Escuela Politécnica Superior de Jaén

Grado en Ingeniería Informática

UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior de Jaén

Trabajo Fin de Grado

APLICACIÓN WEBGL PARA LA GESTIÓN, MONITORIZACIÓN Y PREDICCIÓN DE LA EVOLUCIÓN DE PLANTACIONES DE OLIVAR

Alumno: Pablo Latorre Hortelano

Tutor: Prof. D. David Jurado Rodríguez
Dpto: Departamento de Informática

Tutor: Prof. D. Juan Manuel Jurado Rodríguez
Dpto: Departamento Lenguajes y Sistemas informáticos Universidad de Granada

Julio, 2023

(Página intencionalmente en blanco)

Universidad de Jaén

Departamento de Informática

D. David Jurado Rodríguez y D. Juan Manuel Jurado Rodríguez, tutores del Trabajo Fin de Grado titulado: '**Aplicación WebGL para la gestión, monitorización y predicción de la evolución de plantaciones de olivar**', que presenta Pablo Latorre Hortelano, autorizan su presentación para defensa y evaluación en la Escuela Politécnica Superior de Jaén.

Jaén, Julio de 2023

El alumno:

LATORRE HORTELANO PABLO - 77377195C

Firmado digitalmente por LATORRE HORTELANO PABLO - 77377195C
Fecha: 2023.07.10 10:08:34 +02'00'

Pablo Latorre Hortelano

Los tutores:

JURADO RODRIGUEZ JUAN MANUEL - 53598045H

Firmado digitalmente por JURADO RODRIGUEZ JUAN MANUEL - 53598045H
Fecha: 2023.07.10 10:14:41 +02'00'

JURADO RODRIGUEZ DAVID - 53598046L

Firmado digitalmente por JURADO RODRIGUEZ DAVID - 53598046L
Fecha: 2023.07.10 10:23:49 +02'00'

D. David Jurado Rodríguez

D. Juan Manuel Jurado Rodríguez

(Página intencionalmente en blanco)

Agradecimientos

En esta sección, me gustaría aprovechar la oportunidad que se me brinda para expresar mi mas sincero agradecimiento a todas las personas que a lo largo del desarrollo de este proyecto han sido fundamentales para que pueda salir adelante y culminar con un trabajo del que me siento orgulloso y gracias al cual saco una gran experiencia que me ha permitido desarrollarme en el ambito de la informática.

En primer lugar quisiera dedicar este trabajo a aquellos seres queridos que, aunque ya no se encuentran a mi lado, han sido fuente inagotable de inspiración y apoyo a lo largo de mi vida y que sin estar presentes en esta última etapa no han dejado de ayudarme.

Expresar mi gratitud a mis guias durante el desarrollo del trabajo y los responsables de que yo haya tenido la posibilidad de ser parte de este proyecto mis tutores David Jurado y Juan Manuel Jurado junto con otros integrantes del grupo de Informática Gráfica de la Universidad de Jaén como Francisco Feito o Alfonso López. Gracias por abrirme las puertas de lo que es pertenecer a un grupo de trabajo dentro de la universidad, sabiendo llevarme por el camino correcto y no caer en ideas sin un final coherente.

Agradecer también a los compañeros que han hecho mas ameno todo este proceso compartiendo y acompañandome en el camino curso tras curso, por los animos, los trabajos en grupo que ocasiones se pasaban de creativos o charlas durante estos cuatro años.

Un agradecimiento especial a mi familia, mis padres Aurora y Manuel, así como a mi hermana Marina, que me han facilitado el camino desde el principio permitiendo que pueda centrarme en los estudios y la realización de este trabajo, permitiendo que mi motivación y curiosidad no tuviera limites, por estar siempre a mi lado y en ocasiones pensar mas en mi que en ellos mismos. Gracias por escuchar mis confusas explicaciones sobre todos los proyectos que se me planteaban, por los consejos y por confiar siempre en que aunque yo no viera el camino acabaria llegando hasta el final. Gracias por ser la familia que cualquier persona desearía tener.

Por último, pero no menos importante me gustaria agraceder a mi pareja Lucia por al igual que mi familia estar siempre en los momentos importantes alegrandose por mi y en los peores ayudandome a levantarme, siempre con la paciencia necesaria para lidiar con ello, en definitiva gracias por acompañarme a lo largo de estos cuatro años, por en ocasiones tener la cabeza mas en mi y mi trabajo que en tus propias preocupaciones, por ser la mejor compañera todo este tiempo y una persona inigualable.

Muchas gracias a todos.

FICHA DEL TRABAJO FIN DE TÍTULO	
Titulación	Grado en Ingeniería Informática
Modalidad	Proyecto de Ingeniería
Especialidad (solo TFG)	Tecnologías de la Información
Mención (solo TFG)	Sistemas Gráficos
Idioma	Español
Tipo	General
TFT en equipo	No
Autor/a	Pablo Latorre Hortelano
Fecha de asignación	02/11/2022
Descripción corta	En este trabajo se pretende desarrollar una aplicación web en la que poder integrar conjuntos de datos adquiridos desde distintos sensores (RGB, espectrales, termográfico y modelos 3D) sobre una plantación de olivar. La información reunida será clave para determinar el estado actual de los cultivos y predecir su evolución futura teniendo en cuenta otros factores como histórico de producción y datos meteorológicos. Como resultado de este trabajo se proveerá una aplicación basada en WebGL cuyo uso está dirigido a los responsables de las explotaciones agrarias.

NORMAS APLICADAS EN ESTE DOCUMENTO	
LOCALES	
TFT-UJA:2017	Normativa de Trabajos Fin de Grado, Fin de Máster y otros Trabajos Fin de Título de la Universidad de Jaén (Normativa marco UJA aprobada en Consejo de Gobierno)
TFT-EPSJ:2017	Normativa sobre Trabajos Fin de Grado y Fin de Máster en la Escuela Politécnica Superior de Jaén (Normativa EPSJ aprobada en Junta de Escuela)
TFT-EPSJ	Criterios de evaluación y normas de estilo para TFG y TFM de la Escuela Politécnica Superior de Jaén
NACIONALES E INTERNACIONALES	
ISO 2145:1978	Documentación - Numeración de divisiones y subdivisiones en documentos escritos
UNE 50132:1994	Traducción de la ISO 2145
IEEE:2006	Estilo de referencias y citas de IEEE (Institute of Electrical and Electronics Engineers)

NORMAS UTILIZADAS COMO BASE O REFERENCIA	
NACIONALES	
UNE 157001:2014	Criterios generales para la elaboración formal de los documentos que constituyen un proyecto técnico
UNE 157801:2007	Criterios generales para la elaboración de proyectos de sistemas de información
*Estas normas se han utilizado **como base o referencia** para la inclusión de algunos contenidos y definiciones sobre elaboración de proyectos, entendiendo como **proyecto** la documentación consensuada entre una empresa y un cliente, que da lugar al perfeccionamiento de un contrato para la elaboración de una obra o la prestación de un servicio. Por consiguiente, no debe esperarse la aplicación de estas normas en cuanto a la completitud de los contenidos ni a la organización de los mismos.*	

Contenido

Índice de ilustraciones

Índice de tablas

RESUMEN

El presente Trabajo de Fin de Grado aborda el desarrollo de una aplicación web para la gestión eficiente, monitorización y control de parcelas de olivar. El olivar es un cultivo de gran importancia económica y social en muchas regiones, destacando sobre otras la provincia de Jaén. Por lo que contar con una herramienta digital que facilite su administración y optimización resulta fundamental y de gran ayuda para el sector tanto técnico como administrativo.

El objetivo principal de este proyecto es diseñar y desarrollar una aplicación web que permita a los propietarios y gestores de parcelas de olivar realizar un seguimiento exhaustivo de sus cultivos. Para ello, esta aplicación web suministra una serie de datos individualizados por parcela. Además realiza una reconstrucción 3D del ecosistema. Este hecho permite trabajar con objetos complejos mostrando el estado actual de las plantaciones. Para lograrlo, se han utilizado tecnologías modernas de desarrollo web y se ha adoptado un enfoque centrado en la usabilidad y la funcionalidad.

La aplicación web ofrece funcionalidades clave como la gestión y muestreo de información, la edición y visualización de las parcelas en un entorno virtual y un cierto control de datos. Además los usuarios pueden registrar y organizar información relevante sobre cada una de sus parcelas de olivar, incluyendo datos geográficos, históricos, fechas de plantación, etc. Asimismo, se ha implementado un sistema de detección de entidades con el fin de estimar la posición de cada plantación, permitiendo a los usuarios disponer de un mayor nivel de información hasta ahora desconocido. Este hecho es de vital importancia en tareas optimización de rendimiento y actividades como la recolección y el riego.

En pro de favorecer la visualización de los datos, se ha integrado un módulo de generación de gráficas, que partiendo de los datos históricos proporciona a los usuarios un análisis detallado del rendimiento de sus parcelas o la producción de aceite, entre otros indicadores relevantes. Esta información resulta de gran utilidad para la toma de decisiones estratégicas y la optimización de recursos.

En conclusión, este TFG ha culminado en el desarrollo exitoso de una aplicación web intuitiva y completa para la gestión de parcelas de olivar. Se espera que esta herramienta proporcione a los usuarios una mayor eficiencia en la gestión de sus cultivos, mejorando así su productividad y contribuyendo al desarrollo sostenible del sector olivarero.

ABSTRACT

In this Final Degree Project, a web application for the management, monitoring, and efficient control of olive grove plots is developed. Olive cultivation is of great economic and social importance in several regions, especially in Jaén. Therefore, having a digital tool that streamlines its administration and optimization is crucial and immensely beneficial, both technically and administratively

The main objective of this project is to design and develop a web application that allows the owners and managers of olive grove plots to carry out exhaustive monitoring of their crops, in a thematic data format, i.e. with constant access to their real data, and in a format with more visual access, being able to access data and models that allow them to visualize the most up-to-date state of the plot. To achieve this, modern web development technologies have been used and an approach focused on usability and functionality has been adopted, in addition models and scenes have been developed using three-dimensional rendering engines.

The web application offers key functionalities such as information management and sampling, editing and visualization of the 3D scenes and some data control where users can register and organize relevant information about each of their olive grove plots, including geographical data, historical data, planting dates, etc. In addition, an entity detection system has been implemented in order to estimate the position of the olive tres. This fact allow users to have a higher level of information that was previously unknown or difficult to access and that could be useful to optimize performance and activities such as harvesting and irrigation.

A graph generation module has also been integrated, which, based on historical data, provides users with a detailed analysis of the yield of their plots, oil production and other relevant indicators. This information is invaluable for strategic decision-making and resource optimization.

In conclusion, this TFG has successfully completed the development of an intuitive and complete web application for the management of olive plantations. It is expected that this tool will provide users with greater efficiency in the management of their crops, thus improving their productivity and contributing to the sustainable development of the olive sector.

Capítulo 1: INTRODUCCIÓN

En este Capítulo se presenta la especificación del trabajo, con una estructura y contenidos **inspirados** en los criterios y recomendaciones que establece la norma UNE 157801:2007 - "*Criterios Generales para la elaboración de proyectos de Sistemas de Información*".

A lo largo del documento se utilizarán términos y acrónimos cuya descripción aparecen en el Capítulo 8 Definiciones y abreviaturas

1.1 Motivación

Las plantaciones de Olivar son un bien natural muy presente en la provincia de Jaén, estas conforman gran parte del ecosistema en el que nos encontramos. Principalmente es por este motivo que su estudio y la mejora de su tratamiento puede ser un factor clave en el desarrollo y la mejora del posible producto ofertado, que además supone una parte muy importante de la economía de nuestra provincia. A su vez, el estudio del olivar puede suponer un cambio no solo en nuestra provincia, sino cada vez en más lugares, pues la expansión de este tipo de plantaciones llega cada vez a más zonas del terreno español (Ilustración 1)(Ilustración 2) con las múltiples variantes que pueden presentar, por este motivo las evoluciones en el mismo dejan de ser un interés único de la zona y se está globalizando.

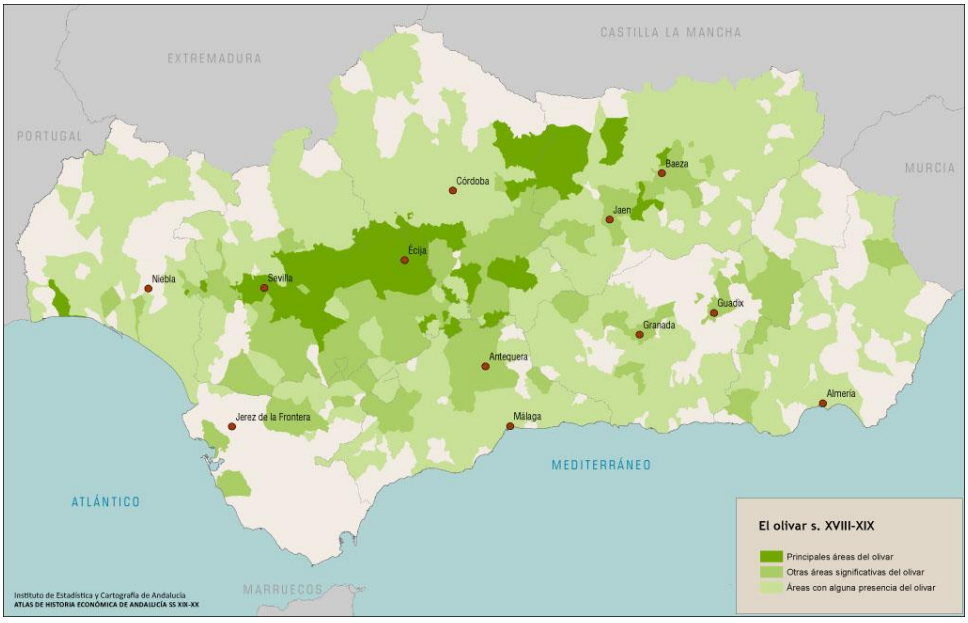


Ilustración 1: Distribución de zonas de Olivar en Andalucía[1].

1 https://www.juntadeandalucia.es/institutodeestadisticaycartografia/atlashistoriaecon/atlas_cap_15.html

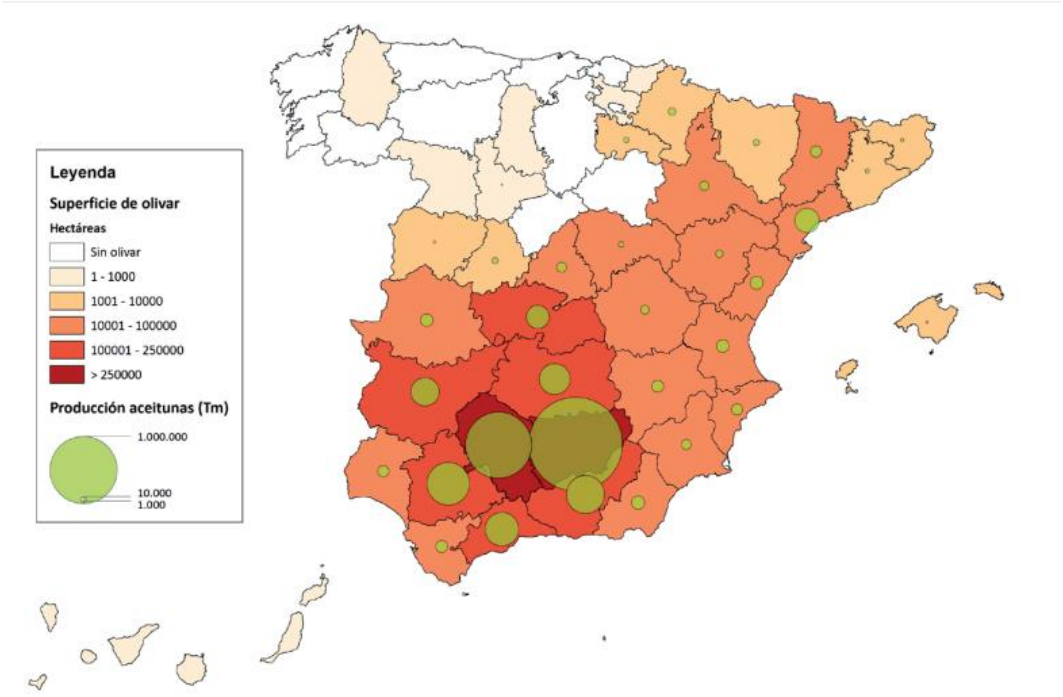


Ilustración 2: Superficie y producción del olivar en España[2].

Los entornos de parcelas de olivar son capaces de ofrecernos una cantidad inmensa de información, esta puede ir desde información meteorológica que nos permite determinar el posible rendimiento de los olivos y su fruto, hasta información del terreno (humedad, temperatura…) que es un factor clave en el correcto desarrollo de una cosecha. Esta información en muchas ocasiones nos ha demostrado que el olivar depende en gran medida de factores externos incontrolables por el ser humano, provocando campañas con muy mal rendimiento debido a estos. Sin embargo, esto no es algo que debiera frenar el interés en tratar de mejorar el rendimiento pues con un correcto análisis de la información se puede buscar extraer el 100% de rendimiento para los recursos disponibles.

Toda esta información no siempre vamos a ser capaces de identificarla o procesarla con los métodos más comunes a los que puede recurrir un agricultor en su día a día en el olivar, es por esto que un cambio y una ayuda es necesaria para poder disponer de la riqueza en información que puede aportarnos el medio, y que gracias a dispositivos informatizados y / o sensorizados podemos ser capaces de analizar y aprovechar a nuestro favor.

2 https://digibug.ugr.es/bitstream/handle/10481/27511/821_Cap_44.pdf?sequence=6&isAllowed=y

A lo largo del tiempo la obtención de datos ha sido siempre una de las mayores limitaciones a la hora de la realización de ciertos proyectos o de ciertos avances en campos de desarrollo, esto no es distinto en el campo agrícola, donde debido a las grandes extensiones de terreno en las que se trabaja la obtención de datos puede llegar a ser dificultosa así como costosa en tiempo y esfuerzo. En la actualidad este proceso ha podido facilitarse bastante con los medios adecuados, pudiendo ir estos desde la sensorización de ciertas extensiones de terreno, permitiéndonos conseguir datos periódicamente, así como el uso de UAVs (Unmanned Aerial Vehicles) para el reconocimiento y modelado del terreno en distintos formatos dependiendo de las cámaras o sensores con los que se les pueda equipar.

Con este planteamiento surge la necesidad de ayudar a los trabajadores del campo, facilitándole ciertas tareas y el acceso a información que antes les resultaba imposible de manejar bien. El motivo de esto viene ligado al hecho de ser un gran volumen de datos, en la mayoría de ocasiones difíciles de capturar por los problemas comentados anteriormente, como a su vez también pudiera ser por la dificultad de interpretación de los mismos.

Por estos motivos durante el desarrollo de este trabajo se detallarán los distintos métodos y / o aplicaciones propuestas para la solución de la problemática, además de su elaboración y posibles usos dentro de la sociedad y el entorno agrícola en el que trabajamos.

1.2 Antecedentes y estado del arte

En este apartado se realiza un análisis sobre el conocimiento previo existente y que se ha desarrollado en el pasado relacionado con este trabajo. Esto nos ayudara a comprender de mejor forma el punto de partida, así como otras posibles soluciones propuestas de manera externa por gente que trabaja en el sector del desarrollo de la agricultura mediante modelos tridimensionales e integración de información.

A su vez también se comentará el estado del arte de los principios y tecnologías expuestos haciendo esto referencia a los que serían los últimos desarrollos tecnológicos en la materia.

La agricultura data del periodo Neolítico (6000–3000 a. C) coincidiendo con la sedentarización del ser humano, desde entonces hasta día de hoy esta ha sufrido una gran cantidad de cambios, las necesidades de la sociedad han cambiado, la cantidad de posibles compradores ha incrementado, los métodos de cuidado, plantación y desarrollo han ido

evolucionando y mecanizándose con el tiempo. Sin embargo, uno de los aspectos se ha mantenido todo este tiempo, siendo siempre el objetivo principal de la agricultura, este consiste en obtener el mayor rendimiento posible a una materia dada consiguiendo así aumentar la producción y por tanto el beneficio.

Una clara alternativa para conseguir maximizar este rendimiento a la vez que las necesidades cambian y los requisitos aumentan ha sido la integración de la tecnología en la agricultura o lo que es más conocido como agricultura de precisión. Esta tecnología avanza al igual que el resto de aspectos y es por esto que nos permite cada vez conseguir acceder a más información y realizar una mayor cantidad de acciones que antes resultaban imposibles.

La agricultura de precisión se ha visto definida de distintas formas a lo largo del tiempo. En sus inicios podía verse definida como la aplicación de las tecnologías para el manejo de la variabilidad espacial y temporal asociada a la producción agrícola [1]. A posteriori y de manera más actual esta recibió otra definición la cual se centraba en el correcto uso de la misma, “La agricultura de precisión surge como una manera de usar el tratamiento correcto, en el sitio adecuado en el momento preciso” [2]**.** En este último artículo también se hace especial hincapié en la necesidad de combinar toda la tecnología disponible con sistemas de información a nuestra ubicación.

Para este proyecto se ha trabajado con el término de agricultura de precisión pues se han empleado una serie de tecnologías que más adelante se explicaran de manera más desarrollada, sin embargo, previamente también se han venido usando tecnologías y se han presentado sistemas que ayudaba al mantenimiento de plantaciones y desarrollo de aplicaciones en este ámbito [3] y que veremos a continuación.

1.2.1 Obtención de datos y generación de nubes de puntos

Los datos son uno de los bienes más necesarios a la hora de llevar a cabo no solo proyectos y su desarrollo sino a la hora de poder controlar entornos como las fincas con las que se trabajaran a lo largo de este proyecto.

Es por esto que la manera de obtener los datos ha sido un campo de investigación y desarrollo a lo largo de la historia que ha ido evolucionando y esto no ha sido menos en la agricultura de precisión, que al ser un campo que fusiona la tecnología con la agricultura ha buscado durante años métodos para mejorar la captura de los mismos, así como conseguir también representaciones realistas de las zonas capturadas.

Uno de los campos más desarrollados en este ámbito y sobre todo ligado a la agricultura de precisión es la conocido como teledetección especialmente referida a la agricultura [4] [5] que implica la obtención de datos sin necesidad de tener contacto con la superficie sobre la que se trabaja. De forma más actual también se podría definir como la captura y análisis de imágenes obtenidas mediante el uso de cámaras, sensores o satélites.

En la actualidad el proceso de teledetección puede realizarse de distintos métodos, mediante el uso de satélites es capaz de otorgarnos una resolución de entre 10-60 metros que es la que es capaz de ofrecernos el proyecto Sentinel-2[3] formado por satélites multiespectrales de órbita polar (Ilustración 3), la variación de la precisión se debe principalmente a que realiza un escaneado mediante el uso de 13 bandas espectrales (Ilustración 4), las bandas con mayor resolución obtienen hasta 10 metros mientras que las de menos se quedan en 60 metros.

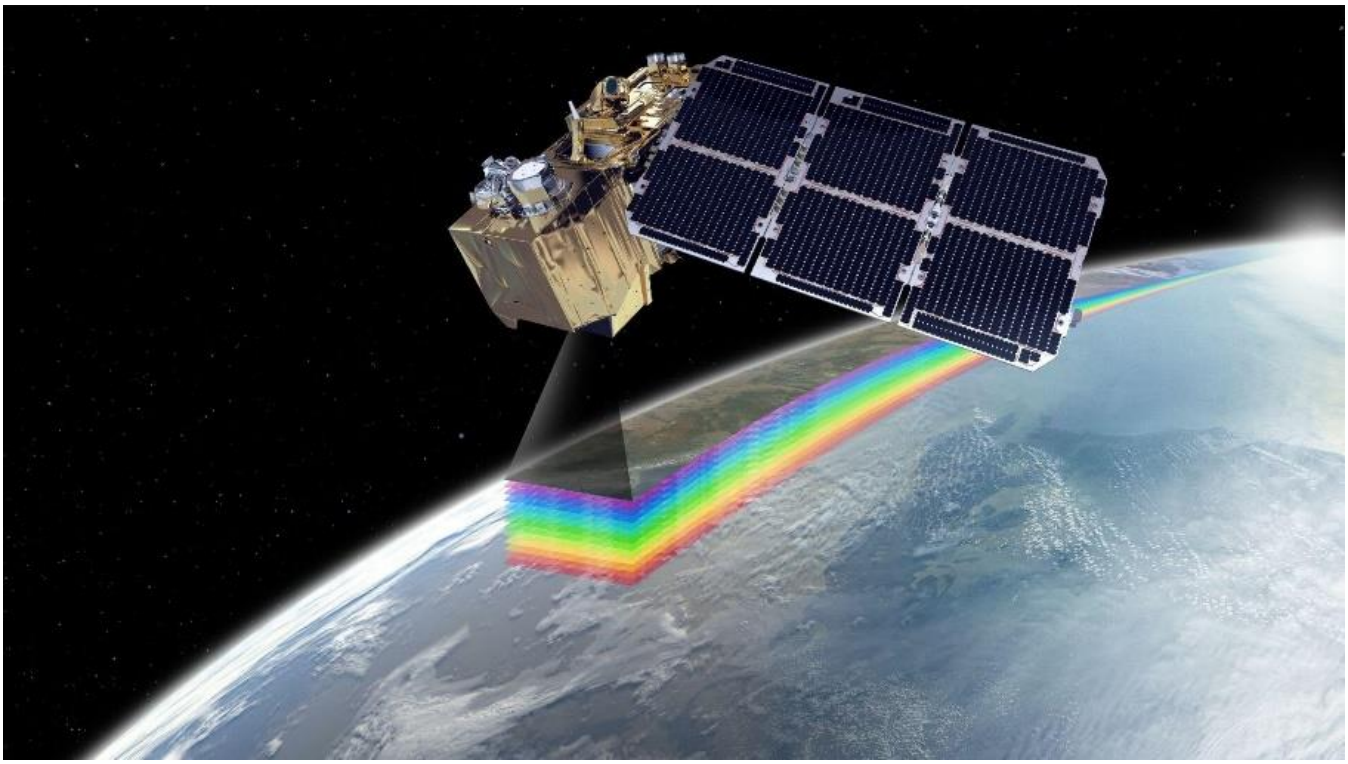


Ilustración 3. Satélite del programa Sentinel 2 y su escaneado por bandas [4].

[3] https://www.esa.int/Applications/Observing_the_Earth/Copernicus/Sentinel-2
[4] https://acortar.link/GiB3CZ

Bandas	Longitud de Onda Central (µm)	Resolución (m)
B1 – Coastal aerosol	0.443	60
B2 – Blue	0.490	10
B3 – Green	0.560	10
B4 – Red	0.665	10
B5 – Vegetation Red Edge	0.705	20
B6 – Vegetation Red Edge	0.740	20
B7 – Vegetation Red Edge	0.783	20
B8 – NIR	0.842	10
B8A – Vegetation Red Edge	0.865	20
B9 – Water Vapour	0.945	60
B10 – SWIR – Cirrus	1.375	60
B11 – SWIR	1.610	20
B12 – SWIR	2.190	20

Ilustración 4:Bandas espectrales de los satélites pertenecientes a la misión Sentinels-2[5].

Con el fin de poder mejorar el alcance para situaciones en las que es necesario trabajar con zonas a gran escala y claramente delimitadas se comenzó a utilizar dispositivos UAVs para el proceso de la teledetección. Con estos dispositivos la resolución era capaz de verse incrementada hasta valores por debajo de un metro alcanzando valores de 0.1 metros o incluso superiores, de esta manera se consiguen mejorar los datos de resolución de manera significativa, así como la generación de posibles modelos más realistas. Los UAVs en su mayoría permiten la adición de más sensores además de aquellos de captura de datos, permitiendo así enriquecer los datos obtenidos.

Sin embargo, la precisión y resolución no lo es todo y dependiendo el proyecto y los recursos a disposición puede ser mejor disponer de imágenes satelitales o vuelos por UAV, las principales diferencias de los dos métodos de teledetección comentados son:

Propiedades	Satélites Sentinel-2	UAV
Resolución	10-20-60 metros	< 1 metro
Frecuencia Informativa	Adquisición automática cada 5 días	Adquisición eventual cuando se realice un vuelo
Costes	Sin coste, libre acceso	Material y operación
Rango	Global (Escalable)	Local

Tabla 1. Comparativa Sentinel 2 – UAV.

[5] https://acortar.link/ldIL9j

En adición a la teledetección, mediante el uso de sensores y programas LiDAR (Ligth Detection and Ranging) es posible la generación de nubes de puntos en tiempo real de la zona sobrevolada, consiguiendo de esta forma una representación realista y con gran cantidad de información en tres dimensiones, además esta podrá ser modelada en el futuro y se podrá trabajar sobre la misma. Estas nubes de puntos podrán verse representadas con valores RGB para su visualización o con valores temáticos como son térmicas, multiespectrales…

Todos estos tipos de información captados con los sensores correspondientes son los que permiten extraer todo el conocimiento tras un procesamiento posterior de la información extraída sobre las fincas, este conocimiento puede ser sobre el estado de la tierra, olivos o clima, así como la generación de modelos 3D que nos permita tener una representación lo más aproximada posible a la parcela en nuestros dispositivos.

1.2.2 Servidor de mapas y capas agrícolas

El propósito de este proyecto como bien se ha comentado y podido ver de manera introductoria es el de tener un sistema de control de fincas o parcelas con toda la información disponible y fusionar esto con la posibilidad de navegación por un entorno tridimensional de la zona. Esta idea pero de manera más simplificada, ya que se desarrolló de manera exclusiva con información en 2D, fue presentada por el Gobierno de España, más específicamente por el Ministerio de Agricultura, Alimentación y Medio Ambiente con la idea de ayudar a los agricultores a poder tener su información agrícola disponible en la web así como poder llevar un conteo y mantenimiento de los datos públicos de las mismas. Este proyecto surge con el nombre de SigPac[6] (Ilustración 5)(Sistema de información geográfica de Parcelas agrícolas comunitarias).

El SigPac surge en el año 2000 y ha sufrido numerosas actualizaciones. Durante muchos años mantuvo un registro no solo de las parcelas sino que también era capaz de almacenar cierta información sobre los olivos de las mismas. Sin embargo debido al sobrecoste del mantenimiento de una base de datos de este calibre esto finalizó siendo inviable y cayendo en la degradación hacia un sistema más simplificado como el que tenemos en la actualidad. Este sistema actúa como un servicio de mapas con división por diferentes capas según el tipo de dato al que queramos acceder, este posibilita y fue pensado con el objetivo de:

[6] https://sigpac.mapama.gob.es/fega/visor/

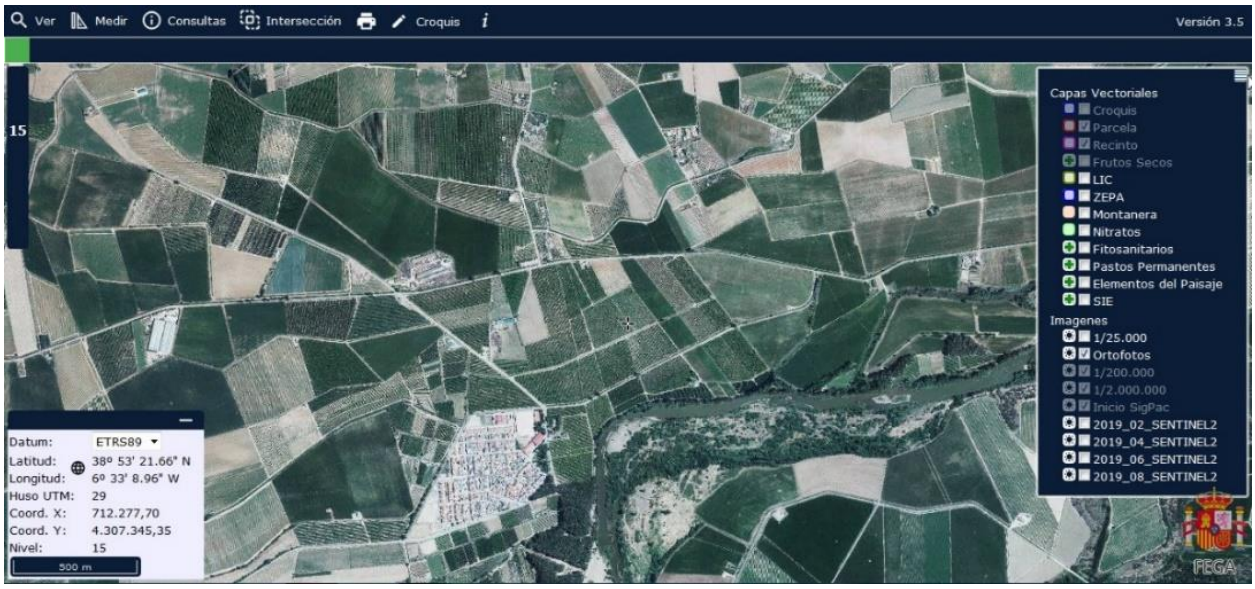


Ilustración 5. Interfaz de la plataforma SigPac del Gobierno de España, en esta se puede observar su función como servidor de mapas con la interfaz ubicada a la derecha con todas las capas disponibles y activables[7].

- **Facilitar a los agricultores ciertos trámites** al disponer de todos los datos
- **Facilitar controles administrativos** al estar toda la información digitalizada
- **Facilitar el control del terreno** permitiendo visitas rápidas y agilización de localización.

Este proyecto no se detuvo aquí, pues fueron múltiples comunidades las que desarrollaron su propio sistema. Un ejemplo de los que se encuentra más mantenido y actualizado fue en Andalucía donde desde la Junta de Andalucía (Consejería de Agricultura, Pesca, Agua y Desarrollo Rural) se desarrolló su propio visor SigPac. No obstante, en la comunidad de Madrid, Castilla la Mancha, etc. Se dispone también de una versión SigPac con la misma interfaz, pero siendo los datos limitados a la comunidad.

Manteniendo esta idea, pero sin estar especificado en el ámbito agrario existen otros servicios como son:

- **Goolzoom**: Aplicación web de acceso mediante pago al ser de empresa privada especializada en búsqueda de datos catastrales, entre sus usos permite el uso de un visor para consultas de catastro, topografía… Mediante acceso directo a Google Maps, permite la descarga de mapas, mediciones de superficies, perfiles y muchas más operaciones. En adición al visor es una

[7] https://sigpac.mapama.gob.es/fega/visor/

aplicación para gestión de urbanismo, catastro y datos como propietarios de propiedades.

- **IBERPIX:** Aplicación web del IGN publicada en el año 2015 con el objetivo de proporcionar una herramienta que facilite la visualización y localización de lugares españoles. Ofrece la posibilidad de visualizar a distintas escalas, ofrece capas vectoriales y otros formatos además de herramientas de medición. Incluye una gran cantidad de cartografía destacando el histórico PNOA (Plan Nacional de Ortofotografía Aérea).

- **SignA**: Aplicación web del IGN publicada en el año 2015 como un sistema de información geográfica nacional, es de libre uso y consulta. Está dotado de herramientas de visualización, navegación y búsqueda de mapas, mediciones, reporte de errores. El portal cuenta con diferentes sistemas de referencia de coordenadas y permite consultas espaciales, cálculo de rutas… Además, está dotado de servicios WMS (Web Map Service), WFS (Web Feature Service), CSW (Catalogue Service for the Web), etc. Con el fin de poder realizar consultas de datos.

Previo a todos estos sistemas donde se muestra la información visual de mapas y capas se debe realizar un preprocesamiento de todo este tipo de información espacial para que esta pueda ser usada correctamente, todo este trabajo previo se realiza con múltiples programas y aplicaciones desarrollados con estos fines. Estas herramientas se conocen principalmente como Software SIG (Sistema de Información Geográfica), estos se refieren a sistemas utilizados para recopilar, gestionar y analizar datos, especialmente los georreferenciados.

Este tipo de software permite al usuario realizar operaciones con la información espacial para poder dejarla en un formato perfecto para su presentación final. Estos programas suelen estar dotados de herramientas que permiten juntar capas de información vectorial, adaptarlas a zonas geográficas delimitadas por el usuario, almacenar datos temáticos sobre los mismos. Además permiten trabajar con información de tipo ráster o imágenes satelitales que permitan la visualización del terreno. Dentro de los Software GIS hay varios que destacan sobre el resto por su popularidad y su uso estos son:

- **ArcGIS**: Es unos de los software más conocidos y utilizados del mundo, perteneciente a Esri una de las empresas más grandes dedicadas al campo

de los SIG. Este software ofrece una gran cantidad de posibilidades, pero su uso es de pago.

- **QGIS**: Es un software de código abierto que ofrece muchas herramientas para análisis de la información espacial, así como para facilitar su utilización en servidores geoespaciales web. Cuenta con una gran cantidad de plugin para aumentar las posibilidades del mismo.

- **Google Earth**: Es uno de los softwares más conocidos, aunque este se especializa de manera más exclusiva en la visualización en alta resolución y modelado tridimensional ya que no permite extraer ni trabajar con facilidad con los datos.

Existen múltiples softwares SIG en la web algunos más útiles y completos que otros, sin embargo, entre los mejores disponibles las diferencias son bastante limitadas y la decisión dependerá de las necesidades y los recursos disponibles, así como del presupuesto.

1.2.3 Identificación automática de entidades

Como se ha comentado en apartados anteriores históricamente han existidos sistemas que almacenaban la información relativa a objetos muy concretos dentro de sistemas de información espacial, un ejemplo puede ser el SigPac que trabajaba con los olivos individualmente hablando. Sin embargo, el mantenimiento de este tipo de información es muy costoso y complicado por ende su uso cayó en picado y dejaron de mantenerse y acabaron desapareciendo. Con el objetivo de solventar esta carencia y poder volver a tener un acceso automático y rápido a objetos como estos se busca poder tener una forma de conocer sus ubicaciones de la manera más eficiente posible. Los recursos disponibles son principalmente imágenes resultantes de vuelos de dron, así como imágenes satelitales de libre acceso.

Una vez se conocen los recursos se debe recurrir al uso de distintos métodos y algoritmos para poder solventar este problema. Estas técnicas pueden ser desde la segmentación de imágenes, aprendizaje supervisado, template matching o blob detection (reconocimiento de regiones) [6].

Todos estos posibles algoritmos coinciden en el hecho de que realizan un procesamiento de las imágenes disponibles y trabajan con distintas herramientas dentro de las mismas con el fin de poder detectar las zonas que difieren unas entidades de otras,

pudiendo así conseguir distinguir las mismas y posicionarlas dentro de las zonas delimitadas.

Para tratar con este tipo de situaciones y poder aplicar métodos de estos tipos existen librerías y programas que tienen ya implementadas funciones del estilo para poder procesar imágenes. Un ejemplo de esto es OpenCV[8], una biblioteca de código abierto especializada en visión artificial y aprendizaje automático que se adapta a múltiples lenguajes de programación, aunque inicialmente esté desarrollada en C++. Esta librería tiene múltiples algoritmos y funciones de análisis de imágenes, extracción de contornos, etc. Estos permiten trabajar sobre los recursos y con mecanismos de clasificación o clustering realizar una segmentación de entidades en las imágenes.

Además de OpenCV que es de las más conocidas existen otros múltiples programas, algoritmos o plataformas como son:

- **Clarifai**: Empresa de Inteligencia Artificial especializada en la visión artificial basado en aprendizaje automático y redes neuronales profundas. Dispone de una API que permite integrarla en cualquier aplicación desarrollada. Su rendimiento general depende como en el resto de casos de la calidad de las imágenes, así como de la cantidad de datos disponibles para su entrenamiento.

- **ArcGIS**: Software de SIG utilizado para análisis y visualización de datos geoespaciales, además de esto presenta una gran variedad de herramientas para procesamiento de imágenes como corrección atmosférica, fusión o clasificación de imágenes, etc. Esto facilita la detección de entidades en adición a las múltiples posibilidades de adición de datos sensoriales, satelitales o de campo.

Estas no son las únicas herramientas disponibles y que facilitan el proceso de identificación automática de entidades, en la web hay muchas más disponibles, por ende, la decisión final de cual implementar dependerá de muchos factores que se tendrán que valorar previo al comienzo del proyecto según los requisitos del mismo.

1.2.4 BBDD espaciales

Los datos son un bien muy preciado y como hemos visto anteriormente en ocasiones difícil de gestionar y obtener, es por esto que una vez disponemos de estos es muy

[8] Código obtenido del repositorio público https://github.com/opencv/opencv

importante poder almacenarlos de manera correcta y tener un acceso eficiente a los mismos.

En cualquier otro caso el almacenamiento y manejo de los datos sería una tarea sencilla a realizar exclusivamente con una base de datos común, esta tarea puede llegar a complicarse y ser compleja una vez se empieza a trabajar con grandes cantidades de datos como es el caso de este proyecto, pero en adición la casuística de este proyecto es que se trabaja con datos georreferenciados por ende hay una serie de información que en una tabla básica de una base de datos común no puede almacenarse de manera correcta ni eficiente.

Con el fin de solventar este problema y poder trabajar con todo tipo de datos surgen las Bases de datos espaciales [7] las cuales cumplen con todas las capacidades de una base de datos normal, pero incluyendo operaciones con datos espaciales, así como consultas especializadas para este tipo de datos. Este campo sigue en expansión y con el tiempo se siguen buscando formas de mejorar los sistemas existentes [8], son además múltiples los sistemas de bases de datos que ya permiten con facilidad incorporar datos de este tipo a sus sistemas, así como asociarse con sistemas GIS.

Algunos de estos sistemas de bases de datos son:

- **PostgreSQL**: Con una extensión denominada PostGIS este sistema de bases de datos agrega un soporte para poder trabajar con datos espaciales complejos, permite entre otras acciones búsquedas por proximidad, intersecciones, uniones…

- **Oracle Database**: Con una extensión denominada Oracle Spatial es capaz de proporcionar a las aplicaciones funciones avanzadas para manejo de datos 2D y 3D, así como permite analizar y trabajar con estos datos incluyendo creación de aplicaciones basadas en mapas.

- **MySQL**: Es una base de datos de código abierto que cuenta con una extensión Spatial que permite almacenar y manipular datos geoespaciales, así como aplicar funciones espaciales.

- **GeoServer:** Es una aplicación más especializada en servidores de mapas, de código abierto y puede ser empleada como backend para bases de datos espaciales, este sistema está más especializado en visualización y publicación de datos espaciales.

1.3 Contexto del problema

El trabajo realizado no dispone de un sistema previo utilizado como base. Si no que se ha diseñado e implementado desde cero, con el fin de ajustarse a las necesidades requeridas. Por este motivo durante el desarrollo de este apartado se realiza una pequeña expansión de la Sección 1.2 con el objetivo de concretar y especificar la información sobre el entorno preciso en el que se enmarca este proyecto de manera más individual y reducida. Además se comentará lo que este proyecto busca aportar al ecosistema de investigación y agricultura actual así como la problemática existente en el sector agrícola que llevo a plantear este proyecto como un posible trabajo a realizar.

1.3.1 Descripción del entorno actual

El objetivo de este trabajo no es otro que el desarrollo de una aplicación web especializada en el sector agrícola y más concretamente en el contexto del olivar jiennense. Como se ha comentado previamente no se dispone de ningún sistema de partida lo cual implica que es un campo aún por desarrollar y que puede tener un impacto de manera directa en distintos sectores socioeconómicos de la provincia de Jaén principalmente.

Históricamente siempre se ha conocido a la provincia de Jaén como una zona dedicada y con parte de su impacto económico proveniente de la producción de aceite y el tratamiento del olivar. En la siguiente ilustración se puede observar la gran superficie que supone dentro de nuestra provincia, colocándolo como la actividad agrícola principal.

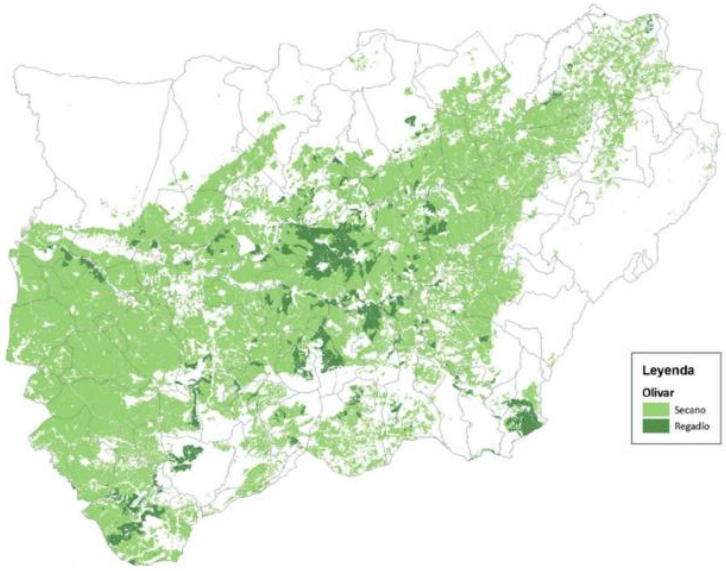


Ilustración 6: Mapa que muestra la superficie de olivar en la provincia de Jaén[9].

[9] https://digibug.ugr.es/bitstream/handle/10481/27511/821_Cap_44.pdf

En adición a los datos de terreno podemos sumar los datos económicos por los que se considera una actividad de interés para fomentar y mejorar en nuestra provincia. A través del ministerio de agricultura, pesca y alimentación es posible acceder a los datos de la campaña Oleica del 2021[10] donde se puede observar que Jaén produjo 524.798,2 toneladas de aceite frente a las 1.389.970,3 toneladas totales de la península siendo esto un 37.76% de la producción nacional, lo que demuestra que es un sector de trabajo y económico muy importante para la región.

Es por estos motivos que, en primer lugar, el desarrollo de este trabajo afecta de manera directa al sector de la agricultura y por ende a los poseedores y / o trabajadores de los distintos terrenos de olivar ubicados dentro de la provincia, más concretamente en el municipio de Marmolejo. Con el desarrollo de esta aplicación web se busca reducir el esfuerzo necesario para obtener el mayor rendimiento posible a las parcelas estudiadas, principalmente a través de la mejora de las condiciones de las mismas al obtener una mayor cantidad de información con la que trabajar tras realizar un correcto pre preprocesamiento de los datos. De esta manera se busca poder impactar en este sector aportando el mayor número de funcionalidades posibles con el fin de aumentar sus recursos y posibilidades en el manejo de sus propiedades.

Un ejemplo de un proyecto similar es el comentado anteriormente proyecto SigPac propuesto por el gobierno de España, este tenía el objetivo de facilitar a los propietarios las gestiones administrativas que se debían realizar con las propiedades disponibles a la hora de obtención se subvenciones o declaración de superficies. Esta aplicación no obstante va más allá pues además de disponer de los datos de las declaraciones de los terrenos busca poder aportar una mayor utilidad al usuario siendo capaz de obtener información sobre el terreno no con un fin puramente administrativo si no con el objetivo de poder mejorar el trabajo sobre la misma.

Otro aspecto a considerar en el desarrollo del proyecto es la necesidad de segmentar modelos 3D e incluso identificar patrones en imágenes texturizadas con el fin de poder estimar la posición de las plantaciones en las diferentes zonas de estudio. Para resolver esta problemática se pueden utilizar técnicas de visión por computador, algoritmos de procesamiento de imágenes o incluso modelos de redes neuronales [9].

[10] Avances e Informes de Situación de Mercado del Sector del Aceite de Oliva y la Aceituna de Mesa (mapa.gob.es)

1.3.2 Resumen de las deficiencias y carencias identificadas

En la actualidad hay a disposición del sector agrónomo información bidimensional sobre las fincas y terrenos del territorio nacional principalmente en un ámbito más administrativo. Sin embargo, la tecnología se ha incrementado y no se ha observado un cambio real que permita acceder a información en tres dimensiones de manera pública y asequible para todo el mundo. La visualización de escenas 3D nos ofrece una nueva forma de representar objetos y obtener mayor nivel de realismo. Trabajar con objetos complejos con una determinada altura, volumen y dimensiones, es de gran ayuda para tomar decisiones más informadas y optimiza el diseño y control de la plantación. En contrapartida el acceso a los datos de elevación del terreno sí que se encuentran accesibles en la red.

Esta deficiencia presentada se pretende solventar mediante el modelado tridimensional de los distintos lugares disponibles mediante el escaneado con sensores LiDAR tras el vuelo de un dispositivo UAV. La utilización de estos sensores nos permite generar una nube de puntos 3D del ecosistema. En el caso de aquellas fincas sobre las que aún no se tenga información LiDAR, el modelado 3D se realiza mediate la extracción de la textura pública de la parcela, así como del DEM (Digital Elevation Model) asociado. De esta forma se consigue una simulación lo más afín posible a la elevación y visualización real, pudiendo de esta forma extraer todos los datos posibles de la misma para trabajar con ella sin la obligatoriedad de disponer de la nube de puntos.

En segundo lugar, existe una deficiencia en el control y mantenimiento de una de las partes más importantes del correcto desarrollo de una parcela agrícola como es el control y supervisión del árbol y/o fruto en cuestión de la misma, siendo en nuestro caso el olivo. Como se comentó anteriormente en un momento inicial se disponía de la ubicación y cierta información adicional de este tipo de entidades, sin embargo, el mal mantenimiento y el coste de este acabaron por dejar al sector sin un sistema preciso de ubicación de este tipo de entidades.

Esta carencia se pretende solventar mediante el uso de técnicas y algoritmos de visión por computador que nos permitan realizar la detección de las plantaciones existentes en cada parcela y almacenar su información para futuras consultas. Una vez realizada la detección de las diferentes plantaciones en las imágenes. Realizamos una proyección del pixel de la imagen hacia nuestro modelo digital del terreno. Esto nos permite obtener la posición exacta de cada plantación en nuestra parcela

1.4 Hipótesis y restricciones

El TFG se define como una asignatura de 12 créditos, lo que supone que la duración total del proyecto será de 300 horas, incluyendo todas las etapas del ciclo de vida, con la excepción del mantenimiento. Por consiguiente, la principal restricción aplicable es la limitación de la duración del trabajo.

En cuanto a las restricciones e hipótesis a nivel de aplicación nos encontramos las siguientes:

- **La dimensionalidad de las nubes de puntos**. Debido al despliegue del sistema como una aplicación web a la cual se accede mediante el uso de un navegador las capacidades de procesamiento se ven muy limitadas en cuanto al uso de nubes de puntos con una gran densidad de puntos. Como se verá en el futuro en los apartados de desarrollo y experimentaciones la aplicación funciona correctamente teniendo que procesar hasta 3 millones de puntos (con nubes originales de hasta 100 millones)**.** Sin embargo, al introducirse como objetivo la funcionalidad de la aplicación en varios dispositivos distintos como pueden ser Tablet o móvil se tiene que reducir la densidad de estas a tamaños de 1 - 2 millones de puntos para obtener tiempos asumibles.

- **La representación tridimensional mediante texturas**. Pese a no ser una restricción estricta obtener una representación lo más afín posible de la textura con la realidad es un efecto complejo debido a la dependencia de la precisión de los modelos de elevación del terreno disponibles, una vez se dispone de un modelo concreto por parcela es necesario aplicar un terreno por mapa de altura, en estas sucesivas operaciones se pierde precisión y por ende aunque se puede obtener un resultado aceptable para el prototipo las alturas podrán variar en caso de mezcla con una nube de puntos obtenida mediante vuelo con sensores LiDAR.

1.5 Objetivo General

El principal objetivo por el que se propone este trabajo es la necesidad de una aplicación web que integre datos multisensoriales con el fin de facilitar diferentes tareas a los agricultores para conseguir una mejor monitorización de fincas, a lo largo de este proyecto nos hemos centrado en las fincas del olivar.

Para poder lograr dicho objetivo general y conseguir así satisfacer a todos los clientes poseedores de fincas es necesario además disponer de objetivos específicos.

