ciudad, Emergencias.

Abstract

"Smart City Overseer " is a control panel that allows us to manage the services of the city in an efficient way by obtaining information of the condition of the city and its incidents, both real and virtual, in real time so as to help the personnel of those services to make decisions and determinate the magnitude of the affluence of events through the big data analysis of tweets corresponding to this geolocation.

Control panel provides a user interface built on a telegram bot, which answers any question referred to the condition of the city such as the air quality, the noise quality, all-night chemists' and incidents.

Everything is supported by a network infrastructure built on Cloud through Amazon Web Services, which fills both app and both with data periodically.

Figura 2: Logo de la Aplicación

KeyWords: Smart Cities, Big data, Cloud, NoSQL, Geolocation, State of the city, Emergencies.

1. Introducción

Con las nuevas tecnologías y el gran avance de la digitalización, muchas de las ciudades comienzan a tener sistemas de medición inteligentes de muchos parámetros como pueden ser la calidad del aire o el seguimiento del tráfico. Este factor nos lleva a plantearnos, ¿Cómo podemos controlar la ciudad? ¿Qué información necesitamos para la mejora de la atención pública?

El proyecto Smart City Overseer trata de satisfacer la necesidad de cualquier rol de la ciudad de tal manera que se pueda ver la información necesaria para éste a simple vista y la posibilidad de tomar respuesta rápida y óptima mediante la confirmación del usuario.

Para este cometido se ha construido una infraestructura que se encarga de obtener la información del estado de la ciudad y su procesamiento. Con esta información procesada, un mapa y numerosas APIs poblaremos el núcleo principal del proyecto, un panel de gestión de Smart Cities.

Introduction

With the new technologies and the great advance of digitalization, lots of cities start to have smart systems which calculate a lot of parameters such as the air quality index or traffic tracking. That make,therefore, the questions of how can we control the city or what data do we need to boost the customer service increase.

The project Smart City Overseer comes to satisfying the necessities of any role of the city so that it can see the necessary data and the possibility of making decisions quickly and optimally through the user's confirmation.

To this purpose, a network infrastructure has been built, which manages to obtain the data of the condition of the city and his processing. With this processed data, a map and many APIs we will fill the main core of the project, a smart cities-control panel.

2. Motivación

Como programadores vimos el comienzo de la informatización de todos los datos de la ciudad y la tremenda necesidad de tener estos datos a una simple vista sobre un mapa y su utilidad respecto a los diferentes roles de la ciudad. Por esto decidimos crear Smart City Overseer de tal manera que pudiera satisfacer todas las necesidades para cualquier persona respecto a su ciudad.

- Crear una interfaz útil y cómoda para el visionamiento de todos los datos disponibles de la ciudad.
- Crear un panel de datos de tal forma que sea escalable para la posibilidad de añadir nuevas fuentes de datos posteriormente.
- Crear un sistema desacoplado de tal manera que la carga se divida entre los varios sistemas.
- Ofrecer información del estado de la ciudad que pueda ser utilizada a la hora de la toma de decisiones.
- Hacer un histórico de la información recogida con el fin de conocer la evolución de la ciudad.
- Posibilidad de juntar fuentes información para formar otras nuevas como por ejemplo una capa de salud de cada uno de los distritos de la ciudad.

Entorno Smart City Overseer

Con el auge del internet de las cosas, las sociedades cada vez están más digitalizadas hasta el punto de convertirse en una red de dispositivos interconectados de modo que la vida cotidiana sin el ciberespacio en la actualidad es una separación inconcebible.

Los ciudadanos cada vez están más acostumbrados al uso de dispositivos digitales y éstos se han convertido en un lienzo sobre el que plasmar la gran cantidad de datos que produce la urbe y ofrecerla de forma totalmente transparente.

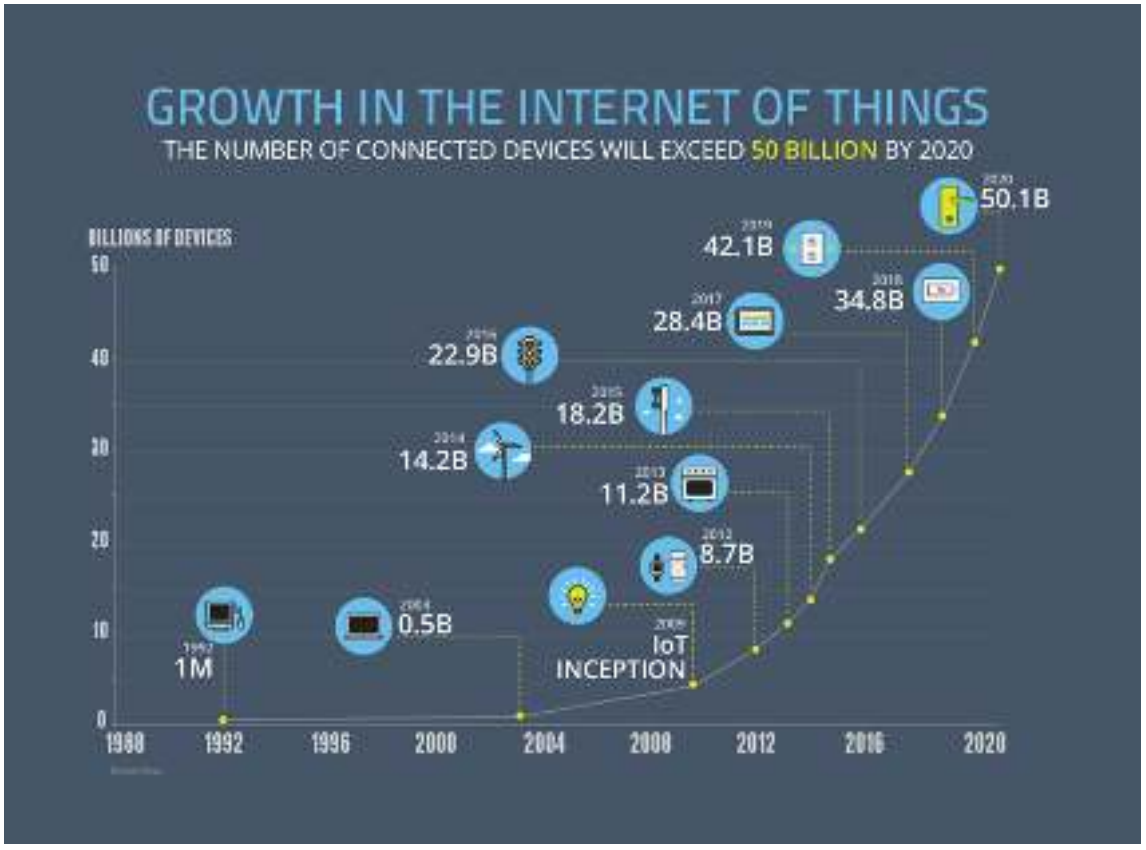


Figura 3: Estudio sobre el internet de las cosas.

Además de esto, el paradigma del Big data es un factor a tener en cuenta en las Smart Cities ya que de los datos que se obtienen se puede hacer un análisis de todos los servicios que se ofrecen en la ciudad.

Estos datos provienen de diferentes fuentes, pueden ser de geolocalización, procedente de sensores que toman medidas medioambientales o de movilidad a través de los sensores de tráfico de manera que se

ahorra y se contamina menos.

Por ejemplo, mediante las cámaras instaladas en las calles y semáforos se pueden prever los atascos que se originan como consecuencia de un accidente o una incidencia. En tiempo real se puede cambiar la ruta de los autobuses o informar al ciudadano de rutas alternativas para evitar el atasco.

Otro ejemplo es el incremento de la seguridad ciudadana mediante el cruce de información de las cámaras de videovigilancia, geolocalización de coches de policía y bomberos, sensores de movilidad o de alertas y detectores de humo y fuego.

La infraestructura que soporta las Smart Cities se surte del Cloud Computing para el procesamiento de toda su información y valiéndose de todas las ventajas que le aporta tales como la elasticidad, la escalabilidad y el buen aprovechamiento de los recursos computacionales minimizando así el impacto ambiental y el gasto energético de la ciudad.

3. Estado del arte

En este capítulo presentamos el estudio de mercado previo al proyecto donde se buscaron aplicaciones similares a lo que se deseaba desarrollar y se analizaron viendo ventajas, desventajas e ideas que pudieran servirnos a la hora de realizar el proyecto.

Smart City Overseer busca aunar lo mejor de cada una de ellas haciendo uso de conceptos de las áreas de Smart Cities, Big Data y Cloud. Desde el punto de vista de las Smart Cities, busca proveer de información de los diferentes servicios de la Ciudad de una manera centralizada a través del panel de control. Desde el punto de vista del Big Data, busca tratar toda la cantidad de información que estos servicios producen y ofrecer la misma desde un servicio centralizado. Desde el punto de vista del Cloud, hace uso de una arquitectura bajo demanda que ofrece la información del Open Data de Madrid y que se ve incrementada en función de la demanda de los usuarios que hacen uso del servicio (Figura 4).

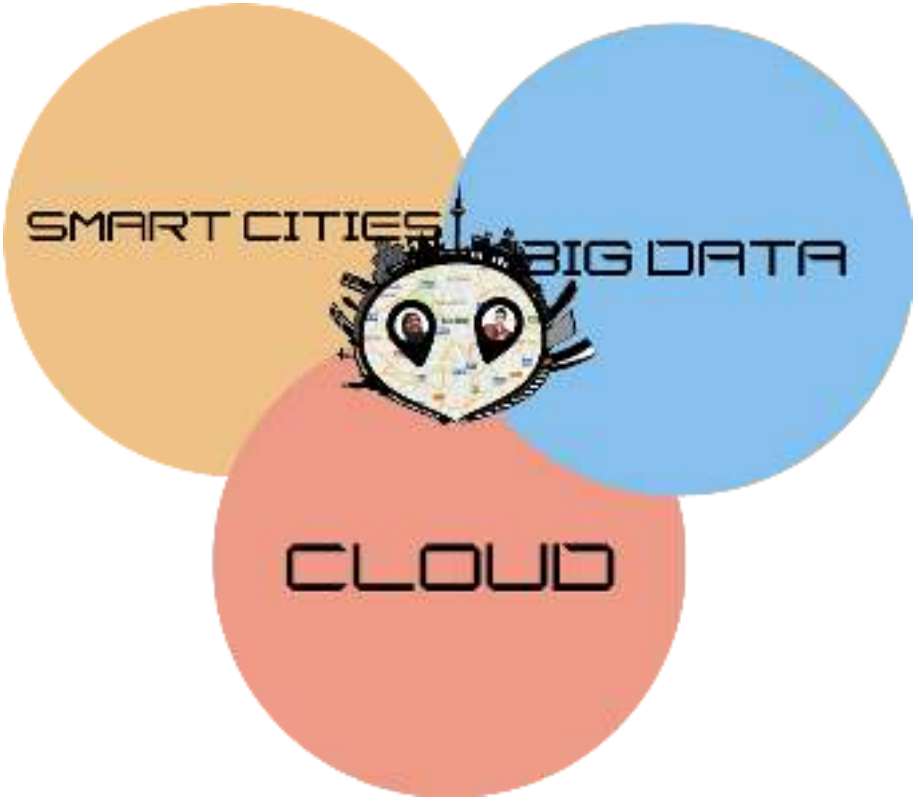


Figura 4: Áreas en las que se centra Smart City Overseer

Nuestra conclusión tras analizar bastantes modelos y aplicaciones fue que no había ninguna aplicación que satisficiera la necesidad de poder integrar capas dinámicamente y mostrarlo dependiendo del rol del usuario y mediante estas capas calcular acciones como el posible despliegue de unidades en caso de un rol de personal de emergencias o policía. Aparte de esto, no encontramos ninguna aplicación de escritorio que pudiera mostrar datos de varios tipos, sin centrarse únicamente en tráfico, meteorología o incidencias.

3.1. FIRMS Web Fire Mapper

FIRMS Web Fire Mapper es una aplicación desarrollada por la NASA para el seguimiento activo de los fuegos en el mundo. Esta aplicación cumple su finalidad pero no se puede limitar su búsqueda de fuego a una ciudad, ni ordenar por prioridad.
Link (URL): `https://firms.modaps.eosdis.nasa.gov/firemap/`

3.2. NavegoMadrid

NavegoMadrid es una aplicación de consulta Web realizada por la EMT para la disposición de sus datos en el mapa y consulta de éstos en tiempo real. Esta aplicación incluye información sobre el estado de los autobuses, metro y cercanías con sus tiempos de espera e incidencias. Esta aplicación tiene un objetivo parecido pero desde nuestro punto de vista incompleto a la hora de analizar toda la ciudad desde un punto de vista inteligente.
Link (URL): `http://www.emtmadrid.es/mapaweb/emt.html`

3.3. Infocar

Infocar es una aplicación de consulta web realizada por la DGT para el análisis en tiempo real del tráfico con sus incidencias y la meteorología peligrosa que pueda afectar a las carreteras en España. Esta aplicación cumple bastantes requisitos pero no incluye los eventos culturales que pueden afectar al tráfico por tanto en comparativa con Smart City Overseer, es una aplicación con menos datos y más limitada.
Link (URL): `http://infocar.dgt.es/etraffic/`

3.4. Aire de Madrid

Aire de Madrid es una aplicación para dispositivos móviles creada por SICE para medir la contaminación del aire mediante cada estación meteorológica de Madrid, con esta aplcación se puede observar los niveles de todos los gases en cada estación, en comparativa con SmartCities Overseer ofrece una parte de la información contemplada pero no dispone de un historial de los datos.
Link (URL): `https://play.google.com/store/apps/details?id=com.phonegap.airemadrid&hl=es`

4. Descripción general

Smart City Overseer es un panel de control que permite manejar los servicios de la ciudad de manera eficiente por medio de la obtención de información en tiempo real del estado, tanto real como virtual, de la ciudad y de sus incidentes para ayudar al personal de estos servicios a tomar decisiones y determinar la magnitud de afluencia de los eventos bajo los análisis de big data de tweets correspondientes a la geolocalización del evento.

Ventajas

- **Ayuda en la toma de decisiones lo más rápido posible respecto a un evento o incidente.** Dependiendo del rol que desempeñe nuestro usuario podrá ver unas alertas u otras y mediante el sistema Smart City Overseer podrá determinar si mandar determinados servicios/unidades al incidente u evento dependiendo de su cercanía o unidades disponibles cercanas de tal manera que la respuesta sea lo más rápido posible teniendo en cuenta todos los factores como el tráfico o las obras.
- **Control de toda la ciudad mediante un vistazo general.** Mediante Smart City Overseer podremos tener un vistazo general de la ciudad con todos su servicios simplemente con la gestión de capas que queramos visualizar.
- **Comprobar los eventos programados por la Comunidad.** En Smart City Overseer se pueden ver todos los eventos programados por la Concejalía de Cultura para los eventos de los siguientes 100 días.
- **Seguimiento de los vehículos en servicio.** Gracias a Smart City Overseer se puede hacer un seguimiento de las unidades mandadas de cada servicio a los distintos eventos e incidentes.
- **Conocer el estado acústico y aéreo de la ciudad.** A través de Smart City Overseer se puede observar si la ciudad tiene unos niveles óptimos en cuanto a contaminantes y ruido.

Descripción técnica

Smart City Overseer es una aplicación multiusos para los diferentes roles de la ciudad; ya sea policía, bombero, concejal, etc. Ésta aplicación proporciona la información que cada uno de ellos necesita sobre la ciudad en el desempeño de su actividad.

Estos roles desempeñan un papel importante en los servicios que gestionan la ciudad. Los servicios son los encargados de mantener el estado de la urbe, la cual poco a poco, debido a accidentes, lluvias, desastres... se irá deteriorando a lo largo del tiempo provocando desperfectos en su infraestructura. A estos desperfectos se les asignan incidencias que dichos roles irán solventando mediante los servicios brindados por Smart City Overseer .

El uso de Smart City Overseer varía según el tipo de rol.

- Para el personal de la policia destacan las siguientes funciones, todas ellas giran en torno a las comisarías que se ubican en Madrid.
    - Gestión de incidencias de la ciudad: El personal policial podrá ver todas las incidencias que suceden en Madrid y podrá atenderlas mandando patrullas para resolverlas. Una vez dicha patrulla llegue al lugar de la incidencia dicha incidencia se dará como resuelta a no ser que la misma patrulla determine a través del contacto con su superior que se necesitan más efectivos en la zona.
    - Gestión de las cámaras de la ciudad: El personal policial podrá ver lo que sucede en cada rincón de la ciudad a través de la recepción de imágenes en tiempo real de las cámaras que están colocadas en Madrid y así, determinar el estado de las calles de Madrid.
    - Gestión del tráfico de la ciudad: El personal policial podrá ver el estado del tráfico de la ciudad con el fin de determinar el grado de congestión de cada calle de Madrid para poder enviar efectivos para descongestionarlas en caso de atasco.
    - Conocer el estado del aire y acústico de la ciudad.

- Para el personal sanitario destacan las siguientes funciones.
    * Gestión de incidencias de la ciudad: El personal sanitario podrá ver todas las incidencias que suceden en Madrid y podrá atenderlas en caso de que haya heridos en un accidente mandando ambulancias para atenderlos. Una vez dicha ambulancia llegue al lugar de la incidencia dicha incidencia se dará como resuelta a no ser que la misma ambulancia determine a través del contacto con su superior que se necesitan más efectivos en la zona.
    * Conocer el estado del aire y acústico de la ciudad.
- Para el personal de bomberos destacan las siguientes funciones.
    * Gestión de incendios de la ciudad: El personal de bomberos podrá ver todos las incendios que suceden en Madrid y podrá atenderlos mandando un camión de bomberos para apagar el fuego. Una vez dicho camión llegue al lugar del incendio dicho incendio se dará como resuelto a no ser que el mismo camión determine a través del contacto con su superior que se necesitan más efectivos en la zona.
    * Conocer el estado del aire y acústico de la ciudad.
- Para la gestión de la plataforma será necesario un administrador, que se encarga de gestionar todo el sistema informático de Smart City Overseer y añadir y eliminar capas al panel de control de forma dinámica.
- Fuera de la plataforma, se dispone de un bot de telegram para todos los usuarios abiertamente desde el cuál se puede conocer el estado del aire y acústico de Madrid, la lista de farmacias de guardia, incidencias de tráfico, los eventos que suceden en la ciudad y las paradas de autobus así como las llegadas de autobuses. Éste se ha desarrollado durante el curso y surgió en base a una idea que tuvimos en la Hackathon de Accenture. El objetivo de este bot es dar a conocer los datos abiertos de Madrid al ciudadano de una manera útil mediante la botnet de telegram de manera que solo necesite el programa de mensajería sin necesidad de un programa externo.

5. Arquitectura

En este capítulo se va a detallar de manera concisa la arquitectura del sistema y su diseño.

Arquitectura de comunicación

Para el montaje del procesamiento necesario tanto para la aplicación como para el bot de Telegram se necesitaba una disposición de los servidores en el Cloud de manera que fueran siempre accesibles por los diferentes módulos de la aplicación y del bot. Para esta accesibilidad se han realizado una separación en dos servidores Amazon Web Services EC2 de manera que uno se ocupe del procesamiento y otro se ocupe de la base de datos (Figura 5) de tal manera que para grandes cantidades de datos puedan estar separados.

