



POLITÉCNICA

UNIVERSIDAD POLITÉCNICA DE MADRID

**ESCUELA TÉCNICA SUPERIOR DE INGENIEROS EN TOPOGRAFÍA,
GEODESIA Y CARTOGRAFÍA**

**TITULACIÓN DE GRADO EN INGENIERÍA DE LAS TECNOLOGÍAS DE LA
INFORMACIÓN GEOESPACIAL**

TRABAJO FIN DE GRADO



**SERVICIO WEB DE SEGUIMIENTO EN TIEMPO REAL DEL
VOLCÁN DE TAJOGAITE (LA PALMA, ESPAÑA) CENTRADO
EN INTEGRAR MAPAS INTERACTIVOS, DATOS DE CALIDAD
DEL AIRE E INCIDENCIAS Y ALERTAS**

Madrid, junio 2024

Alumno: Andrés Sebastián Chasiluisa Diaz

Tutores: Calimanut-Ionut Cira

Ramón Pablo Alcarria Garrido

TABLA DE CONTENIDOS

I.	Tabla de contenidos.....	2
II.	Agradecimientos.....	6
III.	Resumen.....	7
IV.	Abstract.....	8
V.	Índice de figuras.....	9
VI.	Índice de tablas.....	14
VII.	Acrónimos.....	15
1.	Introducción.....	16
1.1	Contexto y justificación	16
1.2	Motivación	17
1.3	Descripción general del proyecto	17
1.4	Estructura del documento.....	18
2.	Objetivos y alcance.....	19
2.1	Objetivos	19
2.2	Alcance.....	19
2.3	Partes interesadas	20
2.4	Proyectos relacionados.....	21
3.	Tecnologías utilizadas en soluciones del estado del arte.....	22
3.1	Tecnologias de gestión de datos.....	22
3.1.1	Pentaho Data Integration.....	22
3.1.2	MongoDB.....	23
3.1.3	Geoserver	23
3.2	Tecnologias implementadas en proyectos node js	24
3.2.1	Back-End.....	24
3.2.2	Front-End	27
3.3	Tecnologias de entorno	31

3.3.1	Docker	31
3.3.2	Quantum Geographic Information System (QGIS).....	32
3.3.3	Visual Studio Code (VS Code)	32
3.3.4	GIT	33
3.3.5	Postman	33
4.	Material y metodología.....	34
4.1	Datos	34
4.1.1	Fuentes de datos	34
4.1.2	Base de dato MongoDB	35
4.1.3	Publicación y servicio de capas Geoserver	51
4.1.4	Archivo de terremotos.....	53
4.2	Hardware	54
4.3	Software	55
4.4	Metodología	55
4.4.1	Estudio del marco del proyecto	55
4.4.2	Planteamiento de requisitos.....	55
4.4.3	Planificación y ejecución del desarrollo de la aplicación.....	56
4.4.4	Redacción de la memoria y entregables	56
5.	Extracción e integración de datos	57
5.1	Estaciones de calidad del aire.....	57
5.1.1	Modelo de datos de entrada.....	57
5.1.2	Extracción de registros de estaciones de calidad del aire	61
5.1.3	Historial de registros de estaciones de calidad del aire	63
5.2	Incidencias de la DGT	65
5.2.1	Modelo de datos de entrada.....	66
5.2.2	Borrado de incidencias obsoletas	71
5.2.3	Extracción de incidencias DGT.....	73

5.3	Extracción de Registros de terremotos	77
5.4	Ejecucion de procesos	77
5.4.1	Actualización de incidencias y registros de medición.....	77
5.4.2	Actualización de terremotos	78
6.	Requisitos del software.....	80
6.1	Requisitos funcionales.....	80
6.1.1	Requisitos de la aplicación	80
6.1.2	requisitos según rol de usuario	100
6.2	Requisitos no funcionales.....	122
7.	Arquitectura del sistema	123
7.1	Adquisición de datos	124
7.2	Almacenamiento de datos	124
7.3	Arquitectura Cliente-servidor.....	124
7.3.1	Estructura del Servidor	125
7.3.2	Estructura del Cliente	126
8.	Proceso del desarrollo.....	130
8.1	Metodología ágil y Scrum	130
8.2	Hitos alcanzados durante el desarrollo.....	130
9.	Interfaz de usuario	144
9.1	Diseño Visual	144
9.1.1	Gama de color	144
9.1.2	Tipografía	145
9.1.3	Iconografía de la aplicación	145
9.1.4	Simbología del mapa.....	148
9.2	Arquitectura de la Interfaz.....	152
9.3	Componentes de la interfaz	153

9.4	Interactividad y dinamismo	158
10.	Resultados y logros.....	160
10.1	Revisión de objetivos	160
10.2	Entregables	161
11.	Presupuesto.....	162
11.1	Hardware	162
11.2	Software	162
11.3	Producción.....	164
11.4	Presupuesto total del desarrollo.....	164
11.5	Coste de despliegue de aplicación.....	165
12.	Conclusiones.....	167
12.1	Futuras mejoras y recomendaciones.....	167
13.	Referencias	169
14.	Anexos.....	176

AGRADECIMIENTOS

Este apartado va dedicado a las personas que me han apoyado en todo mi recorrido universitario, desde mi familia, amigos y compañeros de estudio, por su apoyo y ayuda incondicional.

Además, agradecer como no a la Universidad Politécnica de Madrid, y más en concreto los profesionales de la Escuela de Topografía, Geodesia y Cartografía, por su trabajo y guía durante estos años, a parte dentro de ellos a los profesores y profesoras, de los cuales he podido encontrar siempre la puerta abierta dispuestos a compartir conocimiento, y resolver siempre mis dudas.

A parte y como no, mis tutores en este Trabajo Final de Grado, Ionut Cira y Ramon Alcarria, por su guía, consejo y experiencia.

A todas las personas con las que he podido compartir este camino universitario, muchas gracias por influir en mí y convertirme en el profesional que soy.

RESUMEN

En septiembre del año 2021, un catastrófico evento azotó España. Sobre Cumbre Vieja, en la costa suroeste de la isla de La Palma (islas Canarias), surgió una nueva erupción volcánica, que mantendría en vilo a la población de la isla nada más y nada menos que 85 días, en los cuales millones de toneladas de material incandescente se ocuparían de destruir más de 1.200 hectáreas de la superficie de la isla, generando miles de afectados, entre los municipios de El Paso, Los Llanos de Aridane y Tazacorte.

Por esta situación, se ha desarrollado el proyecto “Visor La Palma”, con la intención clara de tomar información veraz y actual de la isla de La Palma, poniendo especialmente el foco en la actividad volcánica de la zona de Cumbre Vieja. Este proyecto se basó, por un lado, en el desarrollo de una aplicación web, según la arquitectura cliente-servidor, apoyada en las tecnologías de Node JS con Vue y Node JS con Express y, por otro lado, en el diseño de un circuito de procesos de extracción, integración y almacenamiento de datos georreferenciados, llevado a cabo gracias a las tecnologías de Pentaho Data Integration y MongoDB. El diseño de este portal web, se centró en la visualización e interpretación de los datos, mediante elementos cartográficos como cambio de mapas base, control de visualización de capas de datos, controles de escala, herramientas medición, entre muchas más que procuran brindar accesibilidad a datos científicos al máximo número de personas posibles, independientemente de su conocimiento previo, ya que otro punto clave del proyecto está en los usuarios, más específicamente en los Palmeros y Palmeras, pues el portal pretende también ser un espacio que sirva de altavoz, donde además de ver reflejada la situación de su hogar, los habitantes de La Palma puedan aportar datos de eventos volcánicos, climáticos y como estos impactan a su vida cotidiana.

Por lo tanto, en este trabajo, se desarrolló un sistema de creación, validación y publicación de incidencias, a la par que un servicio de alertas, que permiten en conjunto acercar todavía más los datos a los usuarios y a la vida en La Palma.

ABSTRACT

In September 2021, a catastrophic event hit Spain. Above Cumbre Vieja, on the Southwest coast of La Palma Island (Canary Islands), a new volcanic eruption occurred, which would keep the island's population in suspense for 85 days. During which time millions of tons of incandescent material threatened to destroy more than 1,200 hectares of the island's surface, affecting thousands between the municipalities of El Paso, Los Llanos de Aridane and Tazacorte.

Due to this situation, the “Visor La Palma” project was developed, with the clear intention of taking accurate and up-to-date information from La Palma Island, focusing especially on the volcanic activity of Cumbre Vieja area. This project was based on the one hand, on the development of a web application, according to the client-server architecture, supported by the technologies of Node JS with Vue and Node JS with Express, on the other hand, on the design of a circuit of extraction, integration and storage processes of georeferenced data, carried out thanks to Pentaho Data Integration and MongoDB technologies.

The design of this web portal focused on the visualization and interpretation of the data, through cartographic elements such as changing base maps, visualization control of data layers, scale controls, measurement tools, among many more that seek to provide accessibility to scientific data to the maximum number of people possible, regardless of their prior knowledge, since another key point of the project is in the users, more specifically in the Palmeros and Palmeras, the portal also aims to be a speaker space , where in addition to reflect the situation of their home, the population of La Palma can provide data about volcanic and climatic events and how these impact their daily lives.

In this work, a system for creating, validation and publishing incidents was developed, as well as an alert service, which together make it possible to bring the data even closer to users and to life on La Palma.

ÍNDICE DE FIGURAS

Ilustración 1. Mapa de huellas de erupciones volcánicas históricas en La Palma.....	16
Ilustración 2. Logo de Pentaho Data Integration.....	22
Ilustración 3. Logo de MongoDB	23
Ilustración 4. Logo de Geoserver	23
Ilustración 5. Logo de Node JS	24
Ilustración 6. Logo de Vue JS	28
Ilustración 7. Logo de Leaflet	29
Ilustración 8. Logo de Bootstrap	30
Ilustración 9. Logo de Docker.....	31
Ilustración 10. Logo de Quantum GIS	32
Ilustración 11. Logo de Visual Studio Code	32
Ilustración 12. Logo de Git	33
Ilustración 13. Logo de Postman.....	33
Ilustración 14. Logo de la dirección general de trafico	34
Ilustración 15. Registro de ejemplo colección mongo, usuarios	37
Ilustración 16. Registro de ejemplo colección mongo, tokens.....	38
Ilustración 17. Registro de ejemplo colección mongo, registros_calidadAire	41
Ilustración 18. registro de ejemplo colección mongo, registros_calidadAire	47
Ilustración 19. Registro de ejemplo colección mongo, imagenes_incidencias_default	48
Ilustración 20. Registro de ejemplo colección mongo, registros_alarmas	50
Ilustración 21. Destalles de los datos de las parcelas	51
Ilustración 22. Destalles de los datos de las edificaciones	52
Ilustración 23. Destalles de los datos de las carreteras.....	52
Ilustración 24. Destalles de los datos del perímetro de colada volcánica	52
Ilustración 25. Destalles de los datos de las bocas eruptiva	53
Ilustración 26. Transformación Pentaho Data Integration (PDI) de extracción de registros de calidad del aire	61

Ilustración 27. Transformación PDI de historial de registros de calidad del aire	63
Ilustración 28. Transformación PDI de borrado de incidencias obsoletas	71
Ilustración 29. Transformación PDI de extracción de incidencias DGT.....	73
Ilustración 30. Transformación PDI de extracción de terremotos.....	77
Ilustración 31. Trabajo PDI de actualización de incidencias y registros de medición	78
Ilustración 32. Trabajo PDI de actualización de terremotos	79
Ilustración 33. Captura mockup del visor web.....	80
Ilustración 34. Captura mockup del panel de control de capas	81
Ilustración 35. Captura mockup interacción de selección de mapa base	82
Ilustración 36. Captura mockup interacción de visualización localizaciones estaciones calidad del aire.....	83
<i>Ilustración 37. Captura mockup interacción de visualización datos de calidad del aire</i>	84
Ilustración 38. Captura mockup interacción de visualización localizaciones de incidencias.....	86
Ilustración 39. Captura mockup interacción de visualización datos de incidencia	87
Ilustración 40. Captura mockup interacción de visualización de capas servidas por Geoserver	88
Ilustración 41. Captura mockup interacción de visualización de terremotos por lapso temporal	89
Ilustración 42. Captura mockup interacción de visualización de mapa de calor de terremotos por lapso temporal	91
Ilustración 43. Captura mockup funcionamiento herramienta de geocodificación	92
Ilustración 44. Captura mockup funcionamiento herramienta de geocodificación inversa.....	93
Ilustración 45. Captura mockup de funcionamiento de la leyenda del mapa	94
Ilustración 46. Captura mockup de herramientas de control de zoom	95
Ilustración 47. Captura mockup cambio de idioma y traducción de contenido al inglés	96
Ilustración 48. Captura mockup del formulario de contacto	97
Ilustración 49. Captura mockup acceso a manual de uso.....	98
Ilustración 50. Captura mockup de formulario de registro de usuario	99
Ilustración 51. Captura mockup del formulario de acceso para usuarios	100

Ilustración 52. Captura mockup de cuadro de perfil de usuario.....	101
Ilustración 53. Captura mockup formulario de actualización de perfil de usuario.....	102
Ilustración 54. Captura mockup funcionamiento herramienta de medición.....	104
Ilustración 55, Captura mockup formulario de creación de nuevas incidencias	105
Ilustración 56. Captura mockup interacción de visualización registros de incidencias creadas por el usuario sin validar	106
Ilustración 57. Captura mockup formulario de creación de alarmas.....	107
Ilustración 58. Captura mockup interacción de visualización registros de alarmas creadas por el usuario.....	108
Ilustración 59. Captura mockup interacción de visualización de datos registros de alarmas creadas por el usuario.....	109
Ilustración 60. Captura mockup funcionamiento reinicio de alarmas activadas	110
Ilustración 61. Captura mockup funcionamiento eliminación de alarmas de usuarios	111
Ilustración 62. Captura mockup de selección de coordenadas por mapa auxiliar.....	112
Ilustración 63. Captura mockup funcionamiento descarga de registros históricos por estación meteorológica.....	113
Ilustración 64. Captura mockup de acceso de salida de sesión	114
Ilustración 65. Captura mockup interacción de visualización registros de incidencias sin validar para usuarios administradores	115
Ilustración 66 Captura mockup formulario de creación de nuevas incidencias validadas	116
Ilustración 67. Captura mockup formulario de edición y validación de incidencias validadas o no	117
Ilustración 68. Captura mockup acceso a eliminación o rechazo de incidencia.....	118
Ilustración 69. Captura mockup de formulario de búsqueda de perfil de usuario	119
Ilustración 70. Captura mockup menú de control de perfil de usuario encontrado.....	119
Ilustración 71. Captura mockup acceso a funcionalidad de cambio de rol	120
Ilustración 72. Captura mockup acceso a funcionalidad de eliminación de perfil de usuario...	121
Ilustración 73. Diagrama de arquitectura del sistema	123
Ilustración 74. Árbol de archivos del servidor del proyecto	125

Ilustración 75. Árbol de archivos del cliente del proyecto	127
Ilustración 76. Árbol de archivos de la carpeta store del cliente.....	128
Ilustración 77. Línea de tiempo desarrollo del portal web	131
Ilustración 78. Versión final de US conexión mongo servidor cliente. Página principal aplicación	132
Ilustración 79. Versión final de US creación de usuarios. Página de acceso aplicación.....	133
Ilustración 80. Versión final de US geocodificación y mejorar de mapa. Página principal aplicación	134
Ilustración 81. Versión final de US creación incidencias registros. Página principal aplicación, usuario registrado	135
Ilustración 82. Versión final de US creación alarmas registros. Página principal aplicación, usuario registrado	136
Ilustración 83. Versión final de US cambio idiomas. Página principal aplicación	136
Ilustración 84. Versión final de US descarga historial registros estaciones. Página principal aplicación, usuario registrado.....	137
Ilustración 85. Versión final de US creación contacto. Página de contacto aplicación	138
Ilustración 86. Versión final de US implementación capas Geoserver. Página principal aplicación	138
Ilustración 87. Versión final de US conexión PDF ayuda usuarios. Página principal aplicación	139
Ilustración 88. Versión final de US leyenda mapa. Página principal aplicación.....	140
Ilustración 89. Versión final de US selección de coordenadas por mapa. Página de creación de incidencias aplicación	140
Ilustración 90. Versión final de US incidencias DGT. Página principal aplicación, usuario administrador registrado.....	141
Ilustración 91. Versión final de US herramienta medir mapa. Página principal aplicación, usuario registrado.....	142
Ilustración 92. Gama de color principal empleada en el estilo del portal	144
Ilustración 93. Captura estilo de carreteras principales.....	149

Ilustración 94. Captura estilo de edificios	150
Ilustración 95. Captura estilo parcelas	151
Ilustración 96. Captura estilo colada volcánica.....	151
Ilustración 97. Captura estilo bocas eruptivas.....	152
Ilustración 98. Estructura página principal aplicación	152
Ilustración 99. Estructura página de acceso aplicación.....	153
Ilustración 100. Captura de componente cabecera.....	154
Ilustración 101. Captura de componente mapa	154
Ilustración 102. Captura de componente panel de control	155
Ilustración 103. Captura de componente formulario de registro.....	156
Ilustración 104. Captura de componente menú de control de super administrador	156
Ilustración 105. Captura de componente cuadro de datos calidad del aire	157
Ilustración 106, Captura de componente submenú de control capas de actividad volcánica....	158
Ilustración 107. Ejemplo de Interactividad componentes página general aplicación.....	158

ÍNDICE DE TABLAS

Tabla 1. Roles de la colección usuarios	37
Tabla 2. Acrónimos de tipos de incidencia.....	45
Tabla 3. Visibilidad de incidencias por usuario.....	46
Tabla 4. Acrónimos de origen de incidencias.....	47
Tabla 5. Tipos de incidencia del modelo de datos de la DGT	68
Tabla 6. Iconos de la aplicación	146
Tabla 7. Imágenes auxiliares de incidencias según su tipo.....	148
Tabla 8. Simbología de registros	149
Tabla 9. Presupuesto de hardware	162
Tabla 10. Presupuesto de software	163
Tabla 11. Presupuesto de producción	164
Tabla 12. Presupuesto bruto total	164
Tabla 13. Presupuesto total.....	165
Tabla 14. Presupuesto de despliegue de aplicación.....	166

ACRÓNIMOS

APP: Application (aplicación)

CO: Monóxido de Carbono

NO₂: Dióxido de Carbono

O₃: Ozono

SO₂: Dióxido de azufre

ETL: Extraction, Transformation, Load (extracción, transformación, carga)

DB: Data Base (base de datos)

JSON: JavaScript Object Notation (notación de objetos javascript)

WMS: Web Map Service (servicio de mapas web)

HTTP: Hypertext Transfer Protocol (protocolo de transferencia de hipertexto)

JS: JavaScript

API: Application Programming Interface (interfaz de programación de aplicaciones)

URL: Uniform Resource Locator (localizador uniforme de recursos)

HTML: HyperText Markup Language (lenguaje de marcado de hipertexto)

CSV: Comma Separated Values (archivo de valores separados por comas)

QGIS: Quantum Geographic Information System

SIG: Sistema de Información Geográfica

IDE: Integrated Development Environment (entorno de desarrollo integrado)

IGN: Instituto Geográfico Nacional

DGT: Dirección General de Tráfico

PC: Personal Computer (equipo personal)

PDI: Pentaho Data Integration

REST: Representational State Transfer (transferencia de estado representacional)

US: User Stories (historia de usuario)

PDF: Portable Document Format (formato de documento portable)

ECTS: European Credit Transfer and Accumulation System

1. INTRODUCCIÓN

En esta sección, se repasarán las bases conceptuales de este proyecto para justificar cada una de las características y funcionalidades que este presenta.

1.1 CONTEXTO Y JUSTIFICACIÓN

La erupción del Cumbre Vieja a finales de 2021 se suma a la larga lista de erupciones que ha sufrido la isla de La Palma a largo de su vida, convirtiéndola en la isla del archipiélago canario con más erupciones registradas. Debido a la frecuente actividad volcánica que existe en el lugar y teniendo en cuenta los daños producidos por el ultimo evento de este tipo se requiere de un monitoreo constante de los indicadores de actividad volcánica. Este proyecto pretende dar respuesta a este interés social y ofrecer una visión precisa y sencilla de los datos de estos indicadores, proporcionando una fácil interpretación a los usuarios [1].

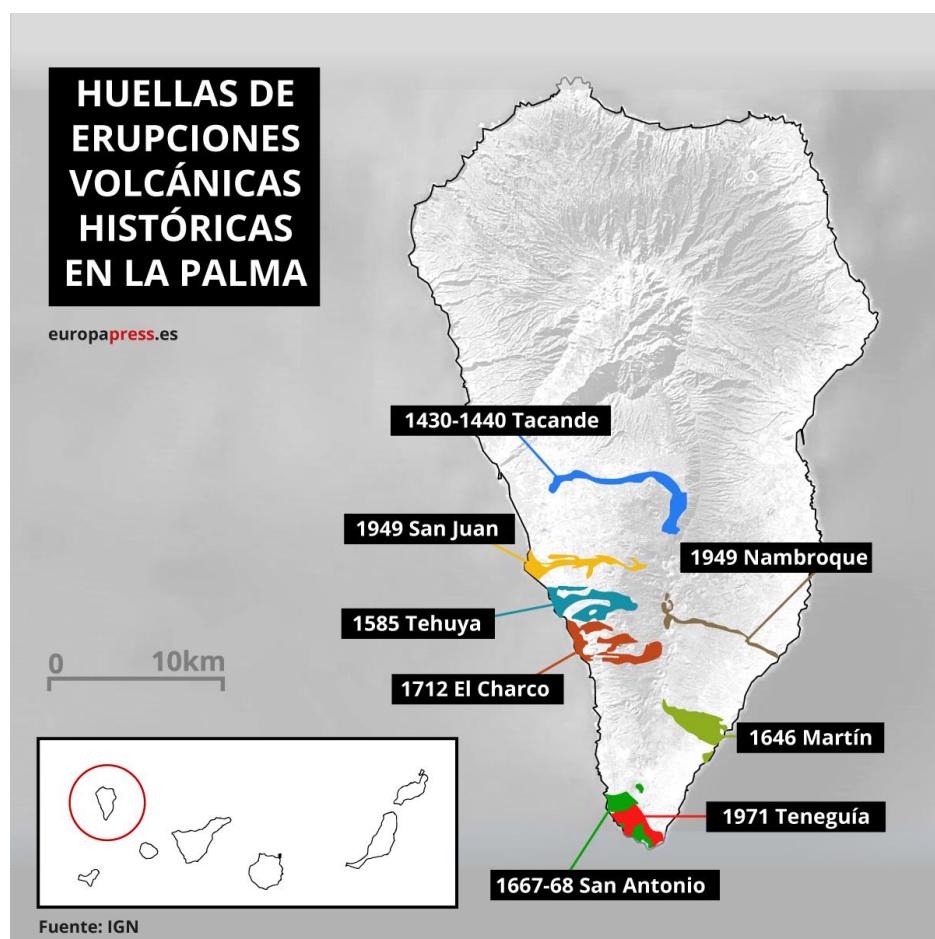


Ilustración 1. Mapa de huellas de erupciones volcánicas históricas en La Palma

Además, aunque el proyecto naciera por las circunstancias de la isla, también pretende dar un paso más, proporcionando una herramienta de comunicación de incidencias para la población de La Palma.

Por otro lado, ya desde antes de la erupción volcánica, el Cabildo Insular de la isla de La Palma empezó a desarrollar proyectos muy interesantes para incorporar nuevas tecnologías que permitan una gestión más eficiente de los recursos, servicios e infraestructura de la isla. Un ejemplo de esto son los proyectos propuestos en el Plan “*La Palma Smart Island*” [2], que se describirán en el Apartado 3 de la memoria. Estas iniciativas son muy diversas, pero bajo mi punto de vista pueden tener un punto de unión en el estudio del medio ambiente y en el de los indicadores de actividad volcánica, brindando al usuario una visión más amplia que permita una mejor valoración global de la situación de la Isla Bonita. Por ello y, en busca de dar esta perspectiva, surgió este Proyecto de Final de Grado.

1.2 MOTIVACIÓN

El objetivo final de este trabajo es el desarrollo de una herramienta versátil y sencilla, que permita el acceso e interpretación de los datos a cualquier tipo de usuario con cualquier tipo de interés o capacidad, siendo además una vía de interacción entre los datos y los usuarios, haciendo a cada uno de ellos posibles colaboradores del proyecto.

1.3 DESCRIPCIÓN GENERAL DEL PROYECTO

El proyecto trata el desarrollo de una aplicación web destinada a ser una herramienta de comunicación, en la cual se centralicen los datos de indicadores de actividad volcánica (registros de calidad del aire, terremotos, etc.), las posibles incidencias producto de estos indicadores e información geográfica de referencia que facilite la interpretación de estos datos (coladas y bocas volcánicas, mapas de calor, carreteras, etc.). Permitiendo, además a los usuarios convertirse en participes de la producción de información, con la opción de registro de nuevas incidencias, aumentando así la capacidad de monitoreo sobre la isla de La Palma.

Cabe destacar, que un punto central del proyecto es dar a conocer la información más coherente con la realidad. Por ello, la aplicación muestra datos continuamente actualizados e incorpora funcionalidades de vigilancia ante aparición de incidencias (servicio de alertas).

1.4 ESTRUCTURA DEL DOCUMENTO

En este documento, se puede encontrar toda la información relativa al proyecto que está organizada y repartida en distintos puntos y secciones. Los primeros apartados se dedican a la presentación del proyecto, contexto, razones por las que surgió la idea, el plan y los objetivos a alcanzar. Esta parte está formada por las dos primeras secciones, Sección 1 (Introducción) y Sección 2 (Objetivos y alcance).

En la Sección 3 “Tecnologías utilizadas en soluciones del estado del arte” y la Sección 4 “Material y Metodología”, se muestran los recursos técnicos en los cuales se sustenta el proyecto, pasando por las tecnologías implementadas, conjuntos de datos y estrategias de trabajo aplicadas.

En cuanto al desarrollo del software producido en este proyecto, se ve expuesto en la Sección 5 “Extracción e integración de datos”, y en la Sección 6 “Requisitos de software”. Aquí se expresan los pasos seguidos para la creación del portal web de La Palma. Por otro lado, cercano a esta parte de la memoria, están las secciones de la 7 a la 9, Sección 7 “Arquitectura del sistema”, Sección 8 “Proceso del desarrollo” y Sección 9 “Interfaz de usuario”. Estas secciones se relacionan a las de la división anterior, ya que se ocupan de mostrar el marco de organización y características seguidas en el desarrollo del software.

Para concluir la memoria del trabajo, se dedicaron la Sección 10 “Resultados y logros”, la Sección 11 “Presupuesto” y la Sección 12 “Conclusiones”. En estos apartados, se realiza un análisis final de la aplicación desarrollada valorando su cumplimiento de objetivos y puntos a seguir desarrollando dentro de esta, así como también una estimación del coste que tendría este proyecto si se lleva a cabo en el ámbito profesional.

Por último, se encuentra la Sección 13 “Referencias” y Sección 14 “Anexos” que sitúan la bibliografía por la cual se sustentó el proyecto y los archivos que junto a este documento ayudan a la interpretación del contenido de mi Trabajo Final de Grado (TFG).

2. OBJETIVOS Y ALCANCE

El objetivo principal de este proyecto es conseguir una eficiente visualización de la gran diversidad de información proporcionada por el Cabildo de La Palma, enfocando toda esta, en el estudio de la actividad volcánica y sísmica ocurrida en la superficie de la isla.

2.1 OBJETIVOS

Los objetivos específicos de este Trabajo Final de Grado son los siguientes:

- ✓ **Creación de un sistema de usuarios:** incorporar un sistema de usuarios compuesto por usuarios de roles distintos, rol de usuario general, administrador y superadministrador. Permitiendo así la autogestión de la aplicación web.
- ✓ **Presentar datos actualizados:** un valor significativo que adquiere este proyecto es la visualización de únicamente datos actualizados buscando así mostrar la visión más adaptada a la realidad de la isla. Este punto es muy importante ya que es diferencial ante proyectos mencionados anteriormente.
- ✓ **Mostrar una variedad adecuada de información:** este objetivo pretende integrar las capas de información más diversas posibles, que puedan ser de interés y de valor a los usuarios del portal web, a la hora de estudiar la actividad volcánica.
- ✓ **Interacción del usuario con los datos:** permitir una alta interacción de los usuarios, mediante la creación de incidencias, y el servicio de alertas, dando un paso más en la gestión de la información.
- ✓ **Buscar una alta accesibilidad:** trabajar para que la fácil interpretación y manejo del portal web, sea posible para el máximo número de usuarios, mediante la creación de manuales de usuario y la opción de una traducción total al inglés.
- ✓ **Interfaz dinámica y sencilla:** diseñar una interfaz atractiva a la vez que práctica e intuitiva, que permita contestar a los intereses de cada tipo de usuario.

2.2 ALCANCE

El alcance de este proyecto es el desarrollo de una aplicación (app), que englobe la visualización de datos ofrecidos por los diversos proyectos mencionados en la Sección 1.2, centrada en el estudio de los indicadores de actividad volcánica. De esta forma, se permite a los usuarios observar la situación actual de la isla desde varios puntos de vista.

2.3 PARTES INTERESADAS

En este punto se analizará las organizaciones o individuos que pueden estar interesadas en este proyecto:

- **Cabildo de La Palma:** la mayoría de los datos de este proyecto son proporcionados por organismos derivados o financiados por el cabildo de La Palma, por ello es notable su interés en el desarrollo de tecnologías que gestionen y muestren su información geográfica. Esto convierte a esta entidad en un potencial destinatario del proyecto, así como potencial colaborador a la hora del mantenimiento de la información y de las funcionalidades de la app [3].
- **Propietarios:** desde el punto de vista social esta web también permite observar la situación de parcelas, viviendas, infraestructura, etcétera. Aunque no es su principal objetivo puede ser un primer acceso, para conocer la realidad de cada propiedad que se sitúa en la isla.
- **Empresas relacionadas:** empresas con interés en conocer o trabajar con los datos o las tecnologías presentadas en este proyecto.
- **Ciudadanos de la isla:** el receptor principal del proyecto son los Palmeros y Palmeras residentes en la isla, ya que son los que experimentan todos los cambios naturales que suceden en La Palma, y que pretende reflejar este proyecto. Además, este portal web aspira también ser un altavoz donde puedan expresar sus observaciones y experiencias.
- **Investigadores externos:** los datos que se muestran, así como las propiedades de interacción que ofrece el visor, presentan una oportunidad nueva de estudio a investigadores, aficionados, o profesionales, de los sucesos volcánicos y naturales tan impresionantes de la isla de La Palma.
- **Organizaciones de salud y medio ambiente:** un punto importante de la app es el dar a conocer el impacto que los procesos naturales pueden tener en la vida de la isla. Por ello, este proyecto se presenta como una vía de comunicación y de estudio de estos impactos, para organizaciones de este tipo.
- **Delincuentes cibernéticos:** por último, siempre se debe tener en cuenta en proyectos de este tipo la posibilidad de sufrir ataques cibernéticos, por ello se ha establecido entre otras medidas sistemas de encriptación de la información, tokens de acceso, etc.

2.4 PROYECTOS RELACIONADOS

Algunos proyectos relacionados son los pertenecientes al plan “*La Palma Smart Island*” [2], programa de cual me gustaría destacar los siguientes trabajos:

- **Open Data La Palma:** portal de datos abiertos de la isla de La Palma, en el que se da acceso a los datos públicos del cabildo, con el objetivo de generar valor y riqueza mediante los productos derivadas de dichos datos, así como generar transparencia e interoperabilidad entre administraciones [4].
- **La Palma Visualizer:** visor web de datos de calidad de aire, obtenidos mediante el monitoreo de gases contaminantes como el Dióxido de Azufre (SO₂), Monóxido de Carbono (CO), Materia Particulada, etcétera [5].
- **El tiempo en La Palma en vivo:** iniciativa del Cabildo de La Palma en colaboración con HDmeteo.com que desarrolla mapas dinámicos basados en datos recogidos de todas las estaciones meteorológicas de la isla de La Palma [6].

Producto de la erupción de 2021 varios de estos proyectos se vieron en necesidad de ampliar su alcance, se aumentó el número de estaciones meteorológicas, creación de nuevos datos relacionados con la actividad volcánica, ... esto finalmente impulso la creación de nuevos proyectos como los siguientes:

- **Riesgo Volcánico La Palma:** desarrollo de un portal web destinado a proporcionar todo tipo de información relativa a la erupción de Cumbre Vieja y al estudio de la actividad volcánica de la isla de La Palma [7].
- **Plan Valle y Asistencia a los Afectados:** planes producidos como respuesta a la emergencia social, medioambiental y agrícola, provocada por los destrozos que dejo la erupción de Cumbre Vieja [8].

3. TECNOLOGÍAS UTILIZADAS EN SOLUCIONES DEL ESTADO DEL ARTE

Las tecnologías empleadas son las más adecuadas para cada uno de sus desempeños correspondientes a este proyecto, ya que fueron elegidas tras poner en valoración todo lo aprendido en el grado, consejo de distintos profesionales de la materia, así como mi experiencia previa y comodidad con cada una de ellas. Por otro lado, cabe destacar, que todas son de código abierto, ya que son de acceso gratuito, a la par que ofrecen una destacada interoperabilidad y personalización, sostenidas por una gran comunidad tecnológica que aporta continuamente en su desarrollo y mejora.

Por ello, en las siguientes secciones, se presentará y explicará las características de cada una de ellas y la razones por las que han sido escogidas para este proyecto.

3.1 TECNOLOGIAS DE GESTIÓN DE DATOS

Las siguientes tecnologías descritas son las correspondientes a la extracción, transformación, almacenamiento y servicio de datos de la aplicación.

3.1.1 Pentaho Data Integration



Ilustración 2. Logo de Pentaho Data Integration

Pentaho Data Integration [9] es una herramienta ETL (Extraction, Transformation, Load), open source, que simplifica la tarea de combinar y transformar datos de diferentes fuentes en un formato que sea fácil de entender y analizar. Funciona como un conjunto de bloques de construcción que te permiten diseñar flujos de trabajo para extraer datos de fuentes como bases de datos, archivos, etc. Además, permite la transformación de estos según las necesidades del proyecto, para finalmente cargarlos, con el formato correcto, en el destino deseado. En resumen, Pentaho Data Integration hace que la manipulación de datos sea accesible y eficiente, lo que facilita enormemente la adquisición e implementación de datos.

En concreto en el proyecto se ha empleado la versión estable en el momento de inicio del desarrollo, PDI 9.4 del 8 de noviembre de 2022.

3.1.2 MongoDB



Ilustración 3. Logo de MongoDB

MongoDB [10] es una base de datos NoSQL, que almacena datos en un formato flexible, utilizando BSON (Binary JSON) para representar la información. Esto permite guardar datos de diferentes estructuras en la misma base de datos sin requerir un esquema fijo. MongoDB es especialmente adecuado para desarrollos particularmente escalables, como aplicaciones web y móviles, a parte, se caracteriza por su facilidad de uso y de gestión de grandes cantidades de datos. En cuanto a la versión utilizada en el proyecto fue la 6.0.6.

3.1.3 Geoserver



Ilustración 4. Logo de Geoserver

GeoServer [11] es una tecnología que permite compartir y publicar datos geoespaciales en línea de manera accesible y eficiente. Funciona como un servidor web especializado en datos espaciales, lo que significa que puedes cargar mapas, imágenes y datos geográficos y luego ofrecerlos a través de servicios web estándar, como Web Map Service (WMS), Web Feature Service (WFS), etc. Esto facilita la visualización y el análisis de información geográfica en aplicaciones web, permitiendo a los desarrolladores crear mapas interactivos y aplicaciones de geolocalización de forma muy sencilla. En cuanto a este proyecto, la tecnología que se aprovecha para comunicar la aplicación web con Geoserver es Web Map Service, ya que permite, de forma muy sencilla, obtener información geoespacial a través de solicitudes HTTP (Hypertext Transfer

Protocol) y recibirlas, directamente, en formato de imagen. Por último, cabe destacar que la versión utilizada es la 2.23.1 de mayo de 2023 [12].

3.2 TECNOLOGIAS IMPLEMENTADAS EN PROYECTOS NODE JS



Ilustración 5. Logo de Node JS

Los proyectos de cliente, Front-End, y de servidor, Back-End, están ambos constituidos por proyectos Node.js [13], en su versión estable 18.16.0 [14]. Esta tecnología de desarrollo permite ejecutar código JavaScript en el servidor, en lugar de solo en el navegador. Esta característica genera una gran facilidad para la creación de aplicaciones web y servidores altamente eficientes y escalables. Cabe destacar la capacidad de los proyectos Node.js de manejar múltiples solicitudes de manera simultánea, lo que lo hace adecuado para aplicaciones en tiempo real como la descrita en este proyecto. Su enfoque en eventos y su biblioteca de módulos extensa facilitan el desarrollo rápido de aplicaciones. Además, al ser de código abierto, una gran comunidad de desarrolladores contribuye a su mejora constante, ofreciendo una amplia gama de recursos y funcionalidades extras, que como se verá a continuación, se han aprovechado en el desarrollo de este trabajo.

3.2.1 Back-End

El Back-End, o lado servidor de este proyecto, es la parte de la aplicación web que se encarga de la gestión y procesado de datos, así como funcionalidades que no son visibles para el usuario final, como envío de correos electrónicos, evaluación de datos, etc. El servidor es responsable de la lógica y la funcionalidad que ocurre detrás del Front -End, permitiendo que la aplicación funcione de manera efectiva, a través, de la gestión de bases de datos, la autenticación de usuarios, la lógica empresarial, la seguridad, la administración de servidores y otras operaciones que son cruciales para el funcionamiento de la aplicación. Por ello y para conseguir un buen desempeño en cada una de las funcionalidades que pretende esta aplicación web, la instalación de módulos externos, gracias a las condiciones del entorno Node.js, es una metodología que se ha repetido constantemente en este proyecto, ya que permite un desarrollo rápido y funcional. A continuación, se presentarán las tecnologías añadidas en el proyecto Node.js del lado servidor.

3.2.1.1 Express.js

Express.js [15] es una tecnología para servidores JavaScript que simplifica la creación de aplicaciones web, ya que actúa como un marco de trabajo, framework, ligero y flexible que facilita la construcción de rutas, manejo de solicitudes y respuestas, gestión de middleware, etcétera. Destacando, por tanto, en su simplicidad y su capacidad de expansión, lo cual significa aumentar la facilidad del desarrollo de nuevas funcionalidades y su organización. La versión empleada en este proyecto es la 4.18.2.

3.2.1.2 Express Fileupload

Express Fileupload [16] es una extensión del framework Express.js, que simplifica la manipulación y el manejo de archivos en las aplicaciones web. Permitiendo la fácil recepción de archivos provenientes de formularios, capacidad de procesamiento y almacenamiento en el servidor. Esta tecnología se utiliza comúnmente en aplicaciones que requieren la subida de imágenes, documentos u otros tipos de archivos. La versión empleada en este proyecto es la 1.4.0.

3.2.1.3 Cors

Cors [17], Cross-Origin Resource Sharing (Compartir Recursos entre Orígenes Cruzados), es una tecnología que se utiliza en aplicaciones web para gestionar las solicitudes. Ya que cuando una aplicación web se ejecuta en un dominio y necesita acceder a recursos en otros, se puede enfrentar a restricciones de seguridad que bloquean estas solicitudes. El módulo cors en Node.js ayuda a resolver este problema permitiendo el control de permisos asignados a cada dominio, en el caso de este proyecto, habilita la recepción de llamadas desde el equipo local. La versión utilizada en el proyecto es 2.8.5.

3.2.1.4 Dotenv

Para comprender la tecnología Dotenv [18], es necesario citar al archivo “.env”, este es un fichero de texto utilizado para almacenar variables de entorno en aplicaciones web. Suele contener configuraciones sensibles, como claves de API (Application Programming Interface) o contraseñas, y se utiliza para separar esta información del código fuente de la aplicación, mejorando la seguridad y la portabilidad. Sin embargo, este aislamiento del código dificulta su accesibilidad y lectura, por ello entra el módulo Dotenv, que permite la carga y la gestión de estas variables de entorno, de manera segura y sin tener que codificarlos directamente en el código. Por último, la versión empleada para la aplicación es 16.0.3.

3.2.1.5 Morgan

Morgan [19] es una dependencia de registro de solicitudes HTTP en Node.js que brinda información útil sobre las solicitudes que llegan a un servidor. Es especialmente útil para el

desarrollo y la depuración, ya que registra detalles como las URL (Uniform Resource Locator), métodos HTTP y códigos de estado de las solicitudes entrantes. Esto facilita la comprensión del flujo de datos, crucial en el desarrollo de end-points, puntos de acceso de datos ofrecidos por el servidor. La versión empleada es la 1.10.0.

3.2.1.6 Bcrypt

Bcrypt [20] es una tecnología de seguridad utilizada para el almacenamiento seguro de contraseñas. Emplea un algoritmo de hash que es especialmente lento y resistente a ataques de fuerza bruta, lo que lo hace ideal para proteger las contraseñas almacenadas en bases de datos. En cuanto a la versión utilizada en el proyecto es la 5.1.0.

3.2.1.7 Mongoose

Como se ha mencionado anteriormente, el proyecto consta de la tecnología de MongoDB, por ello Mongoose [21], simplifica la interacción con estas bases de datos, proporcionando una capa de abstracción para trabajar con MongoDB de manera más fácil y estructurada en aplicaciones Node.js. Mongoose, tiene una metodología de definición de modelos y esquemas para los datos, para facilitar la validación y gestión de estos, además de ofrecer funciones comunes en base de datos como guardar, buscar y actualizar registros. La versión Mongoose implementada en el servidor es la 7.2.1.

3.2.1.8 JSON Web Token (JWT)

JSON Web Token [22] es una tecnología que se utiliza para transmitir información en formato JSON (JavaScript Object Notation) de manera segura entre cliente y servidor. Este objeto JSON este compuesto por tres partes: el encabezado la carga útil y la firma. El encabezado contiene información sobre cómo se firma el token, la carga útil almacena los datos que queremos transmitir y la firma verifica que el token no ha sido modificado en el camino y que, por lo tanto, proviene de la fuente esperada. JWT es ampliamente utilizado para la autenticación y la autorización en sistemas de registro de usuarios, ya que permite verificar la identidad de un usuario de manera segura sin necesidad de mantener estados en el servidor, lo que lo hace eficiente y escalable. La versión empleada en esta aplicación es la 9.0.0.

3.2.1.9 Nodemailer

Nodemailer [23] es una tecnología en Node.js que facilita el envío de correos electrónicos desde una aplicación. Permite la configuración de servidores de correo saliente, agregar destinatarios, adjuntar archivos y personalizar el contenido del correo electrónico, como texto y HTML (HyperText Markup Language). La versión utilizada es la 6.9.3.

3.2.1.10 Csv-Writer

Csv-writer [24] es una tecnología en JavaScript que simplifica la creación y escritura de archivos CSV (Comma-Separated Values). Esta herramienta ayuda a la generación de archivos CSV desde datos estructurados, como tablas o listas, para luego guardarlos en el sistema local de la aplicación. La versión empleada en este proyecto es 1.6.0.

3.2.1.11 Moment

Moment.js [25] es un módulo JavaScript que facilita la manipulación y formateo de fechas y horas. Permitiendo realizar operaciones como agregar, restar, comparar, crear, ... fechas de manera sencilla y eficiente. También se usa usualmente para la creación de registros, dotándolos así de un componente temporal. En este caso, la versión utilizada es la 2.29.4.

3.2.1.12 Nodemon

Nodemon [26] es una dependencia de desarrollo, desarrollada para Node.js, que monitorea los archivos del proyecto en busca de cambios para reiniciar automáticamente el servidor cada vez que se detecta una modificación. Esto significa que se pueden realizar cambios en el código fuente de la aplicación sin tener que detener y reiniciar manualmente el servidor cada vez. Nodemon mejora la eficiencia del flujo de trabajo de desarrollo y es ampliamente utilizado en proyectos de Node.js para facilitar las pruebas y la iteración rápida. La versión de esta tecnología en este proyecto es la 2.0.22.

3.2.2 Front-End

El Front-End o lado cliente, es la parte visible y la interfaz de usuario de un sitio web. Es la capa con la que los usuarios interactúan directamente en sus navegadores o dispositivos. El cliente incluye elementos en su desarrollo tales como el estilo, los botones, los formularios, etc., es decir, el contenido que se muestra en la pantalla. Está construido con tecnologías como HTML (HyperText Markup Language), CSS (Cascading Style Sheets) y JavaScript, y se encarga de presentar la información de manera atractiva y funcional, permitiendo a los usuarios navegar, interactuar y realizar acciones en la aplicación.

Por otro lado, el proyecto constó en este lado cliente de algunas tecnologías más, como se menciona en estos siguientes apartados.

3.2.2.1 Vue.js



Ilustración 6. Logo de Vue JS

Vue.js [27] es una tecnología de desarrollo web que simplifica la creación de aplicaciones web. Se enfoca en la capa de Front-End, permitiendo a los desarrolladores construir interfaces de usuario atractivas y dinámicas con facilidad. Además, destaca su uso de componentes reutilizables y ofrece una biblioteca de funciones que facilitan la gestión de la interfaz de usuario y la interacción con datos en tiempo real. En resumen, Vue.js es una herramienta poderosa y accesible para crear interfaces de usuario atractivas e interactivas. La versión de Vue.js con la que se ha desarrollado la aplicación es la 3.2.13.

3.2.2.2 Vue-Router

Vue-Router [28] es una extensión de Vue.js que facilita la creación de aplicaciones web de una sola página (SPA) al permitir la navegación entre diferentes vistas o páginas en una aplicación sin tener que recargar la página completa. Proporciona un enrutador que mapea URLs a componentes Vue, es decir, que permite definir rutas para diferentes secciones de la aplicación, cargando componentes vue de manera dinámica. Esto crea una experiencia de usuario fluida y rápida en proyectos web, donde solo se actualiza las vistas, sin recargar toda la página. La versión utilizada para esta tecnología es la 4.0.3.

3.2.2.3 Vuex

Vuex [29] es una tecnología del entorno Vue.js que facilita la gestión de datos en una aplicación. Permite almacenar y organizar datos compartidos entre componentes de manera centralizada. Esto significa que, en lugar de pasar datos entre componentes de forma compleja, se almacenan en un solo lugar llamado store (almacén). Vuex es ideal para simplificar la gestión de datos y garantizar el mantenimiento y escalabilidad de un proyecto web. La versión utilizada es la 4.0.0.

