Escuela Superior
de Ingeniería y Tecnología
Universidad de La Laguna

Trabajo de Fin de Grado

Aplicación web de mapas geográficos

Geographic maps web application

Kristian Martínez García

La Laguna, 31 de mayo de 2020

D. **PINO CABALLERO GIL**, con NIF 45534310Z Catedrática de Universidad adscrita al Departamento de Ingeniería Informática y de Sistemas, como tutora

D. **Néstor García Moreno**, con NIF 79085553F contratado por proyecto de investigación adscrito al Departamento de Ingeniería Informática y de Sistemas, como cotutor

C E R T I F I C A (N)

Que la presente memoria titulada:

”*Aplicación web de mapas geográficos*”

ha sido realizada bajo su dirección por D. **Kristian Martínez García**, con N.I.F. 79060432W.

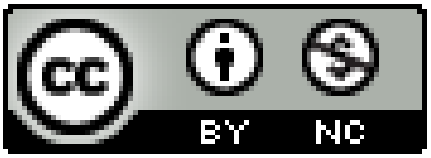
Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 31 de mayo de 2020

Agradecimientos

Agradecerle a mi tutora, Pino Caballero Gil, y a mi cotutor Néstor García Moreno por su apoyo en todo momento del proceso de desarrollo y por su interés en que este proyecto salga adelante. A Carlos Rosa Remedios por su afecto en mostrarnos las instalaciones y el trabajo que se realiza en el 112 Canarias.

A mi familia por apoyarme durante toda la carrera de Ingeniería Informática.

Licencia



Resumen

El objetivo de este trabajo ha sido una investigación continua sobre el conjunto de tecnologías web MEVN (Mongo - Express - Vuejs - Nodejs), librería open source de mapas interactivos (leaflet) y funcionamiento de las aplicaciones web de carácter geográfico en las instalaciones del 112 del Gobierno de Canarias y Hexagon Safety.

*Se ha aplicado el conocimiento de la investigación en el desarrollo de una aplicación web para la representación geográfica de **INR** (nos referiremos a incidentes no rutinarios) que muestren el tipo de riesgo, magnitud, descripción del incidente. El fin de esta aplicación es la demostración práctica de estos conocimientos aportando aspectos similares al aplicativo usado en las instalaciones del 112. Cabe destacar que la modularidad en el proyecto supone un proceso importante, pues la idea es que se puedan conectar varios servicios entre sí para una mayor funcionalidad.*

*Por último, para el desarrollo de la aplicación, el proyecto se ha separado en 2 marcos de trabajo. La primera parte es el **backend**, que trabaja del lado del servidor y se ocupa de la lógica de la aplicación. La segunda parte es el **frontend**, que se enfoca en el usuario, en todo con lo que podemos interactuar y lo que vemos mientras navegamos.*

Palabras clave: INR, MEVN, COVID-19, MVC, HTML, CSS y JAVASCRIPT

Abstract

The goal of this work has been continuous research on the set of MEVN web technologies (Mongo - Express - Vuejs - Nodejs), open source library of interactive maps (leaflet) and the operation of geographic web applications in 112 facilities of the Canary Islands Government and Hexagon Safety.

*The knowledge of the investigation has been applied in the development of a web application for the geographical representation of **INR** (we will refer to non-routine incidents) that show the type of risk, magnitude, description of the incident. The purpose of this application is the practical demonstration of this knowledge, providing similar aspects to the application used in the 112 facilities. It should be noted that modularity in the project is an important process, since the idea is that several services can be connected to each other to greater functionality.*

*Finally, for the development of the application, the project has been separated into 2 frameworks. The first part is the **backend**, which works on the server side and deals with the application logic. The second part is the **frontend**, which focuses on the user, on everything we can interact with and what we see while browsing.*

Keywords: INR, MEVN, COVID-19, MVC, HTML, CSS y JAVASCRIPT

Índice general

Índice de Figuras

Índice de Tablas

Capítulo 1

Introducción

1.1. Motivación

La invención del mapa geográfico ha supuesto un gran avance tecnológico para la humanidad, que aún sigue vigente en la actualidad, tanto para un turista que está de vacaciones en un lugar y necesite consultar el mapa para llegar hasta una dirección como para un cuerpo de seguridad o fuerzas del estado que necesiten verificar el lugar de los hechos de un suceso ocurrido. Para este último, la mayoría de eventos o incidentes que pueden ocurrir (accidente de vehículo, derrame de materiales peligrosos, fenómenos naturales, reyertas, inundaciones...) son mucho más efectivos si se muestra gráficamente la localización donde han tenido lugar el suceso en un mapa geográfico con la información más relevante posible (descripción de lo ocurrido, tipo de riesgo: alto, medio, bajo, coordenadas, agentes que intervienen,etc...). La finalidad recae en organizar y controlar los efectivos destinados junto con los recursos disponibles a los alrededores para poder actuar de la manera más adecuada al evento.

Hay infinidades de aplicaciones que proporcionan información con un detalle asombroso, no obstante la obtención de las licencias para la explotación o distribución de estas suelen indicar un alto valor económico. En este proyecto se apuesta por la libertad de los usuarios/desarrolladores para ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el Software sin coste alguno. Esto se incluye en la definición de software libre que es precisamente lo que se quiere conseguir.

El Centro Coordinador de Emergencias y Seguridad (CECOES) 1-1-2 del Gobierno de Canarias es el que se encarga de dar una respuesta rápida y eficaz a cada incidente. La combinación de profesionales en el sector y una aplicación web libre que incluya mapas geográficos junto con la tecnología geoespacial y una vía de comunicación que admita modificar,

mostrar editar y filtrar datos geoespaciales de forma segura, permitirá crear una unidad de control en instalaciones de instituciones que ayudará dirigir, coordinar y gestionar multitud de incidencias que se generan todos los días.

El presente Trabajo de Fin de Grado presenta una perfecta ocasión para ayudar a las instituciones del 112 así como un buen y profundo aprendizaje de las tecnologías Full Stack.

1.2. Objetivos

Como se comentó anteriormente, el principal objetivo de este proyecto es crear una plataforma web funcional de mapas geográficos que incluya un control de acceso donde los usuarios de la aplicación tengan asignados unos roles preestablecidos que le permitan o impidan realizar funciones como visualizar, introducir, editar, borrar determinada información. Incluye también una capacidad de filtrado de éstos: por área, por riesgo o tipo de evento para su visualización concreta en el mapa. También realizar una barra de herramientas que permita dibujar figuras como líneas rectas, cuadrados, círculos o marcadores en el mapa exponiendo un área, por ejemplo, para indicar la zona afectada por un incendio forestal o una inundación. La finalidad es que el CECOES pueda utilizar dicha herramienta para una mayor eficacia en la respuesta y protección a la ciudadanía.

- Resumen de los objetivos generales:
    - Una plataforma con control de acceso de usuarios que incluya encriptación de contraseña.
    - Distinción de roles y funciones de los usuarios.
    - Mostrar incidentes en el mapa con la información de los mismos.
    - Sistema de filtrado de incidentes segun el usuario
    - Sistema de dibujo de polígonos y figuras en el mapa.
    - Desarrollo modular para en un futuro servicios distintos se puedan integrar perfectamente para una mayor funcionalidad del sistema

Dicha aplicación está orientada para usarse a un nivel insular, no obstante, debido a la situación extraordinaria que se está viviendo globalmente por causa de la pandemia global por el COVID-19 se introducirán datos proporcionados por el Ministerio de Sanidad por cada Comunidad Autónoma

en España, como casos positivos detectados, fallecimientos, recuperaciones, etc. transformados en una información gráfica y visual para el usuario.

1.3. Coronavirus

Lamentablemente este informe está hecho bajo un extraordinario suceso que ha conmovido al mundo. Una pandemia global que se ha cobrado muchas víctimas y que no tiene límites en su dispersión y progresión. Marcando un hecho histórico en países como España, Italia y Francia tengan que cerrar fronteras y mantener a la ciudadanía en confinamiento en sus hogares, suponiendo un riesgo psicológico bastante grande para los invididuos.

Es cierto que nada volverá a ser como antes. Ha marcado un antes y un después, en la que hay que ser precavido y mantener unos niveles de higiene impecable y mostrar respeto al prójimo. En esta situación los ciudadanos debemos estar a la altura ya que la pandemia viene para quedarse al menos todo este año 2020.

1.4. Fases

Las fases del desarrollo del presente trabajo se pueden agrupar en 4 fases diferentes.

Una primera fase o etapa se ha dedicado al estudio para comprender y entender toda la tecnología e información a usar, entre un abanico amplio de tecnologías se eligió el stack MEVN (Mongo - Express - Vuejs - Nodejs). Incluye análisis de herramientas como Hexagon Safety y las aplicaciones contenidas en las instalaciones del 112 Canarias (Grafcan); su diseño, ventajas, desventajas, implementaciones y funcionalidades, así como el diseño de modelo de datos y controladores para alcanzar los objetivos comentados anteriormente

Una segunda fase que ha consistido en valorar y pautar la estructura del proyecto, se planteó 2 ámbitos de trabajo, el primer ámbito es el **backend**, que trabaja del lado del servidor y se ocupa de la lógica de la aplicación (base de datos, controladores, servicios) siguiendo el modelo MVC (Modelo-Vista-Controlador). El segundo ámbito es el **frontend**, que se enfoca en el usuario, en todo con lo que podemos interactuar y lo que

vemos mientras navegamos, para lo cual utiliza tecnología HTML, CSS, JAVASCRIPT, Boostrap y Axios.

La tercera fase consiste en el desarrollo de los 2 marcos ya comentados y de la comunicación de información entre los mismos. Backend: creación de modelos, controladores, configuraciones de rutas, middleware y servicios incluyendo encriptación). En el marco de frontend: creación de componentes, vistas, router, sistema de almacenamiento en el navegador y configuración de las rutas entre otros.

Por último, tenemos la última fase que comprende los métodos de testeos y mejoras de las funcionalidades y despliegue del proyecto, así como la comprobación de requisitos mínimos requeridos.

1.5. Estructura de la memoria

El presente documento está estructurado en siete capítulos, donde el primero es de introducción al trabajo, detallando los objetivos y motivaciones, las etapas del desarrollo del proyecto y la estructura de este documento.

A continuación le sigue el segundo capítulo, donde se muestra una introducción de la herramienta propuesta, se introducen los conceptos fundamentales para entender el funcionamiento del conjunto y se habla del tema actual: El coronavirus.

En tercer lugar, se enuncian las tecnologías y las características que engloban a cada una de ellas.

Por último se encuentra los capítulos que contiene las conclusiones, líneas de futuro tanto en español como en inglés, un presupuesto de lo que ha costado en el proyecto, posibles implementaciones del trabajo futuras y la bibliografía donde se encuentra información relevante que ha aportado utilidad para la realización del trabajo final de grado.