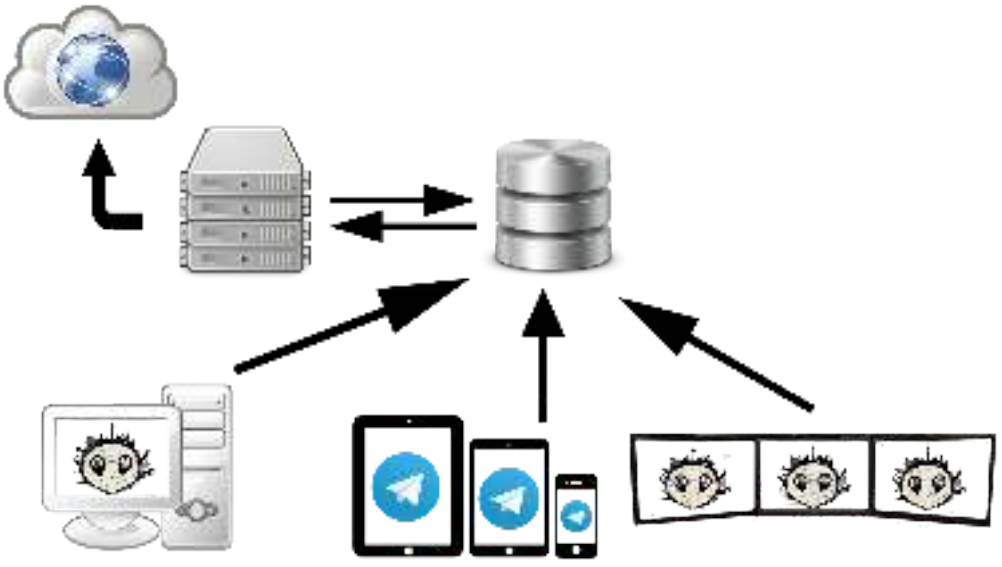


Figura 5: Arquitectura de los sistemas

Arquitectura de los lenguajes

Para las distintas áreas de desarrollo del proyecto hemos ido utilizando distintos lenguajes (Figura 6). Para la parte de procesamiento predomina el lenguaje de programación Python, el cuál hace consultas a MongoDB para insertar datos. Para la parte de la aplicación de escritorio, hemos usado Java, el cual hace uso de Maven para la inclusión de librerías de manera automatizada y usa para la transmisión de información el formato JSON. Para el bot de telegram, hemos usado Python junto con su librería telebot, la cual levanta el bot y permite que se le hagan peticiones a la API de Telegram sirviendo como punto de entrada.

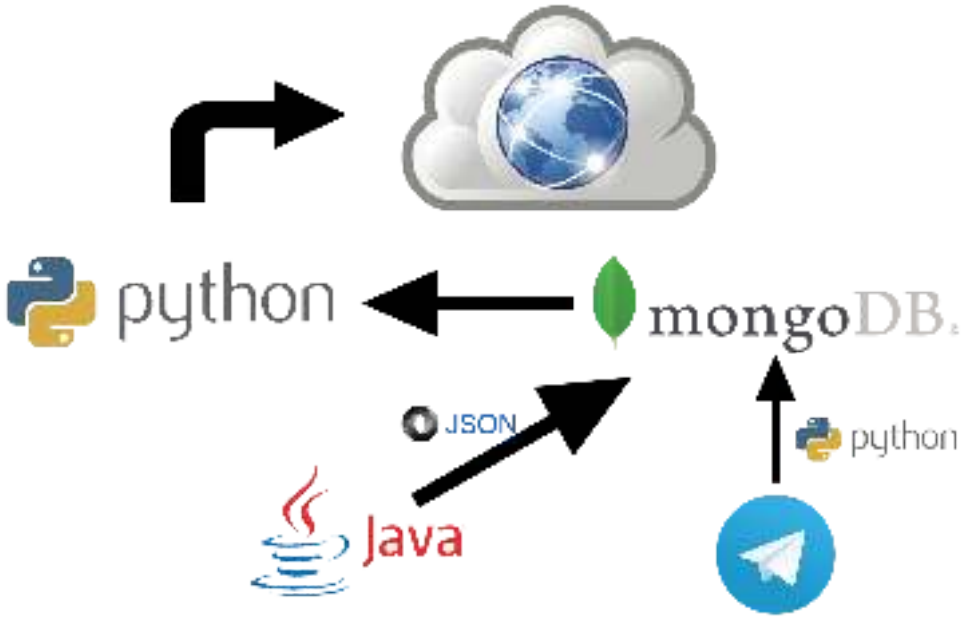


Figura 6: Arquitectura de los lenguajes

Diseño y especificación

El esquema que hemos utilizado para estructurar la aplicación ha sido el modelo vista controlador (Figura 7) ya que permite separar los datos del sistema de la lógica de la aplicación y la interfaz gráfica. Posteriormente, se ha usado el patrón Data Access Object para separar la lógica de la aplicación del acceso a los datos.

Este esquema separa el código en tres niveles:

- **Modelo** es una representación de los datos que maneja el sistema.
- **Vista** es el mecanismo de interacción del usuario con la lógica del negocio.
- **Controlador** es el intermediario entre la vista y el modelo y reacciona ante los eventos del usuario pidiéndole los datos al modelo y una vez que éste los tiene se los pasa a la vista para mostrarlos.

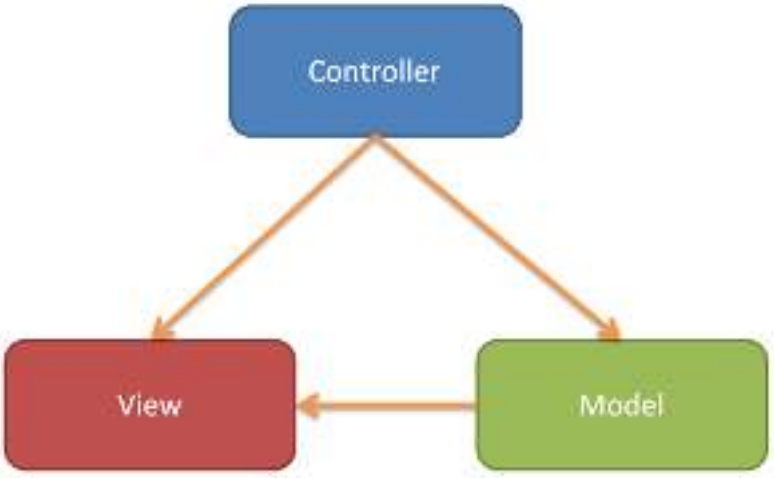


Figura 7: Modelo vista controlador

Gestión de usuarios

Para la gestión de usuarios se dispone de un sistema de autenticación de usuarios que da acceso al panel control mediante nick de usuario y contraseña (Figura 8). Éste verifica de qué rol es el usuario y le lleva a su correspondiente panel de gestión. La gestión de roles está basada en un sistema de permisos.

El administrador podrá configurar los permisos de los roles, los cuáles corresponden a un permiso por cada capa (Figura 9). Para añadir capas, es necesario que no utilicen el mismo identificador y el mismo nombre que alguna colección almacenada en la base de datos. Así mismo, para eliminar capas, solo se pueden eliminar aquellas que no pertenezcan al core de Smart City Overseer. Las capas que están

Figura 8: Login de la aplicación

incluidas al core se especifican en un fichero muy similar al de estructuración de capas que veremos más adelante y que se encuentra almacenado en el servidor de base de datos.

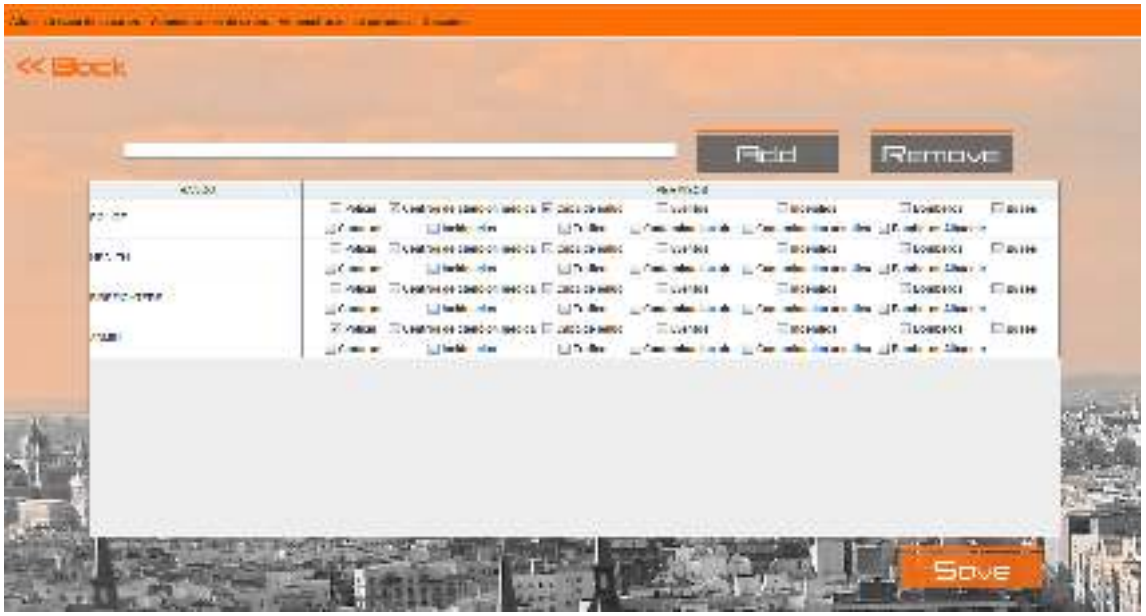


Figura 9: Gestión de permisos

Diagramas

Modelo de la aplicación

El modelo consta de un sistema de plugins (Figura 10). Éste es el encargado de gestionar las fuentes de datos que añade el usuario a la aplicación, de manera que hace a la aplicación totalmente modular y ampliable en cuanto a tema de cantidad de información que pobla la base de datos.

El sistema de plugins se divide en dos partes claramente diferenciadas:

- **BackgroundLayer**: Las capas en segundo plano o BackgroundLayers son fuentes de datos que realizan tareas en segundo plano y como tal, no muestran información al usuario sino que aumentan la funcionalidad de la aplicación calculando, por ejemplo, cuando una patrulla ha llegado a su destino o informando de cuáles son las comisarias más cercanas al incidente.
- **Layer**: Las capas propiamente dichas son añadidas al sistema de plugins añadiendo fuentes de datos a la aplicación. Estas capas pueden ser del core y como tal, vienen añadidas por defecto a la aplicación pero el usuario también puede añadir más capas, dinámicas o estáticas, a la aplicación, ampliando la información útil que el usuario puede utilizar para un mejor desempeño de la realización de sus tareas rutinarias. Las capas presentan dos partes importantes. Por un lado, la información de la capa y, por otro lado, el lienzo de gráficos para el caso en el que la capa quiera mostrar algún gráfico en el mapa como puede ser la capa de salud que muestra un radio donde se ubica el distrito y establece un nivel de salud para el distrito dado. Siendo 3(muy bueno) 2(normal) y 1(malo). Para ver las capas por defecto, ve al Ánexo I.

Figura 10: Modelo de la aplicación

Diagrama de clases de los controladores

La aplicación presenta un controlador principal, el cuál denominamos mapController. Éste a su vez se desglosa en dos controladores secundarios y una información asociada que corresponde con el identificador de sesión del usuario que hemos logueado (Figura 11). Los controladores secundarios son los siguientes:

- **PluginSystem**: El sistema de plugins es el elemento principal sobre el que se sustenta la modularidad de la aplicación ya que la información que se va adhiriendo al panel, se añade en forma de capas sobre las que luego se irán añadiendo permisos a los distintos roles de la aplicación en el caso de que sea necesario.
- **MainFrame**: El controlador de la vista principal es el encargado de relacionarse con el controlador principal y pasárselo al resto de vistas para poder así de esta manera ejecutar acciones sobre el modelo. Éste se muestra con un CardLayout, el cuál va cambiando en función del rol que se está logueando y, a su vez, muestra el abánico de opciones que cada rol puede manejar con otro CardLayout interior.

Figura 11: Diagrama de clases de controladores

Diagrama de clases

La aplicación está estructurada en una serie de controladores que hemos detallado en la anterior sección.

El controlador principal o mapController posee un controlador de la vista, un sistema de plugins y una variable que apunta al identificador de sesión del usuario (Figura 12).

- **El Controlador de la vista o MainFrame** tiene tres paneles correspondientes al rol del Administrador, al resto de usuarios y al panel de autenticación.
  - **El panel del administrador o Admin Panel** tiene la vista del mapa, la vista de añadir usuarios, la de editar usuario y eliminarlo y la vista de gestión de permisos.
  - **El panel del usuario o User Panel** tiene la vista de editar perfil, la vista del mapa y la configuración de la aplicación.
- **El sistema de plugins o Plugin System** contiene una tabla hash que corresponde con los identificadores de los plugins y el plugin (Layer), una tabla hash que hace de equivalencia entre el nombre natural del plugin y el identificador del plugin y una capa correspondiente a los servicios en segundo plano (BackgroundLayer).

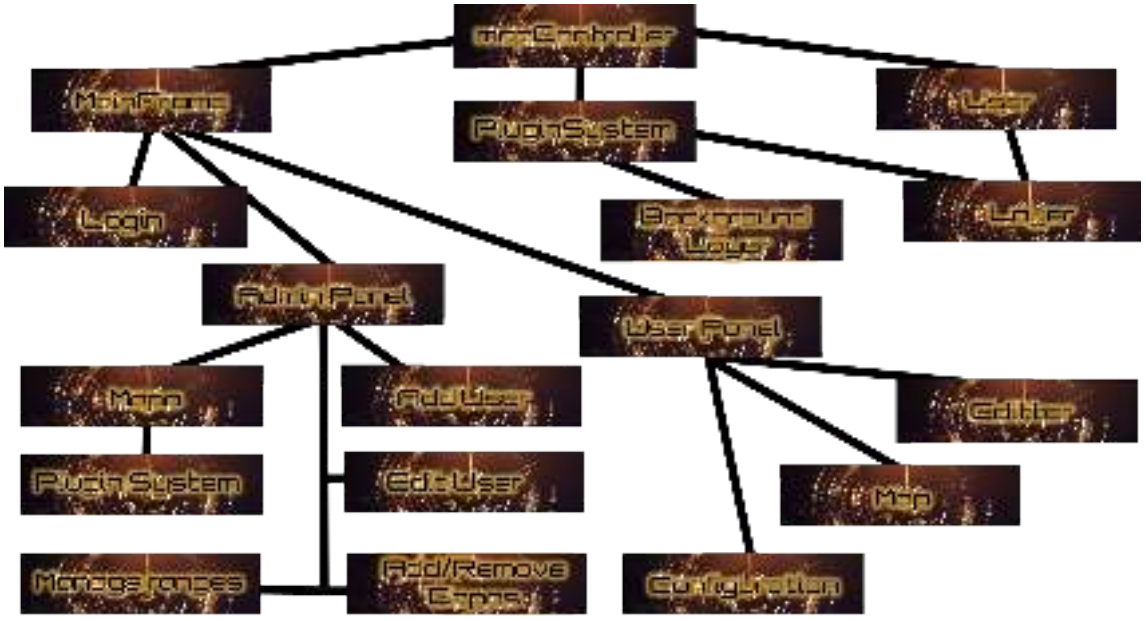


Figura 12: Diagrama de clases de toda la aplicación

Fichero de configuración de capas

El dinamismo de las capas se gestiona a nivel de servidor. La aplicación consta de un fichero de configuración de capas alojado en el servidor donde se encuentran los scripts de procesamiento. En este caso, en el servidor de procesamiento. A nivel de gestión de capas en el fichero de configuración se agrupan las fuentes de datos en capas mediante una sintaxis propia.

La sintaxis sigue el esquema para los grupos de fuentes de datos o capas indicado en el Anexo II.

Dentro de la fuente de datos se incluyen parámetros de configuración tales como url, fname, output y type, que posteriormente se utilizarán para descargar las fuentes de datos en el servidor. El parámetro database sirve para que una vez descargados los datos se suban a la base de datos correspondiente. Y el parámetro isRefresh indica si es una capa histórica o no. En el caso de que sea histórica, con cada actualización periódica se irá añadiendo más información a esa colección. Si no es histórica, lo que se hace es que con cada actualización se borra lo que haya en la colección. De esta manera, nos aseguramos de acumular los datos que nos interesan guardando un histórico y mostrando

otros en tiempo real.

Diseño de la Base de Datos

La base de datos es dinámica y NoSQL, todo esto usando MongoDB de manera que el formato de campos es variable dependiendo de los datos que vayamos a usar. Al ser una base de datos No-Relacional, no tiene implicaciones entre las tablas como se muestra a continuación (Figuras 13,14,15, 16, 17, 18).

Figura 13: Tablas: Cámaras, fuegos activos y contaminación del aire

Figura 14: Tablas: Paradas de bus, farmacias y eventos

Figura 15: Tablas: Parques de bomberos, Incidentes de la EMT e Incidentes de la vía pública

Figura 16: Tablas: Lugares de atención médica, contaminación del aire y comisarías

Figura 17: Tablas: Tráfico, usuarios y vehículos

Figura 18: Tabla: Roles

6. Tecnologías

Antes de proceder con la creación de la aplicación hicimos una evaluación de las diferentes tecnologías que podíamos usar para su realización dotándolo tanto de funcionalidad como seguridad.

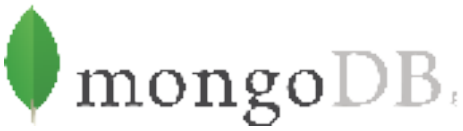
6.1. Tecnologías utilizadas

A continuación, vamos a presentar las tecnologías que hemos utilizado durante toda la realización del proyecto.

6.1.1. Java

Como lenguaje utilizado para la aplicación cliente se determinó usar Java, esta decisión fue realizada debido a que todos los integrantes del proyecto dominan el lenguaje y se le pueden incluir muchas librerías útiles para mejorar su utilidad. Por otro lado, la interfaz gráfica que aporta Java es muy personalizable y por tanto esto se adaptaba bastante bien a lo que nosotros queríamos realizar. Otro aspecto a favor por el que decidimos usar este lenguaje como principal es que podía usarse como nexo entre los demás ya que todas las tecnologías que teníamos en mente podía incluirse de una manera u otra mediante Java.

6.1.2. MongoDB



En cuanto a almacenamiento de datos decidimos usar MongoDB debido a que como base de datos no-relacional tiene bastante éxito gracias a que tiene unas transiciones muy rápidas, y esto para nuestro proyecto es imprescindible.

6.1.3. Python

En cuanto a lenguajes del lado del servidor, hemos utilizado este lenguaje de programación porque es muy legible, posee una gran cantidad de librerías que se adaptan a las que necesitábamos para la parte del proyecto referida al sistema en tiempo real que se encarga de procesar los datos para posteriormente insertarlos en la base de datos. Otro aspecto por el que decidimos usarlo fue por el afán de aprender un lenguaje que tiene tantos usos como éste.

6.1.4. XML

Hemos utilizado este lenguaje de marcado como base para adaptar los datos para la inserción en la base de datos. Es un lenguaje muy cómodo y junto con Python es una herramienta muy poderosa ya que permite a los documentos xml ser convertidos en diccionarios de una manera muy sencilla gracias a la librería de Python "xmltoDict" y ésto agiliza mucho la inserción en base de datos.