3.2.2.4 Vue-i18n

Vue-i18n [30] es una tecnología que facilita la internacionalización (i18n) de aplicaciones del entorno Vue.js, es decir, permite adaptar una aplicación para que disponga de varios idiomas. Esto incluye etiquetas, mensajes, fechas, etcétera. Este componente permite la traducción de

contenidos aumentando la accesibilidad y versatilidad hacia audiencias globales. La versión empleada es la 9.1.10.

3.2.2.5 Typescript

El código JavaScript descrito en el Front-End de la aplicación está totalmente desarrollado en formato TypeScript [31], ya que esta extensión de javascript ofrece la agregación de tipos estáticos, es decir, permite la especificación de los tipos de datos de variables y funciones, produciendo una mayor claridad del código y disminuyendo el surgimiento de errores. TypeScript es especialmente útil en proyectos grandes y complejos, ya que ayuda a detectar problemas de manera temprana y proporciona herramientas de desarrollo sólidas, como autocompletado y verificación de tipos en tiempo real. No obstante, TypeScript se compila a JavaScript estándar, lo que permite que se ejecute en cualquier navegador o entorno de JavaScript, permitiendo la misma accesibilidad y publicación que cualquier proyecto de este tipo. Por último, esta tecnología esta implementada en el cliente con la versión 4.5.5.

3.2.2.6 Babel

Babel [32] es una tecnología que se utiliza para transformar código JavaScript escrito en una versión moderna o futura del lenguaje, a una versión anterior compatible con navegadores y entornos de ejecución más antiguos. Permite la escritura de código en JavaScript utilizando características avanzadas, como sintaxis de ECMAScript 6 (ES6) o posteriores, y luego lo transforma en código ES5 o ES3, que es comprensible para navegadores más antiguos. Por ello Babel es esencial para garantizar la compatibilidad entre navegadores y permitir el uso de las últimas características de JavaScript en aplicaciones web. La versión utilizada es la 5.0.0.

3.2.2.7 Leaflet



Ilustración 7. Logo de Leaflet

Leaflet [33] es una tecnología que simplifica la creación de mapas interactivos en aplicaciones web. Proporciona una biblioteca de JavaScript fácil de usar, así como varias opciones, como la creación de marcadores, dibujo de polígonos, pop-ups, etc. Además, es altamente personalizable, ya que admite una amplia variedad de complementos y fuentes de mapas, lo que lo hace un módulo versátil y adecuado para proyectos de cualquier tamaño. Cabe destacar que, por la

compatibilidad de esta librería con el resto del proyecto, la versión de Leaflet utilizada es la 1.9.3 en TypeScript [34].

3.2.2.8 Leaflet heat

Leaflet heat [35] es una extensión de la biblioteca Leaflet, que permite crear mapas de calor (heatmap) de manera fácil y efectiva en aplicaciones web. Esta tecnología toma puntos de ubicación y los representa visualmente como una superposición de colores que indican la densidad o intensidad de un parámetro de estos puntos en el mapa. Es útil para visualizar patrones de concentración de datos, como la distribución de eventos, mejorando la capacidad de análisis de datos geoespaciales. De igual manera que Leaflet la versión de esta tecnología en TypeScript es la 0.2.2.

3.2.2.9 Leaflet control geocoder

Leaflet Control Geocoder [36] es una extensión de Leaflet que facilita la incorporación de geocodificación en mapas interactivos. Permite a los usuarios buscar ubicaciones ingresando direcciones o nombres de lugares y ver los resultados en el mapa. Esta tecnología utiliza servicios de geocodificación externos, como nominatim (servicio open source de geocodificación desarrollado por OpenStreetMaps), para convertir las consultas de búsqueda en coordenadas geográficas. Para esta tecnología no fue necesario la interpretación en TypeScript, por ello la versión implementada fue la 2.4.0.

3.2.2.10 Bootstrap



Ilustración 8. Logo de Bootstrap

Bootstrap [37] es una tecnología que simplifica el diseño y la creación de sitios web y aplicaciones móviles. Es un marco de trabajo de código abierto que proporciona una colección de componentes y estilos predefinidos, como botones, formularios, navegación y más. Se suele utilizar Bootstrap para crear interfaces de usuario atractivas con facilidad, ahorrando tiempo y esfuerzo en el desarrollo de la aplicación web. La versión implementada de esta librería es la 5.3.0.

3.2.2.11 Axios

Axios [38] es una dependencia de node.js que simplifica el uso de solicitudes HTTP. Se trata de una biblioteca JavaScript que permite realizar peticiones a servidores web para obtener datos, enviar información o interactuar con servicios en línea de manera eficiente y sencilla. Por último, la versión de axios utilizada es la 1.4.0.

3.3 TECNOLOGIAS DE ENTORNO

Por otro lado, las tecnologías que han servido como medios para poder trabajar con las anteriormente mencionadas son las siguientes:

3.3.1 Docker

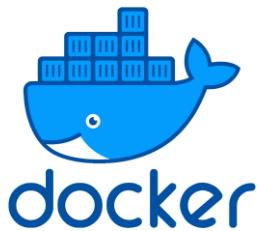


Ilustración 9. Logo de Docker

Docker [39] es una tecnología que simplifica la creación, distribución y ejecución de aplicaciones en contenedores virtuales. Los contenedores son entornos aislados que incluyen todo lo necesario para que una aplicación se ejecute, como el código, las bibliotecas, dependencias, etc. Docker permite a los desarrolladores empaquetar sus aplicaciones en contenedores, lo que garantiza que funcionen de manera consistente en cualquier entorno, esto simplifica el proceso de implementación y escalabilidad de aplicaciones. La versión de Docker aplicada a este trabajo es la 24.0.2.

En el caso de este proyecto esta tecnología permitió la sencilla implementación de un contenedor de MongoDB y otro de Geoserver, servido por Tomcat (Servidor web, que se suele usar para la ejecución de aplicaciones java) [40].

3.3.2 Quantum Geographic Information System (QGIS)



Ilustración 10. Logo de Quantum GIS

QGIS [41] es una tecnología de sistema de información geográfica (SIG) de código abierto que permite crear, visualizar y analizar datos geoespaciales, como mapas y datos de ubicación. Es una herramienta versátil que brinda a los usuarios la capacidad de trabajar con datos geográficos en diferentes formatos y proyecciones. QGIS se destaca por su facilidad de uso y su amplia comunidad de usuarios y desarrolladores que contribuyen a su mejora constante. En resumen, QGIS es una tecnología que simplifica la creación y el análisis de datos geoespaciales de manera accesible y eficaz. Además, dentro de esta tecnología se ha aprovechado especialmente el plugin de **GeoCat Bridge** [42], ya que permite la publicación de datos geoespaciales en servicios de catálogo en línea, como Geoserver. La versión de QGIS que ha sido empleada es la 3.32.2 Lima.

3.3.3 Visual Studio Code (VS Code)



Ilustración 11. Logo de Visual Studio Code

VS Code [43] es un entorno de desarrollo integrado (IDE) de código abierto y gratuito desarrollado por Microsoft. Está diseñado para facilitar la escritura de código en una variedad de lenguajes de programación. A parte de esto, presenta una amplia gama de extensiones que permiten personalizar y ampliar sus capacidades, lo que lo convierte en una herramienta versátil y poderosa para la programación y el desarrollo de software.

3.3.4 GIT



Ilustración 12. Logo de Git

Git [44] es una tecnología que se utiliza para el control de versiones de código fuente en proyectos de desarrollo de software. Funciona registrando los cambios en los archivos a lo largo del tiempo y permite a los desarrolladores colaborar de manera eficiente. Con Git, se puede realizar un seguimiento de las modificaciones, fusionar cambios de múltiples colaboradores y revertir a versiones anteriores del código en caso de problemas. Además, es muy útil a la hora de preservar código, ya que gracias a Git Hub se puede guardar en repositorios en la nube, para así mantener una copia de seguridad externa a cualquier equipo físico. La versión utilizada es la 2.41.0.

3.3.5 Postman



Ilustración 13. Logo de Postman

Postman [45] es una tecnología que simplifica el proceso de probar y enviar solicitudes HTTP a una API. Además, ofrece una interfaz gráfica intuitiva que permite crear, organizar y guardar solicitudes de manera eficiente. Es ampliamente utilizado en el desarrollo de aplicaciones web y servicios para garantizar la calidad y la funcionalidad de las APIs.

4. MATERIAL Y METODOLOGÍA

En este apartado, se repasarán las materias de partida con las cuales se constituyó el proyecto, tanto a nivel de datos como de hardware o software. Además, se explicará la planificación con la cual se desarrolló el trabajo.

4.1 DATOS

Un elemento fundamental en el proyecto es la gestión de los datos, ya que estos son el objeto de comunicación a los usuarios. Por ello se hablará más de esto en los siguientes apartados.

4.1.1 Fuentes de datos

Los datos empleados en este proyecto vienen de cinco vías principales, el Instituto Geográfico Nacional (IGN), la Dirección General de Tráfico (DGT), la Sede Electrónica del Catastro, y los portales Open Data La Palma o La Palma Visualizer mencionados anteriormente en proyectos relacionados.

- **Instituto Geográfico Nacional (IGN):** es el organismo encargado a la gestión de datos e información geográfica en este Caso del Reino de España, permitiendo a los usuarios el acceso a datos geoespaciales como mapas, ortofotos o información sísmica [46].
- **Dirección General de Tráfico (DGT):** es la encargada del desarrollo de mejoras de la estructura general de tráfico, empezando desde la oferta de servicios administrativos, la divulgación de la legislación y comportamientos en pro de la seguridad de la ciudadanía [47]. Además, la incorporación de tecnologías de la información, específicamente la iniciativa del Mapa de estado del tráfico e incidencias [48] permite conocer al proyecto el estado de las vías de tráfico en tiempo real.



Ilustración 14. Logo de la dirección general de trafico

- **Sede Electrónica del Catastro:** el Catastro es el censo estadístico de los bienes inmuebles del estado español, con la descripción física, económica y jurídica de las

propiedades rústicas y urbanas. Además, este organismo presenta servicios de descarga publica que permiten llegar a los ciudadanos a su información geográfica [49].

- El resto de las fuentes de datos tanto **Open Data La Palma** como **La Palma Visualizer** fueron explicadas anteriormente en el punto 2.4 Proyectos relacionados.

4.1.2 Base de dato MongoDB

El almacenamiento principal de los datos del proyecto se da a través de la base de datos Mongo, llamada *TFG_Database*, donde se encuentran las colecciones con datos de registros sistema de usuarios, incidencias, etc. que se encuentran en constante interacción con la aplicación web.

A continuación, se repasarán las colecciones y modelo de datos que se encuentran en la base de datos mongo.

4.1.2.1 Colección usuarios

En esta colección se guarda la información de registro de los usuarios de la aplicación web, por lo que aquí se hallan todos los datos relativos a usuarios que hay en el proyecto. Esta información guarda mucha relevancia, ya que configurará y permitirá las funcionalidades de la aplicación.

Cabe destacar el modelo de datos:

```
Nombre: {
    type: String,
    required: true,
    index: true,
    max: 100
},

Apellido: {
    type: String,
    required: true,
    max: 100
},
Municipio: {
    type: String,
    enum: [
        'Garafía',
        'Barlovento',
        'San Andrés y Sauces',
        'Puntallana',
        'Santa Cruz de La Palma',
        'Breña Alta',
        'Breña Baja',
    ]
},
```

```
        'Villa de Mazo',
        'Fuencaliente de La Palma',
        'Los Llanos de Aridane',
        'El Paso',
        'Tazacorte',
        'Tijarafe',
        'Puntagorda',
    ],
},
Usuario: {
    type: String,
    required: true,
    unique: true,
    index: true,
    max: 100
},
Mail: {
    type: String,
    unique: true,
    index: true,
    required: true,
},
Contraseña: {
    type: String,
    required: true
},
Rol: {
    type: Number,
    default: 0,
    enum: [
        0,
        1,
        5
    ],
    required: true
}
```

En cuanto a los elementos de la estructura de datos se pueden definir como en los siguientes párrafos.

Nombre = campo tipo texto. Guarda el nombre del usuario, es de carácter obligatorio. Además de aportar identificación, permite a la aplicación relacionarse con el usuario. Tiene asignado un índice de búsqueda, para facilitar peticiones por este campo y un máximo de 100 caracteres.

Apellido = campo tipo texto. Guarda el apellido del usuario, es de carácter obligatorio, además de aportar identificación permite a la aplicación relacionarse con el usuario. Asigna un índice de búsqueda (tiene un máximo de 100 caracteres).

Municipio = campo tipo texto. Guarda el municipio que se asigna al usuario, no es obligatorio y restringe el valor al nombre oficial del municipio. Campo interesante, valora la concentración de usuarios y la implicación de la población.

Usuario = campo tipo texto. Identifica de forma inequívoca al usuario. Campo de referencia para el procesamiento de peticiones, atribución de registros, visualización interna, etc. por esto se le añade también un índice de búsqueda que facilite el intercambio de datos.

Mail = campo tipo texto. Guarda el email de referencia para comunicaciones, es obligatorio y ha de ser único. Se le asigna un índice de búsqueda.

Contraseña = campo tipo de texto sin restricción. De carácter obligatorio. Permite el acceso a la cuenta.

Rol = campo tipo numérico y predeterminado a cero que determina las capacidades del usuario en la app. En la tabla 1 se muestra una tabla con los valores posibles de este campo y el rol asociado.

Valor	Rol relacionado
0	Usuario
1	Administrador
5	Super usuario

Tabla 1. Roles de la colección usuarios

A continuación, se muestra un ejemplo de los datos de registro de un usuario.

```
_id: ObjectId('65798270b7dd1f1cacc53e67')
Nombre: "user1"
Apellido: "user1"
Usuario: "user1"
Mail: "user1@localhost.local"
Contraseña: "$2b$10$orLhBNznwhwnJt4R.oLXAeaR1obKBNsGwBNb9cVWbZwbXMSz18t7a"
Rol: 0
__v: 0
```

Ilustración 15. Registro de ejemplo colección mongo, usuarios

Como se puede ver todos los datos se guardan sin ningún tipo de encriptación, a excepción del campo contraseña, para mejorar la seguridad

Aparecen los campos nuevos “_id”, y “-v”. Estos elementos son añadidos automáticamente por mongo. El primero, esta para garantizar el estándar de identificación y registro de objetos en una

colección, y el segundo indica la versión de modificación que presenta el registro, en este caso al no haber sufrido ninguna actualización su valor es 0.

4.1.2.2 Colección tokens

Los tokens son las llaves de registro que como se explicara más adelante, se crean a los usuarios en el momento que acceden a la aplicación web, son únicos y solo se asocia uno por usuario activo en la app. Por ello para mantener el control de estos desde el lado servidor del proyecto, se guarda una copia de cada token activo también en la base de datos. Su estructura de datos es la siguiente.

```
Usuario: {
    type: String,
    required: true,
    unique: true,
    index: true,
    max: 100
},
Token: {
    type: String,
    required: true,
}
```

Como se puede ver el modelo cuenta con tan solo dos campos, que se definirán brevemente a en la siguiente página.

Usuario = texto que indica el nombre de usuario por el que se generó el token de acceso, sigue las características mostradas en el apartado anterior en el elemento Usuario de la colección usuarios, y es el nexo por el que se relacionara el token de acceso con el usuario.

Token = este elemento de tipo string, guarda la información de acceso requerida en el acceso del usuario a las funcionalidades del proyecto, su creación y composición se explicará más adelante en el punto de la funcionalidad de sistema de usuarios.

Por último, se puede ver una captura de ejemplo de un token generado para el acceso de un usuario con nombre “admin”.

```
_id: ObjectId('6558d8c9a41ef00810aec8b3')
Usuario: "admin"
Token: "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyIjoiYWRtaW4iLCJyb2xlIjoxL..."
__v: 0
```

Ilustración 16. Registro de ejemplo colección mongo, tokens

4.1.2.3 Colección registros_calidadAire

Esta colección se encarga de guardar los registros actuales de cada estación de medición de la calidad del aire proporcionados por el portal La Palma Visualizer. Esta opción permite a la web mostrar la vista más real posible del estado del aire en la isla de La Palma.

Veamos el esquema de datos por el cual se define esta colección desde la extensión, del lado servidor, Mongoose.

```
Nombre: {
    type: String,
    required: true,
    max: 100
},
Fecha: {
    type: String,
    required: true,
    max: 100
},
Temperatura: {
    type: Number,
    required: true
},
Humedad: {
    type: Number,
    required: true
},
CO: {
    type: Number,
    required: true,
    min: 0
},
NO2: {
    type: Number,
    required: true,
    min: 0
},
O3: {
    type: Number,
    required: true,
    min: 0
},
SO2: {
    type: Number,
```

```
        required: true,  
        min:0  
    },  
    Latitud: {  
        type: Number,  
        required: true  
    },  
    Longitud: {  
        type: Number,  
        required: true  
    }  
}
```

A continuación, se explica brevemente cada campo.

Nombre = es el nombre que tiene la estación meteorológica. Campo obligatorio en el modelo, ya que aporta datos fundamentales en el proyecto, tiene un máximo de 100 caracteres, y un índice de búsqueda para facilitar peticiones.

Fecha = este campo informa con su valor de tipo texto, el momento exacto en el que fue tomado dicho registro. Tiene una longitud máxima de 100 caracteres y es requerido para los registros de este modelo.

Temperatura = da el valor numérico de la temperatura en grados Celsius (°C), campo obligatorio.

Humedad = entrega el valor numérico del porcentaje (%) de humedad existente en el aire, campo de carácter obligatorio.

CO = indica el valor numérico de monóxido de carbono expresado en microgramos por metro cúbico ($\mu\text{g}/\text{m}^3$), campo de carácter obligatorio y con mínimo de cero para evitar posibles valores negativos, resultado de errores en estaciones.

NO2 = indica el valor numérico de dióxido de nitrógeno expresado en microgramos por metro cúbico ($\mu\text{g}/\text{m}^3$), campo de carácter obligatorio.

O3 = indica el valor numérico de ozono expresado en microgramos por metro cúbico ($\mu\text{g}/\text{m}^3$), campo de carácter obligatorio y con mínimo de cero para evitar posibles valores negativos, resultado de errores en estaciones.

SO2 = indica el valor numérico de dióxido de azufre expresado en microgramos por metro cúbico ($\mu\text{g}/\text{m}^3$), campo de carácter obligatorio y con mínimo de cero para evitar posibles valores negativos, resultado de errores en estaciones.

Latitud = muestra el valor numérico de la distancia existente entre el Ecuador y la ubicación de la estación meteorológica, expresada en grados bajo el sistema de referencia WGS84. Este dato es obligado, ya que es crucial para la representación del registro en el visor web.

Longitud = muestra el valor numérico de la distancia existente entre el meridiano de Greenwich, origen, y la ubicación de la estación meteorológica, expresada en grados, bajo el sistema de referencia WGS84 [50]. Este dato es obligado, ya que es crucial para la representación del registro en el visor web.

A continuación, se mostrará un ejemplo de un registro que sigue esta estructura de datos.

```
_id: ObjectId('64abaa0f378d726ac0461783')
Nombre: "AQ9 - Puerto Tazacorte"
Fecha: "10/07/2023 02:02"
Temperatura: 25.122835159
Humedad: 79.670402527
CO: 205.179992676
NO2: 1
O3: 1
SO2: 0
Latitud: 28.65155948
Longitud: -17.94571698
```

Ilustración 17. Registro de ejemplo colección mongo, registros_calidadAire

Como se puede ver en la ilustración anterior, el registro de esta estación “AQ9 – Puerto Tazacorte”, sigue perfectamente la estructura descrita en este mismo apartado, con el añadido de un campo “_id”, insertado obligatoriamente por MongoDB en todas sus colecciones, para facilitar el manejo de la información expuesta en la base de datos.

4.1.2.4 Colección historial_registros_calidadAire

Esta colección guarda todos los registros provenientes de estaciones de calidad del aire que hayan pasado alguna vez por la misma aplicación, exactamente por la colección *registros_calidadAire*, creando así un histórico de registros que se dispondrá a los usuarios según la estación seleccionada.

Por otro lado, el esquema de datos atribuido a esta colección es exactamente el mismo que la colección explicada en el apartado anterior, *registros_calidadAire*, así mismo que parte de los datos mostrados en esta, coinciden con los existentes en la colección de *registros_calidadAire*, ya que los registros se guardan en ambas colecciones.

4.1.2.5 Colección registros_incidencias

Esta colección de datos maneja los datos de incidencias de todo tipo captadas en la isla de La Palma. Sus orígenes de información pueden ser dos:

1. La importación de datos a tiempo real del Mapa de estado del tráfico e incidencias proporcionado por la DGT.
2. La colaboración de usuarios o administradores que hayan introducido alguna información de incidencias detectadas en la isla, desde la aplicación web.

En cuanto a la estructura de datos de esta colección, definida gracias a la tecnología de mongoose en el lado servidor de la aplicación. Esta consta con muchos campos producto del modelo ya manejado desde la dirección general de tráfico, y también con varios campos introducidos para el manejo de incidencias desde la aplicación web.

```
Nombre: {  
    type: String,  
    required: false,  
    max: 100  
},  
Nombre_es: {  
    type: String,  
    required: false,  
    max: 100  
},  
Nombre_en: {  
    type: String,  
    required: false,  
    max: 100  
},  
  
Imagen: {  
    data: Buffer,  
    contentType: String,  
},  
Fecha: {  
    type: String,  
    required: true,  
    max: 100  
},
```

```

Tipo: {
    type: String,
    required: true,
    enum: [
        'ACC',
        'DER',
        'INC',
        'INU',
        'CAL',
        'ERU',
        'ESC',
        'OTR',
        'OBR',
        'OLA'
    ]
},
Descripcion: {
    type: String,
    required: false,
    max: 300
},
Descripcion_es: {
    type: String,
    required: false,
    max: 300
},
Descripcion_en: {
    type: String,
    required: false,
    max: 300
},
Latitud: {
    type: Number,
    required: true
},
Longitud: {
    type: Number,
    required: true
},
Usuario: {
    type: String,
    index: true,
    max: 100
},

```

```
Administrator: {
    type: String,
    max: 100
},

Validada: {
    type: Boolean,
    required: true,
    default: false,
},
Origen: {
    type: String,
    required: true,
    enum: [
        'DGT',
        'APP',
    ],
},
Code: {
    type: Number,
    index: true,
}
```

A continuación, se procederá a exponer brevemente cada atributo del modelo aquí mostrado.

Nombre = texto breve que permite conocer de forma rápida la incidencia registrada. Con un número máximo de 100 caracteres, y de carácter no obligatorio, ya que el contenido de este campo suele existir en dependencia de si la fuente de datos origen, no es un administrador de la aplicación web.

Nombre_es = este campo representa el nombre de la incidencia en español, tiene un máximo de 100 caracteres y no es un atributo requerido ya que se crea tan solo haber sido revisada la incidencia por un administrador y este aporte información relevante a traducciones.

Nombre_en = igual que el campo anterior pero en inglés.

Imagen = este es un campo especial en el modelo de datos, ya que se ocupa de guardar únicamente información referida a un archivo imagen, que pueda describir la incidencia relatada, no es de carácter requerido.

Fecha = atributo de tipo texto, que indica el momento exacto en el cual la incidencia fue captada si proviene de la DGT o insertada en la base de datos mongo si proviene de la aplicación web. Tiene un máximo de 100 caracteres, y es de carácter obligatorio.

Tipo = en este campo se categoriza la incidencia dentro de una lista de eventos según el que se acomoda más a la descripción de esta. El dato es de tipo texto y de carácter obligatorio, además solo puede contener los valores existentes en el array de enum, acrónimos de los siguientes eventos.

Acrónimo en la enumeración	Evento relacionado
ACC	Accidente de Trafico
DER	Derrumbe
INC	Incendio
INU	Inundación
CAL	Calima
ERU	Erupción
ESC	Escape de gases
OTR	Otro
OBR	Obras
OLA	Alto oleaje

Tabla 2. Acrónimos de tipos de incidencia

Este sistema de abreviación de los tipos de incidencias permite a la aplicación una mayor facilidad a la hora de interpretar los datos en los idiomas inglés y español.

Descripcion = este campo de tipo texto presenta información más detallada de las incidencias registradas. Tiene un máximo de 300 caracteres, y no es obligatorio ya que no es considerado crucial para la interpretación de algunas incidencias.

Descripcion_es = al igual que el campo anterior, representa la información adicional de la incidencia en castellano, también tiene un máximo de 300 valores y no es requerido. Su funcionalidad principal se da al ser aprobada la incidencia registrada, por un administrador del proyecto, y así figurar una información formal en español.

Descripcion_en = este campo es igual al campo anterior, pero en inglés.

Latitud = muestra el valor numérico de la distancia existente entre el Ecuador y la ubicación estimada de la incidencia, expresada en grados bajo el sistema de referencia WGS84. Este dato es obligado, ya que es crucial para la representación del registro en el visor web.

Longitud = muestra el valor numérico de la distancia existente entre el meridiano de Greenwich, origen, y la ubicación estimada de la incidencia registrada, expresada en grados, bajo el sistema

de referencia WGS84. Este dato es obligado, ya que es crucial para la representación del registro en el visor web.

Usuario = este campo existe cuando la fuente de origen de datos es un usuario desde la aplicación web. Entonces en el registro de incidencia se guarda su nombre de usuario, que quedara invisible al Front-End del proyecto, y solo se usará para crear un canal de comunicación, por correo, por el que se le aporte información sobre el estado de su propuesta de incidencia. Su valor es de tipo texto, y máximo 100 caracteres al igual que el existente en el modelo de la colección usuarios. A parte se le ha asignado un índice de búsqueda para agilizar los procesos de comunicación y consulta.

Administrador = este campo existe cuando un administrador del proyecto ha realizado cambios en una incidencia, para así tener un registro de que incidencia manejo que administrador, ante posibles consultas o coordinaciones, al final muestra el nombre de usuario de este administrador, por lo que sigue las mismas características que el campo usuario anterior.

Validada = valor de tipo booleano que indica si la información de la incidencia está verificada. Este campo maneja también la visualización de la incidencia según el usuario que acceda a la aplicación, distinguiendo tres configuraciones

Tipo de usuario	Incidencias visibles
Sin registrar	Solo se envían las incidencias validadas, con valor true en este campo
Registrado	Se le entrega la información de las incidencias validadas con valor true en este campo, y las no verificadas, con valor false, cuyos elementos usuario, coincidan con su nombre de usuario, ya que esto implica, que este usuario registro dichos datos
Administrador	Se le envía la información de todas las incidencias estén validadas o no, vengan de un origen de datos u otro, para facilitarles la verificación de estas

Tabla 3. Visibilidad de incidencias por usuario

Origen = este valor de tipo texto determina el origen propio de la información de la incidencia, distinguiendo si este fue desde el portal de la dirección general de tráfico, o algún usuario autorizado de la aplicación web. Este valor es requerido ya que es importante para la representación y distinción de las incidencias en el visor. Además, sigue una enumeración de dos

valores acrónicos, dejando la puerta abierta a sumar más elementos correspondientes a otros orígenes que enriquezcan los datos del proyecto.

Acrónimo en la enumeración	Origen relacionado
DGT	Dirección General de Tráfico
APP	Aplicación web

Tabla 4. Acrónimos de origen de incidencias

Code = este campo es específico de las incidencias provenientes de la DGT, ya que facilita la identificación de sus registros, y comparar cuales siguen siendo vigentes en el portal de origen y cuales no, y por tanto están desactualizados. Es de tipo número, y presento un índice que ayude a su consulta por este campo.

Ya mencionados todos los elementos del modelo de datos de la colección de incidencias se procederá a mostrar un ejemplo de incidencia que puede tener esta colección.

```
_id: ObjectId('6555f6369c194d08f58cfce9')
Nombre: "OBRAS / LP-109 (0.0 - 15.09) (Incidencia sin editar, origen DGT)"
Nombre_es: "Corte de carretera Barlovento"
Nombre_en: "Windward road cut"
Tipo: "OBR"
Fecha: "13/06/2023 00:09"
Descripcion: "OBRA / MANTENIMIENTO VIA por OBRAS con circulación interrumpida en: ..."
Descripcion_es: "Via cortada por obras con circulación interrumpida en la carretera LP..."
Descripcion_en: "Road closed due to works with interrupted circulation on the LP-109 hi..."
Origen: "APP"
Validada: true
Code: "8690052"
Latitud: 28.82616
Longitud: -17.805876
Administrador: "admin"
```

Ilustración 18. registro de ejemplo colección mongo, registros_calidadAire

Como se ve en la captura de pantalla en los datos se presenta una versión tanto en español como en inglés para así facilitar el acceso a la información a posibles usuarios de otras lenguas que estén interesados en la situación de La Palma. Cabe destacar, como se ha hecho en colecciones anteriores que se añade al modelo descrito un campo “_id”, propio de la tecnología MongoDB, que facilita la consulta y transformación de los registros.

4.1.2.6 Colección imagenes_incidencias_default

Esta colección guarda archivos de imagen auxiliares, correspondientes cada registro a un tipo de incidencia concreto. Esto es necesario ya que a pesar de no ser un valor obligatorio el campo imagen en el modelo de datos de registros_incidencias, si es necesaria una imagen para la correcta visualización de la información de la incidencia en el portal. Se establece una lógica de petición

a esta colección cuando la solicitud de información de incidencia presenta ausencia en este apartado, para así transmitir al cliente la información en el formato correcto para una buena visualización.

Por otro lado, la estructura de datos de esta colección es sencilla ya que no trata datos principales sino datos complementarios a los correspondientes de las incidencias, En los siguientes párrafos se repasará el modelo establecido.

```
    Tipo: {
        type: String,
        required: true,
        enum: [
            'ACC',
            'DER',
            'INC',
            'INU',
            'CAL',
            'ERU',
            'ESC',
            'OTR',
            'OBR',
            'OLA'
        ]
    },
    Imagen: {
        data: Buffer,
        contentType: String,
    },
}
```

Tipo = este campo sigue las mismas características que el campo Tipo de la colección de *registros_incidencias* explicadas en el apartado anterior, ya que es la clave común que tendrán ambas colecciones para relacionarse. Describe los eventos, los cuales pueden caracterizar mejor cada incidencia y sigue la categorización explicada en la tabla 1.

Imagen = este campo imagen sigue las características del semejante, con el mismo nombre en el modelo *registros_incidencias*, y al igual guarda el archivo de la imagen relacionada con el **Tipo**.

A continuación, se mostrará un ejemplo de registro de la colección descrita.

```
_id: ObjectId('6494d58cb8069880664cafb6')
Tipo: "ESC"
» Imagen: Object
```

Ilustración 19. Registro de ejemplo colección mongo, *imagenes_incidencias_default*

La formación de esta colección fue al finalizar el desarrollo de la implementación de incidencias en el proyecto, ya que su contenido es estático de tal forma que hay una imagen auxiliar para cada tipo definido en el modelo de datos, y no se añadirán ni borrarán más registros, para permitir el correcto funcionamiento de la aplicación.

4.1.2.7 Colección registros_alarmas

Esta colección contiene los registros de las alarmas creadas por los usuarios del visor web, las cuales tendrán los datos necesarios para interactuar con las incidencias que puedan surgir e informar al usuario al instante.

La estructura de datos de esta colección es la siguiente:

```
Nombre: {
    type: String,
    required: true,
    max: 100
},
Rango: {
    type: Number,
    required: true,
    min: 99,
    max: 3001,
},
Latitud: {
    type: Number,
    required: true
},
Longitud: {
    type: Number,
    required: true
},
Usuario: {
    type: String,
    required: true,
    index: true,
    max: 100
},
Activada: {
    type: Boolean,
    required: true,
    default: false
}
```

A continuación, se describirá brevemente cada elemento del modelo de datos.

Nombre = valor de tipo texto, enseña la frase o palabra con la que quiere referirse el usuario a esta alerta, es un campo requerido para la correcta visualización de los datos, y con un máximo de 100 caracteres.

Rango = atributo de tipo numérico rango, indica a la aplicación la distancia máxima a la que debe estar la alarma a la incidencia, para que esta se active y comunique al usuario. El valor es obligatorio para el correcto funcionamiento de la herramienta, y tienen unos valores umbrales de 99 y 3001, permitiendo así solo valores de entre 100 a 3000, para procurar un correcto uso de la funcionalidad por parte del usuario.

Latitud = muestra el valor numérico de la distancia existente entre el Ecuador y la ubicación decidida por el usuario para la alarma, expresada en grados bajo el sistema de referencia WGS84. Este dato es obligado, ya que es crucial para la representación de la alarma en el visor web.

Longitud = muestra el valor numérico de la distancia existente entre el meridiano de Greenwich, origen, y la ubicación decidida por el usuario para la alarma, expresada en grados, bajo el sistema de referencia WGS84. Este dato es obligado, ya que es crucial para la representación de la alarma en el visor web.

Usuario = este campo de tipo texto indica el nombre del usuario que creó este registro de alarma, para así poder relacionar las alarmas con los usuarios y enviar la información debida a cada usuario. Un usuario solo recibe las alarmas que ha creado él mismo, por esto el campo es obligatorio, y se le ha añadido un índice de búsqueda para agilizar las peticiones por este campo.

Activada = por último este campo describe el estado de la alarma registrada, siendo de tipo booleano, true si se ha detectado una incidencia dentro del rango definido y false si no hay incidencias reportadas al usuario dentro del rango. Este atributo es obligatorio ya que es clave en la funcionalidad de las alarmas, y de valor predeterminado false, ante cualquier registro nuevo sin definir. En la siguiente imagen se podrá observar los datos de un ejemplo de alarma guardada en el proyecto.

```
_id: ObjectId('657982efb7dd1f1cacc53e70')
Nombre: "Alarma casa"
Rango: 3000
Latitud: 28.651685191661976
Longitud: -17.91429948361714
Usuario: "user1"
Activada: false
__v: 0
```

Ilustración 20. Registro de ejemplo colección mongo, registros_alarmas

Como se puede ver en el ejemplo, esta es la alarma denominada “Alarma casa” para el usuario de nombre user1y permanecerá con el campo activada como false hasta que se detecte por la aplicación una incidencia cercana.

4.1.3 Publicación y servicio de capas Geoserver

Para el resto de información georreferenciada se optó por la descarga de las capas desde las distintas fuentes de datos para modificar su estilo y datos desde QGIS, para después publicarlas en Geoserver desde el plugin de GeoCat Bridge y finalmente emplear la tecnología de servicio de capas proporcionada por Geoserver. En este caso gracias al estándar de comunicación, WMS, se da el traslado de la información desde el servidor web, hasta el Front-End del proyecto. En cuanto a los datos y características presentes en estas capas, se repasarán a continuación en los siguientes apartados.

4.1.3.1 Capa de parcelas

Los datos de esta capa provienen de los servicios Inspire de la Sede Electrónica del Catastro [51], y pertenecen a las parcelas registradas en la isla de La Palma, por lo que cubren toda la superficie de la isla. Como se ve en la siguiente imagen el material de datos publicado desde Geoserver no presenta un gran número de atributos.

Propiedad	Tipo	Nulo permitido	Min/Max Occurrences
geom	MultiPolygon	true	0/1
gml_id	String	true	0/1
nationalCadastralReference	String	true	0/1

Ilustración 21. Destalles de los datos de las parcelas

Como se puede ver se compone por tan solo tres elementos, el correspondiente a la geometría de tipo polígono, geom, su identificador de tipo texto, gml_id, y por último la referencia catastral asociada a cada parcela, mediante el campo nationalCadastralReference, el cual se empleará más tarde, como etiqueta en visualización de la capa desde el visor web. Esto se da ya que se sigue el criterio de solo contar en los servicios de datos con información empleada realmente en el proyecto, lo que supone desde QGIS, antes la publicación de las capas, la eliminación de varios campos en el modelo no interesantes para el cometido del proyecto.

4.1.3.2 Capa de edificaciones

En cuanto esta capa de información de edificaciones proviene de los datos brindados por el portal Open Data La Palma, el cual presenta un dataset con información geoespacial de los edificios encontrados en la isla bonita [52]. En cuanto a los datos trasladados al Geoservver, mediante la publicación de la capa en Quantum GIS.

Propiedad	Tipo	Nulo permitido	Min/Max Occurrences
geom	Polygon	true	0/1
OBJECTID	Integer	true	0/1

Ilustración 22. Destalles de los datos de las edificaciones

Como se ve en la ilustración, solo existen dos campos que se han preservado desde la descarga del origen de datos al servicio de la capa, y estos son el correspondiente a la geometría caracterizadas como polígono, geom, y su identificador en formato numérico entero, OBJECTID.

4.1.3.3 Capa de carreteras

La capa de carreteras presenta las vías de circulación de vehículos principales en la isla, autopistas, autovías y demás carreteras relevantes en la comunicación de la isla, su origen de datos es el dataset de carreteras [53]del portal de descargas Open Data La Palma.

	Nombre	Tipo	Source	Descripción	Nulo permitido	Eliminar
1	the_geom	MultiLineString	the_geom		<input checked="" type="checkbox"/>	
2	OBJECTID	Long	OBJECTID		<input checked="" type="checkbox"/>	
3	NOMENCLATU	String	NOMENCLATU		<input checked="" type="checkbox"/>	

Ilustración 23. Destalles de los datos de las carreteras

En el modelo de datos de carreteras, destaca la existencia de tan solo tres campos, *the_geom*, que indica la geometría de tipo multilínea de las carreteras, el elemento *OBJECTID*, que identifica mediante un numero los datos. Por último, *NOMENCLATU*, campo el cual aporta los códigos o nombres identificativos de las carreteras.

4.1.3.4 Capa de colada volcánica

El estudio de las características volcánicas de la isla es una parte fundamental de las posibilidades que ofrece la aplicación web, por ello la incorporación de datos que aporten en este tema es de suma relevancia, en este caso se añade a los datos información sobre el perímetro de la colada volcánica producto de la erupción del septiembre del año 2021. Datos descargados de la capa “perímetro dron 211101 093 sur” [54] del portal Open Data La Palma.

Propiedad	Tipo	Nulo permitido	Min/Max Occurrences
geom	Polygon	true	0/1
OBJECTID_1	Integer	true	0/1
OBJECTID	Integer	true	0/1

Ilustración 24. Destalles de los datos del perímetro de colada volcánica

En cuanto a los datos contenidos en esta capa, como se puede ver consta de tres, con el campo *geom*, con las geometrías poligonales del perímetro de la colada, y dos campos de identificación denominado como *OBJECTID_1* y *OBJECTID*.

4.1.3.5 Capa de bocas eruptivas

Por su lado, estas bocas eruptivas proporcionan al proyecto las localizaciones de fuga de material volcánico, como lava, gases, etc., que estuvieron activos en la erupción de finales de 2021, aportando así una información muy relevante al proyecto. Esta información geográfica fue obtenida de la capa de “bocas eruptivas IGN” [55] en Open Data La Palma. La estructura de datos impuesta en esta capa es la siguiente:

Propiedad	Tipo	Nulo permitido	Min/Max Occurrences
geom	Point	true	0/1
OBJECTID	Integer	true	0/1

Ilustración 25. Destalles de los datos de las bocas eruptiva

En este modelo, como en los anteriores se busca el transmitir solo la información estrictamente necesaria para la correcta visualización de los datos en el visor, por ello solo existen servidos dos elementos, *geom*, propio de la geometría puntual de las bocas, y el *OBJECTID*, que sirve de identificador de las geometrías.

4.1.4 Archivo de terremotos

La información sobre sismos en la isla puede ser muy ilustrativa a la hora de valorar la situación de la actividad volcánica en La Palma, por esto la descarga de datos de este tipo es muy importante. La obtención de esta información consiste en la incorporación al proyecto de un archivo codificado en java [56], proporcionado por el Instituto Geográfico Nacional (IGN) en su web de “Información sísmica” [57], al lado servidor del proyecto. Este archivo contiene la información de los temblores ocurridos en el archipiélago canario en distintos lapsos de tiempo, siendo estos representados en tres variables java a las que se les asigna el valor de tres colecciones de objetos **GEOJSON**, el primero con los datos de terremotos de los últimos 3 días, el segundo con los de los últimos 15 días y el tercero con los sismos de los últimos 90 días. Este formato de información permite al proyecto mostrar los datos con una variable temporal muy interesante.

En cuanto al formato la estructura de datos de estos registros la estudiaremos mediante el siguiente ejemplo de sismo:

```
"type": "Feature",
"geometry": {
    "type": "Point",
    "coordinates": [
        -16.8558,
        28.5948
    ]
},
```

```
"properties": {  
    "evid": "es2023rnmc",  
    "typeplot": 0,  
    "mag": "1.8",  
    "magtype": "mbLg",  
    "intensidad": "",  
    "depth": "42",  
    "fecha": "2023/09/07 05:42:14",  
    "fechalocal": "2023/09/07 06:42:14",  
    "loc": "ATLANTICO-CANARIAS",  
    "tipoint": "",  
    "onda": "http://www.ign.es/web/ign/portal/vlc-senales-sismicas/-/senales-sismicas/getInfoHora?fecha=2023-09-07&tipoFO=2&tipoSP=2&estacion=TE01&nombrefichero=TE01_2023-09-07&hora=05-06"
```

Como se ve los registros de terremotos contienen una buena variedad de datos, referentes a la localización, fecha del acontecimiento, intensidad, entre otros. Sin embargo, no todos estos datos son útiles para el proyecto, de hecho, la mayoría, a pesar de aportar información relevante de los sismos no son imprescindibles para la representación de estos. Por ello el proyecto únicamente aprovecha los siguientes elementos:

Coordinates = de este campo se extraen las coordenadas que indican la posición del punto, los valores numéricos de la latitud y la longitud respectivamente, bajo el sistema de referencia WGS84 (World Geodetic System 1984).

Mag = la magnitud del sismo permite a la aplicación mostrar la intensidad de la actividad volcánica en el tiempo, mediante representaciones de mapas de calor. Los datos mostrados en estos elementos son de tipo texto, conteniendo un valor numérico decimal que indica la medición de la energía liberada dentro de la escala de magnitud local (ML) o escala de Richter.

4.2 HARDWARE

En cuanto a los elementos requeridos en este aspecto, son mínimos ya que el total del proyecto gira alrededor del desarrollo del software, la búsqueda e implementación de datos, y la panificación de funcionalidades para la aplicación. Por lo que el único elemento de hardware central requerido en el proyecto ha sido el equipo electrónico con el que cuento en mi hogar, PC (Personal Computer), portátil, periféricos y distintos almacenamientos externos.

4.3 SOFTWARE

El software necesario para el desarrollo del proyecto, repasado en los distintos puntos dedicados a las tecnologías empleadas, fueron seleccionado totalmente como licenciado open source, o de libre uso, para facilitar el trabajo con estos softwares, y procurar un buen desarrollo del proyecto.

4.4 METODOLOGÍA

En este apartado se estudia cómo se procedió al desarrollo del proyecto, la división y etapas del trabajo realizado, y lo cometido en cada momento.

4.4.1 Estudio del marco del proyecto

Ya con la decisión tomada de realizar el Trabajo Final de Grado alrededor de una aplicación que pueda servir de ayuda y soporte en la comunicación de la actualidad de la isla de La Palma y su actividad volcánica, surgió este primer paso que atendía a la necesidad de ampliar mi conocimiento sobre los organismos presentes que ya cumplían esta función, los datos que estos manejan y en qué pueden ser útiles al proyecto, permitiéndome así definir un nuevo enfoque que aporte utilidad a los habitantes de la isla.

En esta etapa y mediante la observación de distintas aplicaciones que abordan este tema, me inspire para concretar los cimientos del proyecto, los objetivos que se buscan, la variedad de estrategias que se pueden seguir, el alcance y público interesado en el trabajo, así como la búsqueda de fuentes de datos que pudiesen enriquecer el contenido de la aplicación.

4.4.2 Planteamiento de requisitos

Este segundo espacio en el desarrollo del Trabajo Final de Grado consistió en la cosecha e inventiva de distintas funcionalidades que pudiesen ser necesarias y destacables para el manejo de la aplicación. La definición de cómo llevar a cabo la comunicación de los datos hacia el cliente, encontrar herramientas de utilidad que pueden ayudar al control de la aplicación, sistemática de roles, etcétera, son temas que se desarrollaron en esta fase del proyecto, en la cual se aprovechó la aplicación de mockups y esquematización Balsamiq (véase Anexo 1), para materializar el diseño de estas funcionalidades, facilitando la creación y desarrollo de ideas, así como el rechazo y atascamiento de otras que convirtieron el proyecto en lo que es hoy en día.

4.4.3 Planificación y ejecución del desarrollo de la aplicación

En esta etapa ya con una idea dibujada sobre cómo será la app, se construyó una hoja de ruta en la que se dispusieron los pasos a seguir, las estimaciones de tiempo, la definición de entregables y el orden de tareas. Con el objetivo de seguir un camino claro que permitiese un cómodo y óptimo desarrollo de la aplicación.

Un componente importante en esta etapa fue la elección de una metodología de trabajo, en este caso debido a las características del proyecto y como interés personal, se escogió la metodología ágil de Scrum, adquiriendo así el proceso un enfoque de concepción de versiones incrementales de la aplicación, abriendo el proyecto a la incorporación de nuevas funciones, ideología de trabajo por períodos de entrega definidos, etcétera.

4.4.4 Redacción de la memoria y entregables

Por último, un hito muy importante en el proyecto es la realización de la memoria de trabajo, ya acabado el desarrollo de la aplicación y con una versión final de todos los aspectos de esta, se elabora la documentación relativa al trabajo realizado, con intención de poder comunicar todo el contenido y capacidad del proyecto. Por ello el tiempo y dedicación en la redacción aquí expuesta logró ser una etapa principal en el trabajo, en la cual además de exponer lo realizado, me ayudó a valorar el camino recorrido y el interés del proyecto.

5. EXTRACCIÓN E INTEGRACIÓN DE DATOS

La tecnología de Pentaho Data Integration presenta varios formatos de procesamiento de datos, las principales son las transformaciones y los Jobs o trabajos. Las transformaciones son procesamientos de datos estándar, y los Jobs permiten la gestión de las transformaciones y demás elementos para configurar su ejecución.