Capítulo 2

Introducción a la aplicación

2.1. Definición del problema

El Centro Coordinador de Emergencias y Seguridad (CECOES) 1-1-2 del Gobierno de Canarias, dependiente de la Consejería de Administraciones Públicas, Justicia y Seguridad, es un servicio público que nació para dar respuesta a todas las llamadas de emergencia que se producen en el Archipiélago canario. [1]

Cuando un ciudadano realiza una llamada de emergencia o urgencia, un operador de demanda es el encargado de atender y realizar un cuestionario con una serie de preguntas clave para situar geográficamente el incidente, saber que ocurre y dónde. Mientras el operador de demanda va tomando los datos del incidente y gracias a unos protocolos automáticos, se clasifica la llamada de una manera efectiva y concisa. Así, según el tipo de incidente se activa el servicio del sector competente, ya sea sanitario, de seguridad, extinción de incendios, salvamento y rescate. Cada sector contiene los profesionales competentes y los recursos necesarios para gestionar el incidente. Los responsables de supervisar de toda actividad que se genera garantizando la respuesta más adecuada al ciudadano y el cumplimiento de los protocolos establecidos son el Coordinador Multisectorial y el Gestor Operativo (Ver figura 2.1)

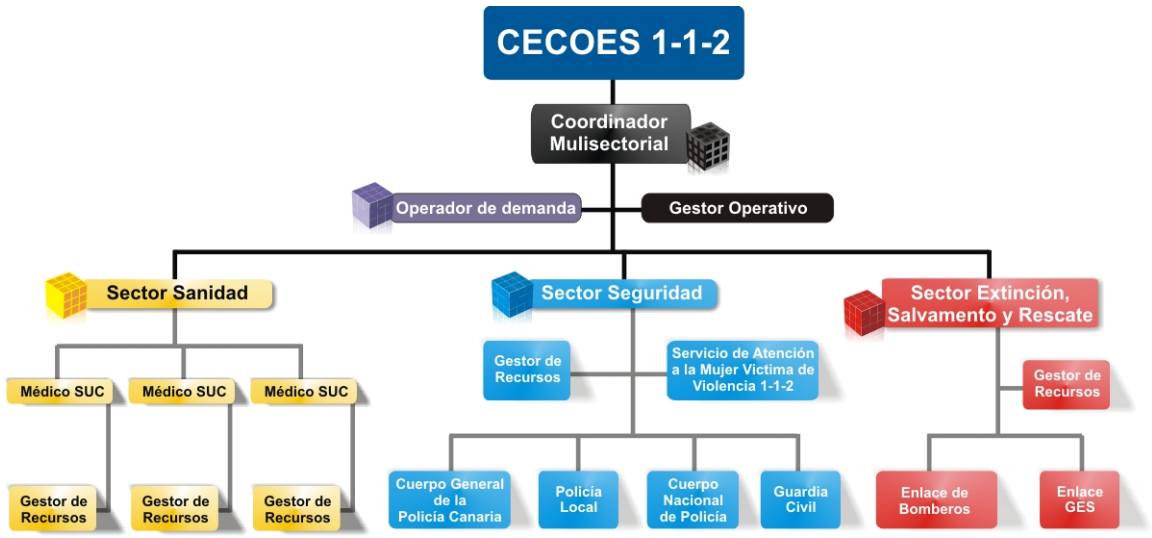


Figura 2.1: Esquema del CECOES

Para garantizar lo anterior, la coordinación entre los distintos sectores y la gestión de los recursos disponibles debe ser de vital importancia, pues el objetivo es garantizar una respuesta rápida y eficaz al incidente ocasionado. La obtención de información como ubicación en tiempo real del incidente, los recursos disponibles más cercanos al mismo, tiempo de llegada, etc... pueden ser requeridos por cualquier personal de emergencias, por lo que es imprescindible que estos estén disponibles para todos los involucrados o en su defecto, se pueda proporcionar acceso a los mismos de forma veloz, eficiente y segura. [2]

2.2. Conceptualización de la propuesta

El presente proyecto tiene como objetivo crear una plataforma web con control de acceso que permita visualizar, introducir, editar y borrar incidentes o figuras en un mapa geográfico según el rol asignado al usuario, la cual sea utilizado por todos los involucrados en una emergencia. desde el operador encargado de recibir las llamadas de los alertantes, los gestores de recursos hasta las organizaciones que proveen los recursos (ver figura 2.2)

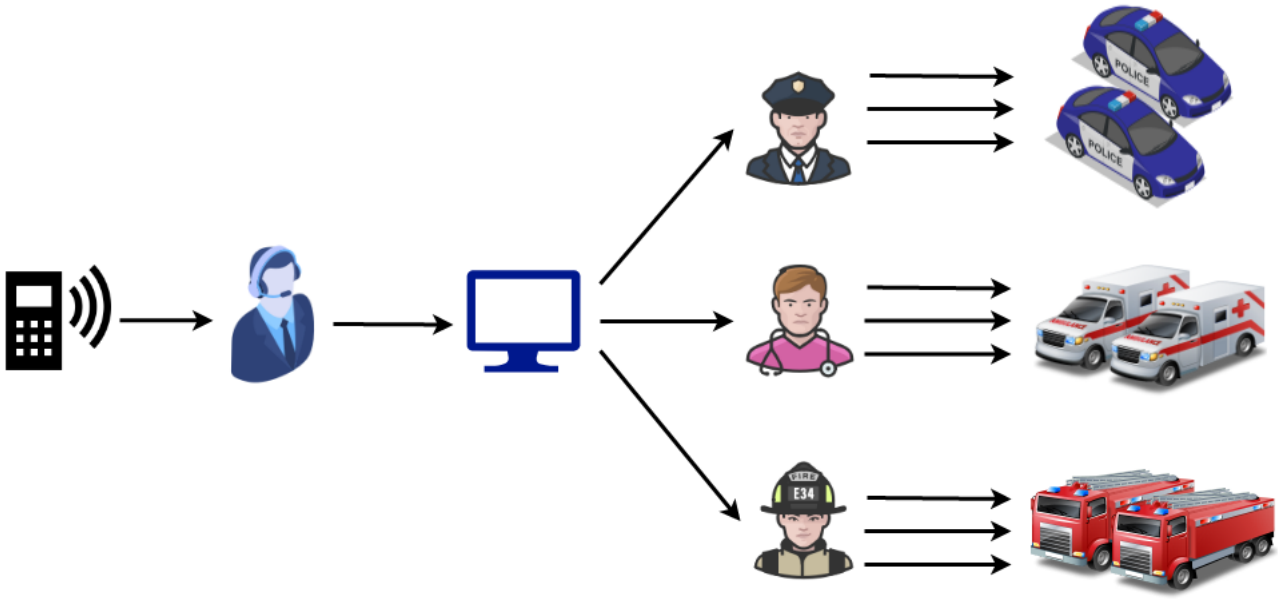


Figura 2.2: Esquema de procesos

Hay dos entes ya comentados anteriormente que suponen la base de la aplicación. El mapa geográfico y los incidentes, clasificados en dos tipos: rutinarios y no rutinarios.

IRN (Incidente no rutinario) es aquel cuya magnitud genera la activación de medios extraordinarios, afecte o no a múltiples víctimas mientras que el IR (Incidente rutinario) es aquel cuyo impacto no supera los niveles de alerta que impliquen el uso de medios extraordinarios. Nos centraremos principalmente en los IRNs. En la figura 2.3 se muestra un ejemplo de un IRN.

Los desastres recientes han llamado la atención sobre la vulnerabilidad de las poblaciones humanas y infraestructura y el costo extremadamente alto de recuperarse del daño que tienen causado. Los ejemplos incluyen el terremoto de Wenchuan de mayo de 2009, el huracán Katrina en septiembre de 2005, y el tsunami del Océano Índico de diciembre de 2004. En todos estos casos los impactos fueron graves, en daños, lesiones y pérdida de vidas, y se extendieron por grandes áreas. En todos estos casos, la tecnología moderna ha traído informes e imágenes a la atención casi inmediata de gran parte de la población mundial y en Katrina en caso de que millones de personas de todo el mundo pudieran ver los eventos a medida que se desarrollaban en tiempo casi real Las imágenes capturadas de los satélites se han utilizado para crear daños. evaluaciones y mapas digitales se han utilizado para dirigir suministros y guiar la esfuerzo de

recuperación. [3]

Figura 2.3: Esquema de procesos

Se utilizará una base de datos no relacional como es MongoDB [4] MongoDB es una base de datos orientada a documentos. Esto quiere decir que en lugar de guardar los datos en registros, guarda los datos en documentos. Estos documentos son almacenados en BSON, que es una representación binaria de JSON.

Una de las diferencias más importantes con respecto a las bases de datos relacionales, es que no es necesario seguir un esquema. Los documentos de una misma colección ()concepto similar a una tabla de una base de datos relacional), pueden tener esquemas diferentes.

2.3. Aplicaciones similares

Centro Control Hexagon

Hexagon es una empresa internacional con múltiples divisiones especializadas (geoespacial, geosistemas, agricultura, minería, infraestructuras y seguridad, etc.) Dentro de esta ultima tienen dos soluciones para el tratamiento de eventos e invidencias “Hexagon Oncall Dispatch” para la gestión operativa y “Hexagon Oncall Planning and Response” para grandes eventos (ver figura 2.4)

El Centro de Control más interesante es el de su primera solución “Hexagon Oncall Dispatch” ya que se utiliza para la gestión de incidentes utilizando soluciones geoespaciales para mostrar información en tiempo real. Ampliamente utilizada por distintos cuerpos y fuerzas de seguridad de múltiples países. Las herramientas más llamativas son:

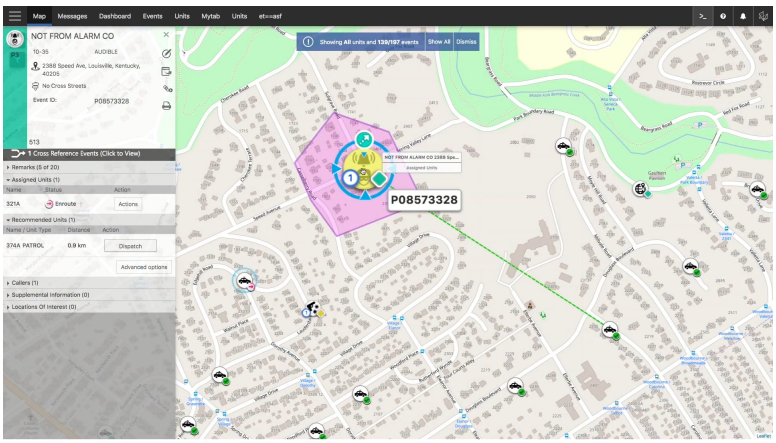


Figura 2.4: Ejemplo de la aplicación Hexagon

- Hexagon OnCall Dispatch se compone de:
  - **Calltaker:** recepción de alarmas y tipificación de incidentes
  - **Dispatchers:** gestión de recursos desde una sala o un PMA
  - **Unidades:** vehículos que reciben, informan y comparten utilizando tablets o radios
  - **Agentes de campo:** reciben, informan y comparten
- Hexagon Oncall Dispatch: Arquitectura
  - Arquitectura web
  - Cliente ligero con cloud híbrida
  - Permite despliegue SaaS
  - Accesible desde cualquier sitio y plataforma
- Hexagon Oncall Dispatch: Mapa
  - Visualizar y ocultar información según se hace zoom in y zoom out para no sobrecargar la interfaz ni al usuario.
  - Capacidad de agrupación (clustering) de incidentes. Se muestra el color del incidente de mayor prioridad en el cluster
  - Iconos de incidentes con estilos diferentes para representar el tipo de incidente
  - Menús contextuales radiales para unidades e incidentes

GRAFCAN

GRAFCAN es una empresa pública de la Comunidad Autónoma de Canarias, con capital perteneciente íntegramente a la Administración Pública

de la Comunidad Autónoma de Canarias, adscrita a la Consejería de Transición Ecológica, Lucha contra el Cambio Climático y Planificación Territorial del Gobierno de Canarias.