6.1.5. GIT

Como sistema de control de versiones para la aplicación, decidimos usar Git, debido a que ya lo habíamos usado antes y nos parece un sistema bastante cómodo para poder trabajar conjuntamente.

6.1.6. Telegram

Hemos usado Telegram para la creación de un bot de manera que la información de uso cotidiano esté disponible para todos los usuarios a pie de calle, de manera que sea una cosa siempre activa y disponible para todos los usuarios que visiten Madrid, teniendo la información actualizada en tiempo real y siempre disponible en el sistema de mensajería.

6.1.7. Twitter

Hemos utilizado la API de Twitter para obtener los tweets de la cuenta de Twitter Emergencias Madrid, ya que ofrecen información de las emergencias e incidencias que hay en Madrid. Dichos tweets contienen, por ejemplo, fotos de los incidencias entre otras informaciones relevantes.

6.1.8. Amazon Web Services EC2

Amazon Web Service dispone de un servicio de cloud, donde nosotros hemos montado nuestras máquinas que sustentan la infraestructura de Smart City Overseer. Esta infraestructura se encarga de poblar tanto el panel de control y el bot de Telegram.

6.1.9. JSON

JSON es el formato que utiliza MongoDB para almacenar los documentos de las colecciones de la base de datos. Éste es el formato que hemos utilizado para intercambiar información entre la aplicación de escritorio y la propia base de datos.

6.2. Tecnologías descartadas

A continuación, vamos a presentar las tecnologías que hemos descartado durante la realización del proyecto.

6.2.1. ArcGis

Al comienzo barajamos la posibilidad de usar ArcGis debido a que junto a Google Maps es lo que más se usa para realizar aplicaciones de mapas. Tras testear el uso tanto de su interfaz online como sus ArcGis applications, decidimos descartarla debido a que no nos daba la elasticidad que nosotros necesitábamos, ya que, necesitaba tener las capas integradas anteriormente a la aplicación y nosotros deseábamos poder añadir más capas elásticamente.

6.2.2. Google Maps

Esta tecnología fue la primera en la que pensamos a la hora de representar datos en mapas, pero tras realizar una pequeña investigación la descartamos debido a su forma de pintar capas por polígonos de marcaje previo.

6.2.3. MySQL

Al comienzo del desarrollo del proyecto pensamos en esta tecnología para la base de datos pero al tener tantos datos dispersos vimos la necesidad de usar una base de datos no relacional y descartamos esta tecnología.

6.2.4. KML

Al comienzo lo utilizamos debido a que al ser un metalenguaje sacado de XML, incluía únicamente lo necesario para incluirse en mapas. Pero debido a que este lenguaje está más especializado en la inclusión de mapas construidos sobre tecnologías web decidimos descartarlo y usar XML para un mejor uso en nuestra aplicación de escritorio.

7. Casos de uso

Nuestro trabajo tiene varios casos de uso ya que, desde un principio, está orientado a que distintos roles de la ciudad lo usen.

Entre estos roles de usuario está el ciudadano de a pie, el cual usará el bot de telegram para ofrecer la información de la ciudad. Luego están los diferentes servicios de la ciudad, entre los que destaca el cuerpo de Policía, el cuerpo de Bomberos, el cuerpo médico y un Administrador.

Caso A: Viendo el estado del aire de la ciudad a través de Smartcity Overseer Bot

Pongámonos en el caso de que el usuario quiere saber el estado del aire de la ciudad para saber si cuando coja el coche para ir a trabajar hay un episodio de contaminación y establezcan que los coches con matrícula par o impar puedan circular con total libertad.

Para ello, el usuario cogerá su móvil y accederá al telegram. Teniendo agregado el bot a su cuenta, pulsará en el botón START para comenzar a preguntarle al bot cómo está hoy el aire de Madrid. Acto seguido, se le desplegará una selección de botones que le ofrecerán la lista de datos a los que el usuario tendrá acceso. Como quiere saber el estado del aire de la ciudad, pinchará en la opción de "Estado del aire". Una vez realizada la petición, el usuario recibirá un mensaje de respuesta con la cantidad de NO2 que hay hoy presente en el aire de la ciudad en cada hora del día (Figura 19).

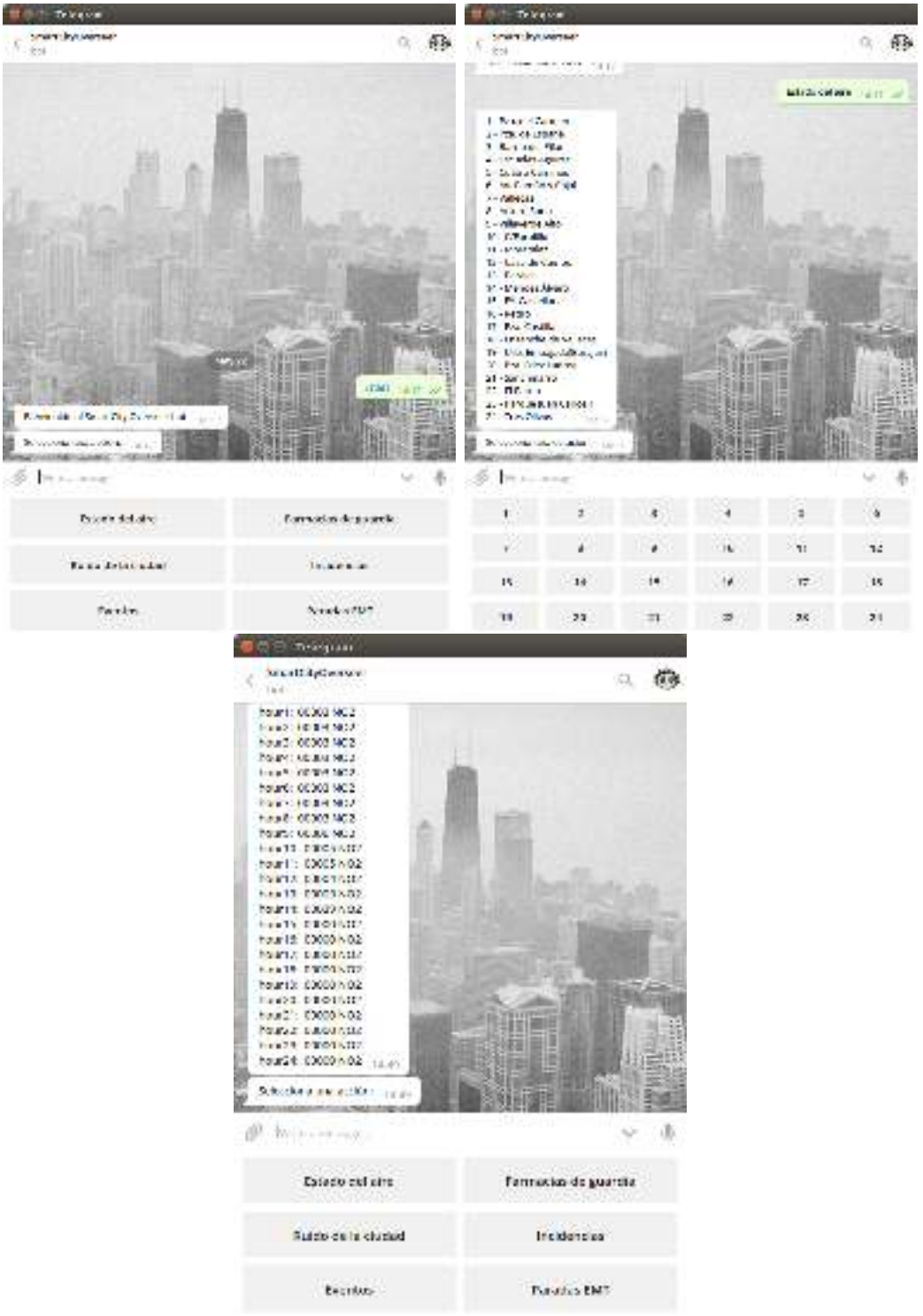


Figura 19: Interfaz Smartcity Overseer Bot

Esto es lo que a un nivel de conceptual de información abstraído sucede, pero lo que sucede internamente es lo que se explica a continuación.

El usuario accede desde el móvil a Telegram y añade el bot como contacto a través de @smartcityoverseer_bot. Una vez agregado el bot, el usuario podrá consultar los datos abiertos de Madrid de una manera sencilla y sin mucha complicación.

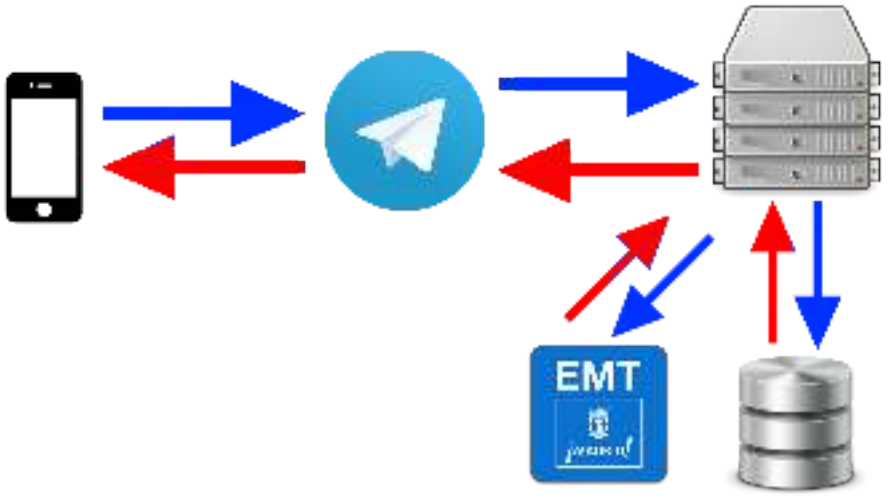


Figura 20: Caso de uso Smartcity Overseer Bot

Lo que sucede durante la consulta de datos envíada por el usuario es que se genera una petición al servidor donde está ubicado propiamente el bot a través del sistema de bots de Telegram. Una vez generada la petición, se establece la conexión con el servidor donde está alojado el bot, programado en Python, y éste a su vez se comunica con el servidor de la base de datos a través de otra petición o a la EMT para resolver los datos pedidos por dicho usuario. Una vez resueltos, estos datos se devolverán al usuario por medio de Python y la botnet de Telegram (Figura 20).

Caso B: Añadiendo una capa

Pongámonos en el caso de que ha salido una nueva fuente de datos que queremos introducir en el panel de uno de nuestros usuarios para que esté disponible su visualización y les facilite el trabajo.

En primer lugar el usuario deberá hacer login en la aplicación e irse

a su menú de capa donde encontrará la funcionalidad de añadir una nueva capa.

Una vez dentro de este apartado (Figura 21), elegiremos el fichero o URL correspondiente a la capa a insertar, el nombre que le queremos dar a la capa dentro de la aplicación, la colección a la que se tiene que añadir, marcar si es una capa histórica y su icono.

Figura 21: Añadir una capa

Después de añadir esta capa iremos a la opción de Gestionar permisos donde podremos asignarle la capa a un rol específico (Figura 22).

Lo que sucede durante este caso, es que, mediante las acciones que hacemos con la vista, se llama al controlador el cual hará las peticiones pertinentes a la base de datos. A la hora de añadir la capa, se creará o se modificará un fichero con el formato que se especifica en el Anexo II y que hemos tratado anteriormente en el capítulo de Arquitectura. Una vez tengamos la capa creada y disponible ya nos aparecerá la vista actualizada y podremos modificar el rol para añadirle la capa, que se insertará en la base de datos como disponible para dicho rol.

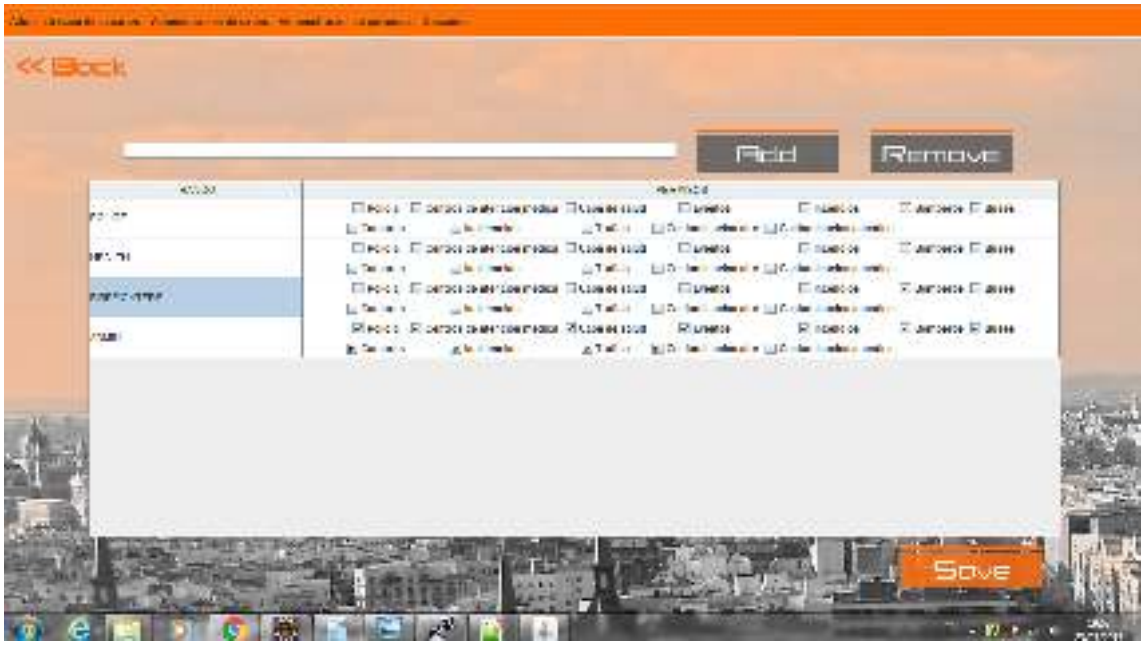


Figura 22: Gestionar permisos de usuarios

Caso C: Solucionando un accidente de tráfico

Pongámonos en el caso de que se ha producido un accidente en una carretera de Madrid. En el accidente han resultado heridas varias personas y la policia tiene que controlar la circulación de dicha carretera para que no se produzca un atasco.

Para ello, accederá al panel de control y se logueará en la aplicación. Activará la capa de Incidencias en la vía pública y allí verá todas las incidencias que suceden en Madrid, entre ellas, la incidencia mencionada. Hará click en la incidencia y, a continuación, saldrá la información de la incidencia y una vista de las comisarías más cercanas al incidente en función de la lista de efectivos que se tenga en cada una. Hará click en el botón para mandar un efectivo al incidente y, inmediatamente, aparecerá representado el efectivo en el mapa y la ruta más rápida al incidente (Figura 23).

Una vez el efectivo haya llegado al accidente, la incidencia se dará como resuelta y desaparecerá la ruta indicada por el efectivo.

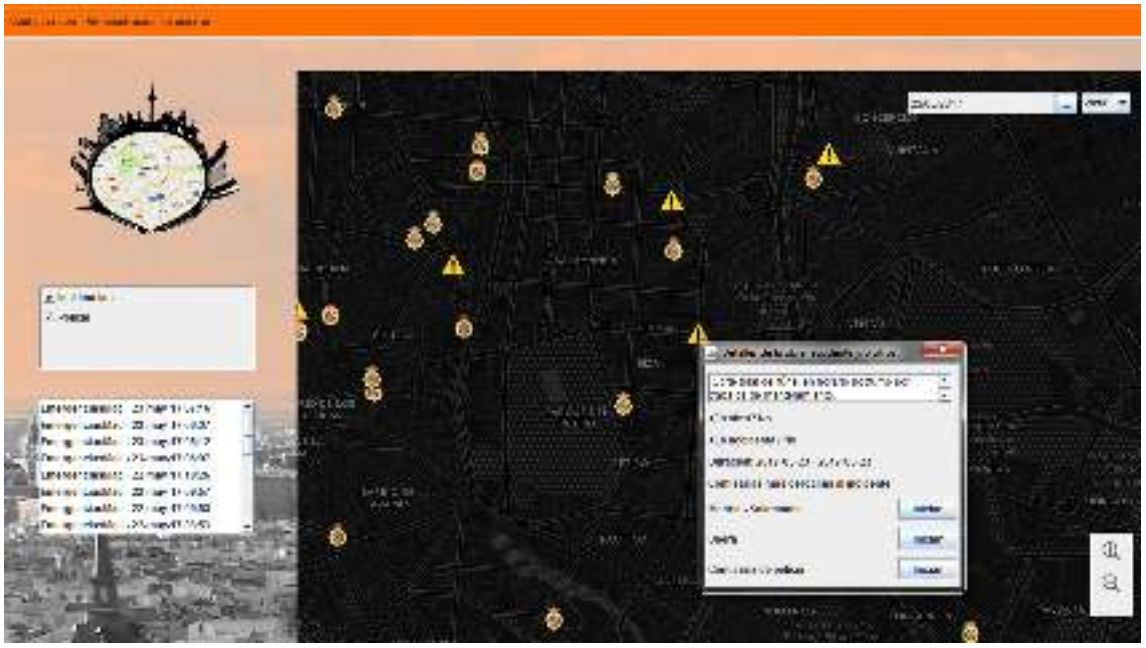


Figura 23: Localizando la incidencia

Lo que sucede durante este caso, es que, mediante las acciones que hacemos con la vista y se llama al controlador, el cual hará las peticiones pertinentes a la base de datos. A la hora de mandar los efectivos, se actualizará el estado de dicho efectivo en la colección vehicles que se muestra en el apartado de Diseño de la base de datos en el capítulo de Arquitectura. Una vez que el efectivo llegue al accidente, se volverá a actualizar el estado del vehículo a disponible para que la próxima incidencia que surga cercana a la zona pueda ser atendida por el mismo (Figura 24).

En un futuro, esta funcionalidad se podría actualizar con el GPS de los coches para ver su movimiento en el mapa.

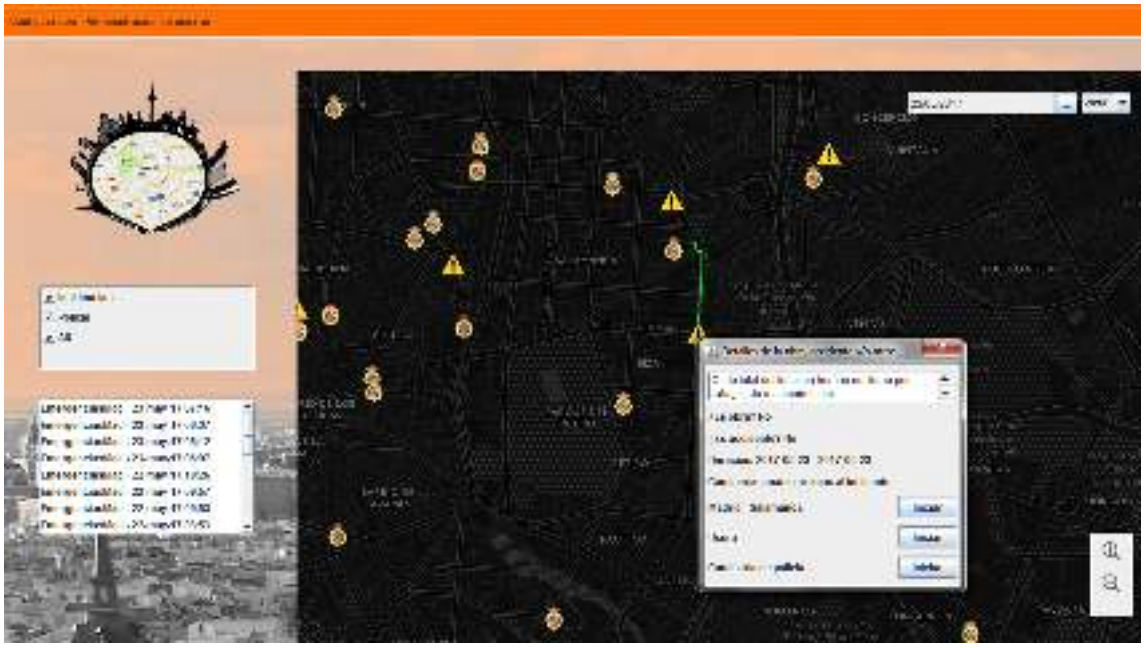


Figura 24: Mandando efectivos

8. Rendimiento del sistema

En este capítulo vamos a detallar aspectos técnicos del rendimiento de los servidores montados en Cloud respaldándolo con ayuda de gráficas obtenidas a través de un servicio de Amazon Web Services llamado CloudWatch.