1.6 Objetivos Específicos

Una vez comentado el objetivo general y el entorno que rodea el trabajo se deben delimitar los objetivos específicos dentro del proyecto.

- **Desarrollar una aplicación web fácilmente accesible** y que cumpla los estándares de usabilidad, con el fin de conseguir una experiencia asequible y satisfactoria para cualquier usuario que quiera disponer de los servicios de esta.

- **Diseño de un entorno 3D capacitado para la carga y trabajo con los modelos disponibles** con el fin de poder aplicar distintos algoritmos de procesamiento de los mismos, de esta manera se busca obtener nueva información a raíz del post-procesamiento realizado a cada modelo. Estos modelos serán principalmente nubes de puntos resultado de la capturada y procesamiento de datos LiDAR (Light Detection and Ranging) obtenidos mediante el sobrevuelo de las fincas con el uso de UAVs capacitados con sensores LiDAR y multisensoriales.

- **Diseñar los algoritmos necesarios para dotar al modelo tridimensional de múltiples usos**, siendo el usuario de esta manera capaz de operar con el mismo pudiendo acceder a la distinta información que nuestro sistema es capaz de ofrecer sobre la finca en cuestión sobre la que se trabaja, a su vez para poder conseguir un buen manejo del modelo será necesario el desarrollo de una interfaz de usuario intuitiva y que permita modificar los distintos parámetros del modelo para un correcto manejo.

- **Dotar a la aplicación de un entorno integrado con un esquema de datos completo y enriquecido**, que permitan al sistema trabajar de manera satisfactoria con todos los recursos de los que dispone mediante el acceso a una base de datos modelada y enriquecida con datos regionales y nacionales disponibles de manera pública, con esto se busca poder trabajar de manera automática con el sistema al encontrarse este adaptado y capacitado para trabajar con cualquier parcela sobre la que se desee realizar un procesamiento avanzado de los datos.

- **Desarrollar algoritmos que permitan a la aplicación obtener nuevo conocimiento** a partir del uso de modelado tridimensional y técnicas de ortofotografía del terreno, esto se conseguiría gracias a métodos de aprendizaje y segmentación sobre las distintas fincas, consiguiendo expandir los datos disponibles a un mayor nivel de alcance. También se procederá con el desarrollo de algoritmos geométricos que permitan interaccionar con las fincas pudiendo recortar o acceder a zonas más concretas de las mismas.

- **Proporcionar a la aplicación web y principalmente al modelo tridimensional la capacidad de adaptarse a su uso en cualquier dispositivo**, siendo de esta manera la aplicación usable en cualquier tipo de dispositivo tecnológico teniendo como única limitación la necesidad de acceso a un punto en la red, de esta manera se busca conseguir un modelo e interfaz usable tanto en un equipo con periféricos disponibles, así como en un sistema táctil y más limitado.

1.6.1 Objetivos secundarios

En este apartado se presentan aquellos objetivos necesarios para cumplimentar cada uno de los objetivos principales.

- **Solapamiento de los distintos modelos de información** tridimensionales en la escena generada consiguiendo así obtener nueva información mediante la combinación de estos modelos.

- **Generación e investigación de algoritmos de agrupamiento y clasificación** con el fin de segmentar no solo imágenes sino también nubes de puntos para la corrección u obtención de información sobre el terreno.

- **Desarrollo de la voxelización de los modelos tridimensionales** que lo permitan para optimizar el rendimiento de la aplicación en todos los distintos tipos de dispositivos.

- **Diseño de un sistema de control de usuarios e información** en el sistema desarrollado permitiendo acceder a información así como actualizarla cuando sea necesario y preciso.

- **Obtención y representación de datos históricos** en formato gráfico para permitir extraer información y conclusiones de forma visual y temporal.

1.6.2 Principales componentes del proyecto

En este apartado se definirán todos los elementos que componen el resultado final de este trabajo y que de manera conjunta a este documento conforman el proyecto final. Los elementos en cuestión serán los siguientes:

- **Código fuente** de la aplicación web, en este se implementan desde los algoritmos de detección que se describirán más adelante en C++, así como los algoritmos de gestión de una escena tridimensional en JavaScript, en adición para completar la aplicación se ha desarrollado la parte web para el correcto manejo del usuario con HTML y CSS.

- **Memoria del trabajo**, se corresponde con este documento donde se explica el proceso de desarrollo del prototipo de aplicación web desarrollado con el fin de que el lector pueda comprender el funcionamiento interno de la aplicación, así como las decisiones que se han tomado a lo largo del desarrollo de la misma. Finalmente, también incluye manuales de usuario e instrucciones para el correcto uso de la aplicación desarrollada.

- **Video demostración** del funcionamiento de la aplicación web, en este se muestra la forma deseada para trabajar con la aplicación, así como todas las funcionalidades disponibles en la misma.

- **Recursos adicionales**, en este apartado se incluyen elementos como son librerías descargadas para el correcto funcionamiento de utilidades del código y de la visualización de la aplicación, ficheros con información de modelados tridimensionales y / o nubes de puntos, texturas u imágenes con información multiespectral para el análisis del terreno.

1.7 Metodología de desarrollo de software

La metodología de desarrollo de software es un conjunto de principios, prácticas y procesos que se utilizan para planificar, diseñar, construir, probar y entregar software de alta calidad. Cualquier proyecto en sus inicios por simple que pueda parecer debe basar su desarrollo en algún tipo de metodología según las características del mismo, existen múltiples tipos de metodologías y es por esto que hay que conocer muy bien los objetivos del proyecto para elegir el modelo de organización correcto.

Este proyecto se encuentra enmarcado dentro de un departamento de informática es por este motivo que existe tanto una parte interesada como una parte que realiza el trabajo y desarrolla las peticiones, esta estructura nos permite desarrollar un trabajo en un ámbito de colaboración, que integrado con una serie de reuniones para poder realizar revisiones y estudiar los avances nos permiten plantear una metodología iterativa y ágil. La metodología más común y que mejor se adapta a esta definición es la metodología Scrumban que integran dos metodologías muy populares como la Scrum y la Kanban [10].

Para poder entender lo que esta metodología nos facilita es necesario conocer cómo funcionan por separado las otras dos.

Kanban [11] (Ilustración 7) es una metodología que nos permite agrupar las tareas planteadas de un proyecto en una serie de fases previamente definidas para poder ubicar cada una en su estado dentro del proyecto. Las ventajas que ofrece este sistema por ende son:

- Este formato no sigue una planificación temporal, sino que busca un proceso continuo donde unas tareas pueden depender de otras y por ende su desarrollo o el avance de las mismas dependerá de los estados dentro del tablero de otras.

- Las tareas dentro de un mismo proyecto pueden contener un valor de prioridad permitiendo de esta manera priorizar la realización de unas sobre otras.

- Cada persona dentro del proyecto tendrá que responsabilizarse no solo de realizar sus tareas asociadas sino también de mantener actualizado el tablero de tareas con su estado actual para poder hacérselo saber al resto de integrantes.

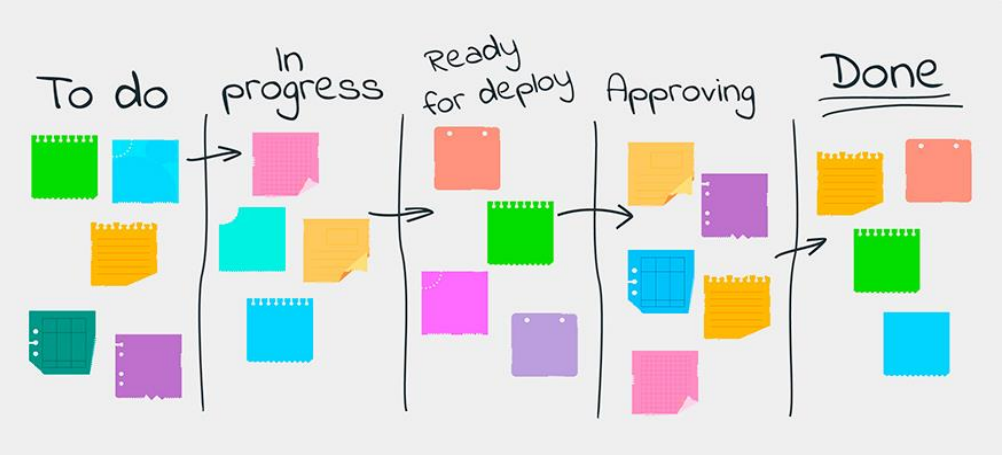


Ilustración 7: Ejemplo de tablero Kanban de organización por estados de las tareas, cada tarea es representada por una nota[11].

La metodología **Scrum** [12] **(**Ilustración 8**)** es una metodología de gestión de proyectos que basa su funcionalidad en la división del trabajo en el tiempo mediante los denominados **sprints o iteraciones,** cada uno de estos tiene asociados unos objetivos y/o tareas a cumplir para un periodo de tiempo determinado consiguiendo así una planificación por etapas iterativa. Los aspectos más importantes de este tipo de metodología por ende serán:

- Esta metodología requiere de varias retroalimentaciones periódicas en cada sprint siendo dos obligatorias una al inicio del sprint que permita delimitar los objetivos que se plantean durante el periodo de este mismo y otro al final o inicio del siguiente que nos permita valorar la productividad y grado de satisfacción del sprint finalizado. En adición es recomendable recibir retroalimentaciones también en periodos intermedios del mismo por posibles atascos o cuellos de botella.

- Scrum se caracteriza también por definir nada más empezar unos roles claramente estipulados como son

    - **Equipo de desarrolladores** que debe organizarse individualmente para cumplir las tareas que se les asignen.

    - **Product Owner,** se corresponde al cliente o la persona encargada de determinar los requisitos del proyecto, debe participar activamente en todo el proyecto.

11 https://www.tirsomaldonado.es/diferencias-entre-kanban-sistema-kanban-y-metodologia-kanban/

- **Scrum Master,** es el coordinador experto en este tipo de metodología y que debe gestionar su correcto desarrollo, imprevistos, cambios necesarios…

- La metodología Scrum además gracias a su control por iteraciones o sprints permite reducir futuros riesgos al completar y probar las distintas partes del proyecto, así como conseguir que el Product Owner pueda ir probando las funcionalidades en versiones parciales sin la necesidad de disponer del resto del proyecto.

Ilustración 8: Esquema básico de la metodología Scrum[12].

Como la fusión de estas dos metodologías surge **Scrumban** [13] que nos permite obtener lo mejor de cada metodología para conseguir la mejor planificación y desarrollo.

Característica	Scrum	Kanban	Scrumban
División por tiempo	Si, sprints semanales	No, basado en eventos	Si, sprints regulables de tiempo
División por eventos	No, los sprints son irrevocables	Si, trabajo continuo por tareas	Si, trabajo continúo ajustado a la planificación temporal
Estimaciones	Previas a cada sprint	Opcionales	Opcionales
Métricas de rendimiento	Burndown	Diagrama de flujo	Tiempo promedio de cumplimiento

[12] https://www.bocasay.com/scrum-method-benefits-web-development/

Límites de alcance	Temporales por sprint	No se puede tener en proceso más actividades de las posibles	Límites de actividades en progreso y de tareas pendientes
Rutinas de trabajo	Propietario asigna tareas a los miembros	Los miembros escogen tareas del tablero	Propietario asigna tareas al tablero y miembros escogen
Reuniones	Previas a los sprint y para revisiones intermedias	No necesario	Planificadas por demanda
Retroalimentación	Posterior a cada sprint	Opcional	Por eventos cortos
Roles	Propietario, Scrum Master, Miembros desarrollo	No necesario	No necesario

Tabla 2: Comparativa tipos de metodologías de desarrollo.

De cara a este proyecto esta combinación de metodologías ha sido de utilidad y ha facilitado el proceso, sin embargo, no se ha realizado un seguimiento estricto de las metodologías y se ha adaptado al proyecto en el que nos encontramos aplicándose de la siguiente manera:

- El proyecto se va definiendo progresivamente mediante el uso de iteraciones o sprints (**Scrum**) que podrán tener una duración variable, dentro de cada iteración se definirán una serie de tareas que en función de la situación en la que se encuentren dentro proyecto tendrán un estado, para esto se dispondrán de un tablero que nos permita organizar su estado (**Kanban**).

- Se diferirá brevemente del sistema de roles establecido por Scrum ya que dado el entorno del trabajo se permite a todos los integrantes (tutores y desarrollador) proponer ideas, mejoras o tareas a desarrollar.

- Las reuniones no estarán estrictamente estipuladas pudiendo variar en periodos de 2-6 semanas entre todos los miembros del proyecto, en estas se buscará declarar posibles nuevas tareas para futuras iteraciones, así como evaluar cuán satisfactoria ha sido el desarrollo de las presentes en la iteración en la que nos encontremos.

- La flexibilidad a posibles cambios o retrasos de tareas ha sido incrementada sobre todo en tareas que no tenían influencia directa en otras futuras, así

como en tareas que tras finalizarse recibían un cambio de perspectiva, de manera que era posible postergar estas a futuros sprints.

Para el desarrollo de esta metodología y como se comentará más adelante en el Capítulo 4, se ha utilizado el software online Jira que nos permite organizar el proyecto mediante la metodología Scrumban al otorgarnos un fichero backlog donde definir tareas, un tablero Kanban básico con todos los estados que queramos definir además de la posibilidad de definir Sprints o iteraciones y finalizarlas cuando así se desee.

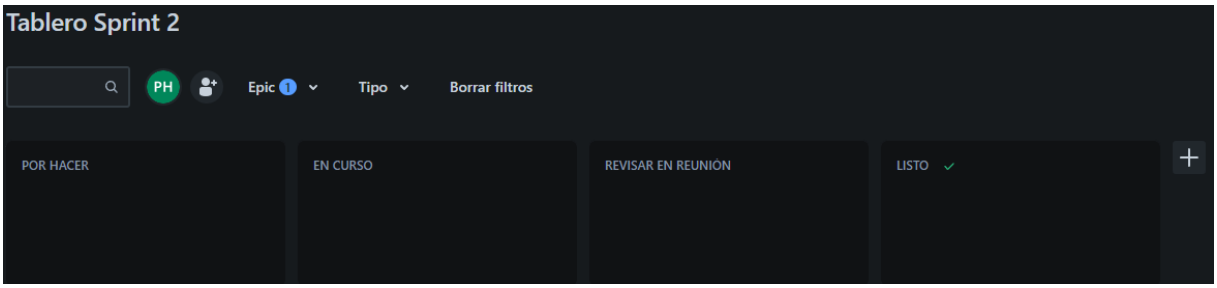


Ilustración 9: Formato de tablero Kanban empleado en cada iteración del proyecto (autor).

1.8 Estructura de la memoria

El fin de esta memoria es el de poder explicar todo el proceso de generación de un proyecto como la aplicación presentada. Para facilitar la comprensión de la memoria en este apartado se detalla la estructura que la misma va a seguir en cuanto al desarrollo de todo el proceso y los pasos seguidos.

Esta documentación se encuentra dividida en 9 capítulos que al comenzar disponen de una introducción para poder dar a entender el porqué de estos.

1. El primer capítulo se corresponde con el mismo en el que nos encontramos, este se desarrolla con el fin de poner en situación al lector sobre las especificaciones más técnicas del proyecto así como toda la historia previa relacionada de alguna manera con este proyecto.

2. El segundo capítulo se corresponde con la planificación del proyecto, en este se alberga la descripción de las tareas que acarrea la realización de la aplicación, así como su distribución temporal. Además se incluyen las planificaciones de presupuesto y costes de personal.

3. El tercer capítulo correspondiente al análisis del sistema con todas los aspectos técnicos o necesarios para la obtención de un sistema correcto,

dentro de este se presentarán los sketch, wireframe o mock ups así como los diagramas necesarios para el sistema como la base de datos o diagramas de casos de uso. Además se estudiarán los posibles riesgos que pueden surgir como los requisitos del proyecto.

4. El cuarto capítulo describe las herramientas y recursos necesarios que se han empleado en el desarrollo del proyecto así como las posibles alternativas en caso de no poder acceder a estas. En adición en este capítulo se incluirá una propuesta de solución completa de la aplicación.

5. El quinto capítulo contiene todo el proceso de desarrollo del trabajo realizado incluyendo las seis iteraciones y las descripción de todas sus tareas en profundidad con el fin de dar a conocer al usuario el proceso necesario para obtener una aplicación como la desarrollada.

6. El sexto capítulo se desarrolla como la última parte relativa a la redacción propia del groso del desarrollo del proyecto albergando las conclusiones obtenidas tras la realización de este trabajo y las posibles mejoras en futuros trabajos.

7. El septimo capítulo alberga las definiciones y abreviaturas menos comunes utilizados a lo largo de este documento, estas se describirán con el fin de dar a entender al lector el significado de estas.

8. El octavo capítulo se corresponde con la bibliografía de la documentación albergando todos los artículos, páginas o documentos utilizados como referencia en la redacción de este documento.

9. El noveno capítulo contiene los distintos apéndices necesarios para la correcta comprensión de todo el proyecto, en este capítulo podemos encontrar la guía del TFG, así como unas guías de instalación para hacer uso del proyecto y un manual de usuario para manejar la página.

1.9 Normas y referencias

A continuación, se presentan las normas, reglamentos y referencias de cualquier tipo de aplicación en la elaboración del trabajo y/o en la puesta en práctica del mismo.

1.9.1 Mecanismos de control de calidad

El aseguramiento de la calidad en la redacción del trabajo implica la verificación de la completitud (falta de omisiones), la integridad de la documentación, así como una redacción clara, concisa y entendible por todos los participantes e interesados en dicho trabajo.

Al estar el trabajo desarrollado por una sola persona y verificado por un/a tutor/a, no es necesario llevar un documento de control de edición y revisión de la documentación. De esta forma, se han utilizado mecanismos básicos para la verificación de la integridad y completitud, que incluyen un control de versiones a nivel de documento y un control sencillo de la trazabilidad de especificaciones y requisitos.

Capítulo 2: PLANIFICACIÓN

En este Capítulo se presenta la planificación que este proyecto ha seguido desde la presentación de las tareas estipuladas a realizar, así como su distribución en el tiempo. Por último, se detallarán en los recursos necesarios para la elaboración de este trabajo si fuera necesario la contratación y obtención de recursos para su realización. Además se realiza el presupuesto y la estimación del coste personal y temporal del proyecto para un posible ámbito más profesional.

2.1 Identificación de tareas

Este apartado tiene como base el detallar la planificación que se ha seguido para el desarrollo de este proyecto que se enmarca en un Trabajo de fin de grado y que por ende se encuentra limitado a 300 horas. Como bien se ha comentado previamente la metodología seguida es iterativa por ende se mencionarán a continuación las iteraciones que se han ido incluyendo a lo largo del proyecto, así como las tareas o hitos que se han planteado en cada una de estas y la estimación lo más exacta posible del tiempo necesario para su realización.

Cabe destacar que durante el desarrollo del proyecto se han ido completando diferentes apartados de esta documentación, dejando una etapa final de documentación y bibliografía para poder terminar de desarrollar las partes más técnicas del mismo y redactar el groso del documento. Por ende, durante las tareas de las iteraciones se omitirá la mención de este aspecto.

- **Previos del proyecto:** En primer lugar, se emplean una serie de días desde el planteamiento del proyecto para poder determinar los objetivos iniciales, así como poder establecer un primer contacto con el tipo de proyecto y las tecnologías a utilizar, recopilando a su vez también información sobre posibles trabajos existentes relacionados y como se desarrollaron. Esta etapa no forma parte del desarrollo del sistema es meramente introductoria.

- **Iteración 1:** Comienzo del desarrollo de la aplicación web. Esta iteración tendrá como trasfondo principal el planteamiento de la interfaz, el inicio de su desarrollo además de la investigación sobre el funcionamiento de nuevas tecnologías a utilizar en la misma, la obtención de gran parte de los datos necesarios y diseño de la base de datos.

- Desarrollo de distintas guías visuales (wireframes y mock up) desarrolladas para conseguir un aspecto de la interfaz de usuario satisfactoria final y un esquema sobre el que trabajar.
- Diseño e implementación de un sistema de gestión de base de datos.
- Inicio del desarrollo de la aplicación web:
  - Creación de un sistema de directorios para la estructuración de la aplicación.
  - Vinculación de la librería Bootstrap y generación de los archivos necesarios.
  - Desarrollo de una interfaz básica inicial y un sistema de inicio de sesión.
  - Desarrollo de una página exclusiva para el modelado tridimensional y que soporte interacción básica con el usuario.
- Búsqueda y obtención de datos espaciales necesarios para enriquecer la aplicación web.

- **Iteración 2:** Una vez finalizada la primera iteración el proyecto girara entorno a la integración de los modelos de datos disponibles en la escena, el trabajo con la información espacial y continuar con el desarrollo de la aplicación, para todo esto se definen las siguientes tareas:
  - Pruebas con el motor tridimensional BabylonJS y su formato de integración en una aplicación web
  - Trabajo con los modelos tridimensionales de nubes de puntos con el fin de obtener una exportación de estos en el formato adecuado.
  - Desarrollo de las conexiones e interacciones básicas de la aplicación web, permitiendo conectar todas las páginas entre sí e interactuar con las mismas.
  - Diseño de un proyecto en QGIS que nos permita administrar la información espacial disponible para obtener el mejor formato de esta.

- Desarrollo de un mapa interactivo que permita al usuario mover y seleccionar cualquier parcela de las disponible entre los datos.

- **Iteración 3:** Esta iteración consiste en la integración de los datos en una base de datos y su vinculación con la página web, sumado al desarrollo de una GUI para el modelado y funcionalidades básicas del mismo, estos objetivos se dividirán en las siguientes tareas:

  - Creación de un sistema de carga de nubes de puntos para la aplicación web, así como herramientas básicas para su interacción.

  - Carga de datos temáticos y espaciales en la base de datos y vinculación de esta con la aplicación.

  - Pruebas e integración de un servidor de mapas con Geoserver y OpenLayers.

  - Pruebas con la librería OpenCV de C++.

- **Iteración 4:** Esta iteración consistirá en el desarrollo de nuevos formatos para el enriquecimiento de la escena y mejoras en el uso de la interfaz de la aplicación. Se definen las siguientes tareas:

  - Integración de una capa de datos formada por texturas y dotada de altura mediante desnivel del terreno.

  - Extensión de los controles de los que dispondrá el mapa interactivo incluyendo operaciones que faciliten el control de este.

  - Desarrollo de algoritmos geométricos para gestión y edición de la nube de puntos e inserción de funcionalidades básicas para el manejo de la escena.

  - Integración de un Script para detección y segmentación de olivos en ortofotos en C++.

  - Mantenimiento de la aplicación para un rendimiento responsive y adecuado para otros dispositivos como Tablet o móvil.

- **Iteración 5:** Esta iteración consistirá en el desarrollo final del modelado, así como la ampliación del sistema con la adición de control de datos por interfaz, las tareas que conforman esta iteración son:
    - Desarrollo de un sistema de voxelizado para las nubes de puntos con el fin de optimizar funciones asociadas a la navegación sobre los puntos de la misma.
    - Diseño e implementación de una GUI que nos permita interactuar con la escena completa del modelado y las funcionalidades desarrolladas previamente.
    - Mejora del script desarrollado para detección de entidades con aplicación de nuevos algoritmos
    - Combinación de datos entre textura y nube de puntos con el fin de enriquecer la aplicación y el conocimiento que se puede extraer.
    - Desarrollo de una interfaz para la gestión de perfiles y datos de la base de datos basado en los distintos niveles de usuarios.
- **Iteración 6:** Esta iteración consistirá en el desarrollo final del proyecto, así como su mejora y puesta en marcha en un servidor, las tareas que conforman esta iteración son:
    - Inserción de datos temáticos finales en la base de datos e integración mediante gráficos en la aplicación web.
    - Mejora de la documentación y estructura del código, así como refinamiento de la aplicación.
    - Mejora del trato con el usuario:
        - Adición de sistemas de aviso para tener en todo momento al usuario al tanto de lo que ocurre en la aplicación.
        - Revisión del diseño y estilos de la interfaz y aplicación desarrolladas con el fin de conseguir la mejor experiencia posible para el mismo.

- Navegación final completa por la aplicación para identificación de posibles fallos inesperados.
- Integración de la aplicación web en un servidor Debian para garantizar su acceso en la web.

- **Documentación y bibliografía:** Esta "iteración" nos permite finalizar el groso de la documentación e indagación de recursos bibliográficos disponibles, no se corresponde con una iteración literal del desarrollo del sistema al igual que ocurría con la primera de planificación, sin embargo, corresponde con el final de este proyecto al terminar de redactarse este documento.

2.2 Planificación temporal

Una vez se han descrito todas las iteraciones y tareas que conformarán el proceso de desarrollo del proyecto es necesario aplicarles una distribución temporal.

La realización de este trabajo se encuentra enmarcada entre los meses de octubre de 2022 a junio de 2023. De este periodo se deberán excluir las vacaciones de navidad donde no se realizó trabajo y obtenemos el periodo de tiempo en el que se ha desarrollado el trabajo y del que disponemos para distribuir las iteraciones correspondientes.

La división final de tiempo podremos verla en la Ilustración 10 con un diagrama de Gantt mejor ilustrada y ubicada temporalmente. En cuestión de días la división ha quedado tal que:

- Previos del proyecto: **10 días**
- Iteración 1: **22 días**
- Iteración 2: **39 días**
- Iteración 3: **15 días**
- Iteración 4: **28 días**
- Iteración 5: **26 días**
- Iteración 6: **15 días**
- Documentación y bibliografía: **88 días**

Como podemos observar la distribución no es muy uniforme ya que hay iteraciones que disponen de un mayor número de tareas o estas son más complejas de realizar que otras que se pueden encontrar en iteraciones previas o posteriores. La duración de la última etapa es tan elevada porque la documentación es un proceso continuo, pero no principal por lo que se lleva una parte minoritaria del tiempo diario.

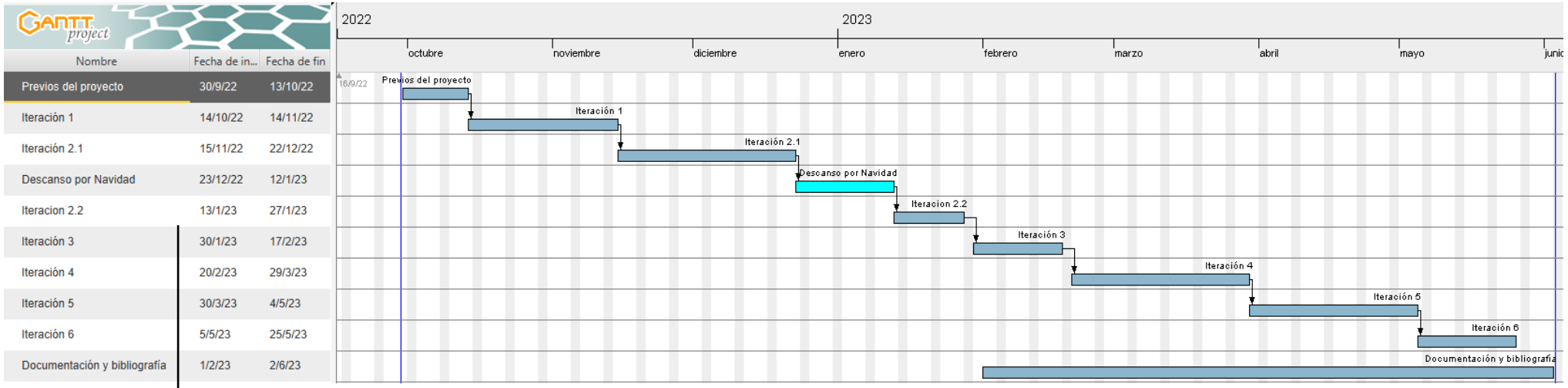


Ilustración 10: Diagrama de Gantt con la cronología seguida para el proyecto (autor).

Previos del proyecto	30/9/22	13/10/22
Iteración 1	14/10/22	14/11/22
Iteración 2.1	15/11/22	22/12/22
Descanso por Navidad	23/12/22	12/1/23
Iteracion 2.2	13/1/23	27/1/23
Iteración 3	30/1/23	17/2/23
Iteración 4	20/2/23	29/3/23
Iteración 5	30/3/23	4/5/23
Iteración 6	5/5/23	25/5/23
Documentación y bibliografía	1/2/23	2/6/23

Ilustración 11: Fechas de trabajo de cada iteración

2.3 Estimación de costes y viabilidad

Con la planificación temporal establecida y las tareas identificadas únicamente faltaría estimar los recursos y el presupuesto final necesario para abordar un trabajo como el elaborado durante este proyecto.

2.3.1 Estimación del tamaño y esfuerzo

Ya que el presente proyecto es un TFG, no existen restricciones de tipo económico, sino de tipo temporal (un número aproximado de horas). Por consiguiente, los cálculos de tamaño del proyecto están supeditados el tiempo disponible. En cuanto al esfuerzo, se dispone de tan un solo efectivo (la persona autora del trabajo).

Para la resolución de los datos de tamaño y esfuerzo necesarios para la realización de este trabajo se utiliza el modelo COCOMO (Constructive Cost Model) el cuál es un modelo matemático de base empírica para estimación de costes software. Este modelo fue desarrollado por B.W Boehm en la década de los 70-80 [14].

2.3.1.1 COCOMO

COCOMO incluye dentro del mismo distintos submodelos que permiten aplicarlo de distintas formas en función de lo que se quiera conseguir, estos son:

- **Básico:** Este modelo está destinado para el cálculo rápido de proyectos pequeños y medianos.
- **Intermedio:** Este modelo añade al modelo básico hasta quince modificadores opcionales para tener en cuenta el entorno de trabajo, con esto se consigue incrementar la precisión del resultado final.
- **Detallado:** Este modelo presenta dos mejoras con respecto al modelo intermedio, en primer lugar, se tiene en cuenta la fase del proyecto donde ocurren las estimaciones permitiendo ajustar los factores en función de la etapa a la que pertenecen. En segundo lugar, se establece una jerarquía de tres niveles de productos. Su clasificación será tal que:
    - Gran variación a bajo nivel: Nivel de módulo
    - Poca variación: Nivel de subsistema

- Restantes: Nivel de sistema

A su vez el COCOMO incluye tres tipos de desarrollo como son:

- **Orgánico:** Este tipo de desarrollo se encuentra orientado al trabajo de un pequeño grupo de programadores que desarrollan software. El tamaño de desarrollo del software debe ser de unos pocos miles de líneas en un entorno conocido por el equipo de desarrollo. No se busca obtener una gran innovación técnica.

- **Empotrado:** Este tipo de desarrollo suele darse en situaciones donde no hay una experiencia previa donde basarse ya que requiere de una gran innovación técnica e impone unas fuertes restricciones de rendimiento.

- **Semilibre:** Es un tipo de desarrollo que busca un punto medio entre los dos tipos anteriores, dependiendo el problema con el que se trabaje se podrá disponer de un grupo que mezcla desarrolladores experimentados y no experimentados.

A continuación, se indicarán los valores constantes para cada tipo de modelo COCOMO además de los valores asociados a los 15 atributos del modelo intermedio para la factorización del coste total.

Tipo	a	b	c	d
COCOMO básico				
Orgánico	2.40	1.05	2.50	0.38
Semilibre	3.00	1.12	2.50	0.35
Empotrado	3.60	1.20	2.50	0.32
COCOMO intermedio / detallado				
Orgánico	3.20	1.05	2.50	0.38
Semilibre	3.00	1.12	2.50	0.35
Empotrado	2.80	1.20	2.50	0.42

Tabla 3: Constantes para los modelos COCOMO según el tipo de proyecto.

Conductores de coste	Valoración					
	Muy bajo	Bajo	Nominal	Alto	Muy alto	Ext. alto
Fiabilidad del software	0.75	0.88	1	1.15	1.40	-
Tamaño de la base de datos	-	0.94	1	1.08	1.16	-
Complejidad proyecto	0.7	0.85	1	1.15	1.30	1.65
Restricciones de tiempo de ejecución	-	-	1	1.11	1.30	1.66
Restricciones de almacenamiento	-	-	1	1.06	1.21	1.56
Volatilidad de la máquina virtual	-	0.87	1	1.15	1.30	-
Tiempo de respuesta del equipo	-	0.87	1	1.07	1.15	-
Capacidad del analista	1.46	1.19	1	0.86	0.71	-
Experiencia en la aplicación	1.29	1.13	1	0.91	0.82	-
Capacidad de los programadores	1.42	1.17	1	0.86	0.7	-
Experiencia en el sistema operativo utilizado	1.21	1.10	1	0.90	-	-
Experiencia en el lenguaje de programación	1.14	1.07	1	0.95	-	-
Prácticas de programación modernas	1.24	1.10	1	0.91	0.82	-
Utilización de herramientas software	1.24	1.10	1	0.91	0.83	-
Limitaciones de planificación	1.23	1.08	1	1.04	1.10	-

Tabla 4: Valores posibles a tomar por los conductores de coste COCOMO.

Una vez planteado el funcionamiento de este modelo matemático se debe determinar a qué tipo de desarrollo y que modelo de COCOMO pertenece nuestro problema, este podríamos decir que es del tipo de desarrollo **semilibre** ya que requiere de cierta innovación técnica y un mínimo de restricciones de rendimiento. Para el modelo se seleccionará el **intermedio** consiguiendo así una mayor precisión en los cálculos que el básico, sin

embargo, obviamos el detallado ya que no necesitamos alta precisión temporal por etapas del proyecto**.** Conocido esto es posible comenzar a desarrollar el problema matemático que se plantea.

Conductores de coste	Valoración
Atributos del producto	
Fiabilidad del software	Nominal (1)
Tamaño de la base de datos	Nominal (1)
Complejidad proyecto	Alta (1,15)
Atributos del equipo	
Restricciones de tiempo de ejecución	Nominal (1)
Restricciones de almacenamiento	Alta (1,06)
Volatilidad de la máquina virtual	Nominal (1)
Tiempo de respuesta del equipo	Bajo (0.87)
Atributos del personal	
Capacidad del analista	Nominal (1)
Experiencia en la aplicación	Nominal (1)
Capacidad de los programadores	Alta (0,86)
Experiencia en el sistema operativo utilizado	Nominal (1)
Experiencia en el lenguaje de programación	Alta (0,95)
Atributos del proyecto	
Prácticas de programación modernas	Alta (0,91)
Utilización de herramientas software	Alta (0,91)
Limitaciones de planificación	Muy alto (1,1)
Factor coste total	0.78926

Tabla 5: Estimación de valores asociados a cada atributo del modelo COCOMO.

Una vez conocemos las valoraciones para cada uno de los conductores de coste y el factor de coste total obtenido de multiplicar todas estas además del modelo COCOMO junto al tipo de desarrollo, es posible aplicar las fórmulas asociadas para realizar los cálculos. Estas fórmulas son:

$$Esfuerzo\ estimado = Factor\ coste\ total * a * \ nº\ lineas\ de\ código^{b}$$

$$Tiempo\ desarrollo = c * Esfuerzo\ estimado^{d}$$

En nuestro proyecto se disponen de 3936 líneas de código, este número se ha visto reducido gracias al apoyo de herramientas que realizan procesos como es el caso para la base de datos. En el proyecto estas líneas se dividen como 357 líneas en código C++, 467 para código SQL y el restante en la programación de la aplicación web (HTML, CSS, JS).

También se dispone de los coeficientes a, b, c, d COCOMO asociados al modelo intermedio y semilibre son 3,0 – 1,12 – 2,5 – 0,35 respectivamente, de esta forma las ecuaciones quedan del tipo:

$$Esfuerzo\ estimado = 0.78926 * 3 * \ 3{,}936^{1{,}12} = 10{,}985\ efectivos\ por\ mes$$

$$Tiempo\ desarrollo = 2{,}5 * 10{,}985^{0{,}35} = 5{,}78\ meses$$

$$Personal\ necesario = \frac{10{,}985}{5{,}78} = \ 1{,}9 \approx 2\ efectivos$$

Finalmente, tras la realización de los cálculos podemos determinar que este proyecto es idóneo para será desarrollado por un total de **dos personas** durante un periodo de **seis meses**. Al tratarse de un TFG estas condiciones no son posibles y se ven limitadas a **un único desarrollador** pero que trabaja durante **9 meses** por ende los resultados no distan en exceso del esfuerzo empleado en la realidad.

2.3.2 Presupuesto

Un proyecto conlleva una serie de costes a lo largo de su periodo de desarrollo. Por este motivo a lo largo de este apartado se detalla el coste económico que conllevaría el desarrollo del mismo. Al formar parte de un TFG no tiene un coste económico real, pero se realiza una estimación, se incluirán los distintos recursos utilizados en función de unas categorías que serán:

- **Hardware:** En este apartado incluimos el coste de los equipos informáticos necesarios para la realización del proyecto, así como otros dispositivos que

nos hayan ayudado en la realización del mismo obteniendo información u otros aspectos.

- **Software:** En ocasiones es posible disponer de software completamente gratuito y no limitante para conseguir los objetivos que se plantean, sin embargo, en la mayoría de los casos para poder disponer del 100% de los recursos es necesario emplear la mejor versión y por ende de pago. En este apartado nos pondremos en la peor situación teniendo en cuenta que estamos en un proyecto de trabajo y se necesita disponer de la mejor versión.

- **Personal y servicios:** Costes derivados de la contratación de personal necesario para la realización de actividades del proyecto.

- **Costes indirectos:** Gastos que no se encuentran directamente relacionados con el desarrollo del sistema, pero son necesarios para que un proyecto salga adelante, gastos de instalaciones, facturas…

2.3.2.1 Recursos Hardware

Durante el desarrollo del proyecto se ha empleado un equipo informático siendo este el principal gasto hardware empleado a lo largo del mismo. Este equipo fue adquirido en el año 2022 salvo por un incremento de RAM producido en el año 2023.

Componente hardware	Precio estimado (€)
Intel Core i5-11400F 2.6GHz Procesador	298,11
Gigabyte B560M DS3H V2 Placa base	100,99
Kingston PC4-25600 DDR4 16GB (2 unidades) Memoria RAM	159,99
Apacer AS2280P4 512 GB Disco duro SSD	49,89
Seagate Barracuda ST10000DM010 1TB Disco duro HDD	45,90
NVIDIA GeForce RTX 3060 12GB GDDR6 Tarjeta gráfica	349,90

Hiditec H13 Caja / torre	74,40
AOC 24B1H 24” Doble monitor de trabajo	233,60
Razer deathadder elite Ratón	72,99
Razer ornata chroma v1 Teclado	103,50
Total	1.489,27

Tabla 6: Coste económico del equipo informático de desarrollo del proyecto.

2.3.2.2 Recursos software

En este apartado se indicarán las herramientas utilizadas en cuanto a software se refiere para la realización del proyecto, muchas de estas aplicaciones u herramientas disponen de una versión gratuita o bien disponible para su uso en un periodo donde el desarrollador sea estudiante. Como se ha comentado previamente en otros casos para obtener un valor real para el coste del proyecto se utilizan las versiones comerciales equivalentes y no gratuitas, a su vez se obviarán aquellas herramientas que en cualquier caso son gratuitas.

El coste de las herramientas de pago mensual se estima en base al tiempo objetivo que se ha necesitado utilizar dicha herramienta, dando 1 mes de margen por posibles necesidades finales inesperadas.

Herramienta	Precio estimado (€)
Windows 11 Pro Sistema operativo	220
Microsoft 365 Personal Software de documentación	7€ / mes * 9 meses = 63
Figma Professional Herramienta de creación de imágenes	12€ / mes * 3 meses = 36
Adobe Photoshop CC Herramienta de manipulación de imágenes ráster	29,99€ / mes * 3 meses = 89,97

Visual Paradigm Herramienta de generación de diagramas	32,20€ / mes * 3 meses = 96,6
Total	505,57

Tabla 7: Coste económico del software del proyecto.

2.3.2.3 Personal y servicios

Durante el desarrollo de este proyecto al ser parte de un TFG el desarrollo completo del trabajo ha corrido a cargo de un único desarrollador es por este motivo que se considera para el presupuesto exclusivamente la necesidad de un contrato de trabajo que se asignará al autor de este trabajo. Para obtener los datos necesarios del puesto de trabajo y el presupuesto asociado al elegido revisamos el **XVII Convenio estatal de empresas de consultoría, y estudio de mercados y de la opinión pública**[13]**,** en este documento desarrollado en el año 2018 el rol que más se asemeja al trabajador necesario para la realización de este proyecto es el de **Analista programador: Diseñador páginas web,** este pertenece al **Área 3 grupo C nivel I**. Este rol es el elegido ya que para el mismo hace falta que el mismo lleve a cabo labores de diseño software de una aplicación web, así como parte de la planificación del mismo**,** por ende, los cálculos del coste del desarrollador del proyecto en el caso de tener que ser contratado por 9 meses serían los siguientes:

Descripción	Coste estimado (€)
Puesto de trabajo	24.640,37 * 9/12 = 16.426,91
Paga extra	3.285,38
Coste inicial	19.712,29
Seguridad social 33%	6.505,05
Coste final proyecto	26.217,35

Tabla 8: Costes finales.

2.3.2.4 Gastos indirectos

Como bien se ha comentado previamente estos gastos vienen derivados y asociados al desarrollo del trabajo, pero sin intervenir en el mismo, este tipo de gastos suelen ser del tipo tarifas de luz, agua, conexión a internet o lugar de trabajo (si acarrea un gasto trabajar en este). Como este tipo de costes es muy difícil de determinar a priori por la duración del mismo y la dificultad para determinar qué porcentaje de consumo real se consume en los

[13] Boletín oficial 2018. Ministerio de empleo y seguridad social

periodos de trabajo se determinarán estos gastos como un **sobrecoste de un 10%** del presupuesto del proyecto final para cubrir los gatos de este tipo que puedan surgir.

2.3.2.5 Coste final

Finalmente, y tras agrupar todos los gastos comentados previamente se resumiría el total del proyecto en la siguiente cantidad:

Origen	Coste estimado (€)
Recursos Hardware	1.489,27
Recursos Software	505,57
Personal y servicios	26.217,35
Total base	28.212,20
Gastos indirectos (10%)	2.821,22
Total Proyecto	31.033,42

Tabla 9: Coste total del proyecto.

2.3.2.6 Amortización

Para concluir el presupuesto se debe incluir la amortización de los bienes durante el periodo de desarrollo del proyecto. Este es el proceso por el cual se puede distribuir el costo de un activo a lo largo de su vida útil pudiendo deducir el costo del mismo a lo largo de varios años ya que no es posible deducir todo en el momento de compra.

Los valores de amortización se obtienen directamente de la tabla de coeficientes de amortización lineal de la Agencia Tributaria[14], lugar donde se establecen los datos con los que calcular la amortización final. Estos valores se corresponden con los siguientes datos:

- Los sistemas y programas informáticos o software disponen de una vida útil de 6 años y un coeficiente lineal máximo del 33%.
- Los equipos para procesos de información disponen de una vida útil de 8 años y un coeficiente lineal máximo del 25%.

Con estos datos es posible calcular la base amortizable como:

$$Base\ Amortizable = Valor\ inicial - Valor\ residual$$

El valor residual es posible calcularlo como:

[14] Coeficientes de amortización Agencia Tributaria española

$$Valor\ residual = \frac{Valor\ inicial}{Vida\ útil}$$

Finalmente, de esto se extra lo correspondiente a la amortización anual a partir de la cuál en función de la duración del proyecto se deducirá la amortización de los componentes del proyecto, para esto las fórmulas son:

$$Amortización\ anual = Base\ amortizable * Coeficiente\ de\ amortización$$

$$Amortización\ proyecto = Amortización\ anual * \frac{Duración\ proyecto\ (meses)}{12\ (meses)}$$

Conocidas estas fórmulas los cálculos de amortización asociados al proyecto serían los siguientes:

Origen	Vida útil (años)	Valor inicial (€)	Valor residual (€)	Amortización anual (€)	Amortización del proyecto (€)
Recursos Hardware	8	1.489,27	186,16	325,78	244,33
Recursos Software	6	505,57	84,26	139,03	104,27
				Total	348,60

Tabla 10: Valores de amortización de los recursos del proyecto.

Finalmente, tras estos cálculos podemos determinar que la amortización de los productos es de **348,60 €** de manera total durante el desarrollo del proyecto que corresponde a un periodo de 9 meses.

Capítulo 3: ANÁLISIS DEL SISTEMA

En este capítulo se especificarán aspectos importantes y necesarios en las fases iniciales del desarrollo del proyecto, para esto se definirán los diseños y análisis del sistema realizados en el inicio del trabajo con el fin de establecer unas bases, así como las especificaciones que debe cumplir el sistema desarrollado con los requisitos previamente establecidos y el análisis de posibles riesgos que pueden surgir durante la elaboración del trabajo.

3.1 Requisitos

En esta sección se especifican de manera general los requisitos principales para el éxito del trabajo y cumplimiento de los objetivos, en secciones futuras se detallarán de nuevo con un alto nivel de abstracción. Para poder discernir de manera correcta y según la norma UNE 157801 se empleará la palabra "*debe*" cuando un requisito sea de obligado cumplimiento y en otro caso se empleará el término "*debería*" para requisitos que no son indispensable para el proyecto, pero sí de interés.

Los requisitos pueden verse divididos según tres tipos conformando cada uno un subgrupo de requisitos para el sistema.

3.1.1 Requisitos de facilidad de uso

Este tipo de requisitos conforman el grupo de estos que nos permiten garantizar que un usuario sea capaz de utilizar el producto para realizar las tareas que desee de manera asequible y como indica el nombre con facilidad. Los requisitos de facilidad de uso identificados en este proyecto son:

- Se debe disponer de una interfaz clara y sin muchos elementos de distracción permitiendo al usuario centrarse en las actividades a realizar, por ende, esta debe respetar los estándares de usabilidad con el fin de poder llegar a todos los usuarios de la red.

- Se debe mantener un contacto constante con el usuario mediante la realización de acciones demandas o el muestreo de mensajes para indicar la situación de la aplicación en cada momento y que el usuario no se desubique dentro de la misma ni pierda la paciencia esperando según que procesos.

- Se debe disponer de un modelado tridimensional usable y que permita una interacción sencilla y pausada para poder comprender de manera correcta las acciones que están ocurriendo en la misma.

- Se debería disponer de una adaptación afín al uso de la aplicación en todo tipo de dispositivos con el fin de hacer esta accesible a los usuarios desde cualquier lugar y momento, esta adaptación es especialmente importante en la sección de modelado y escena tridimensional.

3.1.2 Requisitos funcionales

Este tipo de requisitos son aquellos que definen las funcionalidades internas de un sistema informático (en este caso una aplicación web), estas funcionalidades son acciones que un usuario podrá emplear cuando interaccione con el sistema. Los requisitos funcionales dentro de nuestro sistema son:

- La aplicación debe permitir navegar por un mapa que dispone de una serie de capas vectoriales y ráster, pudiendo seleccionar la zona con la que se trabajara más adelante.

- La aplicación debe disponer de un sistema de carga de modelos tridimensionales en formato de nube de puntos, así como texturas bidimensionales a las que aplicar un esquema de altitud para dotarlas de realismo.

- El sistema debe permitir aplicar algoritmos geométricos para un modelado concreto sobre el que se está trabajando con el fin de obtener la información que se esté buscando.

- La aplicación debe permitir gestionar la escena de manera completa pudiendo determinar de los recursos disponibles cuales visualizar, que perspectiva de observador utilizar o que acciones realizar.

- La aplicación debería permitir la descarga de la escena en un formato que pueda ser procesado en otras instancias por el sistema tridimensional con el fin de agilizar la carga y distribución de ciertos modelos.

3.1.3 Requisitos no funcionales

Los requisitos no funcionales son aquellos que no son utilizados para describir información ni funciones a realizar sino características del correcto funcionamiento del sistema y los requisitos funcionales, son también conocidos como atributos de calidad. Los requisitos no funcionales de nuestro sistema son:

- El sistema deberá reducir su complejidad con el objetivo de dejar la máxima memoria posible para el correcto funcionamiento de la escena con la carga y procesamiento de nubes de puntos, así como algoritmos geométricos que se realicen con esta.

- La aplicación debe ser capaz de procesar, guardar y almacenar los modelos tridimensionales con el formato más optimizado disponible y compatible para agilizar la carga inicial del modelado.

- El sistema debe ser capaz de actuar ante fallos inesperados o de usabilidad ofreciendo respuestas al usuario o continuando su trabajo tratando el error que se ocasione.

3.2 Especificaciones del sistema

Las especificaciones del sistema son un conjunto de características y requisitos técnicos que describen las capacidades y limitaciones de un sistema. En este apartado el objetivo será determinar estas especificaciones como historia de usuario, para esto nos basaremos en los requisitos definidos en la sección previa, además de recordar estos requisitos se explica por qué son necesarios o de utilidad y si se encuentran más enfocados hacia una ayuda al usuario o la persona al cargo del trabajo.

Especificación	Objetivo	Afectado
La aplicación debe disponer de un sistema de carga de modelos tridimensionales	Poder representar cualquier escenario sin dificultad	Desarrollador
El sistema debe permitir aplicar algoritmos geométricos a la escena	Poder obtener información más específica de la parcela	Usuario

La aplicación debe permitir la descarga de la escena en un formato más ágil y flexible	Poder disponer de escenas procesadas previamente de una forma más ágil para ahorrar tiempo en procesos futuros	Desarrollador
La aplicación debe permitir navegar por un servidor de mapas pudiéndose seleccionar parcelas concretas	Permitir acceder a datos más concretos y el groso de las funcionalidades de la aplicación	Usuario
La aplicación debe permitir gestionar la escena de manera completa	Permitir tener un control completo de la aplicación pudiendo elegir lo que ver y manejar en cada momento	Usuario

Tabla 11: Historias de usuario iniciales.

Además de las especificaciones presentadas previamente también es necesario definir los que son los casos de uso de nuestro sistema. Los casos de uso son aquellos que describen la interacción existente entre un usuario y el prototipo desarrollado para cada una de las posibles situaciones que este se podrá encontrar en determinadas circunstancias.

Dentro de los casos de uso los actores son las personas que van a utilizar el sistema para desarrollar una actividad (usuario). Como se comentará en un futuro dentro de nuestro sistema habrá distintos arquetipos de usuario y tendrán distintas funcionalidades destinadas, por este motivo será necesario disponer de tantos tipos de actores como usuarios o roles definidos.

Los distintos casos de uso con los que se podrán encontrar nuestros usuarios son:

- **Inicio de sesión**: Proceso que permitirá a los usuarios poder navegar a través de la aplicación desarrollada, si este no se encuentra dado de alta su acceso será inviable

- **Añadir / Eliminar parcelas**: Adición de nuevos datos de modelado o temáticos sobre una parcela que no son de actualización automática. Eliminación de datos presentes en la aplicación por ser erróneos o estar desactualizados (posible sustitución por datos nuevos).

- **Añadir / Eliminar usuarios**: Acción de añadir usuarios autorizados al uso de la aplicación o eliminar usuarios que ya no disponen del permiso para su uso.

- **Ver mapas de parcelas**: Funcionalidad principal de la aplicación que permite un acceso al servidor de mapas y sus capas de información.

- **Ver información parcela**: Acceso a la información más específica y personal de cada parcela.

- **Ver histórico de datos**: Acceso a los datos gráficos e históricos temáticos de cada parcela.

- **Ver / Trabajar modelado** tridimensional de las parcelas: Acción de entrar a la escena tridimensional de una parcela y posible trabajo en la misma.

Todos estos casos de uso presentados se representarán gráficamente más adelante en la Sección 3.3.5.

3.3 Análisis y diseño del sistema

En este apartado se tiene como objetivo presentar todos los documentos pertenecientes al diseño de las principales bases de la aplicación, en estos recursos se incluyen diagramas y esquemas del prototipo que se plantea desarrollar. Como se comentó previamente esta aplicación surge de cero al no tener una base previa sobre la que construir por ende se especifica en primer lugar el diseño base con el diseño de interfaz, base de datos o arquitectura y se analiza su posible uso mediante casos de uso o análisis de usuarios.