GRAFCAN realiza actividades de producción, mantenimiento y gestión de la información geográfica y territorial de la Comunidad Autónoma de Canarias, siendo la responsable funcional del Sistema de Información Territorial de la Comunidad Autónoma (SITCAN). La empresa es responsable de una permanente dirección técnica de las actividades desarrolladas, orientando los criterios de desarrollo, los procedimientos y la implantación tecnológica. Asimismo, es el responsable de armonizar y coordinar la posible ejecución externa de todas o algunas de las acciones a desarrollar (ver figura 2.5)

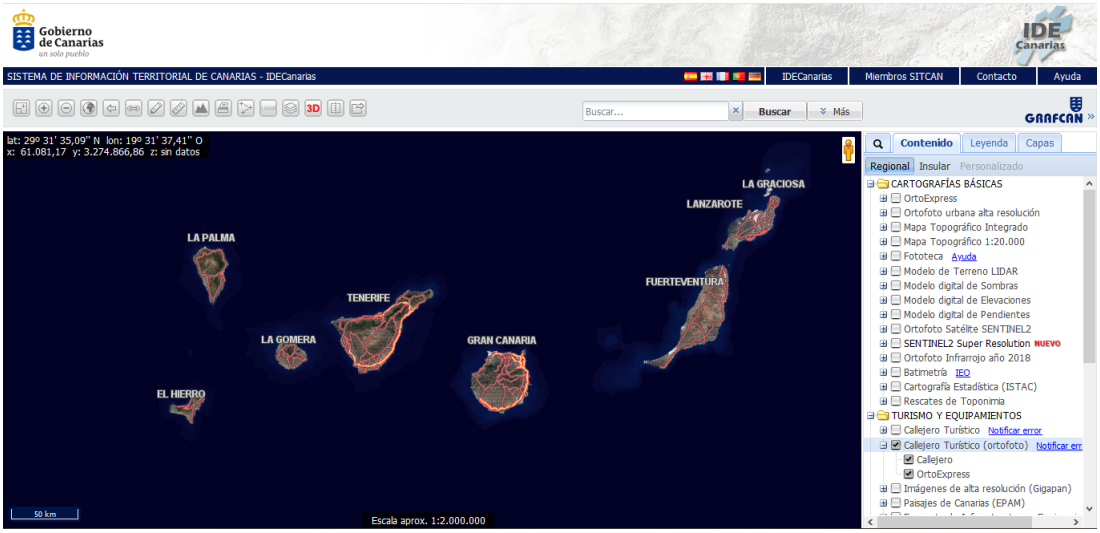


Figura 2.5: Ejemplo del visor de GRAFCAN

SIS ECU 9-1-1 del Ecuador

Sistemas de seguridad ciudadana por georeferenciación y geolocalización para zonas rurales del cantón Cuenca incorporados al SIS ECU 9-1-1 del Ecuador. Se presenta un diseño similar al que dispone el Centro Integrado ECU 9-1-1. [5]

El sistema se amplía a las zonas rurales del cantón Cuenca diseñado en función de la experiencia exitosa en la ciudad de Cuenca, además se considera el sistema del botón de auxilio para las camionetas que están legalmente facultadas para prestar los servicios en las zonas rurales. Al sistema GIS se lo incorporan cámaras de monitoreo en las principales arterias

viales y espacios públicos para continuamente realizar el monitoreo desde el Centro integrado ECU 9-1-1, al igual que receptar llamadas de auxilio y accionamiento de botones de pánico desde sitios estratégicos. En el Centro de monitoreo se tendrán alarmas de emergencia con la georeferenciación de donde provienen los llamados de auxilio rurales para luego de un rápido análisis interno y apoyados en los sistemas de telecomunicaciones se puedan despachar los recursos policiales, bomberos, cruz roja, entre otros. (ver figura 2.6)

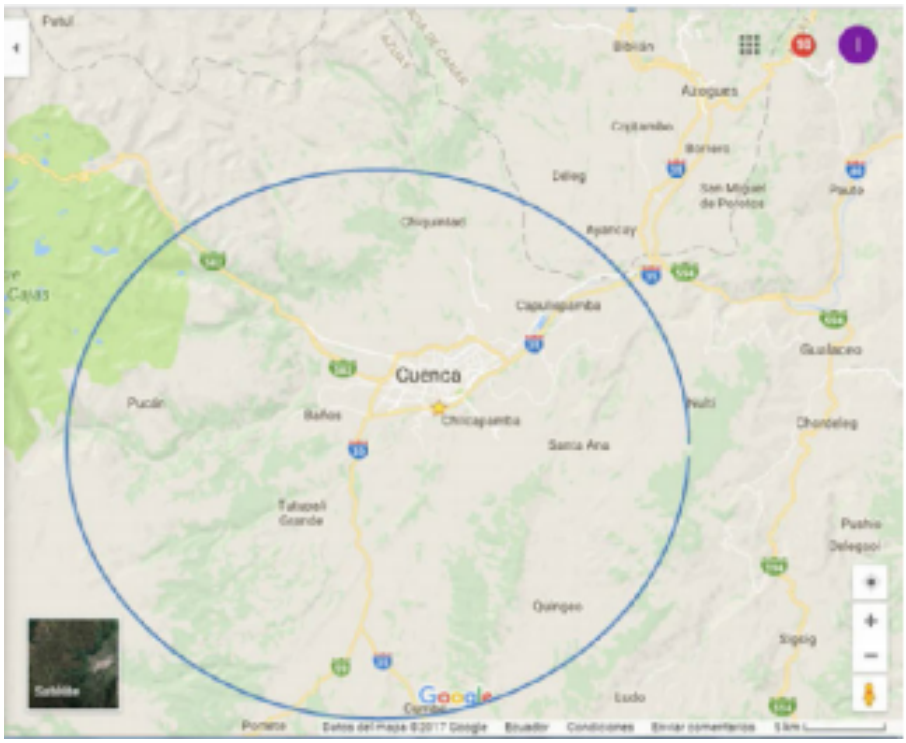


Figura 2.6: Ejemplo del Sistema de Visualización

Capítulo 3

Tecnologías y metodologías

3.1. Características

Como se ha comentado anteriormente, Se ha utilizado en el proyecto el stack MEVN (MongoDb - Express - Vuejs - Nodejs) que consiste en un conjunto de frameworks y herramientas que ayudan al desarrollo de las aplicaciones. Se nombrarán todas las tecnologías y librerías usadas pertenecientes a cada ámbito.

Figura 3.1: Logo Stack MEVN

3.2. Tecnologías y herramientas usadas

Entorno de desarrollo

Visual Studio Code es un editor de programación multiplataforma desarrollo por Microsoft. Es un proyecto de software libre que se distribuye bajo la licencia MIT, aunque los ejecutables se distribuyen bajo una licencia gratuita no libre.

Por defecto, Visual Studio Code es compatible con gran variedad de lenguajes de programación. Además trae un conjunto de características que facilitan el proceso de codificación como puede ser el uso de Snippets, Resaltado de sintaxis, Autocompletado, Debugger, etc. Todas estas características pueden ser extendidas a través de complementos, disponibles en un repositorio central. [6]

Figura 3.2: Visual Studio code

Paquetes NPM

En ambos ámbitos, JavaScript usa un gestor de paquetes node (NPM) para instalar librerías paquetes para ser utilizados. Se demostrarán los paquetes usados en la aplicación por ámbito. [7]

Figura 3.3: Visual Studio code

Webpack

Webpack es una herramienta de compilación (una build tool en la jerga) que coloca en un grafo de dependencias a todos los elementos que forman parte de tu proyecto de desarrollo: código JavaScript, HTML, CSS, plantillas, imágenes, fuentes... [8]

Figura 3.4: Webpack

Babel

Es una herramienta que nos permite transformar nuestro código JS de última generación (o con funcionalidades extras) a JS que cualquier navegador o versión de Node.js entienda. [9]

Figura 3.5: Babel

Backend

- **MongoDb:** Se trata de un sistema NoSQL, es una base de datos distribuida, basada en documentos en formato BSON, que es una representación binaria de los objetos JSON. Además MongoDb permite una gran escalabilidad y flexibilidad en la gestión de la base de datos, beneficiando al sistema propuesto. [10]
- **Mongoose** Es una biblioteca de JavaScript que le permite definir esquemas con datos fuertemente tipados. [11]
- **Express:** Es el framework web más popular de Node, utilizado para construir el backend de un sitio utilizando las funciones y estructuras de NodeJS. [12]
- **Nodejs:** El entorno de tiempo de ejecución de JavaScript. Se utiliza para ejecutar JavaScript en una máquina en lugar de un navegador. [13]
- **JWT (JSON Web Token):** Es un estándar qué está dentro del documento RFC 7519. En el mismo se define un mecanismo para poder propagar entre dos partes, y de forma segura, la identidad de un determinado usuario, además con una serie de claims o privilegios. Estos privilegios están codificados en objetos de tipo JSON, que se incrustan dentro de del payload o cuerpo de un mensaje que va firmado digitalmente. Esto nos permite crear un control de acceso en la aplicación. [14]
- **Bcrypt:** Es una librería de encriptación que utiliza funciones hashing para encriptar contraseñas. Lleva incorporado un valor llamado salt, que es u fragmento aleatorio que se usará para generar el hash asociado a la password, y se guardará junto con ella en la base de datos. [15]
- **Moment:** Es una librería que nos permite un manejo sólido de fechas y horas. [16]

- **Morgan:** Emite un historial de peticiones HTTP request que se solicitan a nuestro aplicación.
- **Nodemon:** Se ha creado para facilitar el desarrollo en Node.js, permite que los cambios en el proyecto se integren en tiempo real en el servidor.

Frontend

- **Vuejs:** es un framework progresivo para construir interfaces de usuario. Tiene enlace de datos bidireccional que permite el desarrollo sin interrupciones de la interfaz junto con la capacidad MVC y aplicaciones interactivas del lado del servidor. [17]
- **Leaflet** Leaflet es una librería JavaScript open source ampliamente utilizada para la publicación de mapas en la web. [18]
- **Leaflet-Draw** Leaflet-Draw es una librería de Leaflet que proporciona una herramienta para dibujar polígonos o figuras en el mapa de leaflet. [19]
- **Axios:** Es un cliente HTTP basado en Promesas para Javascript. Se usará para las llamadas a la API del backend. [20]
- **Boostrap:** Es un framework CSS que te permitirá crear sitios web y aplicaciones web, se ha optado por el conjunto de herramientas Boostrap/BoostrapVue, ya que nos permite hacer un desarrollo adaptable, usable y ligero sin mayores complicaciones, además la documentación de la misma es lo suficiente extensa para su uso. [21]
- **Fontawesome:** Librería que proporciona iconos para un desarrollo visual más completo. [22]

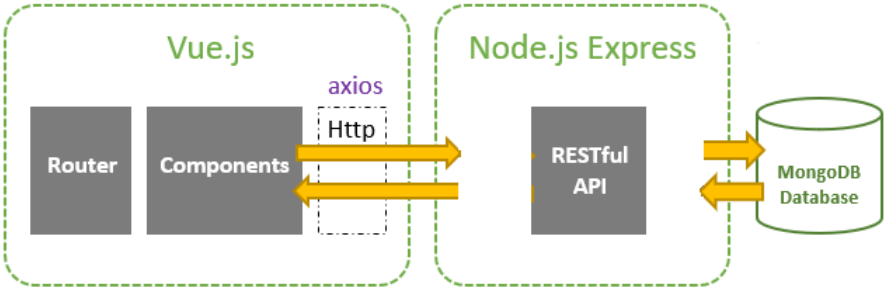


Figura 3.6: Esquema Stack MEVN

3.3. Metodología

Este proyecto de Fin de Grado se ha desarrollado usando la metodología de desarrollo de software XP o eXtreme Programming. Esta metodología es una de las metodologías más utilizadas en cuanto al desarrollo de software y se diferencia de las demás dado que se centra en la retroalimentación entre el cliente (en este caso, el tutor académico) y el equipo de desarrollo (en este caso, el alumno). [23]

Como cualquier otra metodología de desarrollo de software, su objetivo es mejorar la productividad de cualquier proyecto y reducir los riesgos y tiempos extras derivados de cambios en los requisitos del cliente. Para cumplir este objetivo, esta metodología define varios conceptos (ver figura 3.7):

- **Planificaciones**

  Se debe planificar diferentes plazos en el proyecto basándose en las exigencias del cliente, y en base a estas se haría una estimación del coste y la dificultad y se definirán las prioridades.

- **Pruebas**

  Continuamente se han de efectuar pruebas en base a los requisitos del cliente para comprobar que todo funciona correctamente. Estas deben hacerse de forma periódica y automática, de forma que al final de cada planificación, el software esté probado y funcionando correctamente.

- **Simplicidad**

  Al realizar reuniones con el cliente, se simplificará el desarrollo del proyecto, puesto que solo se desarrolla aquello que el cliente quiere, y además, éste participa en el.

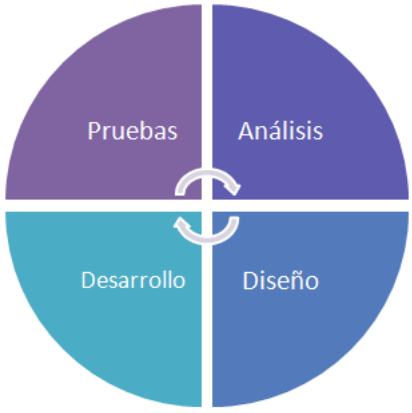


Figura 3.7: Esquema funcionamiento de la metodología XP

Capítulo 4

Diseño y desarrollo

GeoCanCanarias es una plataforma web con control de acceso para usuarios del 112 Canarias. Partiremos del siguiente diagrama de flujo para explicar el diseño y funcionamiento de la aplicación.