Amazon CloudWatch es un servicio de monitorización de recursos y aplicaciones que se ejecutan bajo la infraestructura de Amazon. Para el tema que nos concierne, lo interesante de este servicio es que recopila y realiza el seguimiento de métricas, lo cual nos permite determinar el tráfico de salida y de entrada a través de la red y la utilización de la CPU de las dos máquinas virtuales sobre las que se sustenta la infraestructura Cloud montada.

Huelga decir que estas máquinas estaban hospedadas en una máquina local que virtualizaba los servicios y que al principio era más que suficiente pero con el tiempo se vio que era necesario que esta insfraestructura estuviera en la nube porque facilitaba mucho el testing ya que el panel de control y la infraestructura que lo pobla están desacopladas y el mantenimiento de la misma es mucho más sencillo. Por ello, a finales de Marzo migramos los servidores a la nube y esto quedará reflejado en las gráficas que se mostrarán a continuación.

Además, las gráficas que mostramos del servidor de procesamiento acaban en el mes de Mayo, fecha en la que decidimos tener apagada dicha máquina por cuestiones de hacer más sencilla la tarea de testeo de la aplicación del panel de Control.

El servidor de procesamiento, al cual denominaremos tfg-processing, es el servidor que se encargaba de obtener los datos abiertos de Madrid y del portal de datos abiertos de la EMT, procesarlos y almacenarlos en la base de datos. Este servidor actualiza periódicamente cada 5 minutos.

En cuanto al tráfico de entrada, vemos que en el mes de Marzo y Abril, el tráfico de entrada a través de la red a dicha máquina es mínimo ya que coincide con el periodo en el que se montaron y configuraron las máquinas y no es hasta el mes de Abril cuando empieza a

Figura 25: Tráfico de entrada del servidor de procesamiento a lo largo del proyecto

haber un tráfico de datos constante que oscila en torno a los 10 megas y que presenta picos de más del doble en algunos días que coinciden con los días en los que nosotros quedabamos para trabajar.

Figura 26: Tráfico de salida del servidor de procesamiento a lo largo del proyecto

En cuanto al tráfico de salida, la cantidad en megas de información que sale de la máquina oscila de manera constante en torno a los 5 megas y hasta de una máximo de 20 megas.

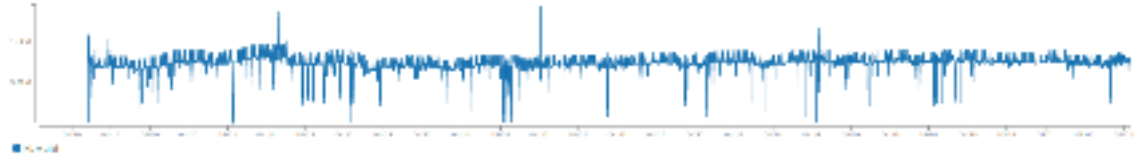


Figura 27: Tráfico de entrada del servidor de procesamiento en detalle

Visto esto a una pequeña escala, mostramos dos gráficas en las que como se puede ver la demanda que hay que satisfacer gira en torno a 5 megas por cada actualización que hace el servidor. Haciendo un cálculo aproximado, sabiento que actualiza cada 5 minutos, nos damos cuenta que actualiza 288 veces al día y que cada vez que actualiza consume 5 megabytes aproximadamente en el tráfico de entrada y de salida. Esto es en total, 1,440 gigabytes al día y en torno a 43,2 gigabytes al mes.

En cuanto a la utilización de la CPU, como vemos en la gráfica, de media se consume el 5 % de utilización de la CPU.

El servidor de base de datos, al cual denominaremos tfg-db, es el servidor donde se encuentra el gestor de bases de datos NO-SQL Mon-

Figura 28: Tráfico de salida del servidor de procesamiento en detalle

Figura 29: Utilización de CPU del servidor de procesamiento en detalle

goDB. Es el servidor del que el panel de control obtiene la información que presenta al usuario final de la aplicación.

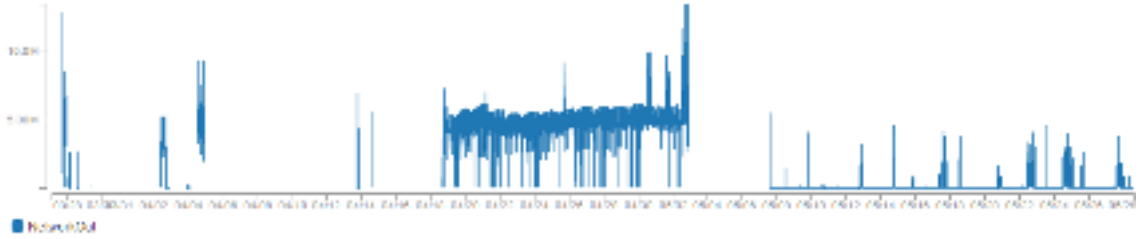


Figura 30: Tráfico de salida del servidor de bases de datos a lo largo del proyecto

En cuanto al tráfico de salida, la cantidad en megas de información que sale de la máquina en el mes de Abril es en torno a 5 megabytes. Esto es debido a que el panel de control en sus primeras versiones tenía todas las capas activadas, es decir, que prácticamente toda la información de las fuentes de datos que el servidor de procesamiento obtenía de los datos abiertos se mostraba integra en el panel. Con el desarrollo del sistema de rangos, la cantidad de información que descarga el panel de control para su presentación es selectiva y por tanto menor, siempre y cuando no actives todas las capas del panel de control.

En cuanto al tráfico de entrada del servidor de la base de datos, es igual al tráfico de salida del servidor de procesamiento.

En cuanto a la utilización de la CPU, como vemos en la gráfica, de media consume el 5 % de utilización de CPU y presenta picos que coinciden con la utilización del bot de Telegram.

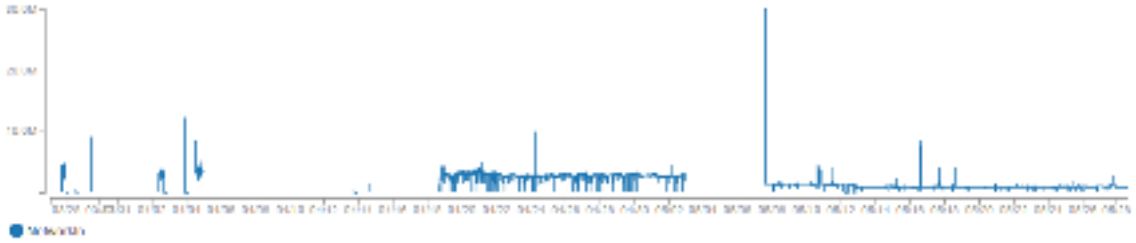


Figura 31: Tráfico de entrada del servidor de bases de datos a lo largo del proyecto

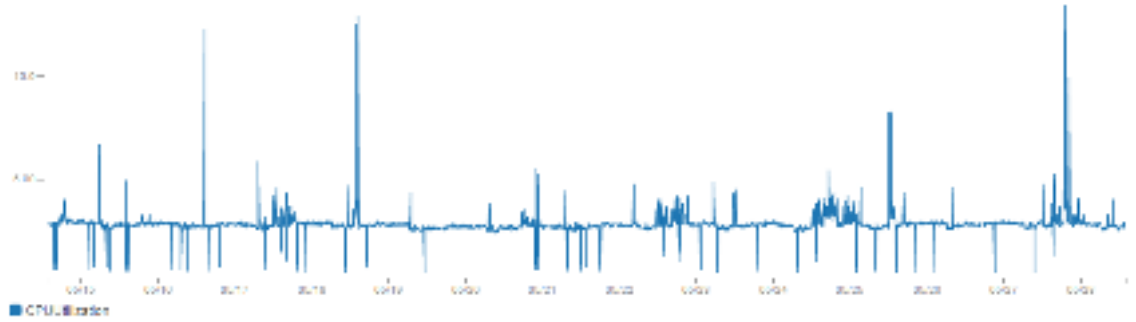


Figura 32: Utilización de CPU del servidor de bases de datos

9. Conclusiones

El objetivo principal del Trabajo de Fin de Grado es, tal y como se ha explicado en el presente documento, la creación de un panel de control para toda la ciudad que por un lado permita la reacción rápida de los diferentes roles que se encargan de la seguridad y la salud de la ciudad para poder mejorar la atención al ciudadano, y una disposición de parte de los datos abiertos de Madrid que son de interés a cualquier usuario a pie de calle mediante el bot de Telegram.

Actualmente existen sistemas para comprobar algunos servicios de los datos abiertos de Madrid, como pueden ser las paradas de la EMT mediante aplicaciones móviles, o aplicaciones para ver la contaminación del aire, pero ninguna aplicación proporciona este uso a un nivel rápido de mensajería ni a un panel de control mediante el que se puedan tomar respuestas a las incidencias.

Para todo esto hemos desarrollado Smart City Overseer , una aplicación que permite el control de todas las incidencias de la ciudad y con la que podremos aumentar dicho control de los datos según este sector de las Smart Cities vaya creciendo.

Por otro lado hemos aprendido bastante en cuanto gestión de servidores cloud, ya que gran parte de nuestro proyecto se centra en el hecho de sustentar una infraestructura de red que sirva de base para las aplicaciones desarrolladas y, en este caso, Amazon Web Services ha sido un proveedor de recursos bajo demanda para el alojamiento de las máquinas en el que hemos tenido que aprender a trabajar con bastante soltura e instalar todo lo necesario.

El desarrollo de todo el proyecto ha sido tremendamente satisfactorio, debido a que muchas de las tecnologías que hemos usado no las conociamos cuando empezamos como puede ser Python, mientras que también hemos podido profundizar más en otras cómo por ejemplo en Java.

Conclusions

The main purpose of this project is, as it has been explained in the current document, the creation of a control panel for all the city that, on the one hand, allows the quick reaction of different roles that manage the security and healthcare of the city to improve the customer's service, and the provision of a part of Madrid's open data through telegram bot, which are interesting for any common user.

Nowadays, there are systems to check some of the services of Madrid's Open Data, such as the EMT stops through mobile applications, or other ones to see the air quality index, but no application supplies neither this quick-level messenger service usage nor a control panel through which answers to the incidents can be made.

For all this, we have develop Smart City Overseer , an application that allows the control of all incidents of the city and with which we will increase the control of the data as this sector of the Smart cities grows.

On the other hand, we have learnt enough about the cloud servers' management since a great part of our project focuses on the fact of it supporting a network infrastructure used as a developed-application basis and, in this case, Amazon Web Service has been an under-demand resource supplier for the lodging of the machines in which we have had to learn to work fluently and to install all the necessary.

The development of all the project has been very satisfactory, because, when we started, we didn't know many of technologies used, such as Python, and meanwhile we have been able to delve deeper into others, as for example in Java.

10. Ampliación futura

Por la modularidad de Smart City Overseer se pueden incrementar el número de servicios que se proporcionan a los usuarios.

1. Conexión con bases de datos de aplicaciones móviles.

Con el tremendo desarrollo de las tecnologías móviles, para cierto contraste de información a la hora de gestionar la activación de la ciudad, se podrían usar conexiones con bases de datos externas de manera que contrastemos información de diferentes sucesos, como por ejemplo Tomtom. Esto aportaría una mayor fiabilidad a las rutas y eventos destacados a la hora de ampliar la aplicación.

2. Soporte de seguimiento de vídeo en tiempo real.

Con el crecimiento del uso de las cámaras en tiempo real por parte de ciertos operativos de policía o bomberos, el poder conectar estas cámaras en tiempo real a la aplicación podría llevar a una mayor disponibilidad de información al rol correspondiente y de esta manera un mayor control de las unidades necesarias para cumplir una cierta tarea.

3. Conexión con el servicio de BiciMAD.

A medida que nuestro proyecto iba creciendo, nos dimos cuenta de que iban apareciendo nuevas fuentes de open data en Madrid, este fué uno de los casos, el servicio de BiciMAD, encargado del seguimiento de las estaciones de bicis, y el seguimiento de las mismas está comenzando a dejar algunos datos abiertos para su control. En un futuro, estos datos podrían integrarse de manera que se tenga una capa de control de las mismas e incluso que se pueda conectar con el bot de Telegram.

4. Seguimiento por GPS en tiempo real de la Policía.

Actualmente se están empezando a mejorar muchos campos de manera que si se pudiera realizar un seguimiento de las patrullas por GPS se podría añadir una capa de seguimiento de las patrullas por GPS para ver su estado y ruta, Esto proporciona un mayor control y la posibilidad de mejor reacción a la hora de enfrentarse a un incidente.

11. División del trabajo

A continuación se procede a detallar las tareas que han realizado cada uno de los miembros del equipo durante el proyecto Smart City Overseer .

Durante todo el proyecto se ha seguido una distribución del trabajo equitativa, donde todos los miembros del grupo han realizado la misma carga de trabajo y de manera que los integrantes del grupo conozcan el proyecto en toda su completitud. Con esta distribución de trabajo facilitabamos la toma de decisiones y el trabajo mediante una metodología ágil de manera que los integrantes del grupo estuvieran en un constante contacto y el desarrollo de Smart City Overseer fuera mediante un conocimiento continuo del sistema.

Christian González Jiménez

Christian durante la etapa de defición aportó ideas, fuentes de datos y ayudó a la definición de funcionalidad de la aplicación.

Ha colaborado en la creación de los conversores que posteriormente se separarian en el servidor de procesamiento, más concretamente se encargó del desarrollo del conversor en Python de KML a XML. Prosiguiendo con el servidor de procesamiento, realizó el fichero de descarga de las fuentes de datos que luego usarían los conversores posteriormente.

Respecto al funcionamiento de los datos de la EMT, ha trabajado colaborativamente para conseguir tratar los datos que necesitarían posteriormente la aplicación y el bot.

Ha colaborado en la instalación de los servicios de la Base de Datos que posteriormente se separaría en el servidor de Base de datos, más concretamente se ha encargado de parte de la configuración del MongoDB y la creación de la Base de datos.

Además respecto a Amazon Web Services se encargó de la creación de las máquinas virtuales y la configuración de la red de dichas máquinas.

Ha colaborado en la creación del Bot de Telegram @smartcityoverseer_bot mediante la creación de la interfaz del bot.

Ha colaborado en el desarrollo de la aplicación en lo relacionado a la gestión de capas en segundo plano y en la adecuación de la información de las capas para que puedan ser visualizadas correctamente en dialógos de la aplicación y, de manera conjunta, la creación del mapa de la ciudad.

También ha estado trabajando desarrollando el sistema de implementación de rutas de la aplicación, el parseador de plugins que permite la inclusión de capas de manera automática gracias a la inclusión de las capas customizadas que puede agregar el administrador y el diseño de los recursos gráficos que se usan en la aplicación haciendo uso de brushes y de shaders tematizados en el Tom Clancys: The division.

Ha colaborado también en la redacción de la memoria, en la búsqueda de recursos que apoyen los datos y en especial en la redacción del punto de tecnologías. El resto de puntos de esta memoria han sido redactados conjuntamente y, posteriormente, revisados por cada uno de los integrantes del grupo.

Javier Villarreal Rodríguez

Javier durante la etapa de defición aportó ideas, fuentes de datos y ayudó a la definición de funcionalidad de la aplicación.

Ha colaborado en la creación de los conversores que posteriormente se separarian en el servidor de procesamiento, más concretamente se encargó del desarrollo del conversor en Python de CSV a XML. Prosiguiendo con el servidor de procesamiento, realizó el fichero de configuración de la inserción de las capas por defecto que usará la aplicación y el bot.

Respecto al funcionamiento de los datos de la EMT, ha trabajado colaborativamente para conseguir tratar los datos que necesitarían posteriormente la aplicación y el bot.

Ha colaborado en la instalación de los servicios de la Base de Datos que posteriormente se separaría en el servidor de Base de datos, más concretamente se ha encargado de parte de la configuración del MongoDB y la creación de los usuarios.

Además respecto a Amazon Web Services se encargó de la instalación del sistema y de los servicios necesarios para subir los scripts y la Base de datos ya creados.

Ha colaborado en la creación del Bot de Telegram @smartcityoverseer_bot mediante la unión de la interfaz con los servicios que ya habiamos creado anteriormente.

Ha colaborado en el desarrollo de la aplicación en lo relacionado al acceso a la base de datos, consultas, la creación del sistema de plugins, la gestión dinámica de capas haciendo uso de SFTP para acceder al fichero de configuración de capas y la inserción de los ficheros de capas cuando se trata de una capa estática y, de manera conjunta, la creación del mapa de la ciudad.

También ha trabajado desarrollado el sistema de gestión de usuarios, el sistema de permisos de la aplicación para dotar de acceso a los usuarios a sólo aquellas capas que necesita para el desempeño de su labor y en la adecuación de los recursos gráficos haciendo uso de

porcentajes que permitan que la aplicación se adapte al tamaño de la pantalla.

Ha colaborado también en la redacción de la memoria, en la búsqueda de recursos que apoyen los datos y en especial en la redacción del punto de Estado del arte. El resto de puntos de esta memoria han sido redactados conjuntamente y, posteriormente, revisados por cada uno de los integrantes del grupo.