3.3.1 Arquitectura de la aplicación

En esta sección se definirá brevemente la arquitectura que sigue la aplicación desarrollada en este proyecto. Se dispone de una arquitectura cliente-servidor que albergara la aplicación web en cuestión.

La aplicación una vez finalizada se ha incluido en un servidor hospedado en una máquina virtual y que nos ofrece acceso constante a la misma vía Internet. El esquema final que ha seguido el proyecto es el siguiente:

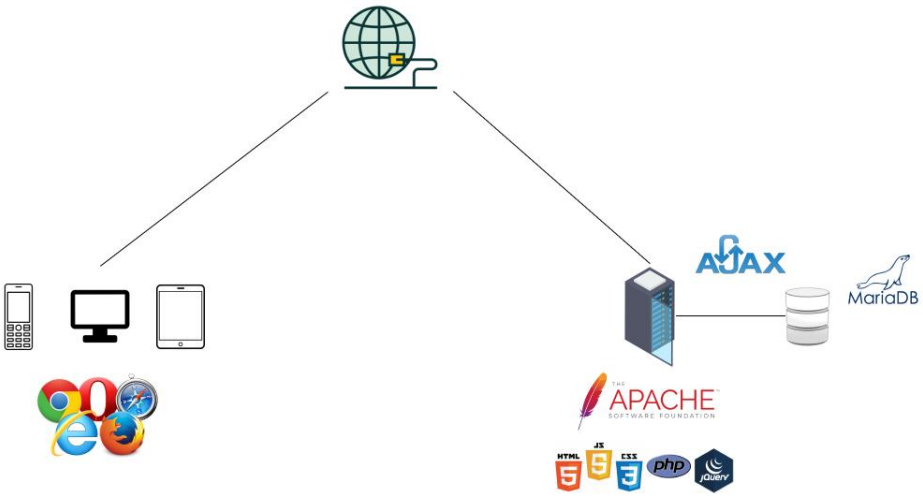


Ilustración 12: Arquitectura del sistema desarrollado (autor).

A continuación, se comentarán las funciones que tendrán tanto servidor como cliente:

- El **servidor** se encuentra instalado como se ha comentado previamente en una máquina virtual y tendrá a su disposición un gestor de bases de datos para almacenar toda la información necesaria para que el proyecto funcione correctamente. Además, tendrá instalado un servidor web que permita a los usuarios acceder a la aplicación hospedada a través de internet cuando y donde quieran.

- El **cliente** puede ser cualquier dispositivo con conexión a la red. En esta los usuarios podrán acceder a través de sus navegadores y podrán realizar distintas tareas que tengan planeadas. Cuando los usuarios realizan la conexión trabajan con el código hospedado en el servidor y los datos almacenados en la base de datos del mismo en tiempo real.

3.3.2 Perfiles de usuario

En este apartado se definirán los posibles usuarios que pueden llegar a utilizar nuestra aplicación, o al menos los casos para los que se ha planteado el trabajo. De forma general todos los tipos de usuario tendrán un acceso completo al prototipo generado en este proyecto, sin embargo, se realiza una distinción porque estos tendrán objetivos diferentes y por ende funcionalidades más afines o menos a su objetivo.

- El **administrador** será la persona o grupo de personas encargadas del mantenimiento de la aplicación. Serán los únicos usuarios con un acceso completo a toda la aplicación web, así como a su base de datos directamente.

Tendrán como tarea principal gestionar toda la información almacenada, así como introducir nuevos modelos, datos y/o usuarios.

- Los **propietarios** serán aquellos usuarios autorizados que demuestren que son los poseedores de alguna parcela sobre la que se trabaja o se quiere comenzar a trabajar con la aplicación web. Estos usuarios tendrán acceso a todas las páginas relacionadas con sus parcelas con datos o información más privada o extraída personalmente para el rendimiento de la misma, así como la posibilidad de editar ciertos datos de las mismas.

- Los **técnicos o agricultores** podrán pertenecer al campo previo de propietarios, pero también pueden ser trabajadores encargados de la parcela y su mantenimiento o desarrollo. Estos tendrán acceso a todos los datos de las parcelas que trabajen, pero no así la capacidad de editar la información libremente.

3.3.3 Diseño de la interfaz

En los inicios de un proyecto de aplicación web que van a usar multitud de usuarios es necesario predefinir las bases de lo que será la interfaz con la que estos se encontrarán y que esta cumpla con todos los requisitos y estándares de usabilidad. En esta sección se presentarán los sketches, wireframes y mock ups realizados al inicio del proyecto donde se presentaban las ideas planteadas para obtener la interfaz final.

En primer lugar, cabe destacar que desde la fase de previos del proyecto se planteó la estructura de la web de la cual se iba a querer disponer y se realizó el diseño de cada interfaz acorde a esto. A continuación, se mostrarán los diseños planteados para el proyecto, será en el primer apartado o sketch donde se explicarán las ventanas que componen cada una de las páginas de la aplicación web.

3.3.3.1 Storyboards / Sketch

También conocidos como sketch se corresponden con bocetos a papel y lápiz que se emplean para definir de manera abstracta las primeras ideas de diseño que se ocurren y como colocar los elementos en la interfaz. Suelen tener además pequeños comentarios sobre algunos aspectos de las mismas.

- **Inicio sesión (**Ilustración 13**):** Es la primera página que verá el usuario al entrar en la aplicación, esta se verá conformada por un formulario de los datos clave para identificarle.

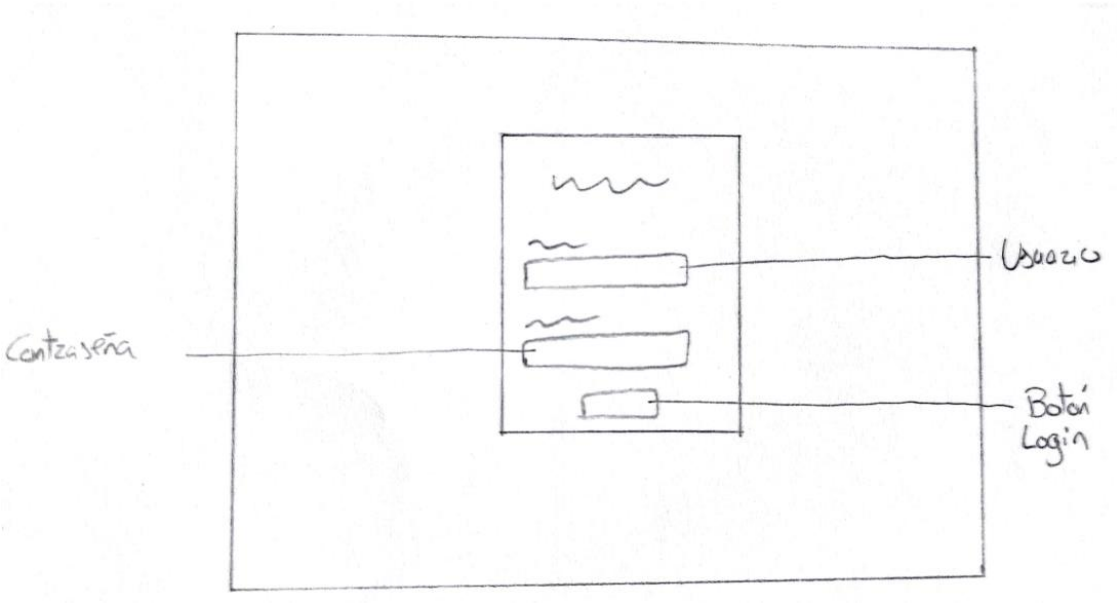


Ilustración 13: Sketch inicio sesión (autor).

- **Home** (Ilustración 14)**:** Página de inicio de la aplicación, una vez se han confirmado las credenciales de usuario será la primera página que se visitará, en esta se dispondrá de un mapa e información básica de la aplicación.

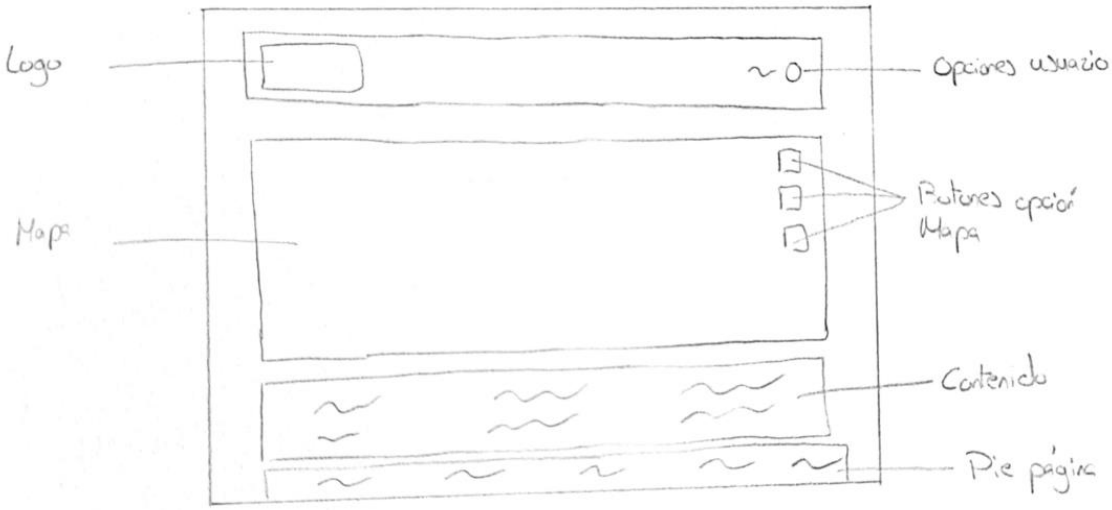


Ilustración 14: Sketch página bienvenida (autor).

- **Información parcela:** Página empleada para mostrar distintos niveles de información de una parcela concreta al usuario. Para esta página se han desarrollado dos ejemplos para los distintos tipos de datos mostrados, uno conformado por gráficas (Ilustración 15) y el otro por datos simples de texto e imágenes (Ilustración 16).

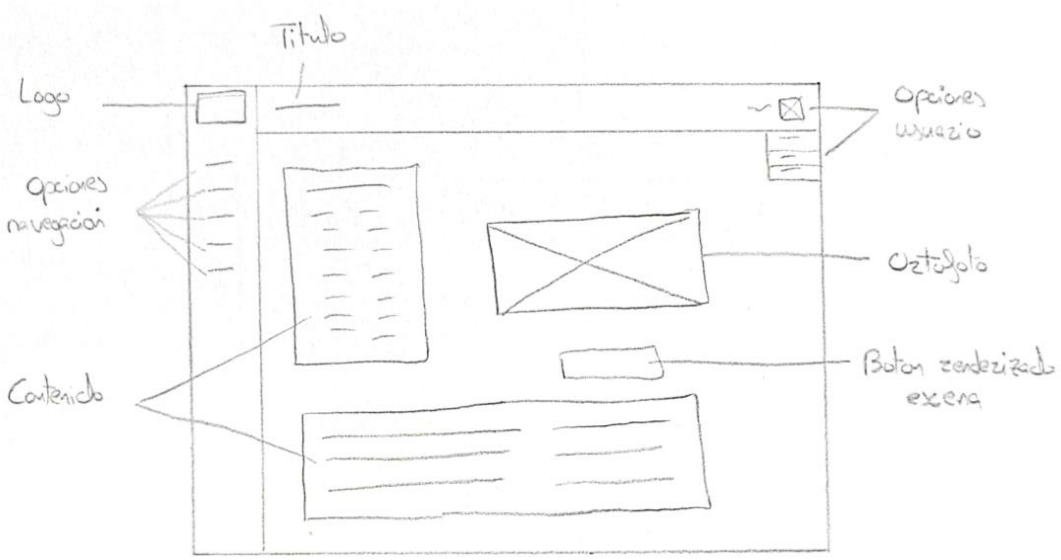


Ilustración 16: Sketch página principal parcela (autor).

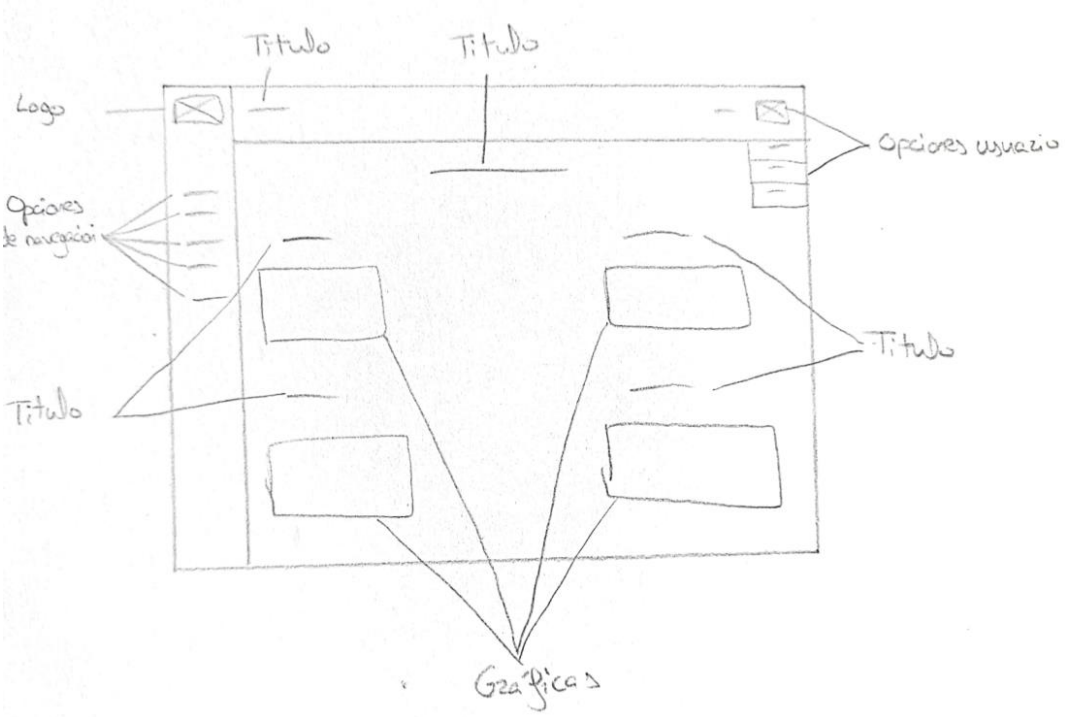


Ilustración 15: Sketch página gráficas parcela (autor).

- **Escena** (Ilustración 17)**:** Página dedicada a la muestra de una escena tridimensional de una parcela. Compuesta por los elementos de esta, así como un menú de opciones.

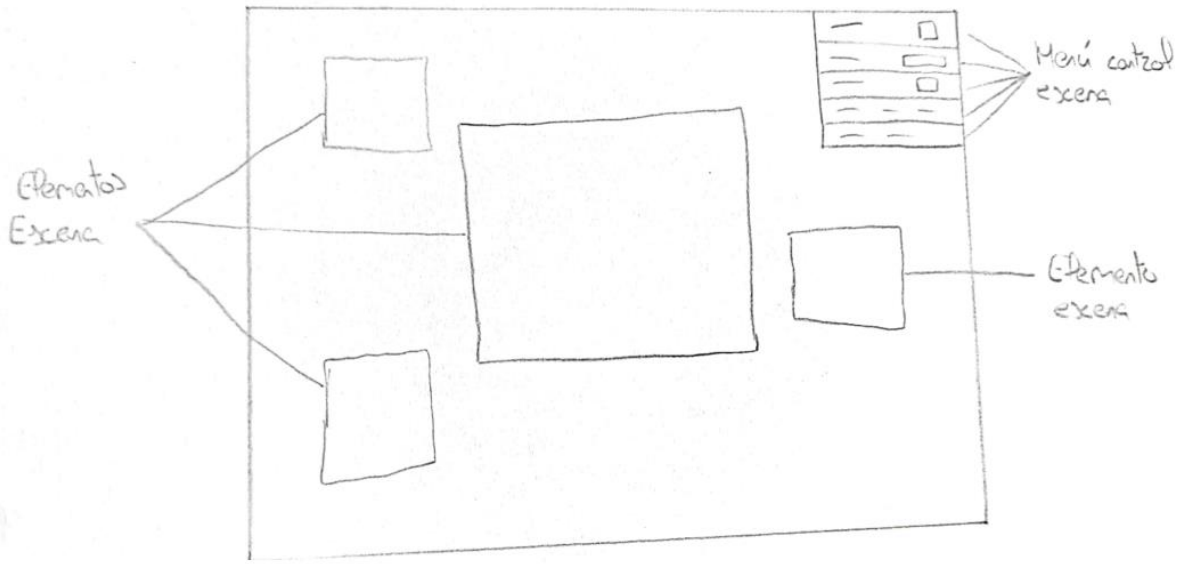


Ilustración 17: Sketch página escena tridimensional (autor).

- **Perfil:** Página dedicada a las funcionalidades disponibles para cada tipo de usuario conjunto a sus datos personales. Esta página se ha visto representada con cuatro ilustraciones, dos representan los tipos de secciones de la página como son listado de usuarios/parcelas (Ilustración 19) y datos directos del usuario (Ilustración 18), las dos restantes representan los formularios que nos podemos encontrar entre algunas opciones de la página de usuario y parcelas respectivamente (Ilustración 21) (Ilustración 20).

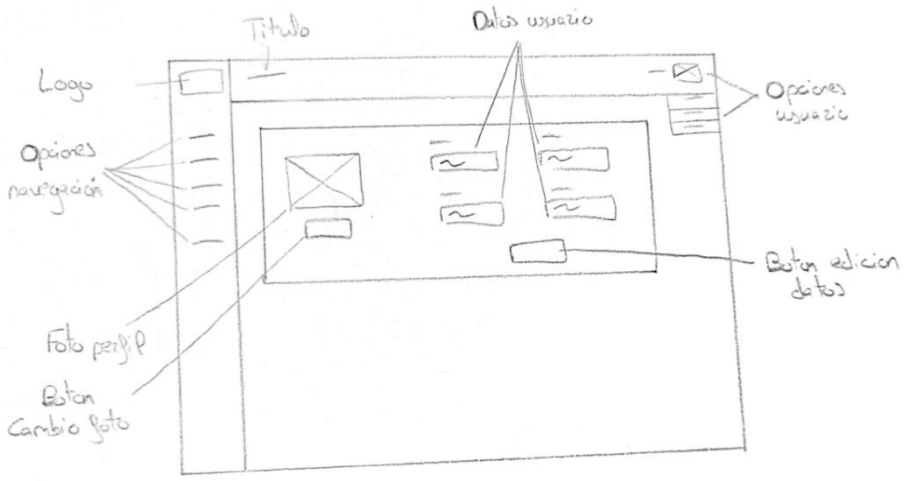


Ilustración 18: Sketch página principal perfil (autor).

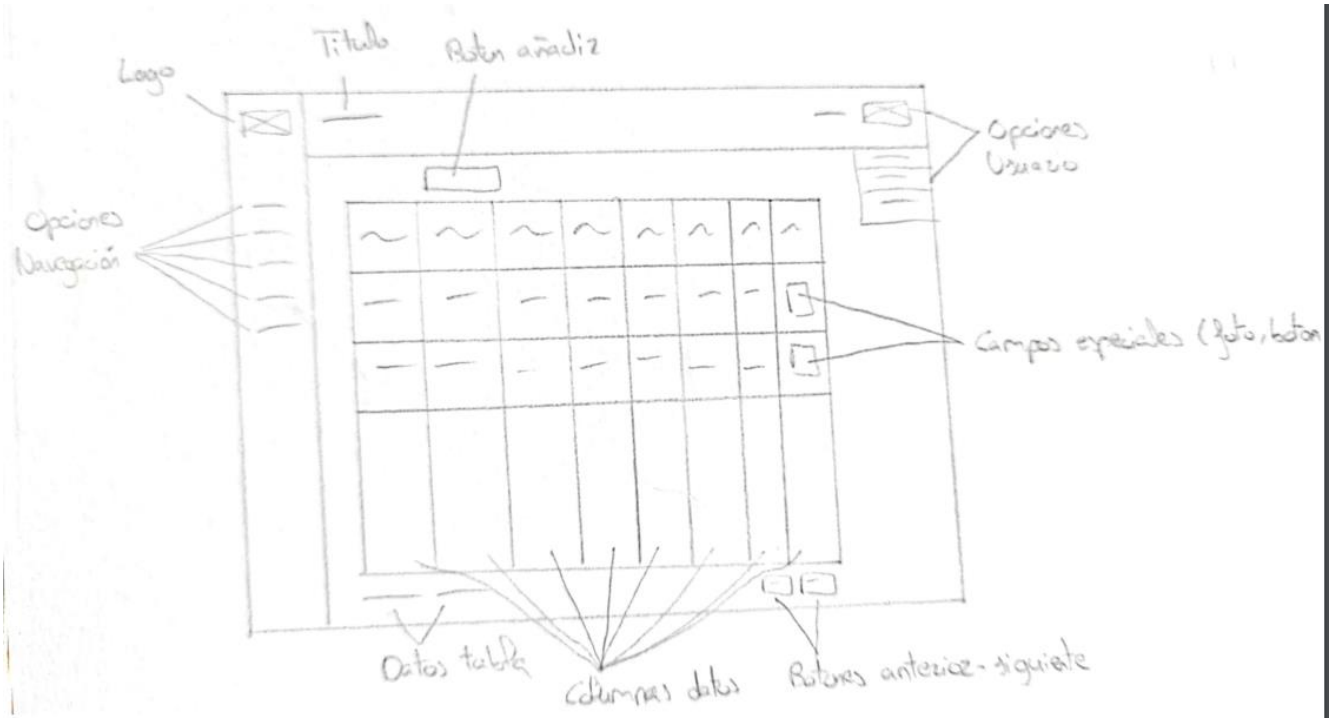


Ilustración 19: Sketch listado usuario/parcela (autor).

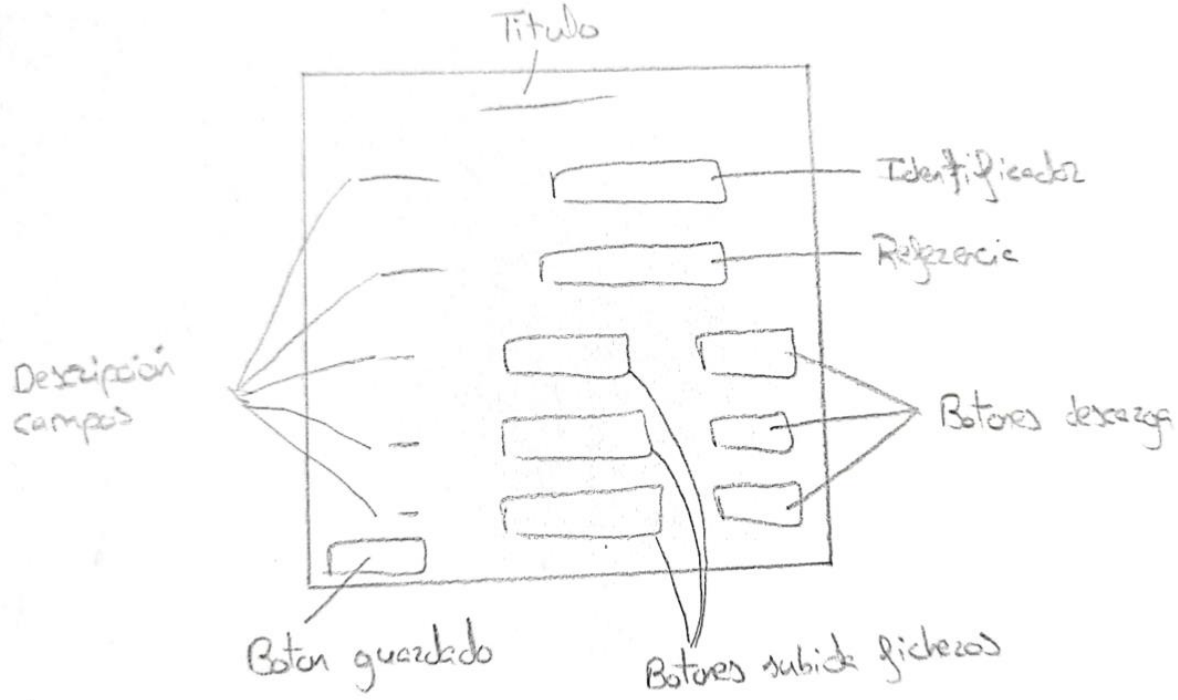


Ilustración 20: Sketch de subida / edición / descarga modelos parcela (autor).

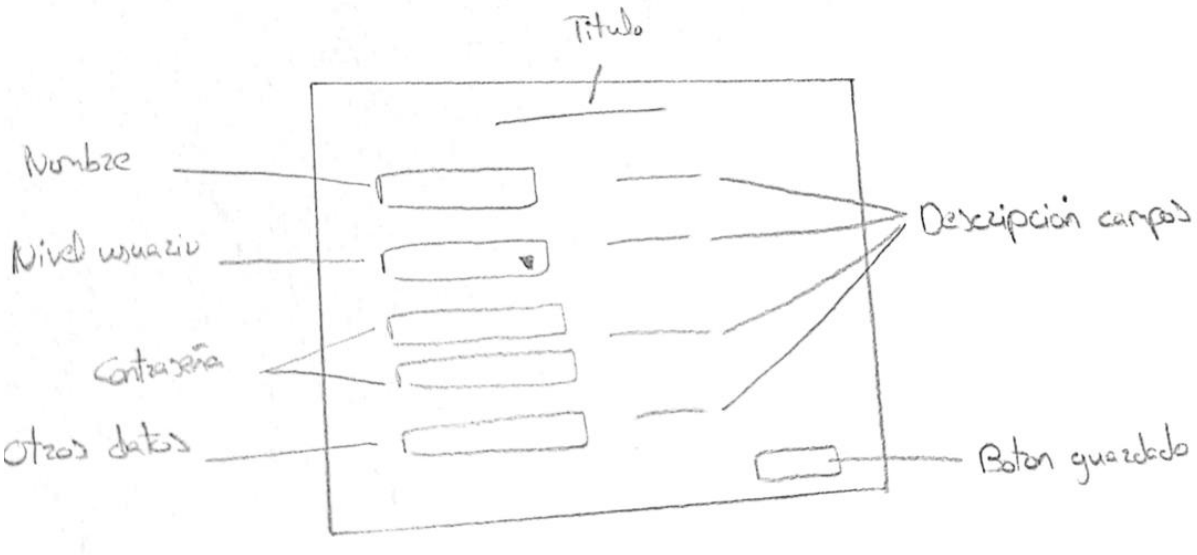


Ilustración 21: Sketch formulario inserción / edición usuarios (autor).

3.3.3.2 Wireframes

Representación de la interfaz de tipo alambre, carece de color, imágenes o estilo su único objetivo es representar la estructuración de los elementos, no se presentan textos finales solo genéricos.

- **Inicio sesión:** Wireframe de la pantalla de inicio de sesión desarrollado a partir del sketch de la Ilustración 22**.** Esta ventana se estructura con un formulario simple al inicio con dos campos para identificar al usuario

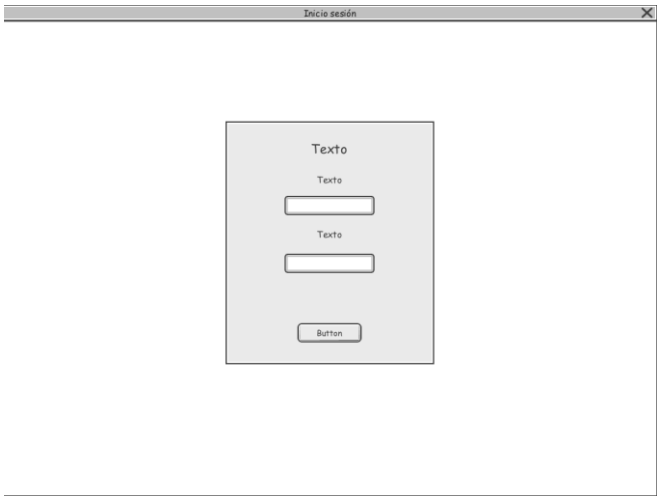


Ilustración 22: Wireframe inicio sesión (autor).

- **Home:** Wireframe de la pantalla de bienvenida desarrollado a partir del sketch de la Ilustración 23**.** Se encuentra formado por una barra superior, un mapa central y texto informativo inferior

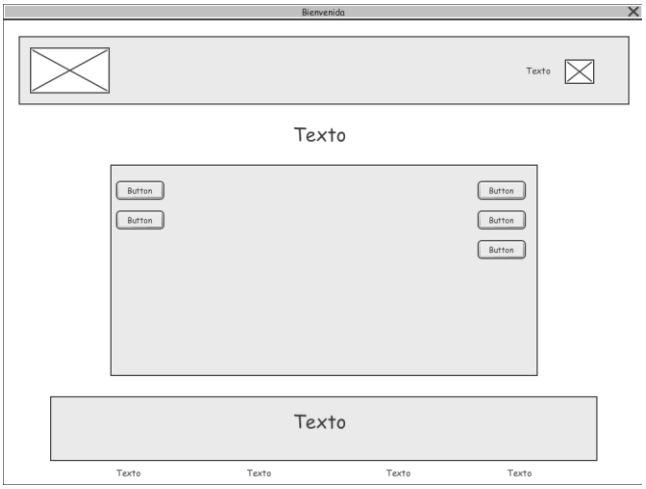


Ilustración 23: Wireframe página bienvenida (autor).

- **Información parcela:** Wireframe de la pantalla de información de cada parcela desarrollado a partir de los sketches previos. Esta página se encuentra formada por tres pantallas, dos pantallas de datos de la parcela (Ilustración 24) y una pantalla de datos gráficos y estadísticos (Ilustración 25).

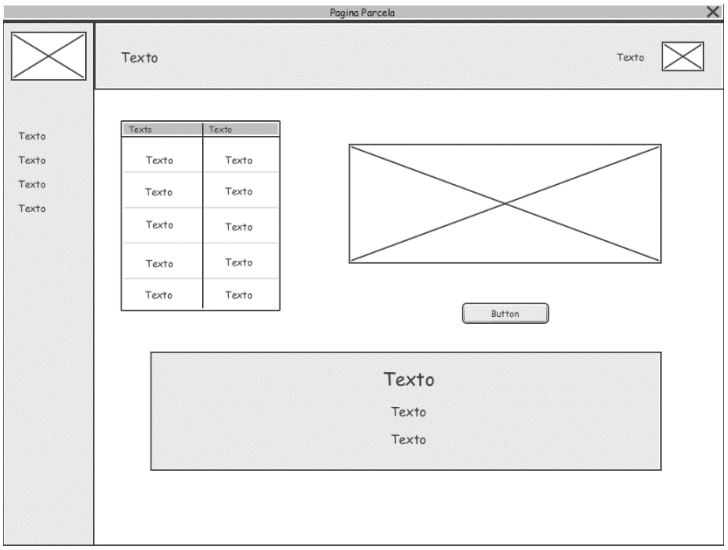


Ilustración 24: Wireframe página datos parcela (autor).

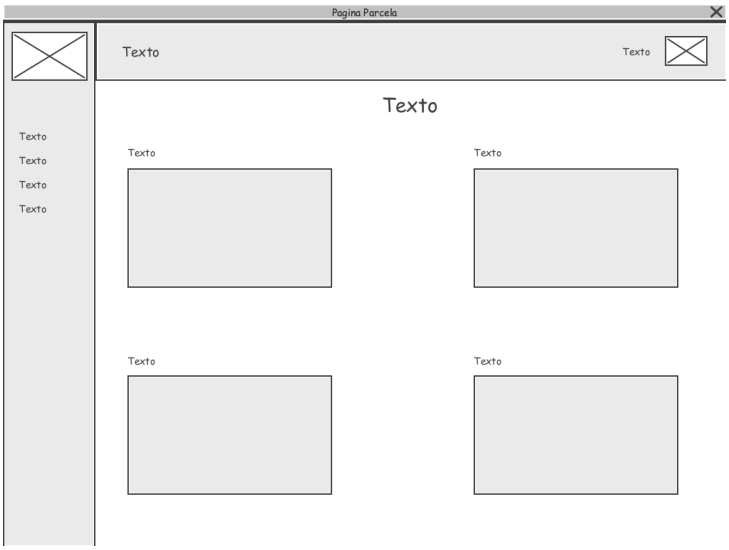


Ilustración 25: Wireframe página gráficas parcela (autor).

- **Escena:** Wireframe de la pantalla de representación de una escena desarrollado a partir del sketch de la Ilustración 26**.** Esta página se encuentra formada por los elementos que conformen la escena y un menú de control superior.

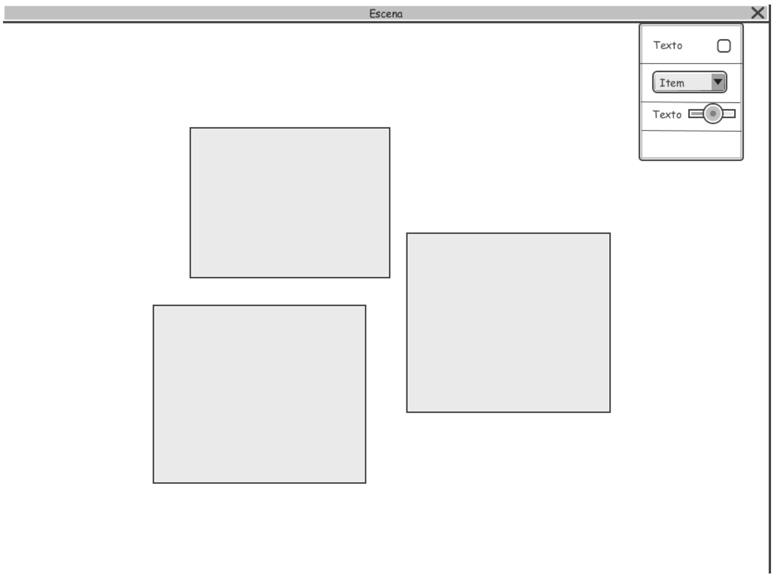


Ilustración 26: Wireframe página escena (autor).

- **Perfil:** Wireframe de la página de perfil desarrollado a partir de los sketches previos. Esta página se encuentra formada por tres pantallas y dos ventanas. Una primera pantalla de página principal del usuario (Ilustración 28) y dos pantallas con listado de usuario y parcelas que siguen la misma estructura (Ilustración 27), a cada una de estas dos pantallas finales se le asocia una ventana como formulario para adición o edición de datos (Ilustración 30 e Ilustración 29).

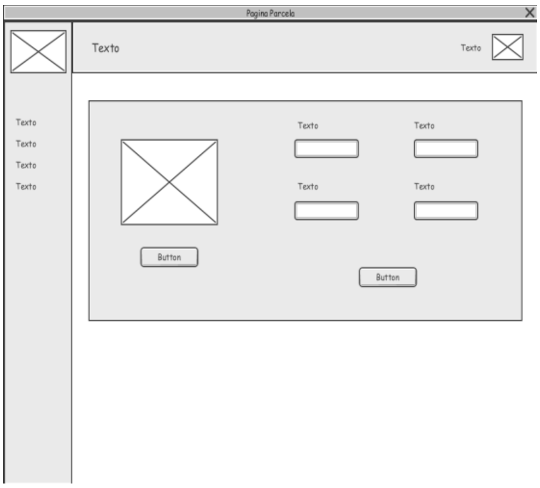


Ilustración 28: Wireframe página perfil (autor).

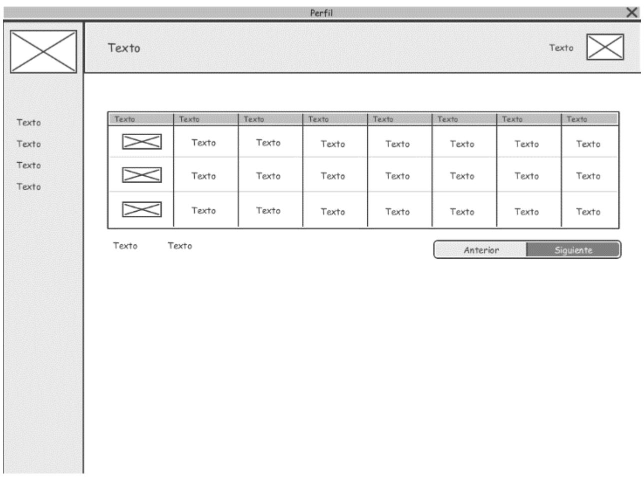


Ilustración 27: Wireframe listados usuarios/parcelas (autor).

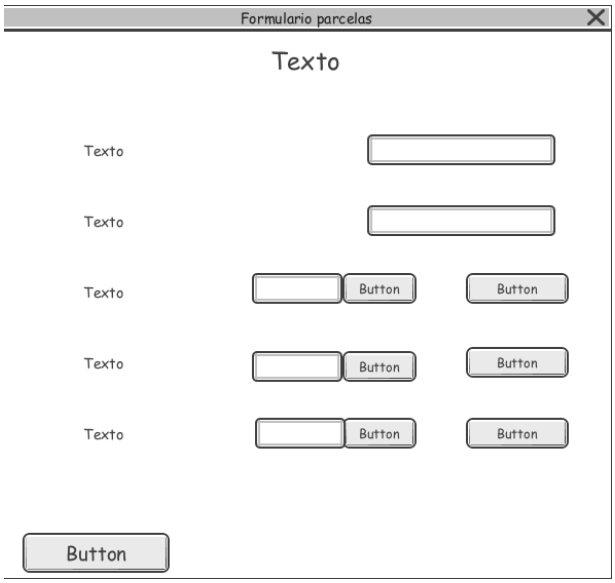


Ilustración 30: Wireframe formulario parcelas (autor).

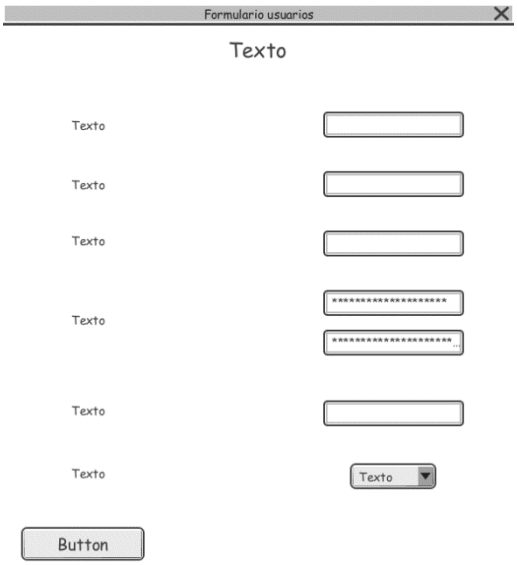


Ilustración 29: Wireframe formulario usuarios (autor).

3.3.3.3 Mock ups

Maquetas finales del proyecto que se enseñan al usuario, estas deben disponer de todos los detalles existentes y previstos para la aplicación, esta maqueta se realiza sin disponer de versión previa y por ende sin nada de código presente por este motivo la representación final puede tener pequeñas diferencias con la maquetada.

A continuación, se presentarán los mock ups realizados al inicio del proyecto y que por ende no disponen de todas las funcionalidades finales del proyecto que se fueron añadiendo conforme avanzaba el proceso, para observar el crecimiento de estos diseños se podrá observar en el Capítulo 5, donde en cada iteración para la cual la interfaz sufra cambios se mostrarán estos.

- **Inicio sesión:** Mock up de la pantalla de inicio de sesión desarrollado a partir del Wireframe de la Ilustración 31**.** Esta pantalla se encuentra conformada por un formulario de inicio de sesión con dos campos y una escena decorativa de fondo trabajada con el mismo programa que se trabajaran las escenas posteriores, sin embargo, esta se encuentra más optimizada al ser siempre la misma.

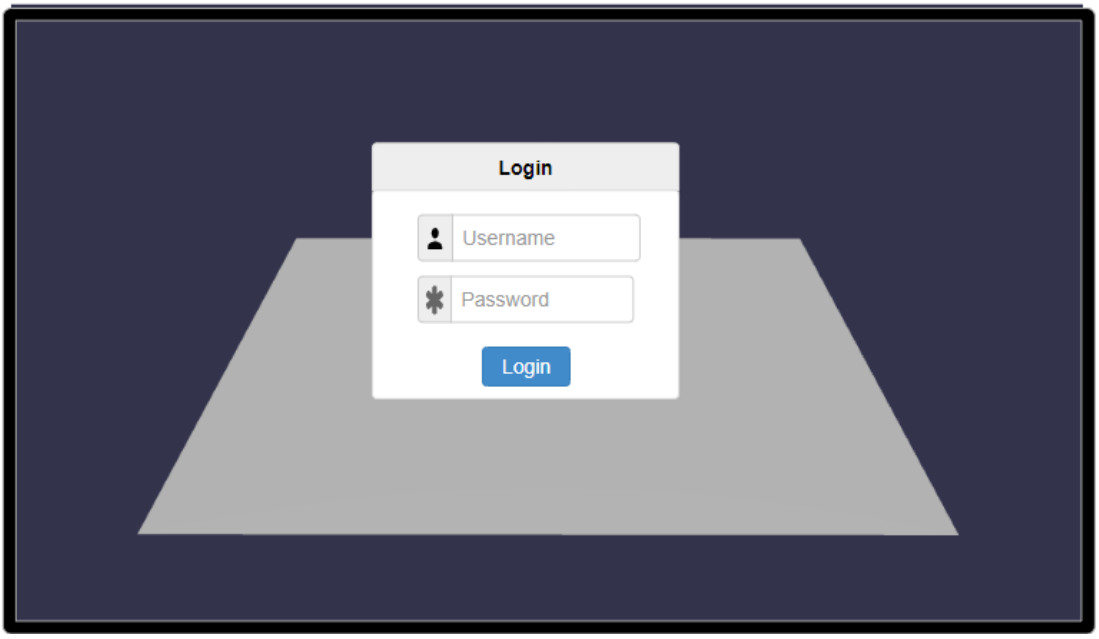


Ilustración 31: Mock up Inicio de sesión (autor).

- **Home:** Mock up de la pantalla de bienvenida desarrollado a partir del Wireframe de la Ilustración 32**.** Esta pantalla se encuentra conformada por un mapa interactivo bajo un servidor de mapas que nos permite delimitar parcelas, así como información relativa a la aplicación.

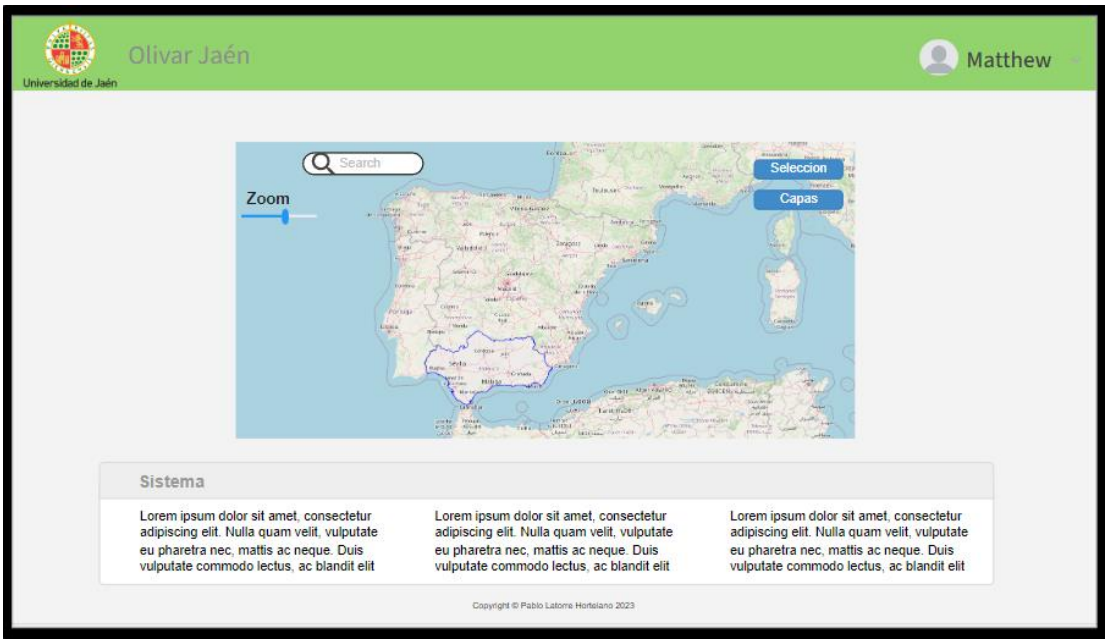


Ilustración 32: Mock up página bienvenida (autor).

- **Información parcela:** Mock up de la pantalla de información de cada parcela desarrollado a partir de los Wireframe del apartado anterior. Esta página se encuentra conformada por 3 pantallas que se dividen en los datos principales de la parcela (Ilustración 34), los datos zonales o sacados del SigPac (Ilustración 33) y un conjunto de históricos en formato gráfica (Ilustración 35).

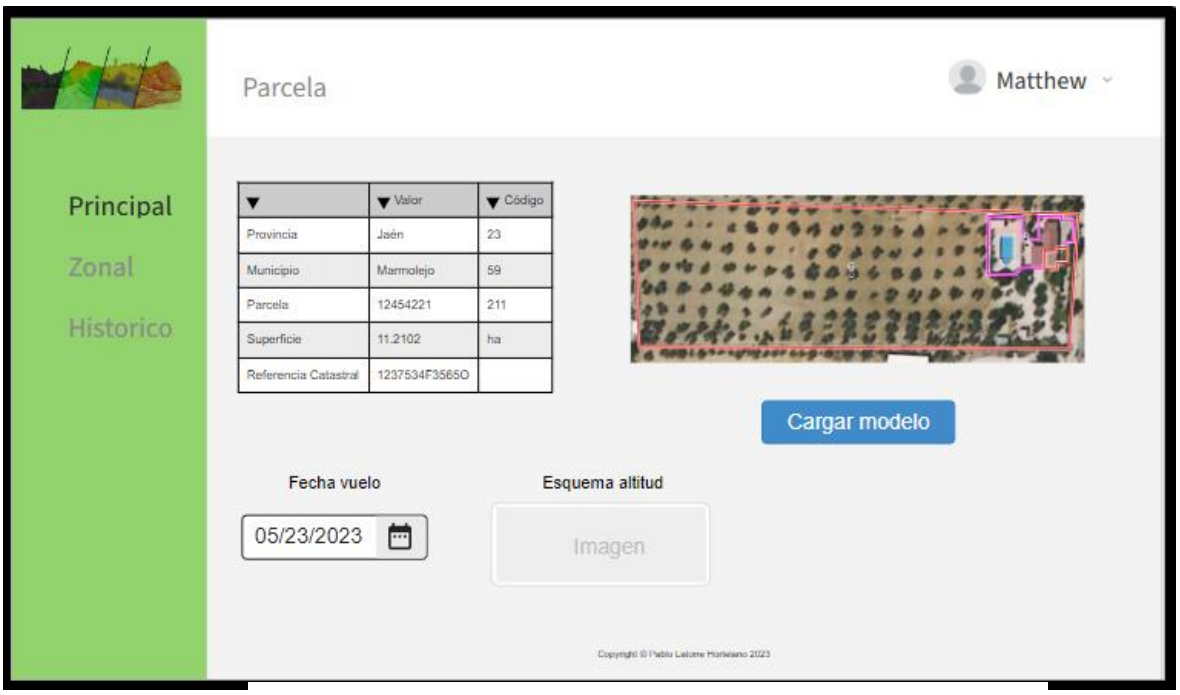


Ilustración 34: Mock up página parcela (autor).

Ilustración 33: Mock up página datos SigPac parcela (autor).

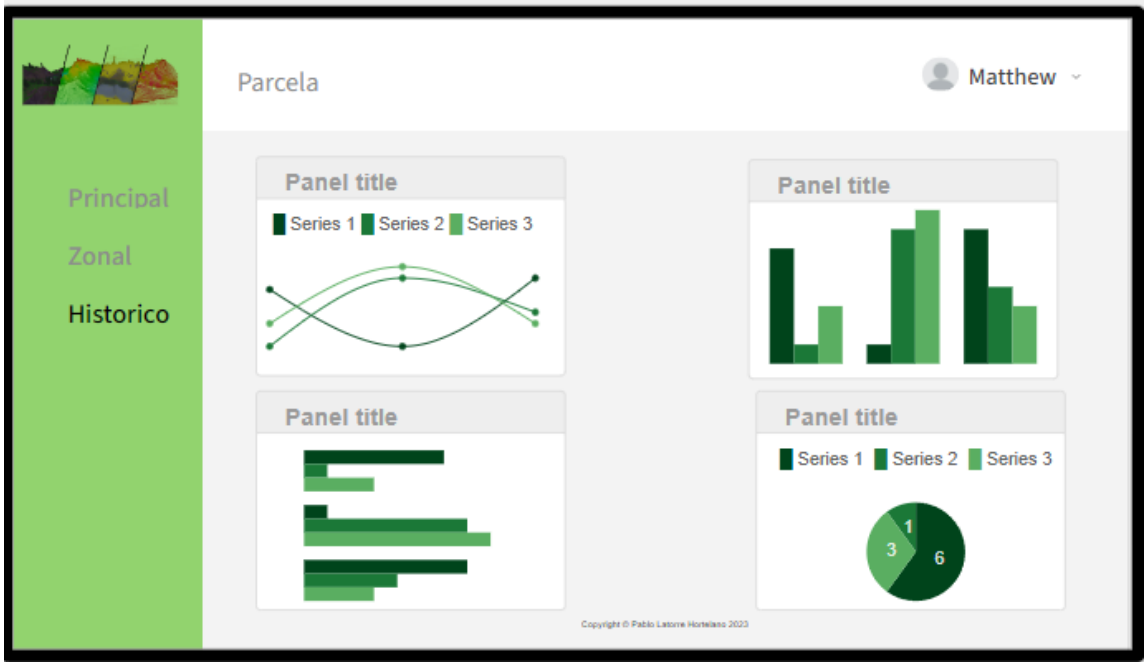


Ilustración 35: Mock up página datos históricos parcela (autor).

- **Escena:** Mock up de la pantalla de representación de una escena desarrollado a partir del Wireframe de la Ilustración 36**.** Esta página se encuentra conformada por los elementos de la escena, así como una interfaz de usuario con distintas opciones para controlar esta.

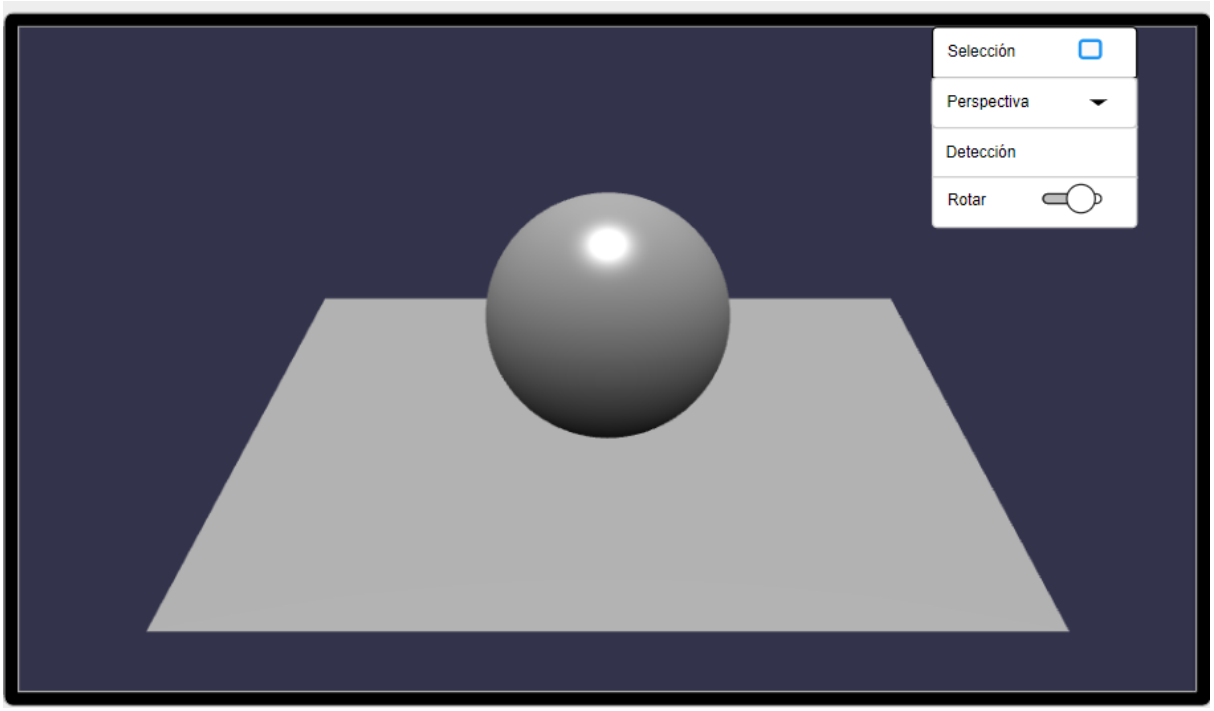


Ilustración 36: Mock up Escena tridimensional (autor).