El proceso de acceso a la aplicación se representa con un usuario sin autenticar a la página de inicio de la aplicación, Si el individuo tiene cuenta con contraseña en la plataforma, podrá acceder a la herramienta del mapa geográfico, en caso contrario el administrador de la aplicación tendrá que crear un usuario para que pueda autenticarse. (ver esquema de procesos Figura 4.1)

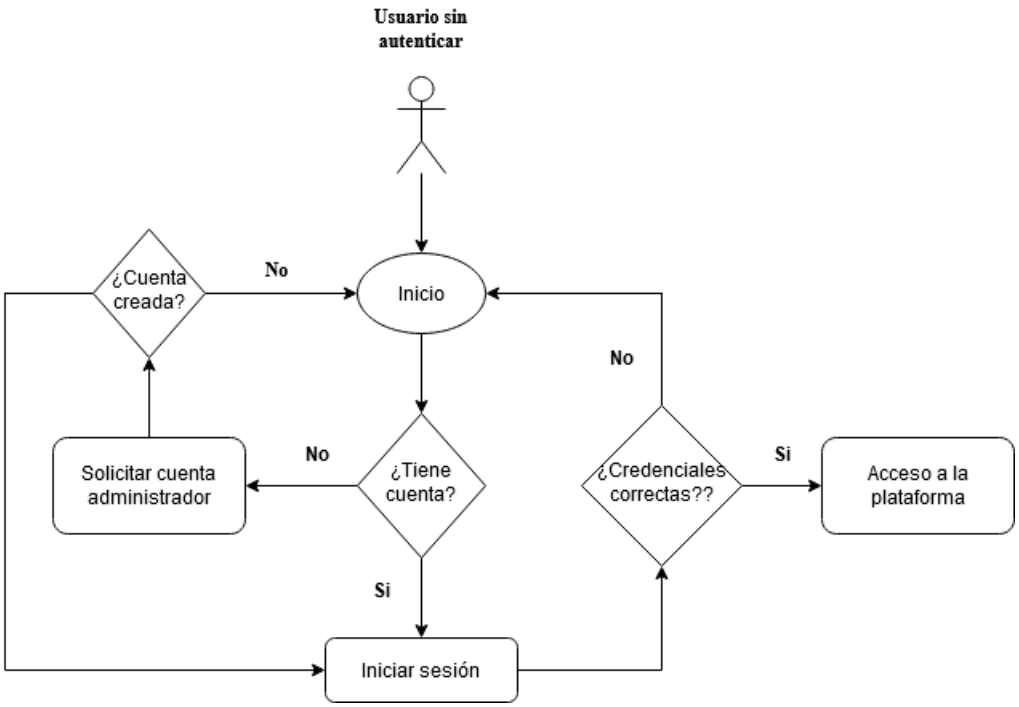


Figura 4.1: Diagrama de flujo acceso de a la plataforma

Los incidentes que se vayan generando se irán introduciendo en el mapa geográfico a medida que el operador de demanda vaya recopilando la

información necesaria, mostrando un marcador en el lugar de la incidencia.

Según el nivel de autorización, se podrá manipular los incidentes. El Gestor Operativo y el Coordinador Multisectorial tendrán la capacidad máxima de realizar funciones ya que es la autoridad máxima, los usuarios de los sectores implicados sólo podrán visualizar, filtrar y finalizar la incidencia cuando ésta haya terminado, los operadores de demanda sólo podrán visualizar y editar la información de los incidentes. ver la figura 4.2 para ver un esquema del flujo.

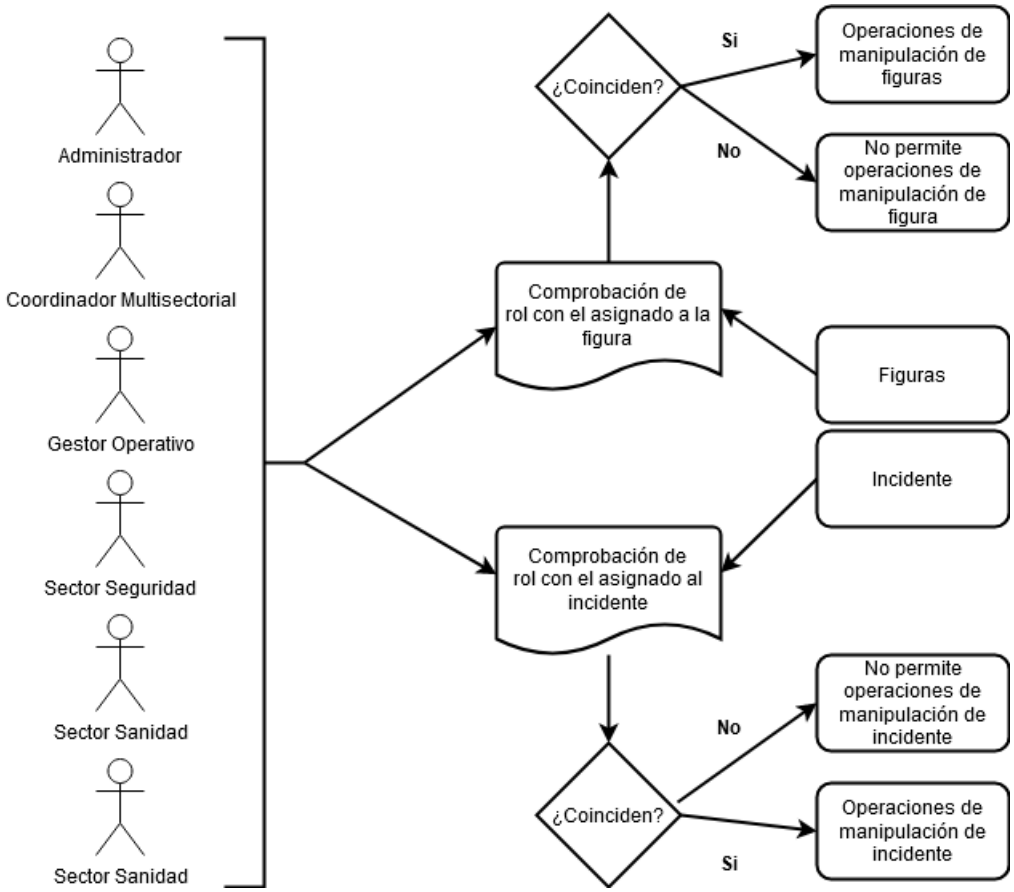


Figura 4.2: Diagrama de flujo Usuarios-Incidentes/figuras

Los incidentes se mostrarán mediante figuras representativas e iconos más acorde a las situación ocasionada. En este caso, lo provee la librería font-awesome, no obstante el siguiente artículo proporciona una información precisa sobre convenciones más comunes para el uso de la simbología en sistemas geográficos. [24]

4.1. Patrón de arquitectura

Antes de empezar cabe resaltar el patrón de arquitectura. MVC son las siglas de Model, View y Controller, que se traduce como modelo, vista y controlador [25]. Este patrón divide la aplicación en esas 3 secciones y cada una se encarga de realizar una función diferente, pero que todas tienen algo en común, que es manejar la lógica de negocio. [26]. En la figura 4.3 se muestra un esquema del mismo.

- La **Vista** muestra al usuario esa información procesada y manipulada.
- El **Modelo** guarda los datos.
- El **Controlador** procesa dicha información.

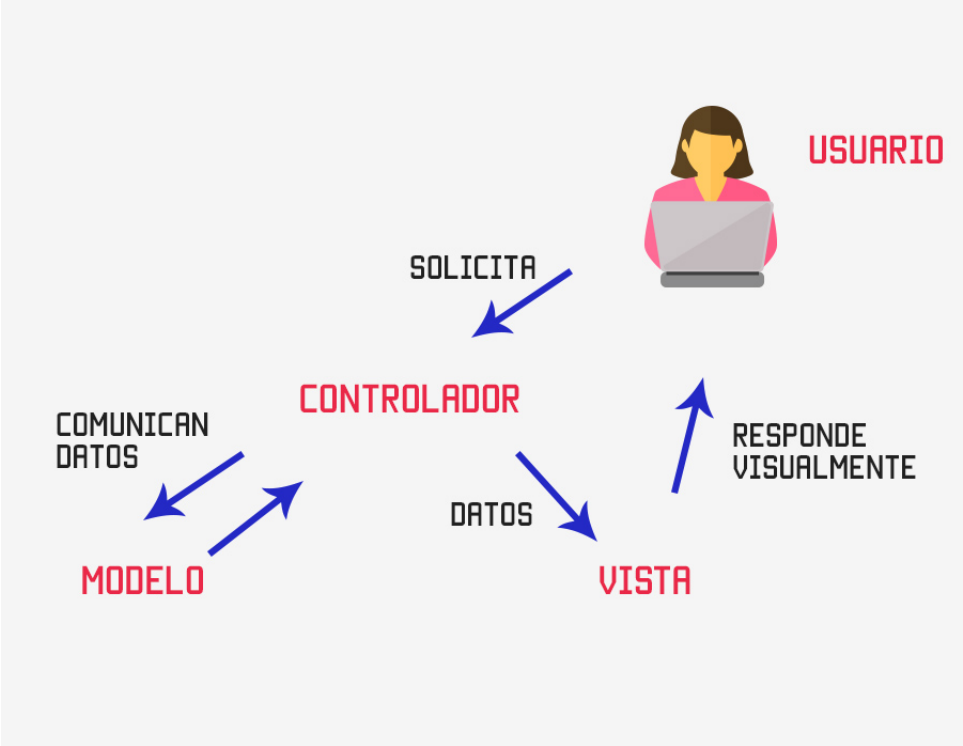


Figura 4.3: Esquema del MVC

4.2. Vista - Frontend

4.2.1. Página principal

La página principal de la plataforma guarda un diseño sencillo y minimalista gracias Boostrap y Flexbox, se compone una barra de navegación

oscura donde se albergarán las pestañas correspondientes en cada momento del sistema y un botón de inicio para redigirir a la página de inicio de sesión. [27] (ver figura 4.4)

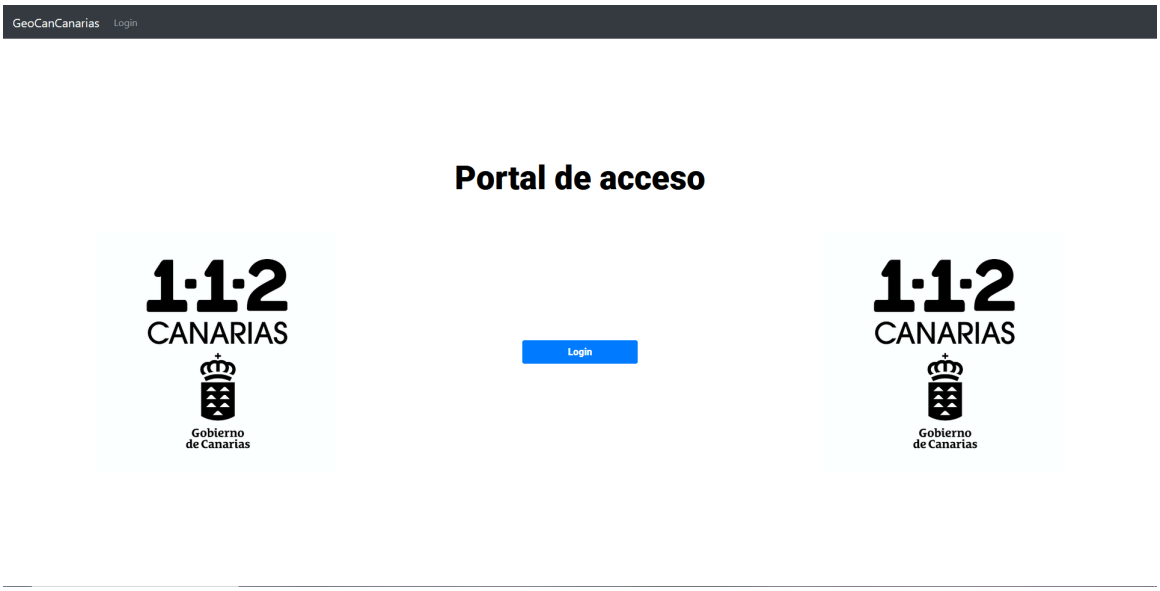


Figura 4.4: Página de inicio de la plataforma

4.2.2. Login

La página de login prosigue con el mismo diseño que la página principal, contiene un formulario de acceso a la plataforma. Si las credenciales son correctas se mostrará herramienta web geográfica, en caso contrario no se podrá acceder. (ver figura 4.5)