Referencias

1. AWS - Amazon Web Services https://aws.amazon.com/es/
2. Ubuntu Server https://www.ubuntu.com/download/server
3. MongoDB https://www.mongodb.com/es
4. Tutoriales MongoDB https://www.tutorialspoint.com/mongodb/
5. MongoDB and Java https://goo.gl/i8MO0w
6. Documentación Java https://docs.oracle.com/javase/7/docs/api/
7. CSV https://goo.gl/HmiIbq
8. JSON Tutorial https://www.w3schools.com/js/js_json_intro.asp
9. Python https://www.python.org/
10. PyMongo https://api.mongodb.com/python/current/
11. PyKML https://pythonhosted.org/pykml/
12. Documentación KML https://goo.gl/UenXbP
13. XMLtoDict https://pypi.python.org/pypi/xmltodict
14. JxMapViewer2 https://github.com/msteiger/jxmapviewer2
15. Telegram https://core.telegram.org/
16. Telebot https://github.com/eternnoir/pyTelegramBotAPI
17. PyEMTMad https://pypi.python.org/pypi/pyemtmad
18. Cámaras de Madrid http://informo.munimadrid.es/informo/Camaras/
19. Nasa Fires https://goo.gl/Yf93I1
20. Incidencias de la EMT http://servicios.emtmadrid.es:8080/rss/emtrss.xml

21. Datos de incidencias de la DGT http://www.dgt.es/incidencias.xml

22. Datos Abiertos del gobierno http://datos.gob.es

23. Datos Abiertos de la Comunidad de Madrid http://datos.madrid.es/portal/site/egob

24. Twitter API https://dev.twitter.com/rest/public

25. ArcGis https://developers.arcgis.com/

26. Documentación KML https://goo.gl/ZaHDYC

Anexo I. Capas por defecto en Madrid

Capa de contaminación del aire

Con esta capa podemos ver la contaminación del aire, con las muestras más cercanas a nuestra hora tomadas o un histórico disponible gracias a la obtención de los datos de las estaciones meteorológicas de Madrid que están disponibles en el portal de datos abiertos de Madrid. En esta capa mostramos el nivel de Contaminación del NO2 que actualmente es el compuesto por el que se rigen los episodios de contaminación de Madrid.

Figura 33: Capa de aire

Capa de contaminación acústica

Con esta capa podemos ver la contaminación acústica, esta capa es similar a la del aire, debido a que son las mismas estaciones las que lo miden aunque con diferentes identificadores. Con ésta podemos ver el nivel de decibelios que se perciben constantemente mandando tres datos desde el más cercano al medio del intervalo de 24h.

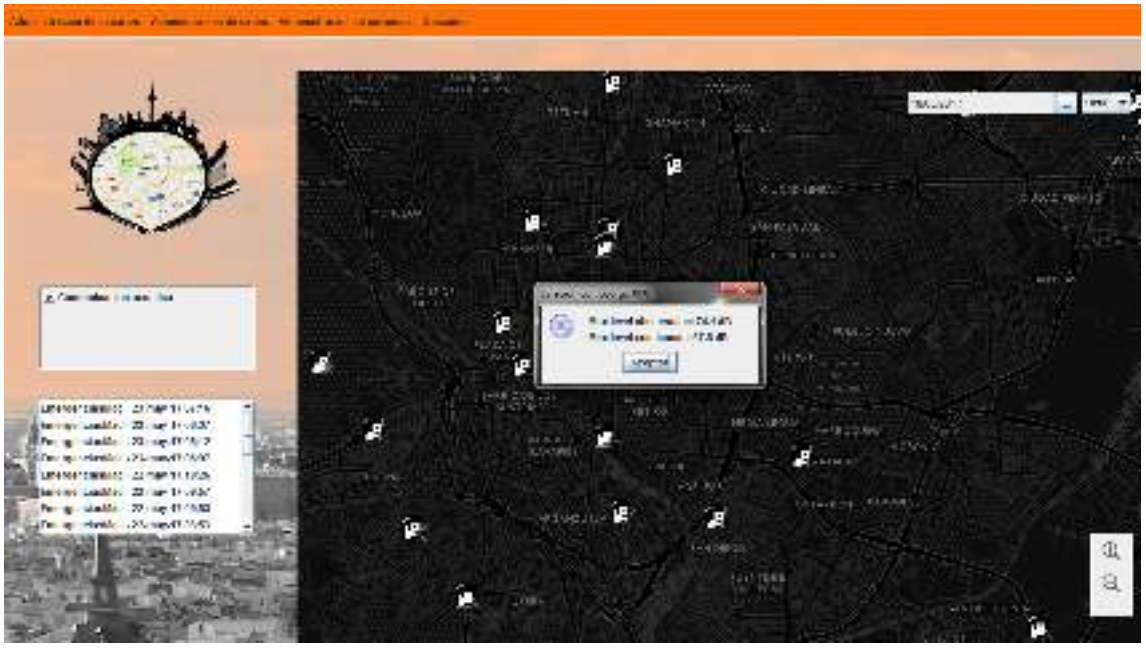


Figura 34: Capa de ruido

Capa de Eventos

Con esta capa podemos ver los eventos disponibles que están programados en el calendario de la Comunidad de Madrid. Estos datos están disponibles mediante tres ficheros en los datos abiertos que los utilizamos para crear la capa:

- Actividades Culturales y de Ocio Municipal en los próximos 100 días: este fichero se actualiza en tiempo real.
- Agenda de actividades deportivas: este fichero se actualiza diariamente.
- Actividades gratuitas en Bibliotecas Municipales en los próximos 60 días: este fichero se actualiza diariamente.

Capa de Incidencias

Con esta capa podemos ver las diferentes Incidencias que se producen en Madrid en el momento en el que se produzcan mediante el

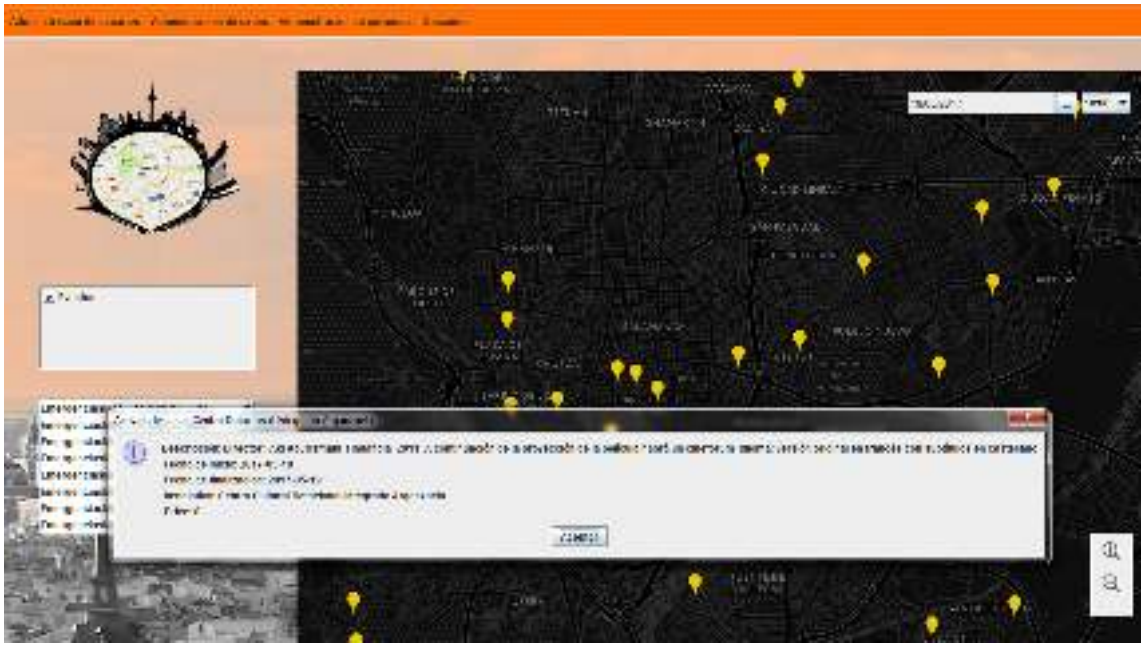


Figura 35: Capa de eventos

fichero que tiene actualización en tiempo real proporcionado por los datos abiertos de la comunidad de Madrid.

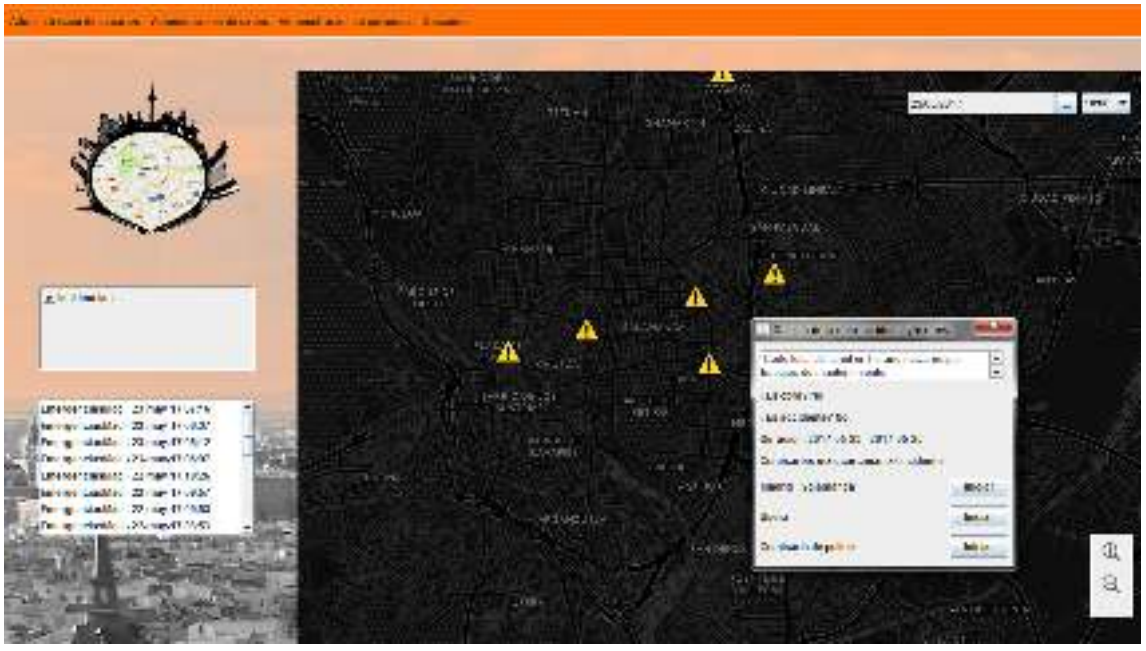


Figura 36: Capa de incidencias

Capa de Comisarías

Con esta capa podemos ver todas las comisarías que existen en Madrid. Esta capa se creó a partir de buscar en varios índices y corroborar la información mediante la búsqueda de la dirección en Google, ya que las comisarías como tal no existen en datos abiertos de Madrid.

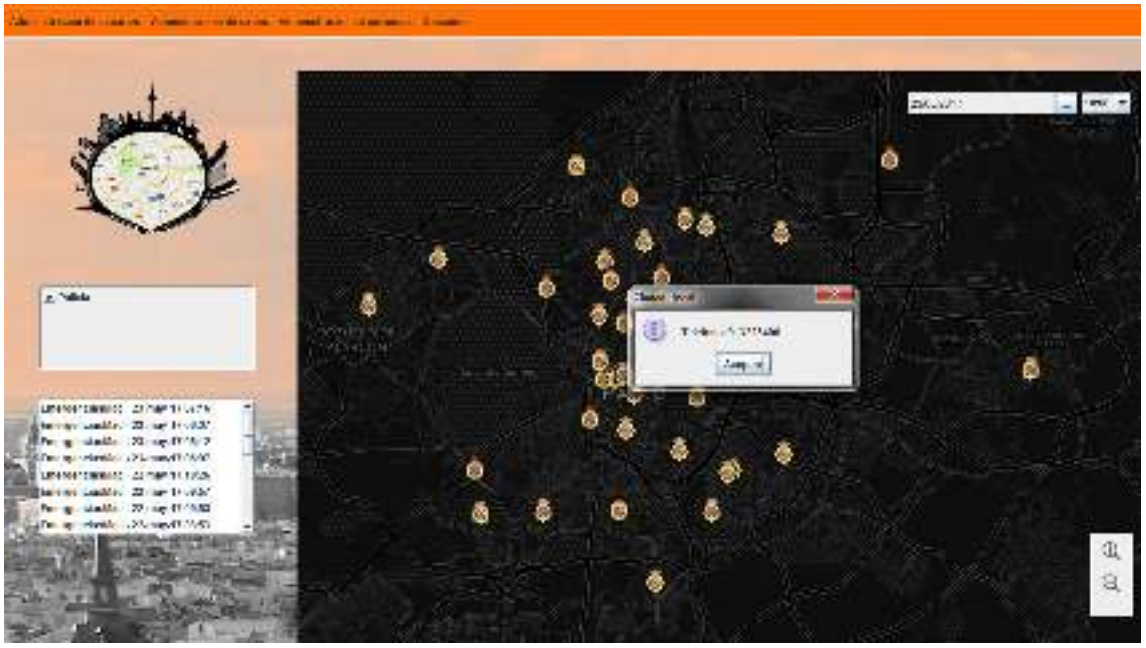


Figura 37: Capa de comisarias de policia

Capa de Parques de Bomberos

Con esta capa podemos ver los parques de bomberos de Madrid gracias a los datos abiertos de Madrid, aunque con una actualización anual, estos datos son bastante fijos y por tanto no es necesario contrastarlos.

Capa de Tráfico

Con esta capa podemos ver el estado del tráfico de Madrid en tiempo real gracias a los datos abiertos de Madrid con su conjunto de datos de Intensidad de tráfico:mapa de tramas. Con esta capa pode-

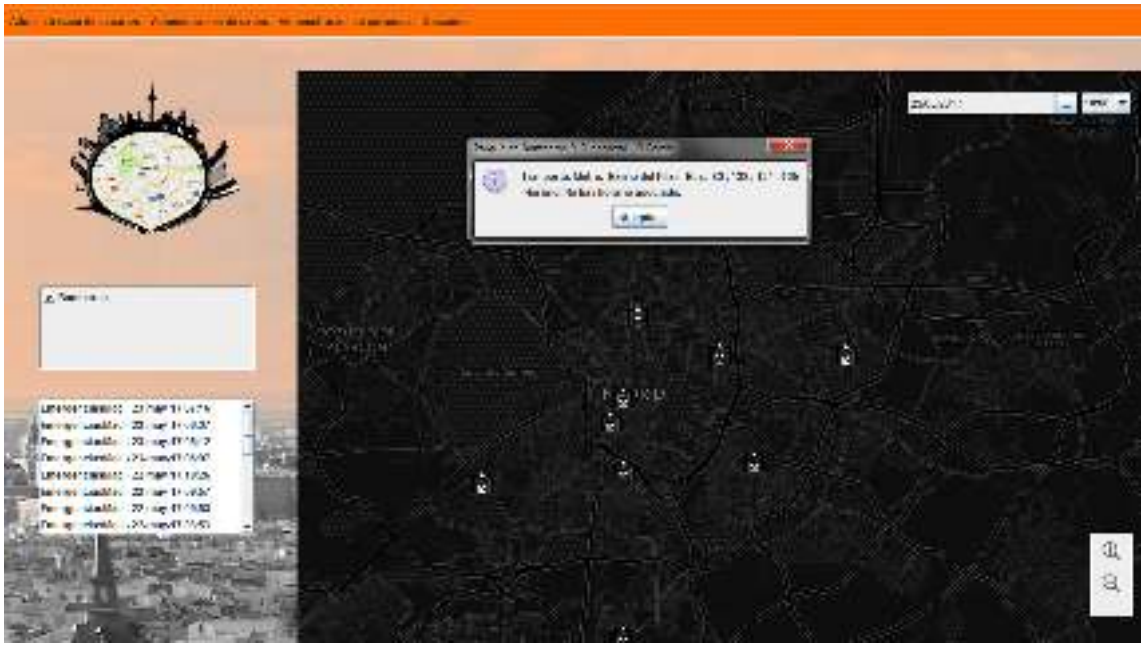


Figura 38: Capa de parques de bomberos

mos ver la intensidad del tráfico desde verde (poco tráfico) hasta rojo (congestionado).

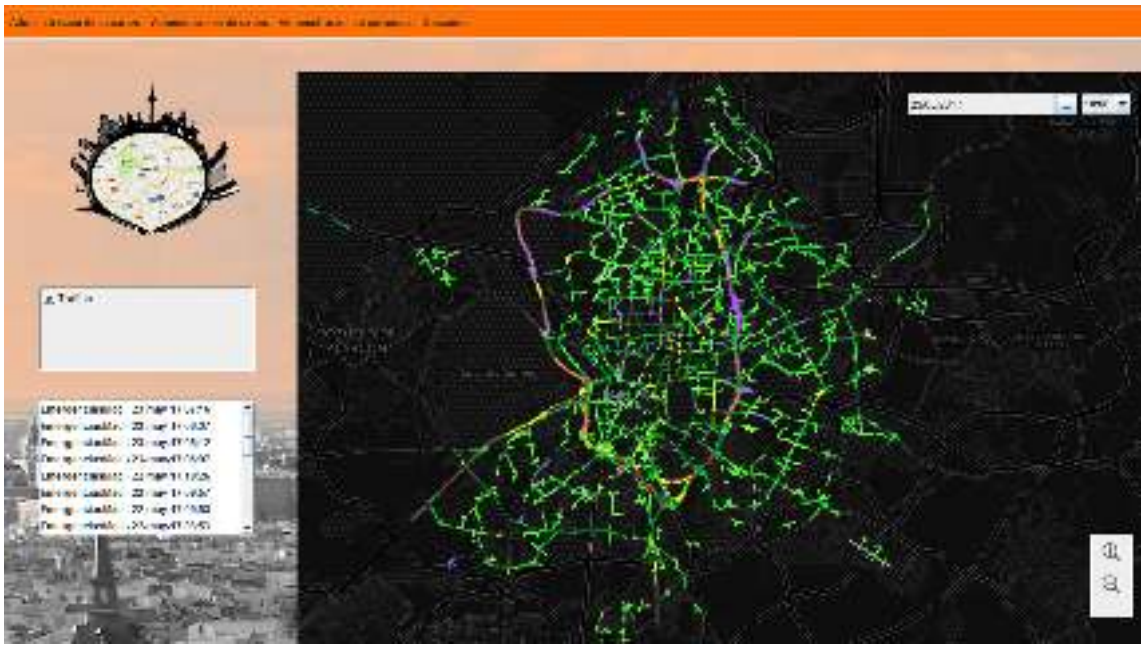


Figura 39: Capa de tráfico

Capa de Cámaras

Con esta capa podemos ver las cámaras de tráfico en tiempo real, debido a que estas cámaras se proporcionan en datos abiertos de Madrid con su ubicación y su imagen mediante la ruta donde está.

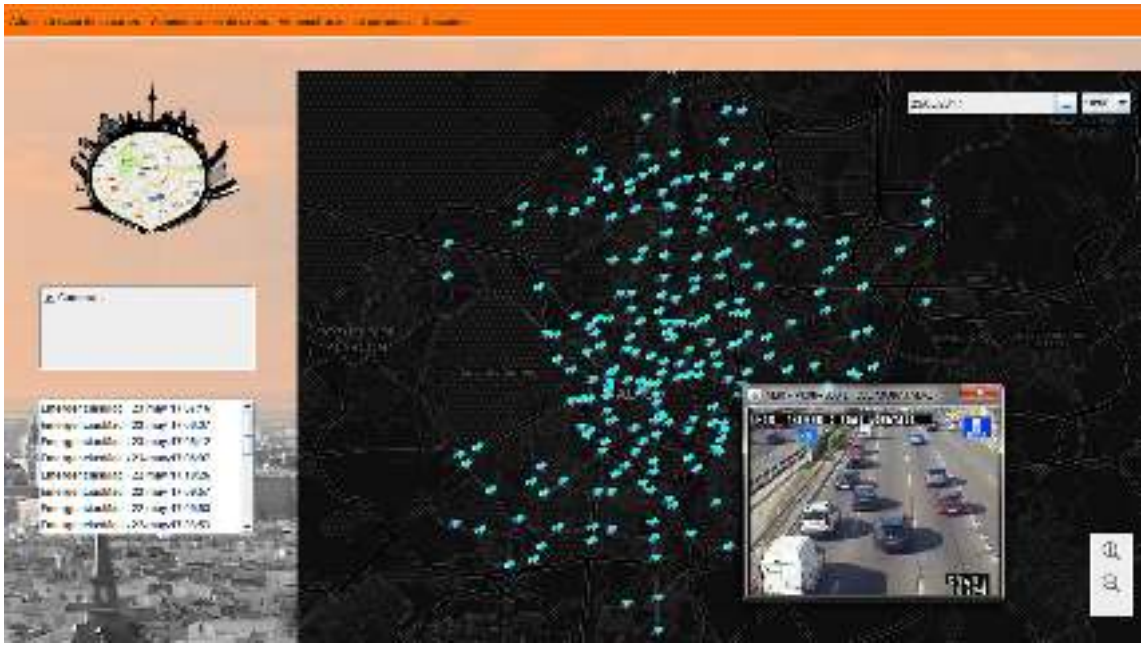


Figura 40: Capa de camaras

Capa de Incendios

Con esta capa podemos ver los incendios de Madrid en tiempo real gracias al FireWebMapper de la NASA.