5.1 ESTACIONES DE CALIDAD DEL AIRE

Los datos pertenecientes a la calidad del aire de la isla de La Palma son obtenidos gracias al portal de La Palma Visualizer, el cual ofrece datos en tiempo real de las estaciones meteorológicas de la isla, por ello y para preservar la verosimilitud de los datos la ejecución de los procesos de descarga es de una vez a la hora.

5.1.1 Modelo de datos de entrada

En cuanto al modelo de datos de llegada son datos en formato JSON dispuestos en un objeto array con entradas correspondientes a los registros actuales de cada estación meteorológica. A continuación, se mostrará un ejemplo de registro recibido sin ninguna transformación previa.

```
"id": "AirQualitySmartSpot:HOP30aea4ca5d4a",
"type": "AirQualitySmartSpot",
"CO": {
    "type": "Number",
    "value": 579.0700073,
    "metadata": {}
},
"NO2": {
    "type": "Number",
    "value": 33.3400001,
    "metadata": {}
},
"O3": {
    "type": "Number",
    "value": 46.7400016,
    "metadata": {}
},
"SO2": {
    "type": "Number",
    "value": 48.8400001,
    "metadata": {}
},
```

```
"TimeInstant": {
    "type": "DateTime",
    "value": "2022-09-19T10:25:56.00Z",
    "metadata": {}
},
"deployment": {
    "type": "Text",
    "value": "cabildo",
    "metadata": {}
},
"humidity": {
    "type": "Number",
    "value": 42.7985038,
    "metadata": {}
},
"illuminance": {
    "type": "Number",
    "value": 0,
    "metadata": {}
},
"location": {
    "type": "geo:json",
    "value": {
        "type": "Point",
        "coordinates": [
            -17.8437232,
            28.4942569
        ]
    },
    "metadata": {}
},
"name": {
    "type": "Text",
    "value": "AQ2 - Casa Forestal Fuencaliente",
    "metadata": {}
},
"operationalStatus": {
    "type": "Text",
    "value": "CONNECTED",
    "metadata": {
        "timestamp": {
            "type": "DateTime"
        }
    }
}
```

```

        "value ": "2022-09-19T10:25:38.00Z "
    }
},
"pollutants ": {
    "type ": "List ",
    "value ": ["NO2 ", "O3 ", "SO2 ", "CO "],
    "metadata ": {}
},
"temperature ": {
    "type ": "Number ",
    "value ": 28.6903953,
    "metadata ": {}
},
"dateModified ": {
    "type ": "DateTime ",
    "value ": "2022-09-19T10:25:56.00Z ",
    "metadata ": {}
}
}

```

Como se ve el objeto recibido tras la petición mantiene un gran número de campos, los cuales no todos serán útiles para el proyecto. En las siguientes líneas se explicará brevemente el contenido de cada campo.

Id = identificador de los registros enviados por las estaciones.

Type = aquí se indica el tipo de registros que ofrece la estación.

CO = este campo describe mediante un objeto JSON la cantidad de monóxido de carbono, expresada en microgramos por metro cúbico ($\mu\text{g}/\text{m}^3$), que detecta la estación meteorológica en un determinado momento.

NO2 = este campo describe mediante un objeto JSON la cantidad de dióxido de nitrógeno, expresada en microgramos por metro cúbico ($\mu\text{g}/\text{m}^3$), que detecta la estación meteorológica en un determinado momento.

O3 = este campo describe mediante un objeto JSON la cantidad de ozono, expresada en microgramos por metro cúbico ($\mu\text{g}/\text{m}^3$), que detecta la estación meteorológica en un determinado momento.

SO2 = este campo describe mediante un objeto JSON la cantidad de dióxido de azufre, expresada en microgramos por metro cúbico ($\mu\text{g}/\text{m}^3$), que detecta la estación meteorológica en un determinado momento.

TimeInstant = este campo describe mediante un objeto JSON el momento exacto en el cual se crea el registro enviado por la estación meteorológica.

Deployment = este campo describe mediante un objeto la organización encargada de mantener la estación meteorológica.

Humidity = este campo describe mediante un objeto JSON la cantidad de humedad, expresada en tanto por ciento, existente en el aire, que detecta la estación meteorológica en un determinado momento.

Illuminance = este campo describe mediante un objeto la cantidad de iluminación que detecta la estación meteorológica en un determinado momento

Location = este campo describe mediante un objeto JSON la localización de la estación, a través de la entrega de un array con los valores de las coordenadas, longitud y latitud, en el sistema de coordenadas WGS84 (World Geodetic System 1984).

Name = este campo describe mediante un objeto JSON el nombre de la estación meteorológica al que pertenece el registro recibido.

OperationalStatus = este campo explica mediante un objeto el estado de la estación en el momento de la creación del registro.

Pollutants = este campo explica mediante un objeto los medidores de gases que presenta la estación meteorológica, a través de un objeto array cuyos elementos son las formulaciones químicas de los gases que puede medir la estación.

Temperatura = este campo describe mediante un objeto JSON la temperatura, expresada en grados centígrados, que detecta la estación meteorológica en un determinado momento.

DateModified = este campo describe mediante un objeto el momento en el que se recibió el registro en la central de almacenamiento de datos.

5.1.2 Extracción de registros de estaciones de calidad del aire

La gestión de los datos de estaciones de medición de calidad del aire requiere de dos transformaciones diferentes, una de extracción de registros y otra de creación de histórico.

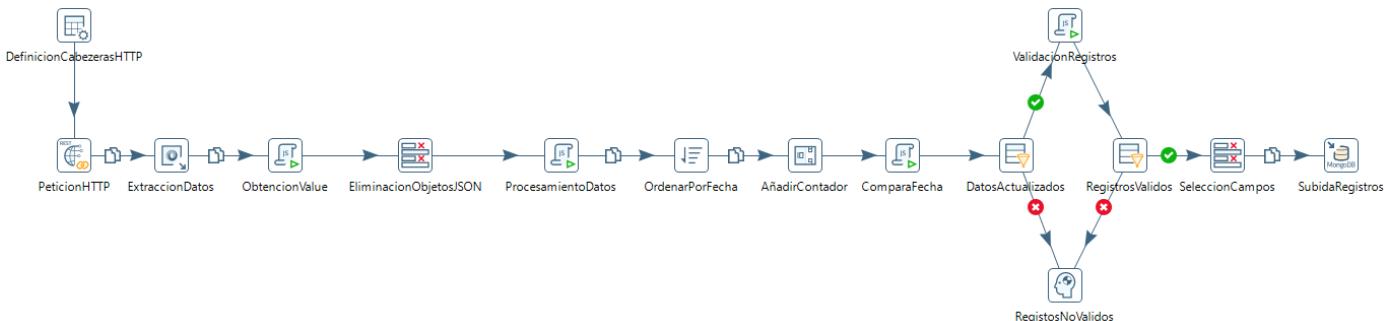


Ilustración 26. Transformación Pentaho Data Integration (PDI) de extracción de registros de calidad del aire

Esta transformación trata la inserción de registros actualizados en la base de datos Mongo. Para ello, se han ejecutado un cierto número de pasos mostrados en la ilustración 14, que se explicaran a continuación.

DefinicionCabezasHTTP: este primer paso es denominado como data grid en la aplicación de Pentaho, y permite la inyección de campos dentro del proceso, en este caso los campos contendrán los parámetros de *url*, *referer*, *user-agent*, *fiware-service* y *fiware-servicepath*, necesarios para el contacto con la fuente de datos.

PeticionHTTP: en esta parte del proceso se realiza la petición y recepción de los datos de registros de calidad del aire. Mediante la herramienta de REST client y los parámetros definidos en el paso anterior, se enviará una petición de datos desde cliente, al servidor del portal de La Palma Visualizer, el cual entregará al proceso los datos de los registros más actualizados de cada estación meteorológica a servicio la administración pública.

ExtraccionDatos: como se vio en el punto anterior, el modelo de datos de los registros tiene una alta variedad de campos, los cuales no todos serán útiles al proyecto, ya que son utilizados para la gestión interna de la red de estaciones o simplemente se desvían del tema tratado, por lo que en este paso JSON input, se seleccionara del objeto recibido que campos son interesantes: *CO*, *NO2*, *O3*, *SO2*, *TimeInstant*, *humidity*, *location*, *name*, *temperatura*.

ObtencionValue: valor JavaScript modificado, permite la ejecución de sentencias JavaScript que puedan modificar los datos, para este caso, al tener en cuenta que los campos recibidos encapsulan cada uno los valores dentro de un objeto JSON, el código de este paso se encarga de extraer el valor neto de cada objeto en insertarlos en nuevos campos equivalentes a los de origen, para así facilitar futuros tratamientos.

EliminacionObjetosJSON: los procesos de Pentaho Data Integration pueden mover una gran cantidad de datos por ello para mantener un buen rendimiento y pulcritud en la transformación se emplea la herramienta de Selecciona/Renombra valores, para como en este caso eliminar del flujo datos poco útiles a futuro, como lo son en ese momento del procedimiento los campos recibidos con objetos JSON.

ProcesamientoDatos: en este paso de JavaScript se edita el formato de los datos, se separa y extrae individualmente los valores de longitud y latitud del objeto *Localización*, y se crea un campo de tipo fecha a partir del campo *Tiempo*, lo cual nos permitirá establecer comparaciones de tiempo más adelante.

OrdenarPorFecha: este paso permite ordenar filas de un procesamiento bajo el valor de un campo. En este caso se ha ordenado de mayor a menor el campo de tipo fecha generado en el último paso, de tal manera que los registros más actuales quedan en la zona superior de la tabla y los más antiguos en la inferior. Cabe destacar que las estaciones pueden entregar registros antiguos al estar inactivas, por ello los procesos se orientan a controlar esta opción.

AñadirContador: este paso denominado como añadir secuencia en la ETL, añade un campo con valores secuenciales que coinciden con el orden de cada registro en la tabla, siendo el primero y más actualizado el 1, el segundo el 2 y así sucesivamente. Este paso permitirá al proceso un mayor conocimiento del orden de datos.

ComparaFecha: este script disgregara los registros según si son actuales o no, primero interceptando el valor del primer registro, más actualizado, y comparándolo con el resto, de tal forma que se entenderá que el registro es actualizado si la diferencia con el primero es menor a 30 horas, esto permitirá obtener datos que se actualicen con periodicidad y estables, ya que al no ser la creación de registros igual en todas las estaciones, se debe dejar un umbral temporal lo suficientemente grande como para cubrir los registros de estaciones activas y excluir a los de las estaciones sin comunicación. Finalmente, este proceso generara un campo booleano que indicara si el registro es válido o no.

DatosActualizados: paso de filtrar filas, permite al proceso la separación de caminos, bajo una condición, en este caso la actualidad de los registros, indicada en el campo creado en el paso anterior. Generando dos caminos uno con registros validos al paso **ValidacionRegistros** y otro con registros incorrectos al paso **RegistrosNoValidos**.

RegistrosNoValidos: el nombre de este paso en la aplicación de Pentaho es de transformación simulada (no hace nada), se trata de un paso simbólico en el flujo que no realiza ninguna

transformación en los datos pero que permite establecer un flujo natural, en este caso su objetivo es recibir los registros no válidos, ya sea por ser antiguos o por contener datos no relevantes.

ValidacionRegistros: a este código JavaScript le llegan solo los registros actualizados, pero no necesariamente válidos, ya que todavía pueden contener datos no interesantes para la aplicación. Y es que los registros a pesar de tener campos dedicados a la medición de gases como CO, NO₂, etcétera, si estos provienen de una estación sin medidores o con medidores defectuosos, los valores entregados serán incorrectos. Por ello en este script se comprueba que estos sean distintos a nulo, cero o uno, en al menos uno de los cuatro campos de gases.

RegistrosValidos: paso de filtrado de filas que permite al proceso disgregar entre registros validos o no, enviando nuevamente los erróneos al paso explicado anteriormente de *RegistrosNoValidos*, y los correctos a la parte final de la transformación, *SeleccionCampos*.

SeleccionCampos: este Selecciona/Renombra valores, permite eliminar del procedimiento los campos auxiliares que han acompañado a los datos en la transformación, parámetros, booleanos, entre otros. Y así considerar únicamente los datos a guardar en la base de datos. Además, en este paso se ha podido adaptar el tipo de dato de cada campo para que concuerde con el manejado en la colección mongo de destino.

SubidaRegistros: el paso final de la transformación, MongoDB output, permite la inserción de nuevos registros en las bases de datos mongo. Por lo cual se guarda la información de los registros validos en la colección *registros_calidadAire*, de la base de datos del proyecto.

5.1.3 Historial de registros de estaciones de calidad del aire

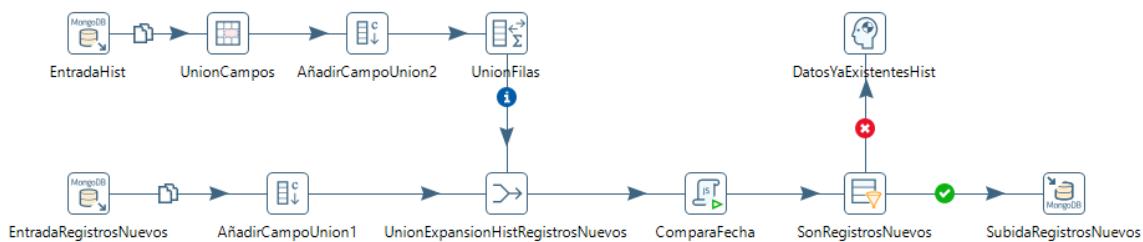


Ilustración 27. Transformación PDI de historial de registros de calidad del aire

Esta transformación se encarga del guardado de registros de calidad del aire ya añadidos a la base de datos del proyecto en una colección alternativa que sirva como histórico de registros, con el deber de poder soportar funcionalidades de peticiones de historial de estaciones a los usuarios. La transformación consta al principio de dos caminos, uno principal con el flujo de datos nuevos y otro secundario con los datos ya existentes en el histórico. A continuación, se explicarán los pasos de la transformación empezando por el camino principal.

EntradaRegistrosNuevos: este paso se denomina MongoDB input, y sirve para obtener datos de una conexión a base de datos mongo, en este caso a los correspondientes a la colección *registros_calidadAire*, a la que se acabaron subiendo los datos en la transformación de extracción anterior. Por lo que se volverá a trabajar con los registros de calidad del aire, pero ya editados y con el formato deseado por el proyecto.

AñadirCampoUnion1: posteriormente al haber dos caminos estos tendrán que converger, y para ello se necesita un nexo o punto común entre los datos, por ello la herramienta de añadir valor constante, se encargara de añadir un valor que actúe como enlace entre ambos caminos.

EntradaHist: para procurar la calidad de los datos del histórico se debe asegurar la no repetición de estos por ello la entrada por este MongoDB input, es fundamental, ya que proporciona al proceso los valores que actúan como identificación de cada registro, es decir, nombre de la estación y fecha dentro del modelo de datos, que nos permitirán distinguir de forma inequívoca cada registro, y así saber si existen en ambas tablas a la vez o no es así. Además, para simplificar el tamaño de datos que maneja el proceso, esta herramienta nos permite ejecutar una query de búsqueda, que permite obtener los valores de estos campos cuyos registros son los más actualizados de cada estación meteorológica, alguna vez implicada en la base de datos.

UnionCampos: esta herramienta es concat field, y como dice su nombre concatena los campos que se le indiquen, esto permitirá que la información obtenida de cada registro del histórico de datos se concentre en tan solo un campo, siendo su valor una cadena de texto, primero el nombre de la estación de medición, un elemento concreto cualquiera como hashtag (#) para separar y luego la fecha.

AñadirCampoUnion2: nuevamente se necesitará de una constante en el flujo que permita la unión de valores, por ello se inserta un nuevo campo con el mismo valor en todos los registros que incurra finalmente en su unión.

UnionFilas: este paso de control de flujo se denomina como agrupar, y da la funcionalidad de concatenar los valores de las filas de un proceso según si tienen un campo con valor común, concretamente el campo introducido en el último paso, que provocará la concatenación de la información ya unida de los registros en una única cadena, con el carácter “%” como separador. Estos tratamientos facilitaran la comparación de los registros de ambas colecciones y el reconocimiento de registros nuevos en el proyecto.

UnionExpansionHistRegistrosNuevos: este paso denominado en Pentaho como multiway merge join, sirve para unir los valores de procesos o caminos, permite la unión de las tablas según un

campo que tengan en común, la constante introducido en pasos anteriores y que tienen mismo valor, por ello resultará la inserción de la misma cadena de información de identificación de registros, creada en la agrupación, en cada fila de registros posiblemente nuevos propios del fujo de datos principal.

ComparaFecha: este script funcionara como un disgregador de registros repetidos, buscando primero en la cadena de texto producto del histórico, el valor del nombre del posible nuevo registro de la estación correspondiente, después si lo encuentra comparará la fecha que dispone esta sección, que es la última fecha que tiene registrada el histórico para datos de esta estación, con la fecha proporcionada por el registro nuevo, si esta es superior y por tanto más actual el registro figurará en un nuevo campo como *nuevo* true, al igual que si no encuentra el nombre en la cadena del histórico, significando esto que es un registro de una estación nueva, sin embargo si la comparación no es exitosa el registro figuraría como repetido y por lo tanto será introducido de nuevo en el histórico.

SonRegistrosNuevos: este paso de filtrar filas permite al proceso dividir el fujo de datos entre registros nuevos y registros repetidos, enviando estos últimos a *DatosYaExistentesHist*, y los registros nuevos al paso *SubidaRegistrosNuevos*.

DatosYaExistentesHist: esta transformación simulada tan solo recibe los datos repetidos en ambas colecciones y así eliminarlo del flujo normal de datos.

SubidaRegistrosNuevos: para concluir la transformación, este MongoDB output, guarda en la colección *historial_registros_calidadAire*, el histórico de registros, los registros nuevos integrados en el proyecto.

5.2 INCIDENCIAS DE LA DGT

Los datos recibidos desde el Mapa de estado del tráfico e incidencias de la DGT, presenta una gran variedad de información enfocada en el estudio del tráfico del estado español, Sin embargo, mucha de esta información, puede ser adaptada y transformada, de tal forma que se convierta en interesante para este proyecto. Por ello a continuación se presentará en primera instancia las características de los datos, y las transformaciones que se le aplican a estos para su correcta subida a la base de datos del proyecto, seleccionado los registros interesantes a este y adaptando los datos de estos al modelo requerido por la colección de incidencias, mencionada anteriormente.

5.2.1 Modelo de datos de entrada

En cuanto, a los datos recibidos de entrada en los procesos, se encuentran varios campos dedicados a la gestión propia de las aplicaciones y tecnologías de la DGT, poco útiles para la aplicación, por ello, aprovechando el siguiente ejemplo de registro de datos, se describirá en los siguientes párrafos los campos atribuidos y su uso o no en el trabajo.

```
"precision": "1",
"poblacion": "MANCHAS (LAS)",
"fecha": "30/11/2022",
"alias": "OTROS / LP-2 (41.0 ~ 43.7)",
"suceso": "OTROS",
"sentido": "AMBOS SENTIDOS",
"descripcion": "<span style=\"margin-top:10px; float:left; clear:both\">
<b>DERRAMAMIENTO</b> con circulación interrumpida en<b></b>: </span> <span
style=\"margin-top:10px; float:left; clear:both\"> - La carretera <b></b> a la altura de <b>MANCHAS (LAS) (SANTA CRUZ DE
TENERIFE)</b> desde el <b> km 41</b> al <b> km 43.7</b> sentido <b>AMBOS
SENTIDOS</b></span><span style=\"margin-top:10px; float:left; clear:both\"> Advertencia:
<b>CORTE TOTAL</b></span>",
"fechaFin": "?",
"lng": -17.880623,
"pkFinal": 43.7,
"provincia": "SANTA CRUZ DE TENERIFE",
"codEle": "7384161",
"causa": "",
"carretera": "LP-2",
"hora": "22:49",
"estado": 1,
"autonomia": "Canarias",
"pkIni": 41,
"icono": "INC_DER DES LS1 CAC.png",
"tipo": "Incidencia",
"horaFin": "?",
"lat": 28.584438,
"nivel": "NEGRO"
```

Precision = en este texto se indica el nivel interno de zoom en el que se muestra el registro, ya que al dirigirse directamente los datos al mapa de incidencias de la DGT, los registros guardan campos como este que manejan la presentación de los datos en su portal, como se entenderá no es relevante para el proyecto, ya que esta funcionalidad la desarrolla el mapa de la DGT para dividir la visualización de los datos según nivel de zoom por importancia del registro, ya que al cubrir toda España, en escalas pequeñas se puede concentrar demasiada información, y dificultar

la presentación del mapa, sin embargo este trabajo, cubre un área significativamente más pequeña, por lo que no sufre esa carga de datos.

Población = este atributo de tipo texto, explica la población en la cual se ha dado la incidencia. Sin embargo, esta información suele mostrarse también en el campo dirigido a la descripción de la incidencia, por lo que puede resultar repetitivo.

Fecha = aquí se indica el día exacto en el que se captó la incidencia, es un valor de tipo texto, y sigue un formato de día / mes / año.

Alias = el alias de la incidencia describe brevemente y según la estrategia definida internamente por la DGT, la incidencia registrada. No suele ser un nombre sencillo, ya que sigue el criterio de mostrar primero el tipo de suceso y su localización en la red de tráfico.

Suceso = este campo es bastante relevante en el modelo, ya que indica el tipo de evento que representa este registro, el campo es de tipo texto y se define según los elementos de la leyenda implementada en mapa de incidencias de la DGT [58]. A parte de los tipos de incidencias se puede distinguir más concretamente las características de estas según su tipo específico el cual se expresa en la app de la DGT según el símbolo designado al registro, este dato se comunica al mapa por el campo **icono**, que se estudiará más adelante.

Tipo de incidencia	Incidencia específica
Retenciones	Retenciones
Puertos	Puertos
Obras	Obras
Eventos	Actividades
	Actividades deportivas
	Rodaje
Restricciones	Vehículos pesados
	Mercancías peligrosas
	Vehículos que necesitan autorización complementaria
	Vehículos especiales

Meteorológicas	Humo / Polvo
	Helada / Hielo
	Lluvia
	Nevada
	Viento
	Otros
Otros	Desconocido
	Derramamiento
	Estado parking
	Obstáculo fijo
	Obstáculo móvil
	Otros
	Tiempos de embarque
	Medidas de regulación
	Embolsamiento de camiones y articulados
	Embolsamiento de vehículos

Tabla 5. Tipos de incidencia del modelo de datos de la DGT

Como se puede ver los datos de incidencias compartidos por la DGT, son muy diversos, y aunque regularmente se centran en el ámbito de carreteras, también hay registros que pueden resultar muy útiles para el proyecto como los de carácter meteorológico o quizás los designados como otros.

Sentido = esta cadena de texto indica la dirección de la carreta en la cual ha surgido la incidencia, este dato se reitera después en el campo dedicado a la descripción, por lo que el dato de este atributo no es muy significativo.

Descripcion = en cuanto a este campo es el dedicado en el modelo a describir la incidencia registrada, es de tipo texto y contiene información de todo tipo como las dedicadas a describir la localización, carácter de la incidencia, o espacio temporal del registro.

FechaFin = en este atributo se expresa el texto dato de la fecha en la que se planifica o prevé que termine la condición de incidencia en la zona señalada, dependiendo de las características esto se puede conocer por la administración de la DGT o no, por esto en varios registros este elemento figura como desconocido ("?"). Este valor puede ser interesante si se conoce, en ese caso será útil para la aplicación web.

Lng = muestra el valor numérico de la distancia existente entre el meridiano de Greenwich, origen, y la ubicación de la incidencia de la DGT, expresada en grados, bajo el sistema de referencia WGS84. Este dato es necesario, ya que es crucial para la representación del registro en el visor web.

PkFinal = punto kilométrico más distante en el que se extiende la incidencia expresada. Este valor se entrega de nuevo automáticamente en el campo **Descripcion**.

Provincia = en este campo se muestra el nombre de la provincia en la que se da el evento registrado, este elemento se reitera en la descripción, para especificar la localización de la incidencia por lo que no es muy interesante por sí solo.

CodEle = valor de tipo texto de caracteres numéricos. Es una variable de control interno propia del sistema de la DGT, útil para identificar y manejar los registros de incidencias captados. Por lo respectivo al proyecto este campo es necesario ya que permite comparar los registros de la base de datos con los ofrecidos por la fuente de datos de la DGT.

Causa = en este campo se comentan los factores por los que surgió la incidencia, sin embargo, como se ve en el ejemplo este suele estar vacío, por lo que actualmente no es interesante para el proyecto.

Carretera = al estar dirigida estos registros en su mayoría a carreteras, la localización de las incidencias se da en vías de transporte o en sus alrededores, por ello se comunica a través de este campo de tipo texto el nombre o código identificativo de la carretera en la cual se da el evento. Sin embargo, esto se repite en el campo **Descripcion**.

Hora = este texto completa el marco temporal del registro indicando el momento exacto en el cual se dio o registró la incidencia, es una información muy ilustrativa ya que permite terminar de ubicar temporalmente la información.

Estado = este campo es propio también de la administración de registros dedicado en la DGT, e indica la situación de la incidencia, no obstante, a lo relativo al proyecto este dato no es interesante.

Autonomía = en cuanto a este dato se encarga en indicar el nombre de la autonomía en la que ocurre el registro, sin embargo, no es un dato importante ya que la zona de interés del proyecto es de un nivel administrativo menor.

PkIni = punto kilométrico más cercano en el que se extiende la incidencia. Este valor se entrega de nuevo automáticamente en el campo **Descripcion**.

Icono = este dato, determina el símbolo por el que se muestra el registro en el mapa de incidencias de la DGT, y por consiguiente también especifica el tipo de incidencia que se da, en el caso del ejemplo es de *suceso*, *otro*, pero de tipo específico *derramamiento*, esta información tan solo es relevante para valorar si la información es interesante, ya que a efectos prácticos el valor útil es el encargado en el campo *suceso*.

Tipo = este campo de manejo interno del sistema de datos de la DGT, indica si el registro es propio de una incidencia o guarda otro tipo de datos, esto no mantiene sentido en el trabajo ya que solo se solicitan datos de incidencias y no de otro tipo.

HoraFin = como se entenderá este dato trata de completar el dato de fecha fin, para terminar de localizar temporalmente el fin de la incidencia, sin embargo, en muchos casos este valor se desconoce, por lo que carece de interés.

Lat = muestra el valor numérico de la distancia existente entre el Ecuador y la ubicación de la incidencia de la DGT, expresada en grados bajo el sistema de referencia WGS84. Este dato es necesario, ya que es crucial para la representación del registro en el visor web.

Nivel = por último este dato de tipo texto indica al mapa de la DGT, mediante colores, la importancia o intensidad que presenta la incidencia, pero esta no es una medida formada en el proyecto, por lo que no tiene lugar en los datos guardados en el proyecto.

5.2.2 Borrado de incidencias obsoletas

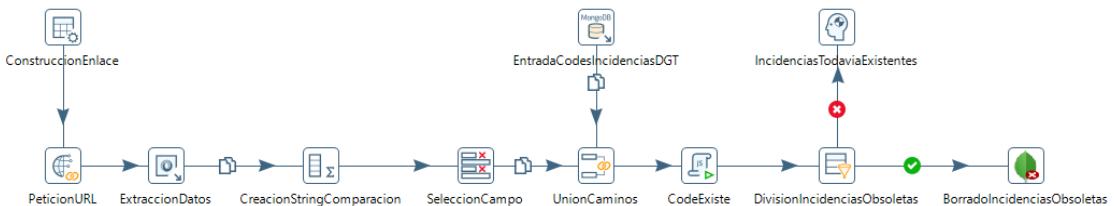


Ilustración 28. Transformación PDI de borrado de incidencias obsoletas

Para el tratamiento y gestión de las incidencias proporcionadas por la Dirección de Trafico, es necesaria una transformación que valore que registros guardados en la base de datos del proyecto siguen siendo proporcionados por la fuente de datos, y por tanto siguen sucediendo en la isla de La Palma. Por otro lado, los registros que hayan sido recibidos por el proyecto, pero no persistan en lo devuelto por la fuente de datos de la DGT no seguirán activos, por eso deberán ser eliminados de la base de datos del proyecto, para que no siga su representación en el visor web del proyecto. Para todo esto, se diseñó la siguiente transformación, con los pasos de procesamiento, que se explicaran a continuación.

ConstrucionEnlace: paso denominado como data grid, permite añadir información de arranque al procesamiento en este caso la respectiva al enlace de petición de datos. Por otro lado, la URL presenta parámetros que permiten seleccionar la información más interesante al proyecto, ya que, a excepción de otras fuentes de datos, esta proporciona datos de todo el estado español y de ámbitos distintos a los enfocados en el proyecto.

```

https://infocar.dgt.es/etraffic/BuscarElementos?latNS=28.926439&longNS=-17.620697&
latSW=28.384151&longSW=-18.102722&zoom=18&accion=getElementos&Camaras=false&
SensoresTrafico=false&SensoresMeteorologico=true&Paneles=false&Radares=false&
IncidenciasRETENCION=true&IncidenciasOBRAS=true&IncidenciasMETEOROLOGICA=true&
IncidenciasPUERTOS=true&IncidenciasOTROS=true&IncidenciasEVENTOS=true&
IncidenciasRESTRICCIONES=true&niveles=true&caracter=acontecimiento
  
```

Como se puede observar en el enlace anterior, los parámetros de la solicitud de `latNS`, `longNs`, `latSW` y `longSW`, forman el bounding box o zona de trabajo de la cual se solicitan datos, en este caso corresponde al encuadre de la isla de La Palma, para así solo cubrir y recibir datos de este terreno. Después el valor del `zoom` es el máximo posible para así obtener todos los registros incluyendo los de precisión más extensa. El campo de `accion` indica el tipo de operación solicitada, en este caso la petición de elementos, en los posteriores parámetros a este se especificara a la API web de la DGT, el ámbito al que deben corresponder los registros que se solicitan, siendo pedidos los propios a `SensoresMeteorologico`, `IncidenciasRETENCION`, `IncidenciasOBRAS`, `IncidenciasMETEOROLOGICA`, `IncidenciasPUERTOS`, `IncidenciasOTROS`, `IncidenciasEVENTOS` e `IncidenciasRESTRICCIONES`, y rechazados los asignados a `Camaras`, `SensoresTrafico`, `Paneles` y `Radares`. Si se observa el modelo de datos de los

registros y la última parte de los parámetros correspondientes a la petición de datos, se puede ver que estos concuerdan con los distintos tipos de valores que pueden ser asignados en el campo suceso, ya que estos son correspondientes a incidencias, ignorando el resto que indican registros de otros campos no interesantes. Por último los dos parámetros de **niveles** y **caracter**, indican que los registros se dividen por niveles y el tipo de acontecimientos.

PeticionURL: posteriormente con este paso de Cliente HTTP, se solicita mediante la URL definida en el paso anterior y según sus parámetros, los datos actuales de registros de incidencias establecidos en la base de datos de la DGT.

ExtraccionDatos: en este JSON input se extraen los valores del objeto JSON recibido, en este caso los datos interesantes para establecer la comparación de registros son los identificadores de estos, los cuales, como se vio en el modelo de datos, vienen expresados en el campo **codEle**.

CreacionStringComparacion: aquí se aprovecha la herramienta Agrupación, que como su nombre indica permite la agrupación de datos, según un valor de campo común. Para facilitar la comparación es óptimo centrar los códigos identificativos en una sola cadena de texto. Para esto se aprovecha el campo *url*, que tiene siempre el mismo valor en todos los registros, se parametriza la herramienta y agrupa los valores de **codEle**.

SeleccionCampo: esta funcionalidad de la tecnología Pentaho Data Integration, es denominada como Selecciona/Renombra valores, y permite la disagregación de campos y su edición de tipo de valor, en este paso en concreto, se elimina la entrada que guarda la URL de petición de datos, ya que no va a ser útil en el resto del proceso, quedándose únicamente como valor de procesado la cadena de texto de comparación.

EntradaCodesIncidenciasDGT: por otro lado, esta segunda entrada de datos aprovecha el proceso MongoDB input para establecer una conexión con la base de datos del proyecto, y así establecer una petición de datos con query, que permita seleccionar de todos los registros con campo Origen igual a DGT, sus valores del campo **Code**, establecidos en la estructura de datos de la colección *registros_incidencias* del proyecto. Así, se pueden sumar a la transformación los valores identificativos, equivalentes a los del campo **codEle** del modelo de entrada, que existen en ese momento en la base de datos del proyecto.

UnionCaminos: finalmente los dos caminos establecidos en el proceso convergen mediante este paso de juntar filas. Se une, el string de comparación que encadena valores de **codEle** de los registros de la base de datos de la DGT, con los valores de **Code**, presentados en la colección *registros_incidencias*. A expresiones de tabla como muestra la tecnología de Pentaho, se

expande, en una columna a parte, el único valor cadena de comparación, sobre la columna de valores de **Code**, proporcionados por la base de datos del trabajo, conseguidos en el otro camino.

CodeExiste: finalmente en este código dado por el paso valor JavaScript modificado, se establece la comparación de registros, a través de una función de búsqueda de texto, que indique si los identificadores recibidos desde el proyecto no existen en la cadena de texto, construida con los identificadores actuales de las incidencias de la colección de datos de la DGT, por tanto, ya no siguen reflejando sus informaciones una situación en la realidad. A niveles del proceso esta función suma a los datos un booleano en formato binario, cero como false o uno como true, 1 si este persiste en la fuente de datos y 0 si el registro ya no figura.

DivisionIncidenciasObsoletas: es un paso de filtrado de filas, que permite desregar los registros procesados según un criterio sobre sus datos. Permitiendo a la transformación, a través del criterio de si la variable booleana es igual a 0, separar los registros, de tal forma que se extraen los registros no actualizados y que deben ser eliminados de la base de datos del proyecto.

IncidenciasTodaviaExistentes: este paso de transformación simulada simplemente recibe los registros que existen en ambas bases de datos, y por consiguiente no hay que eliminar ya que siguen mostrando la realidad de la isla de La Palma.

BorradoIncidenciasObsoletas: para concluir la transformación se da la petición de eliminación de los registros obsoletos en la colección de datos del proyecto, esto es posible gracias a la herramienta MongoDB delete, la cual, cabe destacar, que no suele venir incluida en las versiones estándar de la tecnología Pentaho Data Integration, sino en un plugin denominado comúnmente como pentaho-mongodb-delete-plugin [59]. Volviendo al proceso, este paso crea una petición de borrado de los registros de incidencias del proyecto, cuyos valores del campo **Code**, coincidan con los resultantes de los procesamientos anteriores, que finalmente han sido valorados como obsoletos.

5.2.3 Extracción de incidencias DGT

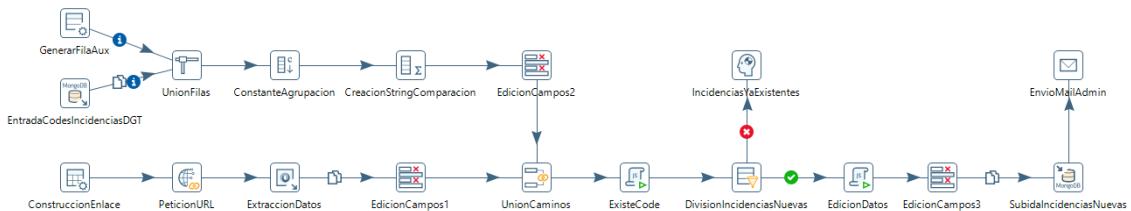


Ilustración 29. Transformación PDI de extracción de incidencias DGT

Otra transformación necesaria en el proyecto para tratar la adquisición y gestión de datos de incidencias con origen de datos de la DGT es la dedicada a la obtención de registros, la cual

permitirá al proyecto añadir registros de incidencias nuevos y actualizados de la zona de la isla de La Palma. Para esto se generaron los siguientes pasos que dan lugar al proceso.

ConstruccionEnlace: en esta tabla de entrada de datos, data grid, se guarda el enlace URL de petición de datos. El cual está formado y parametrizado por los mismos elementos que su homólogo, ubicado en el primer paso de la transformación anterior, borrado de incidencias.

PeticionURL: el cliente web, definido en este paso, procede a establecer la petición de datos expuesta en el principio del proceso, adquiriendo así los registros de incidencias, que se consideraran a añadir al proyecto.

ExtraccionDatos: Aquí se da la extracción de datos del objeto JSON recibido, mediante la función JSON input, se extraen los valores de los campos más interesantes para el proyecto, *alias, suceso, descripción.fechaFin, fecha, hora, lat, lng, codEle*.

EdicionCampos1: después, Selecciona/Renombra valores, permite mantener la limpieza y fluidez del proceso en este caso, sacando del flujo el campo dedicado a el enlace de la petición URL, que ya no es esencial en la transformación.

EntradaCodesIncidenciasDGT: cambiando al camino secundario del proceso, se encuentra esta segunda importación de datos, MongoDB input, la cual, proporciona de la colección mongo incidencias del proyecto, los valores del campo **Code** de los registros cuyo origen se figure como DGT. Estos datos juntos con los **codEle** del modelo de datos del camino principal encaminaran al proceso a identificar que registros de la DGT ya existen en la base de datos y cuales no figuran todavía, por lo que deben ser introducidos.

GeneraFilaAux: por otro lado, este paso de generar filas crea nuevos registros que añade al proceso. Esto se emplea para prevenir la posible situación de que no existan registros de incidencias de la DGT en la base de datos y por tanto no haya capacidad de comparación, que produzca fallar a los siguientes pasos, e interrumpir la transformación. Para ello, mediante esta herramienta, se añade un registro nulo simbólico del campo **Code**, que no interfiera en el proceso de comparación, y permite seguir el flujo si se da la situación antes relatada.

UnionFilas: finalmente, tras acabar la incorporación de datos, este paso, append, permite a la transformación juntar los valores de **Code** añadidos en los pasos anteriores, de tal forma que, al ser ambas entradas de datos de un mismo número de campos, solo uno, se deja la entrada de los valores **Code** del proyecto en la parte superior del flujo, y el auxiliar del otro paso, a la cola del flujo de datos, consiguiendo así la unión de los valores en un único campo **Code**.

ConstanteAgrupacion: ahora el objetivo de los siguientes pasos será la creación de una cadena de texto de comparación, como ya se ha hecho en otras transformaciones del proyecto. Para ello, esta función de añadir valor constante insertara en el flujo de datos un nuevo campo de valor constantes que permita más tarde la agrupación de los valores.

CreacionStringComparacion: agrupar, este paso aprovecha la constante definida antes, para agrupar todos los identificadores de las incidencias DGT del proyecto, más el auxiliar, en un solo texto con el valor \$ de separador. Produciendo una única fila con esta cadena de texto de comparación.

EdicionCampos2: de nuevo en la transformación, selecciona/renombre valores, desengancha del flujo de datos la constante de agrupación definida en el paso **ConstanteAgrupacion**, para solo mantener la cadena de comparación como valor en este camino.

UnionCaminos: este paso es denominado en Pentaho Data Integration como juntar filas, permite la unión de distintos flujos de datos, más específicamente los datos del camino principal de registros de incidencias de la DGT, y la cadena de códigos de identificación, de las incidencias DGT ya guardadas en la base de datos del proyecto. En cuanto al resultado de este paso, es la adición de los registros del camino principal, de un campo nuevo que contiene como valor constante la cadena de comparación creada en el otro lado del proceso, que permitirá la comparación de identificadores de los registros de ambas fuentes de datos.

ExisteCode: el script aquí definido como paso valor JavaScript modificado, entabla la comparación entre identificadores, mediante una función que busca si existe el código de la incidencia de la DGT, en la cadena de identificadores de incidencias ya ubicadas en el proyecto. Posteriormente se guardará el resultado de esta búsqueda en una variable booleana saliente, con formato binario, que figurará como uno, en los registros que existen en ambas fuentes de datos, que por tanto no hace falta añadir en el proyecto, o como cero si no existe en la base de datos del proyecto, y que por tanto debe ser insertada. Cabe destacar, que la opción de que haya una incidencia de origen DGT en el proyecto, que no esté entre las ofrecidas por la fuente de datos DGT, es imposible en un funcionamiento normal de la aplicación, ya que se ejecuta siempre antes la transformación de borrado de incidencias, como se verá en el apartado 5.4.

DivisionIncidenciasNuevas: se repite la estrategia de la anterior transformación y mediante el paso de filtrar filas, se busca en la variable booleana de **ExisteCode**, los valores igual a cero, que indican los registros que deben ser insertados en la aplicación y que por tanto se derivarán para ello, mientras tanto los registros con un valor uno asignado, saldrán del flujo regular, ya que figuran ya en la base de datos del proyecto.

Incidencias Ya Existentes: Aquí se derivan los registros ya existentes, los cuales no interesa su reinserción, este paso de transformación simulada es simbólico para representar correctamente el flujo, pero no implica ninguna transformación en los datos.

EdicionDatos: Para la correcta inserción de los registros, estos se deben adaptar al modelo de datos definido para la colección *registros_incidencias*, para ello se codifico el script definido en este paso. Algunas adaptaciones son el borrado de código HTML en la descripción, la suma del valor de *fechaFin*, si esta existe a la descripción, la conversión de valores entre los campos suceso en el modelo DGT, y tipo en el proyecto, de tal forma que los coincidentes como obras u otros, permanezcan y los diferentes se traduzcan como tipo otros en el modelo del proyecto, o también el criterio de la implementación del campo booleano validada, en el cual, partiendo de la base de que los datos son confiables, el concepto cambia, ya que figuraran como falso siempre que el proceso se dé en un horario laboral, y el personal encargado pueda revisar el contenido de las incidencias, pues este puede seguir en un formato todavía fijo al uso de la DGT y dificultar su entendimiento, además se necesitará añadir la traducción del texto, por otro lado, será verdadero solo si este proceso se da en horario no laboral, para así al menos preservar la actualidad de los datos a los usuarios. Además de esto, se crean nuevos campos que configuraran el envío de mensajes vía mail, para así avisar al administrador sobre la introducción de nuevos datos.

EdicionCampos3: para terminar la modificación de datos en este paso de Seleciona/Renombra valores, se seleccionan únicamente los campos diseñados en el paso *EdicionDatos*, para por una parte su subida a mongo y, por otra parte, él envío de mails.

SubidaIncidenciasNuevas: finalmente, en este paso de MongoDB output se suben los registros detectados como nuevos en la colección de incidencias de la base de datos del proyecto, permitiendo ahora a los usuarios de aplicación web ver las incidencias venidas de la dirección general de tráfico, ya sea por los administradores para editar el contenido, o por los usuarios para poder conocer la realidad de la isla.

EnvioMailAdmin: este paso denominado Mail, permite él envío de información a terceros vía mail, surge como medida para dar facilidad a los posibles encargados de mantenimiento del proyecto, y poder saber instantáneamente que nuevos registros se guardan en la base de datos del proyecto, para dedicarles tiempo de revisión, o simplemente facilitar el control de los datos.

5.3 EXTRACCIÓN DE REGISTROS DE TERREMOTOS

Como se vio en el apartado 4.1.4, los registros de terremotos son proporcionados mediante un archivo java descargable dentro del portal del IGN, y para aprovechar el formato de los datos, con su separación respectiva en distintas variables por lapso de tiempo, se incorpora directamente el archivo al directorio correspondiente al proyecto Node JS de la API del proyecto. Para ello se empleará la transformación mostrada a continuación.



Ilustración 30. Transformación PDI de extracción de terremotos

ConstrucciónEnlace: este paso data grid, se añade como dato del proceso la URL que proporciona el archivo de registros de sismos.

PeticiónURL: aprovechando el enlace proporcionado por el paso anterior, en este elemento denominado cliente HTTP, se establece la petición a la fuente de datos, para añadir así al proceso como un elemento único el contenido del archivo java.

SalidaFicheroJS: finalmente el contenido resultante de la petición, mediante esta salida de fichero de texto, se guarda como archivo de extensión .js en la carpeta *public*, en el directorio del lado servidor del proyecto.

5.4 EJECUCION DE PROCESOS

Ya relatadas las transformaciones Pentaho Data Integration y sus implicaciones en el proyecto, es necesario hablar de la gestión de estas, su coordinación y sus características de ejecución, definidas en los dos Jobs o trabajos PDI (Pentaho Data Integration) explicados a continuación.

5.4.1 Actualización de incidencias y registros de medición

Este trabajo trata la ejecución de las transformaciones dedicadas a los datos del proyecto que requieren actualizaciones regulares, los propios a registros de las estaciones de calidad del aire e incidencias de la DGT. Por esta característica de actualización, el trabajo aquí definido, se ejecuta una vez a la hora, ya que el intervalo de actualización de datos presentada en las estaciones de medición que alimentan la fuente de datos de La Palma Visualizer, tras varias observaciones para valorar su frecuencia, es de media una hora, mientras que la propia de la DGT, solo aporta cambio en los datos, según cambio de estado o incorporación de incidencias, sin una frecuencia regular,

no obstante también requiere una atención sistemática para preservar el objetivo de similitud a la realidad que tiene el proyecto.

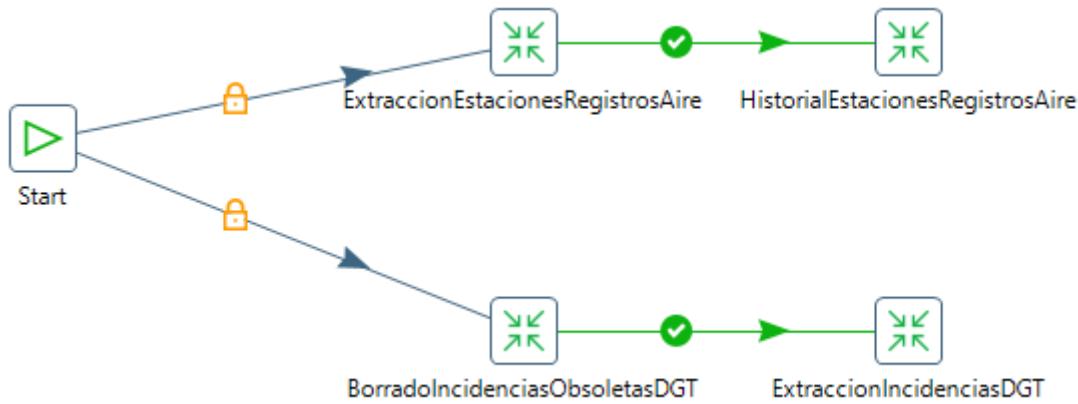


Ilustración 31. Trabajo PDI de actualización de incidencias y registros de medición

Como se ve en la imagen, el trabajo PDI comienza con un elemento de inicio *Start*, en el que se puede parametrizar el intervalo de ejecución de este, y además surge el proceso de ejecución de transformaciones. En este destaca la separación en dos caminos, el superior propio de registros de calidad del aire y el inferior propio de incidencias de la DGT.