- **Perfil:** Mock up de la página de perfil desarrollado a partir de los Wireframe del apartado anterior. Esta página se encuentra conformada por 3 pantallas y 2 ventanas, en primer lugar, tenemos la página inicial del usuario en la cual podrá alterar sus propios datos (Ilustración 37), disponemos de una ventana de listado de usuarios (Ilustración 38) que nos permite mediante un formulario añadir, editar o eliminar usuarios, por último, se dispone de una ventana de parcelas que listan las parcelas con modelos disponibles para la escena (Ilustración 40) así como un formulario que nos permita añadir o descargar los modelos (Ilustración 41).

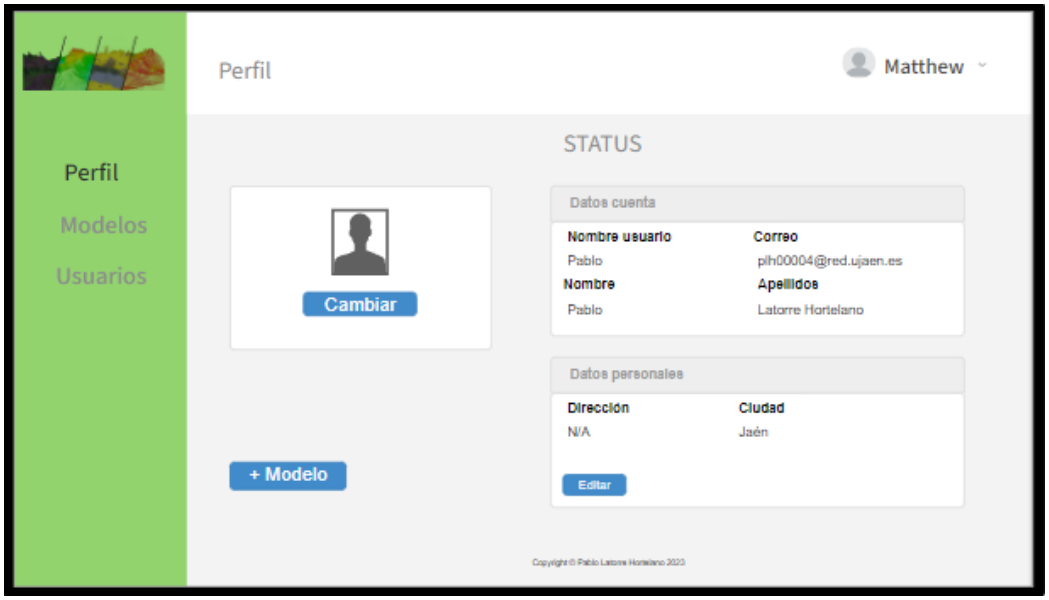


Ilustración 37: Mock up página inicial perfil (autor).

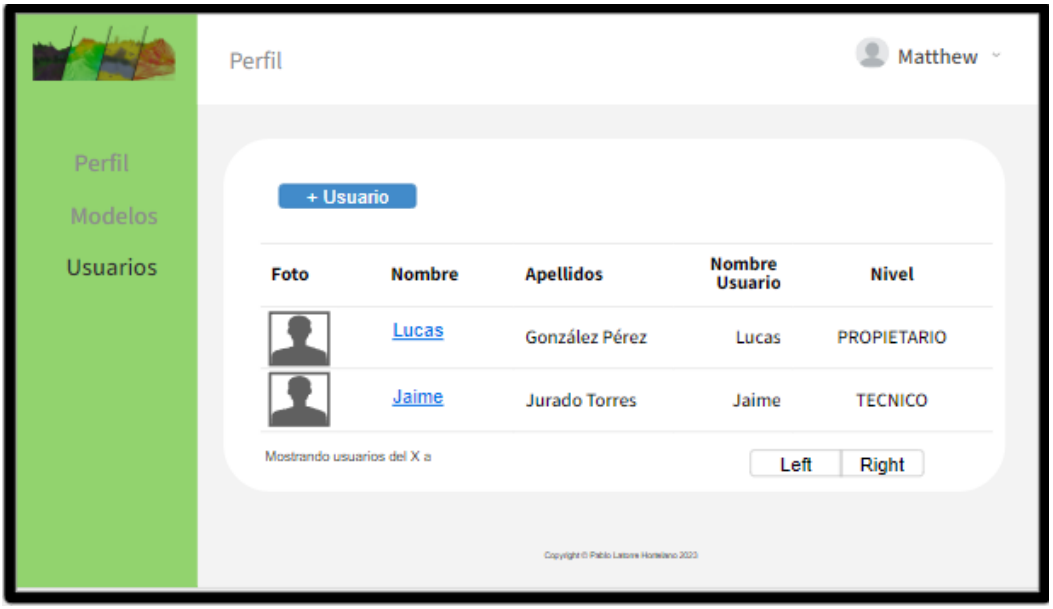


Ilustración 38: Mock up listado de usuarios (autor).

Ilustración 40: Mock up listado parcelas con Modelos (autor).

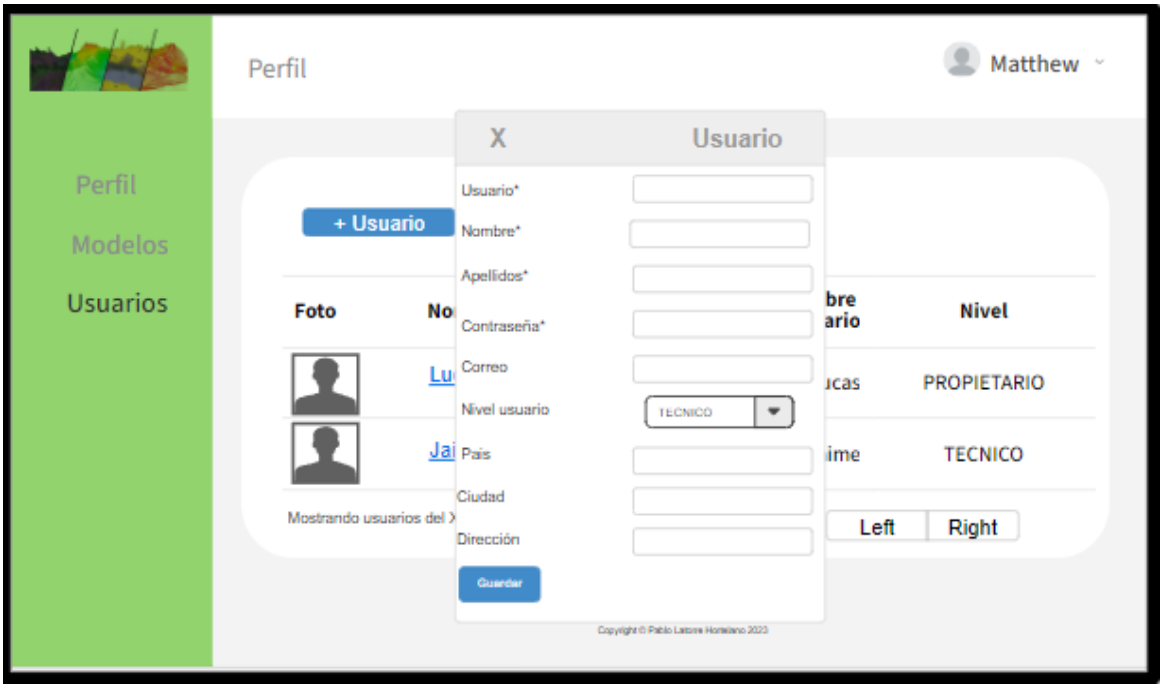


Ilustración 39: Mock up formulario usuarios (autor).

Ilustración 41: Mock up formulario parcelas (autor).

3.3.4 Diseño de la base de datos

En este apartado se presenta el esquema de base de datos del que dispone el sistema final de la aplicación web y que le permite enriquecer de información a la misma, así como mantener una correcta funcionalidad constante en todas las funcionalidades implementadas. El esquema Entidad-Relación resultante del proyecto es el siguiente:

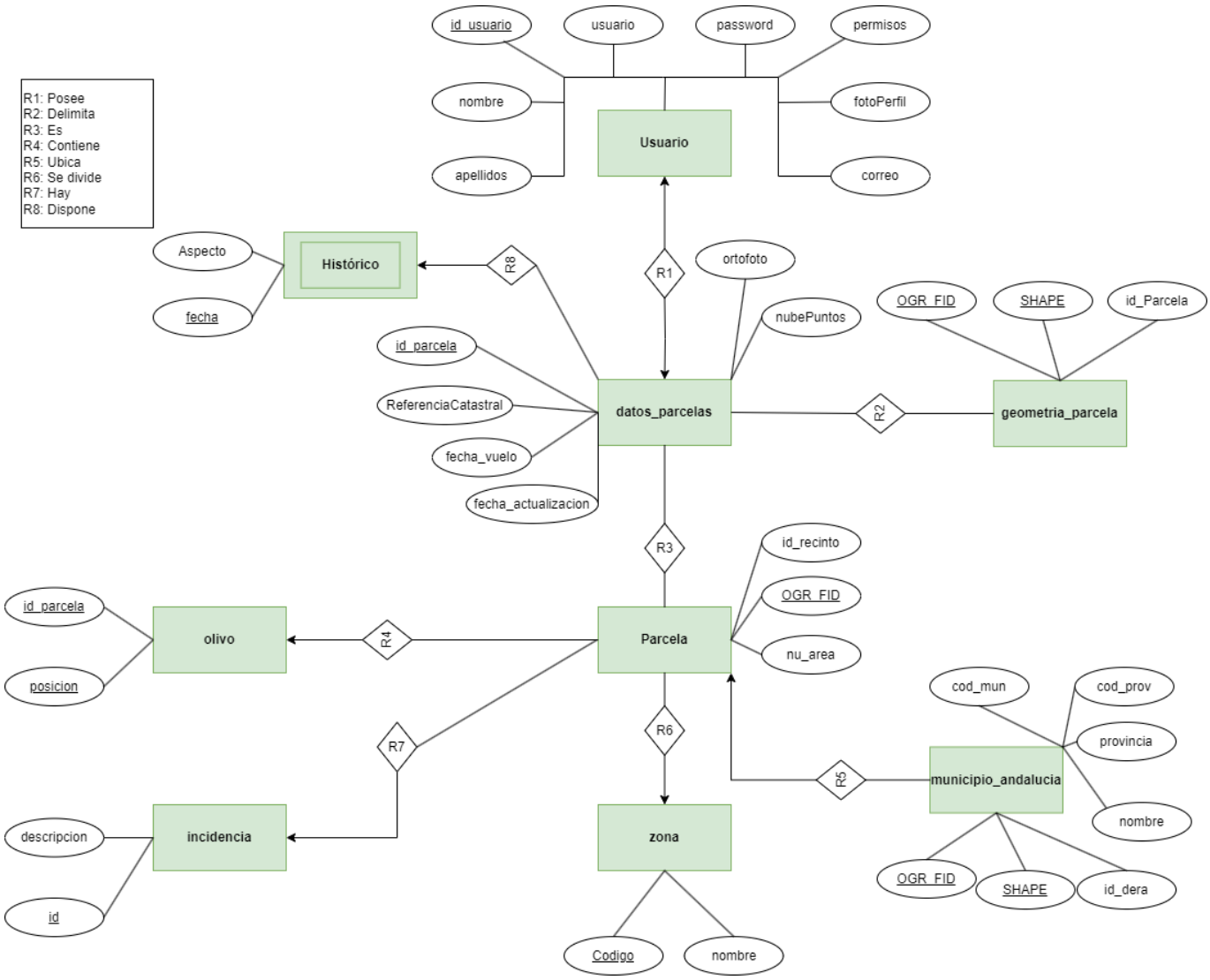


Ilustración 42: Diagrama Entidad – Relación (autor).

Dentro de este esquema podemos diferenciar las siguientes entidades y sus funciones que son:

- Las **parcelas** se almacenarán como entidades propias y se dividirán en tres tablas como son datos_parcela que tendrán los datos temáticos y privados de cada parcela, la tabla parcela que almacena los datos públicos extraídos del

SigPac y la tabla geometría_parcela que almacena la información espacial para la representación de la misma. Las tres tablas estarán relacionadas con una clave primaria común.

- Las **zonas** e **incidencias** son tablas expresamente destinadas al enriquecimiento de datos relativos a cada una de las parcelas para poder realizar análisis también análisis más concretos dentro de la misma.

- Los **olivos** son de base una tabla vacía y que se rellenara de manera automática mediante la detección de este tipo de entidades en las parcelas disponibles, el objetivo es obtener nueva información acerca de estos dentro del entorno trabajado.

- Los **usuarios** se almacenarán dentro de nuestra base de datos, así como sus credenciales codificadas y su estatus dentro de la aplicación.

- Los **municipios de Andalucía** se almacenarán en nuestra base de datos con el único propósito de su uso para la identificación de zonas dentro del servidor de mapas inicial.

- Las tablas de **Temperatura, Humedad y restantes** son las tablas de tipo temporal ya que almacenan datos históricos de estas propiedades sobre cada una de las parcelas en distintas fechas determinadas, su uso es para análisis de rendimiento y extracción de conclusiones. Estas tablas se corresponderán con la tabla de Histórico que se ha añadido solo una vez al diagrama para generalizar de la imagen donde el atributo aspecto adquirirá el nombre del dato histórico concreto. Esta será una tabla de entidad débil con respecto a los datos de la parcela.

- Las tablas **geometry_columns** y **spatial_ref_sys** han sido omitidas dentro del esquema E-R ya que no se relacionan con ninguna otra tabla y son tablas generadas exclusivamente para la correcta gestión de datos espaciales dentro de nuestro sistema de base de datos, estas almacenan los tipos de datos espaciales de cada tabla con datos espaciales y el sistema de referencia correspondiente respectivamente.

Más adelante en el desarrollo se explicará más detenidamente como se ha obtenido la información y rellenado la base de datos aquí expuesta.

3.3.5 Diagramas de casos de uso

Un diagrama de casos de uso es una representación gráfica de los casos de uso que se definen para un proyecto concreto. Los distintos elementos que podemos llegar a encontrar en estos diagramas son:

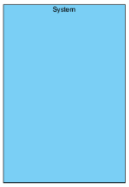
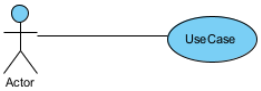
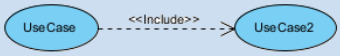
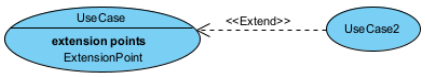
Nombre	Símbolo	Definición
Actor		Representación del usuario que va a utilizar el sistema. Pueden existir varios dentro del proyecto
Sistema		Sistema o aplicación con la que los actores interactúan
Acción / Caso uso		Representación de un caso de uso o actividad a realizar
Asociación		Representa la relación entre un actor y un caso de uso
Inclusión		Relación entre casos de uso que indica que un caso de uso implica el otro (es una subfunción)
Extensión		Relación entre casos de uso que indica que el principal tiene la opción de aplicar el segundo, pero no la obligación

Tabla 12: Elementos existentes en un diagrama de casos de uso.

A continuación, se procederá a mostrar las representaciones gráficas de los casos de uso que se definieron en secciones anteriores y que conforman nuestro proyecto.

En primer lugar, se presenta el diagrama de casos de uso de los usuarios que responden a ser propietarios de alguna parcela o trabajadores de la misma.

Ilustración 43: Diagrama de casos de uso de un propietario y sus técnicos (autor).

Por último, tenemos el diagrama que representa las acciones de un administrador en la página web (Ilustración 44). Se mostrarán únicamente las acciones exclusivas de los mismos, pues estos como se comentó previamente tienen acceso a todas las funcionalidades mostradas en el diagrama anterior (Ilustración 43).

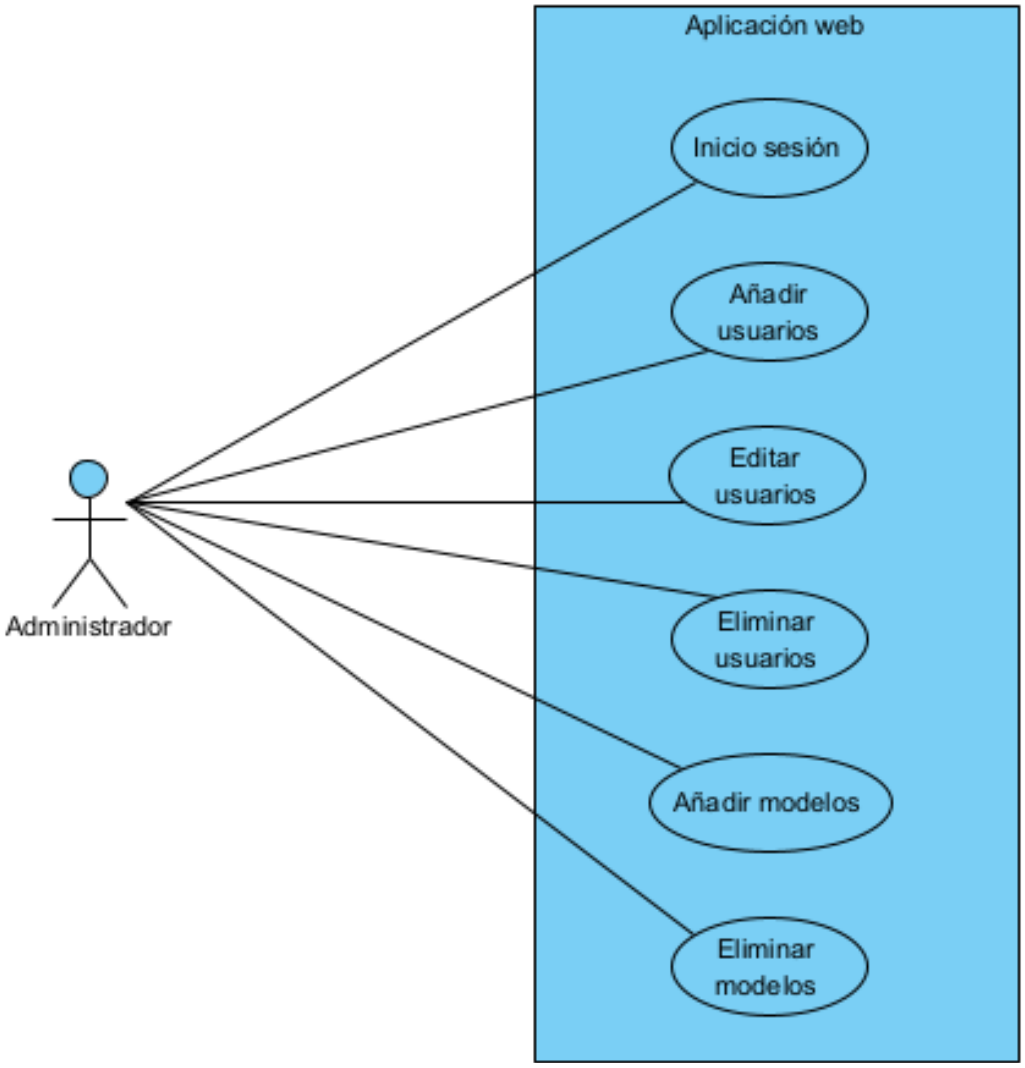


Ilustración 44: Diagrama de casos de uso de un administrador (autor).

3.3.6 Diagramas de secuencia

Una vez presentados los casos de uso que integrarán nuestro sistema se deben presentar de manera más detallada las interacciones de los usuarios con nuestro sistema, para poder conocer esto se emplearán los diagramas de secuencia. Estos diagramas son los encargados de describir cómo interactúan los objetos o entidades del sistema entre sí para completar de manera exitosa algunas tareas.

A continuación, se procederá a mostrar gráficamente la representación de estos diagramas con las distintas funcionalidades del sistema. Previamente se definirán los elementos empleados en estos diagramas ya que, aunque hay muchos solo ha sido necesario el uso de algunos de estos como son:

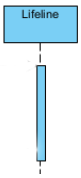
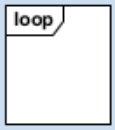
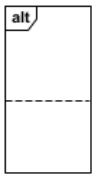
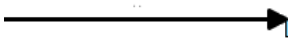
Nombre	Símbolo	Definición
Actor		Representación del usuario que va a utilizar el sistema. Pueden existir varios dentro del proyecto
Línea de vida		Representa a una entidad participante en la acción, la actividad o pasividad del objeto se expresa con un rectángulo sobre la línea
Clase control		Coordinan y controlan los objetos del sistema. Se usa una por caso de uso
Clase limite		Modelan la interacción entre sistema y actores
Operador LOOP		Se utiliza para indicar operaciones que se van a repetir un número determinado de veces y bajo unas condiciones.
Operador ALT		Se utiliza para distribuir el devenir de las acciones en función de una condición, si esta es verdadera se ejecuta la parte superior, sino la inferior.
Operador OPT		Se utiliza para indicar funcionalidades u operaciones optativas
Mensaje		Un emisor inicia una operación y lo notifica al receptor, el emisor quedara en espera de que este la termine.
Mensaje retorno		Indicación al emisor de que el receptor ha terminado la tarea.
Mensaje reflexivo		Mensajes enviados de un objeto hacia sí mismo.

Tabla 13: Elementos empleados en los diagramas de secuencia.

- **Inicio sesión (**Ilustración 45**):** Este caso de uso se encuentra disponible para todos los tipos de usuarios. Este se da en una pantalla de inicio de sesión con un formulario, cuando el usuario lo rellene se realiza una comprobación de si el usuario se encuentra registrado en la base de datos:

    - Si este se encuentra registrado se le redirigirá a la pantalla principal e iniciará una sesión con un tiempo límite de uso.

    - Si no se encuentra registrado se le mostrara un mensaje de error sobre los datos introducidos.

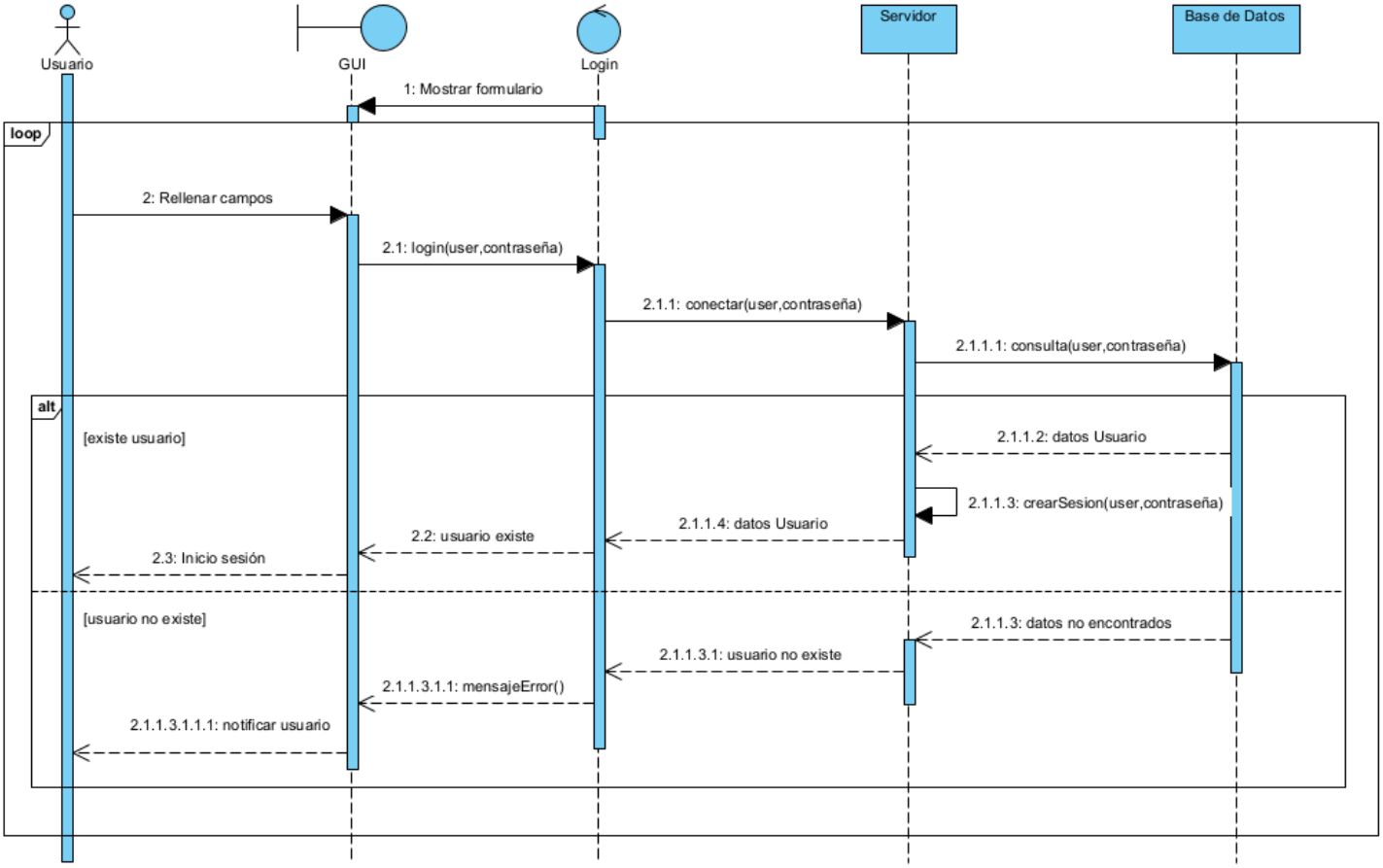


Ilustración 45: Diagrama secuencia Inicio de sesión (autor).

- **Ver mapas (**Ilustración 46**):** Esta funcionalidad está disponible para todos los usuarios. En primer lugar, se carga el mapa, una vez esto ocurra el usuario podrá determinar las capas de información que quiere observar. Cuando tenga claro la parcela a seleccionar la marcará y será redireccionado a una página individualizada de la misma.

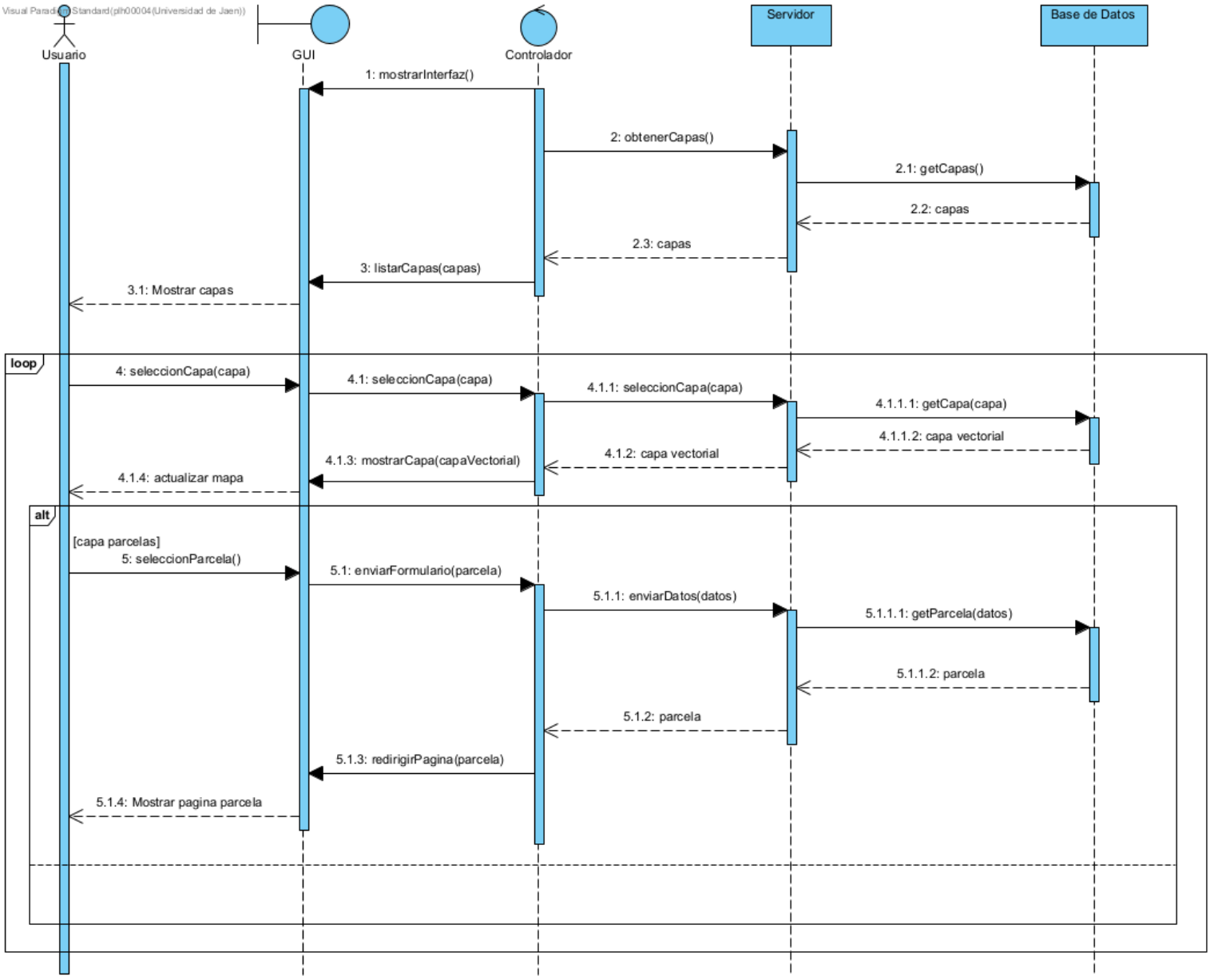


Ilustración 46: Diagrama de secuencia Ver mapa (autor).

- **Ver información parcela (**Ilustración 47**):** Funcionalidad común a todos los usuarios esta permite navegar por la página individual de cada parcela mostrando la información en función de las secciones solicitadas por el usuario.

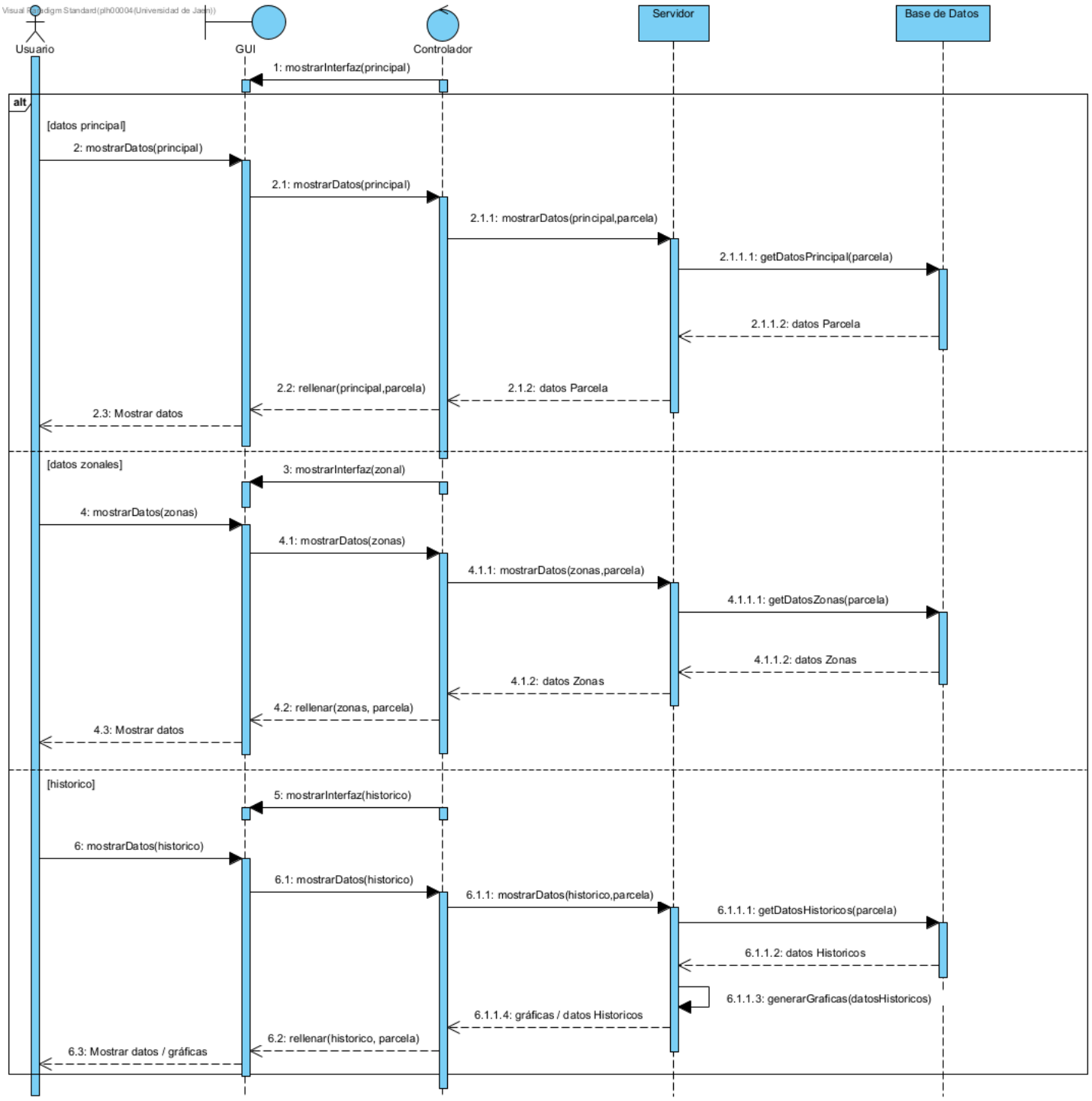


Ilustración 47: Diagrama de secuencia Mostrar datos (autor).

- **Editar información (**Ilustración 48**):** Funcionalidad disponible para administradores y propietarios de parcelas les permite editar información temática de la misma alterando sus valores en la base de datos.

  Para esto se le presentarán al usuario los datos en un formulario ya rellenado y que este podrá editar y enviar al servidor para su posterior actualización. Una vez validados se muestra la página de la parcela con la información actualizada

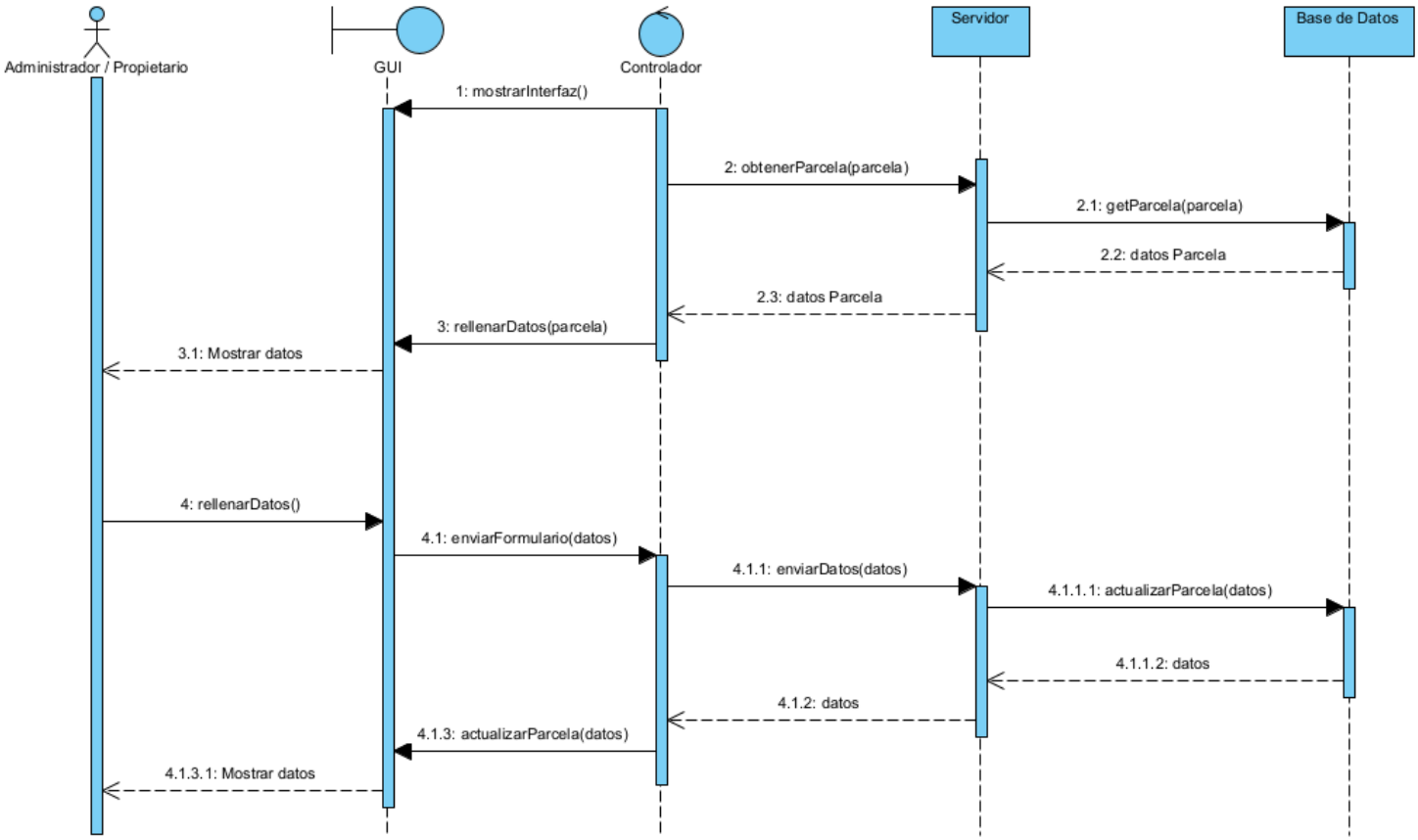


Ilustración 48: Diagrama de secuencia Editar información (autor).

- **Añadir / editar usuarios (**Ilustración 49**):** Funcionalidad asociada a los administradores. El administrador debe rellenar un formulario con los datos básicos nuevos del usuario y el servidor los actualizara en la base de datos. Si el usuario ya existía y se intenta añadir nuevo se muestra un mensaje de error.

  En el caso de la edición el proceso sera muy similar salvo que el formulario se le mostrara al administrador relleno pero con la posibilidad de realizar cambios en los datos.

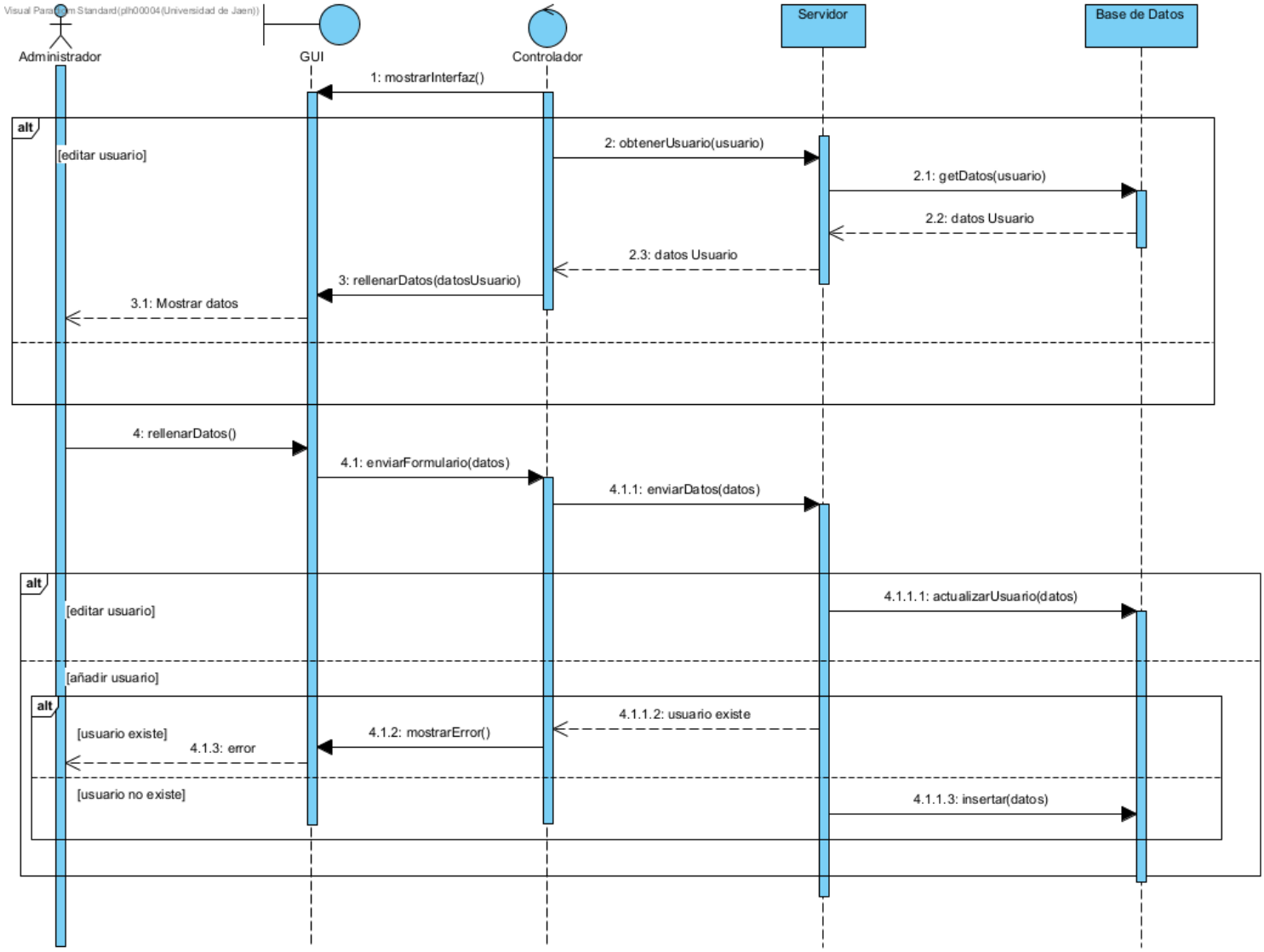


Ilustración 49: Diagrama de secuencia Añadir/Editar usuarios (autor).

- **Eliminar usuarios (**Ilustración 50**):** Funcionalidad asociada a los administradores. El administrador recibirá una lista con la totalidad de usuarios del sistema y podrá indicar aquel o aquellos que quiera eliminar y mandar la petición al servidor que actualiza la base de datos con su eliminación y muestra un mensaje de confirmación de la eliminación del mismo.

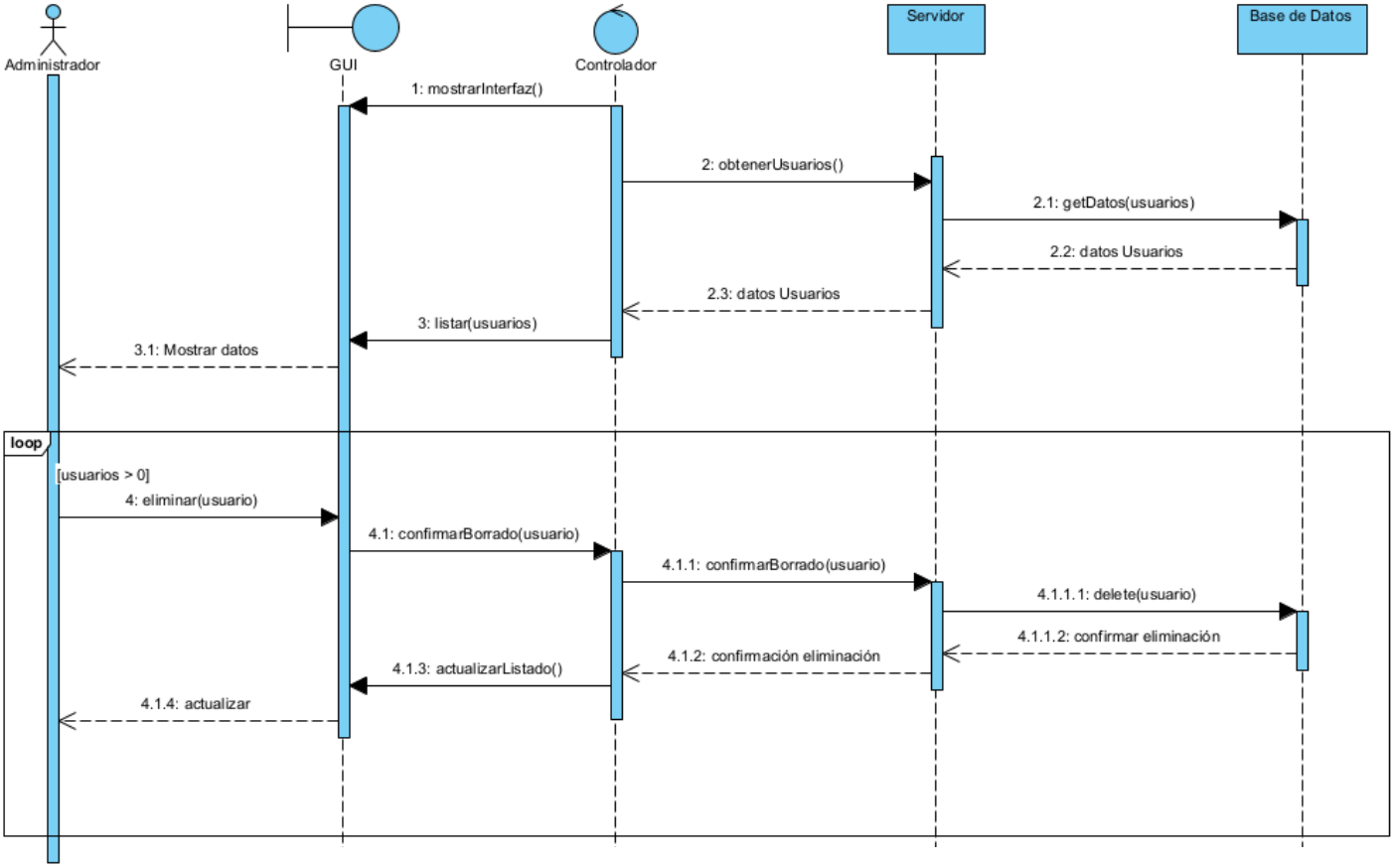


Ilustración 50: Diagrama de secuencia Eliminar usuarios (autor).

- **Añadir modelos parcelas (**Ilustración 51**):** Funcionalidad asociada al administrador y que consistirá en introducir en el servidor los ficheros asociados a los modelos con los nombres correctos correspondiente a las parcelas asociadas. Para realizar este proceso se deberá distinguir entre dos tipos de modelos

    - La inserción de **nube de puntos** consistirá en la subida de un fichero en el formato que se indique con el modelo

    - La inserción de **ortofotos / texturas** se realiza subiendo la imagen al servidor, además de manera opcional se podrá adjuntar un mapa de alturas.

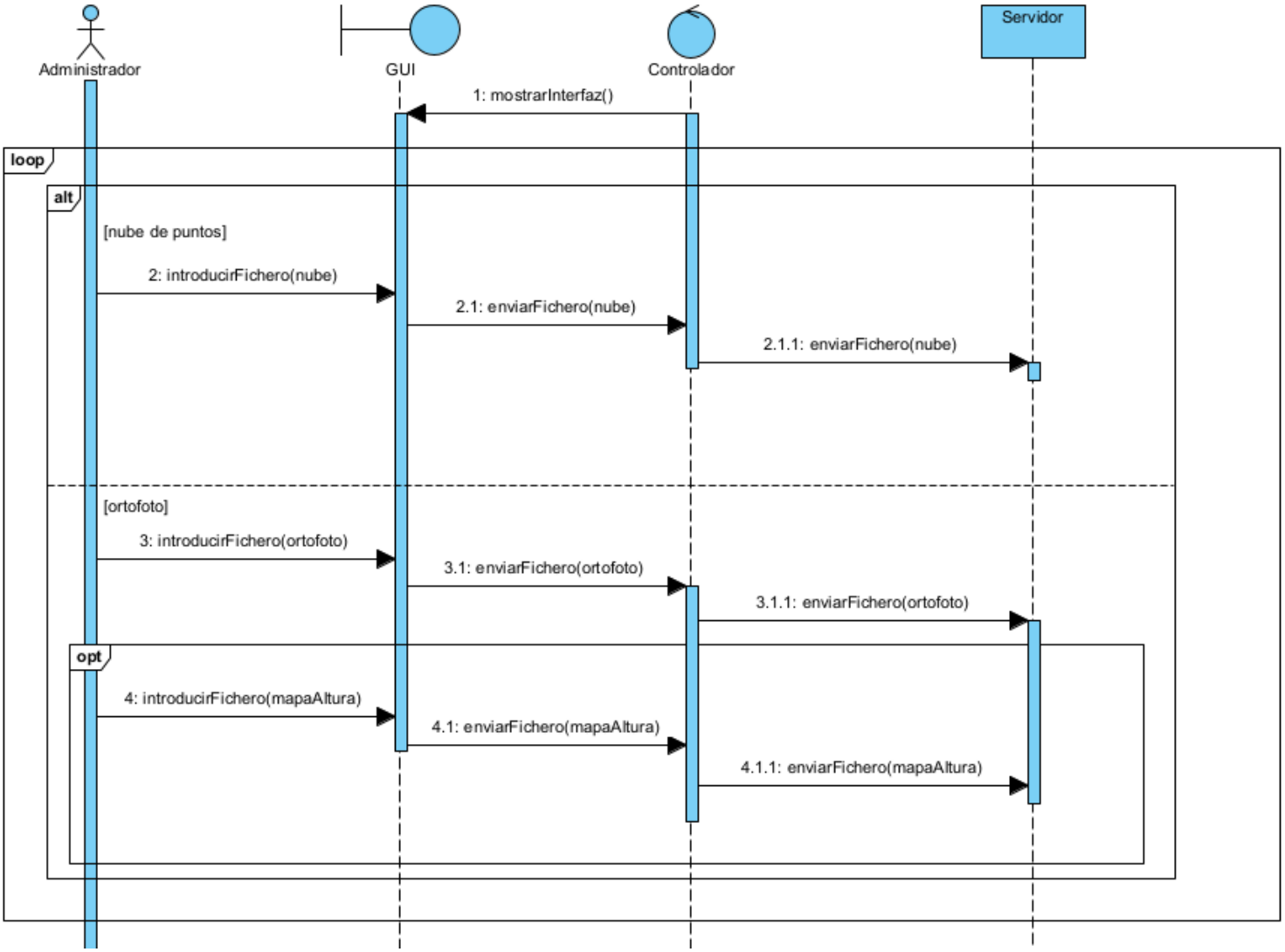


Ilustración 51: Diagrama de secuencia Añadir modelos (autor).