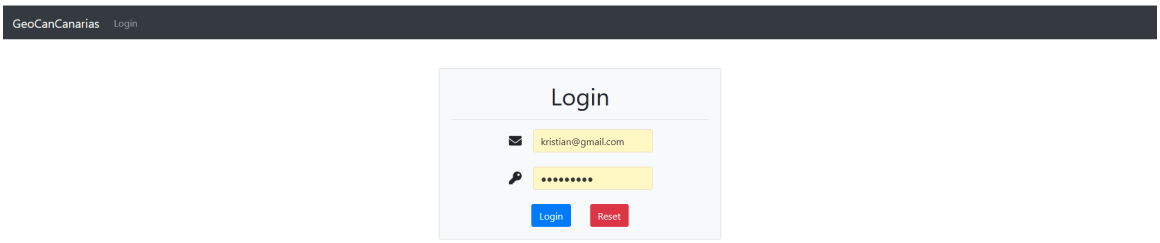


Figura 4.5: Página de login de la plataforma

4.2.3. Mapa

Es una de las pestañas clave de la herramienta, contiene los componentes necesario para el correcto funcionamiento del mapa como la visualización, borrado, edición y filtrado de los incidentes y las figuras. Gracias a la librería Leaflet, nos permite utilizar mapas geográficos y su sistema por capas que permiten añadir marcadores, polígonos e interactuar con ellos. [28] (ver figura 4.6)

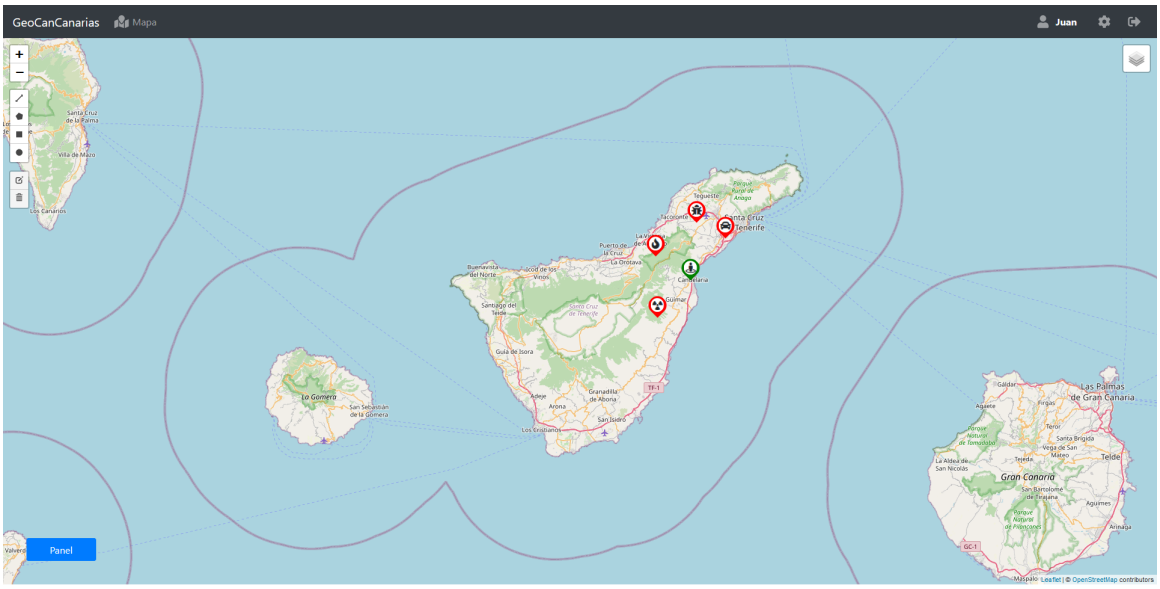


Figura 4.6: Mapa

Componentes

Filtro

Es la herramienta de filtrado de los IRNs. Se puede filtrar por:

- **Riesgo:** Según el riesgo que procesa el incidente (Alto-Medio-Bajo-Infomación)
- **Tipo de IRN:** por fenómenos Naturales, vehículos, materiales peligrosos entre otros
- **Localización:** Se establece un radio donde sólo se mostrarán los incidentes dentro del mismo.

En la figura 4.7 se muestra el formulario para realizar el filtrado deseado.

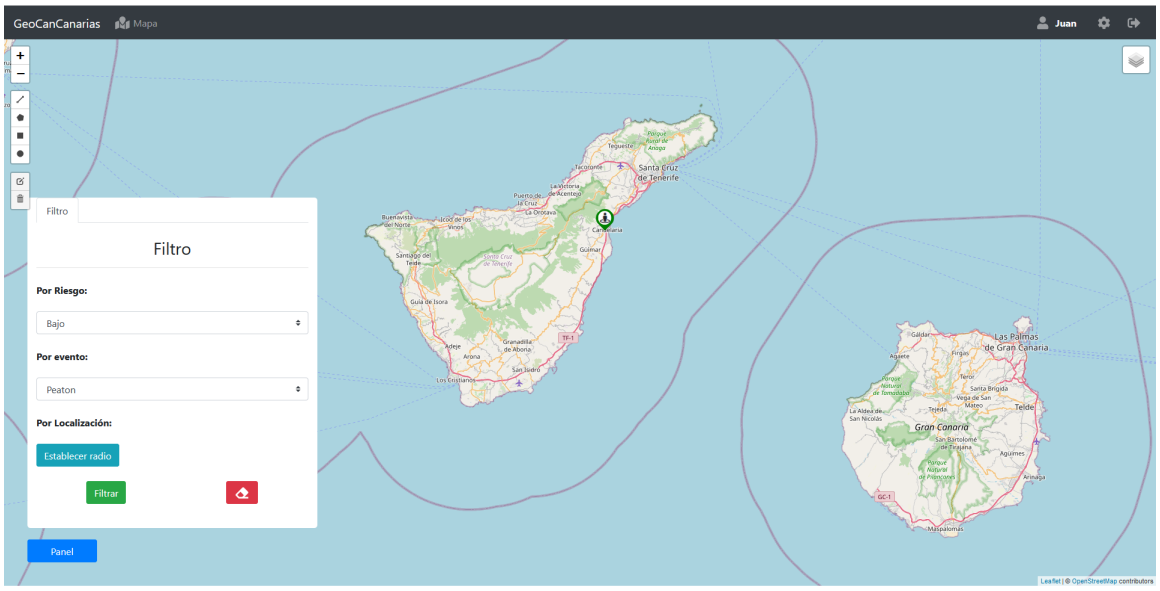


Figura 4.7: Filtro de incidentes

Añadir incidente

Para el caso de Operador de Demanda, se le habilitará la opción para introducir incidentes en el mapa. constará de un formulario donde se recopilarán datos como coordenadas del incidente, tipo de IRN, tipo de riesgo, agentes implicados en el incidente entre otros. (ver figura 4.8)

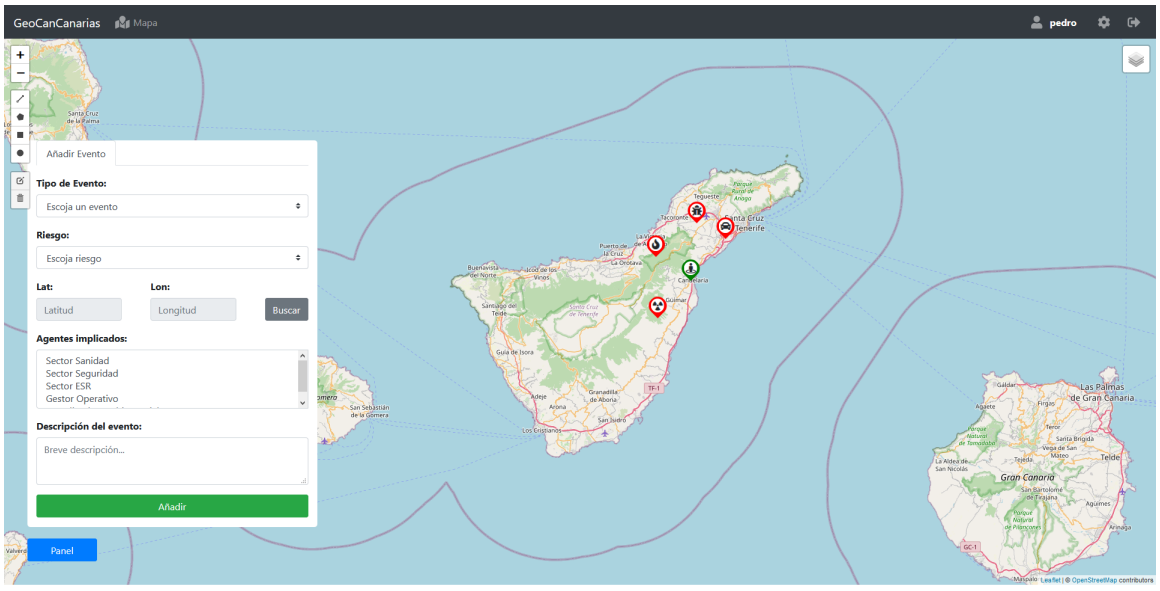


Figura 4.8: Formulario para añadir incidentes

Información del incidente

Pinchando en un marcador generado por el incidente, se muestra un panel con la información que perteneciente a éste, como nombre del IRN, descripción, riesgo. (ver figura 4.9)

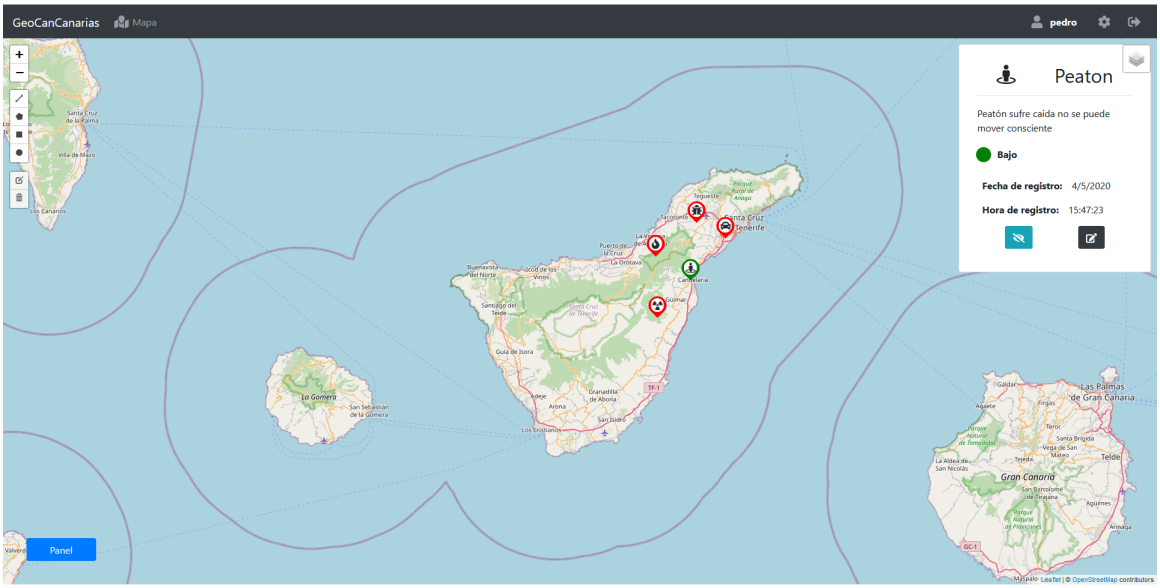


Figura 4.9: Panel de información del incidente

Panel de COVID-19

Panel que indica el número de casos positivos de COVID-19, fallecimientos y recuperaciones totales en España. Además muestra información clasificada por Comunidades Autónomas con la posibilidad de ocultar la información en caso de no quererla mostrar en el mapa. [29] (ver figura 4.10)