Capa de Salud

Con esta capa podemos ver la salud de los distritos de Madrid, esta capa se obtiene a través de consultar las comisarías, los centros de atención médica y los parques de bomberos y usar una fórmula para estimar la salud del distrito.

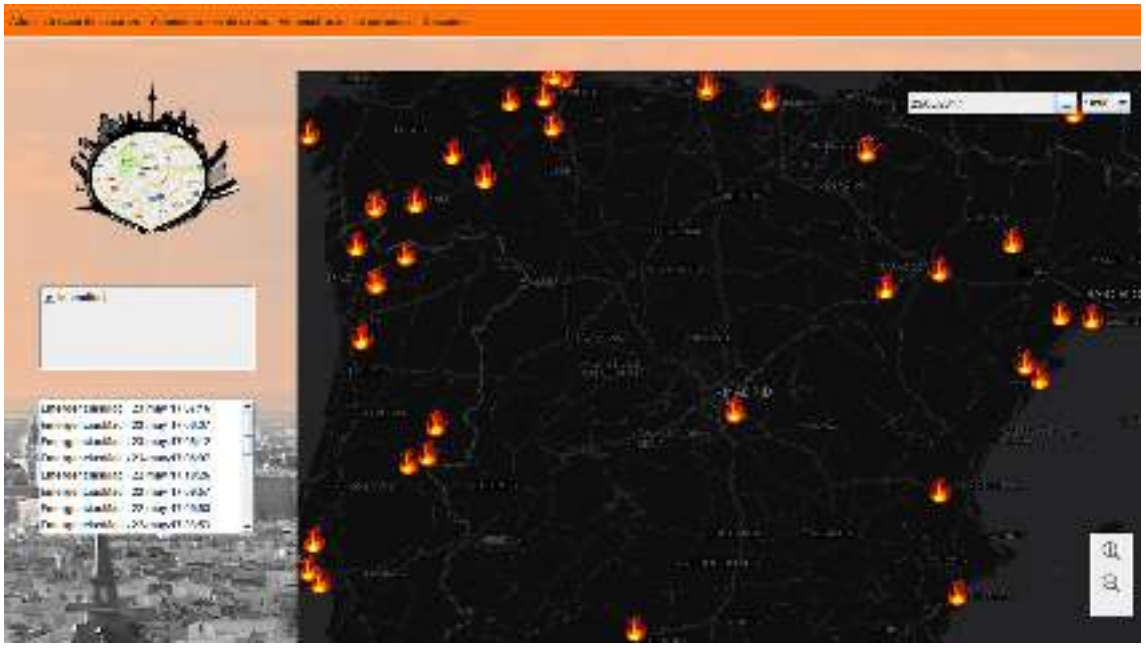


Figura 41: Capa de incendios

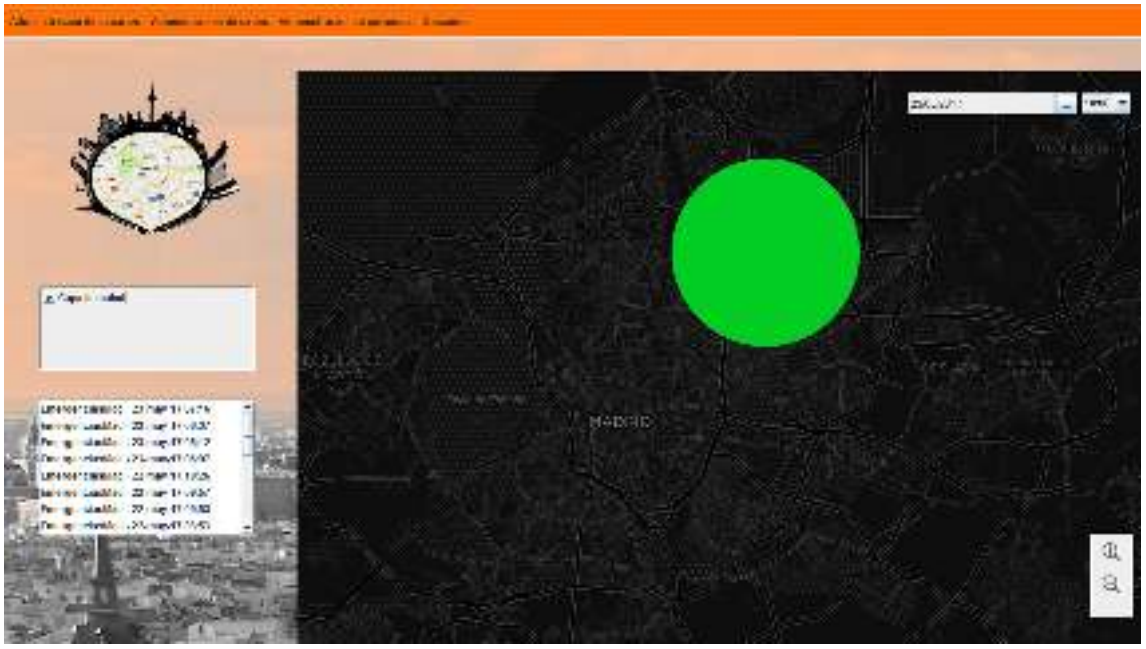


Figura 42: Capa de salud

Capa de Centros de atención médica

Con esta capa podemos ver todos los centros de salud y hospitales de Madrid actualizados en tiempo real gracias a los datos abiertos de Madrid.

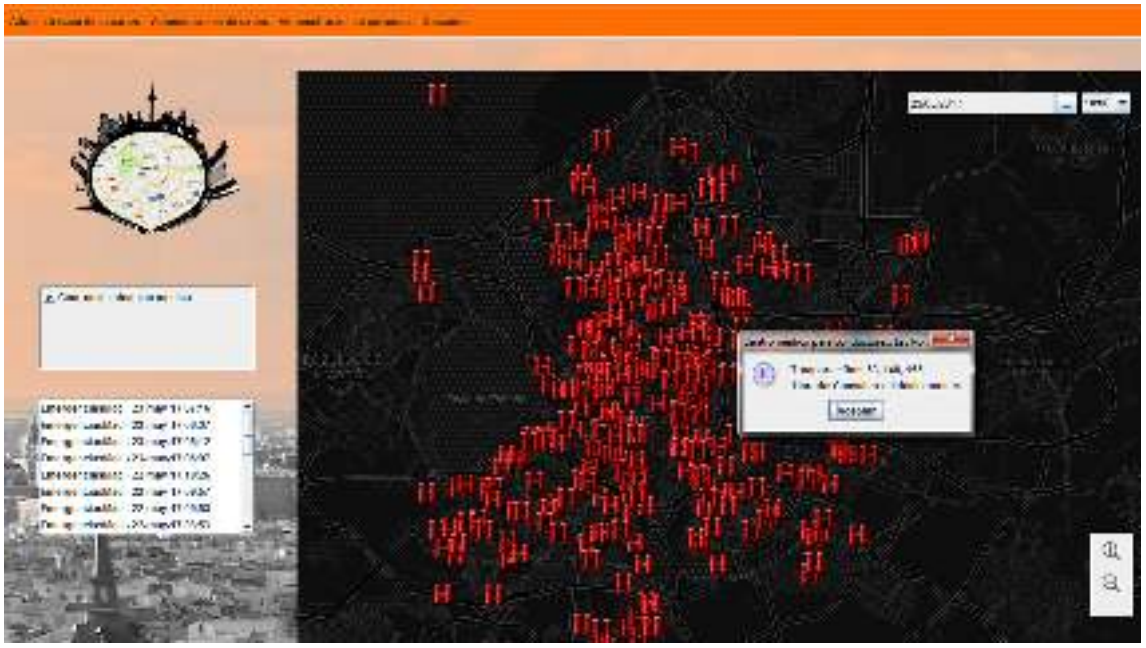


Figura 43: Capa de centros de salud

Capa de Buses

Con esta capa podemos ver toda la información de los autobuses cada vez que seleccionemos una parada, como por ejemplo el tiempo de llegada de los autobuses o las incidencias que afectan a la EMT. Esta capa se genera a través de consultas a la API de la EMT para obtener los datos en tiempo real.

Figura 44: Capa de buses

Anexo II. Configuración de las capas en el servidor

El fichero de configuración de capas. Es el fichero que contiene toda la información necesaria para descargar las fuentes de datos y almacenarlas en la base de datos. Además de esto, sirve para hacer el panel de control modular ya que la aplicación lee este fichero para comprobar el número de capas que hay actualmente en la aplicación. Los parámetros url, fname, output y type son relativos a la descarga de la fuente de datos propiamente dicha. El parámetro database corresponde con el nombre de la colección en la base de datos y el parámetros isRefresh indica si es una capa histórica o no. En función de ello, hace un tratamiento distinto a los datos.

```
[Nombre de grupo]\{

[Nombre de fuente de datos 1]

url=

fname=

output=

type=

database=

isRefresh=

...

[Nombre de fuente de datos N]

url=

fname=

output=

type=

database=

isRefresh=

...

\}
```

Anexo III. Manual del administrador

Una vez instalada la aplicación, crearemos un acceso directo en el escritorio. Acto seguido, haremos click en él. Lo que aparecerá a continuación será el sistema de login de la aplicación como veremos a continuación (Figura 45).

Figura 45: Login de la aplicación

Procederemos a introducir los datos del administrador y haremos click en el botón de Login. Una vez logueado, veremos el panel de inicio del administrador (Figura 46).

El administrador posee un panel de administración que consta de 4 bloques de gestión: “Administración de usuarios”, “Administración de capas”, “Administración de permisos y Opciones”.

Administración de usuarios

La administración de usuarios consta de dos opciones: “Añadir usuario” y “Editar usuario”.

Figura 46: Panel de inicio del administrador

Añadir usuario

Para añadir un usuario procederemos a hacer click en "Administración de usuarios" y después click en "Añadir usuario" (Figura 47).

Figura 47: Click en añadir usuario

Una vez accedemos a dicho panel, veremos que consta de tres cam-

pos: “Nick de usuario”, “Contraseña” y “Rol de usuario” (Figura 48).

Figura 48: Panel de añadir usuario

El Nick de usuario es el identificador de usuario que identifica al usuario cuando logueas en la aplicación. La contraseña es el código secreto que conoce solo el usuario al que pertenece el usuario que vamos a crear. El rol de usuario es el identificador del rango al que se le han asignado determinados permisos de capa para la ejecución de las tareas del usuario. Una vez que entendamos los datos que hay que introducir, procederemos a darle al botón “ADD”.

Si ya existe un usuario con el mismo identificador de usuario. La aplicación nos devolverá un error como el que se indica (Figura 49).

De lo contrario, se añadirá el usuario al panel de control.

Figura 49: Error al intentar añadir usuario que ya existe

Editar usuario

Para editar un usuario, procederemos a hacer click en "Administración de usuarios" y después click en "Editar usuario" (Figura 50).

Figura 50: Click en editar usuario

Una vez accedemos, veremos dos secciones claremente definidas. Por

un lado, el panel que se utiliza para “Buscar”, “Modificar” y “Eliminar usuario”. Por otro lado, la lista de usuarios que tienen acceso al panel.

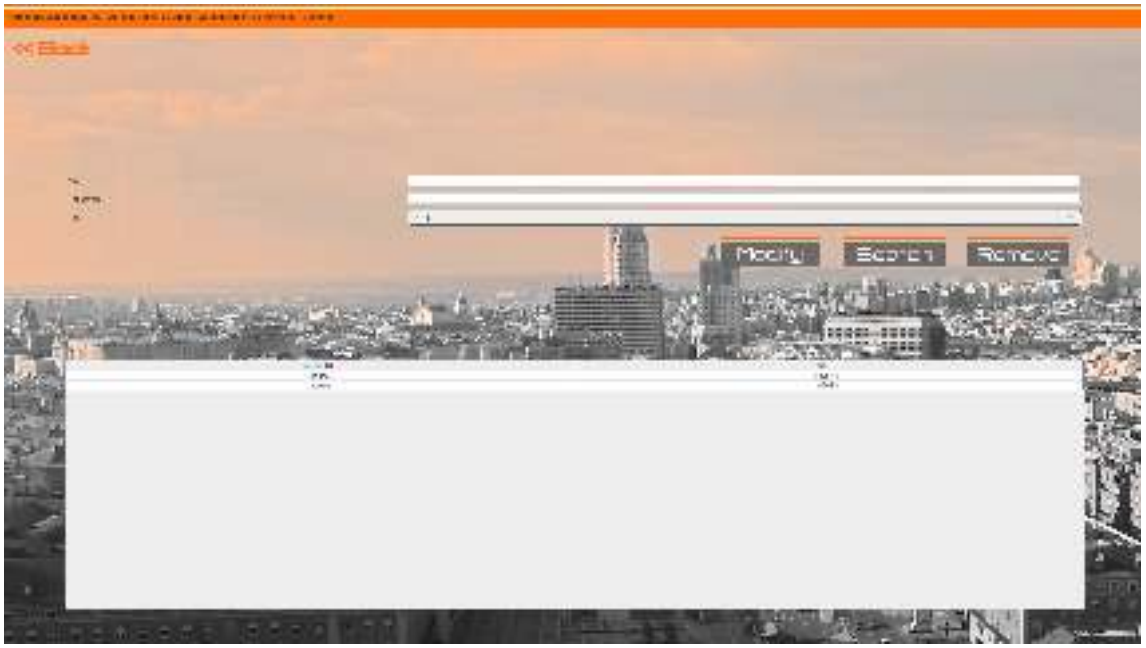


Figura 51: Panel de editar usuario

Habiendo ubicado el panel de búsqueda modificación y eliminación de usuarios, veremos que dicho panel posee los mismos campos que se encontraban en el panel de “Añadir usuario”. Esto es, porque su uso básicamente es el mismo pero se realizan distintas tareas (Figura 51). Lo primero que hay que hacer para editar un usuario, es buscarlo. Con lo cual, introduciremos el Nick del usuario y procederemos a buscarlo pulsando en el botón “Search”.

Si el usuario no existe, la aplicación nos devolverá el siguiente error indicado (Figura 52).

Del mismo modo, si queremos eliminar un usuario y no existe devolverá el mismo error. Una vez buscado, aparecerá su información en los campos designados para ello. Acto seguido, modificaremos su información y le daremos al botón de “Modify”. Si lo que queremos es eliminar un usuario, introduciremos su identificador de usuario y le daremos al botón “Remove”.

La lista de usuario que se encuentra más abajo está dividida en dos columnas. Por un lado, el Nick de usuario y por otro lado el rol del

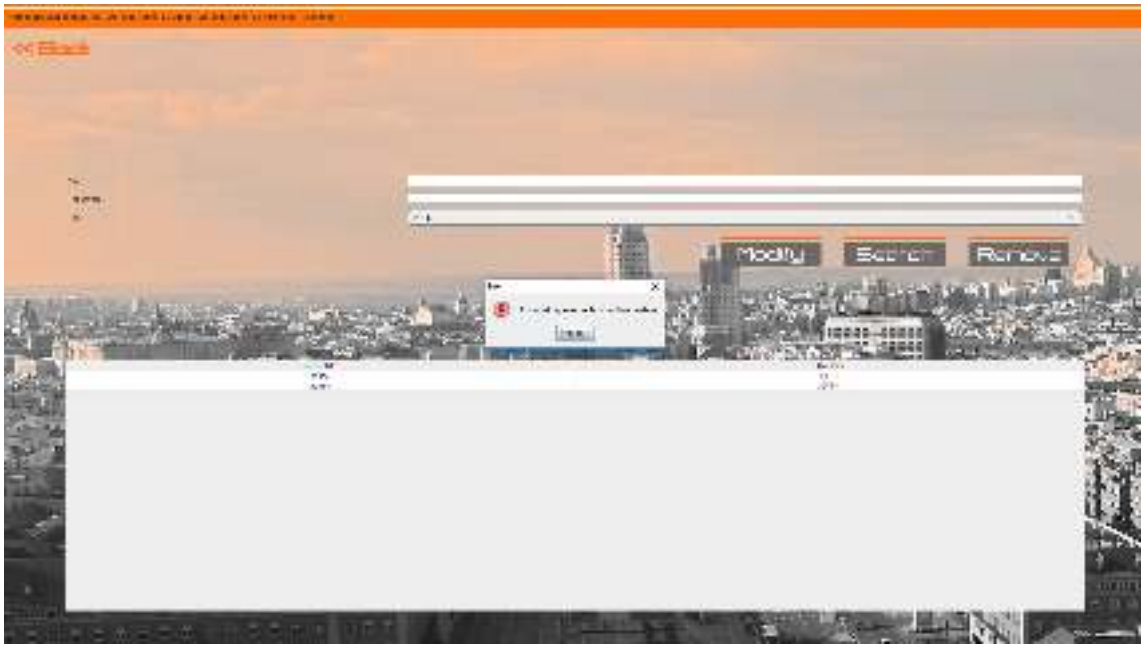


Figura 52: Error al buscar usuario que no existe

Usuario. Esto nos da a conocer la lista de usuarios que tienen acceso a la aplicación y el rango que ostentan. Por tanto, los permisos que tienen para acceder a la aplicación y a las fuentes de datos que se manejan en ella.

Administración de capas

La administración de capas consta de dos opciones: “Añadir capa” y “Eliminar capa”.

Añadir capa

Para añadir una capa procederemos a hacer click en “Administración de capas” y después click en “Añadir capa” (Figura 53).

Una vez accedemos a dicho panel, veremos que consta de seis campos: Nombre de la capa, icono de la capa, URL de la capa, Archivo, Nombre de la colección en la base de datos y un checkbox que indica si la capa es histórica o no (Figura 54).