- **Eliminar modelos parcelas (**Ilustración 52**):** Funcionalidad asociada al administrador y que consistirá en eliminar del servidor los ficheros asociados a las parcelas concretas de los modelos. Se podrá distinguir entre la eliminación de nubes de puntos que consistirá en eliminar los ficheros correspondientes y la eliminación de texturas / ortofotos que conlleva la eliminación de la imagen, así como su mapa de alturas correspondiente.

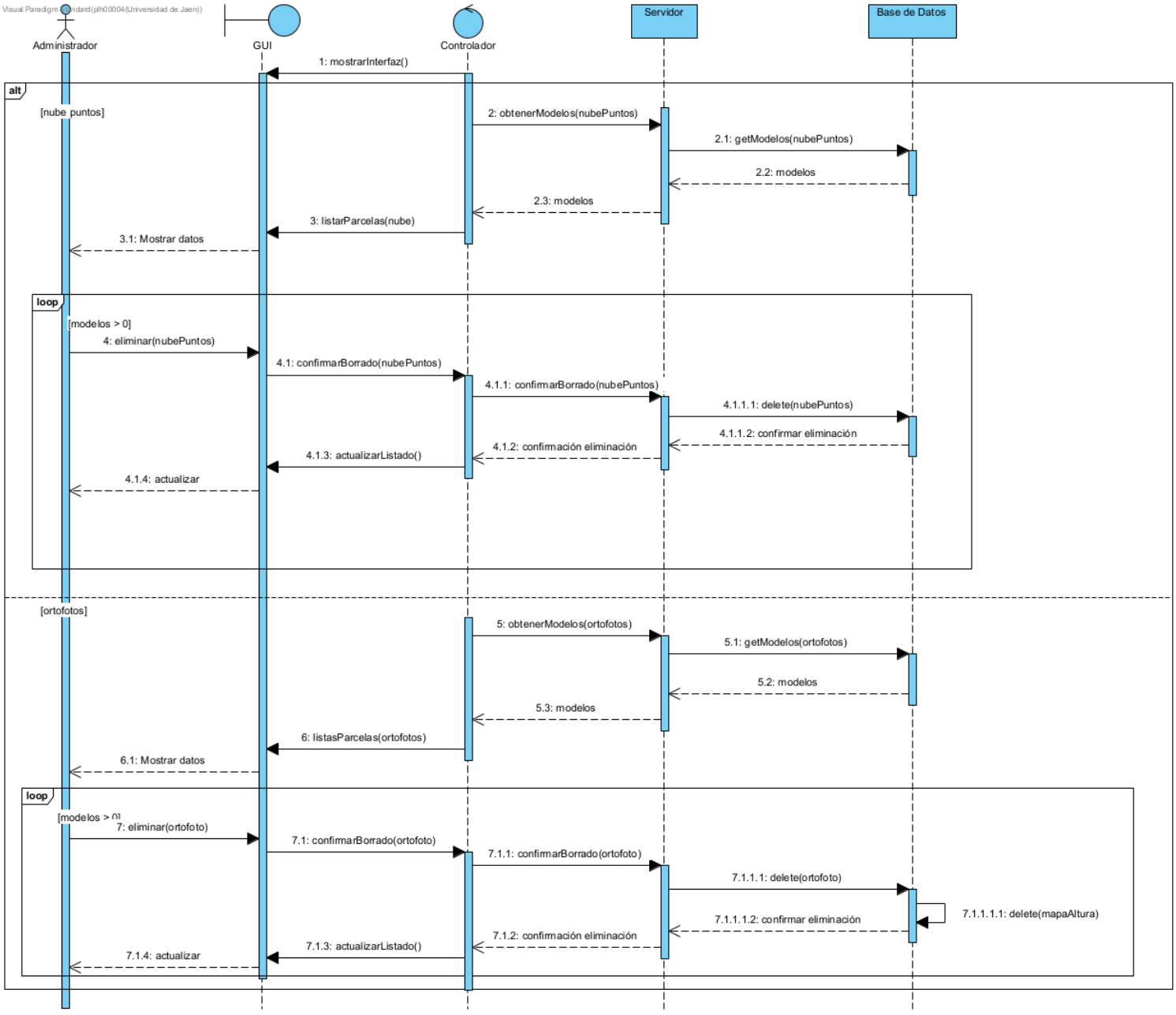


Ilustración 52: Diagrama de secuencia Eliminar modelos (autor).

- **Ver / trabajar escena (**Ilustración 53**):** Funcionalidad disponible para todos los usuarios, la funcionalidad básica será ver la escena, pero también se podrá disponer de otras funcionalidades optativas dentro de la misma. Estas optativas serán moverse por la escena, cambiar de modelo, recortar el modelo…

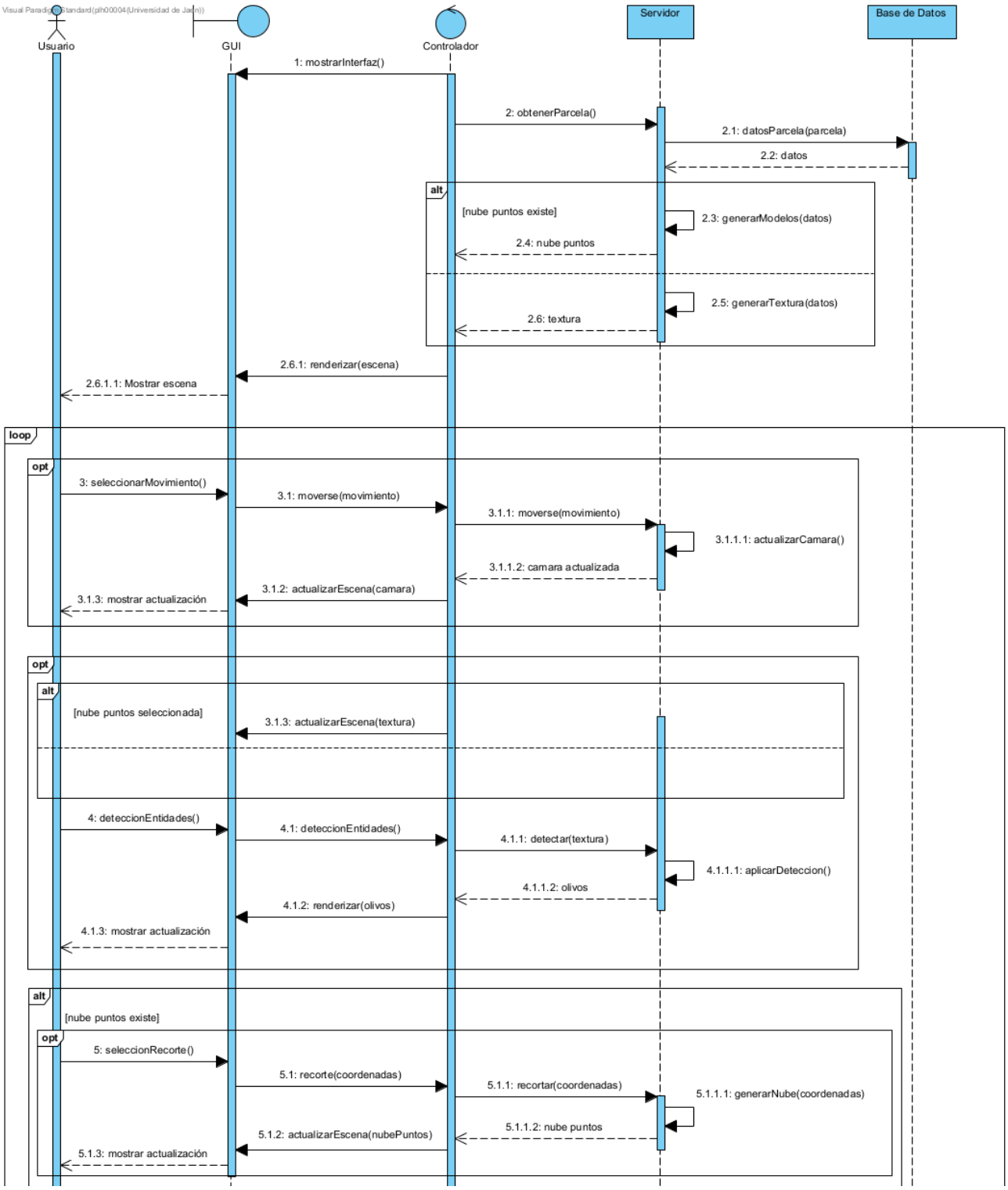


Ilustración 53: Diagrama de secuencia Ver / Trabajar escena (autor).

3.4 Análisis de riesgos

Todos los proyectos surgen con unos ideales y tareas muy concretas en sus inicios sin embargo no siempre estos pueden cumplimentarse de manera satisfactoria y sin percances es por este motivo que al inicio del proyecto es importante realizar un listado de los posibles problemas y riesgos que podemos encontrarnos a lo largo del desarrollo, como afectarían al mismo, el impacto que tendrían y su solución en caso de existir una. Por ende, en esta sección se realiza un análisis de los posibles riesgos de este proyecto.

El proceso de análisis de riesgos se suele asociar con un documento denominado matriz de riesgo, esta matriz combina el impacto del riesgo con su probabilidad de ocurrencia para determinar el nivel de riesgo existente, como podemos ver en la tabla mostrada a continuación (Ilustración 54) en función de la combinación de estos dos parámetros tendremos resultados diferentes.

3.4.1 Matriz de riesgos

Para la correcta realización del análisis de riesgos se ha empleado la matriz y la terminología correspondiente al artículo "**The Application of Risk Matrix to Software Project Risk Management"** [15]**.** Este articulo evalúa el nivel de riesgo en función de la probabilidad de que ocurra el problema y el impacto que tendría en caso de ocurrir.

Impact

Probability (%)	N (Negligible)	Mi (Minor)	Mo (Moderate)	S (Serious)	C (Critical)
86–100	Low	Medium	High	High	High
61–85	Low	Medium	Medium	High	High
41–60	Low	Low	Medium	Medium	High
21–40	Low	Low	Low	Medium	Medium
0–20	Low	Low	Low	Low	Medium

High
Medium
Low

Ilustración 54: Matriz de riesgos asociada al análisis de riesgos[15]

[15] https://acortar.link/SfEAhS

El impacto se podrá dividir en 5 grupos que vendrán determinados por las siguientes descripciones:

- Despreciable: Impacto muy leve, no afecta al cumplimiento de los objetivos del proyecto.
- Menor: Impacto leve, puede alterar el cumplimiento de algún objetivo secundario del proyecto, pero nunca de uno primario.
- Moderado: Impacto moderado, no cumple algunos objetivos secundarios.
- Serio: Impacto consistente, no cumple ningún objetivo secundario, pero si todos los primarios.
- Crítico: Máximo impacto en el proyecto, afecta a la base y los principales objetivos del mismo.

Como podemos ver en esta tabla se distinguen 5 niveles de impacto en caso de que el problema ocurra (Despreciable, Menor, Moderado, Serio, Critico) y 5 niveles de probabilidad de que este ocurra como son:

- 0-20%: Muy baja
- 21-40%: Baja
- 41-60%: Media
- 61-85%: Alta
- 86-100%: Muy alta

A continuación, se comentarán los posibles problemas que podemos encontrarnos en este proyecto y el riesgo de los mismos en función de la tabla superior. Los valores que se otorgan son subjetivos basados en la experiencia durante el desarrollo del proyecto y en trabajos previos del autor de este trabajo.

Problema	Probabilidad	Impacto	Riesgo
Falta de los datos necesarios para el funcionamiento del sistema	Baja	Serio	**Medio**
Interfaz poco usable	Baja	Moderado	**Bajo**

Baja escalabilidad y estabilidad del sistema	Alta	Serio	**Alto**
Rendimiento inadecuado del sistema	Media	Moderado	**Bajo**
Problemas de integración de las diferentes partes desarrolladas	Media	Moderado	**Medio**
Incumplimiento planificación	Baja	Moderado	**Bajo**
Cambios de requisitos u objetivos principales	Muy Baja	Serio	**Bajo**
Incumplimiento de los requisitos u objetivos principales	Baja	Moderado	**Bajo**
Gestión del proyecto inadecuada	Baja	Serio	**Medio**
Pérdida total de los recursos base del proyecto	Baja	Critico	**Medio**

Tabla 14: Recopilación de posibles problemas del proyecto y su riesgo

Una vez se han determinado los riesgos y se han analizado es necesario determinar de manera previa al inicio del proyecto las acciones a tomar en caso de que se cumpla cada uno de estos riesgos. Las posibles acciones determinar en el artículo en el que se basa la sección y la matriz [15] se clasifican en:

- **Terminar**: No es posible eliminar el riesgo sin eliminar o terminar la actividad que lo provoca o bien escogiendo otra alternativa para conseguir el objetivo que planteaba la actividad problemática.
- **Tolerar**: En ocasiones es complicado solventar un problema y puede ser más costoso arreglarlo que aceptar que está ahí y no actuar contra él.
- **Transferir**: Para algunos problemas la mejor solución en ocasiones es traspasar la responsabilidad a otra entidad para que sean ellos los que busquen solución
- **Reducir**: Es la acción más común, se basa en reducir el riesgo a niveles aceptables para el correcto desarrollo del proyecto sin la necesidad de eliminarlo por completo

Una vez conocidas las posibles acciones a tomar es necesario indicar la acción elegida para cada problema mencionado anteriormente (Tabla 14) y el plan para poder realizarla correctamente. Estas acciones serán:

- **Falta de los datos necesarios para el funcionamiento del sistema**
  - Acción: Transferir / Reducir.
  - Plan de contingencia: Solicitud a empresas especialistas en la obtención de los tipos datos requeridos su obtención y posterior envío de estos al desarrollador de este trabajo para poder utilizarlos.

    A su vez se realiza un proceso de búsqueda y obtención de todos los datos de utilidad disponibles en la web por parte del desarrollador del sistema.
- **Interfaz poco usable**
  - Acción: Reducir.
  - Plan de contingencia: Seguimiento exhaustivo del desarrollo de la interfaz en cada iteración del proyecto realizando comprobaciones y comparaciones con los estándares de usabilidad asegurando que se cumplen, así como mejorando la misma mediante la inclusión de todas las funcionalidades necesarias.
- **Baja escalabilidad y estabilidad del sistema**
  - Acción: Reducir
  - Plan de contingencia: Análisis previo al desarrollo de las capacidades máximas del sistema con el que se va a trabajar reduciendo el volumen de datos a manejar dentro de los límites del mismo manteniendo siempre la funcionalidad e integridad de los datos y modelos utilizados.
- **Rendimiento inadecuado del sistema**
  - Acción: Reducir
  - Plan de contingencia: Optimización de todas las funcionalidades desarrolladas en el momento posterior a su realización consiguiendo así obtener siempre el sistema con mejor rendimiento posible.
- **Problemas de integración de las diferentes partes desarrolladas**

- Acción: Reducir
- Plan de contingencia: Investigación inicial de las compatibilidades existentes entre las distintas tecnologías utilizadas para el desarrollo de cada parte del proyecto, asegurando así que la integración es posible, en caso de que no lo sea buscar otras tecnologías disponibles para conseguir esa parte del trabajo o bien formas de conectar en forma de puente mediante el uso de otras herramientas las tecnologías utilizadas.

- **Incumplimiento planificación**
  - Acción: Tolerar
  - Plan de contingencia: Este proyecto se enmarca dentro de un TFG con un límite no estricto de 300 horas, al disponer de esta flexibilidad se plantea la opción de exceder el tiempo estipulado con el fin de acabar cumpliendo la planificación y conseguir todos los requisitos iniciales.

    Si se diera el caso de no poder exceder estos tiempos por ser una restricción estricta se procederá a reducir reuniones o partes del proyecto que limitan tiempo del desarrollo del mismo, así como funcionalidades no necesarias o principales.

- **Cambios de requisitos u objetivos principales**
  - Acción: Reducir
  - Plan de contingencia: Realización al inicio del proyecto una planificación lo más flexible posible con la idea de poder adaptarse a posibles cambios existentes durante la realización del proyecto.

    En el momento de los cambios realizar un ajuste en la planificación y organización del proyecto, así como un análisis del proyecto existente hasta la fecha determinando que partes son útiles para los nuevos objetivos y cuáles no, para facilitar este proceso se deberá disponer de un proyecto bien modularizado desde el comienzo.

- **Incumplimiento de requisitos u objetivos principales**
  - Acción: Terminar
  - Plan de contingencia: Desde el inicio del proyecto se deberá priorizar y tener claro de primera mano siempre cuales son los objetivos

principales que guían este proyecto siendo estos siempre prioridad en el desarrollo del mismo.

Se deberá tener un control del cumplimiento de los mismos y en el caso de incluir tareas o implementaciones que limiten o incumplan estos requisitos se deberán eliminar y buscar otras opciones para conseguir los requisitos secundarios que se buscaban.

- **Gestión del proyecto inadecuada**
    - Acción: Tolerar
    - Plan de contingencia: En caso de que la organización entre miembros del proyecto (desarrollador y clientes / organización) no sea la mejor y no se produzca un seguimiento acorde a lo necesario se deberá tratar de continuar con el trabajo y mantener las tareas a realizar conforme a los últimos detalles y actualizaciones de planificación y trabajo estipulados.
- **Pérdida total de los recursos base del proyecto**
    - Acción: Transferir
    - Plan de contingencia: En el inicio del proyecto se debe establecer una ubicación de los recursos principales del mismo almacenando copias de seguridad y copias en repositorio o la nube, en caso de que la perdida se diera en todos estos casos o la copia no se realizara de manera correcta perdiendo la versión más avanzada se transferirá la responsabilidad a empresas o software especializado en la recuperación de elementos perdidos.

Capítulo 4: TECNOLOGÍAS Y HERRAMIENTAS

En este capítulo se tratarán los temas relativos a las tecnologías utilizadas para el desarrollo de la aplicación, desde lenguajes de programación, entornos de desarrollo, software de terceros para edición, control o documentación. En primer lugar se detalla la solución propuesta para la correcta realización del proyecto, así como el estudio de las diferentes alternativas que se plantearon para solventar los problemas y tareas necesarias. En última instancia se comentarán las herramientas usadas, así como la justificación de su utilidad para entender porque fueron las elegidas.

4.1 Descripción de la solución propuesta

En esta sección se describe de manera breve la propuesta de solución planteada para solventar el proyecto siendo capaces de cumplir todos los objetivos propuestos en el mismo y que pudimos ver en las Secciones 1.5 y 1.6. Es importante remarcar que al tratarse de una aplicación web el esfuerzo se ha visto dividido en dos partes principales, la parte de desarrollo de una buena interfaz donde poder acoplar las funcionalidades deseadas y la parte con el desarrollo de la escena tridimensional y todos los algoritmos y métodos necesarios para obtener la información deseada de la misma.

En primer lugar, es necesario el desarrollo de un entorno o interfaz coherente y accesible para todos los tipos de usuarios que quieran disponer de la aplicación, esta interfaz está sustentada por una base de datos MariaDB, al encontrarse en todo momento en un servidor esta aplicación requerirá de acceso a un punto de internet para su uso.

El uso de la aplicación se encuentra restringido mediante un sistema de inicio de sesión consiguiendo así que solo las personas certificadas puedan utilizar la misma ya que existen datos que puede ser vulnerables en su uso y exposición. Una vez dentro del sistema el usuario es libre de poder moverse por la aplicación observando las parcelas que desee, tanto su información temática y zonificada según usos como la información tridimensional.

En segundo lugar, tenemos la parte de la escena modelada de la parcela seleccionada previamente, dentro de este apartado de la solución se pueden destacar dos aspectos como son el montaje de la escena y carga de los datos, así como el desarrollo de funcionalidades para que el usuario pueda trabajar con la escena.

En el montaje de la escena podemos destacar el muestreo de distintos modelos en función de las necesidades, se dispone de una nube de puntos corregida mediante referencias GPS con PIX4D, consiguiendo así que coincida su posición en el espacio con sus coordenadas originales en el mundo. De manera paralela se desarrolla el uso de una textura con relieve extraída mediante imágenes ráster y satelitales, tras conseguir que estas coincidan en la escena tras alinearlas es posible aplicar segmentación de imágenes sobre la textura identificando la posición de los olivos, gracias a que las dos fuentes de datos se encuentran alineadas finalmente se procede con una proyección de los puntos marcados en la textura sobre la nube de puntos voxelizada previamente pudiendo obtener así las posiciones exactas de los olivos pudiendo almacenar estos datos previamente desconocidos e instanciar modelos 3D para su visualización.

Por último, se desarrollarán funcionalidades para que el usuario pueda trabajar de manera fácil y completa con la información disponible realizando ediciones del terreno y manejando el modelo dinámicamente. Durante el desarrollo de estas funcionalidades será posible indagar en nuevos aspectos de las zonas seleccionadas sin modificar la información base original, manteniendo así la integridad de los datos. Las distintas funcionalidades que se proponen para un sistema completo es la visión y movimiento del modelo, posibilidad de emplear distintos puntos de visión y ediciones geométricas de los modelos.

El esquema de un proceso básico de uso de la aplicación que constaría de un inicio de sesión, selección de parcela y aplicando de una funcionalidad como la detección de entidades se puede observar en la Ilustración 55.

Ilustración 55: Diagrama esquematizado que muestra el proceso completo de las funcionalidades básicas de la escena del proyecto (autor).

4.2 Estudio de alternativas y viabilidad

En esta sección se indican alternativas que se han planteado y considerado a lo largo del trabajo ya que no siempre la primera opción de cara a un proyecto es la resultante y elegida final, de esta manera se describirán las distintas ideas que nos han permitido llegar al final del proyecto.

Al principio del desarrollo se tuvo que tomar una decisión relativa al desarrollo base de la aplicación web junto con su integración con un motor 3D como era BabylonJS para esto se plantearon dos opciones como fueron hacer la aplicación web en WordPress o hacerla manual mediante el uso de plantillas y / o programación del código de la página.

En un principio se probó a utilizar WordPress y desarrollar unas páginas base de ejemplo donde se vieran escenas tridimensionales, sin embargo, al poco tiempo de uso y manejo con BabylonJS se determinó esta opción como invalida ya que ofrecía muy poco manejo de las escenas, así como muy poca versatilidad en el uso global y manejo de la misma. Por estos motivos se optó por probar la segunda opción de realizar la implementación de la aplicación web mediante programación directa de la misma además de la integración de ciertas plantillas de ayuda de la librería Bootstrap.

Esta segunda opción al basarse en un manejo propio de todo el código de la página facilitaba la gestión y manejo de las escenas tridimensionales de BabylonJS, así como la libertad de comunicar entre sí con facilidad todas las páginas y modificar la escena al completo gusto del desarrollador.

Otro punto durante el desarrollo del proyecto que generó la presencia de diferentes alternativas para el estudio de viabilidad de las mismas fue durante el desarrollo de funcionalidades de la propia escena ya que era necesario determinar como de posible y necesario era añadir capas de información al mismo como es el caso de las texturas con relieve o altura.

En primer lugar y para determinar esto se debe comprobar si realmente puede aportar un nivel de información lo suficientemente importante como para considerar comenzar con su ejecución asique era necesario plantearse que aportaba al proyecto que no tuviera ya este dentro de la escena y la respuesta fue clara ya que el trabajo con texturas e imágenes 2.5D nos permitían trabajar con librerías de procesamiento de imágenes para realizar acciones como la segmentación de entidades.

Una vez conocida la utilidad era necesario estudiar la viabilidad y la mejor alternativa para su uso. Como se comentará en la siguiente sección la opción seleccionada fue finalmente la librería OpenCV especializada en procesamiento visual. Dentro de la misma existen múltiples algoritmos y utilidades que permite segmentar las mismas pudiendo determinar posiciones de entidades, cantidad de las mismas… Estas van desde técnicas de clustering, umbralización, clasificación u otras varias. En este proyecto se emplearán varias de estas técnicas en combinación ya que ofrecen una gran colaboración y mejoran la segmentación de los datos para las pruebas realizadas.

Finalmente quedaba confirmar la viabilidad de la inclusión de texturas con elevación así como sus procesos de segmentación, para esto se probaron diferentes formatos de OpenCV ya que este se encuentra desarrollado en diferentes lenguajes de programación, primeramente se probó con el lenguaje mayormente utilizado en el proyecto como es

JavaScript, sin embargo, rápidamente se produjo el paso a C++ que es el lenguaje nativo de OpenCV ya que nos ofrecía una mayor gama de posibilidades en cuanto al manejo de los datos y el uso de las funcionalidades necesarias para la segmentación, sumado a la facilidad del lenguaje y la gran cantidad de documentación disponible.

En cuanto a la inclusión de las texturas surgieron múltiples formas de incluir la elevación de las mismas pues, aunque todas radicaban en el uso de la funcionalidad de mapas de altura era necesario determinar el formato en que le pasamos este mapa pues puede ser mediante un MDT (Modelo Digital del Terreno), un fichero de curvas de nivel, un mapa de sombras o la simulación de pendiente mediante inclinación de un polígono 2D. A lo largo del proyecto se han probado todos estos y aunque son muy similares fue la última opción (simular inclinación) debido a la sencillez y el problema de resolución del resto de datos el que dio los mejores resultados a la hora de simular al terreno real.

4.3 Herramientas utilizadas

A continuación se listan todas las herramientas y lenguajes utilizados a lo largo del proyecto divididos y clasificados según sus características y como pueden afectar al proyecto. La siguiente ilustración es un resumen visual de todo lo que veremos representado en este apartado.

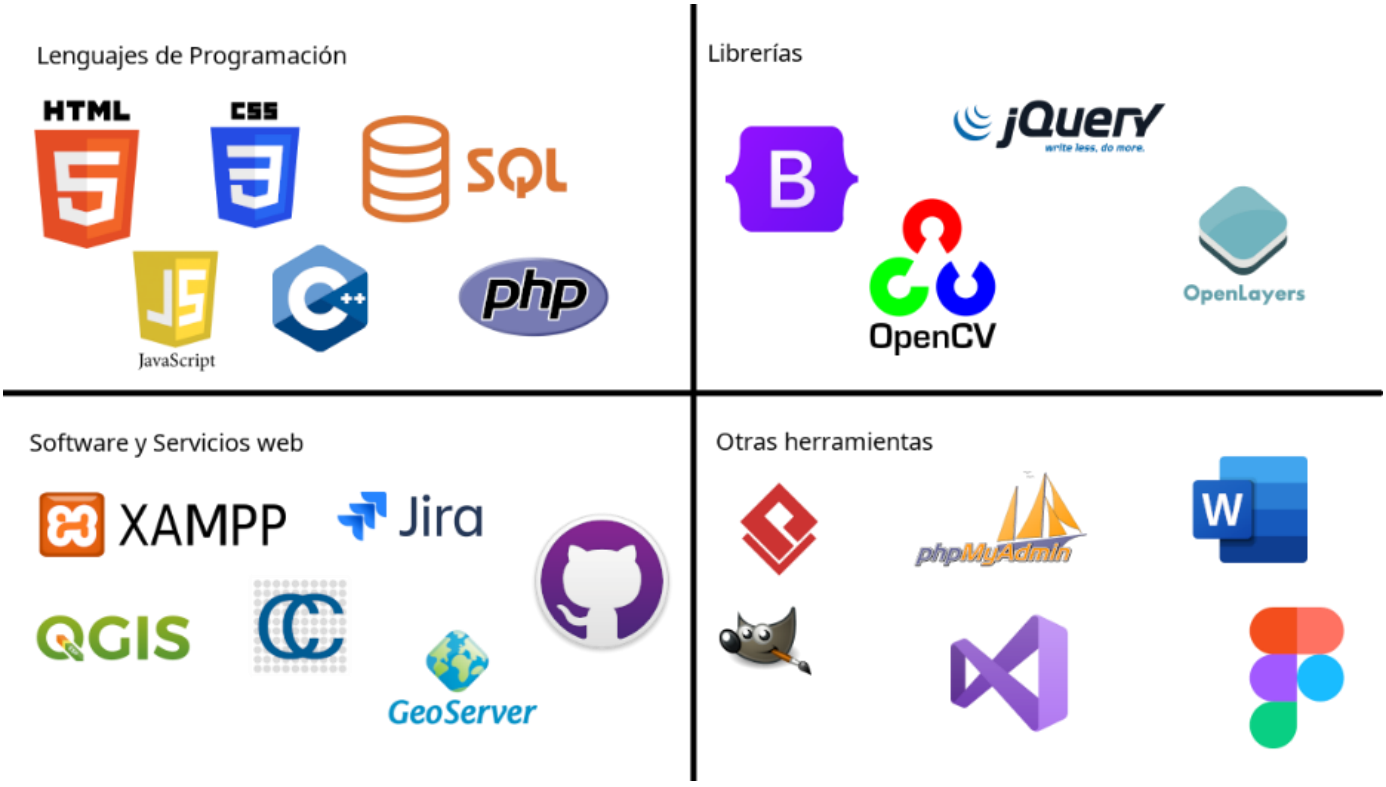


Ilustración 56: Esquema en formato imagen de las tecnologías utilizadas (autor).

4.3.1 Lenguajes de programación

Durante el desarrollo del proyecto ha sido necesario el uso de múltiples lenguajes de programación en función principalmente del ámbito de la aplicación en el que nos movamos. Los distintos lenguajes utilizados han sido:

- **HTML5 (HyperText Markup Lenguage):** Es un lenguaje de marcado implementado para la elaboración de páginas web, ha sido utilizado en este proyecto para la implementación del cuerpo de la aplicación web debido a que es uno de los estándares de programación más avanzados y utilizados.

- **JavaScript:** Es un lenguaje de programación interpretado orientado hacia el lado del cliente e implementado como parte de un navegador web. Por este motivo es el lenguaje escogido para el desarrollo de las distintas interacciones con la aplicación web a nivel de interfaz de usuario, además es el lenguaje que hay detrás del motor 3D que se comentara más adelante se ha utilizado para el desarrollo de la escena.

- **SQL (Structures Query Lenguage):** Lenguaje de programación de dominio especifico, es un lenguaje diseñado para la administración y recuperación de información de bases de datos relacionales. En el desarrollo de este proyecto se dispone de una base de datos implementada en Oracle Database por ende el lenguaje para su manejo y consulta será este mismo.

- **PHP:** Es un lenguaje de programación interpretado que funciona del lado del servidor y que se especializa en el uso para desarrollo web. Este lenguaje se ha utilizado en el proyecto para obtener datos directamente de la base de datos y enlazarlos con la página web, pudiendo realizar consultas en tiempo real mediante uso de tecnologías asíncronas. Este puede ser también empleado para ejecución de scripts en otros lenguajes mediante su llamada a la consola de comandos.

- **C++ 20:** Es un lenguaje de programación muy potente y de alto nivel es un lenguaje muy potente en el procesamiento de aplicaciones gráficas y procesamiento visual gracias a la gran cantidad de librerías de las que está dotado el lenguaje, esto provoco que fuera el lenguaje elegido sobre otros como Python para aplicar las funcionalidades requeridas.

- **CSS (Cascading Style Sheets):** Es un lenguaje de programación basado en el diseño gráfico empleado principalmente para definir y crear estilos en documentos desarrollados previamente en lenguaje de marcado. Este lenguaje es utilizado en el proyecto para otorgar estilos a la aplicación web haciéndolo responsive y más estético, además permite jugar con la visualización de los elementos.

4.3.2 Librerías

A lo largo del proyecto ha sido necesario utilizar distintas librerías con el fin de poder acceder a recursos o realizar operaciones para solventar problemas que surgen a lo largo del trabajo.

- **Bootstrap:** Es una librería multiplataforma de código abierto especializada en el diseño e implementación de páginas y / o aplicaciones web. Contiene una gran cantidad de plantillas de diseño que pueden ser utilizadas y adaptadas a lo que se necesite. Solo es de uso para la parte front-end de la aplicación.

- **OpenCV:** Librería de código abierto disponible en múltiples lenguajes de programación como Python, Java, JavaScript o C++ siendo este último en el que está desarrollado originalmente. Esta librería basada en visión por ordenador y aprendizaje automático es utilizada en el proyecto con el fin de realizar segmentación de imágenes para estimación de la presencia de olivos gracias a la gran cantidad de algoritmos ya implementados.

- **OpenLayers:** Librería de código abierto que emplea el lenguaje JavaScript utilizada para el muestreo de mapas interactivos en un navegador web. Dispone de una API que permite el acceso a fuentes cartográficas de la red, así como el manejo de fuentes descargadas previamente.

- **jQuery:** Es una librería perteneciente a JavaScript utilizada para simplificar la interacción con los documentos HTML, poder manipular y gestionar eventos, así como agregar interacción asíncrona a la aplicación web.

4.3.3 Entornos de desarrollo

Un entorno de desarrollo o IDE[16] (Integrated development environment) es aquella aplicación que facilita el proceso de programación y desarrollo del proyecto a la hora de

[16] Aplicación utilizada para ayudar en el desarrollo de creación de software.

trabajar con una serie de lenguajes. El IDE principal empleado a lo largo del proyecto ha sido **Visual Studio 2022** que nos permite trabajar tanto la parte de desarrollo de la aplicación web con la integración de HTML, CSS, JS y PHP como para el manejo de scripts que se han realizado en el lenguaje C++.

En cuanto al desarrollo de la base de datos se ha empleado **phpMyAdmin** que es una herramienta escrita en MySQL que permite la consulta y administración de bases de datos a través del navegador web.

4.3.4 Motor de desarrollo

El motor de desarrollo es el framework responsable del control de funciones y elementos correspondientes a los recursos a manejar, en nuestro caso trabajaremos con una escena tridimensional y para esta utilizaremos el motor 3D **BabylonJS** que es un motor en tiempo real que se encuentra escrito en JavaScript y cuya principal funcionalidad es la de mostrar sistemas gráficos en navegadores compatibles con HTML5. La documentación relativa a este se encuentra disponible en GitHub y en la página oficial del mismo sitio.

4.3.5 Software

Una parte muy importante en el desarrollo de un proyecto son las herramientas software que empleamos para solventar según que partes y problemas del mismo, en esta sección se incluye el software empleado y que no ha sido mencionado en secciones previas.

- **Software de soporte:** Compuesto por el software que ha sido pieza fundamental en el desarrollo del proyecto debido principalmente a su uso para completar las partes principales del mismo y la obtención de recursos.
    - **XAMPP:** Paquete de software libre que funciona mediante la integración de un servidor web apache, intérpretes de lenguaje para PHP, así como gestión y administración de bases de datos MySQL con phpMyAdmin. Este software ha sido utilizado con el fin de poder trabajar y desarrollar la aplicación web en un servidor local y poder realizar pruebas durante las fases de desarrollo de la misma.
    - **QGIS:** Software libre y de código abierto especializado en su uso para el manejo y administración de la información espacial. En este proyecto ha sido utilizado para la gestión de la información espacial, desde el manejo de capas vectoriales para identificación de parcelas

de olivar en Marmolejo, así como para el manejo de datos tipo ráster en la gestión de la altitud y otros aspectos.

- **Cloud compare:** Software especializado en el manejo y procesamiento de nubes de puntos tridimensionales, es capaz también de procesar mallas de triángulos e imágenes calibradas. Durante este proyecto este software ha sido utilizado para el manejo de las nubes de puntos de parcelas generando operaciones espaciales sobre las mismas y exportándolas en un formato asequible para el navegador y la aplicación desarrollada.

- **Software de gestión:** Es el software encargado de la gestión de los factores externos al proyecto y que no participan directamente en su proceso de desarrollo pero que permiten completar este de una forma regulada y controlada.

  - **GitHub Desktop:** Aplicación de GitHub que te permite realizar un control de versiones sin la necesidad de interactuar con la línea de comandos o un navegador web. Permite dividir el trabajo en distintas ramas según las necesidades del proyecto y tener un pleno control de las versiones estables del mismo garantizando así la integridad de la aplicación.

  - **Jira:** Software propietario especializado en la gestión de proyectos y seguimiento del avance, así como errores o incidencias. A lo largo del proyecto se ha empleado este software para ir haciendo un control de las tareas pasadas, presentes y futuras en función de si estas se encuentran completadas o no a lo largo del desarrollo de la aplicación. Permite además mantener un historial de tiempos en los que se completan las tareas desde que se proponen.

4.3.6 Aplicaciones de apoyo

- **Servicios web:** A lo largo del desarrollo del proyecto en ocasiones es necesario el uso de páginas que ofrecen servicios vía web con el fin de ofrecer acceso en tiempo real a ciertas fuentes o recursos.

  - **Geoserver:** Servicio web empleado en el proyecto, este es un servidor de código abierto escrito en Java y que permite compartir,

editar y acceder a datos geoespaciales. De esta forma se consigue acceder a los datos geoespaciales tratados previamente en QGIS para poder disponer de los mismos en la aplicación web.

- **Otras herramientas:** Además de las comentadas anteriormente existen otras herramientas empleadas a lo largo de todo el desarrollo del trabajo y que han ayudado en mayor o menor medida a conseguir completar u obtener ciertos recursos necesarios para la finalización de la mejor aplicación posible, estas herramientas son:

    - **GIMP:** Herramienta de edición y control de imágenes, empleada para editar imágenes de interés para la aplicación web como son las texturas, realizando cambios de formato y ajustando las mismas para su correcto uso en la aplicación.

    - **Microsoft Word:** Software de tratamiento de textos, esta herramienta se ha empleado para la redacción de este documento, siendo el mismo la documentación asociada al proyecto.

    - **Visual Paradigm:** Herramienta CASE (Computer-aided software engineering) de UML (Unified Modeling Lenguage), es una herramienta de soporte de modelado y proporciona la posibilidad de desarrollo de informes y diagramas. Se ha utilizado para el desarrollo de los esquemas de Ingeniería del software, así como para diagramas UML o de esquemas entidad-relación para la base de datos.

    - **Figma:** Herramienta de edición de gráficos vectorial y generación de prototipos con un enfoque en el uso de la interfaz de usuario y diseño de la experiencia del mismo en un sistema. En este trabajo su uso se debe al diseño de elementos visuales para la planificación de los diseños de la interfaz de la aplicación web.

Capítulo 5: IMPLEMENTACIÓN DEL SISTEMA

En este capítulo se encuentra el groso del desarrollo de la aplicación web, se comienza hablando de todo el proceso seguido para obtener el resultado final etapa por etapa. Como se comentó previamente en la Sección 1.7 el proceso de desarrollo ha sido iterativo y siguiendo una metodología ágil del tipo Scrumban, para esto el proyecto se ha visto dividido en seis iteraciones cuya duración y funcionalidades trabajadas se indicaron en la Sección 2.1 .

5.1 Previos del proyecto

Esta iteración como ya se comentó en apartados previos sirve para poner en contexto al desarrollador acerca de las tecnologías a utilizar, proyectos previamente existentes y posiblemente relacionados con este trabajo así como para la elaboración de un análisis inicial profundo y correspondiente al mostrado previamente en el Capítulo 3, pudiendo de esta forma trabajar con una cantidad de información inicial que permita un desarrollo acorde a los requisitos y objetivos requeridos por el usuario que dirige el proyecto.

5.2 Primera Iteración

Esta primera iteración será la que comience con el desarrollo de la aplicación. En primer lugar, se desarrollarán las distintas guías visuales dándole una explicación al porqué de los distintos aspectos y partes de las que estas disponen. En esta fase también se estructura el diseño de una base de datos para el servidor, así como se realiza una recogida de los datos de interés necesarios para el funcionamiento inicial de la aplicación. Finalmente se realiza una pequeña estructuración del sistema de directorios donde se ubica la aplicación sumada a la creación de las primeras bases de la misma.

5.2.1 Diseño de la interfaz

Esta primera acción se encuentra fuera de la fase de previos del proyecto ya que conlleva una tarea que afecta de manera directa al desarrollo de la aplicación y no puramente por análisis logísticos o de recursos. Los distintos diseños iniciales de la interfaz se mostraron previamente en la Sección 3.3.3.

Como ya se ha comentado este análisis se corresponde con las primeras maquetas de este proyecto, en futuras iteraciones si se considerara que se han realizado cambios considerables o importantes en el diseño se mostrará la evolución de la interfaz.

5.2.2 Diseño de la base de datos

En esta tarea el comedido es el desarrollo de un diseño para la base de datos, sin embargo, este ya ha sido mostrado previamente en la Sección 3.3.4, en ese apartado se mostró el diagrama entidad-relación final empleado en la aplicación y se explicó el propósito de cada tabla, en esta sección por tanto se realiza una expansión de este diagrama entrando a explicar el contenido (atributos) de cada una de las tablas y el porqué de este. En adición se han añadido las claves foráneas que no constan en el diagrama en cada una de sus tablas.

- **Usuario:** Tabla de usuarios con acceso a la aplicación

Nombre	Tipo	Descripción	Observaciones
Id_usuario	Entero	Identificador que hace único a cada usuario	Clave primaria
usuario	Cadena caracteres	Nombre propio de cada usuario dentro de la aplicación	Clave primaria, no se permite nulo
Nombre	Cadena caracteres	Datos personales	
Apellidos	Cadena caracteres	Datos personales	
Password	Cadena caracteres	Contraseña de identificación del usuario	No se permite nulo
Permisos	ENUM	Nivel del usuario dentro de la aplicación	No se permite nulo
FotoPerfil	Cadena caracteres	Datos personales	No se permite nulo
Correo	Cadena caracteres	Datos personales	

Tabla 15: Atributos entidad Usuario.

- **Datos_parcela:** Tabla de datos personales de cada parcela

Nombre	Tipo	Descripción	Observaciones
Id_parcela	Entero	Identificador único de cada parcela	Clave primaria, foránea
ReferenciaCatastral	Cadena Caracteres	Dato de referencia de cada parcela	No se permite nulo
Fecha_vuelo	Fecha	Fecha en la que se realizado un vuelo con dron sobre la parcela	
Fecha_actualización	Fecha	Fecha en la que se actualizo algún dato de la parcela	
Ortofoto	Booleano	Determina si la parcela dispone de alguno de estos modelos en la aplicación	No se permite nulo
nubePuntos	Booleano		No se permite nulo

Tabla 16: Atributos entidad datos_parcela.

- **Geometria_parcela:** Tabla de datos con la geometría de cada parcela

Nombre	Tipo	Descripción	Observaciones
OGR_FID	Entero	Identificador único de cada parcela	Clave primaria
SHAPE	Geometría	Geometría de la parcela por vértices	Clave primaria
Id_Parcela	Entero	Identificador de una parcela	Clave foránea

Tabla 17: Atributos entidad Geometria_parcela.

- **Parcela:** Tabla con los datos SigPac de cada parcela

Nombre	Tipo	Descripción	Observaciones
OGR_FID	Entero	Identificador único de cada parcela	Clave primaria
Id_recinto	Entero	Identificador de cada parcela	
nu_area	Flotante	Área de la parcela	
Cd_prov	Entero	Identificador de la provincia de la parcela	Clave foránea
Cd_mun	Entero	Identificador del municipio de la parcela	Clave foránea
Cd_uso	Cadena caracteres	Identificador del tipo de uso de la parcela	Clave foránea
Incidencia	Entero	Incidencias que sufre la parcela	Clave foránea

Tabla 18: Atributos entidad Parcela.

- **Zona:** Tabla con la clasificación de las distintas posibles zonas en las que se clasifica un terreno.

Nombre	Tipo	Descripción	Observaciones
Codigo	Cadena caracteres	Código único que determina cada zona	Clave primaria
Nombre	Cadena caracteres	Nombre que recibe un tipo de zona concreta	No se permite nulo

Tabla 19: Atributos entidad Zona.

- **Incidencia:** Tabla con las posibles incidencias que puede sufrir un terreno.

Nombre	Tipo	Descripción	Observaciones
ID	Entero	Identificador para cada tipo de incidencia	Clave primaria
Descripción	Texto	Texto descriptivo de lo referente a un tipo de incidencia	No se permite nulo

Tabla 20: Atributos entidad Incidencia.

- **Olivo:** Tabla de almacenamiento de los olivos de cada parcela y su geometría

Nombre	Tipo	Descripción	Observaciones
Id_parcela	Entero	Identificador de la parcela de un olivo	Clave primaria, foránea
Posición	Geometría	Geometría de un olivo dentro de una parcela	Clave primaria

Tabla 21: Atributos entidad Olivo.

- **Municipio_andalucia:** Tabla de los municipios de Andalucía y su geometría

Nombre	Tipo	Descripción	Observaciones
OGR_FID	Entero	Identificador único de un municipio	Clave primaria
SHAPE	Geometría	Geometría de cada municipio de Andalucía	Clave primaria
Nombre	Cadena caracteres	Nombre del municipio	No se permite nulo
Provincia	Cadena caracteres	Nombre de la provincia a la que pertenece	No se permite nulo

Cod_prov	Entero	Código de la provincia a la que pertenece	No se permite nulo
Cod_mun	Entero	Código del municipio al que pertenece	No se permite nulo

Tabla 22: Atributos entidad Municipio_andalucia.

- **Tablas históricos:** Tabla de almacenamiento de los datos históricos de cada una de las parcelas se corresponden con las tablas restantes como son Temperatura, Humedad, Precipitaciones y Producción. Se podrían desarrollar más, pero para el prototipo han sido las consideradas más importantes.

Nombre	Tipo	Descripción	Observaciones
Id_parcela	Entero	Identificador de la parcela de un olivo	Clave primaria, foránea
Fecha	Date	Fecha para la cual se dispone un valor respectivo de la parcela	Clave primaria
Nombre de la propiedad[17]	Flotante	Valor del atributo concreto en la fecha determinada	No se permite nulo

Tabla 23: Atributos entidad Olivo.

5.2.3 Recogida de datos

Con el fin de tener la aplicación lo más actualizada y enriquecida de datos posible es necesario recopilar una cantidad de información lo suficientemente extensa para cumplir con todas las necesidades del proyecto. Sera necesario distinguir distintos formatos y niveles de los que será necesario extraer información para poder entender el propósito de cada tipo de dato extraído.

En primer lugar, será necesario extraer información espacial relativa al entorno de trabajo, que en nuestro caso son las parcelas del terreno ubicado en la zona de Marmolejo. Para esto trabajaremos con la información que nos otorga de manera pública el SigPac en

[17] El nombre de este campo realmente será del tipo temperatura, humedad…

colaboración con la Junta de Andalucía[18], de esta fuente será posible extraer la información espacial relativa a todos los datos de parcelas relativos a provincias de Andalucía, además se disponen de distintas fuentes temporales, en nuestro caso seleccionamos la más actual correspondiente al 2023.

De manera conjunta a esta información espacial por parcela de olivar será necesario extraer información relativa a los datos temáticos de cada una de estas ubicaciones para poder obtener un amplio conocimiento de cada una de estas y trabajar profundamente con la misma. Estos datos se corresponderán con datos de interés, así como históricos que se obtendrán mediante preguntas a posibles propietarios de las parcelas principales a estudiar, así como información pública ubicada en la web como pueden ser datos de temperatura extraídos de AEMET (Agencia Estatal de Meteorología).

Para completa la información espacial vectorial necesaria se requerirá la descarga de otro tipo de información espacial, estando está relacionada con la división espacial en municipios, provincias u otras limitaciones espaciales dentro de Andalucía y en caso de querer expandir el proyecto a futuro dentro de toda España, esta información es posible encontrarla en portales como el IGN (Instituto Geográfico Nacional) o DERA (Datos Espaciales de Referencia de Andalucía).

Por último, será necesario extraer recursos e información espacial en otros formatos como son imágenes o nube de puntos estos más orientados hacia la representación tridimensional. En lo referido a las nubes de puntos su extracción se encuentra directamente ligada al vuelo de un UAV sobre la parcela que requiera los servicios de impresión del modelo tridimensional, este modelo luego puede recibir un post-procesamiento con el fin de darle mayor precisión espacial.

Este será el caso de nuestro proyecto en el cual podemos realizar una corrección de coordenadas con el uso de interpolación, gracias a puntos cuya coordenada real es previamente conocida, todo esto realizado con el software de PIX4D.

En lo que respecta a las ortofotos estas se encontrarán directamente extraídas del IGN usando como referencia el proyecto PNOA, se plantean durante el desarrollo dos formatos de extracción de estas, uno primero correspondiente a la descarga y post-procesamiento en QGIS o bien la obtención dinámica de estas directamente desde el IGN mediante el acceso por URL solicitando la capa mediante WMS.

[18] Página descarga parcelas Andalucía

Por último, se extraerá información espacial vectorial de datos de interés u informativos para dotar de un mayor nivel de información al mapa interactivo que presentemos posteriormente. Este tipo de información se encuentra presente en la página del Ministerio para la transición ecológica y el reto demográfico[19]. Un ejemplo de capa de interés será la relativa a las parcelas y terrenos susceptibles de producir contaminación por nitratos [16] [17].

5.2.4 Inicio desarrollo de la aplicación

Una vez se conocen los modelos finales y se ha recopilado la información necesaria es el momento de comenzar con el desarrollo de la aplicación, para el desarrollo de esta y poder realizar comprobaciones en local de su correcto funcionamiento se ha empleado XAMPP como se comentó en el Capítulo 4.

El primer paso para comenzar a desarrollar esta tarea es la instalación de este software desde el repositorio de apache Friends[20], una vez instalada la versión necesaria para el equipo en el que trabajamos toca ponerlo en funcionamiento, para esto se accederá desde el panel a la base de datos (MySQL / Admin) (Ilustración 57) y se crea una base de datos que por el momento se encuentra vacía, seguido de esto se deberá crear una carpeta para ubicar todos los recursos de la aplicación en la siguiente ruta **<path XAMPP>/htdocs/Carpeta Proyecto**.

XAMPP nos permitirá un acceso local a la aplicación vía navegador, la URL necesaria será de un formato concreto tal que:

http://localhost/carpeta/index.php

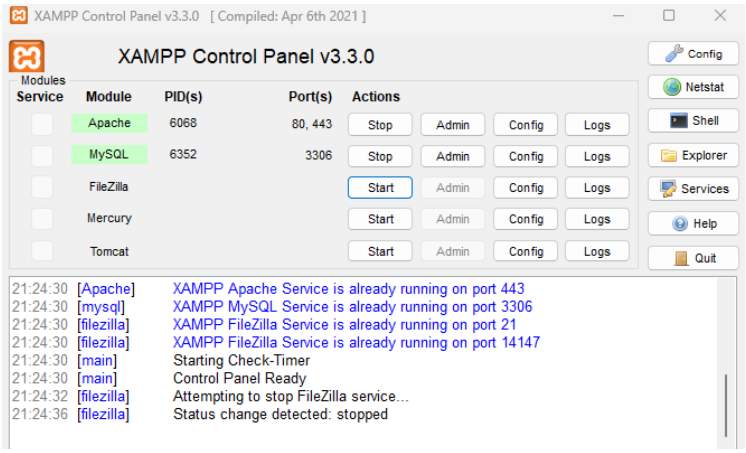


Ilustración 57: Panel de control XAMPP (autor).

[19] Página IDE del ministerio: https://www.miteco.gob.es/es/cartografia-y-sig/ide/
[20] https://www.apachefriends.org/es/index.html

La carpeta de la aplicación tendrá una estructura y jerarquía básicas establecidas desde un inicio y que se irá completando con la adición de futuros recursos que sean necesarios para el funcionamiento de esta. El esquema básico de directorios será el siguiente:

- Carpeta del proyecto
    - Ficheros php / HTML
    - Recursos (Directorio)
        - JavaScript (Directorio con todo el contenido JS)
        - CSS (Directorio con todo el contenido CSS)
        - BD (Directorio con el script de importación de la BD)
        - PHP (Directorio con todos los scripts php)
        - Imágenes (Directorio con todas las imágenes de la web)
        - Otros (Posibles adiciones de futuros directorios)

Una vez disponemos del software instalado y los directorios instanciados se comienza con el desarrollo de una aplicación web básica, esta se integra con la librería Bootstrap con el fin de obtener un diseño lo más visual posible y con un resultado estético, para esto es necesario descargar[21] los ficheros base de esta librería e incluirlos en el proyecto.