Por un lado, en el primer camino dedicado a los datos de calidad del aire, se ve que primero se da el paso de transformación que representa la extracción de registros y luego el paso de transformación del histórico de registros, ya que el histórico se nutre de los datos procesados en la extracción.

Por otro lado, el segundo camino dedicado a las incidencias detectadas por la dirección general de tráfico, tiene primero la ejecución del borrado de incidencias y después la de extracción de nuevas incidencias, pues en esta última transformación se emplean los datos de los registros de la DGT ya existentes en la colección de incidencias, para la comparación con los posibles nuevos, y para no añadir valores sin sentido en esta comparación se decide eliminar antes los registros restantes que no aparecen en la fuente de datos y por lo tanto están obsoletos, mejorando la velocidad de procesamiento de los datos.

5.4.2 Actualización de terremotos

En cuanto este trabajo PDI, gestiona la ejecución de la transformación de descarga de registros de sismos, desde la fuente de datos del Instituto Geográfico Nacional. Como se explicó, estos datos van separados en tres objetos JSON correspondientes a tres lapsos de tiempo distintos, con

unidad mínima común el día, por lo que la actualización de los datos de este archivo y por tanto también de su procesamiento de descarga, es de un día.



Ilustración 32. Trabajo PDI de actualización de terremotos

El Job o trabajo, inicia nuevamente, con un elemento *Start*, que permite programar una frecuencia de ejecución para el proceso. Luego este trabajo solo contiene una transformación para ejecutar, ya que no hay más datos en el proyecto que sigan esta frecuencia de actualización, y es la única dedicada a datos de terremotos.

6. REQUISITOS DEL SOFTWARE

En los siguientes apartados, se repasarán los requisitos de desarrollo de software que se han valorado para la creación de esta primera versión de la app.

6.1 REQUISITOS FUNCIONALES

A continuación, se definirán los requisitos que representan una función concreta en la aplicación, para los cuales se definió primero como idea y después se materializó en mockups como se verá a continuación.

6.1.1 Requisitos de la aplicación

La aplicación web se desarrolló con una serie de objetivos, que se plantearon llevar a cabo mediante la incorporación y coordinación de distintas funcionalidades, apoyadas en las tecnologías presentadas anteriormente en este documento. En este apartado, las siguientes funcionalidades corresponden a las activas en el visor en todo momento, propias al funcionamiento regular de la aplicación, sin ninguna configuración de usuario o registro previo:

- ✓ Visor del mapa web



Ilustración 33. Captura mockup del visor web

Introducción: un componente esencial en la aplicación web, es la visualización de información geográfica, para la cual se recurre a un visor web estilo mapa, en el cual se representan los datos de forma gráfica y sencilla a los usuarios.

Entrada: el visor se muestra en la página principal de la aplicación, por lo que no requiere ninguna entrada de datos o registro previo, permitiendo así en un primer vistazo, informar al usuario sobre los datos manejados en la web.

Proceso: El desarrollo de esta funcionalidad se da únicamente en el lado cliente del proyecto, de tal forma que aprovechando las características de Vue y la dependencia del proyecto Leaflet, se crea un componente denominado *mapa*, en el que se inicia un mapa de Leaflet con las características generales de localización, extensión y zoom indicadas para fijar la atención en la isla de La Palma.

Salida: visor web diseñado para la visualización de datos geográficos de la isla.

- ✓ Control de capas del visor

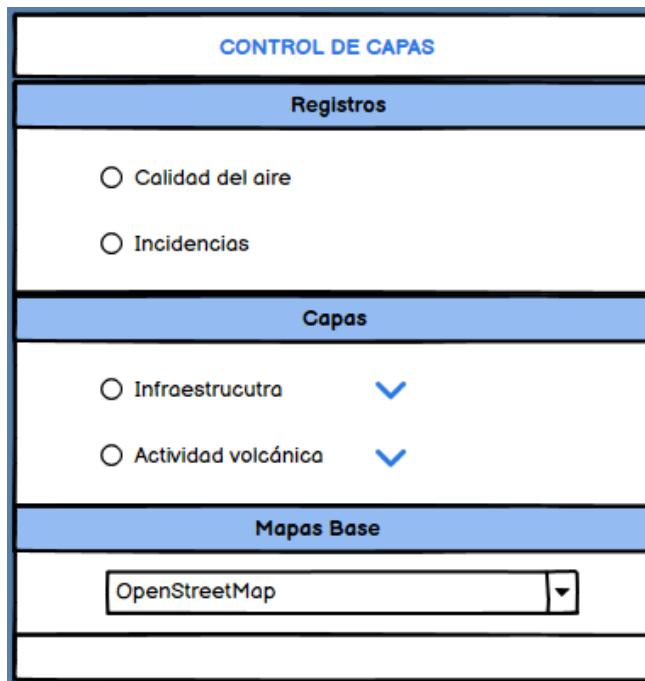


Ilustración 34. Captura mockup del panel de control de capas

Introducción: es necesario dotar a la aplicación de mecánicas de control de capas, y por tanto dar a los usuarios herramientas para el manejo del visor web y los datos que se representen en este, proporcionando así un estilo de trabajo interactivo y dinámico.

Entrada: este órgano de la web se pensó como un menú de control, que permita la selección o formato de la información que el usuario quiere visualizar en el mapa, por ello se recurren a

herramientas como checkboxs, botones de dos estados (marcado o no marcado), o selectores desplegables de varias opciones, que doten al usuario de este control.

Proceso: esta funcionalidad se da únicamente en el cliente web, ya que no requiere de ningún dato o proceso del lado servidor. Por esto, apoyándose en las tecnologías de Vue, Vue-router y Vuex, se crea un nuevo componente llamado *control layers*, donde se darán todos los desarrollos de control de capas, y se comunicara las decisiones del usuario con el componente *mapa*, a través de un almacén de datos Vuex, *map* y su interfaz de uso *usemapstore*.

Salida: finalmente lo que observara el usuario será un panel de control, situado a un lado del mapa, con formato columna con varias secciones, desde el cual pueda manejar la información mostrada en el visor.

- ✓ Selección y visualización de distintos mapas base

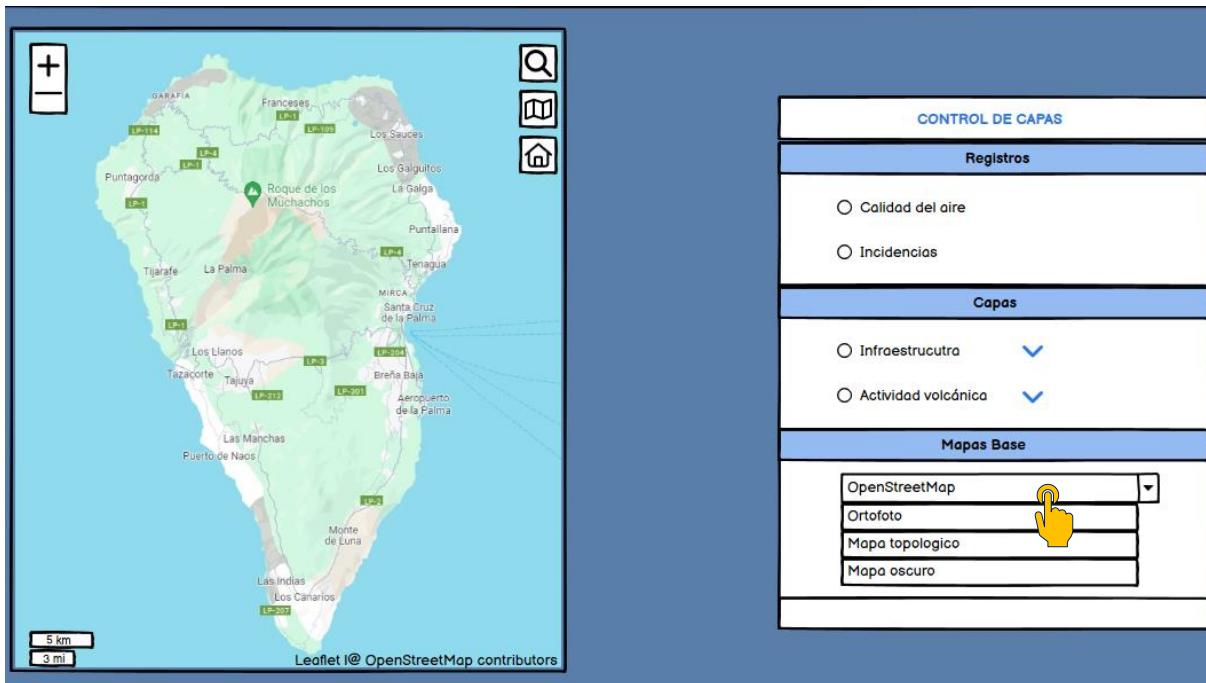


Ilustración 35. Captura mockup interacción de selección de mapa base

Introducción: este proyecto busca dar facilidad de interpretación de los datos al usuario, y un elemento base en brindar esto en la interpretación de datos geográficos son los mapas base, por ello este requisito busca proporcionar una variedad genuina de mapas base que puedan aportar información y facilidad de interpretación al usuario.

Entrada: como aporte por parte del usuario queda la opción de cambiar el mapa mostrado por defecto en el visor, por uno distinto dentro de los disponibles en la web.

Proceso: los desarrollos aquí requeridos pertenecen únicamente al cliente web, pues significan cambios en el visor con datos externos, que no implican al Back-End del proyecto.

Esta funcionalidad se desarrolla en el componente *control layers*, donde aprovechando la interfaz *usemapstore* del almacén *map* se establece comunicación entre el usuario y el visor web por una variable que indica el mapa base seleccionado y un conjunto de otros dos objetos que describen los mapas posibles y sus representaciones en Leaflet.

Además, cabe destacar, que los distintos mapas bases y sus datos vienen dados por conexiones WMS, de servicios externos.

Salida: sección en el panel de control de capas, con un selector que presenta el nombre del mapa base que se muestra en ese momento, donde aparecen las distintas opciones de mapas base que tiene la aplicación y que están disponibles para ser representadas en el visor.

- ✓ Representación geográfica de estaciones de calidad del aire con registros

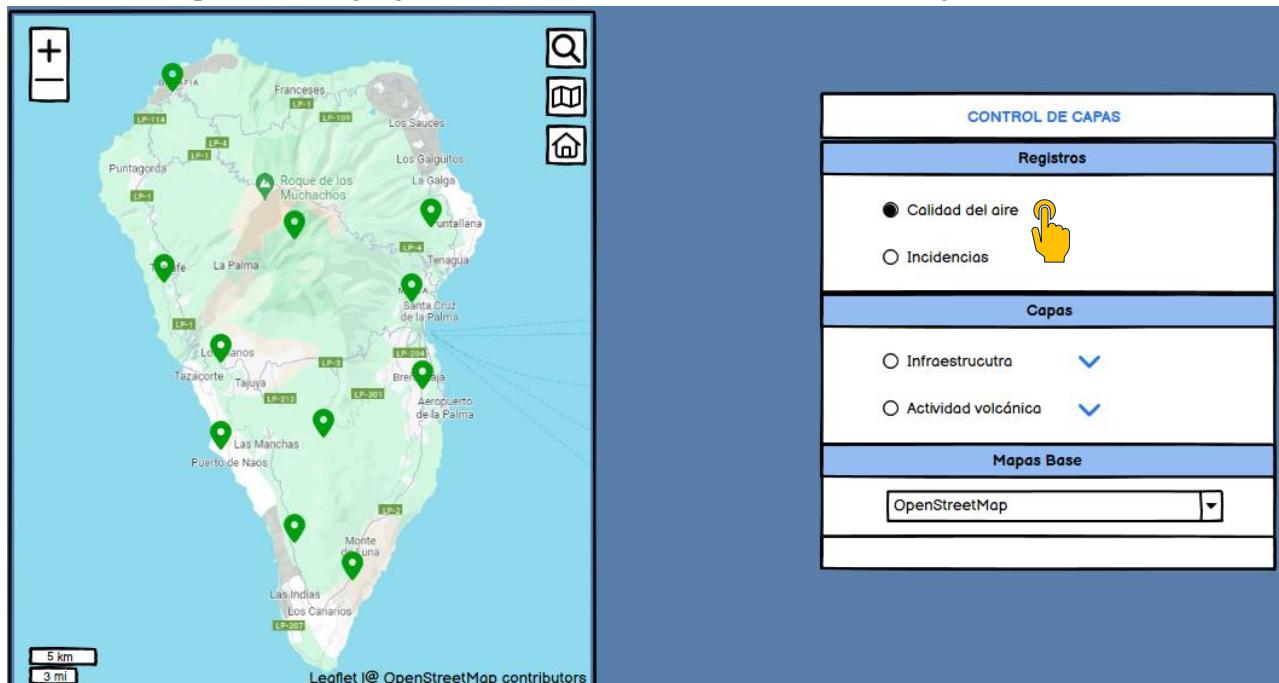


Ilustración 36. Captura mockup interacción de visualización localizaciones estaciones calidad del aire

Introducción: la representación espacial de las estaciones de calidad del aire en el visor del proyecto es muy importante a la hora de expresar la componente geográfica dada por los registros, ya que esta puede orientar el interés del usuario a la hora de elegir de qué registros quiere conocer datos.

Entrada: Como interacción con el usuario queda el control de la visualización o no de las estaciones de medición en el mapa.

Proceso: el desarrollo que se requiere pasa tanto por el lado servidor como el lado cliente del proyecto, ya que implica traslado de datos desde la base de datos Mongo hasta el Front-End.

En primer lugar, la API del proyecto realiza una conexión con la base de datos que permanecerá estable en toda la ejecución de procesos de la API, después se configura un endpoint o punto de conexión que encamine a un proceso de obtención y entrega de valores.

En segundo lugar, por el lado del cliente web, se desarrolla el control de esta funcionalidad en el menú de control de capas, de tal forma que, al recibir la señal del usuario de visualización de la información, se iniciara un proceso en el almacen *map*, de petición de datos al endpoint mencionado, recibiendo así los datos de *_id*, *Latitud* y *Longitud*, que quedaran almacenados en *map*, para ser empleados por el visor en la representación de estaciones de calidad del aire.

Salida: a ojos del usuario se verá en el panel de control de capas una sección destinada a registros con una fila empleada para los de calidad del aire y su checkbox, el cual permitirá controlar la visualización de los iconos de estaciones de calidad del aire en el visor.

- ✓ Visualización de datos registrados de calidad del aire por estación

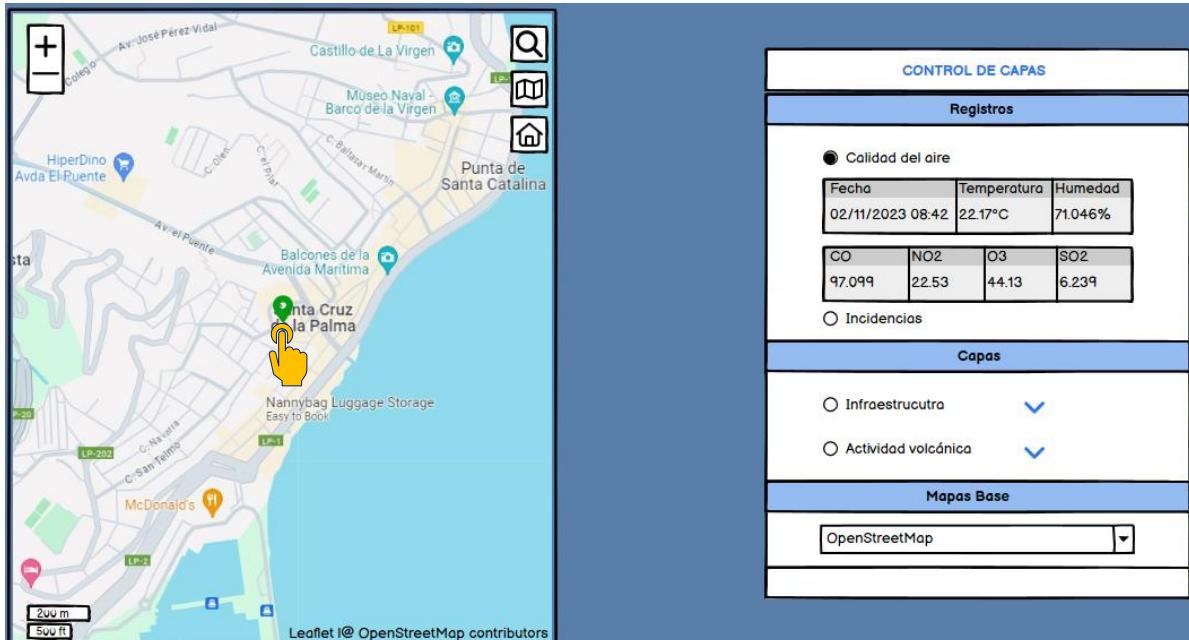


Ilustración 37. Captura mockup interacción de visualización datos de calidad del aire

Introducción: después de dar representación a la componente espacial de los registros, se debe también dar entrada al resto de información que estos presentan, temperatura, medición de gases, etc., por ello se planteó una estrategia de interacción reciproca entre los componentes, donde al igual que controles del panel de capas influían en el visor, se configuraron controles en el visor

que influyen también en la información mostrada por el panel. Extendiendo la información mostrada de cada registro y por tanto la funcionalidad anterior.

Entrada: partiendo de que el usuario pueda ver y por tanto haya activado la capa de registros de calidad del aire, debe elegir entre las localizaciones mostradas y realizar click sobre su icono, produciendo así la aparición del resto de información de este registro.

Proceso: el proceso vuelve a necesitar datos de las colecciones de mongo, por ello participan ambas partes del proyecto. Desde el Back-End, se establece un nuevo endpoint, que atienda a la misma ruta de petición que el creado en el requisito anterior, pero con la espera de recibir un valor extra, el `_id` del registro seleccionado, entonces este punto de conexión desencadenara un proceso del controlador de extracción y envío de datos, en el cual gracias a la conexión establecida con base de datos y el identificador del registro se enviaran a cliente el resto de los valores que presenta el registro.

Por su parte, en cliente se añade un evento de escucha en cada símbolo de símbolo de estación meteorológica dibujado en el visor que atienda al click de estos, y produzca dos procesos, un encuadre a la localización de la estación seleccionada y una petición de datos con el identificador de la estación como parámetro, que atacando al endpoint anterior, permita al cliente obtener la información del registro, y como tiene una estructura fija, también su representación en un nuevo componente Vue `element info`, que se expondrá en el componente de control de capas, que aparte brindara al usuario un control de despliegue que maneje su visualización.

Cabe destacar otras interacciones que se dan con este cuadro de información, como su desaparición si el usuario desactiva la capa o si pulsa sobre un símbolo de registro de otro tipo, y también su contracción si el usuario despliega algún otro subcontrol del panel (selector de mapa base, otras capas, etc.), ya que por cuestiones de diseño solo puede haber un elemento de despliegue activo a la vez.

Salida: El usuario, tras pulsar alguno de los iconos de estaciones de calidad del aire, podrá ver como en el panel de control de capas, justo debajo del apartado de calidad del aire surge una caja con tablas de información sobre la calidad del aire existente en ese punto. Además, también observara en el visor un encuadre dirigido hacia la localización de la estación que ha seleccionado.

✓ Representación geográfica de incidencias

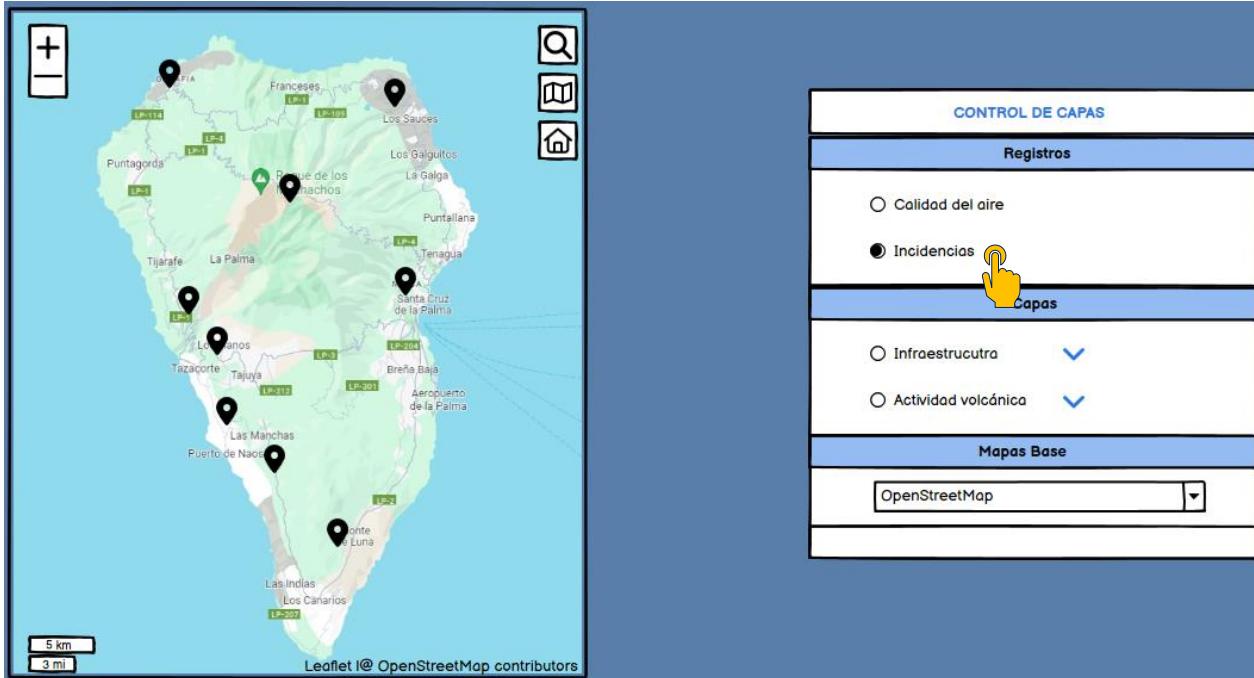


Ilustración 38. Captura mockup interacción de visualización localizaciones de incidencias

Introducción: de igual forma que con las estaciones de calidad del aire, representar en el visor la componente espacial de los registros de incidencias es importante a la hora de informar al usuario la localización de estas, ayudando así a ubicar su atención sobre las incidencias que estén dentro de su zona de interés.

Entrada: Como interacción con el usuario queda el control de la visualización o no de las incidencias en el mapa.

Proceso: El proceso de desarrollo de este requisito es semejante a su homólogo de estaciones de calidad del aire, pero con un par de diferencias relevantes, ya que al ser datos de un modelo distinto y de un tratamiento diferente en el proyecto, el rol de los usuarios pasa a determinar el envío de datos, por lo que se añade un middleware o proceso intermedio existente entre la ruta y el controlador, esto permitirá conocer, si existe en la petición, el rol y nombre del usuario que solicita la información, sin embargo este apartado se da para funcionalidades sin registro previo, por lo que el proceso sigue sin atender a estos parámetros, dando la extracción y envío de los valores de los campos *_id*, *Latitud*, *Longitud* y *Validada* de los registros ya confirmados como veraces, y por tanto que figuren con un true en el campo *Validada*.

En cuanto a los desarrollos en cliente para esta funcionalidad son similares a los establecidos para la Representación de estaciones de calidad del aire, con la diferencia de la elección de simbología,

que pasa a ser una u otra según el valor del campo booleano *Validada* recibido. El cual, en este caso sin registro previo, presenta solo valores true y por tanto todos poseen el mismo ícono.

Salida: al panel de control de capas, en la sección destinada a registros se le añade una nueva fila para incidencias y su checkbox, el cual permitirá controlar la visualización de iconos de localizaciones de incidencias en el visor.

✓ Visualización de datos registrados por incidencia

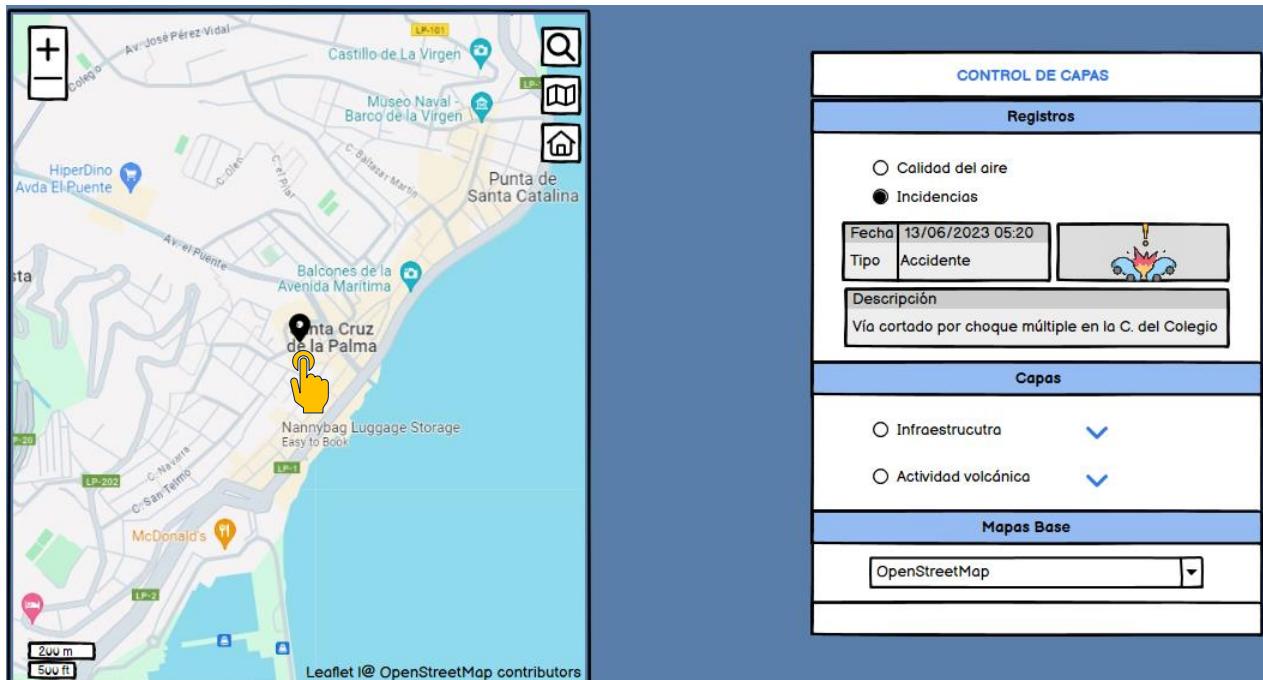


Ilustración 39. Captura mockup interacción de visualización datos de incidencia

Introducción: mostrar el resto de información de los registros de incidencias sigue la misma estrategia de visualización empleada en los registros de calidad del aire, ya que nuevamente al usuario conocer las localizaciones de incidencias puede seleccionar la que le resulte más interesante y así ver el resto de los datos del registro en las tablas emergentes del control de capas.

Entrada: el usuario aquí aporta su interés y selección de al menos alguna de las incidencias mostradas por la funcionalidad anterior.

Proceso: para este requerimiento, se da un proceso similar al expuesto para registros de calidad del aire, ya que a diferencia de las variaciones que presentaba la funcionalidad de representación geográfica de registros con su homóloga de datos de calidad del aire, aquí no se posee ningún middleware o gestión de datos extra por rol, ya que esta primera medida aplica en determinar, en primera instancia, que registros tiene acceso el cliente web y por tanto, solo puede hacer peticiones de datos con identificadores de registros que puede conocer. Por lo demás el desarrollo es igual a excepción de la administración del campo *Imagen*, ya que este no es obligatorio en el modelo de datos, pero si necesario para una correcta visualización en cliente, por esto si los datos del registro

solicitado carecen de imagen, desde el mismo controlador encargado en el Back-End se obtendrá mediante el valor del campo clave *Tipo*, una imagen genérica auxiliar existente en la colección *ímágenes incidencias default*, adecuando así la información del registro a las condiciones de visualización del cliente. Finalmente, estos datos llegarán almacenados a la web en *map*, y se dibujaran mediante el componente *element info* en el panel de control de capas.

Además, mencionar, que los criterios de desaparición y contracción del contenido son los mismos que los expuestos en el punto de visualización de datos registrados de calidad del aire por estación.

Salida: el resultado de esta funcionalidad será la visualización del resto de datos (fecha, tipo, descripción, etcétera) presentados por los registros de incidencias, en un contenedor fijado inmediatamente bajo la fila de incidencias dentro del menú de control de capas.

- ✓ Visualización de información geográfica proporcionada por Geoserver (infraestructura y Actividad volcánica)

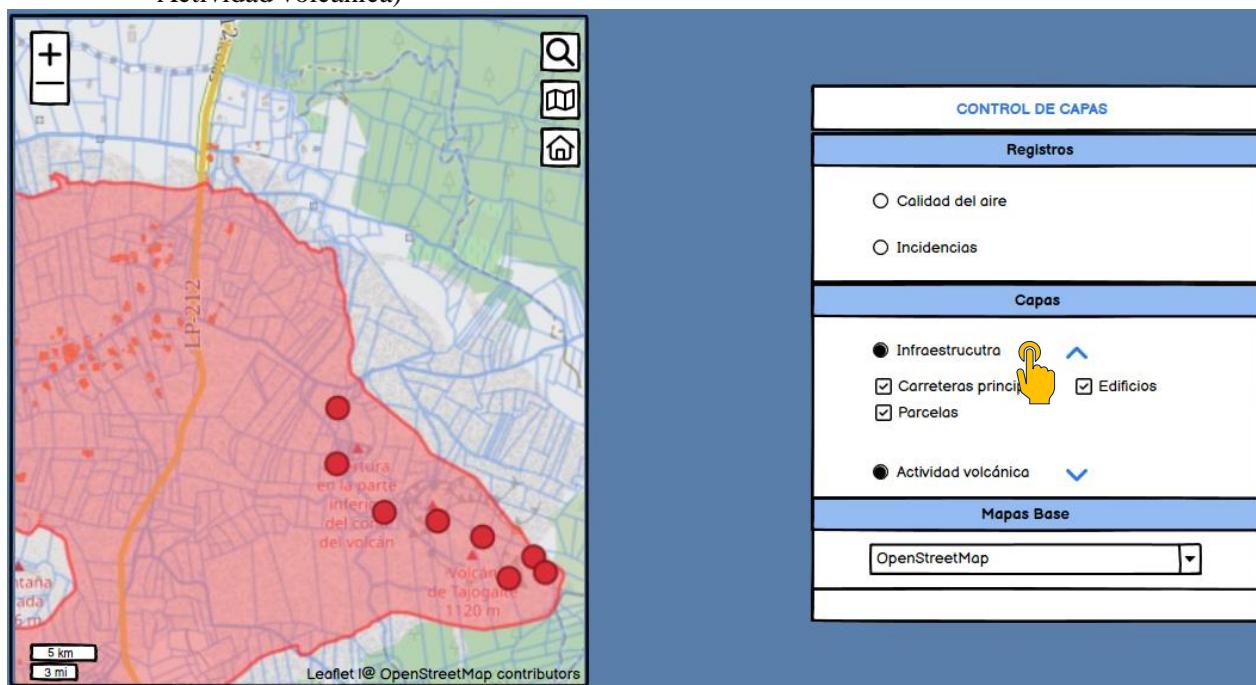


Ilustración 40. Captura mockup interacción de visualización de capas servidas por Geoserver

Introducción: otra parte de los datos de los que consta el proyecto y con componente geográfica son las capas de infraestructura y Actividad volcánica servidas desde Geoserver, útiles para dar a conocer al usuario información de parcelas, edificios, carreteras, bocas eruptivas y coladas volcánicas.

Entrada: como entrada de datos por parte del usuario, es la activación o no de la visualización de las capas de infraestructuras o actividad volcánica, tanto estas como grupo de o de manera individual.

Proceso: el proceso del desarrollo de esta funcionalidad consiste en repetir la estrategia de comunicación entre el componente de control de capas y el del visor, mediante el almacén *map*. Sin embargo, el desarrollo de aquí se caracteriza en la obtención y naturaleza de los datos, ya que, en lugar de realizar una petición de estos a la API del proyecto, se aprovecha el servicio de Web Map Service [60], ofrecido por Geoserver, para así hacer llegar representaciones gráficas de cada capa al mapa Leaflet.

Salida: creación de una nueva sección en el panel de control de capas, dedicado a capas, y con una fila exclusiva a las respectivas de infraestructura y otra a las de actividad volcánica, desde la cual se puede realizar la activación directa de todas las capas en grupo o desplegar un pequeño submenú, donde se puede especificar la visualización de cada capa.

- ✓ Representación geográfica de registros sísmicos según lapso temporal

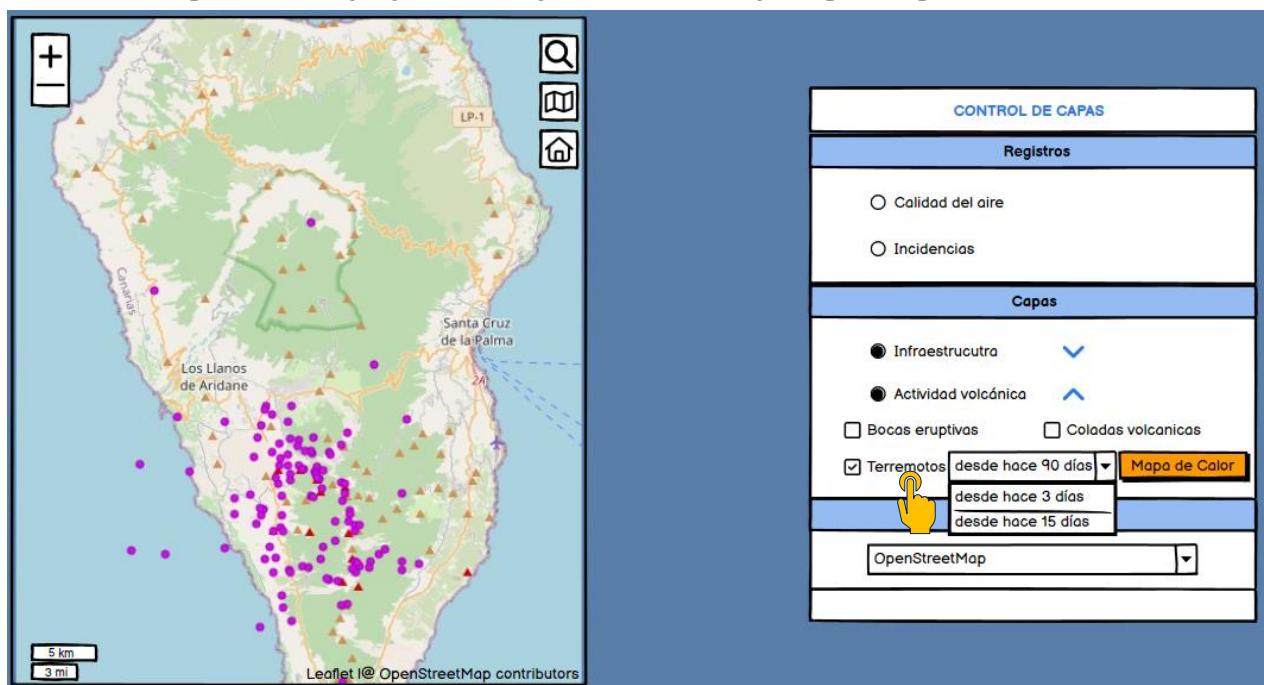


Ilustración 41. Captura mockup interacción de visualización de terremotos por lapso temporal

Introducción: en cuanto a datos a visualizar quedan los respectivos a la actividad sísmica, estos presentan condiciones distintas al resto de capas, como la división temporal, o la distinta disposición a la API, no obstante, de igual manera aportan una información muy útil para el estudio del estado volcánico de la isla.

Entrada: como control del usuario quedan dos elementos, el checkbox que maneja la representación o no de los datos sísmicos sobre el visor web, y el input selector de periodos de tiempo, que parametriza los datos con los que trabaja la aplicación.

Proceso: este desarrollo se da tanto en la API como en el lado cliente. Por un lado, en la API se crea una nueva entrada de datos mediante la lectura del archivo de terremotos guardado en la carpeta *public* del proyecto Node JS. Además se habilita un nuevo endpoint, de nombre de ruta *registrosTerremotos*, para atender a las peticiones de datos del cliente, esta ruta lleva al controlador del mismo nombre, donde se da un proceso de extracción, validación y envío de datos, de tal forma que del archivo se busca primero el objeto JSON asociado al lapso de tiempo solicitado mediante un parámetro de la petición, para después de este extraer los valores de *latitud* *longitud* y *magnitud* de todos los registros del objeto, que además estén ubicados dentro del territorio de la isla de La Palma, creando con ellos un array de datos listo para ser utilizado por el cliente.

Por otro lado, desde el Front-End, se maneja la conexión de ambos componentes, control de capas y visor web, como siempre gracias al almacén *map*, desde el cual se da la petición y guardado de los datos sísmicos cuando el usuario lo indique en el panel de control y así provocar su posterior representación en el visor web.

Salida: con este desarrollo se completa la obtención de datos de la web, mediante la suma de un checkbox al control de capas, desde el cual se maneja la visualización de los datos sísmicos en el submenú de actividad volcánica, añadiéndose a éste, además, un selector para la parametrización de datos por periodo de tiempo.

- ✓ Representación de terremotos por mapa de calor

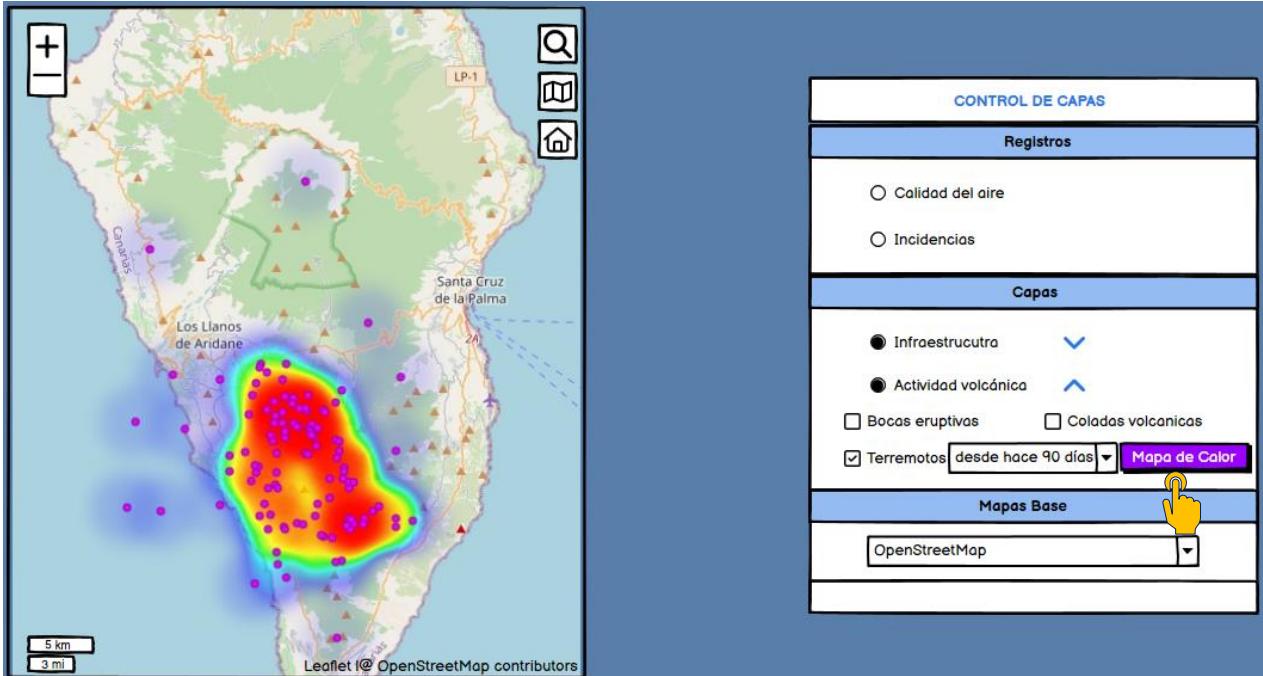


Ilustración 42. Captura mockup interacción de visualización de mapa de calor de terremotos por lapso temporal

Introducción: esta funcionalidad extiende la capacidad de comunicación del apartado anterior, mediante la concepción de mapas de calor a partir de los datos de registros de terremotos. Estos mapas permitirán ilustrar de forma sencilla las zonas con menor y mayor actividad sísmica.

Entrada: criterio de activación o no de la funcionalidad sobre los registros sísmicos mostrados en el visor, mediante un botón de dos estados que controla la visualización de estos mapas.

Proceso: el control de esta visualización sigue la estrategia de interacción entre componentes, en este caso mediante un nuevo botón dispuesto en el panel de control de capas, se maneja la generación de mapas de calor en el componente *mapa*. Para esto se aprovechan los datos de registros de terremotos, localización y magnitud y la extensión Leaflet-heat, que crea objetos de mapa que representen mapas de calor. Además, a nivel de gestión, la visualización del mapa de calor esta enlazada con la visualización de las localizaciones de sismos, ya que, como se vio en el requerimiento anterior es necesaria esta segunda, a la hora de que el cliente web solicite y obtenga la información de sismos requerida también para esta funcionalidad.

Salida: como resultado de este desarrollo se añade un botón de dos estados justo al lado derecho de la entrada de visualización de terremotos, indicando así al usuario la relación de dependencia existente entre este y el anterior requisito.

✓ Herramienta de geocodificación directa



Ilustración 43. Captura mockup funcionamiento herramienta de geocodificación

Introducción: la geocodificación es importante en el proyecto ya que facilita mucho a los usuarios la localización de lugares o ubicaciones de su interés, con tan solo sus nombres populares.

Entrada: inserción del nombre del lugar buscado desde la herramienta de geocodificación implementada en el visor.

Proceso: el desarrollo de esta funcionalidad consiste en la incorporación de la tecnología Leaflet-control-geocoder, al mapa Leaflet, para esto se inicializó el objeto *Geocoder* proporcionado por la extensión, y se parametrizo adecuadamente destacando la elección como servicio de geocodificación de Nominatim. A parte se modificó directamente en la librería los valores de búsqueda en este servicio para solo captar resultados que estén dentro del territorio de La Palma.

Salida: Como resultado de esto el usuario puede ver una herramienta con símbolo de lupa situada en la esquina superior derecha del visor, la cual al pulsar muestra un input para la inserción del nombre del lugar a buscar, después a medida que se encuentran resultados, estos van apareciendo debajo del input.

✓ Herramienta de geocodificación inversa



Ilustración 44. Captura mockup funcionamiento herramienta de geocodificación inversa

Introducción: este requisito se vio necesario para facilitar los valores de coordenadas a los usuarios, sobre todo para una versión anterior de la aplicación en la cual era útil para la creación de registros de incidencias. Sin embargo, la aparición de nuevas funcionalidades, cambio el sentido de esta, a un papel de puro conocimiento de las coordenadas, que pueda ser útil al usuario en futuras operaciones en esta o en otras aplicaciones.

Entrada: los usuarios pueden acceder a esta función mediante el clic derecho del ratón sobre la localización de interés.

Proceso: a través de las opciones de mapa proporcionadas por Leaflet, se añade al visor la aparición de un evento de tipo menú contextual, que muestre las coordenadas del punto de click del ratón, y además se habilita en este texto un evento click, que copie en el portapapeles del usuario las coordenadas mostradas.

Salida: finalmente como salida se da la aparición de un pop up con la característica de mostrar un cuadro con las coordenadas preseleccionadas, el cual hacer click ejecuta la copia de esta en el portapapeles.

✓ Leyenda de símbolos dinámica



Ilustración 45. Captura mockup de funcionamiento de la leyenda del mapa

Introducción: la gran cantidad de símbolos y estilos de capas de los que consta el proyecto requiere añadir una leyenda para proporcionar ayuda al usuario a la hora de relacionar la simbología con los datos representados.

Entrada: para poder acceder a esta funcionalidad el usuario tan solo tiene que hacer click sobre el ícono de la herramienta de leyenda.

Proceso: como proceso lo primero es obtener los distintos símbolos y estilos de capas en un formato adecuado para la leyenda, para esto se obtuvo versiones más pequeñas de la simbología puntual propia de registros, mientras que para las distintas capas servidas por Geoserver, o para la representación de los datos sísmicos, se empleó la petición GetLegendGraphic, para obtener una representación en formato leyenda de los estilos utilizados en estas capas. Después de este proceso, se construyó el cuadro de la leyenda y se enlazo su aparición a un botón de dos estados posicionado en un lado del visor.

Cabe destacar, que al tener los usuarios distinto accesos a datos en dependencia del rol de sus cuentas, o si están registrados o no, la leyenda presenta un contenido dinámico, de tal manera que muestra la simbología de las capas a las que el usuario tiene acceso en cada momento.

Salida: El resultado de estos desarrollos, es por un lado la aparición de un botón en al lado superior derecho del visor, destinado para herramientas, desde el cual se puede manejar la aparición de la leyenda que se representa justo a la izquierda de la barra de herramientas.

✓ Herramientas de control de zoom del mapa

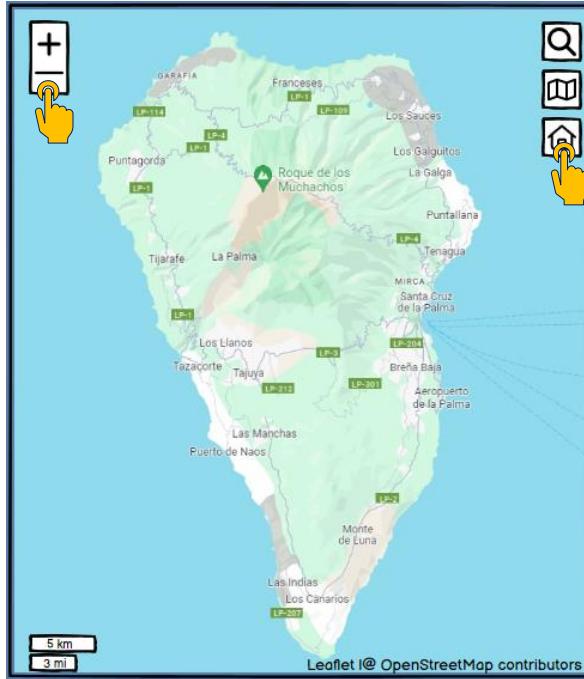


Ilustración 46. Captura mockup de herramientas de control de zoom

Introducción: el control de zoom del mapa es importante en cualquier visor web, por ello desde el proyecto se diseñaron estrategias de control externas a las añadidas por defecto en el mapa Leaflet, botones de zoom in y zoom out, y un botón de vista general,

Entrada: El usuario puede interactuar con estos controles, haciendo click en los botones establecidos en el visor.