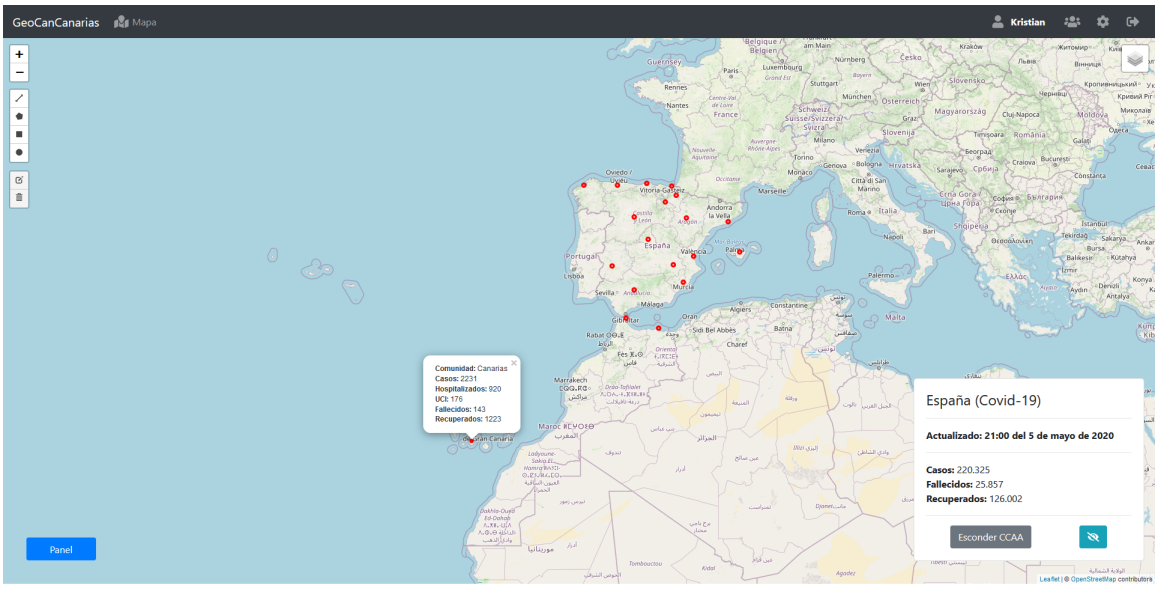


Figura 4.10: Datos COVID-19 España

4.2.4. Panel de Administrador

Contiene todo los usuarios de la plataforma y permite al administrador añadir o borrar usuarios. (ver figura 4.11)

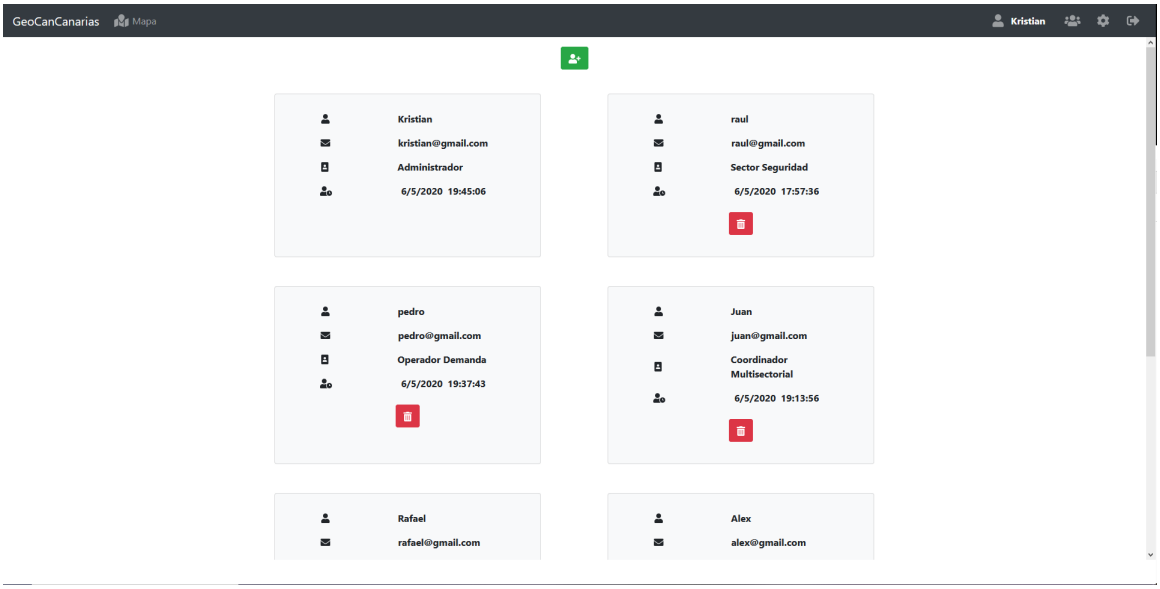


Figura 4.11: Panel de administrador

4.2.5. Información de la cuenta

Pestaña que proporciona una información de la cuenta del usuario que haya iniciado sesión. (ver figura 4.12)

Figura 4.12: Información de la cuenta de usuario

4.3. Vuex, Localstorage - Frontend

Vuex es una librería complementaria para Vue siguiendo el patrón Flux para centralizar el estado de la información, lo que permite que cuando cambie algún calor no se tenga que replicar entre los componentes de Vue.

En nuestro caso se utiliza para centralizar información como los incidentes, figuras recogidos de la base de datos o información del usuario entre otros. (ver figura 4.13)

```
state: {
  session: false,
  token: "",
  email: "",
  rol: "",
  name: "",
  marker: [],
  markerFilter: [],
  mapObject: null,
  shapes: [],
```

Figura 4.13: Ejemplos de estados Vuex

Localstorage es un sistema de almacenamiento que nos permite almacenar datos de manera local en el navegador y sin necesidad de realizar alguna conexión a una base de datos. Se almacena variables como el token validado, el email o el nombre del usuario así evitamos que el usuario tenga que introducir sus credenciales una y otra vez. (ver figura 4.14)

Key	Value
Email	raul@gmail.com
loglevel:webpack-dev-server	SILENT
Name	raul
Rol	Sector Seguridad
Token	eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI1ZWFmZjg0Y2ZjM

Figura 4.14: Ejemplo de información almacenada en el LocalStorage

4.4. Leaflet Toolbar - Frontend

Leaflet Toolbar proporciona interfaces de barra de herramientas flexibles y extensibles para mapas de Leaflet. Cada herramienta proporciona una

capa en el mapa, no obstante no asegura que la figura se mantenga una vez se refresque o apague la aplicación. Para ello, la capa se transforma en información GEOJSON que es un formato estándar abierto diseñado para representar elementos geográficos sencillos y se crea un identificador único para cada figura una vez se inserta en la base de datos. Una vez creada, se obtiene la figura en cuestión y se construye la capa en el mapa para su visualización. (ver figura 4.15)

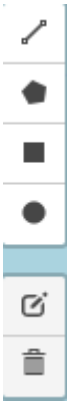


Figura 4.15: Herramienta de Leaflet

El código de la figura 4.16 transforma la capa en un archivo geoJSON y se guardan en dicho archivo los estilos de la figura (Con estilos nos referimos al color, opacidad, grosor de línea, etc.) y su información enviándola al backend para su posterior guardado en la base de datos. Cabe destacar que los archivos geoJSON no soportan los círculos, por tanto hay que introducir manualmente el radio del círculo en el archivo y manipularlo posteriormente para su mostrado en el mapa. (ver figura 4.16)

```
this.mapObject.on(window.L.Draw.Event.CREATED, e => {
  let json = e.layer.toGeoJSON();

  e.layer.options.color = this.colorShape;
  json.options = e.layer.options;
  json.ai = this.aiShape;

  this.axios
    .post("/signupshape", json)
    .then(async res => {
      let parent = this;
      parent.featureGroup.eachLayer(function(layer) {
        parent.featureGroup.removeLayer(layer);
      });
      await this.getShapes(this.featureGroup);
    })
    .catch(e => {
      console.log(e);
    });
});
```

Figura 4.16: Función al crear una figura

La forma de mostrar las figuras en el mapa se realiza con el siguiente fragmento de código ver 4.17. Primero se almacena las figuras en un array en la store de Vuex que correspondan con el rol del usuario. Una vez almacenados, se recoge dicho array y se recorre comprobando si la figura tiene el campo radio, en caso afirmativo, significa que se trata de un círculo, por tanto se procede a crear la capa con los estilos de la figura, sus coordenadas, el id y el radio almacenado manualmente, en caso contrario solo se crea la capa con los estilos que provee la figura y el id de la figura.

De esta manera conseguimos identificar cada figura al guardar y al recopilar la información de la base de datos e incluso poder modificar al momento sin perder información.

```
async getShapes(featureGroup) {
  await this.axios
    .get("/showshapes")
    .then(res => {
      this.$store.commit(
        "setShapes",
        res.data.shapes.filter(shape => {···
        })
      );
      let mylayer;
      this.collection = this.$store.getters.getShapes;
      for (let i = 0; i < this.collection.length; i++) {
        if (this.collection[i].shape.options.radius) {
          mylayer = L.geoJSON(this.collection[i].shape, {
            pointToLayer: (feature, latlng) => {
              feature.options.idShape = this.collection[i].idShape;
              return new L.Circle(latlng, feature.options);
            },
            onEachFeature: (feature, layer) => {
              layer.options.color = feature.options.color;
              layer.options.radius = feature.options.radius;
              this.featureGroup.addLayer(layer);
            }
          });
        } else {···
        }
        function pointToLayer(feature, latlng) {
          return new L.Circle(latlng, feature.options);
        }
      }      You, a month ago • Muchos cambios
    })
    .catch(err => {
      console.log(err);
    });
},
```

Figura 4.17: Función para mostrar las figuras

4.5. Router - Frontend

El router en el frontend se encarga de direccionar las rutas de las vistas necesarias para el correcto renderizado de la misma. En la aplicación se le aplica un campo a cada ruta denominado isAuth, que determina si es posible la redirección en caso de tener autorización. (ver figura 4.18)

Figura 4.18: Esqueleto del router

La función encargada de determinar si se obtiene la autorización pertinente para la redirección está contenida en la siguiente figura 4.19

Siempre se ejecuta antes de usar al router, primero comprueba si hay un Token en la localstorage del navegador, en caso negativo redirige a la vista Login, en caso afirmativo comprueba con una llamada al backend si dicho token es válido y si lo es guarda en el localstorage el email del usuario en cuestión.

```
router.beforeEach(async (to, from, next) => {
  if (to.matched.some((record) => record.meta.isAuth)) {
    try {
      if (localStorage.getItem("Token") == null) {
        next({
          path: "/login",
        });
      } else {
        const res = await axios.get(`http://localhost:3000/api/isAuth`, {
          headers: {
            authorization: `Bearer ${localStorage.getItem("Token")}`,
          },
        });

        if (res.data.auth) {
          store.commit("setEmail", localStorage.getItem("Email"));
          next();
        }
      }
    } catch (err) {
      console.log(err);
      if (err.response.status === 403) {
        next({
          path: "/login",
        });
      }
    }
  } else {
    next();
  }
});
```

Figura 4.19: Función del router

4.6. Configuración del Backend

Para la configuración del backend, haremos uso de las tecnologías comentadas anteriormente. Como se puede ver en la figura , se hace uso de middleswares que proporciona Express como es el caso de Cors que permite realizar solicitudes de un servidor externo e impedir el bloqueo por CORS, JSON y urlencoded para la configuración de las respuestas de la api. (ver Figura 4.20)

```
app.use(morgan("tiny"));
app.use(cors());
app.use(bodyParser.urlencoded({ extended: true }));
app.use(bodyParser.json());
app.use("/api", api);

// Middleware para Vue.js router modo history
app.use(history());
app.use(express.static(path.join(__dirname, "public")));

mongoose.connect(config.db, config.options).then(
  () => {
    console.log("Conectado a DB");
    // app.set('puerto', config.port);
    // app.listen(app.get('puerto'), function() {
    //     console.log(
    //         'Escuchando por el puerto ' + app.get('puerto'),
    //     );
    // });
    server.listen(config.port, () => {
      console.log("Escuchando por el puerto ", config.port);
    });
  },
  (err) => {
    console.log(err);
  }
);
```

Figura 4.20: Configuración de la raiz del backend

Se realiza una conexión a la base de datos gracias al archivo de configuración creado en la raiz del backend (ver figura 4.21) que contiene un puerto establecido, así como la ruta donde se almacenan la base de datos y sus colecciones y por último la clave secreta para los tokens. [30]

```
module.exports = {
    port: process.env.PORT || 3000,
    db: process.env.MONGODB_URI || 'mongodb://localhost:27017/myapp',
    options: {
        useNewUrlParser: true,
        useCreateIndex: true,
        useUnifiedTopology: true,
    },
    SECRET_TOKEN: 'miclavedetokens',
};
```

Figura 4.21: Datos para la configuración del backend

4.7. Modelos - Backend

4.7.1. Usuario

Contiene toda la información del usuario: email, nombre, rol que desempeña, contraseña, fecha de registro y el último acceso a la plataforma. El

identificador de cada usuario se hará mediante el email. (ver figura 4.22)

```
const UserSchema = new Schema({
  email: { type: String, unique: true, lowercase: true, select: true },
  displayName: { type: String, select: true },
  rol: {
    type: String,
    enum: [
      "Administrador",
      "Gestor Operativo",
      "Coordinador Multisectorial",
      "Operador Demanda",
      "Sector Sanidad",
      "Sector Seguridad",
      "Sector ESR",
      "Visitante",
    ],
    select: true,      You, a few seconds ago • Uncommitted changes
  },
  password: { type: String, select: true },
  signupDate: { type: Date, default: Date.now() },
  lastLogin: Date,
});
```

Figura 4.22: Modelo de usuario

4.7.2. Incidente

El modelo de incidente constituye el esqueleto de la información introducida por el usuario: tipo de evento, un icono que representa de forma visual el incidente en el mapa, una descripción del mismo, el tipo de riesgo, la fecha y hora de registro del incidente, las coordenadas dónde se sitúa y los agentes implicados en el evento.