Con todos los recursos ya integrados es posible comenzar con el diseño y programación de las interfaces definidas en los diagramas previamente realizados. La versión final de estas interfaces se presenta en el apartado correspondiente a la última iteración.

5.2.5 Implementación de la base de datos

La implementación de la base de datos se ha realizado con phpMyAdmin que es un programa vinculado con XAMPP y con una interfaz web sencilla, permitiendo al desarrollador crear fácilmente la base de datos, así como obtener sentencias SQL relativas a la misma por si fuera necesario el código fuente para su uso.

[21] https://getbootstrap.com/docs/5.1/getting-started/download/

Como se ha comentado previamente el proceso de creación es sencillo, pero es importante elegir bien los datos de creación pues a posteriori será necesario disponer de estos para su integración en la aplicación. El proceso a seguir para este será:

- Creación nueva Base de datos: Darle un nombre a la base de datos y gestionar sus privilegios si fuera necesario.
- Creación de primera tabla de la base de datos
- Creación de todas las tablas necesarias para completar el diagrama entidad-relación

Algunos aspectos a tener en cuenta y que pudieran crear duda o confusión durante su uso son los siguientes:

- **Creación de claves foráneas** (Ilustración 58)**:** Para poder relacionar tablas entre si será necesario disponer de una clave foránea que nos indique entre que entidades de cada tabla existe la relación.

  La creación de esta se encuentra dentro de cada tabla en la sección estructura / vista de relaciones, una vez en este apartado se podrán crear tantas claves como se necesiten.

Ilustración 58: Sección para creación de claves foráneas phpMyAdmin (autor).

Índices autoincreméntales: En ocasiones hay atributos que por su naturaleza se requiere que una nueva inserción tenga un valor numérico una unidad superior que el anterior, esto pudiera ser el caso de un índice para clave primaria. La obtención de esta propiedad se encuentra en la creación del atributo bajo una checkbox con las siglas A_I (Auto Incrementing).

- **Encriptación de valores** (Ilustración 59)**:** En sistemas con un sistema de usuarios o backend es necesario ofrecer cierta seguridad al usuario de que

sus contraseñas no se verán expuestas o vulnerables para esto es necesario realizar una encriptación de la mismas.

Para conseguir esto con phpMyAdmin será necesario crear el campo con el formato varchar (cadena de caracteres) y a la hora de realizar adiciones a la tabla emplear la función **password_hash()** de php.

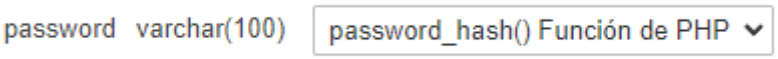


Ilustración 59: Cuadro de selección función para encriptación phpMyAdmin (autor).

Conocidas todas estas situaciones se podrá implementar sin problema una Base de datos como la desarrollada para la aplicación web del proyecto.

5.2.6 Resumen trabajo realizado

Tras finalizar esta iteración se realiza una sesión de feedback con el fin de mostrar el trabajo realizado hasta el momento. En esta reunión se explica que esta primera iteración no tiene un entregable visual de relevancia ya que ha servido para configurar y establecer los primeros pasos para comenzar el desarrollo. Se han descargado los programas necesarios y puesto en funcionamiento una base de datos con el esquema de tablas previamente definido.

En adición a esto es posible mostrar los primeros pasos de la creación de la aplicación mostrando la estética que van tomando las páginas. Por último se mostraron los datos recogidos con los que se va a trabajar a lo largo del proyecto.

5.3 Segunda Iteración

Una vez se disponen de las bases sobre las que se va a construir la aplicación será necesario ir dotando a esta de las funcionalidades básicas que serán necesarias para completarla adecuadamente, con este propósito durante esta fase se inicia la integración del motor 3D BabylonJS en la aplicación, a su vez se empieza a trabajar con las nubes de puntos disponibles para estimar el tratamiento previo necesario para su uso. Además, se deberá continuar con la estructuración de la aplicación estableciendo la jerarquía de la misma y las distintas conexiones existentes entre las páginas. Finalmente se comenzará con el trabajo sobre la librería OpenLayers con el fin de integrar un mapa interactivo que

emplearemos para ubicar las parcelas, ligada a esta funcionalidad se comienza con el tratamiento de las capas de datos espaciales recopiladas en la iteración previa.

5.3.1 Estudio e integración de BabylonJS

La principal funcionalidad de la aplicación y la que la hace distinta de las demás es la posibilidad de interactuar con facilidad sobre una escena tridimensional y poder acceder a nuestra parcela visualmente desde cualquier parte. Para conseguir esto se ha trabajado con el motor gráfico 3D BabylonJS que funciona directamente en JavaScript y dispone de una amplia documentación para poder abarcar el proyecto.

En esta iteración el propósito es conocer cómo integrar este motor en nuestra aplicación web dentro de una página HTML. Para conseguir esto hay que seguir los siguientes pasos:

- Generar un fichero HTML donde vincularemos los scripts de BabylonJS y definiremos el elemento canvas donde se ubica la escena.

```
<!-- Babylonjs Javascript -->
<script src="https://cdnjs.cloudflare.com/ajax/libs/dat-gui/0.6.2/dat.gui.min.js"></script>
<script src="https://assets.babylonjs.com/generated/Assets.js"></script>
<script src="https://preview.babylonjs.com/ammo.js"></script>
<script src="https://preview.babylonjs.com/cannon.js"></script>
<script src="https://preview.babylonjs.com/Oimo.js"></script>
<script src="https://preview.babylonjs.com/earcut.min.js"></script>
<script src="https://preview.babylonjs.com/babylon.js"></script>
<script src="https://preview.babylonjs.com/materialsLibrary/babylonjs.materials.min.js"></script>
<script src="https://preview.babylonjs.com/proceduralTexturesLibrary/babylonjs.proceduralTextures.min.js"></script>
<script src="https://preview.babylonjs.com/postProcessesLibrary/babylonjs.postProcess.min.js"></script>
<script src="https://preview.babylonjs.com/loaders/babylonjs.loaders.js"></script>
<script src="https://preview.babylonjs.com/serializers/babylonjs.serializers.min.js"></script>
<script src="https://preview.babylonjs.com/gui/babylon.gui.min.js"></script>
<script src="https://preview.babylonjs.com/inspector/babylon.inspector.bundle.js"></script>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
<script src='https://cdnjs.cloudflare.com/ajax/libs/vue/2.3.4/vue.min.js'></script>
```

Ilustración 60: Scripts necesarios para integrar BabylonJS (autor).

- Añadir un script propio de carga de una escena en BabylonJS personalizado para su funcionamiento en la aplicación.

```
Procedimiento CargaEscena():
    Intentar:
        engine <- BABYLON.Engine
    Error:
        Notificar
    Si engine = NULL Entonces
        Lanzar excepción
    Fin Si
    modelo <- ObtenerParcelaRenderizar(URL)
    scene <- window[modelo]
    Repetir
        Renderizar scene
    Hasta Que scene = NULL Y scene.activeCamera = NULL

Fin Procedimiento
```

Ilustración 61: Pseudocódigo renderizado bases de la escena (autor.)

- Añadir un script para la recreación de una escena en BabylonJS desde la cual se definirán y manejarán los elementos de la misma.

```
Procedimiento Escena(engine, parcela):
    scene <- BABYLON.scene(engine)
    camera <- BABYLON.Camera()
    ligth <- BABYLON.HemisphericLight()
    creacionTextura(parcela, scene)
    Si fichero(parcela) != NULL Entonces
        nube = creacionParcela(scene,camera,parcela)
    Fin Si
    Devolver scene
Fin Procedimiento
```

Ilustración 62: Pseudocódigo elementos básicos escena (autor).

5.3.2 Trabajo inicial con nubes de puntos

El principal tipo de datos espacial o modelo con el que trabaja la escena es con las nubes de puntos ya que son las que más información nos permitirán extraer, además son los modelos más visuales de los que dispondremos a la hora de presentar la parcela.

Estos modelos se obtienen tras un post-procesamiento realizado al vuelo de un dron sobre una parcela, sin embargo, el resultado final de este proceso no tiene por qué ser el adecuado para una correcta visualización posterior, por este motivo ha sido necesario editar estos modelos para dejarlos en la mejor forma posible para subirlos a posteriori a la web.

La aplicación utilizada para el manejo de estas ha sido Cloud Compare que nos permite una sencilla operatividad con estas. Cuando abrimos la aplicación es posible cargar directamente los ficheros, en nuestro caso trabajamos con un fichero PLY[22] (Polygon File Format) y se abrirá un espacio de trabajo como el visto en la Ilustración 63 donde veremos en el centro el modelo con el que trabajaremos. En nuestro caso pudimos observar tres problemáticas iniciales que fueron:

- Densidad de la nube de puntos: Presentaba una cantidad de 100 millones de puntos que es una densidad no asumible para su uso en un navegador web y por ende debía ser reducida

[22] Formato de archivo diseñado para almacenar datos tridimensionales de escáneres 3D.

- Orientación de la nube de puntos: La nube de puntos utilizaba un sistema de ejes donde ubicaba la altura en el eje Z, sin embargo, en BabylonJS se trabaja con la altura en el eje Y.

- Ubicación de la nube de puntos: La nube se encontraba completamente desplazada en una dirección del espacio y no centrada en el mismo

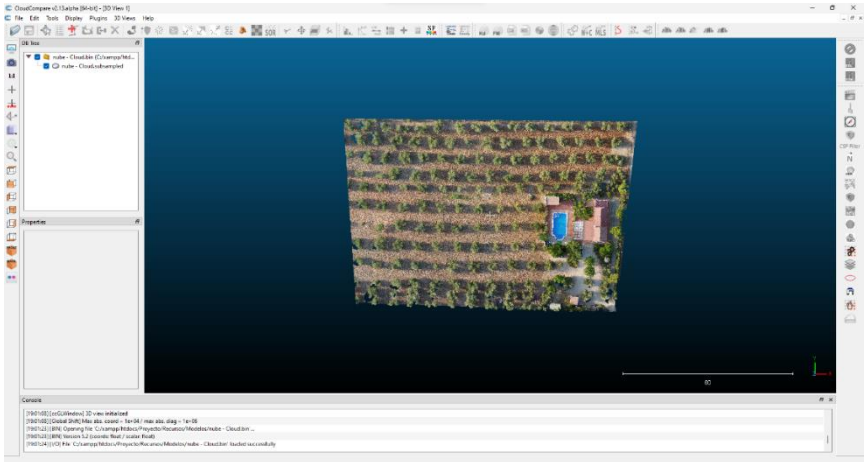


Ilustración 63: Espacio trabajo Cloud Compare (autor).

De entre estos problemas se observó rápidamente que dos tendrían muy fácil solución con las herramientas de Cloud Compare y que el problema restante podría paliarse, pero no solventarse por completo.

En primer lugar, se trabajó con los problemas espaciales, Cloud Compare dispone de la herramienta **Translate / Rotate** ubicada en la parte **Edit** de la navegación, esta nos permite seleccionar toda la nube y rotarla u moverla en los ejes que seleccionemos como se puede observar en la Ilustración 64 donde los checkbox se corresponden al eje donde se permite la traslación y el desplegable los ejes a rotar.

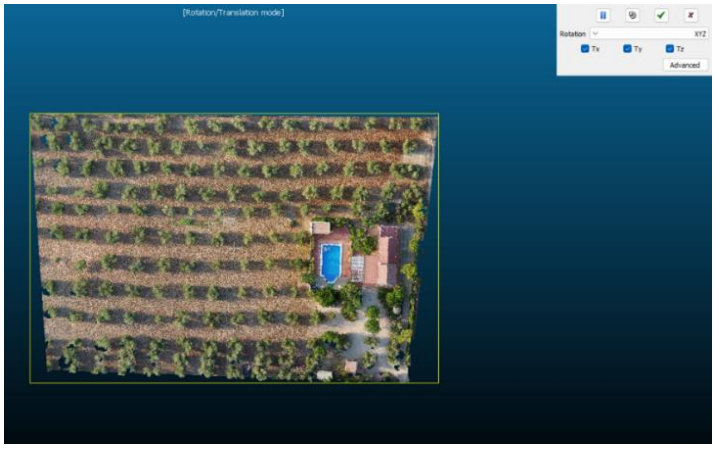


Ilustración 64: Rotación y traslación de la nube de puntos (autor).

Las operaciones se realizarán manualmente clicando sobre la propia nube y desplazándola o rotando alrededor del centro de la misma, una vez hayamos realizado los cambios necesarios seleccionamos el botón verde y se guardarán estos.

De esta forma podemos solventar el problema de la altura en el eje Z con facilidad aplicando una rotación alrededor del eje X hasta ubicar el eje Y vertical con respecto a la parcela, por el contrario con el problema de la centralidad no existe una referencia clara para solventarlo debido a la diferencia de sistemas de referencia respecto de BabylonJS por ende se ha tratado de centrar todo lo posible en la ventana para minimizar la distancia al centro pero será un problema a terminar de solventar dentro del propio entorno de programación de la escena.

Una vez tenemos la nube bien ubicada y con la posición que nosotros queremos toca reducir la densidad de puntos lo máximo posible sin perder calidad de visión y realismo de esta. Para esto se emplea la herramienta **Subsample** ubicada en el botón **Edit** de la barra de navegación. Como podemos ver en la Ilustración 65 esta herramienta lo que busca es reducir el número de puntos determinando un umbral mínimo de distancia entre dos puntos para determinar con cuales quedarnos de todos los posibles

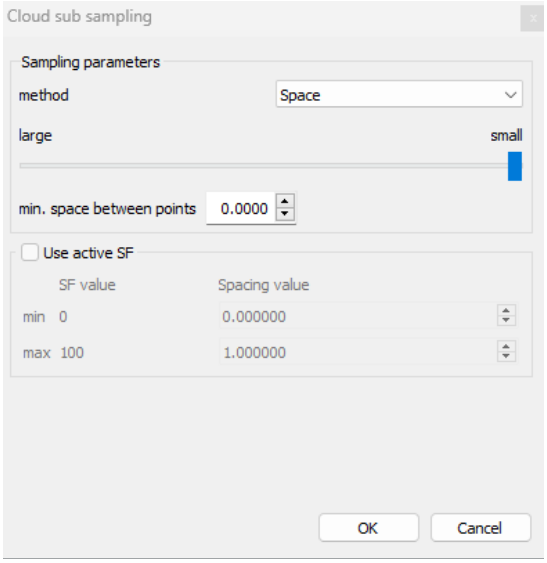


Ilustración 65: Cuadro manejo Subsampling en Cloud compare (autor).

Se realizaron pruebas con esta herramienta para obtener un valor de densidad de nube aceptable (1-2 millones de puntos) para obtener buen rendimiento en un navegador, tras realizar estas la opción elegida fue un umbral de 0.2 para obtener una nube de un total 1.281.300 puntos obteniendo una representación fiel y con una buena visibilidad.

Cuando disponemos de la nube final con todos los cambios realizados toca exportarlo en un formato adecuado para su procesamiento, previo a esto es necesario conocer qué tipo de formatos soporta BabylonJS para su interacción con nubes de puntos por lo que en esta iteración se guarda la nube en un formato estándar PLY y más adelante en el desarrollo de la carga de nubes de puntos en la aplicación se estimará el formato adecuado para trabajar con el modelo.

5.3.3 Comunicación de la aplicación

Cuando se dispone de todos los ficheros de páginas necesarios para nuestra aplicación es posible comenzar a conectarlos entre sí en función de las necesidades de cada página, esto puede conseguir mediante menús de navegación, botones o enlaces de manera que podamos movernos libremente de una página a otra de la aplicación. En la Ilustración 66 podemos observar una representación del posible flujo que podría seguir un usuario que recién entra en la aplicación en función de su nivel de usuario determinando así los niveles de comunicación y jerarquía de la aplicación.

No ha quedado detallado en el diagrama, pero es posible ir directamente a la página de inicio desde casi todas las páginas debido a que el logo ubicado en la navegación nos lleva directos a esta página siempre, la única que no lo tiene es la escena que no dispone de navegación y por ende solo podrá volver a la página de información de la parcela y desde esta podrá seguir la navegación hacia cualquiera otra página.

Una vez establecidas las posibles rutas del usuario es necesario mediante su integración en el código hacer posible este esquema mediante la inserción de herramientas de navegación o formas de llegar a las páginas requeridas cuando el usuario tome la decisión.

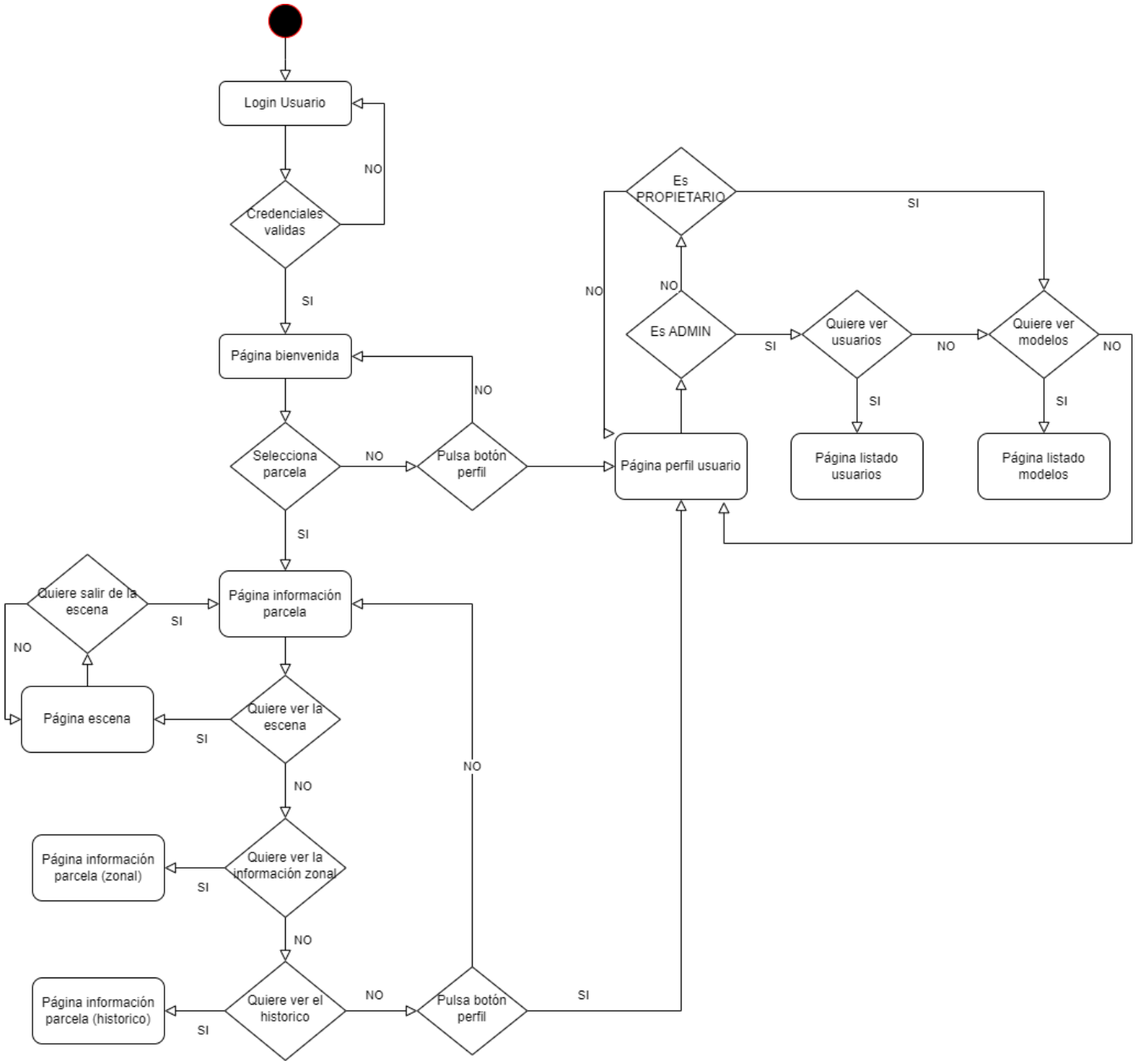


Ilustración 66: Diagrama flujo posibles opciones de un usuario en la aplicación (autor).

5.3.4 Creación del mapa en el sistema

La segunda funcionalidad más importante del proyecto se encuentra en dotarlo de la posibilidad de elegir que parcelas quiere visitar el usuario, la mejor forma para esto es conseguir una representación visual con un mapa.

Son múltiples las opciones de librerías que existen en JavaScript para integrar mapas a nuestra aplicación web, las dos con las que se plantearon y con las que se trabajó fueron Leaflet y OpenLayers siendo esta última la elegida finalmente. Primero se explicarán los motivos que nos decantaron entre una opción y la otra, posteriormente se explicará cómo se integra y desarrolla la parte básica de esta funcionalidad del sistema.

5.3.4.1 Leaflet vs OpenLayers

Estas dos librerías ofrecen la posibilidad de crear sistemas muy similares de mapas interactivos en nuestra página o aplicación web por ende el decantarse entre una u otra se definió por pequeñas diferencias en rendimiento y posibles funcionalidades extra que puedan resultarnos interesantes.

Por un lado, Leaflet [18] es más sencillo y rápido de aprender, por lo general necesita de menor cantidad de código para realizar operaciones iguales en OpenLayers. Además, tiene una gran adaptación en soporte móvil y gran cantidad de plugins.

Por otro lado, OpenLayers [19] soporta todos los protocolos WebGIS, dispone de integración rápida con WebGL con el fin de poder mostrar grandes conjuntos de datos y un diseño modular que permite hacerlo más eficiente.

Finalmente, aunque no por mucha diferencia se optó por realizar el mapa con OpenLayers debido a su eficiencia modular y manejo con grandes cantidades de datos (el número de parcelas es muy voluminoso) además de disponer de funcionalidades concretas de gran interés que se desarrollarán más adelante.

5.3.4.2 Integración OpenLayers en el sistema

La integración de OpenLayers en la página web es muy sencilla y existe una gran documentación para obtener un resultado muy satisfactorio como se comentó en la sección anterior.

Los procesos a seguir para conseguir integrarlo correctamente, aunque sin un orden prestablecido son:

- Crear un contenedor básico en HTML donde se ubica el mapa en cuestión.

```
<div id="map" class="map">
</div>
<script type="text/javascript" src="Recursos/js/Mapa/mapa.js"></script>
```

Ilustración 67: Creación elementos HTML para inserción de un Mapa (autor).

- Incluir los Script básicos para el correcto funcionamiento del mapa y sus estilos.

```
<link rel="stylesheet" href="https://cdn.rawgit.com/openlayers/openlayers.github.io/master/en/v5.3.0/css/ol.css" type="text/css">
<link rel="stylesheet" type="text/css" href="Recursos/js/Mapa/ol-ext/ol-ext.css" media="all">

<script src="https://cdn.rawgit.com/openlayers/openlayers.github.io/master/en/v5.3.0/build/ol.js"></script>
<script src="Recursos/js/Mapa/ol-layerswitcher.js"></script>
```

Ilustración 68: Scripts necesarios para el funcionamiento de OpenLayers (autor).

- Definir un Script para poder determinar el mapa, sus capas y las distintas funcionalidades en JavaScript.

Dentro de OpenLayers existen distintas formas de introducir una capa, en este trabajo nos hemos centrado exclusivamente en tres:

- Mediante un **Geoserver** con WMS o WFS, en caso de disponer de un geoserver con las capas en él es posible acceder a estas vía URL y tener acceso a ellas.

- Mediante un fichero en formato **JSON (**JavaScript Object Notation**)** que almacene la geometría de cada entidad de la capa, con este formato es necesario disponer en local del fichero, pero te aseguras no sufrir perdidas por daño en los servidores o por colapso de la geoserver.

- Mediante una URL de teselas, en este caso en un formato parecido al del Geoserver mediante una URL se solicita un mapa como puede ser el caso de Google Maps o OpenStreetMap.

A lo largo del proyecto se han utilizado todos estos formatos según las necesidades que se presentaban. Entre las funcionalidades básicas donde se encuentra la creación de las capas básicas para la navegación se emplea el método del fichero en formato JSON. El Geoserver se emplea cuando no sea rentable disponer de los datos en local.

Para los mapas base se emplearán las capas mediante URL de teselas, en nuestro caso usamos OpenStreetMap para delimitar por nombre y mostrar la cartografía del terreno y Google Maps para disponer de imágenes satélite.

Como se comentó anteriormente, OpenLayers permite integrar múltiples controles y funcionalidades, sin embargo, en esta iteración solo es necesario desarrollar una para poder navegar correctamente por la aplicación, esta se corresponde con un control de visibilidad de las capas.

Para lograr alcanzar esta funcionalidad existe una librería externa denominada LayerSwitcher cuyo código se encuentra en GitHub[23] y que tras descargar e integrar sus scripts de CSS y JS con nuestra aplicación permite introducir el control en el mapa sin mayor problema.

Finalmente, los datos empleados para la creación del mapa y los grupos de capas con los que se ha trabajado han sido:

- **Vista:** La vista es el elemento de OpenLayers que define como el usuario ve el mapa, para esta variable se ha definido un centro con coordenadas correspondientes al centro de España, un zoom de nivel 6 y unas restricciones de zoom máximo y mínimo de 19 y 6.

- **Capas:** En cuanto a las capas se han definido cuatro grupos de capas para el mapa que son el grupo base donde se encuentran los mapas de fondo, el grupo de limitaciones donde se ubican las divisiones por parcelas, el grupo de capas informativas que contienen capas vectoriales con datos de interés y por último el grupo de capas divisorias de terreno donde se ubican las capas municipales, provinciales y autonómica.

5.3.5 Trabajo con información espacial

Cuando se dispone de toda la información para trabajar con un sistema en la mayoría de las ocasiones es necesario realizar un preprocesamiento de la misma para tenerla en el formato o cantidad adecuada para un mayor rendimiento y correcto funcionamiento.

[23] Código fuente LayerSwitcher: https://github.com/walkermatt/ol-layerswitcher

En este proyecto durante la primera iteración se trató de obtener toda la información posible para el funcionamiento del sistema y en un apartado previo se realizaron las tareas de procesamiento para los modelos.

Sera en esta sección donde se procesa el resto de información espacial como son los datos y capas del SigPac. Para este trabajo se utiliza el software de QGIS.

Para este proyecto se disponen de tres tipos de capas espaciales como son los municipios de Andalucía, las parcelas de Jaén y los fotogramas del PNOA. A continuación, se tratarán de forma individual la extracción de información relativa y necesaria de cada uno de estos datos y se detallará el proceso seguido.

5.3.5.1 Municipios de Andalucía

Esta capa de información es necesaria para la distribución y clasificación de las delimitaciones mínimas que pueden contener una parcela. En nuestro caso trabajaremos exclusivamente para Marmolejo, pero de esta manera tendremos un sistema más expandido ante posibles actualizaciones y mejoras de cara al futuro.

De esta capa obtenida del DERA se esperan extraer un mapa de provincias y uno unificado de toda la comunidad para poder dividir en distintos niveles de información las capas mostradas conforme más cerca te puedas encontrar del mapa.

Para esto tendremos que realizar una serie de operaciones vectoriales en QGIS. En primer lugar, cargaremos la capa que tenemos del DERA obteniendo una visualización como se puede observar en la Ilustración 69 donde se observa la comunidad de Andalucía dividida en bastante polígonos que representan todos los municipios de esta.

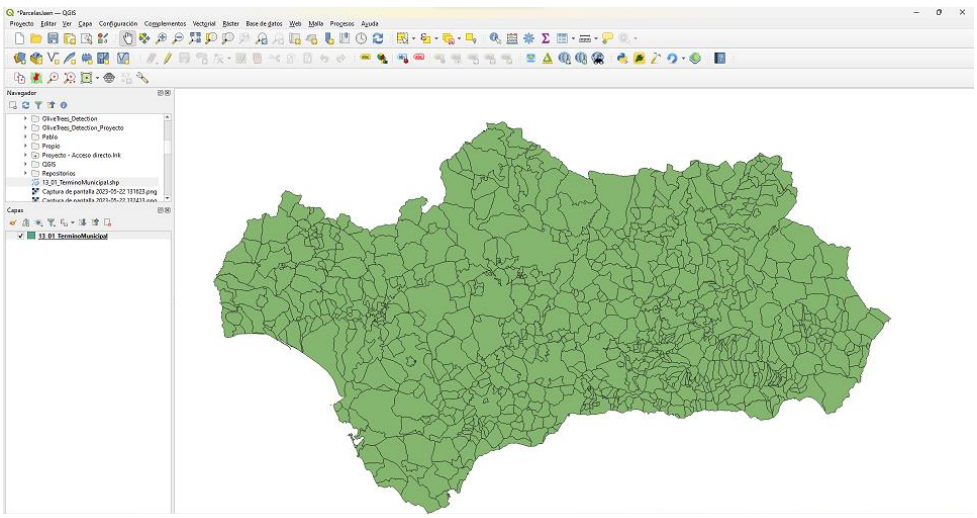


Ilustración 69: Ventana de trabajo en QGIS (autor).

En el caso de querer realizar unificaciones para obtener distintos niveles de información tendremos que utilizar la herramienta **Disolver** ubicada en el menú **Vectorial > Herramientas de geoproceso.** Esta herramienta nos permite juntar geometría en función de un criterio que determine el usuario, por este motivo debemos buscar un atributo de entre los datos de la capa que nos permitan obtener las provincias y otro para la comunidad.

- **Obtención de provincias** (Ilustración 70): Para este caso podemos observar de entre las opciones del Disolver la capa a disolver y una opción para marcar campos o atributos (Ilustración 71). De entre los campos aquel que nos será de ayuda será provincia que nos permitirá aunar las entidades con este valor en común.

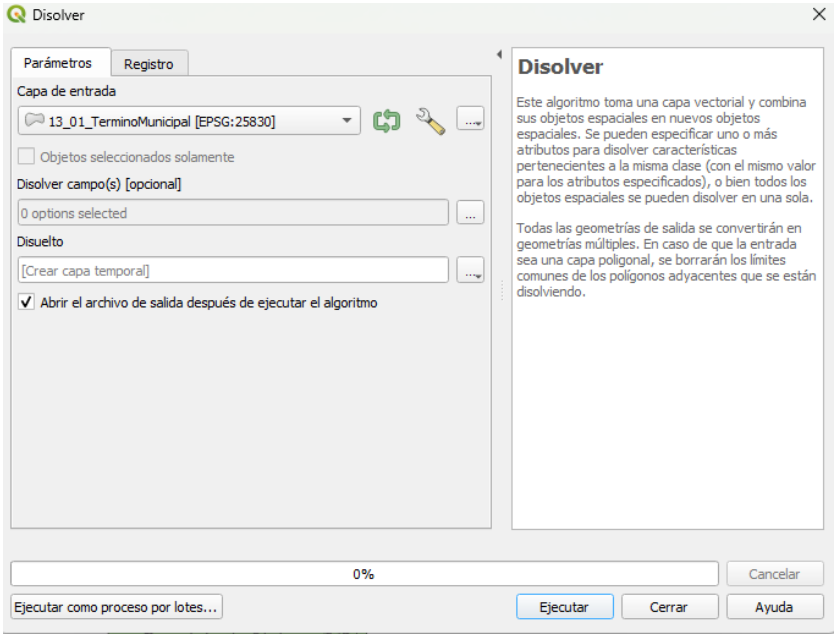


Ilustración 70: Ventana del proceso Disolver (autor).

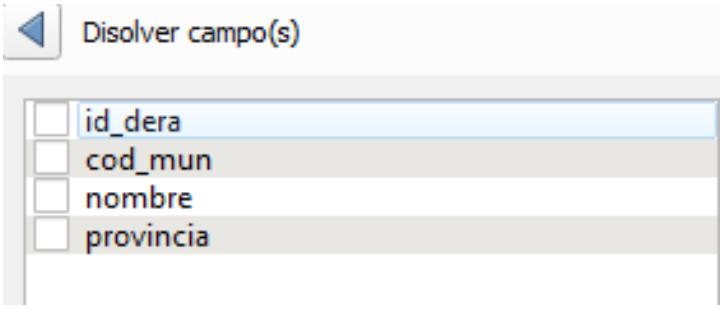


Ilustración 71: Atributos de la capa de municipios (autor).

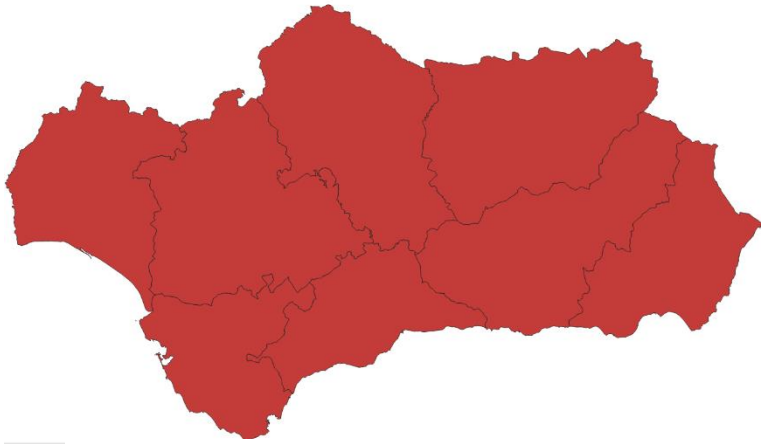


Ilustración 72: Capa resultante con la geometría de las provincias (autor).

- **Obtención de la comunidad**: En este caso el trabajo es mucho más sencillo pues estamos buscando obtener una única geometría, este trabajo conlleva simplemente ejecutar el Disolver sin ningún atributo marcado lo que lleva a esta herramienta a juntar toda la geometría en una sola.

Una vez realizados estos dos procesos contamos con dos nuevas capas en nuestro proyecto, sin embargo, estas se almacenan en formato temporal (Ilustración 73) por ende es necesario guardarlas correctamente para disponer de ellas más adelante.

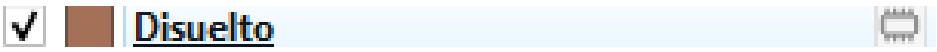


Ilustración 73: Representación de una capa temporal (autor).

Para guardarla correctamente será necesario presionar el botón secundario sobre la capa y elegir la opción **Hacer permanente** o **Exportar > Guardar objetos como.**

5.3.5.2 Parcelas de Jaén

Estas capas de información (Ilustración 75) son las más complejas y costosas de manejar debido a la gran cantidad de geometría que contienen y el extenso número de entidades. Tras empezar a trabajar con estar capas rápidamente se pudo observar la lentitud del proceso por la dimensionalidad del mismo, por este motivo se decidió acotar directamente a la zona de Marmolejo.

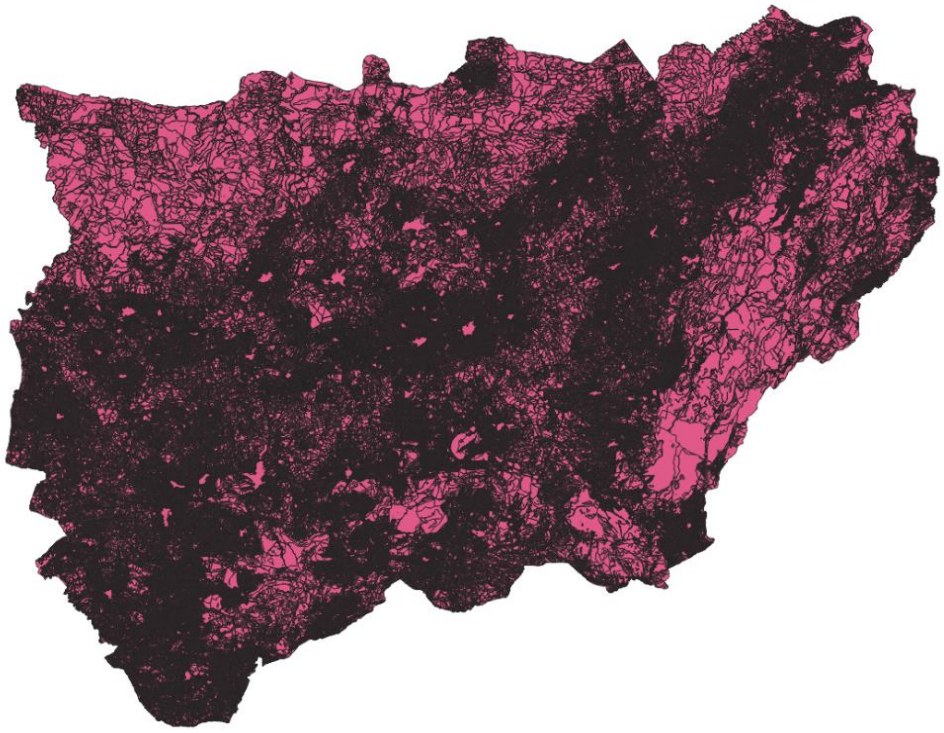


Ilustración 75: Capa parcelas de todo Jaén (autor).

Para conseguir esto se realizado un proceso de selección por valor (Ilustración 74) de atributo donde se seleccionó aquellas parcelas con el código de municipio con el valor del código de Marmolejo (obtenido previamente en la capa de municipios de Andalucía). Una vez se dispone de todas las parcelas seleccionadas (Ilustración 76) se procede a realizar un guardado de la selección de capa. Esta opción guarda exclusivamente la geometría seleccionada y se encuentra tras hacer **clic secundario en la capa > Exportar > Guardar objetos seleccionados como.**

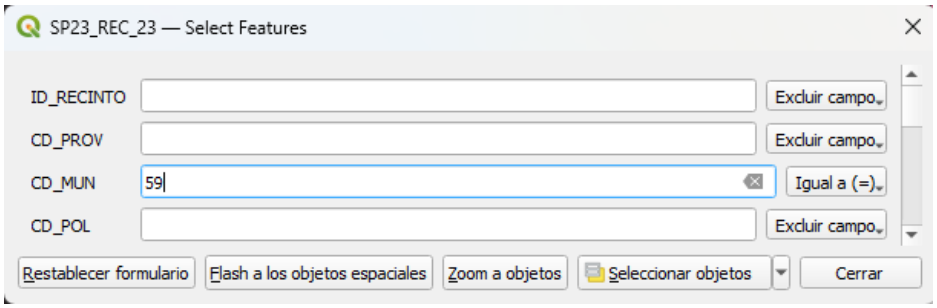


Ilustración 74: Cuadro dialogo selección por valor (autor).

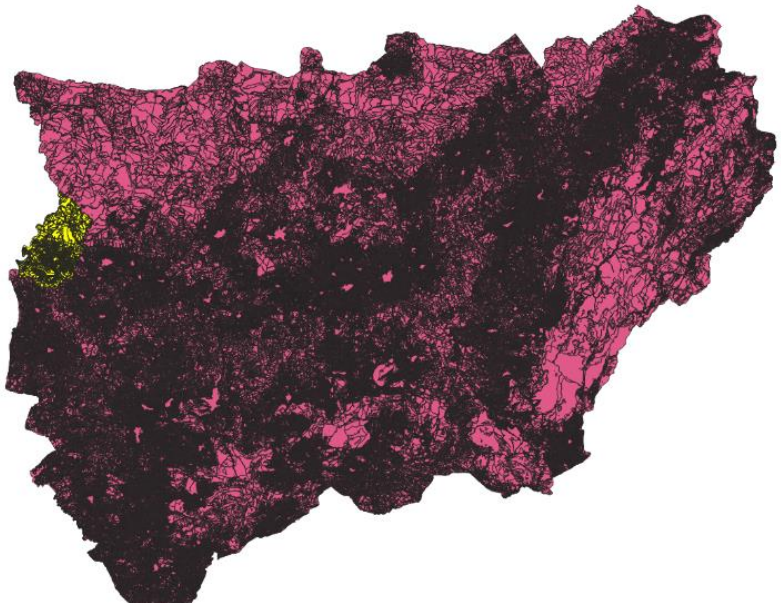


Ilustración 76: Resultado selección por valor (autor).

Una vez se disponía de las parcelas con las que se va a trabajar ya reducidas y tras analizar los datos de estas comparando directamente con los datos del SigPac (Ilustración 77) pudimos observar que estas venían separadas por zonas dentro de una misma parcela y no exclusivamente por parcelas ya que no eran del todo coincidentes lo cual podría suponer un problema según los datos que queramos tratar en cada instante.

Ilustración 77: Representación vectorial de una parcela en el SigPac (arriba) y en nuestro programa QGIS (abajo) (autor).

De cara a los datos zonales temáticos que se presentarán en la aplicación, como son datos del tipo área, pendiente media… Es necesario tener los datos separados por zona (como los tenemos recién descargados) para poder tener todos los datos disponibles y accesibles. Sin embargo, de cara al mapa interactivo lo ideal es requerir de la geometría de la parcela y no las zonas (como se encuentran en el SigPac).

Es por los motivos presentados anteriormente que se decide trabajar con dos capas de información y por ende como se pudo ver en el esquema de la base de datos dividir la información en dos tablas separadas en Parcela y geometria_parcela.

El trabajo con la geometría de la parcela requería de aplicar de nuevo la herramienta Disolver para unificar la geometría por parcelas. Para eso en primer lugar era necesario encontrar el campo que nos permitiría realizar esta operación.

Tras observar los distintos campos del mismo (Ilustración 78) se puede observar que el correspondiente para esta tarea es el campo **CD_PARCELA** en conjunto con el campo **CD_POL** que junta los recintos (CD_RECINTO) de las parcelas para cada parcela concreta que se delimita por un polígono cerrado.

ID_RECINTO	CD_PROV	CD_MUN	CD_POL	CD_PARCELA	CD_RECINTO	CD_USO	NU_AREA	PC_PASTOS	COEF_REG	PDTE_MEDIA	INCIDENCIA

Ilustración 78: Campos de la capa vectorial correspondiente (autor).

Tras tatar de realizar el proceso de disolución de la geometría salto un error (Ilustración 79) del proceso debido a la irregularidad de la geometría con polígonos que se auto cortan o se encuentran unos contenidos dentro de otros. Para solventar este problema se utiliza la herramienta **Corregir geometría** (Ilustración 80) que permite arreglar la geometría de manera previa a la disolución y se encuentra en la caja de herramientas de QGIS.

Objeto (2199) de "sp22_REC_23059" tiene geometría inválida. Por favor arregla la geometría o cambia la opción de procesamiento para "Ignorar objetos de entrada inválidos".
Execution failed after 0.07 segundos

Ilustración 79: Error de la ejecución del Disolver (autor).

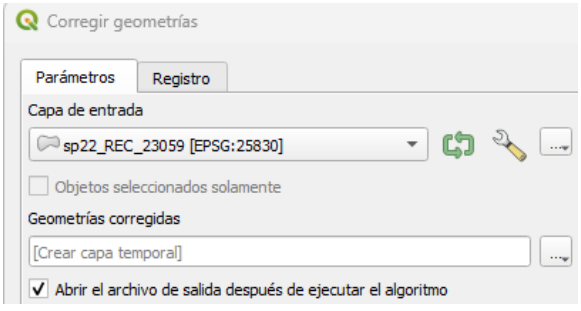


Ilustración 80: Cuadro dialogo para arreglar la geometría (autor).

Una vez arreglada la geometría volvemos a realizar el proceso de disolución y obtendremos la capa de información que queremos para la geometría de las parcelas como podemos observar en la siguiente ilustración.

Ilustración 81: Resultado final del proceso (autor).

5.3.5.3 Fotogramas PNOA

Los fotogramas se obtuvieron con el fin de obtener las mejores ortofotos posibles para el enriquecimiento de la información de la parcela, su obtención se trató en la primera iteración y al trabajar exclusivamente con marmolejo se obtuvieron exclusivamente las 4 capas que conforman la zona del municipio.

Al disponer del trabajo previo realizado sobre las capas de parcelas la extracción de las ortofotos será un proceso sencillo con las herramientas que QGIS nos proporciona, sin embargo, será un proceso de extracción manual. Este consistirá en superponer la capa de información del PNOA con la capa vectorial de parcelas disueltas previamente (Ilustración 82) y emplear la herramienta **Cortar ráster por capa de máscara** ubicada en el menú **Ráster > Extracción.**

Esta herramienta nos permite recortar una capa ráster en función de la geometría de una capa vectorial. Para recortar por una parcela concreta emplearemos la opción que nos ofrece la herramienta de recortar por la selección, esta funcionalidad nos permite seleccionar geometrías concretas de una capa vectorial y recortar solo por estas si se marca la opción (Ilustración 83).

Ilustración 82: Superposición capa PNOA con capa de parcelas (autor).

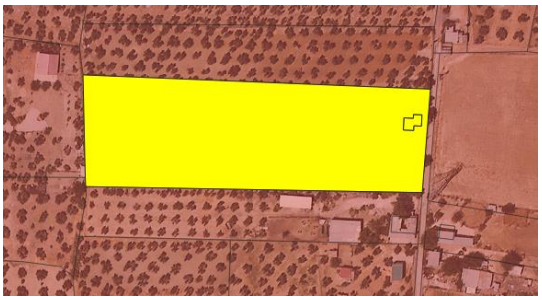


Ilustración 83: Selección de parcela para obtener ortofoto (autor).

Una vez la ejecución ha terminado podremos observar que el recorte no es exacto (Ilustración 85), esto se debe a que el formato resultante es una capa tipo ráster, esta es un tipo de capa espacial que almacena la información en una matriz cuadrada por este motivo la salida será un recorte rectangular, generando zonas negras (sin valor) que trataremos más adelante.

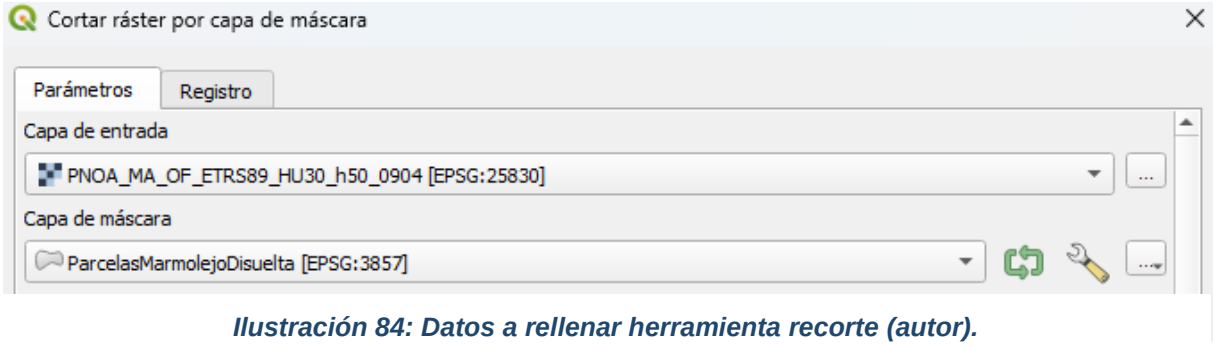


Ilustración 84: Datos a rellenar herramienta recorte (autor).

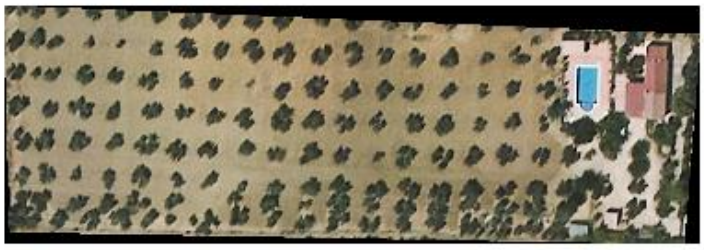


Ilustración 85: Resultado del recorte por capa de mascara (autor).

Al igual que pasaba con operaciones anteriores se genera una capa temporal que guardaremos o exportaremos en formato GeoTIFF (Tagged Image File Format) y en esta ocasión seleccionaremos la opción superior de Imagen renderizada en lugar de datos crudos.

Para poder emplearlo como imagen posteriormente será necesario aplicar un cambio de formato. Esta acción la realizaremos con el programa GIMP de edición de imágenes y que permite trabajar con TIFF. El proceso será muy sencillo y consta de abrir el fichero, añadir una capa alfa, eliminar las zonas negras generadas y exportar en formato PNG.

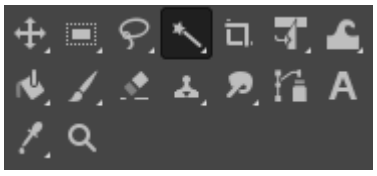


Ilustración 86: Herramienta para eliminar el negro de la imagen (autor).

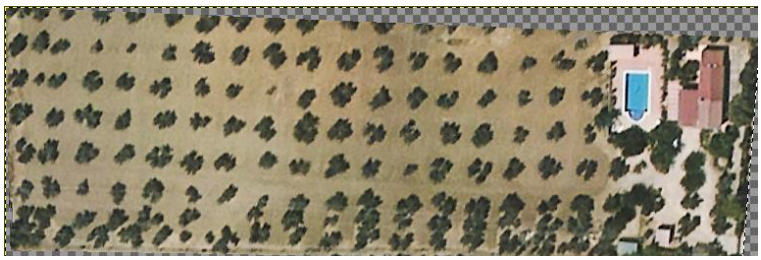


Ilustración 87: Resultado tras eliminar las partes sobrantes (autor).

Este proceso deberá repetirse tantas veces como ortofotos queramos guardar en local de tantas parcelas como sean necesarias. El formato de nombre para las ortofotos será **NombreMunicipio_ID-Recinto.png** facilitando de esta manera que al encontrarnos en una parcela la búsqueda por nombre sea sencilla.

Tras realizar los tres procesos de trabajo con las capas de información tendremos como resultado:

- Tres capas vectoriales de limitaciones territoriales municipales, provinciales y autonómica
- Dos capas vectoriales de limitaciones de parcelas en Marmolejo, una capa de información dividida por zonas de las parcelas y la otra de geometría de las propias parcelas
- Un sistema de obtención de ortofotos mediante recorte por el PNOA.

5.3.6 Resumen trabajo realizado

Tras finalizar esta segunda iteración se realizó una nueva reunión con el fin de concretar como de correcta ha sido la realización de las tareas propuestas, si necesitaran de algún cambio o detallar nuevas futuras tareas.

En esta iteración el entregable a los usuarios era mucho más visual pues ya se había desarrollado una escena de ejemplo en BabylonJS por lo cual podían observar cómo era el funcionamiento de esta y como quedaba visualmente. Sumado a esto se les permitía navegar por la aplicación ya que se había desarrollado como sería toda la comunicación de la interfaz y existía la posibilidad de las primeras interacciones con el mapa desarrollado.

Por último se mostró los niveles y capas de información espacial con los que se iba a trabajar y como se habían extraído, de este misma forma se comentó que este era un proceso manual y no automático algo que se aceptó como correcto para el prototipo inicial.

5.4 Tercera Iteración

En esta iteración se continúa avanzando con la adición de nuevas funcionalidades o mejora de las capacidades ya existentes en la aplicación, se rellena la base de datos creada previamente con datos y se integra en la aplicación dando acceso al sistema a todos los datos de esta, así como a posibles inserciones o borrado de datos. Se desarrolla un sistema de carga de nubes de puntos en la escena para poder trabajar con estos modelos

previamente editados, se genera un Geoserver que permita trabajar con el mapa y los datos vectoriales obtenidos y finalmente se realizarán las primeras pruebas con OpenCV para la detección de entidades.

5.4.1 Inserción de datos

La inserción de los datos es una tarea importante y laboriosa cuando se trabaja con una base de datos de alta dimensionalidad con disparidad de tipos de datos incluyéndose datos de tipo espacial. El proceso para la inserción de datos se ha dividido por ende en datos temáticos o datos espaciales.