Figura 53: Click en añadir capa

Figura 54: Panel de añadir capa

El nombre de la capa es el identificador de la capa que hace unívoca a la capa. Esto es, solo hay una capa con este identificador. El icono de la capa es la imagen que representa una entidad dentro de la capa que se quiere importar. El URL de la capa se trata de una dirección web donde está almacenada la fuente de datos que se va a añadir a la capa. El archivo de la capa se trata del fichero donde está almacenada

la fuente de datos que se va a añadir a la capa. De tal manera, que si se añade una URL la capa será dinámica, es decir, que la información se almacena de manera periódica en el tiempo o si se añade el archivo la capa será estática, es decir, que la información se almacena únicamente en ese fichero y no puede ser actualizada. El nombre de la colección en la base de datos es el nombre que tendrá la colección de la base de datos una vez se añade la fuente de datos al panel de control. Este nombre es univoco, o lo que es lo mismo, que solo puede haber una fuente de datos asociada a una colección. El checkbox indica si una capa es histórica o no. Esto es, que la información de la capa será almacenada periódicamente y será accesible mediante un selector de información localizado en el panel de inicio del mapa. Una vez introducidos los datos en el formulario, se procederá a dar al botón Add. Si ya existe una capa con el mismo identificador de usuario, se dará el error indicado (Figura 55).

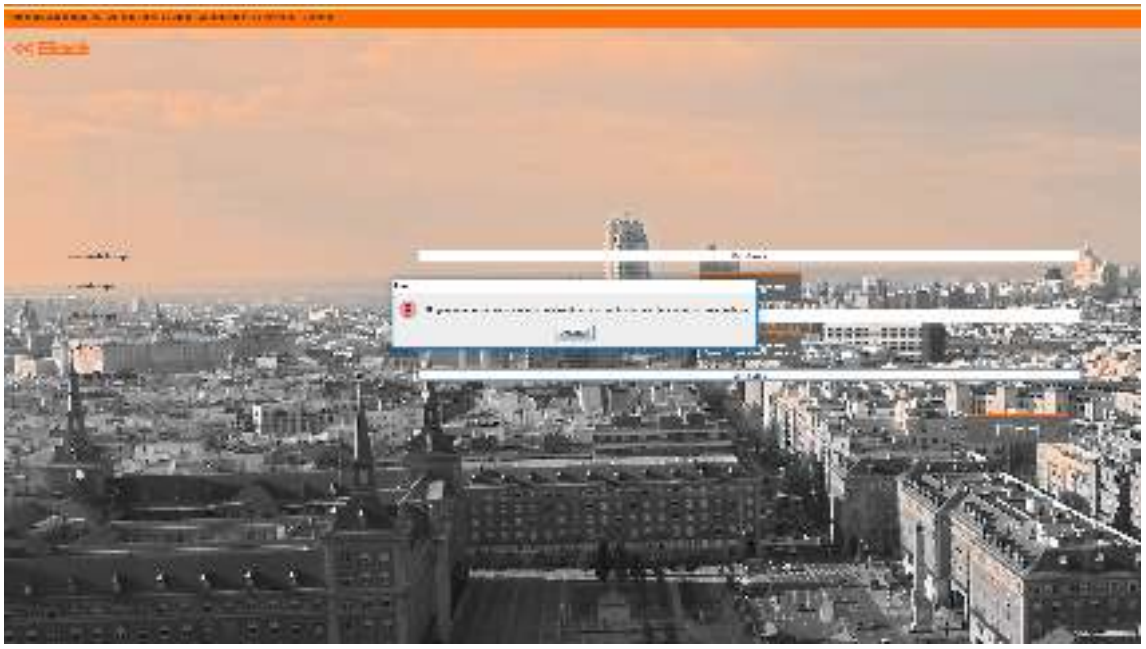


Figura 55: Error al añadir capa con identificador existente

Si ya existe una capa con el mismo nombre de la colección de la base de datos, se dará el error indicado (Figura 56).

Si no se ha introducido un icono en el formulario, se dará el error indicado (Figura 57)

Si no se ha producido ningún error, la capa será añadida al panel

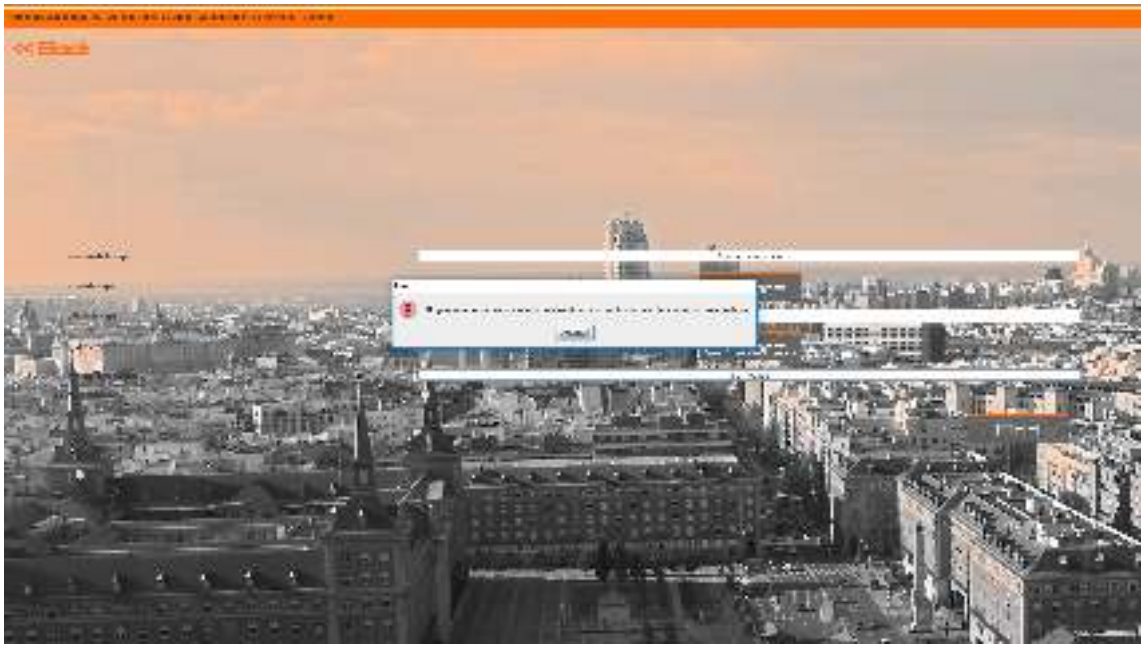


Figura 56: Error al añadir capa con nombre de colección existente

Figura 57: Error al añadir capa sin icono

y desde el mismo podrá ser asignada a cualquier rol de usuario que la necesite o bien podrá ser eliminada si la capa ya no resulta de utilidad.

Eliminar capa

Para eliminar una capa procederemos a hacer click en “Administración de capas” y después click en “Eliminar capa” (Figura 58).

Figura 58: Click en eliminar capa

Una vez accedemos, veremos que se trata de una tabla que lista todas las capas que se encuentran disponibles en el panel de control. Esta tabla consta de tres columnas.(Figura 59).

La columna Capa muestra el nombre de la capa, la columna Tipo de capa muestra el tipo de capa y la columna acciones muestra las acciones que se pueden realizar sobre la capa. En este caso, se puede eliminar la capa.

Si quieres eliminar una capa, lo único que hay que hacer es ubicarla en la capa y seleccionar el botón “Remove” en la Columna asociada a las acciones de la capa.

Si se trata de una capa del core, te dará el error indicado (Figura 60).

Si hay usuarios que tienen asociada dicha capa en sus permisos de rango, dará el error indicado (Figura 61).

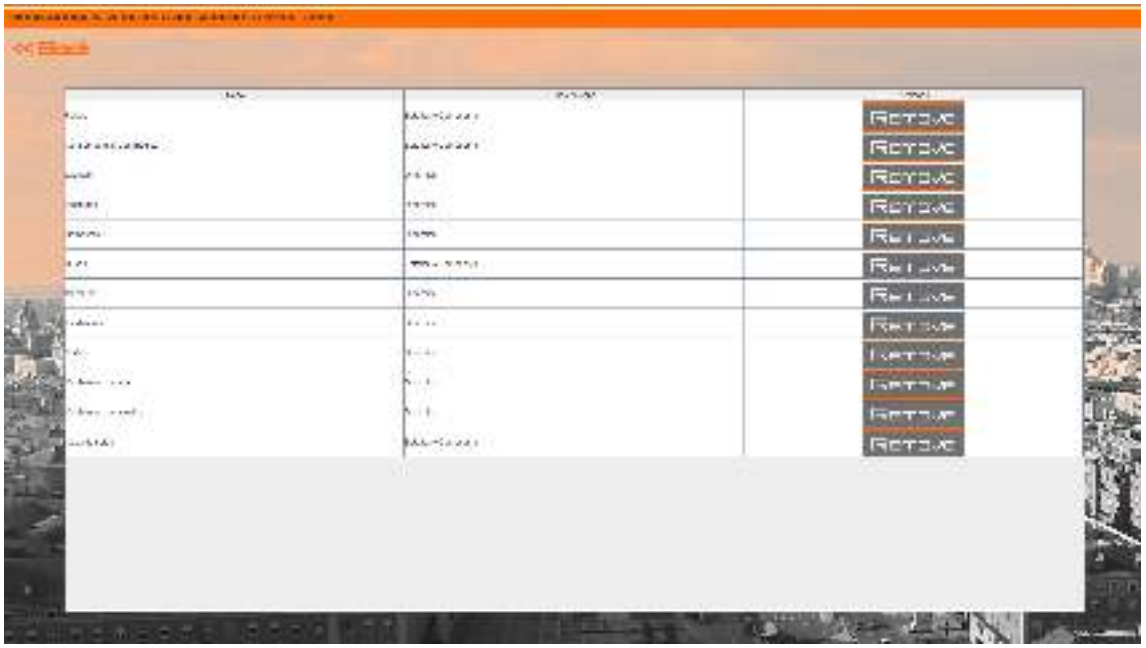


Figura 59: Panel de eliminar capas

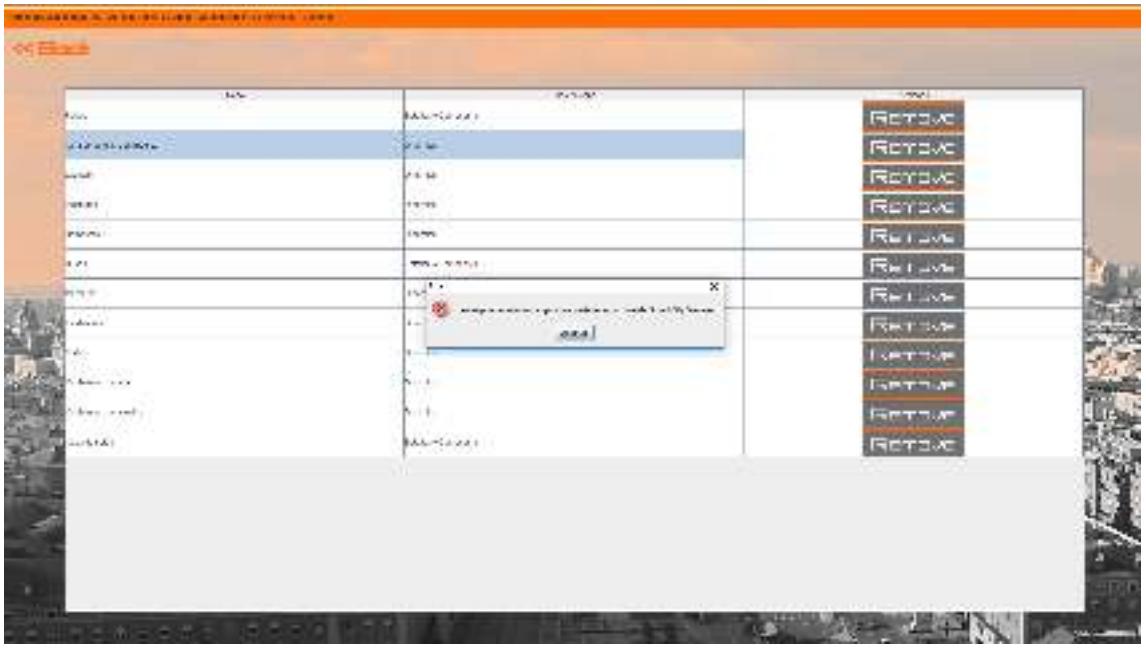


Figura 60: Error al eliminar una capa del core

Si no da ningún error, se eliminará la capa y se harán las correspondientes actualizaciones en las vistas.

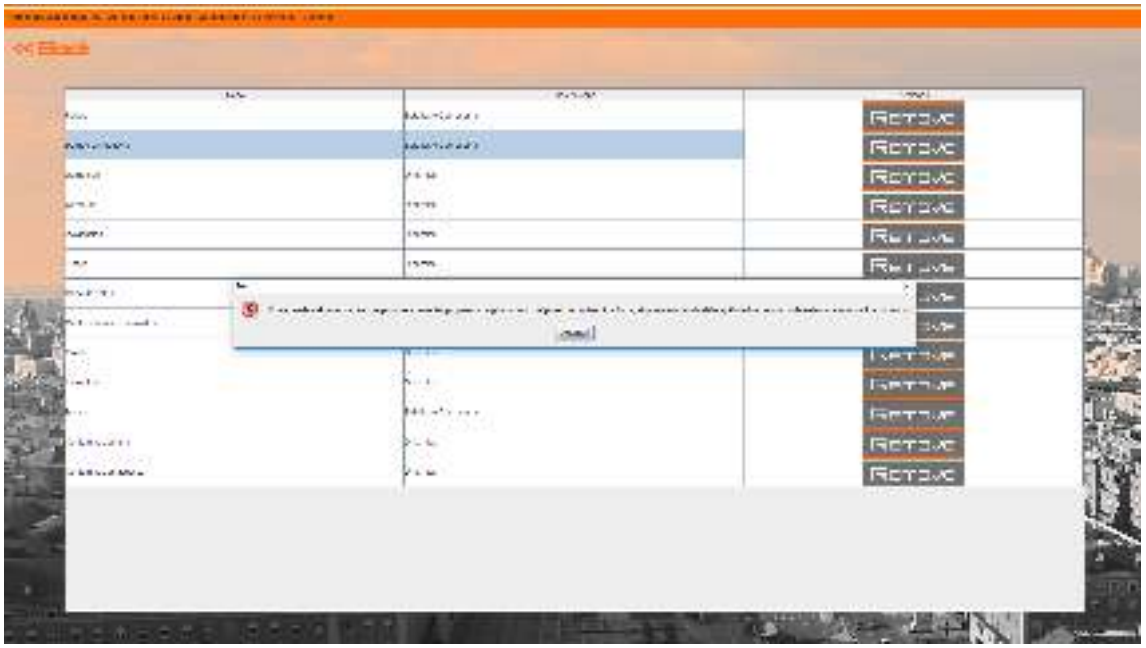


Figura 61: Error al eliminar una capa que está asignada a un usuario

Administración de permisos

La administración de permisos consta de dos opciones: Añadir rango y modificar permisos.

Añadir rango

Para añadir rango procederemos a hacer click en Administración de permisos y después click en "Gestionar permisos" (Figura 62).

Una vez accedemos a dicho panel, veremos que consta de dos partes claremente diferenciadas. Por un lado, un campo y un botón desde donde se podrán añadir rangos. Por otro lado, una tabla en la que se pueden gestionar los permisos de los diferentes roles de usuario (Figura 63).

Para añadir un rango, tenemos que introducir el nombre del rango en el campo asociado para ello y pulsar el botón "Add". Una vez pulsado, se añadirá el rango a la tabla y se podrán gestionar los permisos de dicho rol.

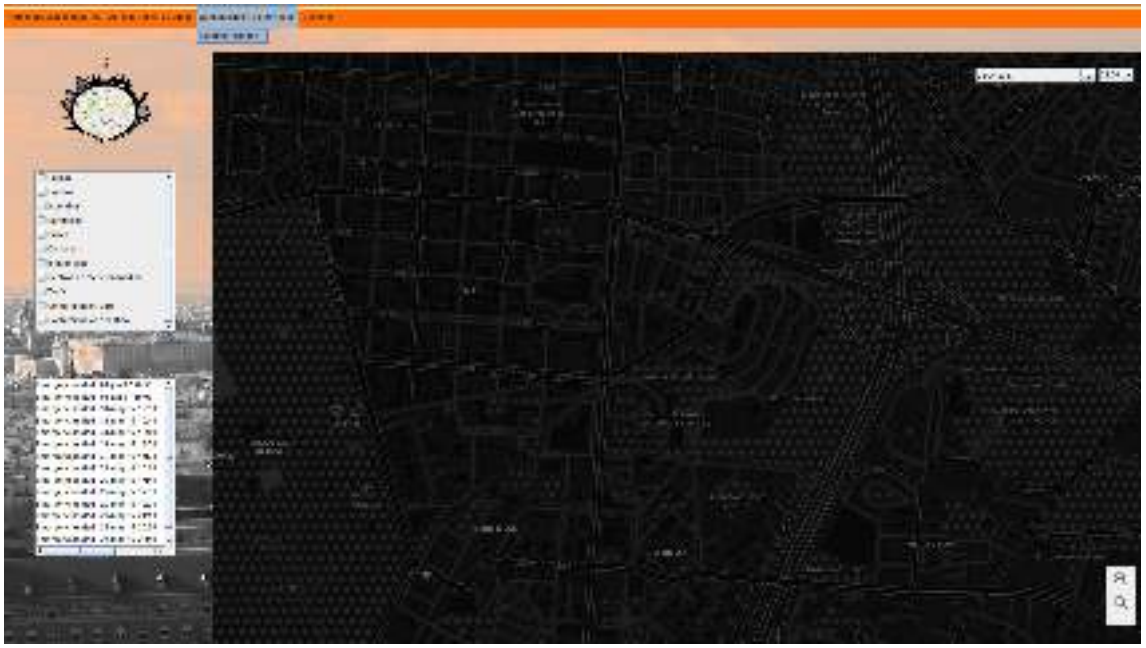


Figura 62: Click en gestionar permisos

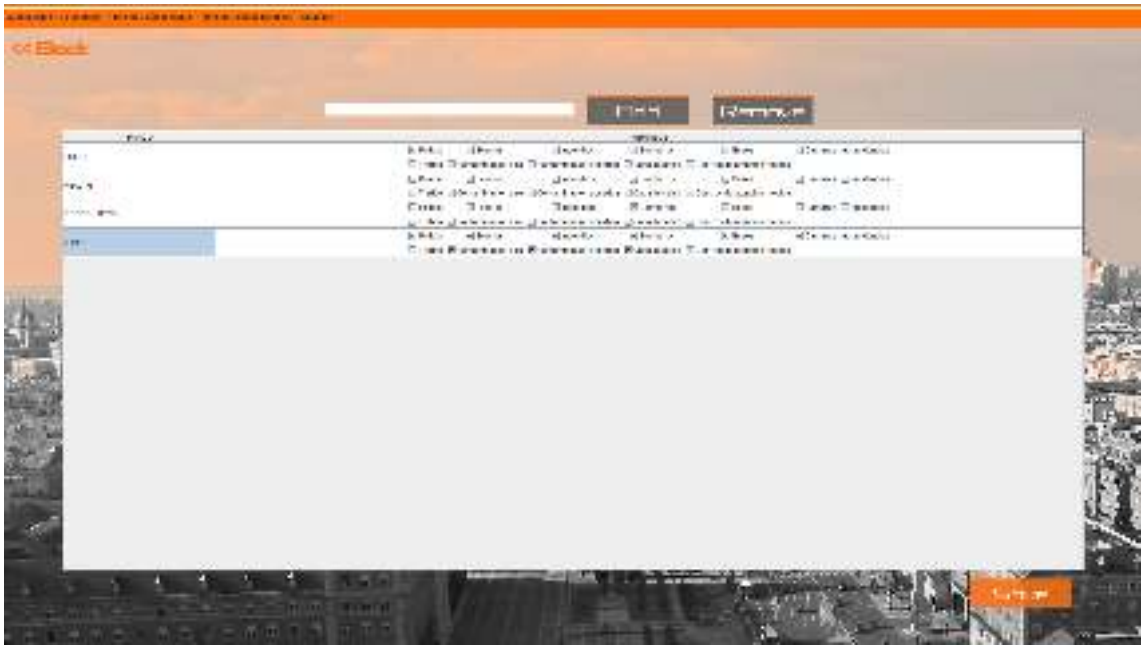


Figura 63: Panel de gestión de permisos

Eliminar rango

Para eliminar rango procederemos a hacer click en "Administración de permisos" y después click en "Gestionar permisos".