Cada evento se identifica de manera única mediante su id que se autoincrementa a medida que se añadan más incidentes a la base de datos. (ver figura 4.23)

```
const AutoIncrement = require("mongoose-sequence")(mongoose);

const eventSchema = new Schema(
  {
    //El id se genera automaticamente con el Autoincrement
    type_event: { type: String },
    icon: { type: String },
    description: { type: String },
    risk: String,
    signupEvent: { type: Date, default: Date.now() },
    lat: Number,
    lon: Number,
    ai: {
      type: [String],
      enum: [
        "Sector Sanidad",
        "Sector Seguridad",      You, a few seconds ago • Uncom
        "Sector ESR",
        "Gestor Operativo",
        "Coordinador Multisectorial",
      ],
    },
  },
  { idEvent: false }
);

eventSchema.plugin(AutoIncrement, { inc_field: "idEvent" });
```

Figura 4.23: Modelo de Incidente

4.7.3. Figura

El modelo figura contiene los campos necesarios para poder almacenarla en la base de datos: shape contiene la figura completa, sus coordenadas, opciones de estilo, como color, grosor de la línea de contorno, etc. los agentes implicados en los que mostrar dicha figura en el mapa.

Como se comentó anteriormente, cada figura dispone de una identificación propia una vez introducida en la base de datos. Esto facilita la edición o el borrado de las figuras una vez pintadas en el mapa. (ver figura 4.24)

```
const AutoIncrement = require("mongoose-sequence")(mongoose);

const drawSchema = new Schema(
  {
    //El id se genera automaticamente con el Autoincrement
    shape: { type: Object },
    ai: {
      type: [String],
      enum: [
        "Sector Sanidad",
        "Sector Seguridad",
        "Sector ESR",
        "Gestor Operativo",
        "Coordinador Multisectorial",
      ],
    },
  },
  { idShape: false }
);

drawSchema.plugin(AutoIncrement, { inc_field: "idShape" });
```

Figura 4.24: Modelo de Figura

4.8. Controladores - Backend

4.8.1. Usuario

Contiene las funcionalidades que puede optar los usuarios (ver figura 4.25), entre ellas están:

- Inicio de Sesión
- Registro de usuario
- Mostrar usuarios
- Eliminar usuario
- Mostrar información del usuario
- Último acceso a la plataforma

```
function signUp(req, res) {
  const user = new User({
    email: req.body.email,
    displayName: req.body.displayName,
    password: req.body.password,
    rol: req.body.rol,
  });

  user.save((err) => {
    if (err)
      return res.status(500).send({
        message: `Error al crear el usuario: ${err}`,
      });      You, a month ago • Muchos cambios

    return res
      .status(201)
      .send({ message: "El usuario se ha registrado con éxito" }); //.send({ token: services.createToken(user) });
  });
}

function signIn(req, res) {···
}

function deleteUser(req, res) {···
}

function showUsers(req, res) {···
}

function showInfo(req, res) {···
}

function editUser(req, res) {···
}

function changelastLogin(email) {···
}
```

Figura 4.25: Función de registro del controlador Usuario

4.9. Router - Backend

El router en el backend crea manejadores de rutas montables y modulares. En la aplicación se encarga de gestionar las rutas de las peticiones recibidas por el frontend y transferirlas al controlador correspondiente. (ver figura 4.26)

- **POST:** Se encargan de introducir objetos en la base de datos como usuarios, figuras o incidentes. En caso del inicio de sesión los datos se envía en un paquete para evitar vulnerabilidades de seguridad.
- **GET:** Se encargan de recopilar datos como los incidentes, figuras o usuarios.
- **PUT:** Se encargan de editar la información ya almacenada.
- **DELETE:** Se encargan de eliminar los datos requeridos cómo algún usuario en concreto.

```
// POST

api.post("/signup", userCtrl.signUp);      You, a month ago • Muchos cambios
api.post("/signupevent", eventCtrl.signUpEvent);
api.post("/signupshape", drawCtrl.signUpShape);
api.post("/signin", userCtrl.signIn);

// PUT

api.put("/editshapes", drawCtrl.editShapes);
api.put("/editevent", eventCtrl.editEvent);
api.put("/edituser", userCtrl.editUser);

// DELETE

api.delete("/deleteshapes", drawCtrl.deleteShapes);
api.delete("/deleteinfo", eventCtrl.deleteInfo);
api.delete("/deleteuser", userCtrl.deleteUser);

// GET

api.get("/showshapes", drawCtrl.showShapes);
api.get("/showevents", eventCtrl.showEvents);
api.get("/showinfo", userCtrl.showInfo);
api.get("/showusers", userCtrl.showUsers);
```

Figura 4.26: Función del router en el backend

4.9.1. Incidente

Contiene las funcionalidades que puede optar los incidentes (ver figura 4.27), entre ellas están:

- Registro de incidente
- Eliminar incidente
- Editar incidente
- Mostrar incidentes

Figura 4.27: Función de mostrar incidentes del controlador

4.9.2. Figura

Contiene las funcionalidades que puede optar las figuras muy similares a los incidentes (ver figura 4.28), entre ellas están:

- Registro de figura
- Eliminar figuras
- Editar figuras
- Mostrar figuras

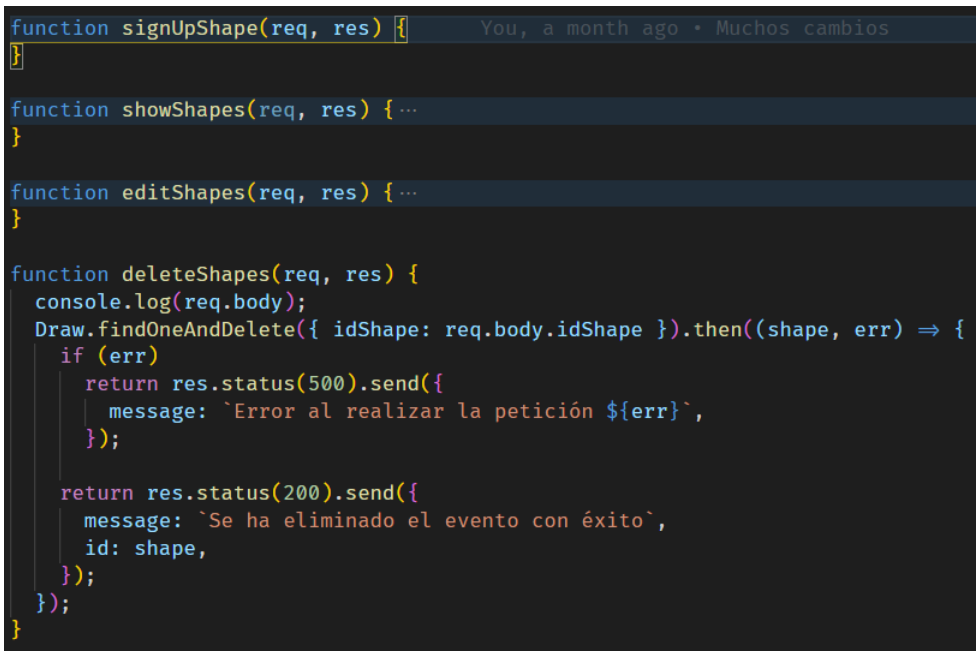


Figura 4.28: Función de borrado del controlador figuras

4.10. Servicios - Backend

Tokens

Se trata de los servicios que integra el backend para distinguir entre un token sea válido o no.

- **CreateToken:**

  Se trata de una función que se encarga de generar el token gracias a una palabra secreta definida anteriormente.

- **DecodeToken:**

  Es una función que se ocupa de decodificar el token con la palabra secreta, si la fecha del token no está caducada y la codificación coincide, el token es válido. (ver figura 4.29)

```
function createToken(user) {···
}

function decodeToken(token) {
  const decoded = new Promise((resolve, reject) => {
    try {
      const payload = jwt.decode(token, config.SECRET_TOKEN);
      resolve({
        payload: payload.sub,
        message: "Login correcto",
        status: 200,
        auth: true,
      });
    } catch (err) {
      let status = 403; //El status por defecto es 403 (Forbidden) pero cambia a 401 (Unauthorized) si el token ha expirado
      if (err.message == "Token expired") {
        status = 401;
        err.message = "El token ha expirado";
      }
      reject({
        status: status,
        message: err.message,
        auth: false,
      });
    }
  });
  return decoded;
}
```

Figura 4.29: Función DecodeToken que indica si el token es válido

Ministerio de Sanidad

Actualización diaria de la situación de COVID-19 en España, características epidemiológicas de los casos de COVID-19 (referidos siempre a casos con confirmación virológica por PCR), así como de indicadores de evolución de la pandemia. Resultados obtenidos a partir de la notificación agregada diaria de las CCAA al Ministerio de Sanidad, y de la información individualizada de las CCAA a la Red Nacional de Vigilancia Epidemiológica (RENAVE) (datos individualizados RENAVE). No obstante, la información puede estar sujeto a cambios y variaciones que puedan inferir en la aplicación.

En la figura 4.30, se presenta la funcion donde se hace una llamada a la URL del Ministerio de Sanidad de España para obtener los datos correspondientes, como los casos positivos, fallecimientos o personas ingresadas en la UCI, entre otros. Se utilizan promesas para la obtención de los datos, que es simplemente un objeto que puede o no devolver algún valor en la línea de tiempo presente y futuro.

```
async function callApiDataCovid() {
  let objectResponse = {
    globalTotalData: {
      FECHA: "",
      casosTotales: 0,
      hospitalizadosTotales: 0,
      fallecimientosTotales: 0,
      uciTotales: 0,
    },
    ccaa: "",
    record: "",
  };

  let dataPromise = new Promise((resolve, reject) => {
    setTimeout(
      resolve,
      100,
      axios.get("https://cnecovid.isciii.es/covid19/resources/agregados.csv", {
        headers: {
          "Content-Type": "application/octet-stream",
        },
      })
    );
  });

  await Promise.all([dataPromise])
    .then((values) => {
      //objectResponse.globalTotalData = csvToArray(values[0].data);      You, a
      objectResponse.record = csvToArray(values[0].data);
      //objectResponse.ccaa = csvToArray(values[2].data);
    })
    .catch((err) => {
      console.log(err);
    });

  const NUM_CCAA = 19;

  objectResponse.record = objectResponse.record.slice(
    Math.max(objectResponse.record.length - NUM_CCAA, 1)
  );

  objectResponse.record = objectResponse.record.slice();

  for (let i = 0; i < objectResponse.record.length; i++) {···
  }

  return objectResponse;
}
```

Figura 4.30: Función de llamada al Ministerio de Sanidad

Una vez obtenido los datos, se procesan con la función csvToArray de la figura 4.31. Se encarga de parsear los datos obtenidos en las promesas, de un modo que sea legible para la aplicación y ayuden a la visualización de la misma en la plataforma.

```
function csvToArray(data) {
  const onlyData = data.split('"NOTA')[0];
  const arrayDatos = onlyData.split("\n").filter((dato) => dato.length !== 0);
  const keys = arrayDatos.shift().split(",");
  const comunidades = arrayDatos
    .splice(0)
    .map((comunidad) => comunidad.split(","));
  const result = comunidades.map((data) => {
    const object = {};
    keys.forEach((key, index) => (object[key] = data[index]));
    return object;
  });
  return result;
}
```

Figura 4.31: Función de parseo de los datos obtenidos por el Ministerio de Sanidad

4.11. Seguridad - Backend

Para abordar la seguridad en la plataforma, se ha implementado una función hashing de contraseñas contenida en la librería Bcrypt anteriormente comentada. La función contiene un valor llamado salt, que es un fragmento

de dato aleatorio (Generalmente un valor) que se usa para generar el hash asociado a la password y se guarda junto con ella en la base de datos. Así evita que dos passwords iguales generen el mismo hash y los problemas que ello conlleva.