Proceso: desde el código del visor en el componente *mapa*, se añade el objeto control scale al mapa Leaflet, añadiendo así los botones de zoom out y zoom in al visor. Por otro lado, para desarrollar el botón de vista general, se crea un evento que reaccione fijando la vista del mapa a la inicial, volviendo a la vista general de la isla de La Palma.

Salida: El usuario ve tres botones más en el visor del mapa, dos en el lado superior izquierdo, correspondientes a las funciones de zoom in y zoom out, y otro en el puesto inferior de la barra de herramientas del visor, dedicado a la función de vista general.

- ✓ Traducción total al inglés

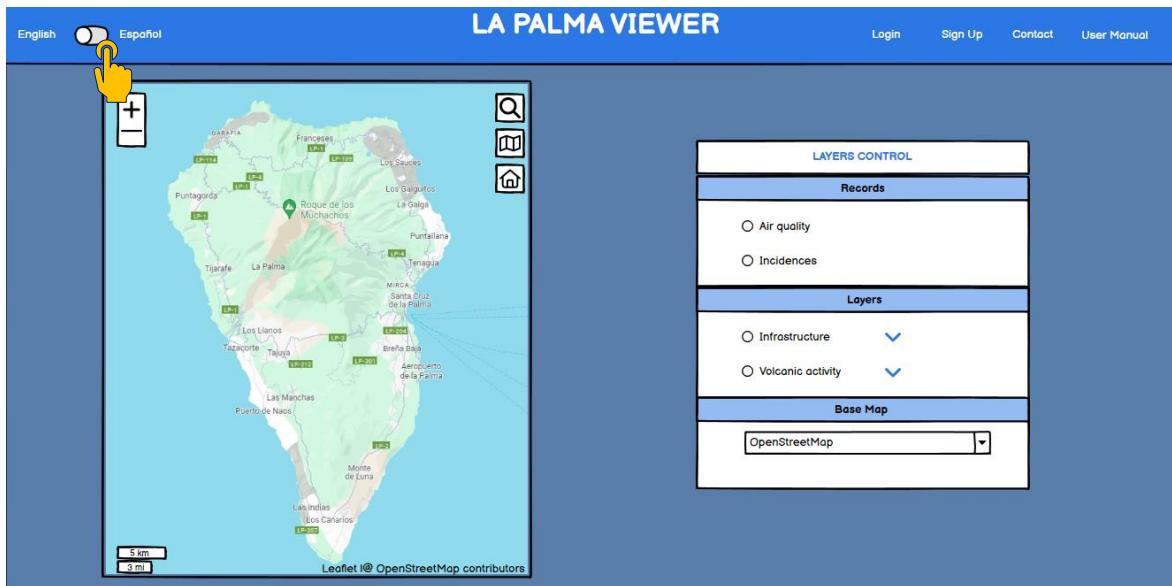


Ilustración 47. Captura mockup cambio de idioma y traducción de contenido al inglés

Introducción: para aportar accesibilidad al portal a otros usuarios que no tengan el castellano como primera lengua, se optó por incorporar a la aplicación la capacidad de mostrar la información también en inglés y permitir este control de idioma como herramienta del portal.

Entrada: el usuario interactúa con esta función por medio del control de un botón de dos estados, siendo uno establecido por defecto para el idioma español y otro para el inglés.

Proceso: este requerimiento es posible gracias a la implementación de la tecnología de Vue-i18n y las características de los modelos de datos del proyecto, con campos específicos para traducciones, así se dan estrategias de internalización de la información al proyecto, manejadas todas desde un control de dos estados configurado en la componente *header* del cliente y un almacén de datos de nombre *i18n*, el cual permite conocer el idioma seleccionado por el usuario a todos los componentes de la aplicación.

Salida: desde la perspectiva del usuario, podrá ver en el lado izquierdo de la cabecera de la aplicación, un botón de dos estados, igual en cualquier página y ruta, desde el cual se puede modificar los textos de toda la web.

- ✓ Formulario de contacto con la organización

The mockup shows a contact form titled "Contacto". It contains five input fields with asterisks indicating they are required: "Nombre*", "Apellido*", "Mail*", "Asunto*", and "Mensaje*". Below these fields is a green rectangular button labeled "Enviar".

Ilustración 48. Captura mockup del formulario de contacto

Introducción: en continuidad de dar un paso más en la interacción con el usuario, se creó una funcionalidad que permitiese el contacto de este con la administración del proyecto, para así atender a posibles propuestas, peticiones, solución de fallos o brindar información del proyecto que pueda resultar útil.

Entrada: como inserción de datos por parte del usuario, está el formulario de contacto, con entradas para los datos de nombre, apellido, mail, asunto y mensaje. Estos valores deben ser aportados obligatoriamente por el usuario para posibilitar el envío del contenido a la administración.

Proceso: En primer lugar, se establece un modelo virtual de datos según el criterio de obtener los esenciales para dar una comunicación fluida y sincera entre el usuario y la administración. Estos serán solicitados, mediante un formulario ubicado en una ruta exclusiva para crear este contacto. Luego tras el usuario llenar el formulario y ejecutar su envío, se validan el formato de los datos desde cliente y mediante un nuevo almacén de datos Vue llamado *authentication* se genera una petición *post*, con toda la información aportada, hacia un endpoint de la API del proyecto dedicado a este proceso, desde allí se extrae el contenido del mensaje y gracias a la extensión Nodemailer y el sistema de roles, se envía un correo electrónico con los datos introducidos al mail asociado a la cuenta de usuario con rol super usuario, propio del personal de la administración.

Salida: el usuario puede ver como en el encabezado resulta un enlace a la ventana de contacto del portal, desde donde puede llenar el formulario de contacto, con el mensaje que desea hacer llegar a la administración de la web.

✓ Lectura de manual de uso



Ilustración 49. Captura mockup acceso a manual de uso

Introducción: a medida que se van incorporando funcionalidades al proyecto gana más sentido la existencia de este requisito, ya que proporciona al usuario un documento ilustrativo de todas las opciones que presenta el portal web, para que así los usuarios puedan utilizarlas a su gusto.

Entrada: para llegar a esta funcionalidad el usuario debe pulsar sobre un link que le lleve a la documentación del portal.

Proceso: Lo primero es organizar las explicaciones y exposiciones de todas las funcionalidades del portal en un archivo. Después este archivo se sitúa en una ubicación a disposición del lado servidor, en este caso la carpeta *assets*, y desde allí con el objeto *path* de Node JS, el controlador encargado expone el archivo de documentación a medida que recibe peticiones desde el punto de acceso correspondiente. En este caso las peticiones son simples y no requieren de parámetros específicos ya que la API sirve directamente el documento, por lo que desde cliente se hace llegar al usuario este archivo, únicamente mediante un enlace expuesto por el servidor.

Salida: como resultado del desarrollo el usuario ve en el menú de navegación de las páginas del portal un enlace a la documentación de ayuda.

✓ Alta de usuarios

The mockup shows a registration form with the following fields:

- Nombre* (Name) - Text input field
- Apellido* (Last Name) - Text input field
- Municipio (Municipality) - Text input field
- Usuario* (Username) - Text input field
- Mail* (Email) - Text input field
- Contraseña* (Password) - Text input field

At the bottom is a green button labeled "Registro" (Register).

Ilustración 50. Captura mockup de formulario de registro de usuario

Introducción: un paso central dentro de la creación del sistema de usuarios es la obtención de la información de los usuarios registrados, para esto se requiere una funcionalidad que permite la toma inicial de datos del usuario, para así poder crear su perfil .

Entrada: como entrada de datos el usuario debe aportar a la aplicación obligatoriamente su nombre, apellido, nombre de usuario, mail y contraseña. Además, se podrá aportar de forma opcional el dato de municipio, que representa donde el usuario enfoca su interés dentro de la isla de La Palma.

Proceso: Este desarrollo se da en ambas partes del proyecto tanto en el lado servidor de la aplicación como en el cliente. Por un lado, desde la API, se abre un endpoint *signup* que atienda a peticiones *post*, desde este punto de entrada se da a un controlador, que valida los datos recibidos por el cliente, comprobando que el mail sigue el formato debido y no se repite en alguna otra cuenta de usuario, y que el nombre de usuario elegido es único, después si salen correctas se encripta la contraseña del usuario mediante la extensión Bcrypt, se guardan los datos en la colección usuarios y se le asigna automáticamente el rol de usuario general, terminando así la creación de la cuenta en la aplicación. Además, después de esto se inicia sesión con la cuenta de usuario recién creada, para lo que a través de Jsonwebtoken, se genera un token de acceso con el nombre del usuario y el rol, que se guarda en la colección de *tokens* y se envía al cliente.

Cabe destacar que cualquier fallo en la validación o la ejecución del proceso de subida de datos es reportado al cliente para mantener informado al usuario del procedimiento de registro.

Por otro lado, desde el cliente se crea una nueva vista propia a una nueva página de la web, la cual se conecta con el resto desde un enlace del componente cabecera, desde aquí se codifica el formulario de registro, así como un cuadro de alerta que permite conocer al usuario de cualquier error en los datos o fallo en el proceso, después tras ejecutar el envío de datos en el formulario se realiza una conexión con el almacén *authentication*, encargado de los procesos de configuraciones de cuentas de usuario e inicia un proceso de validación propia sobre los datos de entrada. Después se genera una petición *post* hacia el servidor, del cual se espera o bien una respuesta de error con su código, para identificar y comunicar al usuario, o bien una respuesta de confirmación con el token de acceso, que indica la correcta creación de la cuenta. Luego se inicia sesión de la cuenta recién creada y se redirige a la página principal con las funcionalidades correspondientes ya activas.

Salida: a partir de esto el usuario puede tener acceso a través de un enlace de la cabecera del portal, a una nueva página de registro de usuarios que contará con el formulario de registro, y encima de este un cuadro de información de errores, que aparecerá únicamente al darse algún fallo en el proceso de registro.

6.1.2 requisitos según rol de usuario

A continuación, se repasarán los requisitos definidos por funcionalidades empleadas para el sistema de roles o disponibles a solo usuarios registrados, según su rol.

- ✓ Inicio de sesión

The mockup shows a login form with a dark blue header containing the word 'ACESSO' in white. Below the header is a light gray background area. On the left, there are two input fields: one labeled 'Usuario' and another labeled 'Contraseña', both with placeholder text boxes. At the bottom is a green rectangular button with the word 'acceso' in white.

Ilustración 51. Captura mockup del formulario de acceso para usuarios

Introducción: a parte de la creación de usuarios, una funcionalidad básica del sistema de cuentas es el proceso de iniciar sesión, por el cual la aplicación da acceso a los usuarios a las distintas funcionalidades reservadas.

Entrada: los datos necesarios para ejecutar el proceso de inicio de sesión son el nombre de usuario y la contraseña de este.

Proceso: desde el cliente se creó una nueva vista encargada de representar la página de login del portal, donde se recogen las credenciales de acceso del usuario y se guardan en el almacén de datos *authentication*, el cual tras comprobar que los datos requeridos se han aportado, envía una petición *post* con las credenciales al punto de acceso de la API correspondiente, quedando a espera de una respuesta. Por otro lado, ya con la cuenta de usuario creada y los datos del usuario en base de datos del proyecto, en el lado servidor, se abre una ruta de acceso llamada *login* que recibe la anterior petición, y lleva los datos al controlador *login*, desde el cual se empieza el proceso de comprobación de credenciales, que si resulta correcto genera la creación, registro en base de datos y envío al cliente web, del token de acceso. Sin embargo, si esta comprobación de credenciales falla o se da algún error en el proceso, se envía el código del fallo al cliente para ser reportado al usuario.

Salida: el usuario ve desde el portal un enlace correspondiente a la página de acceso situado en la cabecera desde la cual podrá insertar sus credenciales de usuario en un pequeño formulario y así poder iniciar sesión en la aplicación. Si este proceso falla se le muestra un cuadro de aviso con el tipo de error.

- ✓ Visualización del perfil del usuario

The mockup shows a user profile page with the title 'PERFIL' at the top. Below it, there are four input fields: 'Nombre:' with the value 'Andrés', 'Apellido:' with the value 'Chasiluisa', 'Municipio:' with the value 'El Paso', and 'Mail:' with the value 'as.chasiluisa@alum'. At the bottom of the form is a green 'Editar' button.

Ilustración 52. Captura mockup de cuadro de perfil de usuario

Introducción: este requisito permite a los usuarios la visualización de los datos de registro aportados por ellos, para así mantenerlos informados de que datos suyos utiliza y tiene guardados el proyecto.

Entrada: aquí no hay necesidad de un aporte de datos, tan solo la decisión del usuario de cambiar de página por el acceso establecido para observar el cuadro de perfil de usuario.

Proceso: la representación de los datos de la cuenta al usuario consta de desarrollos en los dos lados del proyecto, tanto cliente como servidor. Para ello, se genera desde el Back-End una ruta de acceso *profile*, que pueda atender peticiones *get* de solicitud de datos de la cuenta de usuario, además como esta operación es exclusiva para usuarios registrados con token en activo, se añade un middleware, de paso intermedio, que se ocupa de verificar la validez del token y comprobar que existe en la lista de tokens activos registrados en la base de datos. Luego los datos guardados en el token de acceso, nombre de usuario y rol, se llevan al controlador asignado, donde a partir del nombre del usuario, empieza un proceso de extracción y envío de los datos de la cuenta de usuario al cliente web.

Por su lado desde el cliente se abre una nueva vista correspondiente a la página de perfil, con acceso en la cabecera de la aplicación, esta página nada más se empiece a ejecutar se comunica con el almacén *authentication* para realizar la petición de los datos del perfil de usuario, y poder mostrarlos en un menú informativo como placeholders de un formulario desactivado.

Salida: el usuario podrá acceder ya una vez registrado, a una nueva página de gestión de perfil de usuario, nuevamente mediante el acceso de un enlace situado en el menú de navegación de la cabecera. En esta página podrá ver un cuadro en formato formulario con los datos de su cuenta, y justo debajo un botón de edición.

- ✓ Modificación de datos del perfil del usuario

PERFIL

Nombre:	Andrés
Apellido:	Chasiliusa
Municipio:	El Paso
Mail:	ascha

Enviar cambios

Volver

Ilustración 53. Captura mockup formulario de actualización de perfil de usuario

Introducción: tras la visualización de los datos del perfil de usuario, el siguiente paso es editar los datos de su cuenta, ya que después del registro puede ser de utilidad a los usuarios la capacidad de actualizar datos por razones externas como la pérdida de un mail, el cambio de municipio, etc.

Entrada: los datos que pueden ser requeridos para esta funcionalidad son los valores actualizados de los campos nombre, apellido, municipio y mail, que quiera cambiar el usuario.

Proceso: esta funcionalidad se ofrece en la página de perfil de usuario creada para el requisito anterior, permitiendo así ya no solo la visualización de los datos de la cuenta de usuario sino también la actualización de estos. Para ello, se genera desde el Back-End un nuevo acceso en la ruta *profile*, que pueda atender peticiones *put* de actualización de datos, y que de nuevo cuente con el middleware de revisión de token de acceso, para asegurar así las credenciales del usuario. Después de la recepción y revisión de la petición *put* se trasladan los datos recibidos al controlador de actualización asignado, iniciando este un proceso de validación de datos para después si este es positivo realizar una operación *update* sobre el registro de la cuenta de usuario correspondiente. Después se comunica la culminación del proceso al cliente web.

Por su lado desde el cliente, en la página de perfil, se comunica con la funcionalidad mediante el botón de edición presente debajo de los datos del usuario, este permite la activación del formulario de datos, con su respectivo botón de enviar, que desencadena, en el almacen de datos Vuex *authentication*, la creación de una petición *put* de actualización con los nuevos datos implementados por el usuario para su perfil, esta se envía al servidor del proyecto, quedando a la espera del correspondiente mensaje de confirmación o de fallo, que se trasmisirá al usuario mediante la visualización de los nuevos datos recién introducidos o un mensaje de error.

Salida: el usuario al pulsar el botón de edición ve el formulario habilitando, listo para inserción de nuevos datos. Además, en este estado desde los dos botones situados debajo del formulario se puede finalizar el estado de edición con el botón volver, o realizar la actualización de datos con el botón de envío de cambios. Posteriormente de la ejecución de cualquiera de estos dos procesos, se regresa al estado de solo visualización de datos, donde el usuario si ha realizado cambios en el perfil, vera los nuevos datos recién introducidos indicando por tanto la actualización de estos en la base de datos del proyecto.

- ✓ Herramienta grafica de medición de distancias

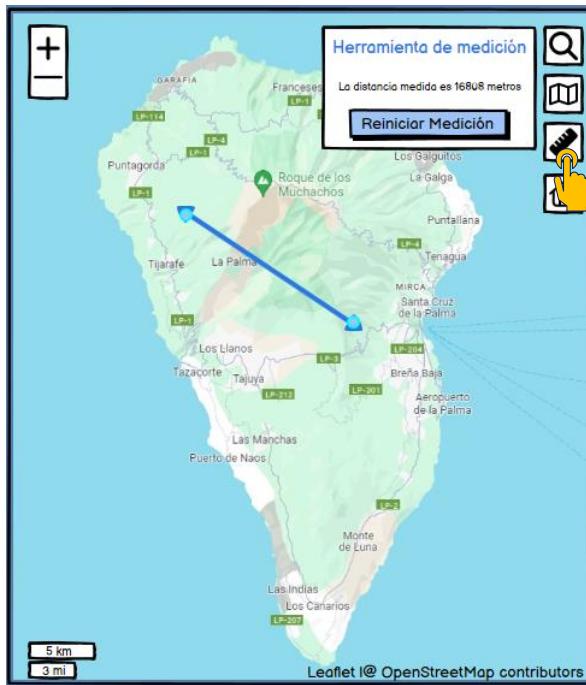


Ilustración 54. Captura mockup funcionamiento herramienta de medición

Introducción: la capacidad de medir distancias dentro del visor resulta muy útil para los usuarios, ya que permite valorar mejor las creaciones de alarmas o la distancia entre puntos de interés del usuario con registros u otros datos.

Entrada: el usuario por su parte debe aportar dos localizaciones de la isla, de las cuales quiera conocer su distancia entre sí.

Proceso: el desarrollo de esta funcionalidad se da en el componente *mapa* del visor. Primero se crea un evento de mapa que se enlaza a un botón de la barra de herramientas del visor que inicie un proceso de dibujo, de tal forma que al realizar click izquierdo en el mapa se dibuje un punto en la localización seleccionada. Luego tras el usuario haber dibujado dos puntos, sus coordenadas pasan a definir una polilínea, que simbolice la distancia entre ambas localizaciones en el mapa. Por otro lado, tras tener las coordenadas de los dos puntos elegidos por el usuario, se calcula la distancia entre estos mediante la función *distanceTo()* y se comunica este valor al usuario por el menú de la herramienta, en el cual también se programa un botón de reseteo que elimina las geometrías añadidas y reinicia el evento de dibujo de puntos.

Salida: en la web el usuario podrá ver una herramienta con símbolo de regla en un lado del visor, la cual al hacer click despliega a un menú con indicaciones para realizar la medición. Siguiendo

las indicaciones, se dibujan las geometrías de los puntos y la polilínea, para posteriormente mostrar en el menú de la herramienta el valor de la distancia medida y un botón de reinicio.

- ✓ Menú de creación de incidencias

The form is titled "NUEVA INCIDENCIA". It features five input fields with labels: "Nombre" (Name), "Tipo de incidencia" (Incident Type), "Coordenadas" (Coordinates), "Imagen" (Image), and "Descripción" (Description). A green "Enviar Incidencia" (Send Incident) button is located at the bottom.

Ilustración 55, Captura mockup formulario de creación de nuevas incidencias

Introducción: esta funcionalidad permite al proyecto recoger los datos de nuevas incidencias aportados por el usuario y así su integración a los registros de incidencias procesados por la aplicación.

Entrada: como entrada de datos serán necesarios obligatoriamente los valores de nombre, tipo de incidencia y coordenadas, además como opcional se obtendrán los valores de imagen y descripción.

Proceso: desde el lado cliente se creó una nueva vista encargada de representar la página de creación de nuevas incidencias, la cual se conectará, a la página principal, mediante un botón situado en el componente de control de capas, visible solo a los usuarios registrados. Por otro lado, sobre esta página se codifica el formulario de entrada de datos que se enviarán a la API del proyecto mediante la petición *post* construida por el almacén *map*. Por su lado el servidor tras recibir la petición inicia un proceso de validación de datos, el cual, en caso de ser exitoso, permitirá la subida de esta nueva incidencia a la base de datos del proyecto. Como en casos anteriores tanto por posibles fallos como por la confirmación de éxito en la operación, llegará un mensaje al cliente, que se tratará y se hará llegar al usuario.

Salida: como resultado de este desarrollo los usuarios verán un botón con signo más (+) en la sección de incidencias del panel de control desde el cual pueden acceder a la página de creación de incidencias, donde observaran un formulario de creación de incidencias en el cual pueden aportar la información de una nueva incidencia que quieran registrar.

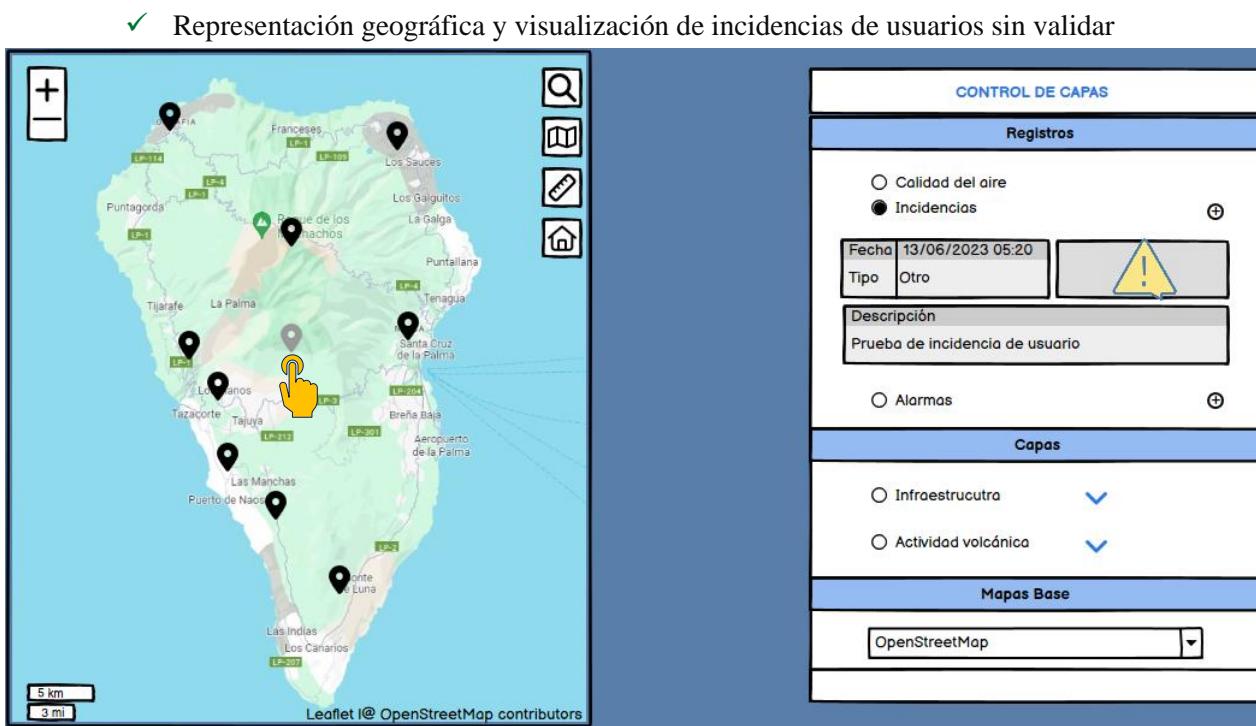


Ilustración 56. Captura mockup interacción de visualización registros de incidencias creadas por el usuario sin validar

Introducción: esta funcionalidad complementa la visualización de incidencias ya que en la respuesta a la creación de nuevas incidencias de usuarios se establece también una estrategia de comunicación de la información de estas, aunque permanezcan sin validar, poniéndolas así a conocimiento de los usuarios que las reportaron.

Entrada: en este caso no es necesario aportar ninguna entrada de datos extra ya que se añade a la funcionalidad de visualización de incidencias general.

Proceso: se añade un proceso al controlador que atiende a las peticiones de obtención de datos, este nuevo proceso se lanza a partir de la lectura del token de acceso y los parámetros de nombre de usuario y rol que este contiene, así si el rol pertenece al de un usuario general, busca mediante su nombre de usuario los registros que hayan sido reportados por él y junto a los registros de incidencias validadas, envía sus datos al cliente.

Salida: gracias a esta funcionalidad, el usuario observa los registros de incidencias validadas y no validadas creadas por él, activando así distinta simbología de representación para estas, diferenciando las verificadas, de las creadas por el usuario.

- ✓ Menú de creación de Alarmas

The form is titled "NUEVA ALARMA". It contains three input fields: "Nombre" (Name), "Rango" (Range), and "Coordenadas" (Coordinates), each with a text input box. At the bottom is a green button labeled "Guardar Alarma" (Save Alarm).

Ilustración 57. Captura mockup formulario de creación de alarmas

Introducción: desde este requerimiento se ofrece al usuario la capacidad de generar alarmas que le avisen automáticamente de alguna incidencia surgida dentro de la zona definida por ellos.

Entrada: los datos a aportar por el usuario son el nombre con el cual se quiera referir el usuario a la alarma, el rango de activación de esta, y las coordenadas en la cual se sitúa su centro.

Proceso: igual que el proceso de creación de registros de incidencias se conecta una nueva página al componente de control de capas, esta página tiene codificado un formulario que recoja los datos de definición de la alarma, cuando el usuario señala él envío de datos el almacén *map* realiza una validación previa de estos y los retransmite a la API por una petición *post*. Esta petición es recibida por el punto de acceso correspondiente, para empezar el procedimiento de subida a la base de datos, sin embargo, antes a través del middleware de verificación de token se comprueban las credenciales de usuario, obteniendo el rol y el nombre del usuario, para así después desde el controlador de registros de alarmas, se empiece nuevamente un proceso de validación de datos y una comprobación del supuesto de que alguna de las incidencias ya validadas se sitúan dentro del rango de alerta definido, siendo en caso afirmativo la activación instantánea de esta alarma y la comunicación de esto vi correo electrónico al usuario. Finalmente se realiza la subida del registro a la base de datos, culminando el objetivo del requerimiento.

Salida: Como salida se visualiza un ícono más (+) en la sección de alarmas dentro del control de capas, este botón permite la llegada a la página de creación de registros de alarmas, desde el cual el usuario podrá aportar los datos de creación de la alarma.

- ✓ Representación geográfica de alarmas registradas

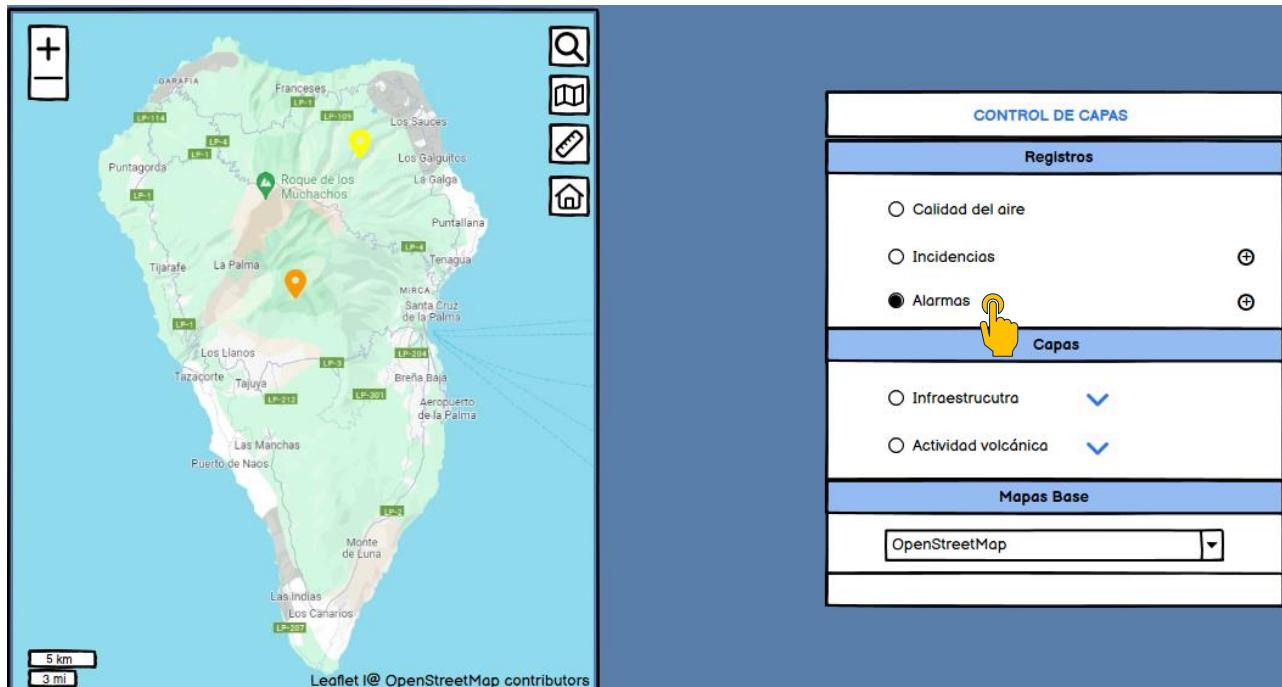


Ilustración 58. Captura mockup interacción de visualización registros de alarmas creadas por el usuario

Introducción: desde este punto se desarrolla la capacidad de que los usuarios puedan localizar sus alarmas dentro del visor web, pudiendo así estudiar la distribución de estas respecto otros registros o datos en el mapa.

Entrada: como aportación el usuario debe activar el botón de visualización de los registros de alarmas, para que así se dibujen las localizaciones de las alarmas del usuario en el visor.

Proceso: el desarrollo aplicado para esta funcionalidad es similar al del resto de visualizaciones de registros, desde el cliente se envía una petición de datos de alarmas, que el endpoint correspondiente recibe, pasa al middleware que verifica el token y extrae el nombre de usuario, pasando después esta información al respectivo controlador que extraerá las localizaciones e identificadores de las alarmas que coincidan con el nombre de usuario proporcionado. Por último, se envían estos datos de vuelta al cliente, donde se almacenan en *map* y se dibujan en el visor.

Salida: como resultado de los desarrollos dados se crea una nueva sección en el panel de control de capas dedicado a las alarmas creadas por cada usuario, con un checkbox desde el cual se puede manejar la visualización de las localizaciones de alarmas en el mapa.

- ✓ Visualización de datos de alarmas registradas

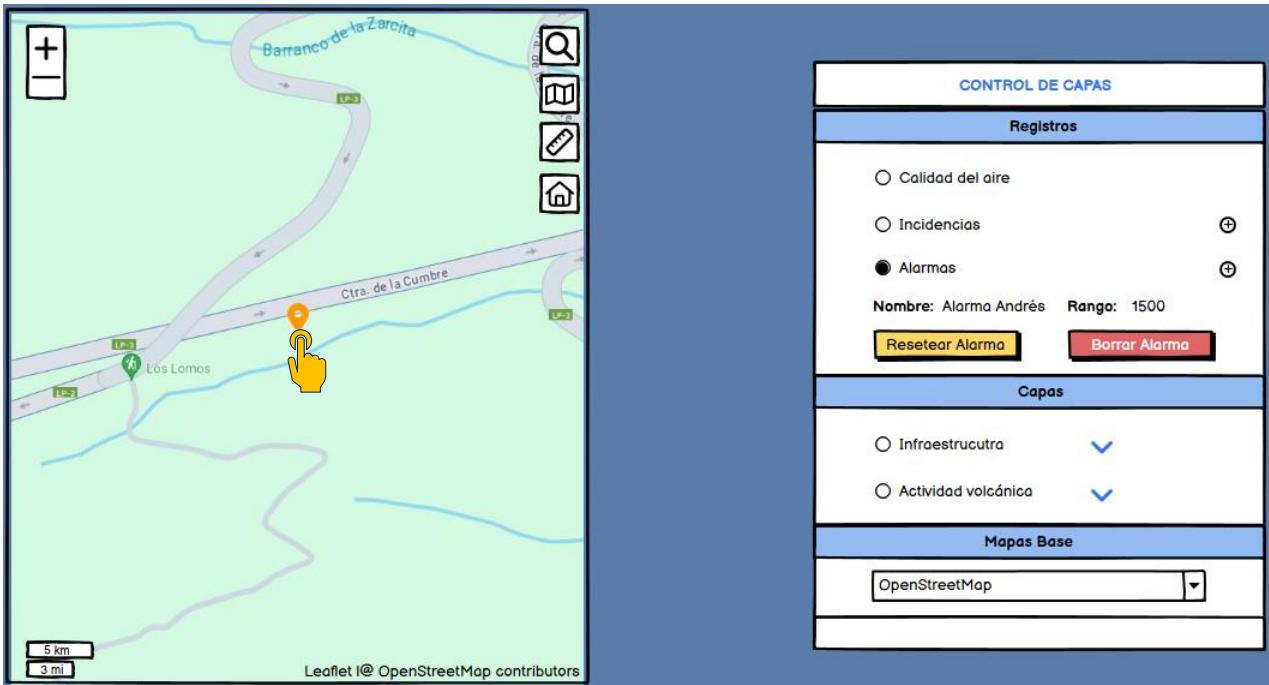


Ilustración 59. Captura mockup interacción de visualización de datos registros de alarmas creadas por el usuario

Introducción: después de la creación de alarmas los usuarios, además de conocer su localización gracias al requerimiento anterior, deben tener la posibilidad de conocer el resto de sus datos, nombre y rango, así como otras funciones que se presentaran más adelante.

Entrada: ningún dato es necesario para el funcionamiento de este proceso, tan solo la decisión del usuario de activar la visualización de estos datos, por los mecanismos establecidos.

Proceso: Para la entrega de datos de alarmas, se realiza en el almacén *map* una petición de datos con el identificador de la alarma como parámetro y con el token de acceso, desde la API, se recibe esta petición y se pasa por el middleware de verificación del token, ya verificado el acceso se llega a un controlador que extrae y envía los datos de la alarma identificada de vuelta al cliente, para ser mostrados al usuario desde el componente *Element Info*, representado esta vez justo debajo de la sección del panel dedicado a las alarmas.

Salida: como resultado tras el click en alguna de las localizaciones de alarmas mostradas en el visor, se genera un cuadro de datos justo debajo de la sección de alarmas del panel de control, donde se muestra la información relativa a la alarma.

- ✓ Desactivación de alarma tras aviso

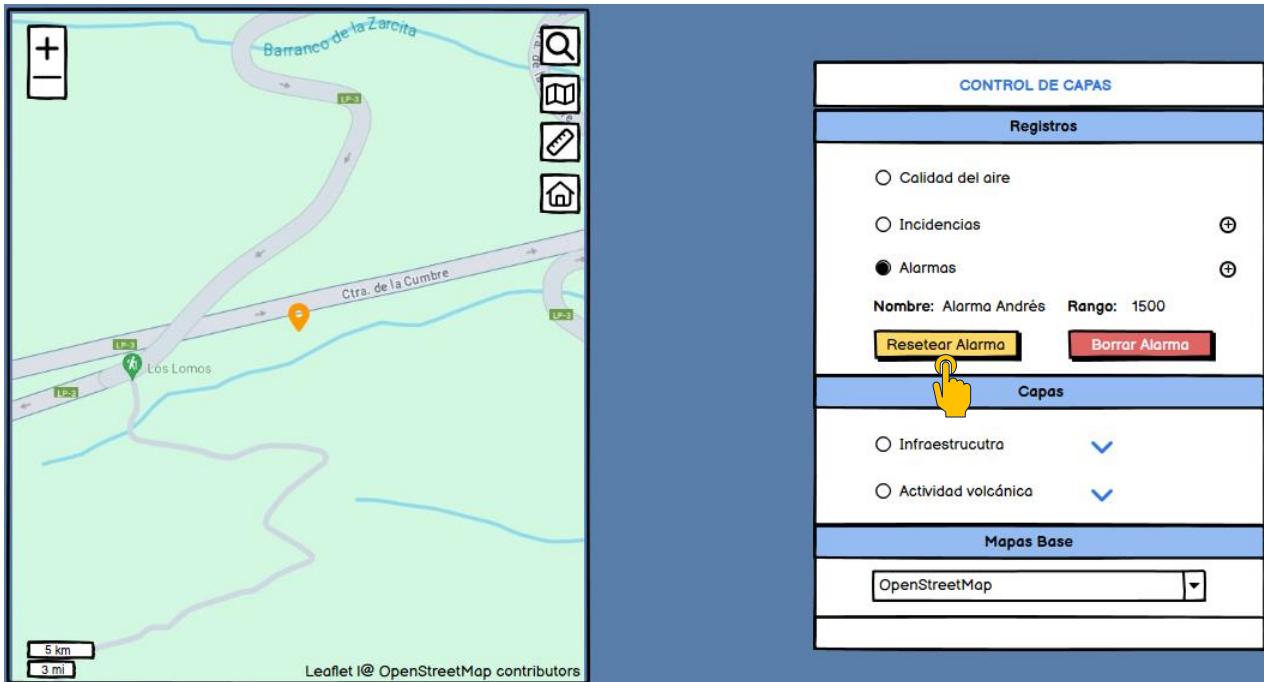


Ilustración 60. Captura mockup funcionamiento reinicio de alarmas activadas

Introducción: esta función permite a los usuarios el cambio del estado de la alarma cuando esta está activada, infiriendo así en el cambio de su simbología en el mapa.

Entrada: decisión de cambiar el estado de la alarma seleccionada a desactivada, tras su activación por la presencia de incidencias cercanas.

Proceso: por un lado, desde el cliente se da acceso a este requisito mediante un botón de inicio situado justo debajo de los datos de la alarma mostrados por el requerimiento anterior. Así mediante este botón se inicia en el almacén *map* un proceso de envío de datos mediante una petición *put* con el identificador de la alarma. Esta petición llega a la API, donde se comprueba el acceso por el middleware y se empieza un proceso de actualización del campo activada dentro del registro de la alarma indicada, pasando este a un valor false. Posteriormente si el proceso resulta exitoso se envía una confirmación de esto al cliente, para que este actualice los datos del registro mediante el reinicio de las variables que manejan los datos de alarmas.

Salida: este desarrollo resulta a nivel del usuario en la visualización de un botón de reinicio en los cuadros de información propios de las alarmas con el campo activado como true.

✓ Eliminación de alarma

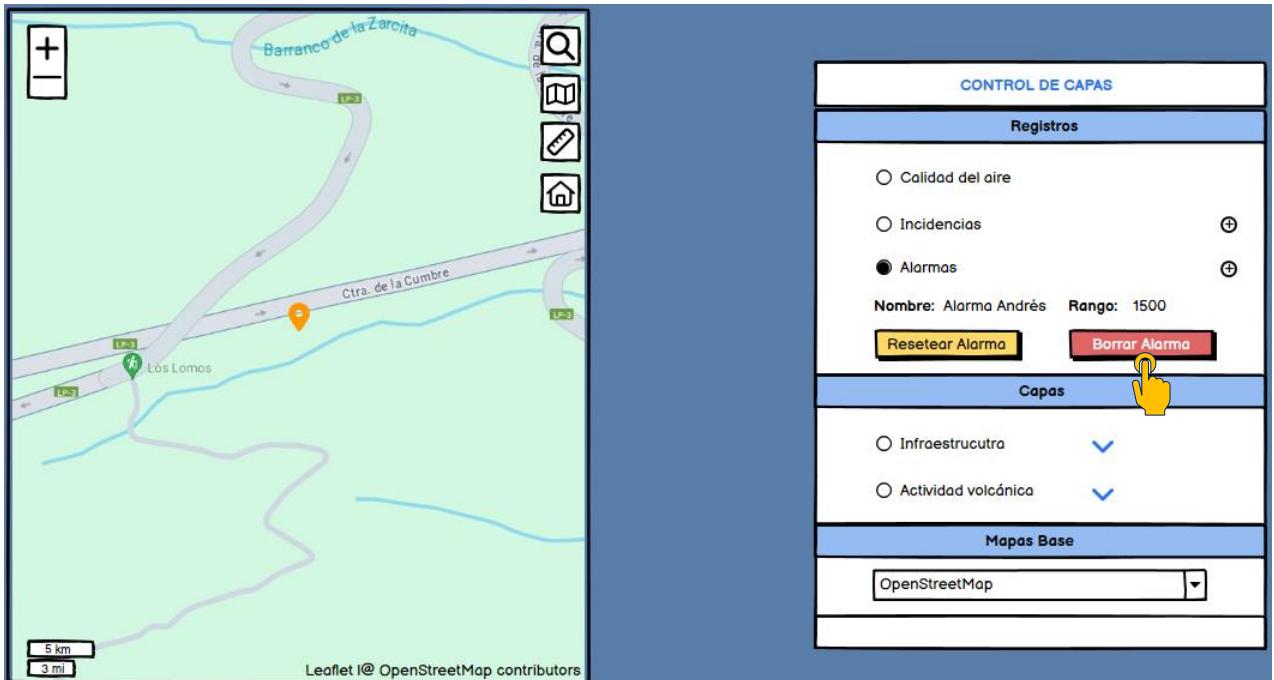


Ilustración 61. Captura mockup funcionamiento eliminación de alarmas de usuarios

Introducción: por último, en la gestión de los registros de alarmas el proceso de eliminación de estas completa el flujo de vida de estos registros, de tal forma que cuando el usuario ya no ve necesaria la existencia de la alarma, este pueda borrarla de la base de datos.

Entrada: decisión de eliminar las alarmas que ya no sean útiles al usuario desde el menú de gestión de la alarma.

Proceso: al cuadro de información de alarmas se le añade un nuevo botón de eliminación de registro, siempre visible a diferencia del de reinicio definido en el punto anterior. Este nuevo botón lanza a través del almacén *map*, una petición *delete* con los datos del token e identificador del registro.

Por otro lado, en el Back-End del proyecto, el endpoint correspondiente recibe la petición *delete* y tras comprobar el token de acceso se borra el registro de la alarma indicada en la base de datos mongo. Posteriormente si el proceso resulta exitoso se envía una confirmación al cliente, para que este actualice los datos de registros de alarmas que guarda en variables del almacén *map*.

Salida: este desarrollo resulta a nivel del usuario en la visualización de un botón de eliminación de alarma en los cuadros de información de todos los registros de alarmas del usuario.

- ✓ Mapa auxiliar para selección de coordenadas en los menús de creación

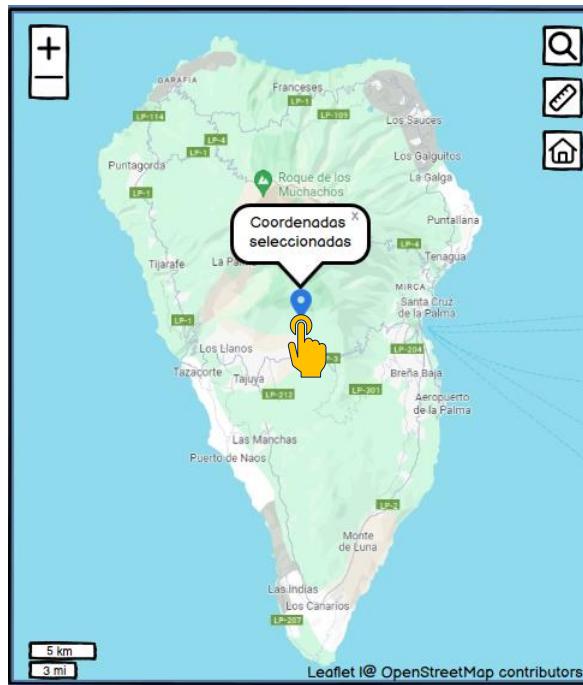


Ilustración 62. Captura mockup de selección de coordenadas por mapa auxiliar

Introducción: como facilidad a los usuarios a la hora de generar registros se creó esta idea de mapa interactivo input complementario de coordenadas. Permitiendo así a los usuarios buscar la localización de interés con la ayuda de visores situados en las mismas páginas de creación de registros.

Entrada: para usar esta funcionalidad los usuarios deben interpretar el mapa para localizar y clicar el punto de interés del cual quieren crear el registro.

Proceso: desde el cliente se crea un nuevo componente *mapa* muy similar al principal pero sin interacciones con el panel de control de capas, sino solo con el menú de creación de registros ya sea de alarmas o incidencias, después se creó un evento de mapa que se dé al hacer click izquierdo sobre el visor, para que cuando este se dé se represente con un símbolo la localización seleccionada y se realice un centrado sobre esta, además se extraen las coordenadas obtenidas y se transforman al formato requerido por la API, para ser comunicadas y presentadas en el menú de creación correspondiente, sobre el input dedicado a coordenadas.

Además, al visor se le añadirán funcionalidades también presentes en el principal, como el control de zoom, la herramienta de medición y la geocodificación directa.

Salida: como resultado al lado izquierdo de los menús de creación en las páginas correspondientes se visualiza un visor con mismas dimensiones al principal, preparado para la ayuda en la selección y búsqueda de localizaciones por parte del usuario.