Los datos espaciales se han obtenido directamente del proyecto en QGIS donde se trabajó para su procesamiento en la iteración anterior, la inserción se ha realizado con un plugin del propio software QGIS conocido como **MySQL Importer** y que permite realizar una conexión con una base de datos MySQL y subir datos espaciales a la misma. El proceso es el siguiente:

1. Se instala el plugin en la sección **Complementos > Administrar e instalar complementos > MySQL Importer**
2. Se accede al mismo a través del menú **Base de datos > MySQL Importer**
3. Se abrirá una ventana de dialogo como la de la Ilustración 88 donde deberemos introducir nuestros datos del hospedaje de la base de datos, en nuestro caso estos serán:
    a. Server IP: localhost
    b. Port: El puerto definido en XAMPP para la base de datos, se podrá consultar en el propio panel de XAMPP
    c. Username: root
    d. Password: En blanco
    e. File: El fichero SHP con los datos de la capa a exportar a la base de datos

f. CRS: El sistema de referencia empleado en nuestro sistema, para nuestro trabajo este será el **EPSG:3857–WGS 84 / Pseudo-Mercator**

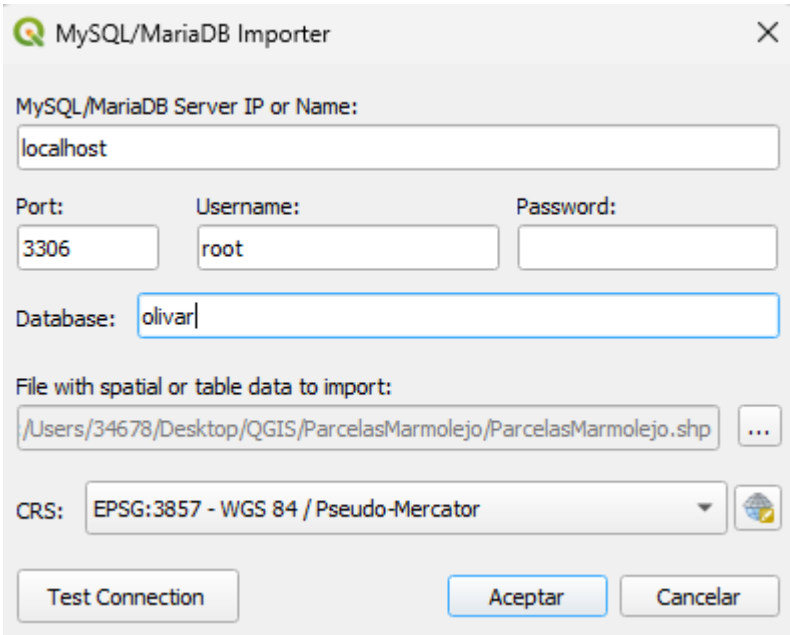


Ilustración 88: Ventana dialogo plugin SQL Importer de QGIS (autor).

Una vez con todos estos datos introducidos pulsaremos en aceptar y los datos se subirán a la base de datos, este proceso se realiza tantas veces como datos queramos disponer en la base de datos, en nuestro caso se realiza en tres ocasiones.

- Tabla de municipios de Andalucía.
- Tabla de parcelas con sus datos divididos en formato SigPac por zonas.
- Tabla de geometría de parcelas disueltas por recintos.

La primera inserción genera en nuestra base de datos otras dos nuevas tablas que se denominarán Spatial_ref_sys y geometry_columns cuyo funcionamiento y utilidad se explicó en la Sección 3.3.4.

Una vez insertados los datos espaciales es necesario realizar algunas operaciones para reordenar la base de datos como renombramiento de tablas, creación de claves foráneas o eliminación de atributos no necesarios / repetidos con el fin de tener todo el sistema bien conectado y estructurado.

Los datos temáticos por otra parte se han realizado con una inserción manual para las entidades necesarias para un correcto funcionamiento del sistema como puede ser un

usuario base como administrador o las primeras entidades de parcelas con las que trabajamos, esto se debe a que estos datos de carácter algo más personal no se encuentran publicados en la web o de fácil acceso.

En futuras iteraciones se desarrollarán más en profundidad interfaces para la administración de este tipo de datos donde se permitirá la edición, adición y borrado de estos con los permisos adecuados.

5.4.2 Desarrollo de un sistema para la carga de nubes de puntos

Una vez disponemos de los modelos de nubes de puntos preprocesados de la primera iteración el trabajo a realizar consistirá en su inclusión en la página relativa a la escena BabylonJS ya desarrollada, para esto es necesario encontrar el formato de nube de puntos con el que trabajar y la forma de importarlo.

En primer lugar, se buscó la forma de poder integrar estos modelos de forma directa en la escena mediante el uso los formatos más comunes como son PLY o SBF, sin embargo, BabylonJS solo trabaja con importación de ficheros del tipo GLB, glTF u OBJ.

Cabe destacar que la exportación a este tipo de ficheros es posible, sin embargo, en el transcurso de esta o realizando un cambio de formato entre distintos ficheros se produce la perdida de color (Ilustración 89) de la propia nube aspecto que debido a la importancia de la visualización no es aceptable para nuestros intereses.

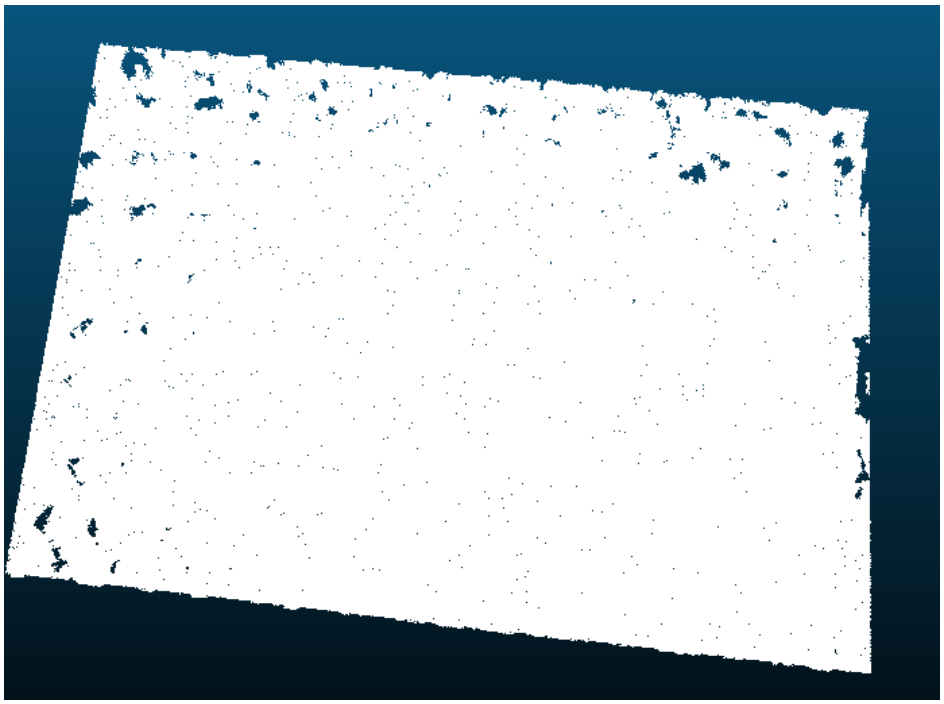


Ilustración 89: Nube de puntos en formato OBJ Cloud Compare (autor).

Finalmente se optó por utilizar una herramienta integrada en el propio BabylonJS que es el PCS (point cloud system) que nos permite crear sistemas de nubes de puntos, para su funcionamiento necesitamos poder acceder punto por punto exclusivamente. Esto se puede conseguir generando un fichero TXT desde la aplicación de Cloud Compare tal y como se puede observar en la siguiente imagen.

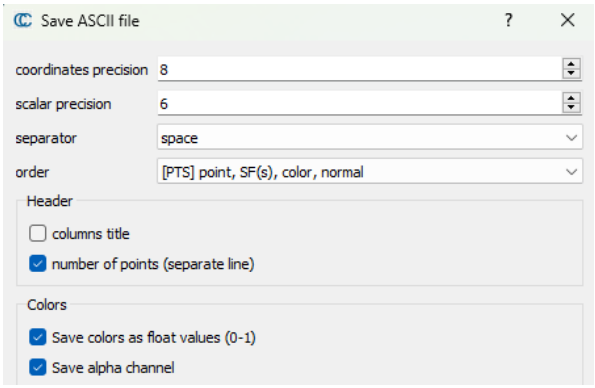


Ilustración 90: Cuadro de dialogo para guardado de nube en formato TXT (autor).

Una vez disponemos del fichero necesario lo nombraremos con el siguiente formato *Municipio_ID-Parcela.txt,* de esta forma tendremos un fácil acceso a este en cuanto conozcamos los datos de la parcela en cuestión. El último paso que quedaría sería desarrollar el PCS dentro de la escena de BabylonJS la estructura del código para esto es la siguiente:

```
Procedimiento creacionParcela(scene,camera,parcela):
    datos <- fichero(parcela)
    nube <- array()
    Para i <- i Hasta tam(datos) Con Paso 1 Hacer
        Si datos[i] > puntoMax Entonces
            puntoMax <- datos[i]
        Fin Si
        Si datos[i] < puntoMin Entonces
            puntosMin <- datos[i]
        Fin si
        nube[i] <- datos[i]
    Fin Para
    Si tam(nube) > 0
        puntoMedio <- (puntoMax + puntoMin) / 2
        pcs <- BABYLON.PointCloudSystem(scene)
        Para i <- 1 Hasta tam(nube) Con Paso 1 Hacer
            p.position <- nube[i].position - puntoMedio
            p.color <- nube[i].color
            pcs.addPoint(1, p)
        Fin para
        pcs.buildMeshAsync()
    Fin si
Fin Procedimiento
```

Ilustración 91: Pseudocódigo carga nube de puntos (autor).

En primer lugar se abrirá el fichero con JavaScript para posteriormente recorrerlo línea por línea insertando los datos en un vector a la vez que se van comparando las coordenadas con el fin de encontrar los puntos extremos de cada eje.

Una vez tenemos los puntos extremos de cada eje podemos delimitar el punto mínimo y máximo de la Bounding Box que envuelve la nube de puntos y con esto delimitar el centro de la misma. Este centro responde a la siguiente formulación.

$$Punto\ medio = \frac{Punto\ máximo + Punto\ mínimo}{2}$$

Obtenido el centro es posible llevarlo al centro del sistema de coordenadas de mundo con una traslación y realizar la misma modificación para el resto de puntos uniformemente consiguiendo de esta forma ver la nube centrada, de esta manera se solventa finalmente la problemática planteada en la segunda iteración con la posición de la nube de puntos.

Con la posición y color de la nube de puntos en un vector es el momento de la creación del PCS, para esto se deberá recorrer el vector y llamar en cada iteración a la función **pcs.addPoints(1, myfunc)** con los parámetros:

- 1: Por el número de puntos a insertar en esta iteración
- myfunc: Función personalizada a realizar en cada iteración, en nuestro proyecto esta función establece la posición y color de cada punto de la nube con los datos del vector.

Finalmente se cierra el PCS y se construye una Mesh. Este ciclo es capaz de recoger cualquier nube previamente procesada y mostrarla en el centro del sistema como se puede ver en la Ilustración 92.

Ilustración 92: Ejemplo nube recién cargada en el sistema (autor).

5.4.3 Gestión capas vectoriales con Geoserver

Un geoserver es un software web de gran utilidad que permite acceder a datos espaciales desde cualquier lugar, así como mostrarlos en distintos formatos, en este proyecto esta herramienta se ha empleado para acceder a datos que no merece la pena tener almacenados en local, así como para convertir los datos en distintos formatos fácilmente.

El proceso seguido para trabajar con un Geoserver y trabajar con las capas vectoriales comienza por la instalación del software que se encuentran en la página oficial[24]. En nuestro caso trabajaremos con el **Windows installer** de la versión estable del momento**.**

Una vez disponemos del .exe en nuestro equipo lo ejecutaremos y dejaremos todos los valores por defecto, un aspecto a resaltar es que es necesario disponer de una versión de Java para poder trabajar con el software.

Una vez este se encuentra instalado se podrá acceder a este desde el navegador con una URL tal que *http://localhost:8080/geoserver/*. Cuando nos encontremos dentro de esta dirección se nos requerirán las credenciales que determinamos al instalar el servicio y finalmente tras esto podremos acceder a trabajar con el software.

Cuando entramos podemos observar que por defecto Geoserver presenta unas capas de ejemplo para que puedas ver como trabajar con los datos, las formas de observarlos mediante mapas como OpenLayers, GML… Así como las formas de exportarlos en otros formatos mediante WMS o WFS.

En nuestro caso emplearemos Geoserver para dos funcionalidades que son:

- Disponer de las capas en la web pudiendo obtener acceso a estas una vez el geoserver se encuentra instalado en el mismo dispositivo que el servidor.
- Trabajar con los formatos de las capas para cambios de formatos o de proyección de las mismas.

El segundo caso será el más utilizado para obtener ficheros en el formato deseado en local. Previo a trabajar con las capas es necesario disponer de estas en el servicio, para esto volveremos a trabajar con QGIS que mediante el uso del plugin **GeoCat Bridge** nos permite subir capas del proyecto al servidor de la web.

[24] Página Oficial Geoserver: https://geoserver.org/

Una vez instalado el plugin el proceso a seguir será el siguiente:

1. Apertura del plugin y conexión con el servidor correspondiente (Ilustración 93).

2. Selección capas a subir (Ilustración 94).

3. Confirmación del proceso

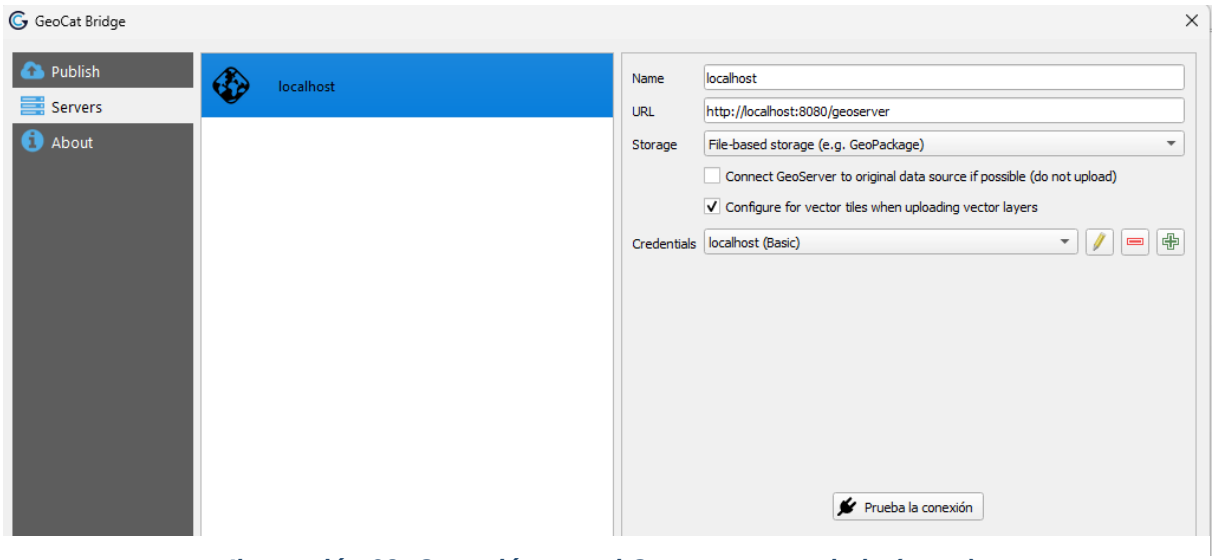


Ilustración 93: Conexión con el Geoserver asociado (autor).

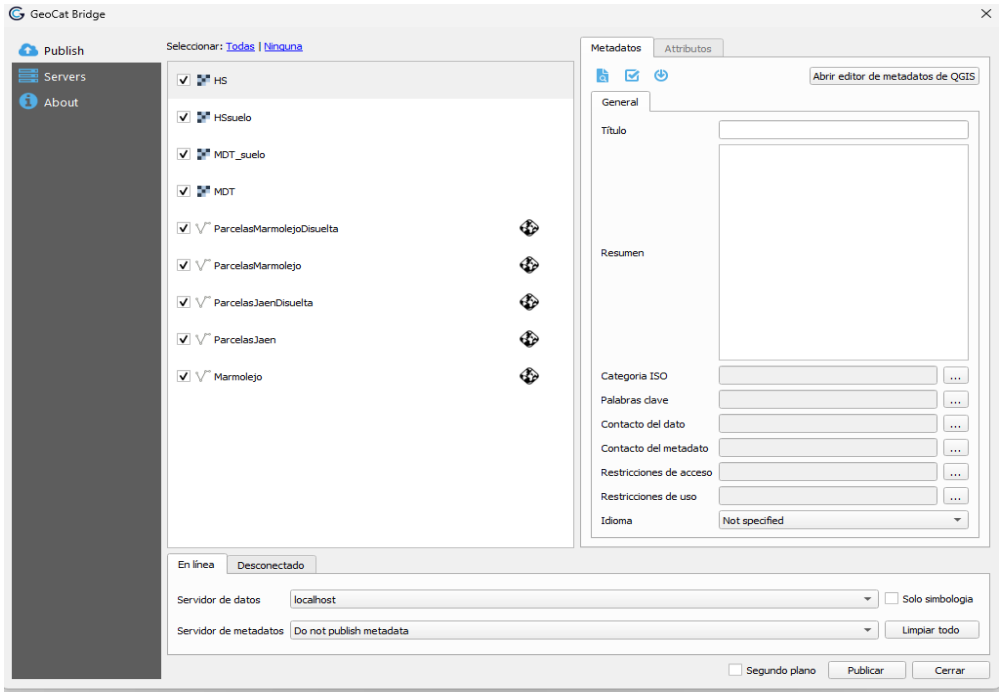


Ilustración 94: Selección capas subir (autor).

Una vez terminado el proceso y cómo podemos ver en la Ilustración 96 las capas se encuentran ubicadas en el servidor de mapas y podremos comenzar a trabajar con ellas. El principal uso que le daremos como hemos comentado previamente será el de exportar los datos en el formato deseado, para esto seleccionaremos el formato de visión de la capa deseada como **JSON** y se nos abrirá una ventana como se puede ver en la Ilustración 95 donde se presentan los datos de la capa en formato de texto plano estructurado.

Tipo	Título	Nombre	Formatos habituales	Todos los formatos
	Andalucia	ParcelasJaen:andalucia	OpenLayers GML KML	Seleccionar una
	Marmolejo	ParcelasJaen:marmolejo	OpenLayers GML KML	Seleccionar una
	Municipios	ParcelasJaen:municipios	OpenLayers GML KML	Seleccionar una
	ParcelasJaen	ParcelasJaen:parcelasjaen	OpenLayers GML KML	Seleccionar una
	ParcelasJaenDisuelta	ParcelasJaen:parcelasjaendisuelta	OpenLayers GML KML	Seleccionar una
	ParcelasMarmolejo	ParcelasJaen:parcelasmarmolejo	OpenLayers GML KML	Seleccionar una
	ParcelasMarmolejoDisuelta	ParcelasJaen:parcelasmarmolejodisuelta	OpenLayers GML KML	Seleccionar una
	Provincias	ParcelasJaen:provincias	OpenLayers GML KML	Seleccionar una

Ilustración 96: Visualizador de capas Geoserver (autor).

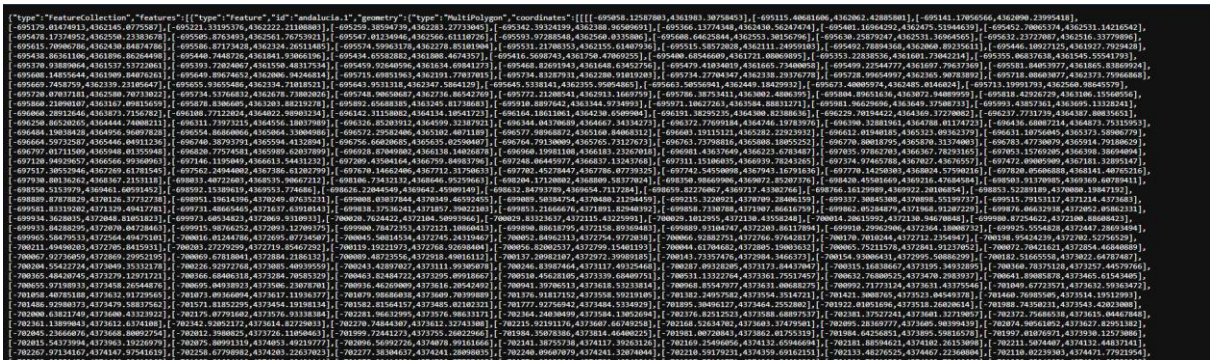


Ilustración 95: Exposición datos en formato GeoJSON Geoserver (autor).

Una vez cargado todo el texto, ya que en ocasiones por su volumen puede llegar a tardar, procedemos a descargarlo y almacenarlo en local con el nombre de la capa que represente, pudiendo de esta manera trabajar con el mapa interactivo y las capas necesarias.

En caso de querer trabajar con la capa desde el Geoserver que como se comentó en apartados previos es una de las posibles opciones, será suficiente con disponer del enlace WMS o WFS que de acceso a la capa en cuestión, siendo estos enlaces del tipo:

http://localhost:8080/geoserver/earth/wfs?service=WFS&version=1.0.0&request=GetFeature&typename=cities&outputFormat=application/json

Como se puede observar es necesario indicar el servicio requerido, el formato de muestra de los datos y la capa con la que trabajar. En apartados posteriores cuando se requiera de más funcionalidades se empleará este recurso.

5.4.4 Integración base datos en la aplicación

Una correcta comunicación entre la base de datos y la aplicación es un aspecto clave para el correcto funcionamiento del sistema debido a la gran cantidad de información almacenada en la base de datos y que es necesaria para el funcionamiento de nuestra aplicación.

Para realizar esta conexión se ha empleado el lenguaje PHP y varios scripts realizados en función de las necesidades, así como la tecnología AJAX de JavaScript que nos permite actualizar la página en tiempo real en función de acciones que el usuario realice.

En primer lugar, se realizó un script de conexión con la base de datos pudiendo de esta manera mediante el uso de unas credenciales acceder siempre que se necesite a la base de datos.

Este script se llama desde el resto de los scripts php cuando se quiera realizar una consulta. Para esto bastara con ejecutar el comando include de php y el nombre del fichero.

Dentro de los scripts se realizarán las consultas en lenguaje SQL y se lanzarán las ejecuciones con funciones propias internas de PHP. Las funcionalidades principales utilizadas son:

- $conexión->query(Consulta) es una función que realiza una **Consulta** a la base de datos y almacena la información en una variable, si la consulta es fallida devuelve false, en caso contrario devolverá un objeto del tipo mysqli_result del que podremos extraer la información con la función que se presenta a continuación.

- $consulta->fetch_array(MYSQLI_BOTH) es una función que almacena el resultado de la consulta realizada en un array asociativo donde las claves se corresponden con el nombre de cada atributo.

- Mysqli_num_rows($consulta) es una función que devuelve el número de filas devueltas por la base de datos para una consulta dada

- $_REQUEST[elemento] es un elemento de PHP, se corresponde con un array asociativo que almacena el contenido de los GET, POST y COOKIE, gracias a esta variable podemos pasarle datos al script y solicitarlos bajo un sobrenombre **elemento** que elijamos.

Una vez conocemos las posibles funciones a utilizar dispondremos de distintos tipos de operaciones que podremos realizar, como es el caso de inserciones, actualizaciones o consultas de datos.

Dentro de estas funciones además de trabajar con las bases de datos se podrán crear en texto plano elementos HTML que luego insertar en la página mediante JavaScript como puede ser el caso de tablas rellenas con los datos de las consultas o dotar de nombre o funcionalidades a secciones o inputs del sistema.

Finalmente, además de tener una estructura predefinida con AJAX también es posible necesitar de consultas o datos de la base de datos en tiempos de carga de la página o situaciones similares. Estos casos en nuestro sistema se ven ejemplificados en el inicio de sesión y cierre de la sesión donde es necesario verificar las credenciales o acabar con la sesión creada. Estos dos casos se presentarán de manera individual y responderán ante el uso de un elemento del tipo form de HTML encargado de llamar al código en PHP.

El inicio de sesión (Ilustración 97) es un proceso fundamental en una aplicación web para la cual requieres de un acceso restringido. Cada página de la interfaz dispondrá de una sección previa a la cabecera de código PHP. Se distinguen dos formatos para este código que son:

- Index: Esta página inicial o de inicio de sesión realiza una comprobación previa de si el usuario se encuentra loggeado previamente es decir si existe una variable SESSION con un valor distinto de nulo, si esto es verdadero se le redirigirá a la página de bienvenida.

- Resto: Realizaran una comprobación similar, pero con el objetivo inverso, se pretende que sea imposible acceder a estas sin estar con una sesión iniciada, para esto se comprueba si la variable SESSION tiene valor, si es así se permitirá cargar la página, en caso contrario se le redirigirá a la página Index.

Una vez realizada estas comprobaciones se carga la página por completo. Si nos encontramos realizando el proceso de iniciar sesión el código que se esconde detrás de la misma realiza un proceso de validación de los datos introducidos y en caso de coincidir con la base de datos, esta crea una sesión y nos redirigirá a la página de bienvenida.

```
Procedimiento inicio_sesion():
    usuario <- $_POST['usuario']
    contraseña <- $_POST['contraseña']
    session_start()
    $_SESSION['tiempo'] <- time()
    $_SESSION['usuario'] <- usuario
    $_SESSION['contraseña'] <- contraseña
    consulta <- consultaSelect(usuario)
    Si consulta['contraseña'] = hash(contraseña) Entonces
        CargarPagina(Home)
    Sino
        $_SESSION <- NULL
        CargarPagina(Index)
    Fin Si
Fin Procedimiento
```

Ilustración 97: Pseudocódigo inicio sesión (autor).

5.4.5 Primeros trabajos con OpenCV

La detección de entidades fue una funcionalidad que se planteó en este proyecto con la intención de extraer información desconocida y costosa de obtener y manipular desde datos públicos como son las ortofotos extraídas del PNOA.

El objetivo de esta tarea es la de comenzar a trabajar con OpenCV para poder desarrollar un Script básico para detectar posibles contornos que se puedan relacionar con la figura de olivos en una parcela y ubicarlos en la escena.

Con el objetivo ya marcado el trabajo consistió en la creación de un proyecto en C++ donde ir realizando pruebas con las distintas herramientas y funcionalidades de OpenCV para extraer un resultado mínimo aceptable de segmentación de olivos para varías ortofotos.

Las funcionalidades empleadas en la elaboración de este primer script fueron:

- Blur: Funcionalidad de suavizado de imágenes empleado para la eliminación de ruido como son contenidos de alta frecuencia (bordes).
- Canny: Funcionalidad para detección de bordes de las entidades de una imagen bajo un umbral determinado

- Contours: OpenCV dispone de varias utilidades centradas en la delimitación de contornos basándose en los bordes encontrados mediante Canny permitiendo devolver estos y trabajar más adelante con ellos.

Todas estas se emplearon en un código que tenía la siguiente estructura:

```
Procedimiento Deteccion():
    material <- leerImagen()
    gris <- cv.Material()
    cambioColor(material, gris, cv.COLOR_RGBA2GRAY)
    blur(gris)
    canny <- cv.Material()
    umbral <- 150
    Canny(gris, canny, umbral)
    filtrado <- cv.MatVector()
    findContours(canny, filtrado)
    minArea <- 20
    filtradoContornos(filtrado, minArea)
    Para i <- 1 Hasta tam(filtrado) Con Paso 1 Hacer
        Para x <- 1 Hasta tam(filtrado[i]) Con Paso 1 Hacer
            punto_1 <- filtrado[i]
            punto_2 <- filtrado[i + 1]
            dibujoLinea(punto_1, punto_2)
        Fin Para
    Fin Para
Fin Procedimiento
```

Ilustración 98: Pseudocódigo primer script detección (autor).

Una vez se disponía del código final se realizaron pruebas en distintas ortofotos teniendo los siguientes resultados:

- **Parcela 1**

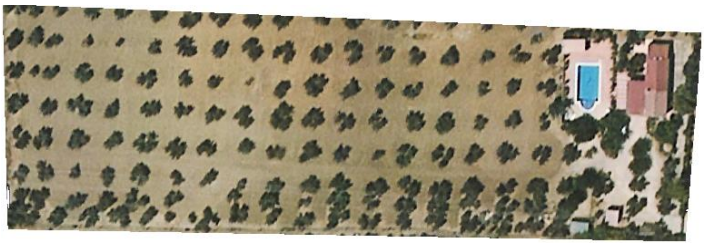


Ilustración 100: Parcela pruebas 1 (autor).

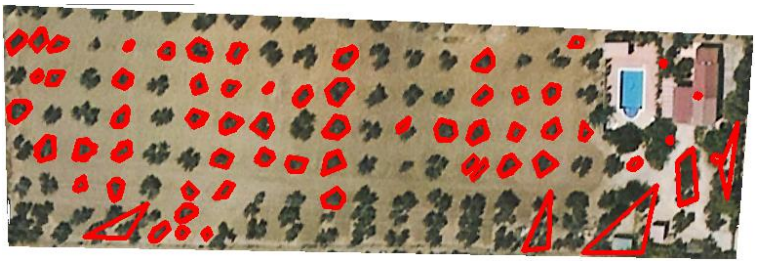


Ilustración 99: Resultados pruebas 1 (autor).

- **Parcela 2**

Ilustración 101: Parcela prueba 2 (autor).

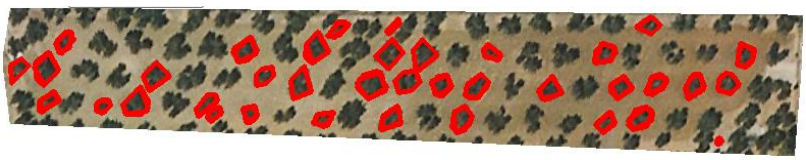


Ilustración 102: Resultados pruebas 2 (autor).

- **Parcela 3**

Ilustración 104: Parcela prueba 3 (autor).

Ilustración 103: Resultados prueba 3 (autor).

Parcela	Positivos	Falsos positivos
1	53/130	6
2	35/105	1
3	29/54	9

Tabla 24: Tabla resultados primer Script OpenCV.

Como podemos observar los resultados no son excesivamente satisfactorios, pero ya nos permiten levemente situar una cantidad considerable de olivos. Los principales problemas que podemos observar se encuentran en el posicionamiento de objetos externos al olivar como edificaciones ya que los mejores resultados se obtuvieron en la segunda parcela donde se exclusivamente hay terreno de olivar. Esto se tratará de mejorar en iteraciones futuras como podremos ver en la Sección 5.6.3.

5.4.6 Resumen trabajo realizado

Una vez finalizada la tercera iteración se procede con una nueva reunión con los usuarios clave del proyecto, en esta se les mostro el ejecutable actual de la aplicación. En este sumado a lo que ya vieron en la iteración pasada podían por primera vez acceder y moverse por la nube de puntos así como visualizar por la aplicación datos concretos al vincular la base de datos y acceder a un mayor número de capas desde el mapa.

Por último se les mostró el script de detección de olivos en ortofotos y los resultados que acarreaban los mismos. Con respecto a esto se comentó que sería un apartado que se debería tratar de mejorar y por ende en iteraciones futuras se añadiría esto como una tarea. Sin embargo los pasos a seguir eran los correcto y las tareas entregadas correspondían con lo solicitado.

5.5 Cuarta Iteración

Una vez llegada esta iteración casi todas las funcionalidades básicas definidas como objetivos del proyecto se encuentran presenten en nuestra aplicación y el trabajo restante se centra en terminar de desarrollarlas o bien mejorarlas de cara al usuario o con el fin de incrementar sus utilidades en la aplicación. En primer lugar, se realiza una integración de scripts de C++ con la aplicación web y se desarrollarán algoritmos para la edición y trabajo con la nube de puntos a la vez que se trabaja con la inserción de texturas con altura para enriquecer la información disponible en la escena.

De manera externa a la escena se ampliarán las funcionalidades y la facilidad en el manejo del servidor de mapas disponible. Finalmente se realizarán las primeras pruebas de rendimiento y usabilidad de la aplicación web en dispositivos ajenos a un ordenador como puede ser móvil y Tablet.

5.5.1 Integración OpenCV en la aplicación

Una vez terminada de desarrollar una primera versión de un script para la detección de entidades con OpenCV y probado que realiza una detección aceptable para los recursos empleados, es el momento de integrarlo en la aplicación y es esta se enfocó con dos planteamientos posibles.

El primero consistía en integrar un script de C++ en la aplicación web de forma que pueda ejecutarse cuando se desee y no tener que disponer de los resultados precargados manualmente. El segundo consistía en emplear la versión OpenCV de JavaScript y transcribir el código completo a este lenguaje.

La segunda opción fue la primera que se probó y se observó que podía llegar a funcionar pues todas las funcionalidades de un lenguaje se encontraban en el otro y por ende era posible realizar una adaptación fiel a este, sin embargo esta opción se descartó debido a que el rendimiento al emplearlo era menor y que es un lenguaje que presenta una mayor complejidad a la hora de adaptar código extenso y orientado a objetos o con una gran cantidad de clases detrás, por ende se optó por integrar la ejecución de un script de C++.

Para conseguir integrar la ejecución de un script de C++ a la aplicación web había que plantearse que era necesario, en primer lugar, el script devolvía una imagen donde señalaba cada uno de los olivos detectados por ende era necesario almacenar la imagen, pero además de esto era de utilidad conocer la posición de cada uno de los olivos por lo cual era necesario recoger de alguna manera este valor.

Este proceso requería de una adaptación del código en varios aspectos, primero era necesario cambiar el sistema de rutas de los directorios que empleaba el mismo de manera que se adaptara al sistema de directorios de la aplicación web. Además, era necesario encontrar una forma de recoger los datos, sin embargo, para esto primero era necesario conocer como ejecutar un script desde el navegador y la aplicación para poder conocer cómo se devuelven estos.

Finalmente se optó por realizar una ejecución mediante un script de PHP que empleara la función **system** que es un comando de PHP que nos permite ejecutar un programa externo y mostrar su salida. De esta forma tendríamos que emplear la siguiente orden:

System("scriptDeteccion.exe Parcela")

Donde Parcela es el id de la parcela a utilizar para poder acceder a la ortofoto de la misma, esta orden ejecuta el script y devolverá los valores como si fuese el terminal, esto implica que tendremos acceso a aquellos que indiquemos que imprima por consola, por ende, para conseguir los datos se deberá alterar el script de C++ para que imprima todos los valores de posición de olivos por consola, recogerlos con PHP y procesarlos y almacenarlos individualmente en una estructura de datos con JavaScript. Además de estos cambios también se deberán sustituir las rutas relativas a la obtención de recursos y adaptarlas a las rutas del sistema de directorios propio de la aplicación.

Una vez tenemos todos los cambios realizados para amoldar el script a nuestra aplicación web es necesario almacenarlo correctamente en nuestro sistema de directorios, para esto se ha generado un nuevo directorio en la carpeta Recursos denominado **Scripts** donde se ubicarán todos los scripts que pudieran ser necesarios en el futuro de la aplicación.

Para que un Script de C++ funcione correctamente debe disponer de todos los archivos de configuración necesarios como pueden ser del formato .dll o el propio .exe de ubicación. Estos ficheros se encuentran en la carpeta Debug o Release que te genera Visual Studio en el proyecto (Ilustración 105) y se deberán introducir también dentro de la carpeta de Scripts en la funcionalidad a la que se asocie.

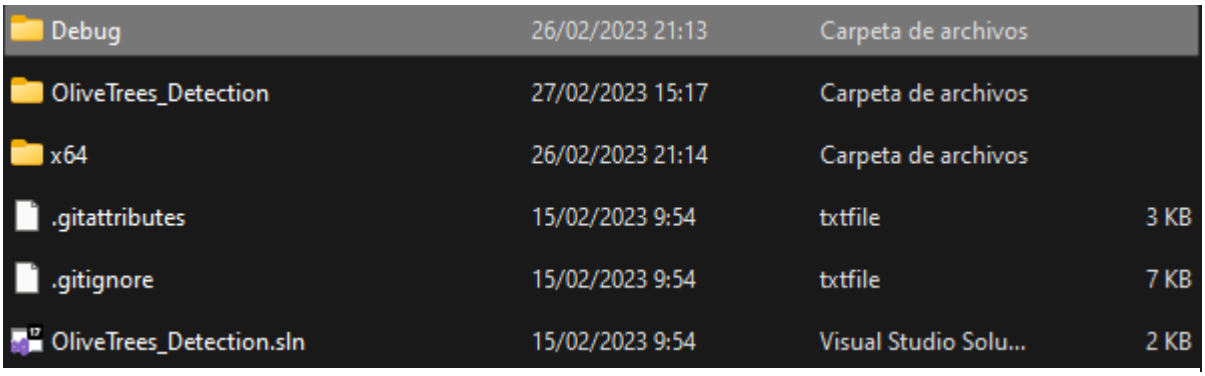


Ilustración 105: Directorio principal proyecto VS con la carpeta Debug marcada (autor).

5.5.2 Desarrollo algoritmo recorte de nubes de puntos

Con la nube de puntos una vez cargada existen múltiples funcionalidades de la escena de BabylonJS que nos permitirá obtener información y movernos por el modelo sin restricciones espaciales más allá de las que establezca el propio desarrollador. Sin embargo, existen aún funcionalidades que se pueden desarrollar y que no son intrínsecas a la documentación de BabylonJS por lo cual es necesario recrearlas por nosotros mismos.

En esta tarea se plantea la necesidad de poder recortar una nube de puntos en función de una selección poligonal que establezca el usuario pudiendo de esta manera acotar la parcela y observar o trabajar exclusivamente con una zona de la parcela.

Para conseguir esto se divide el trabajo en dos partes, la recreación de una zona poligonal interactiva determinada por el usuario y el recorte de la nube con el diseño de un algoritmo que determine si un punto se encuentra dentro o fuera del polígono.

Antes de comenzar a explicar el proceso es necesario recalcar que el recorte se producirá en dos dimensiones por ende lo primero que se realiza al iniciar esta funcionalidad es colocar la vista cenital de la nube y trabajar con el recorte de la misma en función de las coordenadas X y Z obviando cortes por altura.

5.5.2.1 Punto en polígono

El algoritmo del punto en polígono tiene múltiples algoritmos que permiten obtener un resultado coherente y funcional, sin embargo, en ocasiones estos algoritmos son muy dependientes de la forma del polígono pues algunos solo funcionan con polígono convexos.

Para este proyecto se ha optado por la implementación del algoritmo radial [20] cuyo funcionamiento para determinar si un punto se encuentra dentro de un polígono es

determinar si la suma de los ángulos que forman este punto con los vértices adyacentes dos a dos es igual a 2π.

Este proceso se realiza con todos los puntos de la nube de puntos para determinar cuáles de ellos pertenecen o no al recorte, los que pertenezcan se añadirán a un nuevo array de puntos que se representa finalmente al aplicar le recorte.

El criterio de este algoritmo se basa en el concepto de la orientación de los puntos, en el código correspondiente trabajamos con tres puntos que es suficiente para conformar un triángulo. La orientación de los puntos en el triángulo puede seguir un sentido horario u antihorario, en el caso del sentido horario nos permite considerar que el área del triángulo o bien el ángulo que forman los segmentos con los que trabajamos es negativo, por el contrario, el sentido antihorario nos devuelve un valor positivo. Tras recoger todos los valores de la comparación con todos los vértices podemos extraer una conclusión sobre la posición del punto en el polígono.

El algoritmo radial implementado recorrerá todos los vértices dos a dos hasta volver al inicio y presenta tres casuísticas como son que está en una arista que devolverá 0, que este fuera y devolverá -1 o que este dentro y devuelva 1.

El pseudocódigo para la aplicación de este algoritmo es el siguiente:

```
Procedimiento Punto_Poligono(Punto p, Vertices v):
    suma <- 0
    Para cada Segmento formado por v[i],v[i+1] Hacer
        a <- AnguloEntre(p,segmento)
        Si Absoluto(a) = π Entonces
            Devolver 0
        Fin Si
        suma <- suma + a
    Fin Para
    Si suma = 0 Entonces
        Devolver -1
    Fin Si
    Devolver 1
Fin Procedimiento
```

Ilustración 106: Pseudocódigo algoritmo radial (autor).

```
Procedimiento AnguloEntre(Punto p, Segmento s):
    vector_1 <- Vector(p,s[0])
    vector_2 <- Vector(p,s[1])
    prod_vectorial <- Dot(vector_1, vector_2)
    seno <- Absoluto(prod_vectorial) / (Modulo(vector_1) * Modulo(vector_2))
    Si prod_vectorial > 0  Entonces
        Si seno > 0 Entonces
            Devolver Arcoseno(vector_1,vector_2)
        Sino Entonces
            Devolver -Arcoseno(vector_1,vector_2)
        Fin Si
    Sino Entonces
        Si seno > 0 Entonces
            Devolver -Arcoseno(vector_1,vector_2)
        Sino Entonces
            Devolver Arcoseno(vector_1, vector_2)
        Fin Si
    Fin Si
Fin Procedimiento
```

Ilustración 107: Pseudocódigo extracción ángulo entre dos segmentos (autor).

5.5.2.2 Recreación polígonos en BabylonJS

Una vez disponemos del algoritmo de recorte es necesario proceder con pruebas de su funcionamiento, así como otorgar la posibilidad al usuario de determinar sus propios polígonos manualmente.

Para este apartado es necesario la recreación de un polígono en la escena de forma que el usuario pueda ver la porción de nube de puntos que está encerrando para el recorte (Ilustración 108). Para conseguir esto se recrea el proceso mediante la creación dinámica de líneas de modo que con cada clic se establecerá un nuevo vértice y mientras el usuario se mueve por el sistema se generarán y borrarán líneas para simular que esta se mueve con el puntero hasta que se realice una nueva pulsación. Para terminar el polígono habrá que realizar doble clic y este se cerrará solo.

Ilustración 108: Escena con polígono de recorte asociado (autor).

Ilustración 109: Recorte de la parcela (autor).

5.5.2.3 Aplicar el recorte

Una vez se dispone de todo el proceso terminado con el polígono mostrado por pantalla el usuario podrá determinar cuando quiere que se realice el recorte. Una vez activada esta opción el proceso seguido por la aplicación será el siguiente:

1. Eliminación visual del polígono.

2. Movimiento de las coordenadas del polígono a la zona de trabajo:

   En una primera instancia las coordenadas del polígono resultante se encuentran en la altura donde la cámara que es elevada respecto a la nube en 200 para el valor de Y, por este motivo es necesario bajar estas coordenadas y ajustarlas para su presentación en la nube, para esto es necesario multiplicar las coordenadas en X y Z por el valor de la altura en nuestro caso 200.

3. Recorrido de los puntos de la nube aplicando la función Punto en polígono, si devuelven un valor verdadero se introducen en la nueva nube.

4. Representación de los puntos relativos al nuevo vector de puntos (Ilustración 109).

5.5.3 Extensión de las funcionalidades del mapa interactivo

El mapa interactivo se desarrolló en iteraciones anteriores, con este el usuario era capaz de observar un mapa topográfico y satelital, así como distintas capas procesadas en Geoserver y descargadas en formato JSON. El mapa se encontraba dotado de un controlador de capas y la opción de seleccionar las parcelas para acceder a las páginas principales de estas.

En esta tarea los desafíos que se plantean son los de añadir nuevos controles con el fin de mejorar la experiencia del usuario en el control del mapa. Algunos de los controles añadidos son de fácil desarrollo gracias a las facilidades presentadas por la documentación de OpenLayers y que algunos ya se encuentran programados por lo que solo es necesario añadirlos al código. Tras observar ejemplos y debatir las distintas posibilidades los controles que se decidieron añadir son:

- Un control para pantalla completa

- Dos controles de zoom

- Un control de selección para determinar cuándo se puede clicar una parcela
- Un control de posición del ratón y de escala
- Una búsqueda de municipios y parcelas

En primer lugar, se tratarán aquellos controles que son de fácil adición ya que siguen un esquema general común, estos son el control de pantalla completa, escala, los de zoom y la posición del ratón. La creación de estos controles requiere siempre de una única llamada a una función nativa de OpenLayers que es ***ol.control.NombreControl()*** para la cual dentro del paréntesis se podrán ajustar parámetros de cada tipo de control (Ilustración 110).

Para el control de posición de ratón se deberá definir la proyección y el formato de impresión de la coordenada.

Para la escala el tipo de representación en nuestro caso en formato de barras. El elemento de pantalla completa no requiere de cambios ya que trabaja bien con los valores por defecto.

Por último, a la slider de zoom se le han realizado cambios en el estilo al hacerlo levemente más grande y añadirle el nivel de zoom en formato numérico impreso en este.

Finalmente, los controles restantes que son la búsqueda y la selección si tienen más profundidad pues son controles ajenos a la librería y añadidos personalmente por el desarrollador. La estructura de estos consiste en generar un control con la sintaxis ***ol.control.Control()*** pasándole como parámetro el elemento HTML que hospeda la funcionalidad en nuestro caso serán contenedores con la etiqueta **div**.

```
var mousePosition = new ol.control.MousePosition({
    className: 'mousePosition',
    projection: "EPSG:3857",
    coordinateFormat: function(coordinate){
        return ol.coordinate.format(coordinate, '{x} , {y}',6);
    }
});

map.addControl(mousePosition);
```

Ilustración 110: Código ejemplo de la creación de un tipo de control (autor).

Para la selección se desarrolla simplemente un botón que cambie el valor de un atributo global booleano al ser pulsado, siendo este atributo el que determine si en el script generado en la segunda iteración se puede llegar a seleccionar una parcela o no.

El control de búsqueda por municipios es algo más complejo ya que requiere de acceso a las entidades de la capa de municipios cosa que OpenLayers no facilita en sus funcionalidades por ende el proceso seguido es el siguiente:

1. Se recibe en un campo HTML search dinámicamente la escritura del usuario mediante el uso de un listener que responda ante cada actualización.

2. Cada vez que el texto mostrado supere los 2 caracteres se recurrirá a una búsqueda mediante php en la base de datos.

3. Se devuelven los elementos coincidentes en nombre y se listan por pantalla.

4. En caso de recibir un clic se procesa una petición WFS al Geoserver por URL que devuelve la entidad del municipio.

5. Se representa la entidad y se hace zoom a la zona.

Este proceso se encontrará activo dinámicamente cada vez que el usuario se encuentre sobre la barra de búsqueda.

Una vez terminado el desarrollo de este último control el mapa nos quedaría con un aspecto como el de la Ilustración 111.

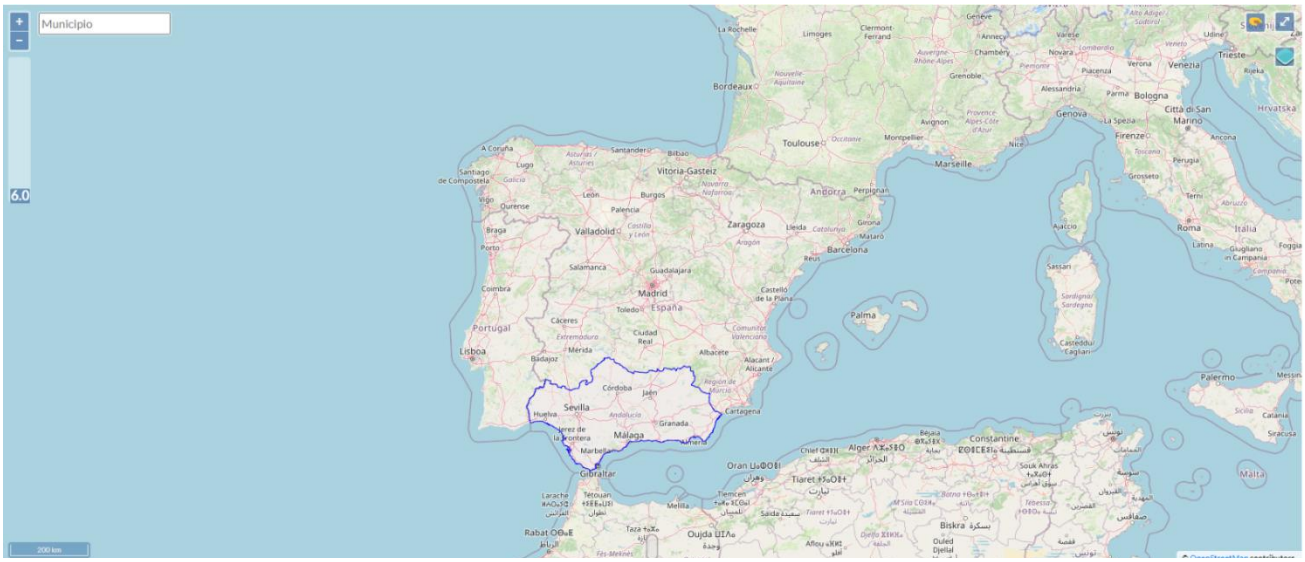


Ilustración 111: Mapa interactivo final con todos los controles (autor).

5.5.4 Trabajo con texturas en la escena

Como se ha comentado durante los inicios de este documento, así como en otros muchos apartados un sistema donde siempre se disponga de nube de puntos para todas y cada una de las parcelas es una situación utópica, por este motivo se plantea como objetivo poder disponer de datos aún sin haber realizado un vuelo aún en la parcela.

Para cumplir este requisito se plantea la representación de la parcela con información gráfica mediante texturas, siendo estas las ortofotos extraídas del PNOA, de esta forma si no disponemos de nube tenemos una representación de la parcela y si disponemos de nube podemos enriquecer la información mediante combinación de datos.

El objetivo final planteado para esta tarea es disponer de una escena representada por un polígono, estando definido por las coordenadas de la parcela (disponibles en la base de datos). Sobre este polígono se aplica la textura extraída de las ortofotos disponible en el servidor y se aplica en caso de contar con él un mapa de sombras para aplicarle relieve y tridimensionalidad al polígono obteniendo el modelo más fiel posible a la parcela original.

Una vez planteados los objetivos de esta tarea se procederá a explicar cómo se ha desarrollado este aspecto de la escena.

La representación de una parcela como una entidad bidimensional no deja de ser disponer de un polígono al cual le aplicamos como textura una ortofoto de la propia parcela. Este proceso es sencillo dentro de BabylonJS ya que dispone de entidades polígono en su documentación que solo requieren de pasarle los vértices correspondientes.

En esta tarea la intención es obtener la representación más afín posible a la realidad de la parcela y para esto lo que haremos será extraer las coordenadas reales de la parcela que tenemos almacenadas en la base de datos.

Para la extracción de coordenadas tendremos que realizar una consulta a la base de datos donde extraigamos el atributo **SHAPE** de la tabla geometría_parcela como texto para procesar las coordenadas. Esto es posible conseguirlo con la función **AsText(atributo)** de SQL quedando una consulta tal que:

Select AsText(SHAPE) FROM geometría_parcela where Id_recinto = VALOR

Esta consulta nos devolverá las coordenadas con un formato concreto para el cual en primer lugar y entre paréntesis nos devuelve la geometría exterior de la parcela (la que buscamos) y de manera posterior a esta se presentarán entre paréntesis también las

coordenadas de tantos huecos como tenga el polígono de nuestra parcela. Como en nuestro caso queremos representar el polígono relleno por completo ignorando los huecos nos quedaremos con la primera parte exclusivamente.

Una vez disponemos de las coordenadas tendrán los valores correspondientes a la proyección en la que se subieron los datos a la base de datos, para disponer de esta centrada se genera la Bounding Box del polígono con los puntos máximo y mínimos, aplicaremos finalmente una traslación a todas las coordenadas ubicando el centro de la Bounding Box en el centro de coordenadas de la escena.

Finalmente, tras esto habrá que generar el polígono y aplicarle textura. Las funciones en BabylonJS para conseguir esto son:

- **Crear el polígono:** BABYLON.Mesh.CreatePolygon(coordenadas).
- **Crear el material:** BABYLON.StandardMaterial(escena).
- **Asignar textura:** Material.diffuseTexture = Babylon.Texture(imagen, escena).
- **Aplicar el material:** Polygon.material = material.