En el proceso de desencriptado funciona de la siguiente manera: El usuario otorga una contraseña, el sistema recoge la contraseña encriptada de la base de datos y realiza el hash de la nueva contraseña con la salt almacenada en la base de datos, si ambas coinciden, significa que la contraseña es válida, de lo contrario la función negará la comparación. (ver figura 4.32)

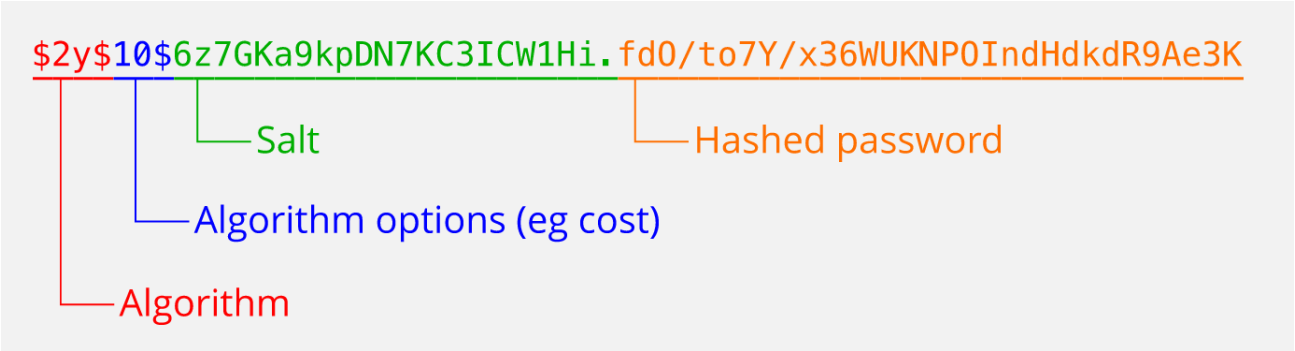


Figura 4.32: Hash de una contraseña

Otra medida de seguridad es la aplicación de roles a los distintos usuarios, cada usuario tiene un rol en la plataforma que le permite o no manipular con los elementos de la aplicación.

4.12. Pruebas

Se testearon las rutas del backend con el programa POSTMAN, que es una plataforma de colaboración para el desarrollo API. Una vez implementadas las funcionalidades del controlador, se testeaban con una petición al backend con el programa. Como se puede ver en la figura 4.33 se coloca en el body los datos a enviar al backend y se realiza la petición a la URL que corresponda. el resultado es un objeto JSON con los datos e información procesados desde backend. Esto ahorraba tiempo y código en el frontend.

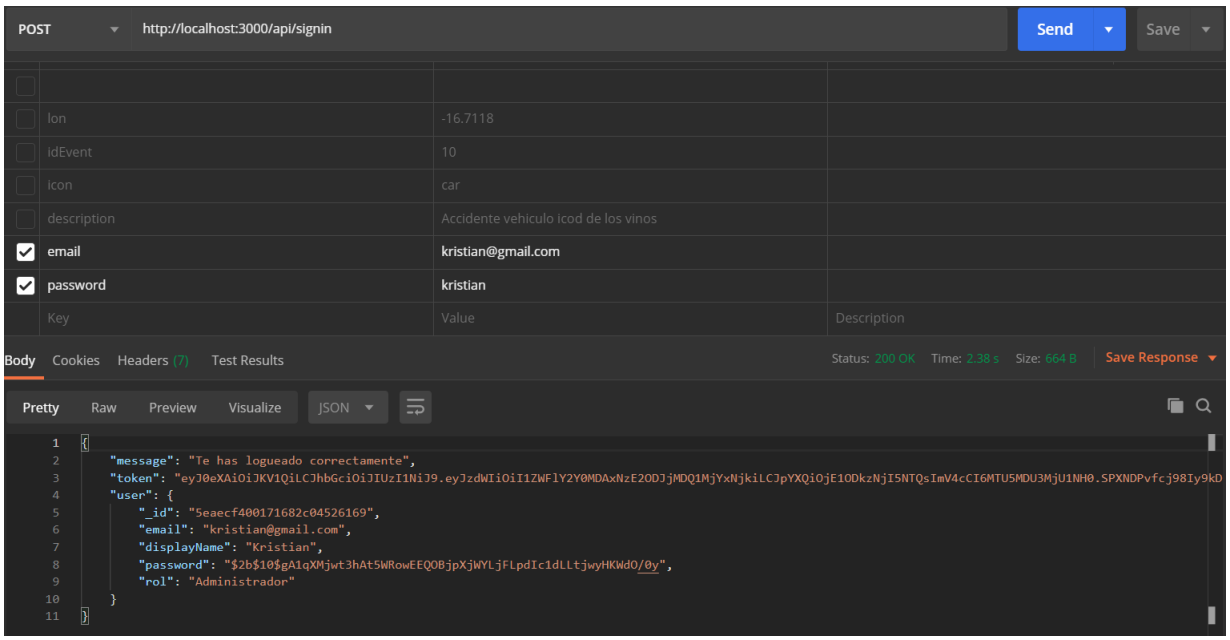


Figura 4.33: Petición POST desde el programa POSTMAN

En el caso de frontend, al usar un framework que contiene un sistema de reactividad. Se realizaban cambios en el código fuente y se comprobaba los cambios realizados de manera casi instantáneas en el navegador.

Capítulo 5

Conclusiones y líneas futuras

En términos generales, se han cumplido los objetivos principales de este proyecto. No obstante, la falta de tiempo evita que se puedan realizar mejoras que incrementarían el rendimiento y la calidad de la plataforma. Como por ejemplo implementar la tecnología sockets para una visión en tiempo real en cualquier sesión del usuario en cuanto a la visualización o manipulación de los incidentes y figuras.

Esta aplicación se ha diseñado con frameworks y librerías open source para que el uso sea de libre licencia y los desarrolladores puedan modificar el código para posibles mejoras de rendimiento y de funcionalidad.

No dudo que puede llegar a ser una aplicación potente para las instituciones canarias, pues el objetivo es que esta herramienta sirva para una mayor gestión de los incidentes que ocurren cada día en el archipiélago canario. Con mantenimiento y unas mejoras en las funcionalidades podrían llegar a ser una aplicación bastante útil.

Capítulo 6

Summary and Conclusions

In general terms, the main goals of this project have been reach. However, the lack of time prevents improvements that can be made that increase the performance and quality of the platform. For example, implement the sockets technology for a real-time view in any user session regarding the detection or modification of incidents and figures.

This application has been designed with open source frameworks and libraries for the use of the license free, users and developers can modify the code for possible performance and functionality improvements.

I have no doubt that it can become a powerful application for the Canary Islands institutions, as the objective is this tool will serve to better manage the incidents that occur every day in the Canary archipelago. With the maintenance and the improvements in the functionalities it could become a quite useful application.

Capítulo 7

Presupuesto

Este capítulo recoge una propuesta de presupuesto que estima el coste del trabajo según el tiempo empleado en este proyecto. No es necesaria ninguna infraestructura en donde se desplieguen las herramientas desarrolladas, el coste total de este proyecto constaría del sueldo al programador durante el tiempo de desarrollo.

Tareas	Horas	Presupuesto
Investigación y conceptos previos	20 horas	7€/hora
Análisis de requisitos	5 horas	7€/hora
Investigación de las tecnologías a usar	15 horas	7€/hora
Diseño de la aplicación	20 horas	30€/hora
Desarrollo del lado del servidor	50 horas	30€/hora
Desarrollo del lado del cliente	70 horas	30€/hora
Pruebas	30 horas	10€/hora
Total	210 horas	4.780€

Tabla 7.1: Resumen de tipos

Bibliografía

[1] 112-Canarias. http://www.112canarias.com/info/. Último acceso 2020-05-19

[2] Cova, Thomas J., GIS in emergency management. Geographical information systems 2.12, 1999.

[3] Goodchild, Michael F., Glennon, J. Alan, Crowdsourcing geographic information for disaster response: a research frontier. International Journal of Digital Earth 3.3: 231-41, 2010.

[4] Calvo, Kenneth, Durán, Johan, Quirós, Esteban, Malinowski, Elzbieta. *MongoDB: alternativas de implementar y consultar documentos.* IX Congreso Internacional de Computación y Telecomunicaciones, COMTEL, Lima. 2017.

[5] Icaza Álvarez, Daniel. EJE 07-10 Sistemas de seguridad ciudadana por georeferenciación y geolocalización para zonas rurales del cantón Cuenca incorporados al SIS ECU 9-1-1 del Ecuador. Memorias Universidad Del Azuay XVI: 413-418, 2017.

[6] Visual Studio Code.. https://code.visualstudio.com/. Último acceso 2020-05-19

[7] NPM https://www.npmjs.com/. Último acceso 2020-05-19

[8] WEBPACK. https://webpack.js.org/. Último acceso 2020-05-19

[9] Babel. https://babeljs.io/. Último acceso 2020-05-19

[10] MongoDb. http://www.mongodb.com/ . Último acceso 2020-05-19

[11] Mongoose. https://mongoosejs.com/. Último acceso 2020-05-19

[12] Express. http://www.express.com/ . Último acceso 2020-05-19

[13] Nodejs. http://www.nodejs.com/. Último acceso 2020-05-19

[14] JSON Web Tokens. https://jwt.io/ . Último acceso 2020-05-19

[15] Bcrytp. https://github.com/dcodeIO/bcrypt.js. Último acceso 2020-05-19

[16] Momentjs. https://momentjs.com/. Último acceso 2020-05-19

[17] Vuejs. https://vuejs.org/. Último acceso 2020-05-19

[18] Leaflet. https://leafletjs.com/. Último acceso 2020-05-19

[19] LeafletDraw. http://leaflet.github.io/Leaflet.draw/docs/. Último acceso 2020-05-19

[20] Axios. https://github.com/axios/axios. Último acceso 2020-05-19

[21] Boostrap. https://bootstrap-vue.org/. Último acceso 2020-05-19

[22] Fontawesome. https://fontawesome.com/. Último acceso 2020-05-19

[23] Andrés Navarro Cadavid, Juan Daniel Fernández Martínez, Jonathan Morales Vélez, Revisión de metodologías ágiles para el desarrollo de software. Prospect. Vol. 11, No. 2, Julio - Diciembre de 2013.

[24] Robinson, Anthony C. , Pezanowski, Scott, Troedson, Sarah, Bianchetti, Raechel, Blanford, Justine, Stevens, Joshua, Guidero, Elaine, Roth, Robert E. , MacEachren, Alan M. *Symbol Store: sharing map symbols for emergency management. Cartography and Geographic Information Science*, 40(5), 415-426, 2013.

[25] YDíaz González, Yanette, Fernández Romero, Yenisleidy. *Patrón Modelo-Vista-Controlador.* Revista Telemática 11.1: 47-57, 2012.

[26] Modelo-Vista-Controlador. https://codigofacilito.com/articulos/mvc-model-view-controller-explicado. Último acceso 2020-05-19

[27] Gore, Anthony. *Full-Stack Vue. js 2 and Laravel 5: Bring the frontend and backend together with Vue, Vuex, and Laravel.* Packt Publishing Ltd, 2017.

[28] Santiago, Marcos, Reyna, Marias de. *OpenLayers y Leaflet: Code-Combat. VIII Jornadas de SIG Libre.* http://www.sigte.udg.edu/jornadassiglibre/ Último acceso 2020-05-19

[29] Ministerio de Sanidad. https://www.mscbs.gob.es/profesionales/saludPublica/ccayes/alertasActual/nCov-China/situacionActual.html. Último acceso 2020-05-28

[30] Rigaux, Philippe, Scholl, Michel, Voisard, Agnes. *Spatial Databases with application to Gis*. Elsevier, 2001