- ✓ Descarga de datos históricos de estaciones de calidad de aire

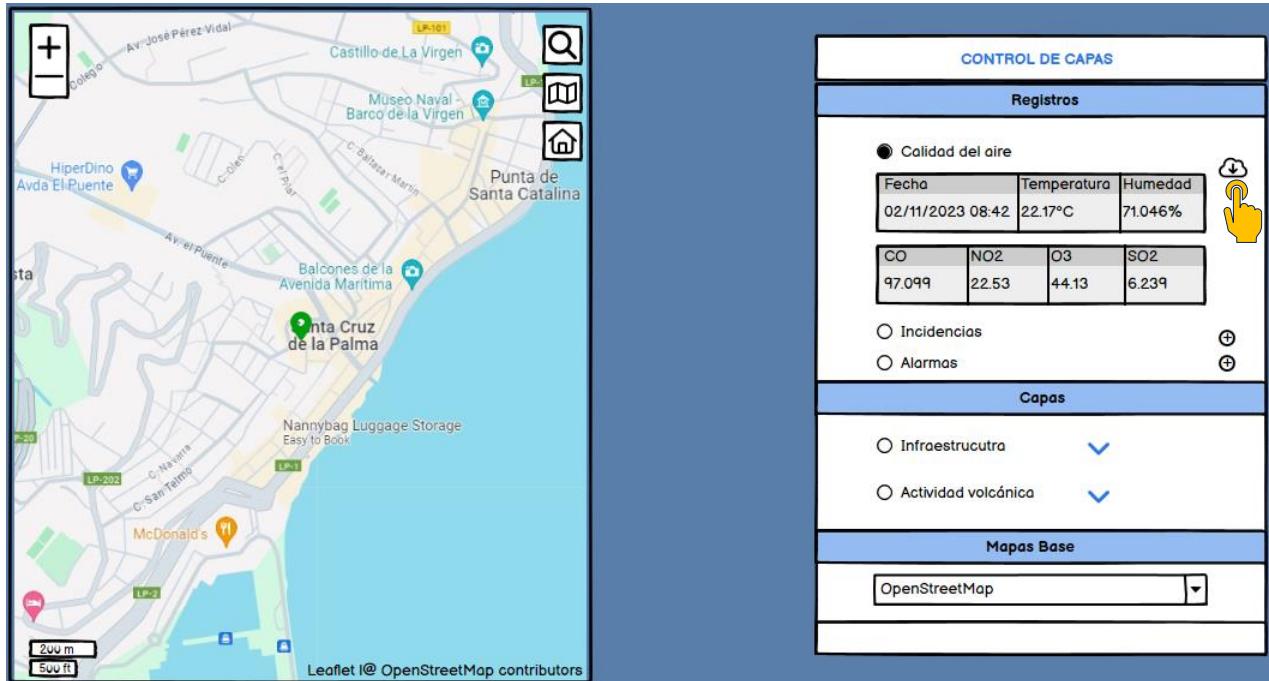


Ilustración 63. Captura mockup funcionamiento descarga de registros históricos por estación meteorológica

Introducción: este requerimiento busca proporcionar a los usuario registrados la información histórica de datos de calidad del aire por estación meteorológica.

Entrada: como entrada está el criterio del usuario en elegir de que estación meteorológica quiere conocer los registros históricos.

Proceso: para esta funcionalidad se necesita una nueva petición de datos a la API del proyecto, sin embargo, esta vez no se realiza esta acción desde el almacén *map* sino directamente desde el componente que recibe la indicación del usuario *element info*, el cual contiene un botón de descarga que, al ser pulsado, iniciara el proceso de construcción y solicitud de datos desde el mismo componente enviando como parámetros el token, identificador del registro, e idioma actual de la aplicación. Desde la API se recibe esta petición, se comprueba el token de acceso y se empieza el proceso de búsqueda de datos obteniendo primero el nombre de la estación y después datos históricos de esta mediante el identificador del registro. Luego con estos se construye un archivo CSV, en formato tabla y con el idioma seleccionado, para finalmente proporcionar el archivo a descargar, el cliente recibe la respuesta de descarga y ejecuta la conexión para finalmente proporcionar el archivo con registros históricos al usuario.

Salida: finalmente a nivel de interfaz el usuario observa un nuevo símbolo de descarga de registros empleado como botón, que ejecuta el proceso de obtención del histórico de la estación, y su proporcional descarga.

✓ Salida de sesión

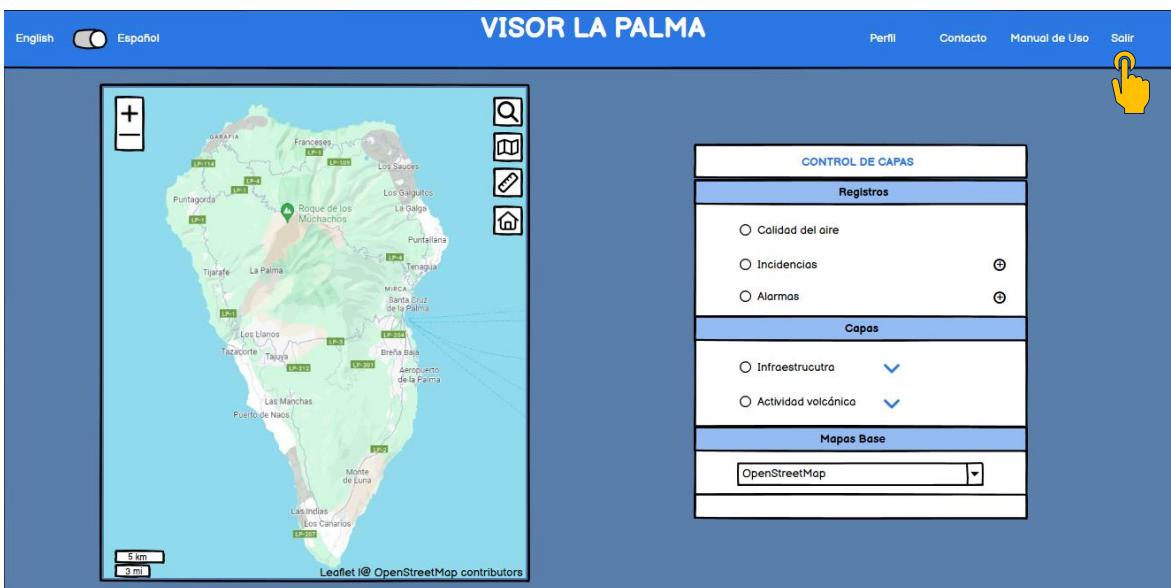


Ilustración 64. Captura mockup de acceso de salida de sesión

Introducción: tras abrir sesión también aparece la necesidad en la aplicación de ofrecer al usuario la opción de cerrarla.

Entrada: decisión de cerrar la sesión e indicarlo en la aplicación.

Proceso: en el lado cliente se habilita en la cabecera el input de salida de sesión, el cual al ser ejecutado indica al almacén *authentication* el lanzamiento de una petición *delete* al endpoint *logout* de la API del proyecto, desde allí se verifica el token de acceso proporcionado, y también se elimina de la base de datos, posteriormente se confirma el éxito del proceso al cliente, para desde allí borrar toda la información relativa al usuario y así cerrar su sesión.

Salida: los usuarios registrados pueden ver en el menú de navegación de la cabecera una entrada más de salida de sesión, con la que pueden interaccionar para ejecutar la función.

- ✓ Representación geográfica y visualización de incidencias sin validar, rol administrador

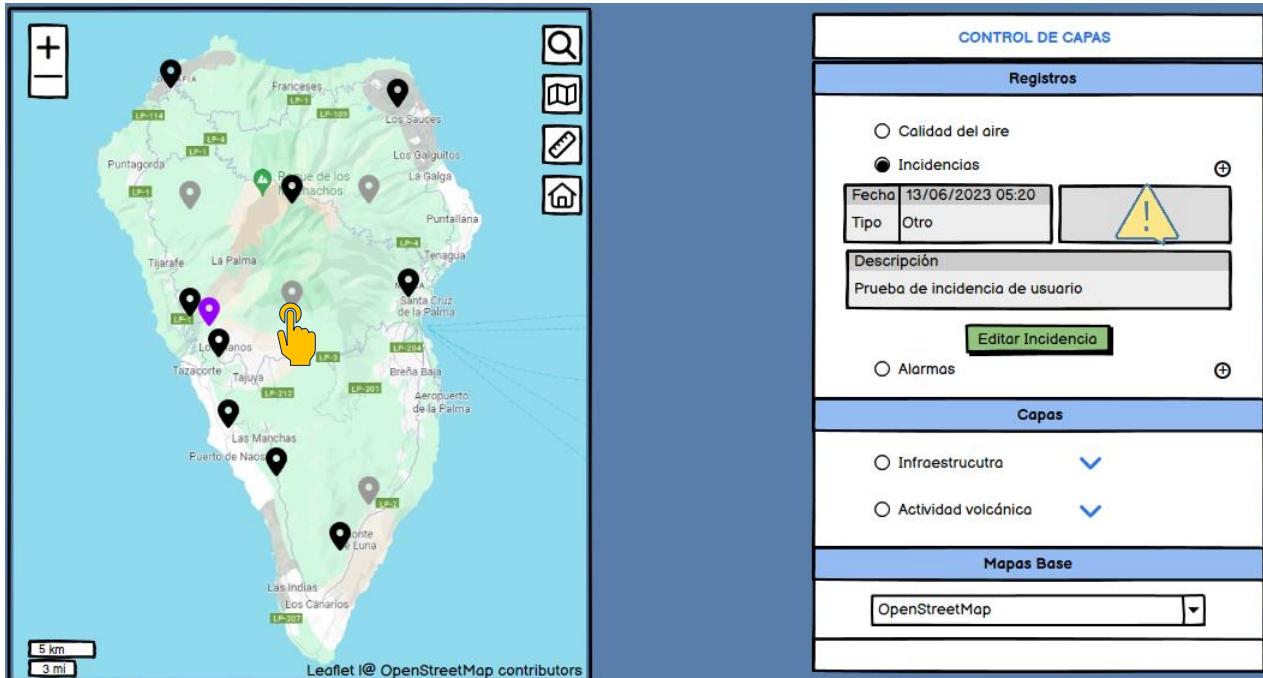


Ilustración 65. Captura mockup interacción de visualización registros de incidencias sin validar para usuarios administradores

Introducción: esta funcionalidad complementa la visualización de incidencias, como respuesta a la validación de nuevas incidencias por parte de administradores, para añadir contenido veraz al proyecto y atender a las peticiones de usuarios. Se establece la comunicación total de todo tipo de incidencias, sin validar o validadas a los usuarios con rol administrador, con objeto de habilitar su función de validación y gestión incidencias.

Entrada: en este caso no es necesario aportar ninguna entrada de datos extra ya que se añade a la funcionalidad de visualización de incidencias general.

Proceso: se añade un proceso al controlador que atiende a las peticiones de obtención de datos, este nuevo proceso se lanza a partir de la lectura del token de acceso y el valor de rol que este contiene, así si este contiene el valor asociado a un usuario administrador, busca todos los registros de incidencias presentes en el proyecto, ya hayan sido validados o no y los envía al cliente web. Dando allí la representación geográfica de todos los registros de incidencias existentes en la colección *registros_incidencias* de la base de datos Mongo.

Salida: gracias a esta funcionalidad el administrador observa registros de incidencias de todo tipo, validos, sin validar creados por usuarios, sin validar añadidos de la Dirección General de Tráfico, etc. Además, se activa la distinta simbología de representación para estas, diferenciando las verificadas, de las creadas por usuarios o introducidas a revisar de la DGT.

- ✓ Creación de incidencias validadas

The image shows a split-screen interface. On the left is a map of a volcanic area with roads labeled 'Ctra. de la Cumbre' and 'Barranco de la Zarzita'. It includes a legend with zoom controls (+/-), a search icon, and a location pin. Below the map is a scale bar from 5 km to 3 mi. At the bottom is the text 'Leaflet © OpenStreetMap contributors'. On the right is a form titled 'NUEVA INCIDENCIA' with fields for: Nombre en español* (Incidencia Andrés), Nombre en inglés* (Andres incidence), Tipo de incidencia* (Otro), Coordenadas* (28.7720, -17.7823), Imagen (empty box), Descripción en español* (prueba de incidencia), and Descripción en inglés* (incidence test). A green 'Enviar Incidencia' button is at the bottom.

Ilustración 66 Captura mockup formulario de creación de nuevas incidencias validadas

Introducción: para los usuarios administradores se modifica la creación de incidencias, mediante la posibilidad de definir también los campos de traducción presentados en el modelo de datos de incidencias y la validación automática de las incidencias creadas.

Entrada: se requieren los datos definidos en la funcionalidad de creación de incidencias con la suma de los valores de los campos de traducción al inglés.

Proceso: el proceso es similar al empleado en la creación de incidencias para usuarios generales, con solo la diferencia de la incorporación de dos entradas de datos más ocupadas para la traducción al inglés de los campos ya existentes, después la diferencia aparece en la API, que detecta el rol del usuario guardado en el token, guarda los datos definidos en los campos de traducción dedicados tanto en castellano como en inglés y valida los registros, mediante la definición del campo validada igual a true, siendo así visible para todos los usuarios registrados o sin registrar de la aplicación. Además, al ser ahora esta una incidencia validada se verifica la distancia que existe entre esta y las alarmas definidas por los usuarios, de tal forma que, si la incidencia entra en alguna de las zonas vigiladas, se envía un mail al usuario propietario de la alarma, para así avisarle de la presencia de esta nueva incidencia, y se actualiza el valor de *activa* a true en el registro de la alarma correspondiente.

Salida: como resultado de este requerimiento la vista de la página de creación de incidencias de los usuarios administradores cambia respecto a las de los usuarios generales, con la incorporación de dos entradas de datos extras dedicados a los campos en inglés.

✓ Edición y validación de incidencias

The mockup consists of two main parts. On the left is a map of a rural area with roads labeled 'Barranco de la Zarcita', 'Ctra. de la Cumbre', and 'Los Lomos'. It includes a scale bar (5 km/3 mi), a compass rose, and a legend with icons for search, edit, and home. On the right is a form titled 'EDITAR INCIDENCIA' with the following fields:

Nombre en español*	Incidencia Andrés
Nombre en inglés*	Andres incidence
Tipo de incidencia*	Otro
Coordinadas*	28.7720, -17.7823
Imagen	(Empty input field)
Descripción en español*	prueba de incidencia
Descripción en inglés*	incidence test

At the bottom are two buttons: a green 'Guardar Cambios y Validar Incidencia' button with a yellow hand cursor icon pointing to it, and a red 'Borra Incidencia' button.

Ilustración 67. Captura mockup formulario de edición y validación de incidencias validadas o no

Introducción: para los usuarios con rol de administradores se ofrece esta funcionalidad de gestión de registros de incidencias, que permite la edición de registros y la validación de nuevas incidencias, a partir de las ya creadas por los usuarios generales.

Entrada: como entrada se debe aportar la actualización de datos de incidencias validadas o la revisión y complementación de los datos de una incidencia sin validar para verificar su veracidad y así confirmar su validación.

Proceso: Por otro lado, la edición de incidencias implica la creación de una nueva vista en cliente, para una nueva página de edición, que está accesible a los usuarios administradores desde el cuadro de datos de incidencias. En esta nueva página los usuarios tendrán un menú y visor similar al de la página de creación de incidencias, con las mismas entradas de datos, en las cuales se aprovecharán los datos ya guardados en cliente mostrando como placeholders la información actual del registro. Por otro lado, la comunicación con la API es distinta, ya que ahora se emplea una petición de actualización datos tipo *put*, con los cambios realizados a la incidencia. Desde la API se recibe esta petición al endpoint correspondiente, después se verifica el token y se extrae el rol del usuario para comprobar que es administrador, posteriormente se realiza un proceso de actualización del registro de la incidencia señalada de la base de datos, donde se cambia el valor

del campo *validada* a true si este no figuraba ya así, implicando por tanto la suma o modificación de incidencias validadas en el proyecto, y así la necesidad de revisión de alarmas, comprobando si la incidencia subida se localiza dentro del rango de alguna de las alarmas de los usuarios, para si es así cambiar su valor de *activada* a true y avisar de esto vía mail al usuario propietario. Por último, se envía la confirmación o código de error del proceso realizado al cliente, que se ocupara de transmitir la respuesta al usuario.

Salida: el usuario administrador vera un botón de edición en el cuadro informativo de incidencias situado en la sección incidencias dentro del panel de control de capas, desde este tiene acceso a la página de edición de incidencias, donde se encontrara un menú muy parecido al de creación con las diferencias de que vera los valores ya presentes en el registro dentro de los inputs de entrada de datos, incluyendo también el de la localización en el del mapa de ayuda, estos permanecen así en primera estancia a espera de aplicar modificaciones del usuario, luego el botón de envío de datos indica la validación automática del registro.

- ✓ Eliminación de registros de incidencias

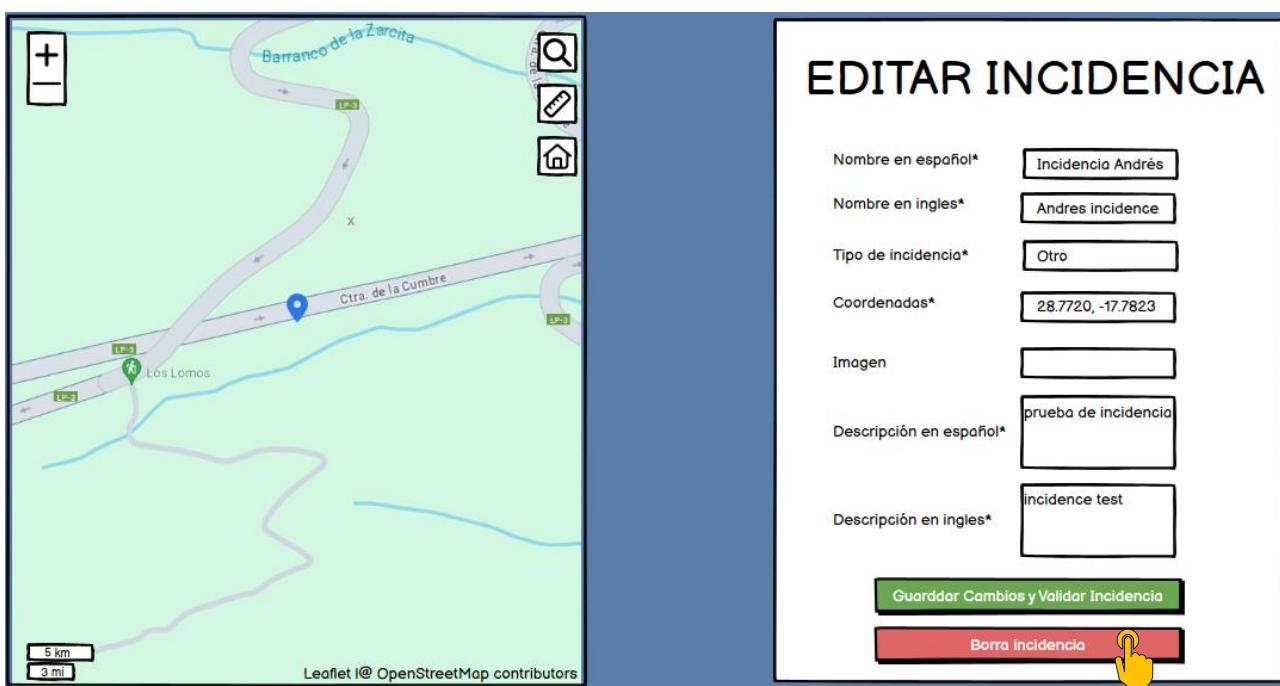


Ilustración 68. Captura mockup acceso a eliminación o rechazo de incidencia

Introducción: por último, en la gestión de incidencias a los usuarios administradores también se les proporciona esta funcionalidad de eliminación de incidencias, completando así el flujo de vida de estos registros, cuando ya no son representativos de la realidad de la isla de La Palma.

Entrada: criterio y valoración de eliminación de incidencias, que ya no aporten información valida y de interés en el proyecto.

Proceso: para completar la página de edición se añade a esta un botón de eliminación de incidencias, el cual se enlaza al envío de una petición *delete* desde el almacén *map*, esta petición entrega a la API del proyecto el token e identificador del registro proporcionados por el usuario administrador. Desde el endpoint establecido se deriva la petición al middleware de revisión de token, que tras comprobar el acceso y extraer el rol, comunica este proceso con el controlador, que revisa el rol del usuario y ejecuta entonces la eliminación del registro en la base de datos mongo. Finalmente, si todo ha ido bien se envía la confirmación de la ejecución del proceso al cliente.

Salida: el usuario administrador, ve en la página de edición de incidencias un botón de eliminación de incidencias justo debajo del botón de envío del formulario de actualización.

- ✓ Búsqueda y visualización de perfiles de usuario

The mockup shows a search interface with a title 'CONTROL DE SUPERADMINISTRADOR'. Below it is a text input field labeled 'Usuario a modificar' and a green button labeled 'buscar usuario'.

Ilustración 69. Captura mockup de formulario de búsqueda de perfil de usuario

The mockup displays a user profile with the name 'user1'. It shows the following details:
Nombre: Andrés Apellido: Chasiluisa
Municipio: El Paso Mail: as.chasiluisa@alumnos.upm.es
Rol: user
Buttons: 'cambiar rol' (in orange), 'user1' (in red), and a red 'volver' button.

Ilustración 70. Captura mockup menú de control de perfil de usuario encontrado

Introducción: para el usuario super administrador se le ofrecen funcionalidades en objetivo a la gestión de perfiles del resto de usuarios, para ello existe este primer requerimiento de creación de un menú de búsqueda de perfiles de usuario.

Entrada: para aplicar esta búsqueda se debe aportar el nombre del usuario del que se quiere conocer y gestionar el perfil.

Proceso: se crea una nueva vista que represente a la página de control del super administrador, esta solo está accesible al usuario con este rol, y esta será la única página accesible al usuario super administrador. En esta vista se crea un formulario con un solo input destinado al nombre de usuario de la cuenta que el super administrador quiere visualizar o gestionar, desde aquí se recoge la información del usuario, con la que se genera la petición de datos a la API. Ya recibida la petición se revisa el token de acceso con su rol, y después a partir del nombre de usuario, se obtienen el resto de los datos de la cuenta *nombre, apellido, municipio, mail* y *rol*. Estos se envían de vuelta al cliente, el cual, al recibirlos con la respuesta de confirmación del proceso, en la página de control, esconde el formulario de búsqueda y dibuja un menú de control con los datos del usuario recibidos. Este menú también cuenta con un botón de regreso al formulario de búsqueda por si el super administrador prefiere visitar el perfil de un usuario distinto y reiniciar el proceso.

Salida: como salida el usuario super administrador al registrarse en la aplicación es dirigido a la página de control de super administrador, esta página es la única a la que puede acceder este usuario, y solo puede ser visitada por él, para así preservar la seguridad de la aplicación, siendo ofrecidas a él solo las funcionalidades de la página de control. En primera instancia en esta página visualiza un pequeño formulario de búsqueda de perfiles de usuario según nombre de usuario, que al ejecutarse y ser encontrada la cuenta, muestra los datos del usuario buscado en un menú de gestión de perfiles de usuarios.

- ✓ Cambio de rol de perfiles de usuario



Ilustración 71. Captura mockup acceso a funcionalidad de cambio de rol

Introducción: para facilitar la gestión de obtención o perdida de roles de administradores, se ofrece esta funcionalidad al super administrador, permitiéndole el control de los roles de los usuarios que usan la aplicación, teniendo en cuenta siempre la lógica de trabajo que mantenga la administración encargada.

Entrada: criterio de cambio de rol para una cuenta de usuario determinada.

Proceso: en el menú de gestión de perfiles de usuario presentado en el requerimiento anterior, se incorpora un texto de cambio de rol, el cual al ser pulsado ejecuta en el almacén de datos *authentication* una petición *put* de cambio de rol, con los valores del token de acceso, nombre del usuario y el rol que se le va a asignar. Desde la API se recibe la petición desde la ruta de entrada *super admin control*, a partir de aquí se comprueba la validez del token de acceso por el middleware de verificación, y se lanza el proceso de cambio de rol en el controlador, mediante la modificación al rol definido en el perfil identificado por el nombre de usuario, después del éxito del procedimiento se renvía al cliente la información actualizada del usuario, para que este la vuelva a representar en el menú y conserve así la realidad de los datos.

Salida: por parte del usuario super administrador, observa un texto de cambio de rol en un lado del menú de gestión de perfiles, justo debajo de los datos del usuario, desde este botón puede cambiar el rol figurado en la cuenta, de tal forma que, si fuese de usuario general pase a administrador, y viceversa.

- ✓ Eliminación de perfiles de usuario



Ilustración 72. Captura mockup acceso a funcionalidad de eliminación de perfil de usuario

Introducción: por último, el requerimiento de eliminación de cuentas completa la gestión de perfiles de usuario de la aplicación, completando el flujo de vida de las cuentas de usuario desde la misma aplicación.

Entrada: criterio de eliminación de la cuenta seleccionada.

Proceso: nuevamente en la página de control, en el menú de gestión de perfiles de usuario se añade un texto que represente la eliminación de la cuenta del usuario, y que tras ser clicado ejecute el proceso de petición *delete* de la cuenta seleccionada, enviando como parámetros el token y nombre de usuario a eliminar. En la API se recibe esta petición, se comprueba como siempre el token de acceso y su rol, para dar pie al proceso de búsqueda y borrado de la cuenta de usuario con el nombre proporcionado, enviando tras esto la confirmación a cliente que vuelve a dibujar el formulario de búsqueda de usuarios, finalizando así el procedimiento.

Salida: ahora se muestra también al lado derecho del menú de gestión de perfiles de usuario, justo al lado del cambio de rol, un texto con el nombre del usuario entre tachado, que comunica al super administrador el lanzamiento del proceso de eliminación de la cuenta de usuario allí definida.

6.2 REQUISITOS NO FUNCIONALES

- ✓ **Interfaz de usuario dinámica e intuitiva**, aprovechando las propiedades de Vue, y sus componentes, mediante la conexión de e interacción entre estos, combinados con estilo atrayentes en sus elementos.
- ✓ **Fácil interpretación de datos**, para lograr llegar al máximo de público posible se orientan las funcionalidades estrategias y mecanismos de la aplicación a la muestra sencilla e ilustrativa de los datos, y al enfoque cotidiano que cualquier ciudadano o ciudadana de La Palma pueda aprovechar día a día.
- ✓ **Alto rendimiento en la visualización de datos**, procurando comunicaciones rápidas y sencillas entre cliente y servidor, y la buena visualización de estas en los componentes como el visor de tipo mapa.
- ✓ **Seguridad y protección de datos**, mediante la comprobación y validación sistemática de los accesos a funcionalidades y datos importados al proyecto, tanto en el cliente web como en el servidor.

7. ARQUITECTURA DEL SISTEMA

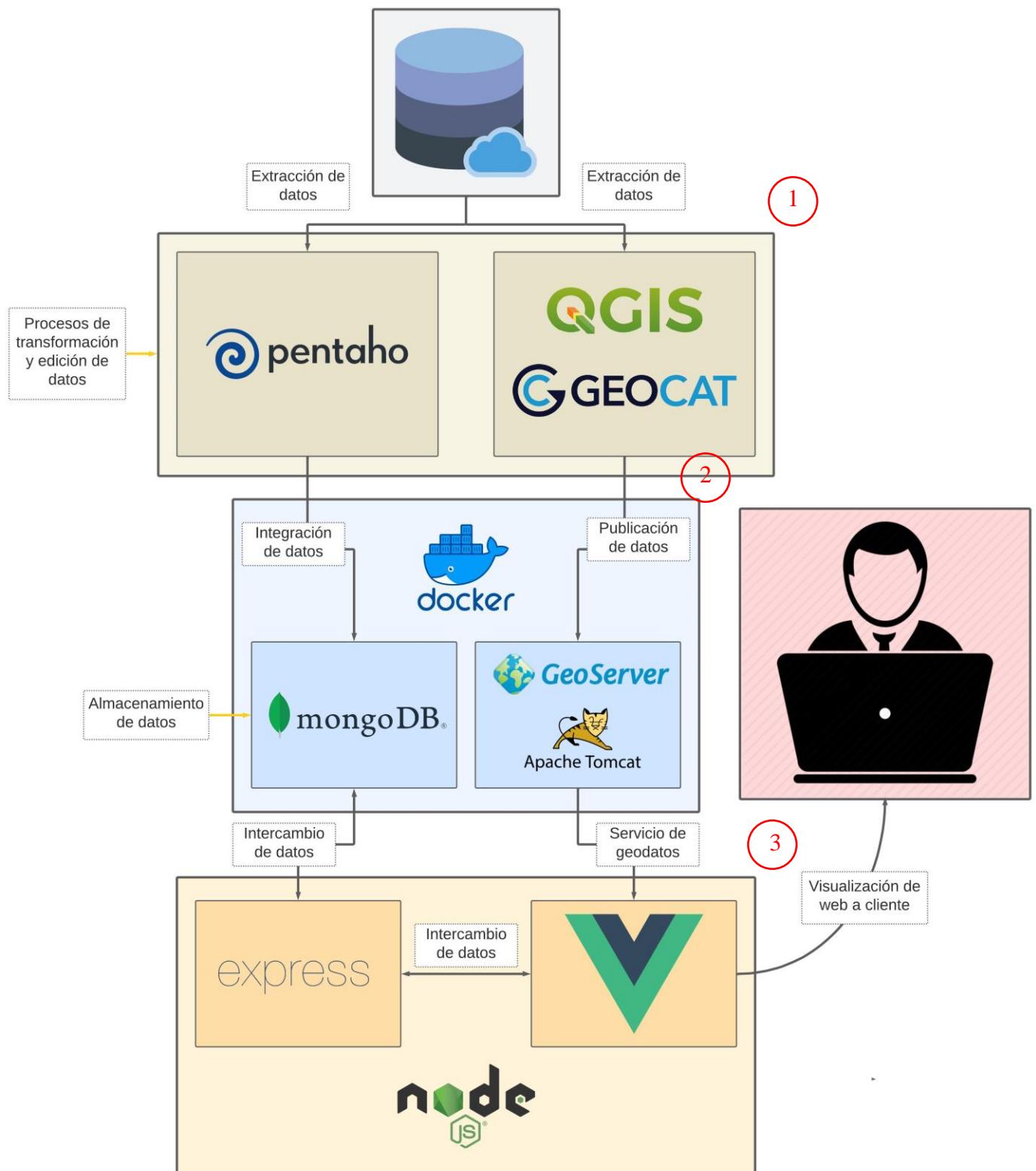


Ilustración 73. Diagrama de arquitectura del sistema

Como se puede ver en el diagrama, de la página anterior, existen tres bloques principales, el número 1 de adquisición de datos, el número 2, correspondiente a los contenedores Docker, de almacenamiento de datos, y el punto 3, propio de la estructura cliente servidor. Estos bloques se introducirán brevemente en los siguientes apartados.

7.1 ADQUISICIÓN DE DATOS

La adquisición de datos viene dada de distintas fuentes de información, las correspondientes a registros de cualquier tipo son consultadas por Pentaho Data Integration, aplicación que realizará procesamientos y demás transformaciones para lograr una buena integración en la base de datos no relacional de MongoDB. Por su parte, las correspondientes a información geográfica, tratan datos estáticos, es decir, que no requieren una actualización constante a diferencia de los registros, por ello han sido descargadas manualmente y dotados de estilo gracias a la tecnología QGIS, para posteriormente, mediante su extensión GeoCat Bridge, ser publicados en Geoserver.

7.2 ALMACENAMIENTO DE DATOS

Este punto del proceso está compuesto principalmente por la base de datos Mongo, ya que guarda información tanto proveniente de la integración realizada por Pentaho Data Integration, como por la guardada por el servidor del proyecto en express.js. A parte también se almacenan datos, en Geoserver ya que guarda las capas y el estilo de este fruto de su publicación desde QGIS, para dar respuesta a las solicitudes de visualización formuladas directamente por el cliente Vue.

7.3 ARQUITECTURA CLIENTE-SERVIDOR

El bloque más importante de este proyecto es el de la arquitectura cliente-servidor, un modelo fundamental en el desarrollo web, que se fundamenta en la separación principal de dos entidades, el cliente y el servidor [61].

El cliente, Es la parte de la aplicación que se ejecuta en la máquina del usuario, generalmente a través de un navegador web. El cliente solicita recursos o servicios al servidor y presenta la información al usuario.

El servidor, Es la parte de la aplicación que se ejecuta en una máquina remota o en la nube. El servidor recibe las solicitudes del cliente, procesa la lógica de negocio, accede a bases de datos y envía las respuestas correspondientes de vuelta al cliente.

Como se puede ver esta arquitectura permite la separación de preocupaciones, lo que significa que el cliente y el servidor pueden evolucionar de manera independiente, esto causa una

especialización concreta de cada elemento, lo cual se ve en la diferencia de entornos, ya que, por un lado, el Back-End emplea la tecnología express.js, mientras que, por otro lado, el Front-End funciona mediante Vue.js. Esta y demás características únicas de cada proyecto se repasarán a continuación.

7.3.1 Estructura del Servidor

Como se puede ver en la figura 3, el Back-End está formado por varios archivos y carpetas, algunos propios de un proyecto node.js, como la carpeta node_modules, y otros propios a la organización del sistema como la carpeta middlewares. En los próximos subapartados se repasarán brevemente el desempeño que tiene el código de cada elemento del árbol de archivos.

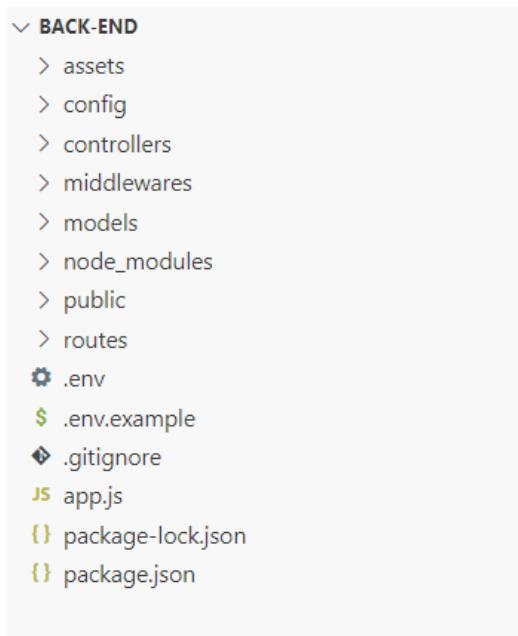


Ilustración 74. Árbol de archivos del servidor del proyecto

Carpeta assets: en este espacio se guardan los distintos archivos externos a los que tiene acceso el proyecto, manuales de usuario, archivos temporales, etc.

Carpeta config: se trata las conexiones con la base de datos mongo y con el correo electrónico, con el cual trabaja Nodemailer. A parte, se guarda un archivo de errores, donde se encuentran almacenados los distintos mensajes de error que puede enviar el servidor.

Carpeta controllers: esta carpeta guarda el contenido principal del servidor, pues se encuentran todos los procesos de respuesta a las peticiones HTTP recibidas por el Back-End, controladores. La organización dentro de este espacio es paralela a la de la **carpeta routes**, es decir, a cada ruta diferente le corresponde un controlador que atienda a los endpoints, que esta presenta.

Carpeta middlewares: guarda todos los middlewares que presente el servidor. Los middlewares son preprocesos que realiza el servidor antes de pasar al controlador, por tanto, un paso intermedio entre recepción de la solicitud y controlador de esta. Suele servir para validar la información recibida o la obtención de datos derivados de esta necesarios para el procesamiento de la petición.

Carpeta models: es un espacio requerido por la tecnología Mongoose, donde se ubican los modelos de datos presentados por las tablas de Mongodb, que se van a consultar.

Carpeta node_modules: esta carpeta es característica de cualquier proyecto de node.js, ya que es donde se almacenan los módulos y librerías que permiten el correcto funcionamiento de la aplicación.

Carpeta public: aquí se guarda cualquier archivo que deba estar disponible para el acceso de cualquier otro componente del servidor.

Carpeta routes: las rutas de acceso son un parte fundamental de cualquier servidor, estas son las puertas de entrada de solicitudes de cliente. En cuanto a su organización en este proyecto es la de creación de un archivo individual para cada ruta distinta que presenta el servidor, en estos archivos se describen si se esperan solicitudes *get*, *post*, *update*, ... y su procesamiento correspondiente, ofrecido por el controlador de la ruta.

Archivos “.env”: son los archivos donde se guardan las variables de entorno del proyecto. En este caso existen dos, ya que uno corresponde al local y otro es el que se encarga de informar sobre las variables de entorno necesarias al ser publicado en el repositorio Git.

Archivo gitignore: se recopilan los path de los archivos que no entran en el control de versiones ofrecidos por Git.

Archivo app.js: el archivo más importante del proyecto, ya que configura el despliegue de cualquier proyecto node.js.

Archivos package.json: son los archivos que indican en cada proyecto de Node.js que módulos son necesarios para el correcto funcionamiento de la aplicación.

7.3.2 Estructura del Cliente

Por su lado la estructura del Front-End es muy similar a la del Back-End, ya que ambos son proyectos node, que comparte muchas configuraciones en común por ello en este apartado, se evitará caer en repeticiones y se omitirá la descripción de elementos citados anteriormente.

En cuanto al árbol de archivos del lado cliente, es sumamente extenso ya que tiene una profunda organización que busca la optimización del orden del código.

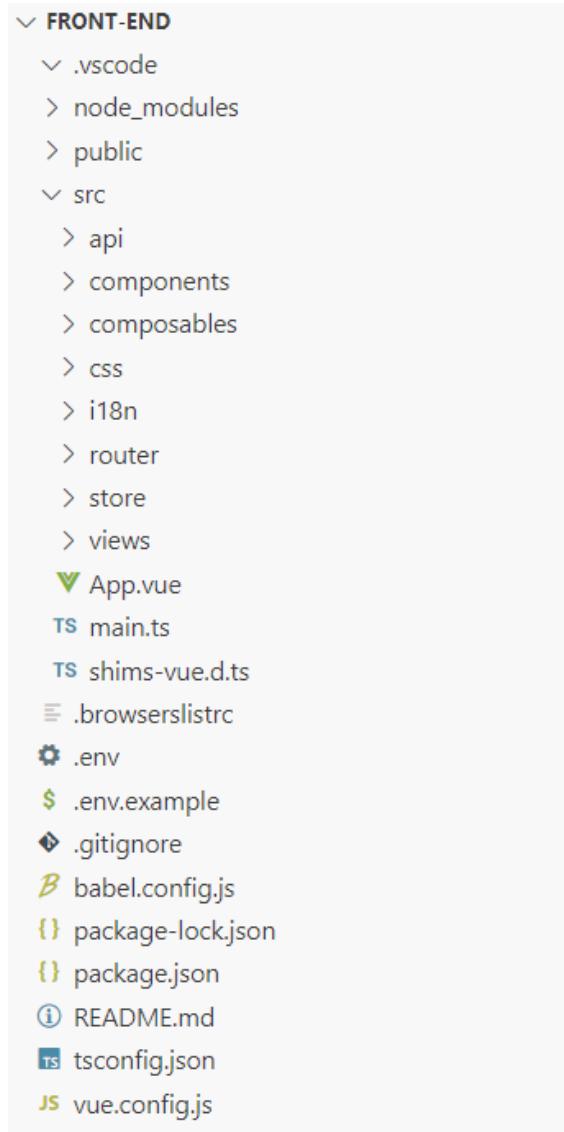


Ilustración 75. Árbol de archivos del cliente del proyecto

Carpeta src: se trata del contenido principal del proyecto, donde se da el código fuente del cliente, que interactúa directamente con el usuario. Tiene un largo desglose, una multitud de archivos y carpetas que pretenden la optimización de la organización y una mejor interpretación del código.

Carpeta src/api: esta carpeta guarda las distintas APIs empleadas en el Front-End. Facilitan la comunicación entre el servidor y el cliente, mediante la exportación de una serie de objetos que ayudan a la interpretación de las respuestas del servidor, así como a la formación y envío de solicitudes a este.

Carpeta src/components: aquí se guardan los componentes Vue del Front-End, es decir, unidades reutilizables de la aplicación Vue, que ayudan a la organización del proyecto. En concreto en este

trabajo cada componente se guarda en una carpeta correspondiente que alberga dos archivos, *nombreComponente.vue* y *nombreComponente.ts*, encargado el primero en la codificación HTML y CSS del proyecto dentro del entorno Vue, y el segundo en codificación Typescript del componente.

Carpeta src/composables: naturalmente en esta carpeta se encuentran los componibles, archivos encargados de dar acceso a los elementos guardados en el store o almacén. Su disposición es de la configuración de un archivo composable por cada almacén de datos dispuesto en el proyecto Vue.

Carpeta src/css: guarda los archivos de estilos globales, para facilitar el acceso a clases y diseños compartidos por los distintos componentes y vistas, simplificando así el código de la aplicación.

Carpeta src/i18n: aquí se ubican los archivos JSON encargados de almacenar las traducciones de la aplicación web, en este caso las destinadas al idioma de español y de inglés.

Carpeta src/router: esta carpeta presenta únicamente el archivo *index.ts*, fruto de la tecnología vue.router, que sirve como índice de las rutas existentes en el visor web, asignando las vistas correspondientes a cada una de ellas.

Carpeta src/store: en esta sección del árbol están los stores del proyecto, los cuales derivan del módulo Vuex y facilitan la comunicación de los datos entre las distintas partes del cliente.

Como se puede ver esta carpeta se ve compuesta por otras tres, una para cada almacén distinto de datos, separados de forma estratégica para la buena organización de los datos, a parte existe un *index.js* principal en el que se declaran los tres almacenes distintos. Estos siguen a su vez una estructura de árbol, permitiendo el desarrollo de sus componentes en los 5 más importantes.

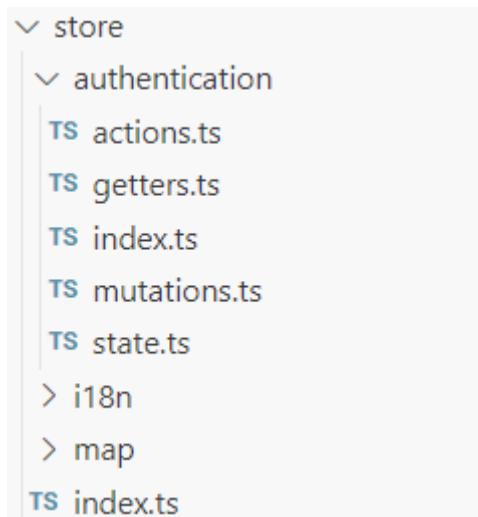


Ilustración 76. Árbol de archivos de la carpeta store del cliente

Carpeta src/store/authentication/actions.ts: este archivo guarda las funciones encargadas de solicitar datos al Back-End requeridos para el desarrollo del proyecto.

Carpeta src/store/authentication/getters.ts: este archivo guarda las funciones encargadas de facilitar la lectura de datos al resto de elementos del cliente.

Carpeta src/store/authentication/index.ts: declaración de este store y de sus distintos archivos.

Carpeta src/store/authentication/mutations.ts: este archivo guarda las funciones encargadas de modificar las variables del store. Suele usarse para asignar los datos recibidos desde una función de actions a un variable del state.ts.

Carpeta src/store/authentication/getters.ts: este archivo guarda la declaración de las variables de un store y las inicializa al valor elegido.

Carpeta src/views: en esta carpeta se guardan las vistas del proyecto web, las cuales manejan el contenido que se visualiza en la web para cada ruta de acceso distinta.

Archivo src/app.vue: este es el componente principal del proyecto, ya que define la plantilla en la cual se dibuja la aplicación web en todo momento y configura su visualización.

Archivo src/main.ts: en cuanto este arco es el homólogo al app.js, del servidor, ya que se encarga del arranque y despliegue del proyecto.

Archivos babel.config.js, tsconfig.json y vue.config.js: son archivos de configuración de las tecnologías Babel, Typescript y Vue, que personalizan el funcionamiento de cada una de estas tecnologías en el proyecto Node JS del lado cliente.

8. PROCESO DEL DESARROLLO

En este apartado se tratará las directrices, metodología y pasos con los que se trabajó en el desarrollo del proyecto, las razones por las que se eligió cada parámetro de trabajo, así como afecto cada uno de estos en el transcurso del desarrollo de la app final.

8.1 METODOLOGÍA ÁGIL Y SCRUM

Debido a las características del proyecto y mi interés personal en aportarle un de desarrollo de software ágil, la metodología seguida en el proyecto es Scrum [62].

Scrum es un marco de trabajo específico dentro de las metodologías ágiles, por lo que sigue los principios de adaptabilidad, flexibilidad y entrega incremental, propios de esta ideología, sumando además componentes de trabajo propios, como las tareas específicas (User Stories) o los ciclos de desarrollos fijos (Sprints).

En concreto en el proyecto, se funcionó con Scrum, debido a la facilidad de modificación y cambio de funcionalidades, ya que el tratamiento de estas como User Stories (US) permitía un manejo fácil de los desarrollos, dando una planificación de trabajo flexible y con versiones funcionales de la aplicación desde los primeros desarrollos. Además, la división del trabajo en Sprints concretos de 4 semanas, producían un fácil y fluido control del desarrollo de la aplicación. Por esto la clave permaneció en el sencillo tamaño de las US, a la hora de realizar estimaciones de trabajo precisas que se pudiesen ubicar de manera efectiva en periodos de entrega iguales. Aportando así dos ventajas que me gustaría destacar, la facilidad de solventar imprevistos, y la capacidad de ofrecer continuamente una versión funcional de la aplicación, esta combinación me permitía valorar continuamente el portal, en busca de la invención e implementación de nuevos requerimientos que aumentasen la calidad del producto final.

8.2 HITOS ALCANZADOS DURANTE EL DESARROLLO

Antes de nada, me gustaría destacar que el registro de tareas o User Stories, que representaron hitos en el desarrollo, se hizo mediante la tecnología de gestión de versiones Git, ya que aprovechando elementos de distribución y subida de datos, como ramas o commits me permitió tener un muy buen control y reparto del trabajo, a la par que un guardado de seguridad útil ante posibles incidentes en máquinas o equipos. Por ello en el proyecto Git se planificó una estructura lineal, en la cual existe una rama principal, troncal que comunica el resto de las ramas, presentando por tanto en cabeza la versión estable de la aplicación, y un conjunto de ramas aisladas solo comunicadas con la principal, dedicada cada una al desarrollo de una US concreta.

A continuación, se mostrará un grafismo de la línea de trabajo que ha seguido el proyecto, señalando en cada Sprint mensual las tareas realizadas, que se procederán a explicar inmediatamente después.