Una vez accedemos a dicho panel, veremos que consta de dos partes

claremente diferenciadas. Por un lado, un campo y un botón desde donde se podrán eliminar rangos. Por otro lado, una tabla en la que se pueden gestionar los permisos de los diferentes roles de usuario.

Para eliminar un rango, tenemos que introducir el nombre del rango en el campo asociado para ello y pulsar el botón "Remove". Una vez pulsado, se eliminará el rango de la tabla.

Gestionar permisos

Para gestionar permisos procederemos a hacer click en "Administración de permisos" y después click en "Gestionar permisos".

Una vez accedemos a dicho panel, veremos que consta de dos partes claremente diferenciadas. Por un lado, un campo y un botón desde donde se podrán añadir y eliminar rangos. Por otro lado, una tabla en la que se pueden gestionar los permisos de los diferentes roles de usuario.

Para gestionar permisos, tenemos que ubicar donde está el rango del cual queremos modificar su permisos y mediante un proceso de selección de casillas de verificación decidiremos cuáles son los permisos que tiene que tener dicho rol.

Una vez gestionados los permisos, procederemos a darle al botón "Save" y, inmediatamente, se guardarán los permisos asociados a todos los roles.

Opciones

Las opciones son las tareas básicas que puede realizar el administrador de la aplicación. Esto es, la configuración de la información del servidor del cual obtiene la información de las capas, la edición de la información de usuario (lo explicaremos más adelante en el manual de usuario) y el cierre de la sesión.

Configuración de la información del servidor

Para configurar la información del servidor procederemos a hacer click en “Opciones” y después click en “Configuración” (Figura 64).

Figura 64: Click en configuración

Una vez accedemos a dicho panel, encontraremos 8 campos. Estos campos corresponden con la URL del servidor, DNS del servidor, Usuario SFTP del servidor, Certificado del servidor, Usuario del servidor de la base de datos, contraseña del usuario del servidor de base de datos, nombre de la base de datos y distrito del cual queremos ver su salud en la Capa de Salud (Figura 65).

El URL del servidor es la dirección web asociada al servidor donde se encuentra la base de datos. EL DNS es el sistema de nombres asociado al servidor donde se encuentran los scripts de procesamiento. En este caso, el servidor de procesamiento. El usuario SFTP y contraseña del usuario SFTP es el identificador del usuario y la contraseña asociada al servidor SFTP que se encuentra en la máquina donde está el servidor de la base de datos. El certificado del servidor es el certificado que sirve para desencriptar la información que envía el servidor a la aplicación y nos permite también mandar la información encriptada al mismo. El usuario y contraseña de la base de datos es el identificador

Figura 65: Panel de configuración de información del servidor

del usuario y la contraseña asociada al acceso al MongoDB. El nombre de la base de datos es el nombre de la base de datos de MongoDB de la que se obtiene la información. El distrito es un selector para elegir el distrito del que quiere obtener la información de salud del distrito en la capa de Salud. Si quieres modificar esta información porque has creado otro arquitectura de red similar a la nuestra con los scripts del proyecto simplemente tienes que cambiar esta información y darle al botón Save.

Cerrar sesión

Para cerrar sesión, procederemos a hacer click en “Opciones” y después click en “Cerrar sesión” (Figura 66).

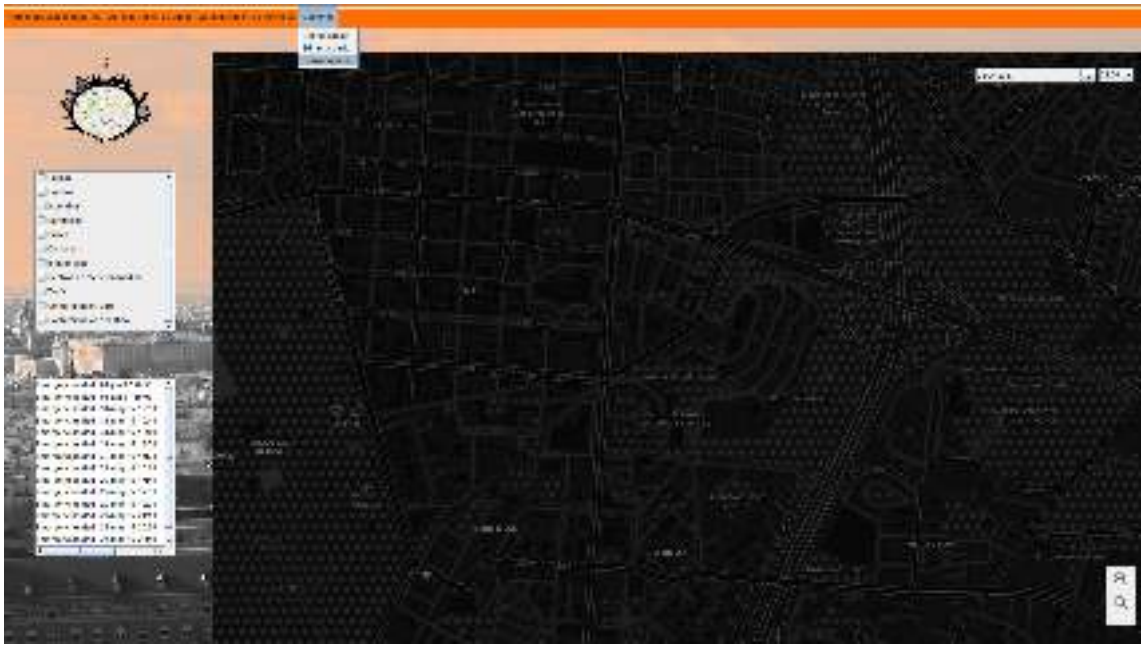


Figura 66: Click en cerrar sesión

Anexo IV. Manual de usuario

Una vez instalada la aplicación, crearemos un acceso directo en el escritorio. Acto seguido, haremos click en él.

Lo que aparecerá a continuación será el sistema de login de la aplicación como veremos a continuación (Figura 67).

Figura 67: Login de la aplicación

Procederemos a introducir los datos del usuario y haremos click en el botón de "Login". Una vez logueado, veremos el panel de inicio del usuario (Figura 68).

El usuario posee un panel de gestión que consta de 2 bloques de gestión: "Configuración" y "Opciones".

Configuración de la información del servidor

Para configurar la información del servidor procederemos a hacer click en "Opciones" y después click en "Configuración" (Figura 69).

Una vez accedemos a dicho panel, encontraremos 8 campos. Estos campos corresponden con la URL del servidor, DNS del servidor,

Figura 68: Panel de inicio del usuario

Figura 69: Click en configuración

Usuario SFTP del servidor, Certificado del servidor, Usuario del servidor de la base de datos, contraseña del usuario del servidor de base de datos, nombre de la base de datos y distrito del cual queremos ver su salud en la Capa de Salud (Figura 70).

El URL del servidor es la dirección web asociada al servidor donde

Figura 70: Panel de configuración de la información del servidor

se encuentra la base de datos. EL DNS es el sistema de nombres asociado al servidor donde se encuentran los scripts de procesamiento. En este caso, el servidor de procesamiento. El usuario SFTP y contraseña del usuario SFTP es el identificador del usuario y la contraseña asociada al servidor SFTP que se encuentra en la máquina donde está el servidor de la base de datos. El certificado del servidor es el certificado que sirve para desencriptar la información que envía el servidor a la aplicación y nos permite también mandar la información encriptada al mismo. El usuario y contraseña de la base de datos es el identificador del usuario y la contraseña asociada al acceso al MongoDB. El nombre de la base de datos es el nombre de la base de datos de MongoDB de la que se obtiene la información. El distrito es un selector para elegir el distrito del que quiere obtener la información de salud del distrito en la capa de Salud. En este caso, lo único que podrás modificar es el distrito seleccionado para la capa de Salud. Para ellos, pincharas en el selector y seleccionaras el distrito que desees. Una vez seleccionar, pulsará el botón “Save”.

Opciones

Las opciones son las tareas básicas que puede realizar el usuario de la aplicación. Esto es, la edición de la información de usuario y el cierre de la sesión.

Editar usuario

Para editar un usuario, procederemos a hacer click en Opciones y después click en “Editar usuario” (Figura 71).

Figura 71: Click en editar usuario

Una vez accedemos a dicho panel, veremos que consta de tres campos: Nick de usuario, Contraseña y Rol de usuario (Figura 72).

El Nick de usuario es el identificador de usuario que identifica al usuario cuando logueas en la aplicación. La contraseña es el código secreto que conoce solo el usuario al que pertenece el usuario que vamos a crear. El rol de usuario es el identificador del rango al que se le han asignado determinados permisos de capa para la ejecución de las tareas del usuario. El usuario lo único que podrá modificar es su contraseña. Una vez que entendamos los datos que hay que modificar, procederemos a pulsar el botón “Modify”.

Figura 72: Panel de editar usuario

Cerrar sesión

Para cerrar sesión, procederemos a hacer click en “Opciones” y después click en “Cerrar sesión” (Figura 71).

Figura 73: Click en cerrar sesión

Ya hemos visto todas los bloques de gestión que el usuario puede

gestionar pero esto no es todo lo que el usuario puede hacer. A continuación, vamos a mostrar una serie de acciones que el usuario puede hacer. Esto es, seleccionar capas y mirar el histórico de las mismas en el caso de que sean una capa histórica, gestionar las incidencias de la ciudad y ver los tweets de Emergencias Madrid.

Gestionar las incidencias de la ciudad

Desde el inicio del panel de usuario, seleccionamos las capas de la Policias y de la incidencias. Vemos que hay una incidencia en el distrito de Salamanca en el Hotel Wellington. Ésta posee una descripción de la incidencia, un campo que indica si es una obra, un campo si indica si es una accidente, la duración de la incidencia. Y abajo encontramos una lista de comisarias cercanas al incidente y una lista de parques de bomberos cercanos al incidente. En el caso de que la incidencia sea un accidente y poseas permisos para la capa de Centros de atención médica también se mostrarán los centros de atención médica y hospitales cercanos al incidente (Figura 74).

Figura 74: Panel de inicio de usuario con capa de Incidencias seleccionada

Como vemos que no es ni una obra, ni un accidente y además no surge la necesidad de mandar un camión de bomberos, procedemos a

enviar una patrulla de policia. Para ello, vemos que la comisaria más cercana es Usera. Para ver la información pulsamos el botón enviar y si tiene efectivos disponibles, aparecerán ya que si no los tuviera, no aparecería en comisarias cercanas (Figura 75).

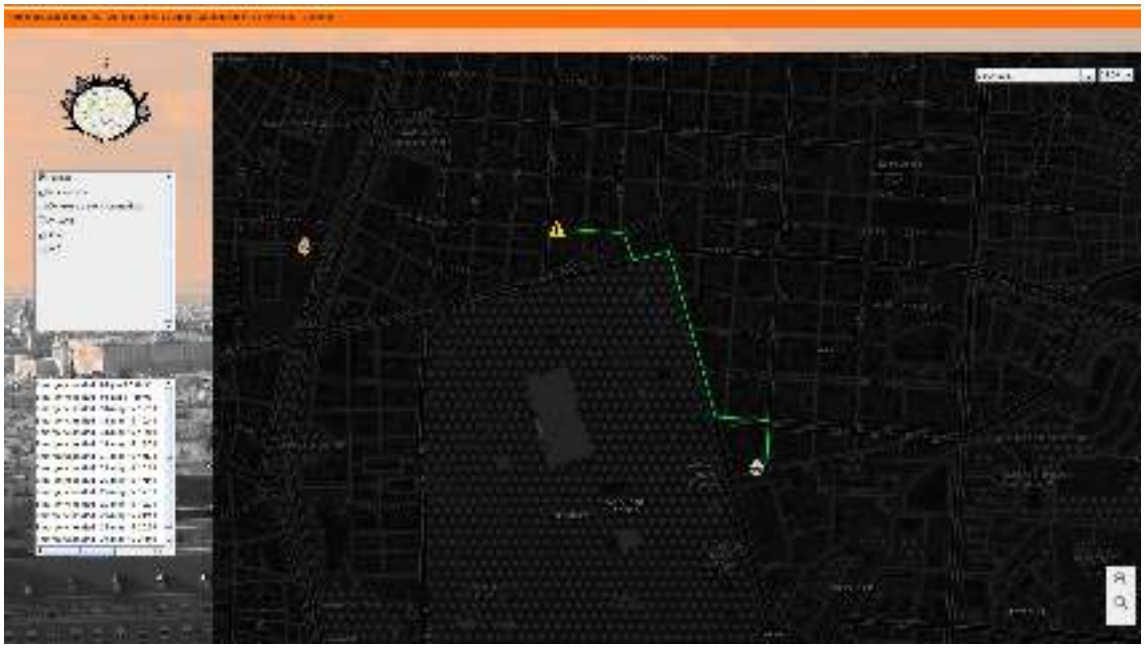


Figura 75: Mandando patrulla al incidente en el Hotel Wellington

Además de gestionar las incidencias podemos ver un histórico de las incidencias. Vamos a ver las incidencias del día 1 de Junio y las del 31 de Mayo (Figura 76).

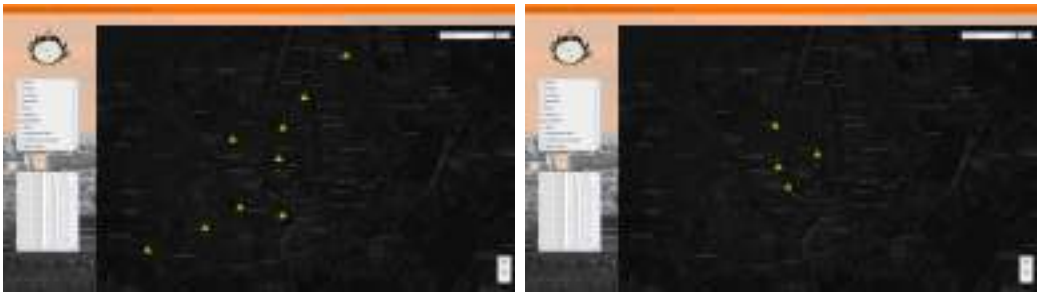


Figura 76: Histórico de incidencias del 31 de Mayo y 1 de Junio

Mirar el histórico de capas históricas

Ya hemos visto que la capa de Incidencias es una capa histórica. Esta no es la única capa que es histórica. A continuación vamos a

mostrar un par de ejemplos más de capas históricas mostrando la información de los días 1 de Junio y 31 de Mayo para el caso de la capa de Contaminación de aire y de los días 30 y 31 de Mayo para el caso de la capa acústica (Figuras 77 y 78).

Figura 77: Histórico de capa de Contaminación del aire del 31 de Mayo y 1 de Junio

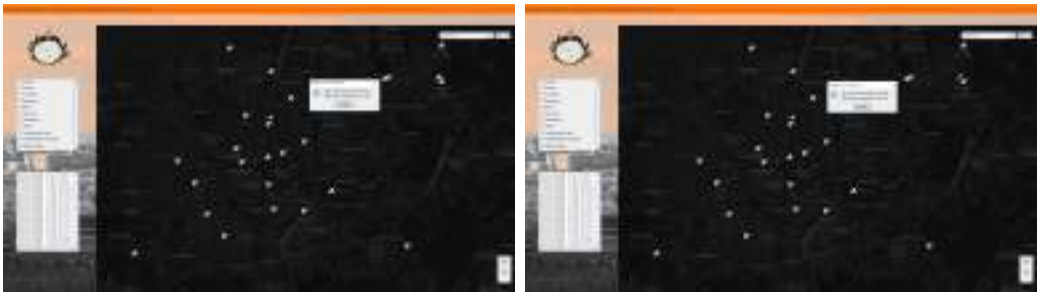


Figura 78: Histórico de capa de Contaminación acústica del 30 y 31 de Mayo

Ver los tweets de Emergencias Madrid

Desde el inicio del panel de usuario, vemos que al principio del panel, ubicado debajo del selector de capas está la lista de tweets. Para ver los tweets simplemente tenemos que hacer doble click en uno de ellos y lo veremos como se indica (Figura 79).

Figura 79: Viendo tweets de Emergencias Madrid

Glosario de términos

Smart Cities

Smart City Overseer viene del concepto "Smart city" en español "Ciudades Inteligentes" debido a que toma datos que se modifican automáticamente mediante los servicios de la ciudad como pueden ser las cámaras de tráfico en tiempo real, las incidencias y mucho más.

Este concepto, está basado en la sostenibilidad de la ciudad y su capacidad de respuesta de tal manera que pueda responder a las necesidades básicas de cualquier institución o empresa tanto en el plano económico como en el operacional.

Gracias a este concepto las "Ciudades inteligentes" disponen de mediciones de la contaminación y el estado de la ciudad, y con esto Smart City Overseer es capaz de mostrar el estado en tiempo real de la ciudad.

Big data

El "Big data" o tratamiento de datos masivos, esta muy presente en Smart City Overseer ya que debemos tratar los datos de la ciudad y estos datos son de un tamaño gigantesco, de tal manera que se promueva un desacoplamiento de las funciones tratadas para la recolección y el tratamiento de datos.

Este tratamiento de datos masivos afecta a Smart City Overseer tanto debido a tantos datos que poseen las ciudades y sin olvidar que estos datos cada día crecen que se usaron dos servidores completamente desacoplados, uno para la recolección de datos y su procesamiento y otro para la base de datos con los datos ya tratados para servir a la aplicación y el bot.

Cloud

El “Cloud”, “Cloud Computing” o “Computación en la nube” es completamente necesario en Smart City Overseer debido a que se tiene la necesidad de tener disponibles los datos estemos donde estemos con nuestra aplicación o bot. Para este “Cloud” hay muchos servicios entre ellos “AWS - Amazon Web Services” que nos facilitan la creación y personalización de las máquinas como las necesitamos.

Geolocalización

La geolocalización es la capacidad de obtener la ubicación geográfica de un objeto o componente, Por tanto es un factor muy importante en Smart City Overseer ya que localizamos la ubicación de todos los datos que recibimos de la ciudad.

NoSQL

El “NoSQL” es una clase de sistemas de gestión de bases de datos donde los datos son almacenados sin estructuras fijas olvidando las tablas del sistema “SQL” y de manera que se puedan tratar datos masivos.

Estado de la ciudad

El concepto “Estado de la ciudad” dentro de la aplicación lo entendemos como cualquier dato representado sobre la ciudad que nos aporte su situación actual respecto a este categorizadamente por secciones, como puede ser la contaminación del aire o la salud del distrito.

Emergencias

El concepto de “Emergencia” dentro de Smart City Overseer es referido a cualquier incidente en la vía pública que el rol que estemos

tratando pueda solucionar y deba solucionarlo en el menor tiempo posible.