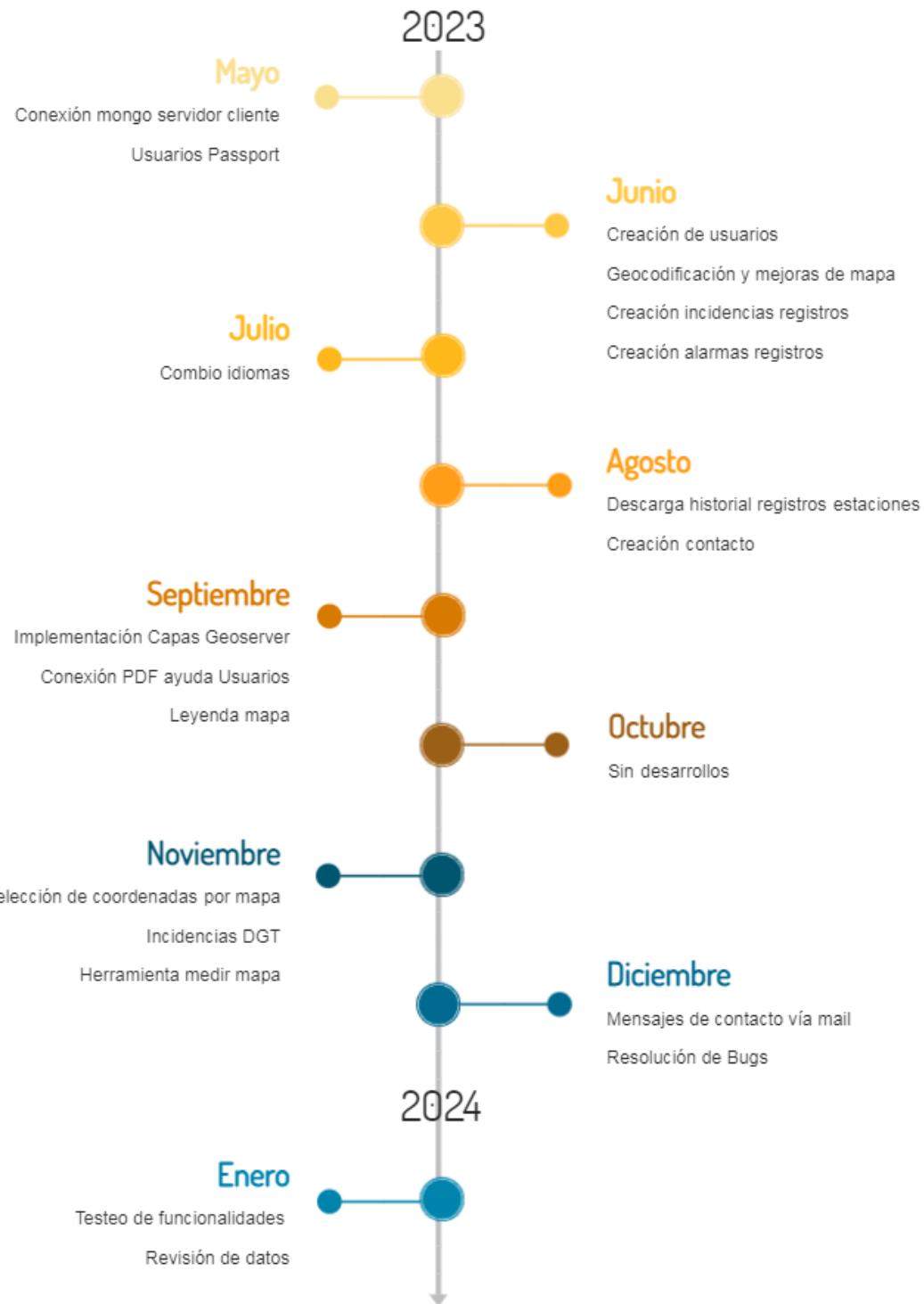


Ilustración 77. Línea de tiempo desarrollo del portal web

➤ Conexión mongo servidor cliente

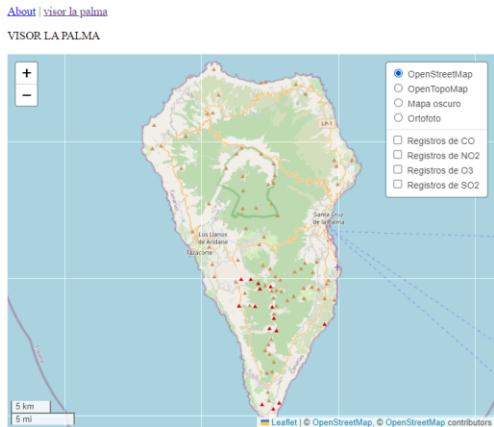


Ilustración 78. Versión final de US conexión mongo servidor cliente. Página principal aplicación

Se crea los cimientos de los proyectos Node JS, dividiéndolos en cliente y servidor, instalación de tecnologías principales como Express en Back-End o Vue en Front-End, así como resto de extensiones fundamentales para el desarrollo del proyecto, tales como Mongoose y Leaflet.

En este desarrollo se realizó la conexión entre servidor y base de datos, construcción de primeros modelos de datos, en cliente, la creación de almacenes de datos Vuex, aparición de los primeros componentes como el del mapa, pero sobre todo el inicio de la interacción entre las dos partes del proyecto, dando por primera vez intercambios de datos entre API y cliente web.

Cabe destacar también la creación aquí, de las transformaciones Pentaho Data Integration que proporcionan los datos de estaciones de calidad del aire y terremotos a la aplicación.

➤ Usuarios Passport

Este paso consistió en la creación del sistema de usuarios mediante la tecnología Passport [63]. Sin embargo, si bien si se terminó de desarrollar como se esperaba, tras valorar y explorar estrategias similares se optó por desechar este progreso, y empezar otro camino para la creación del sistema de usuarios. No obstante, se llegaron a fraguar aquí páginas como la de registro y acceso, modelos de usuarios, definición de roles, etcétera, que permanecerían como base en el siguiente desarrollo. Además, es un claro ejemplo del aprovechamiento de la metodología de trabajo empleada y su capacidad de plantear cambios y encarar imprevistos.

➤ Creación de usuarios

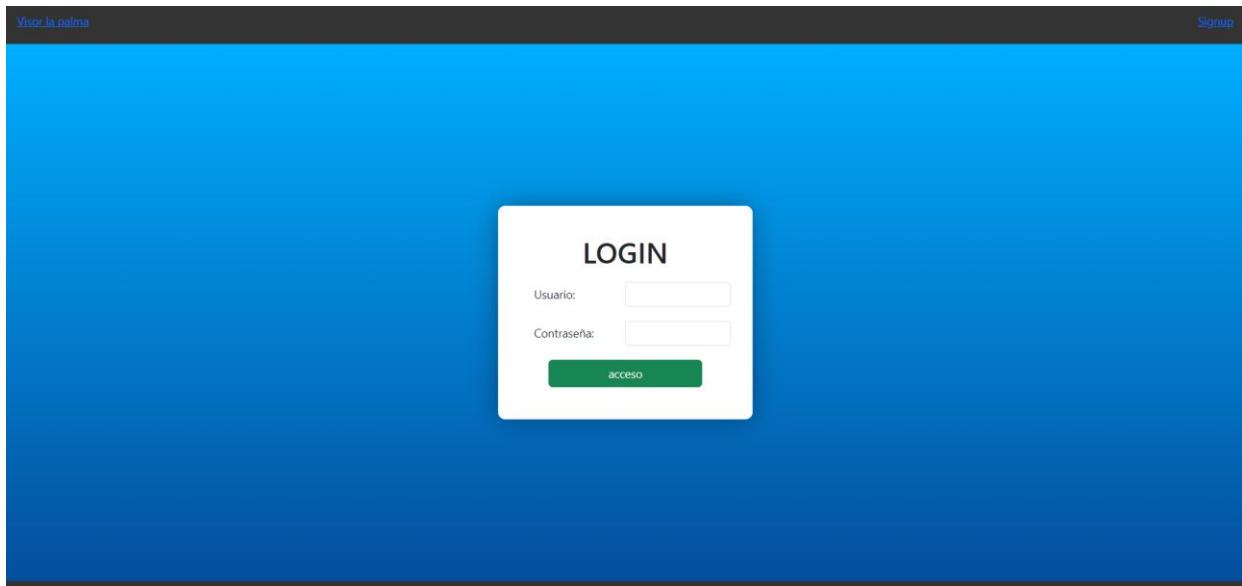


Ilustración 79. Versión final de US creación de usuarios. Página de acceso aplicación

Finalmente, la creación del sistema de usuarios se dio empleando la estructura ofrecida por Json Web Token, que permitía una gestión simple y rápida de los controles de acceso de usuarios, así como una fácil implementación dentro del proyecto. Esta incorporación llevada a cabo en el servidor del proyecto, también significó cambios en el cliente, la suma de nuevas páginas (registro, acceso, perfil de usuario y control de super administrador), las primeras mejoras en el diseño de la interfaz del portal, etcétera.

Cabe destacar, las primeras entradas de datos desde cliente al proyecto, apareciendo por tanto la validación de los primeros datos, y los mensajes de control, confirmación y error, intercambiados entre cliente y API.

Además, para terminar la gestión de este sistema se definió la jerarquía de roles, y con ella las funcionalidades proporcionadas por el menú de control de superadministrador, el cambio de roles, y la eliminación de cuentas de usuario.

➤ Geocodificación y mejoras de mapa



Ilustración 80. Versión final de US geocodificación y mejorar de mapa. Página principal aplicación

Tras la creación de unas bases de desarrollo en el proyecto, tales como el sistema de usuarios, y las comunicaciones base de datos, API, servidor. Se empiezan a desarrollar el resto de las funcionalidades que completan y caracterizarán a la aplicación final. En este punto el proyecto en lado servidor, paso a un desarrollo en segundo plano, que se centró en la apertura de nuevas rutas y controladores que permitiesen la comunicación de datos de estaciones de calidad del aire al cliente, así como también la solución de errores de gestión de roles y usuarios, que faltaron de aplicar en el paso anterior.

Por su lado, el lado cliente, sufrió un buen avance, ya que se dio una gran mejora del estilo de la interfaz, definiendo varias características que permanecen hasta esta última versión, cabecera, diseño del cuerpo, pie de página, estilos de componentes, formularios, menús, ..., sin olvidar aportar valor al componente central de la aplicación, el visor web. El cual, gracias a la aparición del control de capas como herramienta de gestión de información, pudo desarrollar varias nuevas funcionalidades como el cambio de mapas base, el control de visualización de estaciones de claridad del aire y la aparición de eventos, permitiendo la comunicación reciproca entre ambos componentes. Además, también se le añadió la primera herramienta de mapa, la función de geocodificación.

➤ Creación incidencias registros



Ilustración 81. Versión final de US creación incidencias registros. Página principal aplicación, usuario registrado

En esta tarea se trata el desarrollo de los procesos de comunicación y trabajo de los registros de incidencias, destacando por un lado la transmisión de datos de incidencias y, por otro lado, la aparición de los procesos de manejo de estos registros, creación, eliminación, edición y validación de incidencias. Esto provoca claro en el Back-End del proyecto la creación de puntos de acceso que atiendan distintas peticiones, nuevos controladores de proceso, mientras que en el Front-End, se generan nuevas vistas que atiendan a la entrada de datos en distintas páginas y se arreglan los componentes mapa y control de capas, para la representación de los datos de incidencias.

Cabe destacar también la implementación de la tecnología Nodemailer, que abre la comunicación entre la aplicación y los usuarios vía correo electrónico.

Además, se añade en esta etapa la herramienta del visor de vista principal, permitiéndose así un mejor control de la presentación del mapa.

➤ Creación alarmas registros

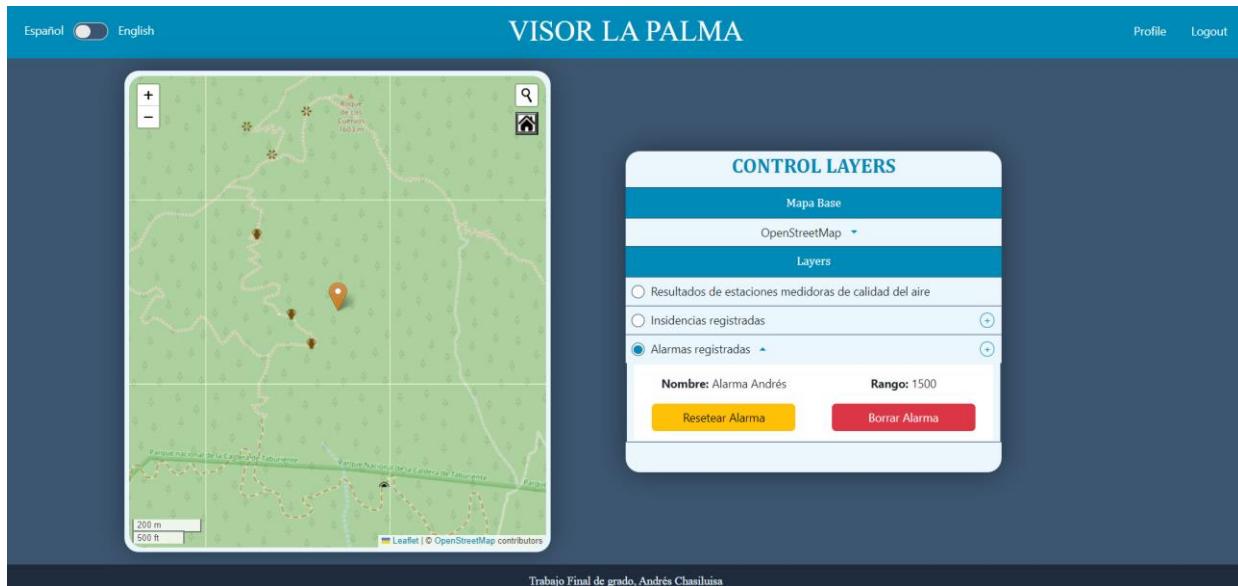


Ilustración 82. Versión final de US creación alarmas registros. Página principal aplicación, usuario registrado

En cuanto a este paso del proyecto, se realiza lo equivalente al anterior, pero en respectivo a los registros de alarmas. Se implementan en el lado servidor, las nuevas rutas y controladores necesarios, mientras que en el lado cliente se realiza lo propio mediante la habilitación de una nueva vista que representen las páginas de creación de alarmas, y la adaptación a estos desarrollos en los componentes mapa y control de capas. De esta forma, se desarrollan las funcionalidades de visualización, representación, creación y gestión de los registros de alarmas.

➤ Cambio Idiomas

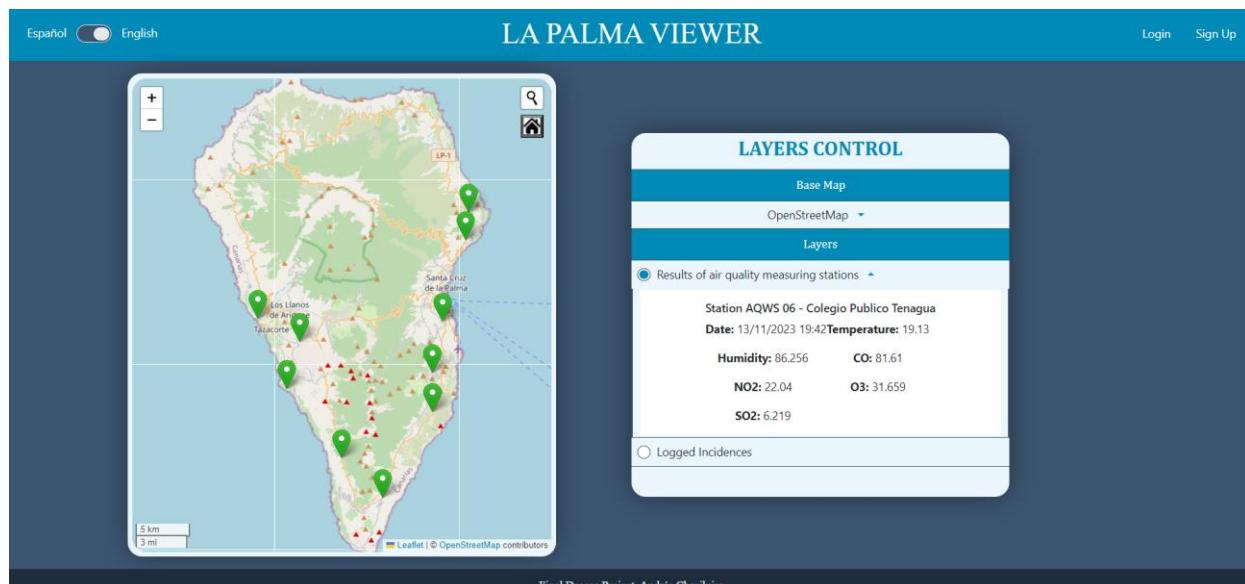


Ilustración 83. Versión final de US cambio idiomas. Página principal aplicación

Después de incorporar una gran parte del contenido de la aplicación, el desarrollo a partir de aquí se centra en modificaciones más específicas y concretas, por ejemplo en esta etapa del trabajo se centró los esfuerzos en la implementación de la tecnología de internalización i18n, así como también en la actualización en este sentido del modelo de datos de incidencias, para proporcionar así una vista total del portal en los idiomas de inglés y de castellano.

➤ Descarga historial registros estaciones



Ilustración 84. Versión final de US descarga historial registros estaciones. Página principal aplicación, usuario registrado

Para completar las funcionalidades ofrecidas para los datos de calidad del aire, en este punto del desarrollo, se crea la transformación PDI y modelos Mongo que proporcionan los datos históricos de calidad del aire a la aplicación, para finalmente preparar en el servidor una ruta de acceso a estos datos en formato de archivo CSV, y al cliente una entrada a usuarios registrados por la que puedan lanzar la petición de datos.

Por otro lado, se define la representación y estilos de los cuadros emergentes de información de registros, adoptando las tablas como formato de muestra de datos. También se desarrolló ese característico código de color en los cuadros de gases, que permiten una valoración rápida de la calidad del aire para el ser humano.

➤ Creación contacto

VISOR LA PALMA

Español English

Contacto

Nombre*

Apellido*

Mail*

Asunto*

Mensaje*

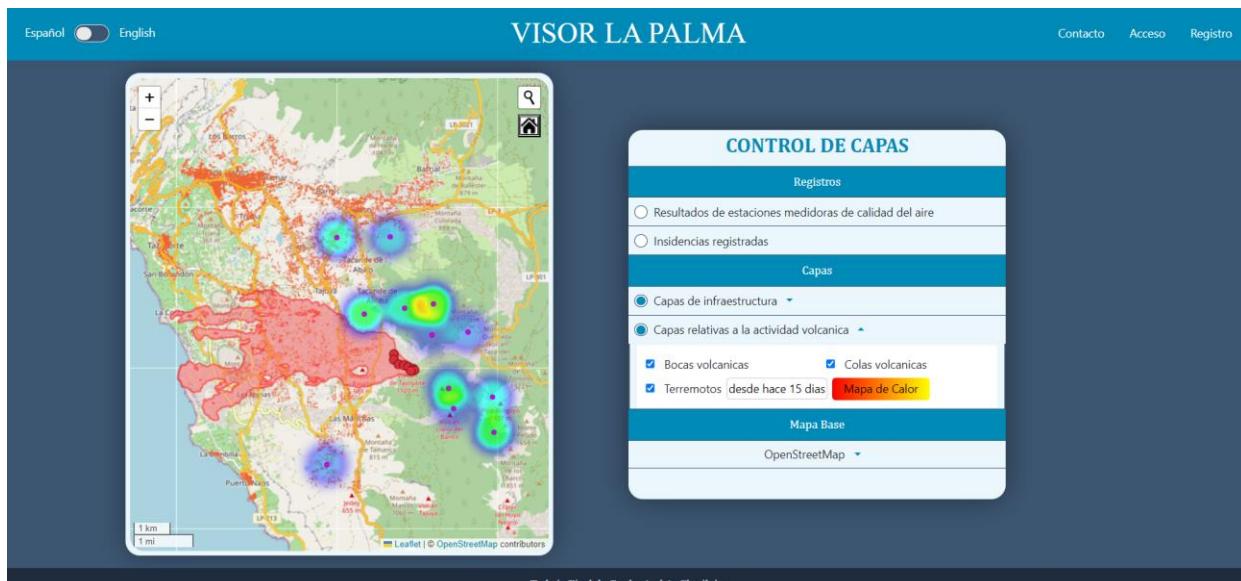
Enviar

Trabajo Final de Grado, Andrés Chaslinusa

Ilustración 85. Versión final de US creación contacto. Página de contacto aplicación

En este paso del proyecto se ocupa el desarrollo de las peticiones de contacto, añadiendo así esta funcionalidad a la aplicación, no obstante, en una primera versión, donde las peticiones de contacto se guardaban en una colección Mongo a espera de ser leídas por la administración, estrategia que se cambiara más adelante a la implementada en la versión final de la aplicación.

➤ Implementación Capas Geoserver



Trabajo Final de Grado, Andrés Chaslinusa

Ilustración 86. Versión final de US implementación capas Geoserver. Página principal aplicación

Ya instaladas las representaciones de registros en el cliente, se empieza la incorporación de nuevos datos de interés al proyecto. En primer lugar, se descargaron de las fuentes correspondientes las nuevas capas de datos, se les asigno un estilo y se subieron a Geoserver, para desde allí ser servidas mediante la comunicación WMS al portal web. Desde allí el componente mapa, aprovechará esta comunicación para representar los datos.

Por otro lado, el tratamiento de la información de terremotos, como se vio en su requisito, es distinta, ya que requiere comunicación entre la API y el cliente para el traslado de datos, y ya en cliente emplear la tecnología Leaflet-heat, para generar mapas de calor con estos registros.

Por último, cabe destacar, el cambio de orden en los elementos del componente de control de capas, pasando a ser ya el final, con la sección de mapas base debajo de las de capa y registros, mejorando así la presentación del componente ante la suma de más elementos.

➤ Conexión PDF ayuda Usuarios



Ilustración 87. Versión final de US conexión PDF ayuda usuarios. Página principal aplicación

Esta tarea sirvió para la suma de la funcionalidad de lectura de manual de uso, dando así los desarrollos en servidor y cliente necesarios para la puesta en marcha de este requisito. Además, se aprovechó la User Story para aplicar algunos arreglos en textos sin traducir, que estaban fuera de los ficheros de internalización del cliente.

➤ Leyenda mapa



Ilustración 88. Versión final de US leyenda mapa. Página principal aplicación

Aquí se volvió a terminar de actualizar algunas funcionalidades que se habían quedado atrás en el desarrollo, como los menús de control de súper administrador o el botón de vista general, de herramientas del mapa. Aparte se incorpora el desarrollo de la leyenda del mapa, con su gestión de simbología por rol de usuario.

➤ Selección de coordenadas por mapa

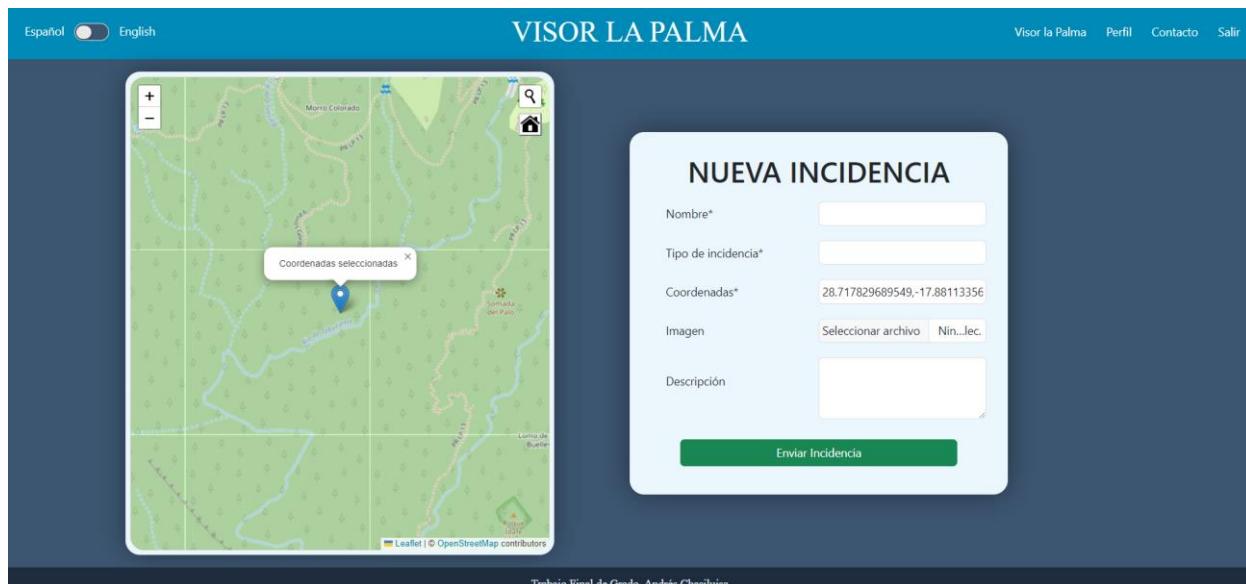


Ilustración 89. Versión final de US selección de coordenadas por mapa. Página de creación de incidencias aplicación

Aquí se volvió a corregir algunos parámetros de textos sin traducir, y destaco el desarrollo de la funcionalidad de selección de coordenadas por mapa auxiliar, en las páginas de creación de registros de incidencias y alarmas.

➤ Incidencias DGT

The screenshot shows the main interface of the 'VISOR LA PALMA' application. At the top, there are language switches for 'Español' and 'English', followed by the title 'VISOR LA PALMA'. On the right side, there are links for 'Perfil', 'Contacto', 'Manual de Uso', and 'Salir'. Below the title, there's a 'CONTROL DE CAPAS' (Control of Layers) sidebar with sections for 'Registros' (Registers), 'Alertas' (Alerts), 'Infraestructura' (Infrastructure), and 'Actividad volcánica' (Volcanic Activity). The 'Registros' section is currently selected, showing a list titled 'OTROS / LP-211 (3.8 - 4.68) (Incidencia sin editar, origen DGT)'. It contains two entries: 'Fecha' (Date) 30/11/2022 23:11 and 'Tipo' (Type) Otro. Below this is a 'Descripción' (Description) section with the text: 'DERRAMAMIENTO con circulación interrumpida en: - La carretera LP-211 a la altura de TODOQUE (SANTA CRUZ DE TENERIFE) desde el km 3.8 al km 4.68 sentido AMBOS SENTIDOS Advertencia: CORTE TOTAL'. A green 'Editar Incidencia' (Edit Incident) button is located at the bottom of this section. The main area of the screen displays a map of La Palma with several location markers and geographical labels like 'Tazacorte', 'La Laguna', 'Las Manchas', and 'Montaña de la Breña'. A scale bar indicates distances up to 1 km and 1 mi.

Ilustración 90. Versión final de US incidencias DGT. Página principal aplicación, usuario administrador registrado

Se volvió de cierta manera a desarrollos anteriores, ya que se volvió a necesitar de transformaciones Pentaho, para obtener y gestionar la información de incidencias proporcionadas por la DGT, se actualizó el modelo de datos para soportar esta nueva fuente de datos, así como se adaptó controladores y componentes del cliente, para el manejo de estas nuevas incidencias, dando así la correcta implementación de estos nuevos registros en el portal web.

Además, nuevamente se revisaron desarrollos anteriores en busca de posibles fallos, producidos por los continuos cambios implementados. Aquí se adaptó la eliminación de cuentas de usuario, para ampliar este proceso a la también eliminación de sus alarmas o de su participación en registros de incidencias.

➤ Herramienta medir mapa



Ilustración 91. Versión final de US herramienta medir mapa. Página principal aplicación, usuario registrado

Finalmente, este el ultimo desarrollo como US aplicado en el proyecto y consistió en el desarrollo de la herramienta de medición en los componentes de mapas para usuario ya registrados, tanto como el principal como el de ayuda en selección de coordenadas, así como también en pequeñas mejoras del estilo, como el cambio de concepto en los botones de herramientas del visor, y la incorporación de punteros especializados al pasar por entradas de funcionalidades, como en el enlace de manual de usuario.

En los siguientes apartados se trataron desarrollos no correspondientes a ser definidos como User Stories, ya que son complementarios a los ya mencionados. Sin embargo, también tienen su importancia, ya que fueron necesarios para garantizar el correcto funcionamiento de la aplicación.

➤ Mensajes de contacto vía mail

En este punto de la aplicación web, los mensajes de contacto se almacenaban por campos en una colección mongo, sin embargo, esta característica implicaba que la administración tuviese que revisar constantemente la base de datos para poder acceder al mensaje de contacto. Proceso el cual iba en contra del principio de accesibilidad y fácil manejo que siempre ha buscado el proyecto. Por esto se cambió el paradigma de comunicación, primero dotando al super administrador de un correo electrónico, que representase el medio de comunicación con la administración y luego se aprovechó la tecnología de Nodemailer, para a partir de los datos de

contacto introducidos por los usuario construir los mensajes que llegarían por mail al super administrador. Mejorando así la accesibilidad a los mensajes y la metodología de trabajo de esta funcionalidad.

➤ Resolución de bugs

Este paso es el último desarrollo dado en el proyecto ya que con la aplicación ya en una versión final se arregló algunos fallos de interacción que presentaban las funcionalidades, destacando la capacidad del super administrador de borrar su propia cuenta, lo cual no tiene sentido en el sistema de roles de la aplicación, por lo que se bloqueó esta opción desde servidor.

➤ Testeo de funcionalidades

Por último, para dar un repaso final de la aplicación dediqué tiempo a la prueba conjunta de las distintas funcionalidades del portal, en busca de posibles errores o puntos de mejora, en este paso pude investigar posibles nuevas líneas de mejora, que podrían enriquecer al proyecto en un futuro.

➤ Revisión de datos

Finalmente, para dar como acabado el desarrollo, revise archivos y datos del proyecto en busca de posibles residuos de antiguos implementaciones, como colecciones obsoletas en la base de datos, o archivos temporales que ahora carecían de interés.

9. INTERFAZ DE USUARIO

9.1 DISEÑO VISUAL

El diseño de la interfaz de la web es un componente muy importante dentro de la presentación del proyecto, ya que esta debe ser coherente con las estrategias y mecánicas de presentación elegidas, a la vez que debe facilitar el entendimiento de datos y facilidad de navegación dentro de las funcionalidades ofrecidas por el portal.

9.1.1 Gama de color

Parte fundamental del diseño de la aplicación se da por su elección de colores, los cuales siguen una gama cromática en la que destaca la tonalidad del color azul, ya que este es representativo en varias aplicaciones que cubren la información de la isla, siendo además empleados para escudos regionales o logotipos de instituciones propias como la del Cabildo de La Palma, entre otros símbolos, ya que al final es un color que se relaciona con el mar, parte destacada de la isla bonita.



Ilustración 92. Gama de color principal empleada en el estilo del portal

Como se puede observar la gama de color de la web representa distintas tonalidades:

Dark Color: se usó para la caracterización de bordes, permitiendo también la distinción de elementos y la sensación de profundidad. Además, destaca su aparición en el pie de página donde permite distinguir el final de la web sin resaltar ni quitar protagonismo al contenido de esta.

Secondary Color: este color, como indica su nombre se adoptó en el portal como característico de elementos secundarios, el fondo del cuerpo, bordes de tablas de datos, etc. Aportando a estos elementos una menor importancia visual que los caracterizados por el resto de las tonalidades de la gama

Primary Color: el color principal de la aplicación, implementado en títulos, cabeceras, botones y logotipo. Se trata del color que presenta mayor nivel de saturación de la gama, diseñado así para resaltar los elementos sobre el resto.

Light Color: finalmente esta última tonalidad de azul, muy cercana al blanco, se utilizó para el fondo de componentes, como puede verse en el control de capas, contenedor del mapa, menús, formularios, etc. Facilitando así la lectura de los contenidos de estos componentes.

White: finalmente este color completo la gama empleada, siendo elegido para el fondo de cuadros desplegables, contenedores de información e inputs de entrada de datos, así como color de textos dentro de elementos con tonos llamativos como los del **Dark Color** y **Primary Color**, donde el color de texto por defecto, el negro, no permitía una fácil lectura.

9.1.2 Tipografía

La tipografía empleada en la aplicación es realmente muy sencilla, ya que se basa en la participación de dos familias específicas, pertenecientes al entorno de Microsoft:

Segoe UI: se trata de un estilo de fuente derivado de Segoe, que es muy utilizado en las interfaces de usuario. En lo relativo al proyecto, se trató como una fuente predefinida por defecto, adoptado por la mayoría de los textos, como subtítulos, datos de registros, etiquetas, etc. Aprovechando también sus versiones en **Bold**, **Entre tachado** y **Subrayado**, para la distinción de los distintos elementos de la interfaz [64].

Cambria: este tipo de letra es usualmente empleada por sus propiedades. Una lectura agradable y buena estética, aprovechándola para la distinción de títulos y textos relevantes dentro de la aplicación. En este caso, se usaron sus versiones Estándar, *Italic* y **Bold**, ya que era suficiente a la hora de categorizar los textos que estas cubren en la app.

Por último, los tamaños presentados en los textos son los proporcionados por las etiquetas de títulos de HTML <h1, h2, h3, ...> o de párrafo <p>, con las que se estableció la jerarquía de importancia en la información textual del portal.

9.1.3 Iconografía de la aplicación

En el portal web se aprovechan varias imágenes, que sirven como comunicación a los usuarios para acceder a funcionalidades de herramientas, peticiones de datos, o simplemente como iconos de la aplicación [65], a continuación, se mostrara una tabla con los iconos empleados y su papel en la aplicación web.

Símbolo	Función
	Este ícono se encarga de representar la aplicación, como logotipo, expresa mediante la silueta de la isla y un pin de localización, el carácter de la aplicación de comunicar información georreferenciada de La Palma.
	Esta imagen que representa la bajada de archivos de la nube simboliza el acceso a la funcionalidad de descarga de datos históricos, ofrecida en el cuadro de datos de registros de calidad del aire.
	Pasando a iconos de herramientas del visor, este ícono de lupa caracteriza al botón que despliega la entrada de búsqueda de localizaciones, mediante la función de geocodificación ofrecida por el visor web.
	Estos dos símbolos representan de forma distinta un mapa desplegándose y si utiliza para representar el estado de la funcionalidad de leyenda, ya que, si se muestra el primero de borde oscuro, indica que la función no está desplegada, y por tanto que si se pulsa sobre él se desplegará, mientras que el segundo de contenido de color negro representa el estado contrario, la visualización de la leyenda y, por tanto, su desactivación si este se vuelve a pulsar.
	Por otro lado, se sigue una estrategia parecida al de los símbolos anteriores para esta caso de la representación de la herramienta de medición, ya que se utiliza la primera imagen de regla de borde negro, para representar que la herramienta no está en funcionamiento, para que así si el usuario pulsa sobre esta, active por un lado la funcionalidad, y por otro, se cambie el símbolo del botón al de la regla de color negro, indicando ahora que la herramienta está activa, y que se puede desactivar al hacerle click encima.
	Por último, este símbolo de casa indica en el mapa, el regreso a la vista inicial definida por defecto en la aplicación, que permite la visualización de toda la isla. Aportando así al usuario un fácil y rápido control del zoom en cualquier situación de la escala.

Tabla 6. Iconos de la aplicación

Otra parte destacable en el uso de tipografía dentro de la aplicación es la disposición de imágenes auxiliares [66] empleadas en la visualización de incidencias sin imagen aportada, por esto se continuará exponiendo cada una de ellas y se explicará cómo se establece relación directa con la variedad de tipos que mantiene el modelo de registros de incidencias.

Imagen auxiliar	Asignación de tipo de incidencia
	Se representa el choque de dos vehículos, simbolizado así a las incidencias causadas por Accidente de Tráfico (ACC).
	La imagen enseña la caída de rocas por un desfiladero, visualizando así las incidencias de Derrumbe (DER).
	El dibujo de árboles ardiendo pretende transmitir, las incidencias de tipo Incendio (INC).
	Aquí se comunica la presencia de anegación de agua en localizaciones no preparadas, relacionando así el tipo Inundación (INU).
	Este símbolo de arena en suspensión representa las posibles llegadas de Calima (CAL) a la isla.
	Esta representación de un volcán en proceso de expulsión de lava representa las incidencias de Erupción (ERU).
	El humo de la imagen simboliza las incidencias de tipo de Escape de Gases (ESC) posibles en la isla de La Palma.

	Como territorio con gran presencia de costa, el dibujo de esta ola de gran tamaño representa las incidencias de Alto Oleaje (OLA).
	Se representa a un técnico de la construcción moviendo tierra, como simbología de Obras (OBR) en espacio público.
	Por último, esta comunicación de atención o alerta comunica las incidencias que puedan ser de un tema distinto, Otro (OTR).

Tabla 7. Imágenes auxiliares de incidencias según su tipo

9.1.4 Simbología del mapa

Los datos representados en el visor del portal adoptan una simbología específica con la intención de lograr una correcta visión de estos en conjunto, a la par que transmitir información esencial de cada conjunto de datos.

9.1.4.1 Representación de registros

A continuación, se repasarán los distintos símbolos de mapa asignados a geometrías puntuales propias de registros [67], o producidas por funcionalidades de la aplicación.

Símbolo	Dato asociado
	Localización obtenida de geocodificación
	Localización de estaciones de calidad del aire
	Localización de incidencias validadas
	Localización de incidencias sin validar

	Localización de incidencias sin validar con origen de la DGT
	Localización de Alarmas inactivas
	Localización de alarmas activadas
	Localización de Terremotos

Tabla 8. Simbología de registros

Como se puede ver los símbolos no pretenden comunicar ningún dato intrínseco a su diseño, simplemente señalan en el visor la localización de los datos, y quizás mantener una relación entre ellos a través de aplicar colores similares para datos equivalentes, un ejemplo de esto son el conjunto de simbología de incidencias validadas y sin validar o las alarmas inactivas o activas.

9.1.4.2 Estilo de capas ofrecidas por Geoserver

Por su lado, estas capas de información si constan con un estilo y categorización específicos, ya que su correcta visualización en el mapa dependía de esto. En los siguientes apartados se estudiará cada estilo asignado a capas.

Carreteras principales

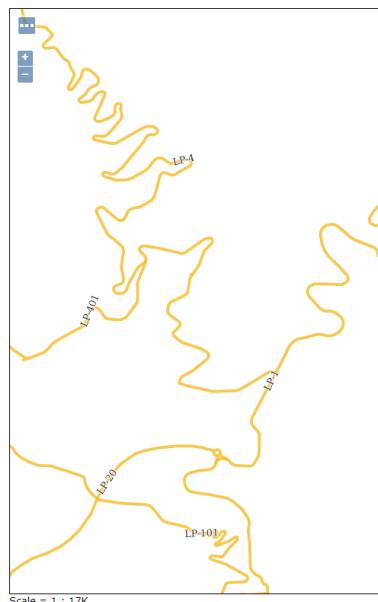


Ilustración 93. Captura estilo de carreteras principales

El estilo diseñado para esta capa de carreteras consiste en aplicar a la geometría lineal de la capa, un tono amarillo fuerte característico de muchos estilos de mapa en elementos de este tipo. Además, se añadieron etiquetas del código de carreteras, permitiendo así una mejor identificación de cada elemento, mediante un color de texto negro con un fino halo blanco.

 Edificios

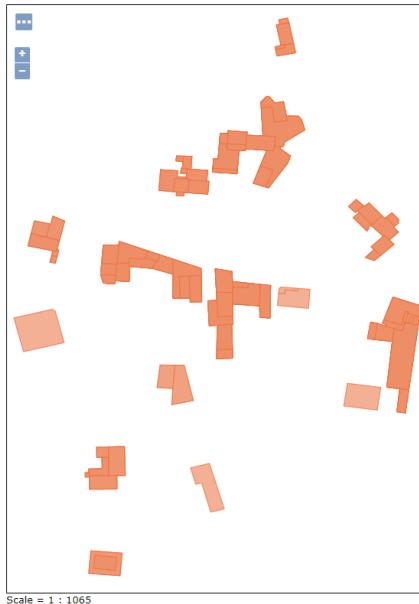


Ilustración 94. Captura estilo de edificios

En cuanto al estilo construido para esta capa, se caracteriza por dibujar a los polígonos, con un relleno naranja transparente y un contorno también naranja, más saturado, que permite distinguir la forma de las geometrías a la vez que observar los elementos que se puedan mostrar por debajo, mapa base, otras capas, etcétera.

Parcelas



Ilustración 95. Captura estilo parcelas

Las parcelas son los elementos que se visualizan con un mínimo de escala más grande, ya que por sus características es la capa de información más baja en la jerarquía de datos, por ello, se le da unas tonalidades azuladas sin mucha saturación, con transparencias en el relleno, pero más profundidad de color en los bordes, consiguiendo así la distinción de polígonos. Cabe destacar también la presencia de etiquetas que señalan la referencia catastral de cada parcela bajo un color oscuro.

Coladas volcánicas

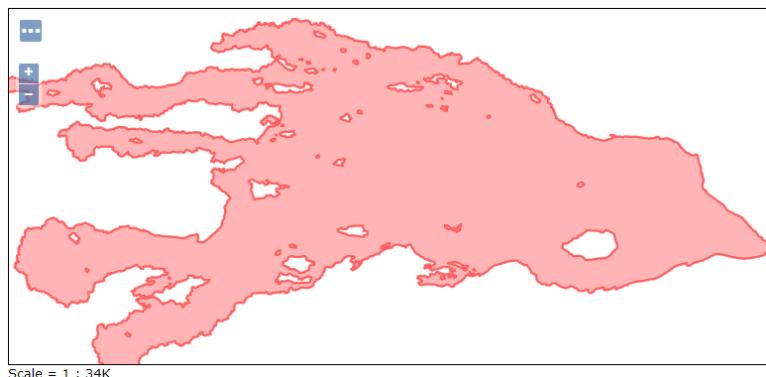


Ilustración 96. Captura estilo colada volcánica

El estilo de esta capa sigue un color rojo, asociado al fuego y lava propio de estos elementos, y nuevamente como polígono adquiere un relleno con transparencia que permite ver si se necesita la información aportada por el mapa base u otros elementos.

 Bocas eruptivas

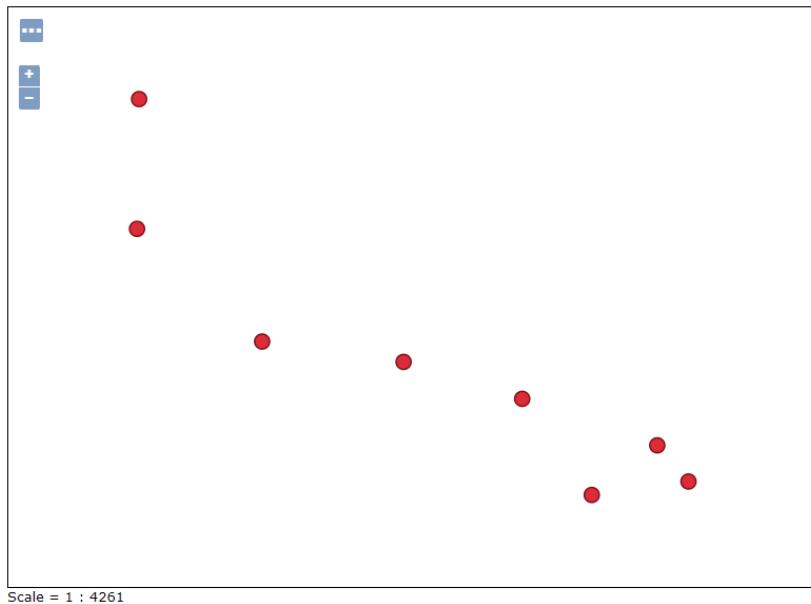


Ilustración 97. Captura estilo bocas eruptivas

Por último, el estilo de esta capa de puntos trata un tono rojo intenso tanto en relleno y borde, que hace destacar estos datos sobre el resto, además presenta un tamaño en radio de punto grande que le da todavía más relevancia dentro de la jerarquía visual.

9.2 ARQUITECTURA DE LA INTERFAZ

Las siguientes Imágenes señalan los puntos que componen la estructura de la web:

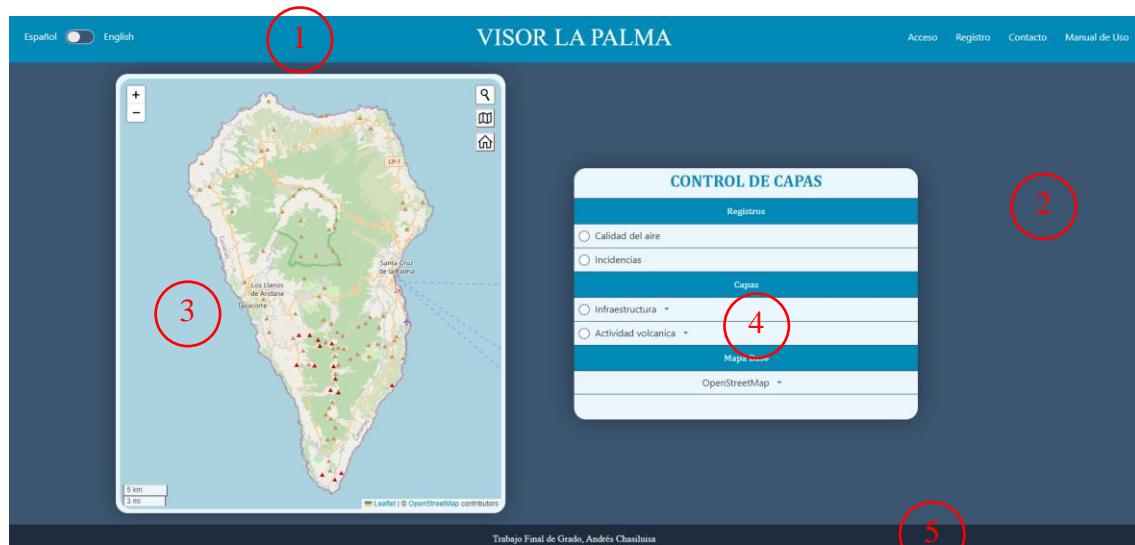


Ilustración 98. Estructura página principal aplicación

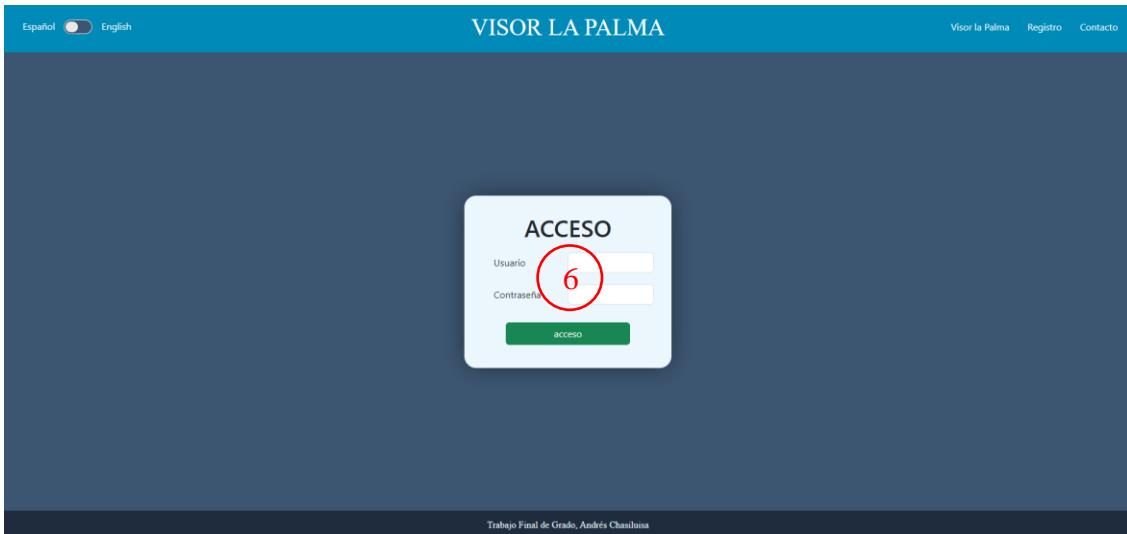


Ilustración 99. Estructura página de acceso aplicación

Como se ha podido ver en las ilustraciones la estructura del portal está definida por tres partes, En la parte superior de la web, se puede ver la cabecera 1, justo debajo, y con el tamaño más grande dentro de la aplicación y conteniendo el resto de los componentes, se sitúa, el cuerpo de la web 2, por último, al final de la interfaz está el pie de página 5. Esta estructura es usual en muchas páginas web, ya que permite una comunicación rápida e intuitiva a los usuarios.

En cuanto al contenido de la aplicación este se dispone todo en el cuerpo 2, y dentro se divide como en máximo dos elementos principales, siendo en este caso, el primero, dispuesto a la izquierda de la web, correspondiente a un visor o mapa y el segundo 3, dispuesto a la derecha de la web, correspondiente a un menú o formulario 4, ejemplo de esto es la página principal del proyecto, plasmada en la primera imagen. Por otro lado, existe también el caso de que se muestre solo un elemento en la sección de cuerpo 2, como se ve en la segunda imagen del apartado, siendo este situado en el centro y empleado como menú o formulario 6.

9.3 COMPONENTES DE LA INTERFAZ

Cabe destacar en esta sección de la memoria, la utilización de la librería de estilos de Bootstrap, la cual permitió dotar de estilos a muchos de los elementos mostrados en la web, tales como botones, celdas de tablas, contenedores de datos, etcétera, aportando así un estilo sencillo e intuitivo a estos componentes.

En este apartado se repasarán la estructura interna de los componentes, así como los estilos aplicados a cada uno diseñados según su contenido y función dentro de la página.

 Cabecera



Ilustración 100. Captura de componente cabecera

Este elemento se puede dividir en tres partes distintas, la primera la relativa al control de cambio de idioma, situado a la izquierda. De este control destaca el uso de un botón de estilo moderno, popular en interfaces de aplicaciones móviles, que señala según su posición el idioma seleccionado en el portal.

En el centro del componente se puede ver el título del portal, en un mayor tamaño para así destacar ante el resto de los elementos. Cabe destacar también el uso del color blanco en la fuente para permitir una mejor capacidad lectura respecto el fondo.

Por último, en el lado derecho se muestra la barra de navegación que permite a través de sus enlaces desplazarse entre las diferentes páginas del portal. Esta presenta textos variables según la página en la que se encuentre el usuario y si está registrado o no.

 Mapas

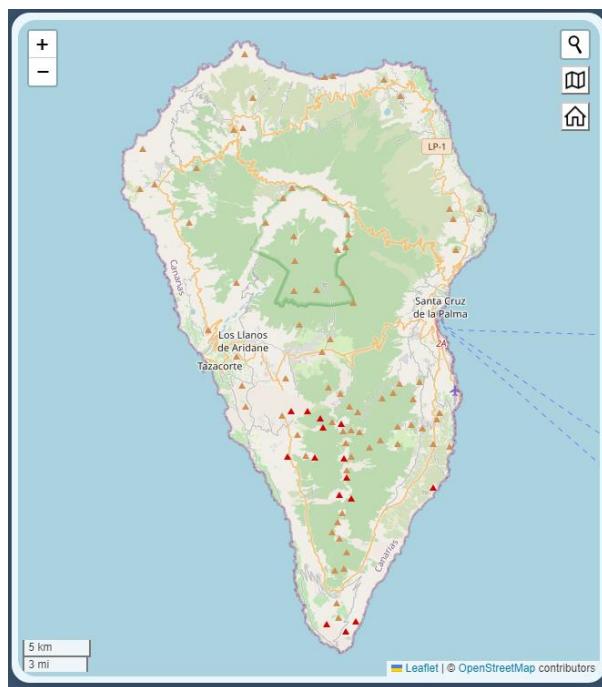


Ilustración 101. Captura de componente mapa

Este elemento se repite en varias páginas de la web, mostrando además funcionalidades distintas según ubicación y registro. Sin embargo, su estructurara permanece siempre igual, destacando

primero un contenedor que permite delimitar la presencia del visor, y separarlo del fondo, visualizándose así un fino contorno.

Ya dentro del visor, se observa en la esquina superior izquierda los controles básicos de zoom, y justo enfrente, arriba a la derecha, la barra de herramientas del mapa. Por otro lado, abajo en la esquina inferior izquierda, está la escala del mapa, para que en la otra esquina se vea con claridad el organismo creador del mapa base. Aparte cabe destacar, que los menús de herramientas se dibujan siempre justo a la izquierda de la barra de herramientas.

Panel de control

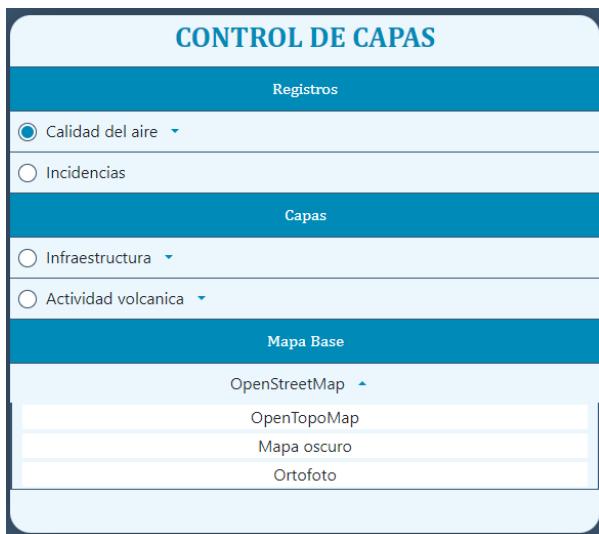


Ilustración 102. Captura de componente panel de control

En este diseño se ha buscado la estructuración de datos, empleando una forma de filas apiladas, en las que se aprovecha el color principal para destacar título, encabezados y botones, mientras que el color claro para el resto de celdas, construyendo así la jerarquía visual del componente, además como ya se ha visto en otras imágenes y se observa en esta ilustración los elementos desplegables, se presentan en cuadros ligeramente menos anchos que el resto de filas a la vez que presentan un cambio de color de fondo a uno más claro, produciendo así la sensación de despliegue.

A partir de aquí se ve como en los componentes destaca un estilo de esquina redondeada, tanto en el componente como en los checkboxes principales, ya que así a parte de dar estética al diseño, el uso de elementos redondeados junto con la selección de colores de colores transmite la sensación de un diseño armonioso.

⊕ Formularios



REGISTRO

Nombre*

Apellido*

Municipio

Usuario*

Mail*

Contraseña*

registro

This image shows a registration form titled "REGISTRO". It consists of six input fields: "Nombre*", "Apellido*", "Municipio", "Usuario*", "Mail*", and "Contraseña*". Below the input fields is a green button labeled "registro". The entire form has rounded corners and a light blue background.

Ilustración 103. Captura de componente formulario de registro

Aquí se mantiene el criterio de redondeo de esquinas, destacando dentro del contenedor una disposición de flujo tipo columna, situándose arriba el título del formulario, en las filas del medio, se realiza una división del espacio en dos partes, siendo la parte de la izquierda dedicada a los nombres de los parámetros que se deben insertar, y en la parte derecha los cuadros de entrada de datos. Finalmente, debajo en las últimas filas se sitúan los botones de acción, ya sea el principal de envío de formulario con fondo verde, o botones de vuelta regreso a un estado anterior, o eliminación de datos, con un fondo rojo.

⊕ Menús de control



user1

Nombre: Andrés Apellido: Chasiluisa
Municipio: El Paso Mail: as.chasiluisa@alumnos.upm.es
Rol: user

cambiar rol user+ volver

This image shows a user profile page titled "user1". It displays the user's information: Nombre: Andrés, Apellido: Chasiluisa, Municipio: El Paso, Mail: as.chasiluisa@alumnos.upm.es, and Rol: user. Below this information are two buttons: "cambiar rol" and "user+", and a large red "volver" button at the bottom.

Ilustración 104. Captura de componente menú de control de super administrador

Los menús de control siguen una estructura menos fija, ya que unen la capacidad de mostrar datos a la vez que funciones de tratamiento de estos, ejemplo de estos son el menú de super administrador, y el menú de alarmas. Como se ve no tienen un título específico, ya que la figura

llamativa de los menús se aprovecha para ubicar a los usuarios sobre qué datos manejan, después la separación del contenido es en dos columnas de datos, dejando el espacio izquierdo de cada una de estas para el nombre del campo, y el derecho para el valor de este. En el caso concreto de la ilustración se separa el dato de rol y se sitúa en una columna central, ya que este es el único dato que el super administrador puede transformar.

Por otro lado, debajo de los cuadros de datos se sitúan, los accesos a funcionalidades, los cuales se disponen de igual manera en dos columnas dejando una para cada elemento, sin embargo, si el número no es par, el valor sobrante como en el ejemplo, el botón de volver, se coloca en la columna central del menú ocupando los espacios sobrantes. A parte en esta sección de los menús se da un detalle extra y es que si la alguna de las funcionalidades ofrecidas no es ni de eliminación o vuelta (color rojo), o envío de dato (color verde), se le caracteriza con un color amarillo intenso, distinguiendo así del resto de código de color.

Cuadros de datos



Estación AQWS 07 - Colegio Publico San Antonio		
Fecha	Temperatura	Humedad
02/11/2023 08:42	22.17 °C	71.046 %
Mediciones en $\mu\text{g}/\text{m}^3$		
CO ⓘ	NO2 ⓘ	O3 ⓘ
97.099	22.53	44.13
SO2 ⓘ		6.239

Ilustración 105. Captura de componente cuadro de datos calidad del aire

Estos cuadros son propios del panel de control en la comunicación de datos de registros de calidad del aire y de incidencias, no son fijos ya que se dan tras la interacción con el visor. Su propósito principal es la comunicación de datos, aunque para usuarios registrados según el rol, también pueden ser lugar de aparición de accesos a funcionalidades, botones o iconos, como el visto en el ejemplo.

Están compuestos por tablas de datos dibujadas en un borde azul oscuro fino, con títulos de campos en negrita y en **Primary Color**. Además, pueden tener características de visualización extras como aquí en el cambio de tonos de fondo según la valoración de calidad del aire por gas.

Submenús de control

- Bocas eruptivas
- Coladas volcánicas
- Terremotos desde hace 3 días
- Mapa de Calor

Ilustración 106, Captura de componente submenú de control capas de actividad volcánica.

A diferencia de los menús de control estos solo se ocupan de brindar funcionalidades al usuario, en este caso de visualización de datos sobre el visor, son propios de los desplegables de las secciones de actividad volcánica e infraestructura. Su estructura es de doble columna dejando el espacio izquierdo para el input de control y el lado derecho para el nombre de la capa. Sin embargo, aquí en ejemplo, se da un caso especial, ya que la capa de terremotos necesita importación de parámetros, por lo que estos nuevos inputs se dibujan justo a la derecha de la celda regular.

9.4 INTERACTIVIDAD Y DINAMISMO

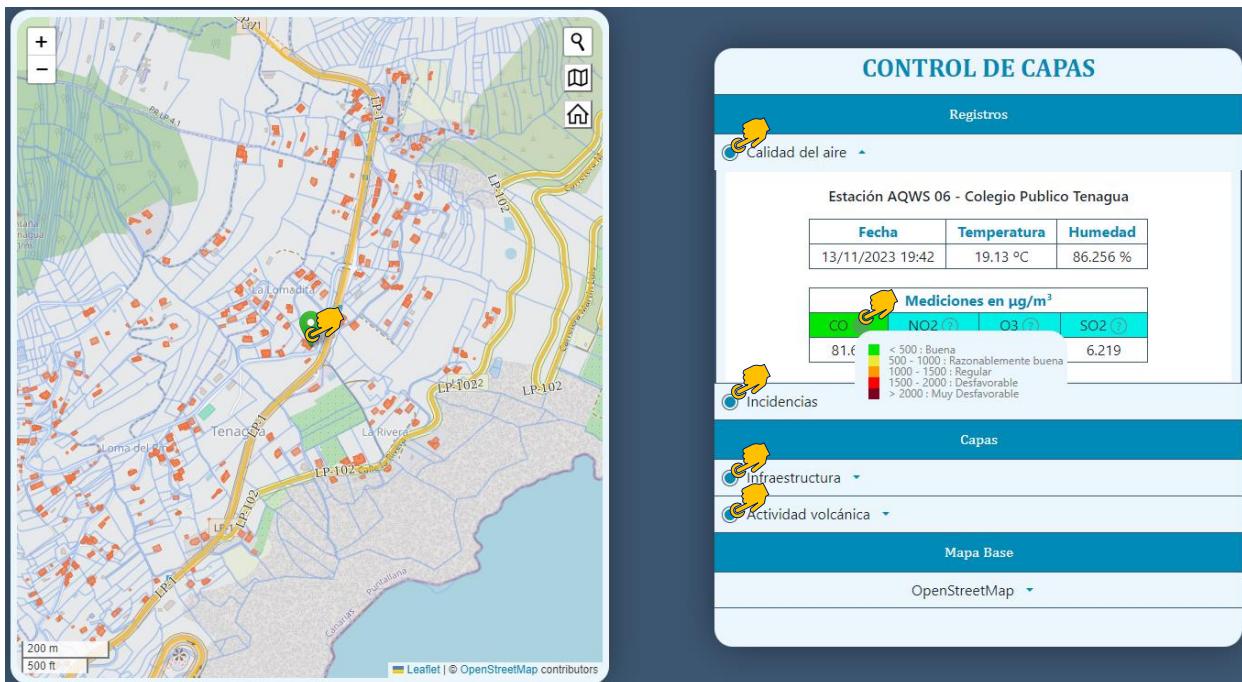


Ilustración 107. Ejemplo de Interactividad componentes página general aplicación

La interactividad y dinamismo es un punto central en el diseño de la aplicación, bajo mi criterio cuando se aporta información geoespacial mediante un visor de estas características, se deben ofrecer el máximo número de funciones de gestión de información, control de zoom,

representación de datos, etc., para capacitar a los usuarios de un fácil manejo y personalización de la información, adaptando así la información a cada usuario. Esto al final solo se puede conseguir con la implementación de estrategias de comunicación interactivas en componentes dinámicos, tal y como se ha ido viendo en las funcionalidades desarrolladas expuestas en este documento.

Además, debido a la gran cantidad de funcionalidades que realizan actualizaciones a valores dentro de la base de datos, se da siempre después de alguna operación de este tipo o bien un reinicio (o “reseteo”) de la información o bien una repetición de solicitud de datos, para así mostrar siempre solo datos actualizados en la aplicación.

10. RESULTADOS Y LOGROS

Finalmente, se repasan los puntos finales del documento, momento para realizar la valoración del trabajo y realizar entrega de los componentes que construyen el proyecto.

10.1 REVISIÓN DE OBJETIVOS

Con el trabajo realizado, y con una versión estable de la aplicación en plena capacidad de funcionamiento, se puede valorar si los objetivos y expectativas dispuestas en el proyecto han sido alcanzados o al contrario quedan por desarrollar.

- ✓ **Creación de sistema de usuarios:** el primer objetivo que se planteó en desarrollo es básico para el funcionamiento de la web. Se han explorado varios caminos para alcanzarlo y cumplir con la división de usuarios en roles, la aportación de funciones de autogestión de usuarios y la seguridad de acceso y se ha así logrado el objetivo planteado.
- ✓ **Presentar datos actualizados:** la selección de fuentes de datos confiables que además presentan una alta periodicidad de actualización, junto con el desarrollo de criterios de extracción de datos y transformación, convierten los datos mostrados por el portal en el resultado de un gran número de procedimientos que verifican y velan por la utilidad y actualidad de la información del proyecto.
- ✓ **Mostrar una variedad adecuada de información:** el proyecto cuenta con información geográfica de todo tipo, desde capas de carreteras, hasta datos de calidad del aire, la variedad de información es un eje central en el planteamiento de la aplicación. Por esta razón, se puede garantizar que la información mostrada (ya sea mediante registros, capas, o mapas base), fue cuidadosamente seleccionada ante la previsión de posibles necesidades del usuario. Destaca también la orientación de los desarrollos en este aspecto en base a una alta escalabilidad, facilitando la incorporación o eliminación de datos si así se requiere.
- ✓ **Interacción del usuario con los datos:** Las estrategias de creación de incidencias, sistema de alarmas, comunicaciones vía mail, descarga de datos, permiten al usuario una alta conectividad e impacto en los datos del proyecto, cumpliéndose así el objetivo marcado.
- ✓ **Buscar una alta accesibilidad:** La web presenta varios mecanismos orientados a dar acceso al usuario a los contenidos y funcionalidades ofrecidos. Este acceso se facilita mediante el cambio de idiomas íntegro, facilidad de controles o el diseño intuitivo del portal. Sin embargo, existen puntos que se pueden mejorar; en particular, el desarrollo de simbología específica, ventanas emergentes que expliquen funcionalidades, etc.. No

obstante, el desarrollo realizado abre con seguridad la aplicación y la situación de la isla de La Palma a una gran cantidad de público.

- ✓ **Interfaz dinámica y sencilla:** Como se ha visto en el documento, el dinamismo presentado por los componentes de la aplicación es una directriz continua, el comportamiento y mecánicas de los elementos del portal han sido un punto central del desarrollo. Por esta razón, se ha obtenido un producto final atrayente y lleno de oportunidades, capaz de adaptarse a todo tipo de usuarios.

10.2 ENTREGABLES

Como se vio en la Sección 7 (Arquitectura del sistema), el proyecto está compuesto por tres bloques principales, de los cuales se aportarán en cada caso entregables que expresen los desarrollos aquí tomados.

- **Adquisición de datos:** carpeta zip con transformación y trabajos desarrolladas en Pentaho Data Integration (véase Anexo 2).
- **Almacenamiento de datos:** Archivos de recreación de los contenedores empleados en el proyecto (véase Anexo 3).
- **Cliente servidor:** repositorio Git con el código de los desarrollos realizados en el proyecto y la ultimas versión estable de la app (véase Anexo 4).
- **DEMO Visor Web La Palma:** video que muestra la aplicación web (véase Anexo 5).

11. PRESUPUESTO

Una parte importante en proyectos de este tipo es el presupuesto, ya que la financiación es imprescindible para poder llevar a cabo trabajos de desarrollo web. A continuación, se expondrá el presupuesto requerido para la programación de la aplicación, desglosado en detalle en tres partes distintas, hardware, software y coste de producción, para después unir el conjunto de resultados y hacer un último balance total del coste que requeriría realizar este Trabajo Final de Grado.

11.1 HARDWARE

En cuanto al presupuesto dedicado al hardware del proyecto, para realizar el trabajo es necesario adquirir un equipo que cumpla al menos con las características de mi PC (mencionado en el apartado 4.2). Esta condición sirve para plantear una estimación de coste que cumpla las expectativas del proyecto, a pesar de que el desarrollo de la aplicación no ha tenido exigencia técnica de ningún tipo.

A continuación, se muestra la tabla de costes que ha tenido el equipo del proyecto, así como su estimación de uso según el número de meses de trabajo presentados en la Sección 8, junto con el proceso del desarrollo, vida útil del hardware empleado y el cociente de estos, factor de amortización, que nos permite obtener el coste total del elemento.

Material	Coste bruto	Unidades	Tiempo de uso (meses)	Vida útil (meses)	Factor de amortización	Coste Total
Equipo personal	2.000€	1	9	120	3/40	150€

Tabla 9. Presupuesto de hardware

11.2 SOFTWARE

Por otro lado, el software empleado en el proyecto se enumeró en gran parte ya en los apartados de la Sección 3 (sobre las tecnologías utilizadas en soluciones del estado del arte). En la Sección 3, se puede observar que todas son de código abierto. No obstante, fuera de las propias del desarrollo de la aplicación he utilizado otras tecnologías que, si requieren de licencia como el sistema operativo del PC, Windows 11 Home [68] o el paquete de Microsoft Office [69] empleado

para la redacción de la memoria y otros documentos. Por ello en la siguiente tabla se plasmará los costes del software necesario para este proyecto:

Material	Coste bruto	Unidades	Tiempo de uso (meses)	Vida útil (meses)	Factor de amortización	Coste total
Windows 11 Home	109,99€	1	9	120	3/40	8,25€
Microsoft Office	299€	1	9	120	3/40	22,43€
Pentaho Data Integration	0€	1	9	-	-	0€
Mongo DB	0€	1	9	-	-	0€
Geoserver	0€	1	9	-	-	0€
Node JS	0€	1	9	-	-	0€
Docker	0€	1	9	-	-	0€
QGIS	0€	1	9	-	-	0€
VS Code	0€	1	9	-	-	0€
Presupuesto total						30,68€

Tabla 10. Presupuesto de software

Cabe destacar que como los meses de vida útil de los productos software presentados en la tabla 10 coinciden con los meses de vida útil del equipo personal. Los productos software elegidos están vinculados de forma indefinida al dispositivo presentado en la tabla 9, por lo que podrán utilizarse mientras el equipo personal este operativo.

11.3 PRODUCCIÓN

Para la producción de este trabajo, es necesario un profesional con nociones en Sistemas de Información Geográfica y en desarrollo Full-Stack. Por ello, el perfil que más se acomoda a este trabajo es él de ingeniero especializado en análisis GIS (ingeniero en tecnologías de la información geoespacial). Actualmente, en España, este puesto de trabajo se ve remunerado por 24.000€ al año [70], aproximadamente 2.000€ al mes y 14,29€ a la hora.

Por otro lado, este proyecto ha sido desarrollado bajo el marco de Trabajo Final de Grado, que consta de 12 créditos ECTS (*European Credit Transfer and Accumulation System*), estándar europeo que fija cada crédito en 30 horas de trabajo, por lo que esto determinaría el tiempo final que requiere el analista Gis para llevar este proyecto.

Profesional	Horas requeridas	Sueldo/hora	Coste total
Ingeniero en las tecnologías de la información geoespacial	360	14,29€	5.040€

Tabla 11. Presupuesto de producción

11.4 PRESUPUESTO TOTAL DEL DESARROLLO

Finalmente, tenemos el presupuesto total que requiere, en primera instancia, la suma de los presupuestos de hardware, software y producción, quedando así el valor bruto final del coste del trabajo,

Presupuesto	Coste
Hardware	150€
Software	30,68€
Producción	5.040€
Presupuesto bruto	5.220,68€

Tabla 12. Presupuesto bruto total.

A este presupuesto es necesario añadirle incrementos porcentuales correspondientes al IVA (Impuesto al Valor Añadido), beneficios y riesgos posibles, que pudiesen afectar al desarrollo de la aplicación. A continuación, se muestra una tabla final con el presupuesto necesario para el desarrollo del portal web, contando con los valores de beneficios e imprevistos que se suelen tener en proyectos de este tipo [71] y el valor de impuesto dedicado a este trabajo en el territorio español.

Presupuesto	Coste
Presupuesto bruto total	5.220,68€
IVA (21%)	1.096,34€
Beneficios (20%)	1.044,14€
Imprevistos (10%)	522,07€
Presupuesto bruto	7.883,23€

Tabla 13. Presupuesto total

Como se puede ver en la tabla 13, el coste final para la producción de una aplicación de este tipo es de 7.883,23€.

11.5 COSTE DE DESPLIEGUE DE APLICACIÓN

A parte del desarrollo de la aplicación web es importante el despliegue del proyecto, para que esta pueda estar operativa y a disposición de los ciudadanos de La Palma y demás destinatarios.

En este caso teniendo en cuenta las características del proyecto y las opciones que ofrecen la diversa cantidad de servicios en la nube, he elegido para el despliegue los servicios proporcionados por Amazon Web Service (AWS), ya que son de los más confiables del mercado y ofrecen un gran número de productos que permiten un acople sencillo y óptimo de la aplicación en un entorno de producción. Los productos de Amazon seleccionados para la puesta en producción son los siguientes:

- **Amazon EC2 (t3.medium):** Amazon Elastic Compute Cloud es un servicio que ofrece una gran variedad de instancias virtuales con gran margen de personalización, que permiten la ejecución de aplicaciones en la nube. En concreto el producto T3 [72], es una instancia de alta escalabilidad que permite una capacidad de consumo en ráfagas, ideal para aguantar picos de uso dados por descarga de datos y además ofrece una gran rentabilidad. Esta máquina permitiría al proyecto ejecutar en la nube ambos lados, tanto cliente como servidor, así como Geoserver, garantizando su correcto funcionamiento.
- **Amazon DocumentDB (db.r6g.large):** es un servicio de base de datos administrada diseñado específicamente para almacenar bases de datos de Mongo DB, mantiene una alta escalabilidad y un gran número de productos. En concreto db.r6g.large [73] permite un rendimiento notable, escalabilidad automática y fácil mantenimiento. Por esto es el producto indicado para el traslado de las colecciones Mongo del proyecto.
- **Amazon Route 53:** Por último, Route 53 [74] permite una fácil y rápida asignación de dominio y configuración DNS (Sistema de Nombre de Dominios) a las máquinas de Amazon, dando así una perfecta compatibilidad con la instancia del proyecto.

Finalmente, ya con los productos seleccionados he realizado la estimación de costos que podría tener el lanzamiento a producción de la aplicación, tomando en cuenta previsiones horas de servicio mensuales (720) y de número de peticiones por zona (1 millón), se puede obtener el siguiente presupuesto para un año.

Producto	Precio bruto por año	Tiempo de uso (meses)	Coste total
EC2 t3.medium	201,48€	12	16,80€
db.r6g.large	2.460,12€	12	205,01€
DNS Route 53	40,32€	12	3,36€
Presupuesto total			225,17€

Tabla 14. Presupuesto de despliegue de aplicación

Por lo que el presupuesto, ya tras haber desarrollado la aplicación y realizar su despliegue, durante al menos un año es de 225,17€ al mes.

12. CONCLUSIONES

Este proyecto ha requerido una cantidad esfuerzo y tiempo considerable debido a sus características, pero me ha permitido adquirir varias experiencias de gran utilidad en mi vida profesional.

La adquisición constante de conocimientos, la investigación de datos y lógica del negocio, o el aprendizaje en la utilización de tecnologías, son destrezas importantes que he ido desarrollando a medida que avanzaba el trabajo.

Es importante también destacar la importancia de implementar una metodología clara de trabajo, que permiten la planificación y la división de tareas. En estas tareas, los profesionales, obtienen un rol específico que mejora la calidad del proyecto. Este trabajo ha requerido también asumir los roles de desarrollador software, cliente, desarrollador, usuario, etc. Así también comprendí la complejidad que adquiere cada papel y la pieza clave que representan cada uno en un proyecto web.

Volviendo al material resultante del Trabajo Final de Grado, me parece que la versión final de la aplicación es sumamente interesante, cumple con los objetivos y alcance planificado al inicio, proporcionando un producto útil con una gran variedad de funcionalidades y una calidad de datos cuidada al detalle, convirtiéndolo así en una aplicación única, y que cumpliría a la perfección en su afán de comunicar y dar voz a los sucesos ocurridos en la isla de La Palma.

12.1 FUTURAS MEJORAS Y RECOMENDACIONES

En mi opinión, y esperando la mejora futura de la aplicación, me gustaría enumerar los aspectos en los que puede desarrollarse y las medidas que podrían aportar valor en cada sección:

- **Funcionalidades:** tras ver la versión final de la aplicación y con una vista retrospectiva en el desarrollo de funciones de la web, se puede valorar como algunas no aprovechan todos las tecnologías y estrategias implementadas. Un ejemplo puede ser la ejecución de revisión de alarmas de forma independiente aprovechando la periodicidad de importación de datos dada por los procesos PDI de incidencias de la DGT, o también el desarrollo de la figura del manual de usuario, permitiendo conocer distintas versiones de éste según el rol establecido y el idioma seleccionado.
- **Interfaz y diseño:** se puede ampliar el desarrollo de elementos de comunicación con el usuario y mejorar su experiencia en la aplicación mediante la incorporación de cuadros de información con breves explicaciones de cada funcionalidad al pasar por encima. También se podría añadir un sistema de símbolos de mapa que mejore el sistema de marcadores actual, o trabajar mejora del diseño del encabezado al ser un usuario

registrado para mostrar la información básica del perfil (nombre de usuario y rol), y demás procesos que aumentarían la calidad del portal.

- **Seguridad:** un aspecto importante es la seguridad de datos. Se podrían implementar mediante medidas adicionales de control de entrada de información como el CAPTCHA (*Completely Automated Public Turing test to tell Computers and Humans Apart*), para verificar la identidad de usuarios, o el aumento de procesos de encriptación de datos que aseguren la inviolabilidad de la información proporcionada por los usuarios. Además, se podrías diseñar un acuerdo legal con los usuarios que permita el respaldo jurídico a la hora de la gestión de sus datos también es un punto importante dentro de estas mejoras.
- **Conectividad con organismos relacionados:** Por último, un aspecto que podría ser enriquecedor para la aplicación es la conectividad con organismos públicos apuesten por brindar ayuda a posibles ciudadanos afectados o atiendan a peticiones de estos. Un ejemplo de estos puede ser la presencia del Plan Valle, o la conexión con otros portales mencionados en el Apartado 2.4 “Proyectos relacionados”, donde los usuarios puedan expandir el estudio de la información, más allá de la mostrada en la app.

13. REFERENCIAS

- [1] Canarias7, «La Palma cumple dos años desde que se apagó el 'Tajogaite',» 25 Diciembre 2023. [En línea]. Available: <https://www.canarias7.es/canarias/la-palma/palma-cumple-dos-anos-apago-tajogaite-20231225104848-nt.html>. [Último acceso: 23 Junio 2024].
- [2] La Palma Smart Island, «Inicio - La Palma Smart Island,» [En línea]. Available: <https://www.lapalmasmartisland.es/>. [Último acceso: 23 Junio 2023].
- [3] Cabildo de La Palma, «Inicio | Cabildo de La Palma,» [En línea]. Available: <https://cabildodelapalma.es/es/>. [Último acceso: 23 Junio 2023].
- [4] Open Data La Palma, «La Palma Open Data,» [En línea]. Available: <https://www.opendatalapalma.es/>. [Último acceso: 23 Junio 2024].
- [5] La Palma Visualizer, «Home - Grafana,» [En línea]. Available: <https://lapalma-visualizer.hopu.eu/d/ft0F2bv7k/home?orgId=1&kiosk>. [Último acceso: 23 Junio 2024].
- [6] El tiempo en La Palma, «HD Meteo live! - La Palma Smart Island,» [En línea]. Available: <https://www.lapalmasmartisland.es/hd-meteo-live/>. [Último acceso: 23 Junio 2024].
- [7] Riesgo Volcánico La Palma, «Riesgo volcánico,» [En línea]. Available: <https://riesgovolcanico-lapalma.hub.arcgis.com/>. [Último acceso: 23 Junio 2024].
- [8] Plan Valle, «PLAN VALLE,» [En línea]. Available: <https://www.gobiernodecanarias.org/infovolcanlapalma/planvalle/>. [Último acceso: 23 Junio 2024].
- [9] Pentaho Data Integration, «Pentaho + Platform for Complete Data Management | Pentaho,» [En línea]. Available: <https://pentaho.com/pentaho-community-edition/>. [Último acceso: 24 Junio 2024].
- [10] MongoDB, «MongoDB: The Developer Data Platform | MongoDB,» [En línea]. Available: <https://www.mongodb.com/>. [Último acceso: 24 Junio 2024].
- [11] GeoServer, «GeoServer,» [En línea]. Available: <https://geoserver.org/>. [Último acceso: 24 Junio 2024].
- [12] GeoServer 2.23.1, «GeoServer,» [En línea]. Available: <https://geoserver.org/release/2.23.1/>. [Último acceso: 24 Junio 2024].

- [13] Node.js, «Node.js - Run JavaScript Everywhere,» [En línea]. Available: <https://nodejs.org/en>. [Último acceso: 24 Junio 2024].
- [14] D. Adams, «Node.js - Node v18.16.0 (LTS),» 13 Abril 2023. [En línea]. Available: <https://nodejs.org/en/blog/release/v18.16.0>. [Último acceso: 24 Junio 2024].
- [15] Express, «Express - Node.js web application framework,» [En línea]. Available: <https://expressjs.com/>. [Último acceso: 24 Junio 2024].
- [16] R. Burunkov, «GitHub - richardgirges/express-fileupload,» [En línea]. Available: <https://github.com/richardgirges/express-fileupload>. [Último acceso: 24 Junio 2024].
- [17] Í. M. Prado, «GitHub - expressjs/cors,» [En línea]. Available: <https://github.com/expressjs/cors>. [Último acceso: 24 Junio 2024].
- [18] motdotla, «GitHub - motdotla/dotenv,» [En línea]. Available: <https://github.com/motdotla/dotenv>. [Último acceso: 24 Junio 2024].
- [19] Í. M. Prado, «GitHub - expressjs/morgan,» [En línea]. Available: <https://github.com/expressjs/morgan>. [Último acceso: 24 Junio 2024].
- [20] A. S. Mahapatra, «GitHub - kelektiv/node.bcrypt.js,» [En línea]. Available: <https://github.com/kelektiv/node.bcrypt.js>. [Último acceso: 24 Junio 2024].
- [21] V. Karpov, «GitHub - Automattic/mongoose,» [En línea]. Available: <https://github.com/Automattic/mongoose>. [Último acceso: 24 Junio 2024].
- [22] Jake Lacey, «GitHub - auth0/node-jsonwebtoken,» [En línea]. Available: <https://github.com/auth0/node-jsonwebtoken>. [Último acceso: 24 Junio 2024].
- [23] S. Mookerjee, «GitHub - nodemailer/nodemailer,» [En línea]. Available: <https://github.com/nodemailer/nodemailer>. [Último acceso: 24 Junio 2024].
- [24] R. Inagaki, «GitHub - ryu1kn/csv-writer,» [En línea]. Available: <https://github.com/ryu1kn/csv-writer>. [Último acceso: 24 Junio 2024].
- [25] I. I. Chernev, «GitHub - moment/moment,» [En línea]. Available: <https://github.com/moment/moment>. [Último acceso: 24 Junio 2024].

- [26] R. Sharp, «GitHub - remy/nodemon,» [En línea]. Available: <https://github.com/remy/nodemon>. [Último acceso: 24 Junio 2024].
- [27] Vue.js, «Vue.js - The Progressive JavaScriptFramework | Vue.js,» [En línea]. Available: <https://vuejs.org/>. [Último acceso: 25 Junio 2024].
- [28] Vue-Router, «Vue Router - The official Router for Vue.js,» [En línea]. Available: <https://router.vuejs.org/>. [Último acceso: 25 Junio 2024].
- [29] Vuex, «What is Vuex | Vuex,» [En línea]. Available: <https://vuex.vuejs.org/>. [Último acceso: 25 Junio 2024].
- [30] Vue I18n, «Vue I18n | Internationalization plugin for Vue.js,» [En línea]. Available: <https://vue-i18n.intlify.dev/>. [Último acceso: 25 Junio 2024].
- [31] TypeScript, «TypeScript: JavaScript with syntax for types,» [En línea]. Available: <https://www.typescriptlang.org/>. [Último acceso: 25 Junio 2024].
- [32] Babel, «Babel · Babel,» [En línea]. Available: <https://babeljs.io/>. [Último acceso: 25 Junio 2024].
- [33] Leaflet, «Leaflet - a JavaScript library for interactive maps,» [En línea]. Available: <https://leafletjs.com/>. [Último acceso: 25 Junio 2024].
- [34] @types/leaflet, «@types/leaflet - npm,» [En línea]. Available: <https://www.npmjs.com/package/@types/leaflet>. [Último acceso: 25 Junio 2024].
- [35] Leaflet heat, «GitHub - Leaflet/Leaflet.heat,» [En línea]. Available: <https://github.com/Leaflet/Leaflet.heat>. [Último acceso: 25 Junio 2024].
- [36] P. Liedman, «GitHub - perliedman/leaflet-control-geocoder,» [En línea]. Available: <https://github.com/perliedman/leaflet-control-geocoder>. [Último acceso: 25 Junio 2024].
- [37] Bootstrap, «Bootstrap · The most popular HTML, CSS and JS library in the world,» [En línea]. Available: <https://getbootstrap.com/>. [Último acceso: 24 Junio 2024].
- [38] D. Mozgovoy, «GitHub - axios/axios,» [En línea]. Available: <https://github.com/axios/axios>. [Último acceso: 25 Junio 2024].
- [39] Docker, «Docker: Accelerated Container Application Development,» [En línea]. Available: <https://www.docker.com/>. [Último acceso: 25 Junio 2024].

- [40] Tomcat, «Apache Tomcat® - Welcome!», [En línea]. Available: <https://tomcat.apache.org/>. [Último acceso: 25 Junio 2024].
- [41] Quantum Geographic Information System, «Bienvenido al proyecto QGIS!», [En línea]. Available: <https://www.qgis.org/es/site/>. [Último acceso: 25 Junio 2024].
- [42] GEOCAT, «GeoCat Bridge - Publish from your favorite desktop GIS», [En línea]. Available: <https://www.geocat.net/bridge/>. [Último acceso: 25 Junio 2024].
- [43] VS Code, «Visual Studio Code - Code Editing. Redefined», [En línea]. Available: <https://code.visualstudio.com/>. [Último acceso: 25 Junio 2024].
- [44] GIT, «Git», [En línea]. Available: <https://git-scm.com/>. [Último acceso: 25 Junio 2024].
- [45] Postman, «Postaman API Platform | Sign Up for Free», [En línea]. Available: <https://www.postman.com/>. [Último acceso: 25 Junio 2024].
- [46] Instituto Geográfico Nacional (IGN), «Inicio - Instituto Geográfico Nacional», [En línea]. Available: <https://www.ign.es/web/ign/portal>. [Último acceso: 23 Junio 2024].
- [47] Dirección General de Tráfico (DGT), «DGT - Inicio», [En línea]. Available: <https://www.dgt.es/inicio/>. [Último acceso: 23 Junio 2024].
- [48] Mapa de estado del tráfico e incidencias, «Información de tráfico», [En línea]. Available: <https://infocar.dgt.es/etraffic/>. [Último acceso: 23 Junio 2024].
- [49] Sede Electrónica del Catastro, «Sede Electrónica del Catastro - Inicio», [En línea]. Available: <https://www.sedecatastro.gob.es/>. [Último acceso: 23 Junio 2024].
- [50] GISGeography, «World Geodetic System (WGS84)», 10 Marzo 2024. [En línea]. [Último acceso: 26 Junio 2024].
- [51] Servicios Inspire del Catastro, «CARTOGRAFÍA CATASTRAL - INSPIRE», [En línea]. Available: <https://www.catastro.minhap.es/webinspire/index.html>. [Último acceso: 26 Junio 2024].
- [52] Open Data La Palma, «Edificaciones. | Edificaciones | Open Data La Palma», [En línea]. Available: https://www.opendatalapalma.es/datasets/1c93601970fb41b480599c54fff25e4f_0/explore?location=28.691250%2C-17.789659%2C11.00. [Último acceso: 25 Junio 2024].
-

- Open Data La Palma, [En línea]. Available: <https://www.opendatalapalma.es/datasets/red-de-carreteras/explore?location=28.669229%2C-17.841515%2C11.60>. [Último acceso: 26 Junio 2024].
- Open Data La Palma, «perímetro dron 211101 0930 sur | perímetro dron 211101 0930 sur | Open Data La Palma,» [En línea]. Available: <https://www.opendatalapalma.es/datasets/lapalma::per%C3%ADmetro-dron-211101-0930-sur/explore?location=28.618112%2C-17.895898%2C14.96>. [Último acceso: 26 Junio 2024].
- Open Data La Palma, «bocas eruptivas IGN | bocas eruptivas IGN | Open Data La Palma,» [En línea]. Available: <https://www.opendatalapalma.es/datasets/bocas-eruptivas-ign/explore>. [Último acceso: 26 Junio 2024].
- IGN, «terremotosCANA.js,» [En línea]. Available: <https://www.ign.es/web/resources/volcanologia/tproximos/json/terremotosCANA.js>. [Último acceso: 26 Junio 2024].
- IGN, «Información sísmica - Instituto Geográfico Nacional,» [En línea]. Available: <https://www.ign.es/web/resources/volcanologia/tproximos/json/terremotosCANA.js>. [Último acceso: 26 Junio 2024].
- DGT, «Leyenda de iconos,» [En línea]. Available: <https://infocar.dgt.es/etraffic/jsp/leyendas.jsp>. [Último acceso: 25 Junio 2024].
- M. Dianto, «GitHub - maasdi/pentaho-mongodb-delete-plugin,» [En línea]. Available: <https://github.com/maasdi/pentaho-mongodb-delete-plugin>. [Último acceso: 25 Junio 2024].
- Open Geospatial Consortium, «Web Map Service - Open Geospatial Consortium,» [En línea]. Available: <https://www.ogc.org/standard/wms/>. [Último acceso: 27 Junio 2024].
- J. Terra, «What is Client-Server Architecture? Everything You Should Know,» 7 Agosto 2023. [En línea]. Available: <https://www.simplilearn.com/what-is-client-server-architecture-article>. [Último acceso: 27 Junio 2024].
- J. Martins, «Scrum: conceptos clave y cómo se aplica en la gestión de proyectos,» 15 Febrero 2024. [En línea]. Available: <https://asana.com/es/resources/what-is-scrum>. [Último acceso: 27 Junio 2024].

- [63] Passport, «Pasport.js,» [En línea]. Available: <https://www.passportjs.org/>. [Último acceso: 27 Junio 2024].
- [64] Microsoft, «Segoe UI font family,» [En línea]. Available: <https://learn.microsoft.com/es-es/typography/font-list/segoe-ui>. [Último acceso: 27 Junio 2024].
- [65] Icons8, «leaflet imagenesleaflet Icons Logos Symbols,» [En línea]. Available: <https://icons8.com/icons/set/leaflet-imagenesleaflet>. [Último acceso: 27 Junio 2024].
- [66] Flaticon, «Iconos vectoriales y stickers PNG, SVG, EPS, PSD y CSS,» [En línea]. Available: <https://www.flaticon.es/>. [Último acceso: 27 Junio 2024].
- [67] T. Pointhuber, «GitHub - pointhi/leaflet-color-markers,» [En línea]. Available: <https://github.com/pointhi/leaflet-color-markers>. [Último acceso: 27 Junio 2024].
- [68] PC componentes, «Microsoft Windows 11 Home Standard,» [En línea]. Available: https://www.pcccomponentes.com/microsoft-windows-11-home-standard?s_kwcid=AL!14405!3!649510106339!!!g!2319299510154!&gad_source=1&gc lid=Cj0KCQjwsuSzBhCLARIaIcdLm6yOCSv4_aRfJoVCJjmd5mtF7FVqmzzmRUefg SDquKxXt1xpgIOnIQaAjMGEALw_wcB. [Último acceso: 27 Junio 2024].
- [69] Microsoft, «Comprar Office Hogar y Empresa 2021 - Descarga y precios,» [En línea]. Available: <https://www.microsoft.com/es-es/microsoft-365/p/office-hogar-y-empresa-2021/cfq7ttc0hpn4?activetab=pivot:informaci%C3%B3ngeneraltab>. [Último acceso: 27 Junio 2024].
- [70] jooble, «Analista gis salarios,» [En línea]. Available: <https://es.jooble.org/salary/analista-gis#:~:text=%C2%BFCu%C3%A1nto%20gana%20Analista%20gis%20en,14%C29%20%E2%82%AC%20por%20hora..> [Último acceso: 27 Junio 2024].
- [71] workamajig, «Project Profitability,» [En línea]. Available: <https://www.workamajig.com/blog/project-profitability>. [Último acceso: 27 Junio 2024].
- [72] Amazon Web Service, «Instancias T3 de Amazon EC2,» [En línea]. Available: <https://aws.amazon.com/es/ec2/instance-types/t3/>. [Último acceso: 28 Junio 2024].
- [73] Amazzon Web Service, «Precios de Amazon DocumentDB (con compatibilidad con MongoDB),» [En línea]. Available: <https://aws.amazon.com/es/documentdb/pricing/>. [Último acceso: 27 Junio 2024].

[74] Amazon Web Service, «Precios de Amazon Route 53,» [En línea]. Available: <https://aws.amazon.com/es/route53/pricing/>. [Último acceso: 27 Junio 2024].

14. ANEXOS

En este apartado se pueden encontrar los accesos a los materiales anexos a esta memoria:

Anexo 1

[Archivo PDF](#) del Mockup de definición de requisitos, esquema de la app.

Anexo 2

[Carpeta de archivos](#) de transformaciones y trabajos Pentaho Data Integration.

Anexo 3

[Carpeta de archivos](#) de arranque de contenedores Mongo y Geoserver.

Anexo 4

[Enlace a repositorio GitHub](#) con el Cliente y Servidor del Visor La Palma.

Anexo 5