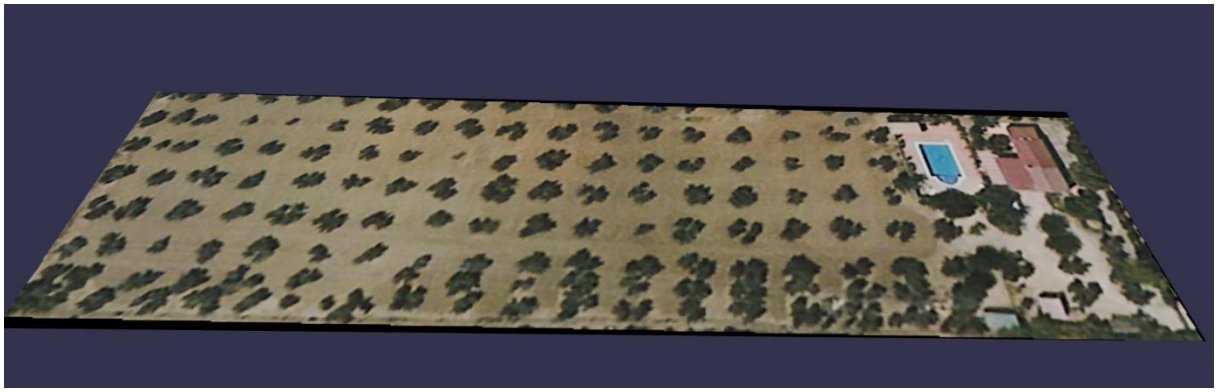


Ilustración 112: Resultado tras aplicar la textura al polígono (autor).

Por último, quedaría dotar de inclinación al modelo para generar una representación tridimensional del mismo. Esto se consigue con la simulación de elevación del polígono mediante la alteración de la coordenada Y de los vértices del mismo.

Para conseguir esto tendremos que trabajar con métodos internos de BabylonJS para la modificación de la Mesh construida así como con la nube de puntos para tratar de extraer las coordenadas de altura correspondientes.

En primer lugar, se emplearán y explicarán las funcionalidades de BabylonJS. En primer lugar, deberemos crear un objeto del tipo VertexData que se corresponde con un objeto que almacena las posiciones, índices y normales de los vértices, estas serán importantes ya que son las que debemos alterar para pasar de un polígono 2D a misma altura a uno con inclinación para simular la pendiente de la parcela.

Al objeto de VertexData será al que asignaremos los nuevos valores accediendo a sus atributos **positions** e **índices**. Los índices serán los mismos que el polígono original y se obtendrán con la función **Polygon.getIndices()**. Por último los cambios se asignarán y aplicarán a la mesh con la función **VertexData.applyToMesh(polygon).** Una vez conocido como aplicar cambios a la mesh será necesario obtener las coordenadas de altura correctas.

Esta última parte se conseguirá mediante la asociación con la nube, en el caso donde se encuentran ambos modelos alineados deberemos recorrer la nube buscando el punto más cercano a cada vértice basado en coordenadas x/z y coger su altura, para los posibles puntos de la textura que no colisionen con la nube por estar esta última incompleta se mantendrá una altura de 0.

En iteraciones futuras este proceso se mejorará gracias a la voxelización ya que podremos obtener los voxeles correspondientes a la coordenada del polígono y mirar exclusivamente en los puntos de este voxel.

Esta funcionalidad será implementada solo en casos con nube de puntos disponible en el resto se deberá disponer exclusivamente de un polígono sin inclinación. Además, esta funcionalidad se encuentra limitada debido a las dificultades de coincidencia espacial de ambos modelos en un mismo sistema de referencia, es por este motivo que se encuentra implementada pero debido a esta problemática no pudo obtener una representación visual hasta futuras iteraciones donde se consiguió una coincidencia espacial lo más acertada posible. A continuación se muestra la imagen resultado del plano con inclinación que consiguió visualizarse al desarrollar la Sección 5.6.4.3

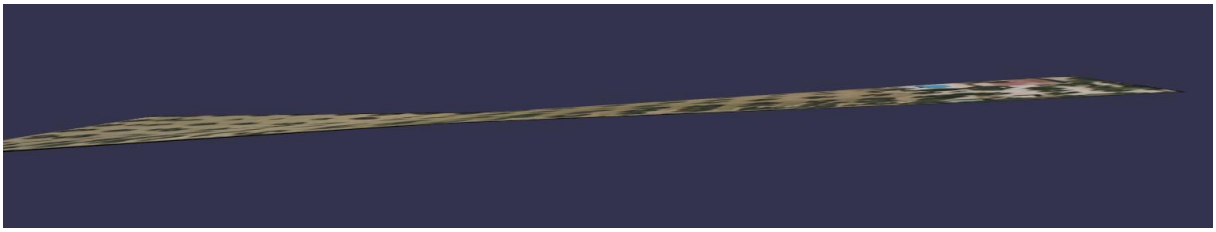


Ilustración 113: Simulación de inclinación del plano con respecto a la nube (1) (autor).

Ilustración 114: Simulación de inclinación del plano con respecto a la nube (2) (autor).

5.5.5 Pruebas en dispositivos móvil / Tablet

Una vez finalizadas las tareas de esta iteración y con muchas de las funcionalidades básicas disponibles ya en la aplicación desarrollada llega el momento de realizar pruebas de rendimiento y aspecto visual es dispositivos externos al ordenador ya que uno de los objetivos y requisitos de esta práctica era la accesibilidad a la aplicación desde cualquier lugar y por ende cualquier dispositivo.

Para la realización de estas pruebas al disponer del código y recursos en local se empleó un túnel seguro mediante SSH esta técnica también conocida como SSH Tunneling permite a un usuario navegar por la web con seguridad a través de un túnel SSH hacia un servidor SSH remoto. Para conseguir esto se empleó el servicio **Ngrok**[25] que está especializado en la creación de este tipo de túneles estableciendo nuestro servidor en un subdominio de la web.

De esta forma si disponemos del servidor en la dirección http://localhost:8080 como es nuestro caso con XAMPP el servicio genera una URL del tipo *http://xxxxx.ngrok.io* accesible desde cualquier otro dispositivo y que apunta a nuestro servidor.

Una vez conocemos la herramienta se procederá a explicar el proceso para utilizarlo.

1. Entrar en la página ngrok.com con una cuenta de correo de Google o GitHub
2. Descargar la carpeta comprimida que nos ofrece
3. Descomprimirla
4. Depositar el ejecutable resultante en la carpeta donde se ubica nuestra página / aplicación web
5. Abrir el terminal y ejecutar el comando: ngrok config add-authtoken <token>

[25] Página oficial Ngrok: https://ngrok.com/

6. Posteriormente ejecutar el comando ngrok http 80

Después de realizar estos pasos la terminal se rellena con la información que se puede observar en la Ilustración 115 entre la cual podremos encontrar el dominio que buscar para acceder, conforme se acceda a la página se irá actualizando el terminal con las peticiones recibidas debido a la demanda de otros usuarios.

Ilustración 115: Terminal tras ejecutar ngrok (autor).

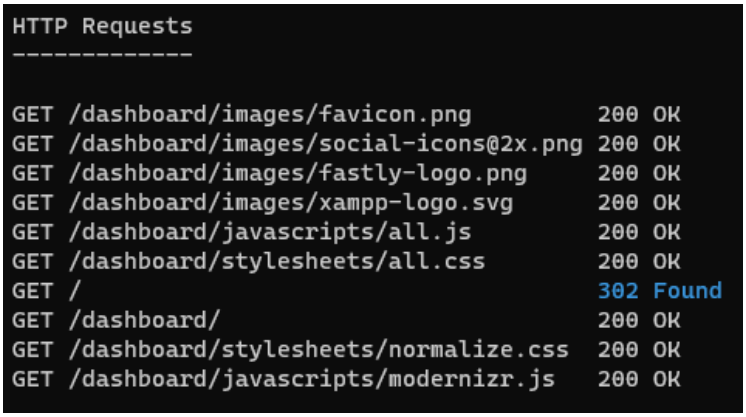


Ilustración 116: Peticiones resultantes de una conexión desde un terminal (autor).

Una vez realizado todo este proceso es posible realizar las pruebas necesarias desde Tablet y móvil para extraer conclusiones de rendimiento y estilismo visual de la interfaz.

Tras probar en varios dispositivos se sacaron conclusiones acerca de posibles requisitos mínimos para poder ejecutar correctamente ya que tras realizar ejecuciones con la escena que es la página que más recursos puede necesitar se encontraron dispositivos donde la carga tardaba demasiado o no llegaba a cargar quedándose pillado. Estos requisitos mínimos se estimaron en al menos disponer de 3GB de RAM en el mismo para obtener tiempos de respuesta aceptables, aunque contando con que siempre estos siempre serán mayores que en un ordenador.

Finalmente, en cuanto al estilismo y visibilidad surgieron ciertos problemas o partes de interfaces que no resultaron del agrado del desarrollador al pasarse a dispositivos más pequeños. Por este motivo se añadieron dos nuevos scripts que es encargan de ajustar la página a nuevas dimensiones o estilos en función del tamaño de la pantalla donde se trabaje, para esto se trabaja con lo que se conoce como **Media Query** que es una característica principal de CSS que permite adaptar los diseños en función del tamaño de la ventana. Aunque principalmente pertenece al lenguaje de diseño CSS también pueden aplicarse algunas funciones con estas en JavaScript.

5.5.6 Resumen trabajo realizado

Al finalizar esta etapa y gracias al desarrollo de pruebas en dispositivos móviles se pudieron realizar las pruebas de feedback y de muestra de los entregables en sus propios dispositivos, en estos pudieron trabajar con el nuevo mapa interactivo y sus funcionalidades, pudieron probar la detección de olivos ellos mismos y observar correctamente los planos texturizados.

Por último pudieron probar una de las funcionalidades que más cuello de botella podía llegar a producir como era el recorte de nube de puntos, en esta fase al ver que según el recorte el tiempo podía llegar a alargarse se optó por proponer como tarea futura la voxelización de la nube y probar como optimizaría el tiempo.

5.6 Quinta Iteración

Esta iteración se corresponde con la última de desarrollo de funcionalidades principales de la aplicación para el cumplimiento de la mayor parte de los requisitos planteado inicialmente. En esta se desarrolla GUI con el fin de facilitar el manejo al usuario de la escena y poder acceder a todas las funcionalidades disponibles, además se realiza una mejora del algoritmo de detección de olivos en las texturas con el fin de obtener mejores resultados.

Finalmente, para completar la aplicación se aplicarán algoritmos con el fin de conseguir una retroalimentación de información entre las texturas y la nube de puntos, así como un proceso de voxelización para optimizar rendimiento y se genera un sistema backend para la gestión de modelos y usuarios por parte de administradores.

5.6.1 Desarrollo de la voxelización

La Voxelización [21] es una operación cuyo principal objetivo es la reducción de complejidad computacional de ciertas operaciones mediante la representación aproximada de la geometría con Voxeles que son geométricamente iguales a un cubo.

En nuestro proyecto la Voxelización puede llegar a facilitar ciertas tareas ya que uno de los principales cuellos de botella de la misma es que al ser una aplicación que funciona en un navegador se encuentra mucho más limitada en rendimiento. La Voxelización se representa mediante una estructura de datos en forma de matriz tridimensional donde cada celda almacena los puntos que espacialmente se ubican dentro de esta.

5.6.1.1 Proceso de Voxelización

El proceso necesario para la generación de este tipo de estructura de datos necesita exclusivamente de un tamaño de voxel que en nuestro caso no será muy grande por la limitación computacional, así como los valores máximos y mínimos en cada eje de la nube, una vez disponemos de estos datos el proceso de construcción es muy simple y se corresponde con el siguiente pseudocódigo.

```
Procedimiento Voxelizacion(x, y, z, tamVoxel)
    //Se suma uno ya que la función entero redondeará hacia abajo
    numVoxelesX <- Entero((x.Max - x.Min)/tamVoxel) + 1
    numVoxelesY <- Entero((y.Max - y.Min)/tamVoxel) + 1
    numVoxelesY <- Entero((z.Max - z.Min)/tamVoxel) + 1

    voxelizacion <- array(numVoxelesX)
    Para i <- 1 Hasta numVoxelesX Con Paso 1 Hacer
        voxelizacion[i] <- array(numVoxelesY)
        Para x <- 1 Hasta numVoxelesY Con Paso 1 Hacer
            voxelizacion[i][x] <- array(numVoxelesZ)
            Para z <- 1 Hasta numVoxelesZ Con Paso 1 Hacer
                voxelizacion[i][x][z] <- array()
            Fin Para
        Fin Para
    Fin Para
Fin Procedimiento
```

Ilustración 117: Pseudocódigo creación estructura de datos Voxelización (autor).

El proceso requiere exclusivamente de determinar en primer lugar el número de voxeles por eje, de esta forma se consigue la creación de tantas celdas de la matriz como sean necesarias por eje. Una vez obtenida la estructura de datos será necesario rellenarla de puntos, esto se realiza durante el proceso de creación del PCS que vimos en la Ilustración 91.

En el pseudocódigo relativo a la carga de la nube de la ilustración antes mencionada será necesario añadir de forma previa a la creación de la nube una llamada a la función Voxelización que retorne la estructura de datos. Finalmente, cada punto se asigna a una celda, la celda correspondiente en cada eje al punto se extraerá con la fórmula:

$$Eje = Entero(\frac{coordenadaEje - minNube}{tamVoxelEje})$$

Tras aplicar esta fórmula a cada uno de los tres ejes obtendremos los tres valores de celda necesarios donde insertar el punto dentro de la estructura de datos. Una vez realizado esto el proceso de Voxelización estará completo.

Para comprobar la correcta ejecución de la voxelización procederemos a dibujarla para ver que encierra correctamente a la nube y por ende el proceso es exitoso. Para esto utilizaremos la función **showBoundingBox** de BabylonJS.

En primer lugar, debemos crear las cajas que harán de voxeles y sobre las que aplicaremos la Bounding Box, para esto utilizaremos la opción **MeshBuilder.CreateBox** de BabylonJS donde nos genera una caja del tamaño que indiquemos, en nuestro caso será el tamaño de voxel. Posteriormente la movemos a la posición correspondiente de este voxel cuyo centro será la posición mínima del Voxel más el tamaño dividido por dos. Por último, crearemos un objeto del tipo **Mesh** que hace de padre de la caja y al que asignaremos la información de los límites de está poniendo el valor de showBoundingBox a true y eliminando finalmente la caja original para que no interfiera en la visualización.

El código resultante y la visualización final de la voxelización se pueden ver en las siguientes ilustraciones.

```
const box = BABYLON.MeshBuilder.CreateBox("box", { height: tamVoxel, width: tamVoxel, depth: tamVoxel });
box.position = new BABYLON.Vector3(auxX + tamVoxel / 2, auxY + tamVoxel / 2, auxZ + tamVoxel / 2);
let parent = new BABYLON.Mesh("parent", scene);

box.setParent(parent);

let boxMin = box.getBoundingInfo().boundingBox.minimumWorld;
let boxMax = box.getBoundingInfo().boundingBox.maximumWorld;

parent.setBoundingInfo(new BABYLON.BoundingInfo(boxMin, boxMax));

parent.showBoundingBox = true;
box.dispose();
```

Ilustración 118: Código relativo a la creación de las Bounding Box (autor).

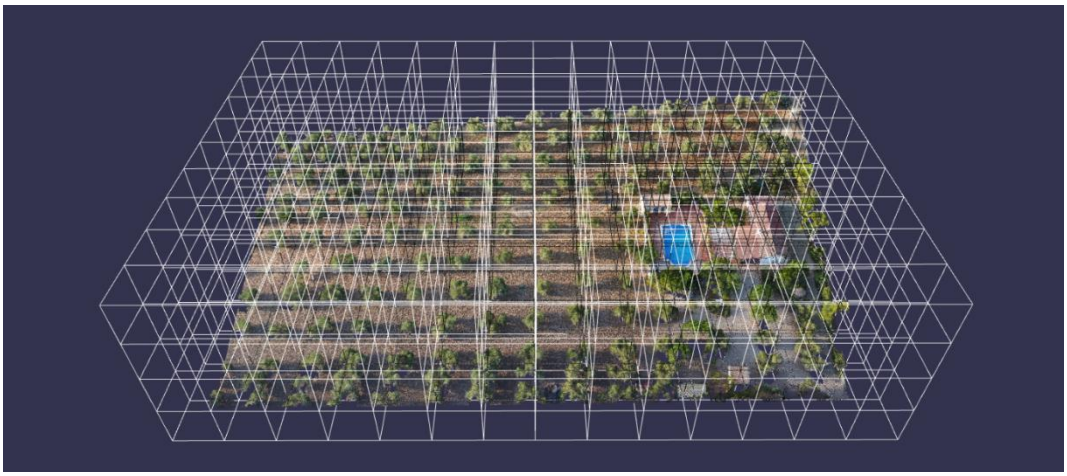


Ilustración 119: Representación escena voxelizada (autor).

De estas dos imágenes cabe reseñar que los valores de **aux** del código principal son los que almacenan los valores mínimos de las esquinas de cada voxel. En la visualización además podemos observar que hay muchos voxeles que se encontrarán vacíos, esto se debe a la distribución de esta nube que tiene una gran inclinación lo que provoca grandes diferencias de valores entre el mínimo y máximo de Y.

5.6.1.2 Aplicaciones

En el caso de nuestra aplicación la única operación que puede reducir su complejidad es el recorte de la nube de puntos aunque disponer de la Voxelización puede ser de ayuda ante posibles adiciones futuras de interacciones con la nube.

En el proceso de recorte la Voxelización puede ser de utilidad ya que en lugar de necesitar consultar a todos los puntos si se encuentran dentro de la nube será necesario preguntar exclusivamente a los puntos de los voxeles que caigan dentro del recorte. Para esto tendremos que quedarnos con los voxeles máximos y mínimos de cada eje para los puntos del polígono de recorte y recorrer exclusivamente los puntos ubicados en estos voxeles. El proceso final de recorte pasará de recorrer todos los puntos a realizar el proceso mostrado en el pseudocódigo de la Ilustración 120.

```
Procedimiento Recorte(coordenadas, voxelizacion)
    voxelMinimos, voxelMaximos <- ObtencionMinimos(coordenadas, voxelizacion)
    nubeRecortada <- array()
    Para i <- voxelMinimos.x Hasta voxelMaximos.x Con Paso 1 Hacer
        Para x <- voxelMinimos.y Hasta voxelMaximos.y Con Paso 1 Hacer
            Para z <- voxelMinimos.z Hasta voxelMaximos.z Con Paso 1 Hacer
                Para p <- 0 Hasta puntosVoxel Con Paso 1 Hacer
                    Si puntoPoligono(voxelizacion[i][x][z][p], coordenadas) Entonces
                        nubeRecortada <- voxelizacion[i][x][z][p]
                        Si datos[i] > puntoMax Entonces
                            puntoMax <- datos[i]
                        Fin Si
                        Si datos[i] < puntoMin Entonces
                            puntosMin <- datos[i]
                        Fin si
                    Fin Si
                Fin Para
            Fin Para
        Fin Para
    Fin Para

    puntoMedio <- (puntoMax + puntoMin) / 2
    pcs <- BABYLON.PointCloudSystem(scene)
    Para i <- 1 Hasta tam(nubeRecortada) Con Paso 1 Hacer
        p.position <- nubeRecortada[i].position - puntoMedio
        p.color <- nube[i].color
        pcs.addPoint(1, p)
    Fin para
    pcs.buildMeshAsync()
Fin Procedimiento
```

Ilustración 120: Pseudocódigo recorte con voxelización (autor).

Una vez implementado el código es el momento de comprobar si realmente la voxelización supone una mejora de rendimiento para la escena para conseguir esto se realizarán múltiples ejecuciones de recorte de la misma zona con ambos métodos y se realizarán mediciones de tiempo para observar el resultado y los puntos donde realmente supone una mejora este cambio. Cada recorte se ha realizado en 5 ocasiones y el tiempo resultante es la media de los tiempos obtenidos.

Puntos	Método	Tiempo (ms)
830031	Clásico	827
	Voxelizado	1006
386044	Clásico	570
	Voxelizado	501
240246	Clásico	530
	Voxelizado	380
57905	Clásico	389
	Voxelizado	157

Tabla 25: Tabla tiempos recorte parcela.

Con el fin de poder ilustrar de mejor forma los tiempos obtenidos en la tabla anterior de comparación de métodos se ha realizado la siguiente gráfica donde podemos observar que a menor cantidad final de puntos en el recorte mayor es el incremento de la diferencia debido al menor número de comprobaciones necesario, al contrario, en la primera ejecución que contenía toda la nube es más costoso recorrer todos los voxeles incluyendo vacíos que el comprobar secuencialmente los puntos.

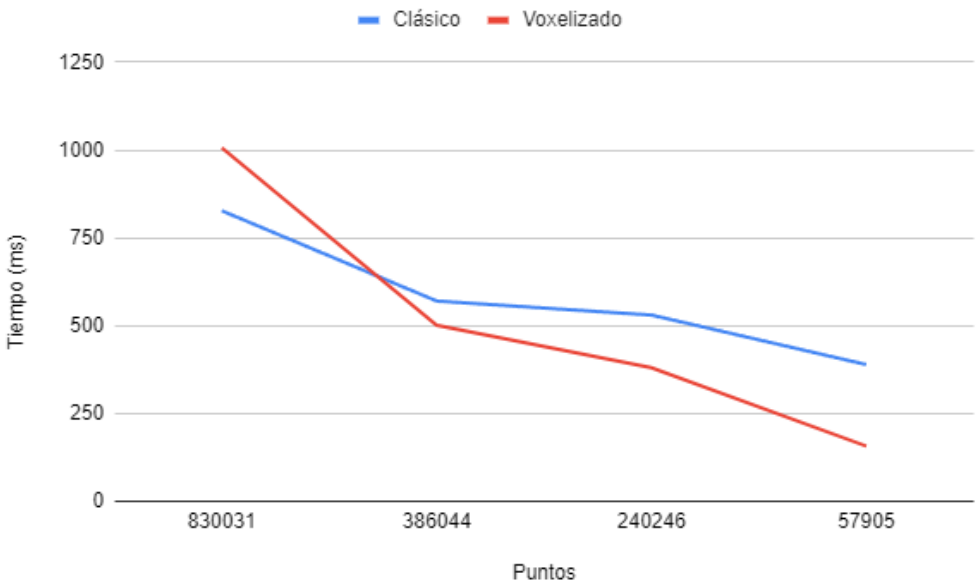


Ilustración 121: Gráfica comparativa métodos de recorte (autor).

5.6.2 Desarrollo de una GUI para la escena

El control de la escena es un aspecto muy importante y para el cual se debe facilitar todo lo posible el trabajo al usuario con el objetivo de que este se sienta cómodo y seguro ante las acciones que va a realizar en cada momento.

Por este motivo y con el objetivo de disponer de una interfaz lo más amigable posible para su interacción se plantea la tarea de desarrollar un menú o interfaz de usuario especial para el manejo de ciertas funcionalidades de la escena.

El trabajo correspondiente para obtener un resultado satisfactorio pasa por dos fases que son la obtención de una interfaz visualmente y funcionalmente aceptable y una segunda fase de cambios en la interfaz base y desarrollo de la gestión de funcionalidades por parte de la misma.

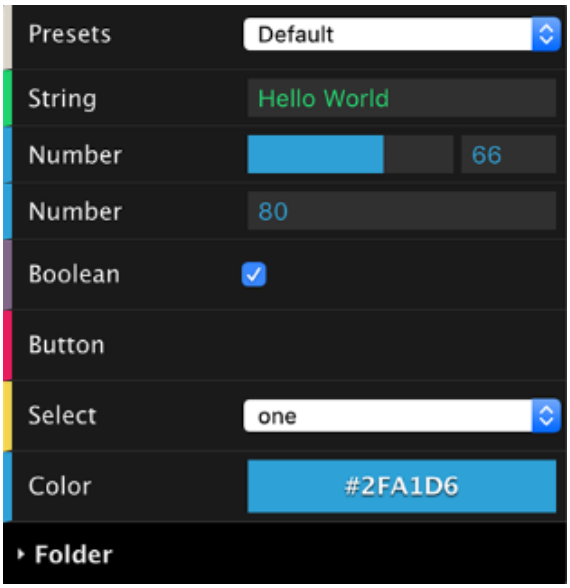


Ilustración 122: Ejemplo interfaz usuario Dat GUI[26]

Para el desarrollo gráfico de la interfaz y funcionalidades básicas el código ha sido extraído de la librería ThreeJS que es específica para gráficos y renderizado 3D para WebGL. La interfaz de usuario se denomina Dat GUI [22] y es una interfaz simple diseñada especialmente para la interacción con escenas 3D, su aspecto básico es del estilo mostrado en la Ilustración 122.

Una vez tenemos el código relativo a esta interfaz descargado podemos introducirlo en el fichero de la escena y comenzar a editar el código JavaScript para manejar las funcionalidades que deseemos. Las principales funcionalidades y opciones que se quieren manejar son:

- Cuando se encuentra activada la opción de selección para recorte. Determina mediante un booleano si es posible iniciar la selección poligonal.
- Formato del modelo a presentar en cámara (nube puntos, ortofoto, mixto). Cambiar entre los modelos a visualizar en la escena.
- Nivel de zoom y rotaciones / traslaciones. Cambian las propiedades de la cámara para modificar la visión de la escena

[26] https://github.com/claus/react-dat-gui

- Detección entidades. Botón que ejecuta el script de detección de entidades.
- Opción de navegación (volver página anterior).
- Tipo de perspectiva de la cámara. Cambian entre dos cámaras existentes con funcionalidades distintas.
- Opción para reconstruir la nube de puntos completa si se ha ocurrido algún recorte.
- Ejecución algoritmo de Intersección Ray-Voxelization para proyección puntos.

Para conseguir todo esto se deberá tener acceso a entidades y variables definidas previamente en la escena. El proceso de generación de todos estos botones es simple y estándar y se ha basado todo en lo encontrado en la documentación de la librería.

5.6.3 Desarrollo de un nuevo Script para detección de entidades

Durante la tercera iteración se desarrolló el script de detección de entidades que posteriormente en la cuarta iteración se integró con éxito en nuestra aplicación web y en nuestra escena, sin embargo, como se comentó en las iteraciones previas este script es muy básico.

En esta iteración se pretende mejorar el proceso de detección aumentando la tasa de acierto mediante la aplicación de un mayor número de algoritmos de procesamiento de imágenes, así como el refinamiento de los utilizados en el primero.

En comparación con el primer script los cambios realizados son los siguientes:

- **Cerrado de contornos**: En el script previo se extraían con Canny [23] [24] los bordes y contornos de la imagen, pero algunos de estos permanecían abiertos y esto llevaba a errores en la clasificación, en este script se cierran los contornos con la función **morphologyEx.**
- **Agrupamiento:** Una vez se dispone de los contornos cerrados se establece el centro de cada contorno y se mandan a la función **dbscan2d**[27] (Density-based Spatial clustering of applications with noise) [25] que agrupa estos en clústeres.

[27] Algoritmo de agrupamiento espacial basado en densidad de aplicaciones con ruido.

- **Segmentación de clúster:** Finalmente de los clústeres resultantes se agrupan aquellos que sean muy similares / cercanos y nos quedamos con los clústeres finales que serán las posiciones centrales de los olivos.

En caso de querer acceder al código del Script final de detección de entidades de este proyecto se podrá acceder desde su repositorio en GitHub[28]

Una vez explicado el contenido del nuevo algoritmo el siguiente trabajo era el de realizar pruebas sobre distintas parcelas y observar los resultados, estos fueron los siguientes:

Parcela 1

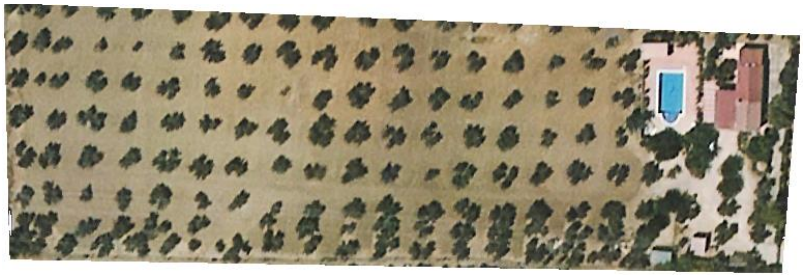


Ilustración 123: Parcela pruebas 1 (autor).

Ilustración 124: Resultados pruebas 1 (autor).

Parcela 2

Ilustración 126: Parcela pruebas 2 (autor).

Ilustración 125: Resultados pruebas 2 (autor).

Parcela 3

Ilustración 127: Parcela prueba 3 (autor).

Ilustración 128: Resultados prueba 3 (autor).

[28] Repositorio Olive-Segmentation: https://github.com/PabloLHo/Olive-Segmentation.

Parcela	Positivos	Falsos positivos	% Incremento Positivos	% Incremento Falsos Positivos
1	110/130	13	107%	116%
2	87/105	3	148%	200%
3	41/54	16	41%	77%

Tabla 26: Tabla resultados primer Script OpenCV.

Como se puede observar el porcentaje de entidades bien marcadas ha crecido considerablemente, por otro lado, el número de falsos positivos también, sin embargo, estos datos son más grandes en porcentaje debido a que de base había pocos falsos positivos por lo que un ligero incremento genera un gran porcentaje.

En adición al script y con el objetivo de poder manualmente solventar algunos de los problemas de la detección se ha implementado en la aplicación la posibilidad de manipular las detecciones manualmente (Ilustración 129), para esto se ha colocado en cada posición detectada un objeto 3D con forma de árbol que permite ser seleccionado y que tras seleccionarse te ofrece las posibilidades de eliminarlo (falso positivo) o bien desplazarlo si la posición esta levemente movida hacia algún lado. Los cambios realizados se almacenarán para interacciones inmediatas después de la detección, en caso de cambios de modelos o rehacer la detección se deberán volver a aplicar los cambios realizados.

Ilustración 129: Detección editable (autor).

5.6.4 Fusión de datos entre plano texturizado y nube de puntos

En ocasiones cuando se dispone de distintas capas de información la fusión entre estas pudiera considerarse de interés para la obtención de nueva información o enriquecimiento de datos en alguna de las capas disponibles.

Para este proyecto se desarrolla una funcionalidad que nos permitirá situar los olivos dentro de la nube de puntos. Para esto lo que haremos será aprovechar la detección realizada previamente sobre la textura que nos otorga posiciones exactas de los centros de estos y aprovechando la coincidencia espacial de textura y nube se procederá con un proceso de interacción entre ambas capas de información.

El proceso consta de un algoritmo simple de intersección Rayo-Voxelización para el cual se lanza un rayo desde la posición exacta donde se encuentra el olivo detectado hacia arriba (Ilustración 131) y se almacenarán para cada rayo los puntos intersecados y algunos cercanos espacialmente y se definirán como puntos del tipo olivo.

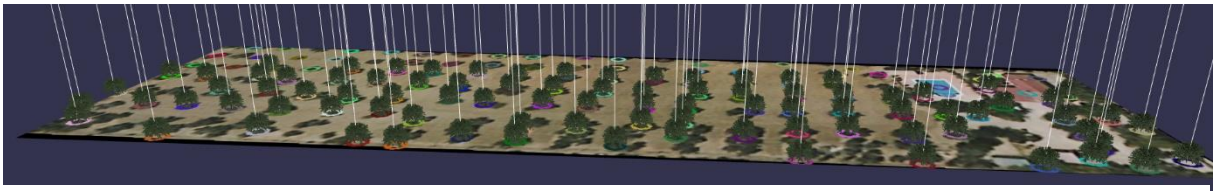


Ilustración 131: Visualización ejemplo lanzamiento rayo (autor).

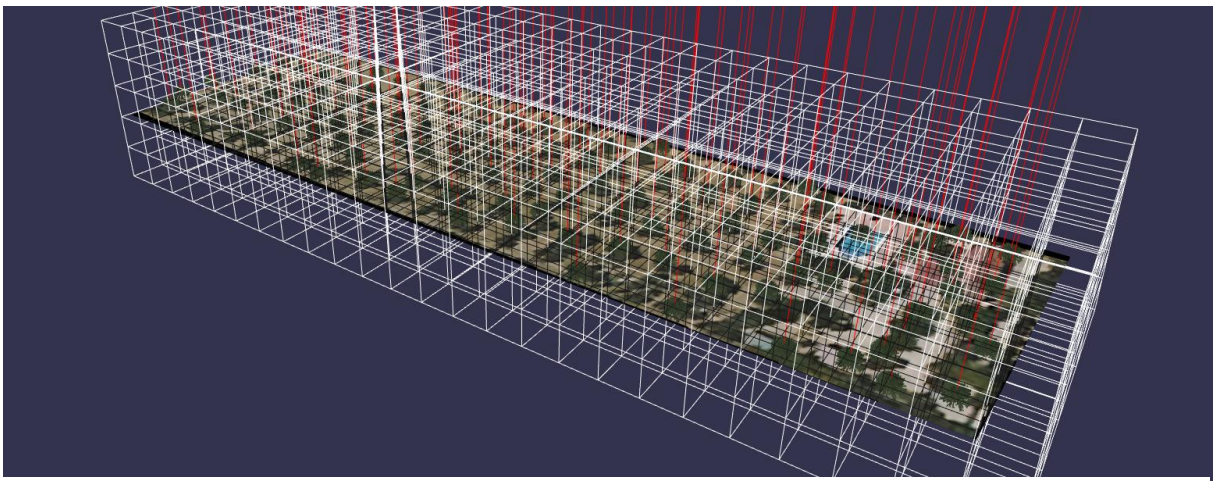


Ilustración 130: Visualización lanzamiento de rayos (rojos) y como intersecan con la Voxelización (autor).

Con este proceso es posible conseguir extraer información relevante como puede ser una posición más exacta del olivo en la base de datos, extracción de una posible estimación de la altura del olivo, cercanía con otros olivos… Con el fin de mejorar todo lo posible el algoritmo desarrollado podría tratarse de enriquecer la información, asegurar la detección y eliminar posibles fallos de clasificación del script, marcando exclusivamente aquellas zonas de puntos dotadas de una gama de color correspondiente al color del olivo.

Finalmente, a continuación se presenta el pseudocódigo correspondiente de todo el proceso de enriquecimiento de la información:

Como podemos observar en el pseudocódigo asociado y pudimos ver en la Ilustración 130 previamente presentada el proceso seguirá el siguiente orden:

```
Procedimiento ProcesoNubeTextura(coordenadas)
    olivos = array(coordenadas)
    Para i <- 1 Hasta tam(coordenadas) Con Paso 1 Hacer
        olivo <- ImportarMesh(modelo)
        olivo.position <- coordenadas[i]
        olivos[i] = olivo
        ray <- lanzarRayo(olivo, vector(0,1,0))
        voxeles <- Ray_Voxelization(ray, voxelizacion)
    Fin Para
Fin Procedimiento
```

Ilustración 132: Pseudocódigo combinación de información entre nube y textura (autor).

1. En primer lugar, requerirá de la obtención de las coordenadas de la detección de entidades por OpenCV.

2. Se posiciona una Mesh de referencia importada con formato de árbol desde la cual se lanza un rayo de forma perpendicular a la superficie por ende con el vector dirección (0,1,0).

3. Se obtendrán los voxeles intersecados por el rayo.

Una vez finalizado esto y como veremos más adelante los puntos internos de los voxeles extraídos serán analizados para poder obtener conocimiento de estos y obtener más información para la base de datos.

5.6.4.1 Intersección Rayo-Voxelización

Para la ejecución del algoritmo de intersección se ha recurrido a la voxelización con el fin de mejorar el rendimiento y acceder exclusivamente a los puntos necesarios para esto, además, se ha estudiado el funcionamiento de múltiples formas de intersecarlos, la más conocida es el algoritmo del Ray-Traversal [26]. Este algoritmo busca mediante el uso de la posición y dirección del rayo determinar de forma incremental con que voxeles va chocando el rayo y almacenando estos voxeles, a posteriori podremos realizar un análisis de los puntos incluidos en los voxeles almacenados para determinar cuales se corresponden con olivos. El pseudocódigo relativo a la adaptación del código de este algoritmo para JavaScript ha sido el siguiente:

```
Procedimiento Ray-Traversal(rayo, voxelizacion, voxeles)
    //En caso de intersección devuelve la zona de entrada y salida
    tMin, tMax <- rayoInsersectaVoxelizacion()
    Si rayoInsersectaVoxelizacion() Entonces
        tMin <- Max(tMin, 0)
        tMax <- Max(tMax, 1)
        ray_start <- rayo.origin() + rayo.direction() * tMin
        ray_end <- rayo.origin() + rayo.direction() * tMax

        //Repetir este proceso para los tres ejes sustituyendo Eje por X,Y,Z
        current_Eje_index <- Max(1, (ray.start.eje - minEje) / tamVoxel)
        Si rayo.direction().Eje > 0 Entonces
            StepEje <- 1
            deltaEje <- tamVoxel / rayo.direction().Eje
            tMaxEje <- tMin + minEje +
                    currentEje_index * tamVoxel - ray_start.Eje / rayo.direction.Eje
        Sino Si rayo.direction().Eje < 0 Entonces
            StepEje <- -1
            deltaEje <- tamVoxel / -rayo.direction().Eje
            previus_Eje_index <. current_Eje_index -1
            tMaxEje <- tMin + minEje +
                    currentEje_index * tamVoxel - ray_start.Eje / rayo.direction.Eje
        Sino
            StepEje <- 0
            deltaEje <- tMax
            tMaxEje <- tMax
        Fin Si
        Mientras 1 <= current_X_index <= numVoxeles_X &&  1 <= current_Y_index <= numVoxeles_Y
            && 1 <= current_Z_index <= numVoxeles_Z Hacer
                voxeles <- voxelizacion[current_X_index][current_Y_index][current_Z_index]
                Si tMaxX < tMaxY Y tMaxX < tMaxZ Entonces
                    current_X_index <- current_X_index + stepX
                    tMaxX <- tMaxX + deltaX
                Sino Si tMaxY < tMaxZ Entonces
                    current_Y_index <- current_Y_index + stepY
                    tMaxY <- tMaxY + deltaY
                Sino
                    current_Z_index <- current_Z_index + stepZ
                    tMaxZ <- tMaxZ + deltaZ
                Fin Si
    Sino
        Devolver Vacio
    Fin Si
Fin Procedimiento
```

Ilustración 133: Pseudocódigo algoritmo Ray-Traversal (autor).

A posteriori de realizar la implementación de algoritmo previamente explicado se encontró también que se disponía de una función nativa en la documentación de BabylonJS denominada **intersectsBox** perteneciente a la clase **Ray** lo cual nos permitía pasarle al rayo una Bounding Box y que la función devolviera True o False en función de si intersecaba o no. La aplicación de esto era muy sencilla pues exclusivamente tenías que recorrer toda la voxelización preguntando si intersecaba o no. Tras observar que los resultados eran

coincidentes se optó por realizar múltiples pruebas con el fin de analizar rendimiento y tiempos para determinar cuál es la mejor opción para nuestra escena.

Olivos	Voxeles	Método	Tiempo (ms)
87	**625**	Ray Traversal	7123
		BabylonJS	7675
168	**1000**	Ray Traversal	16823
		BabylonJS	17852
269	**3190**	Ray Traversal	97619
		BabylonJS	103909

Tabla 27: Análisis tiempos entre algoritmos de Intersección Ray-Voxelization.

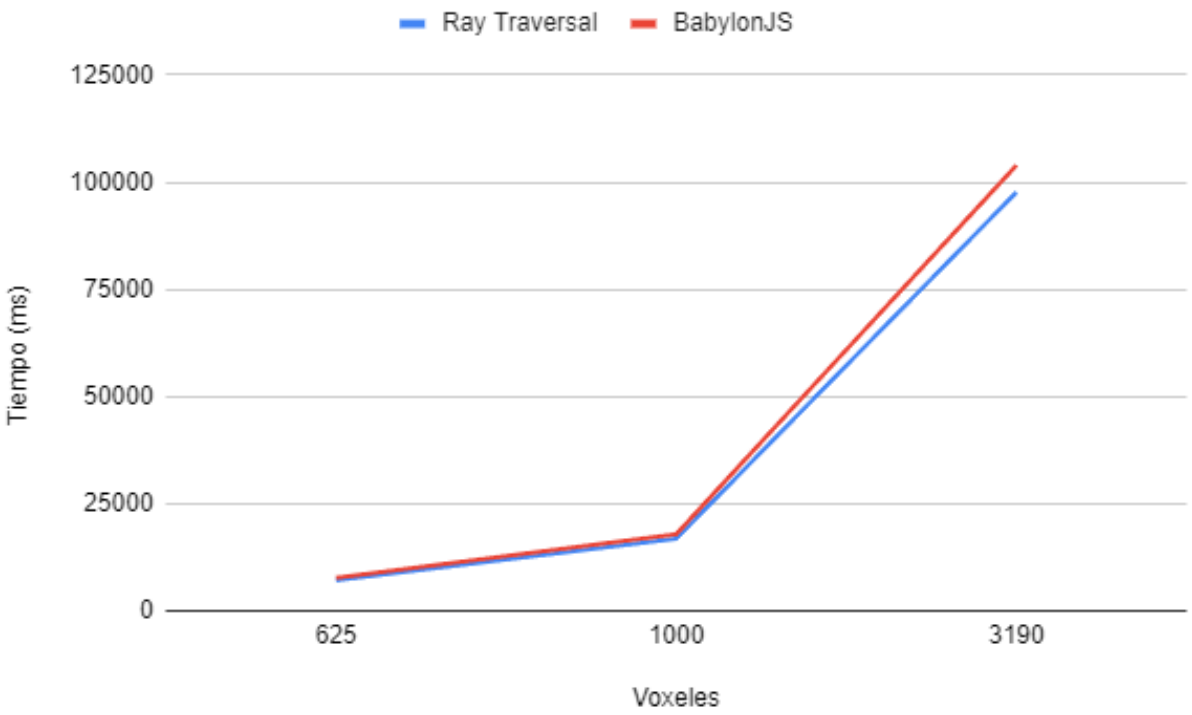


Ilustración 134: Gráfica de tiempos entre primeros métodos de intersección entre voxel y rayo (autor).

Como podemos observar la diferencia no es muy grande pero el algoritmo del Ray-Traversal se impone levemente ya que no debe recorrer tantos voxeles como el método por BabylonJS pero en cuanto a la simplicidad de implementación y cuando el número de

voxeles no es de gran tamaño se puede observar que el rendimiento del formato nativo de BabylonJS tiene un gran funcionamiento, sin embargo entre estos dos se opta por el Ray Traversal ya que no solo es más eficiente en tiempo sino también en memoria ya que el nativo de BabylonJS requiere de almacenar todas las Bounding boxes de los voxeles.

Por último y durante la realización de la experimentación llegamos a la conclusión de que el algoritmo Ray Traversal implementado podría simplificarse y encontrar una mejor solución ya que este algoritmo tiene en cuenta todas las posibles direcciones del rayo en los tres ejes, sin embargo, en nuestro problema los rayos siempre irán con la dirección (0,1,0), es decir hacia arriba en el eje Y, por este motivo es suficiente con encontrar el voxel de partida del rayo y en caso de intersecar ir incrementando el valor de voxel al vecino en Y para obtener los restantes. De esta forma el código final del algoritmo es tal que:

```
Procedimiento Ray-Traversal-Simplificado(rayo, voxelizacion, voxeles)
    Si rayoInsersectaVoxelizacion(rayo,voxelizacion) Entonces
        ray_start <- rayo.origin()

        current_X_index <- Max(1, (ray_start.X - minX) / tamVoxel)
        current_Z_index <- Max(1, (ray_start.Z - minZ) / tamVoxel)
        current_Y_index <- Max(1, (ray_start.Y - minY) / tamVoxel)
        Si rayo.direction().Y > 0 Entonces
            StepY <- 1
        Sino Si rayo.direction().Y < 0 Entonces
            StepY <- -1
        Sino
            StepY <- 0
        Fin Si
        Mientras 1 <= current_Y_index <= numVoxeles_Y Hacer
                voxeles <- voxelizacion[current_X_index][current_Y_index][current_Z_index]
                current_Y_index <- current_Y_index + stepY
    Sino
        Devolver Vacio
    Fin Si
Fin Procedimiento
```

Ilustración 135: Pseudocódigo Ray Traversal simplificado (autor).

La mejora en eficiencia no es excesivamente considerable pues el algoritmo Ray Traversal ya se encontraba bastante optimizado pero con estos cambios se consigue un código simplificado y optimizado para un problema con situaciones muy concretas de uso y que por tanto puede centrarse en casos aislados.

Olivos	Voxeles	Método	Tiempo (ms)
87	**625**	Ray Traversal	7123
		Simplificado	6825
168	**1000**	Ray Traversal	16823
		Simplificado	16033
269	**3190**	Ray Traversal	97619
		Simplificado	95132

Tabla 28: Análisis de tiempos entre Ray Traversal y método simplificado.

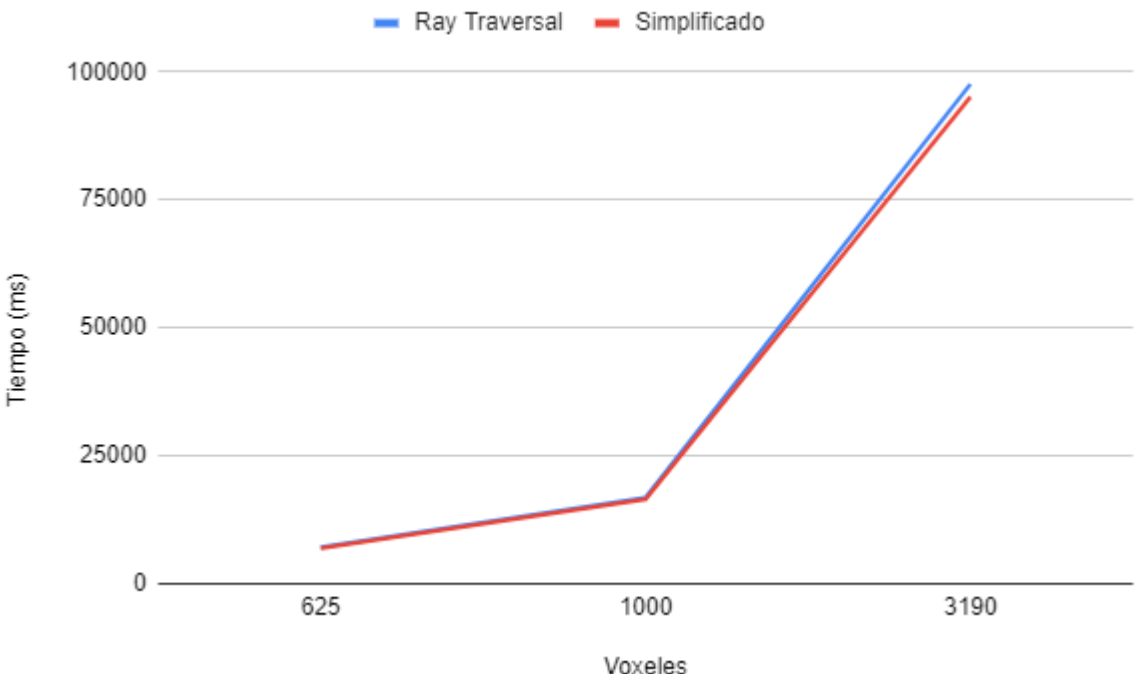


Ilustración 136: Análisis de tiempos entre el método simplificado y el Ray Traversal (autor).

5.6.4.2 Análisis de puntos

Como pudimos ver en la introducción de esta apartado el algoritmo principal implementado deja para el final el análisis de los puntos de cada voxel, será en esta sección donde se detalle los procedimientos y funcionalidades de esta parte del algoritmo.

Esta funcionalidad se desarrolla con la intención principal de obtener las posiciones reales de los olivos dentro de la nube de puntos, para esto en primer lugar nos basaremos en los voxeles intersecados con el fin de acotar el espacio de búsqueda para reducir complejidad.

Una vez disponemos de estos voxeles acotados recogeremos todos los puntos pertenecientes a estos y aplicaremos una primera selección. Esta se basa en una segunda reducción del conjunto a analizar, quedándonos exclusivamente con aquellos puntos cuyo color pertenezca a la gama del verde para escoger las zonas pertenecientes a posibles hojas de olivo. Esto se conseguirá mediante la obtención de aquellos puntos cuya componente Green en RGB sea ampliamente superior a ambas componentes restantes.

Con estos procesos realizados disponemos ya de todos los puntos que queremos analizar en nuestra muestra. El objetivo como se ha comentado es detectar olivos en la nube de puntos, una vez tenemos una primera muestra de las zonas con puntos de colores relativos al olivo es el momento de agrupar por zonas, para esto se ha desarrollado un algoritmo de clustering no supervisado que agrupa por clústeres bajo un umbral a determinar. Con esto se pretende obtener el número de olivos de la parcela, así como su posición y una posible estimación de su altura.

El algoritmo de clustering desarrollado es un algoritmo básico que se basa en el cálculo de distancias de cada nuevo punto con respecto a un centroide asociado a cada clúster de manera que estos irán no solo actualizándose sino también generando nuevos clústeres para conjuntos de datos muy separados con el objetivo final de obtener una estimación lo más afín posible a la realidad. Todo este proceso conjunto se podrá observar en el pseudocódigo mostrado en la Ilustración 137.

```
Procedimiento Olivos_Nube(voxeles, umbralDistancia)
    posiblesOlivos <- Array()
    Para i <- 1 Hasta tam(voxeles) Con Paso 1 Hacer
        Para x <- 1 Hasta tam(voxeles[i]) Con Paso 1 Hacer
            color <- voxeles[i][x].color
            Si color.g > color.b && color.g > color.r Entonces
                posiblesOlivos.push(voxeles[i][x])
            Fin Si
        Fin Para
    Fin Para

    clusters <- Array()
    centroides <- Array()
    clusters.push(new Array(posiblesOlivos[1]))
    centroides.push(posiblesOlivos[1])
    Para i <- 2 Hasta tam(posiblesOlivos) Hacer
        min <- Infinito
        indice <- -1
        Para x <- 1 Hasta tam(clusters) Hacer
            Si Distancia(posiblesOlivos[i], centroides[x]) < min Entonces
                min <- Distancia(posiblesOlivos[i], centroides[x])
                indice <- x
            Fin Si
        Fin Para
        Si min < umbralDistancia Entonces
            clusters[indice].push(posiblesOlivos[i])
            centroide <- recalcularCentroide(clusters[indice])
            centroides[indice] <- centroide
        Si no Entonces
            clusters.push(new Array(posiblesOlivos[i]))
            centroides.push(posiblesOlivos[i])
        Fin Si
    Fin Para
Fin Procedimiento
```

Ilustración 137: Pseudocódigo función para agrupamiento y detección de olivos en nubes de puntos (autor).

Una vez el código se prueba y funciona es necesario ajustar el parámetro del umbral de distancia a valores que permitan discernir correctamente entre olivos, para esto se realizaron múltiples ejecuciones del mismo y se observaron el número de clústeres resultantes, así como su representación para tratar de ir ajustando. Durante las pruebas se representaban los centroides con unas esferas generadas en la escena y con un diámetro igual al espacio que ocupa el clúster para poder observar mejor como se agrupan los puntos, las sucesivas pruebas realizadas son las siguientes.

En primer lugar, se decidió probar con un tamaño pequeño de distancia pensando en la posibilidad de encontrar varios olivos muy cercanos al ser parcelas dedicadas exclusivamente a esto. Por este motivo el primer valor probado fue un umbral de tamaño **dos** donde el resultado fue el siguiente:

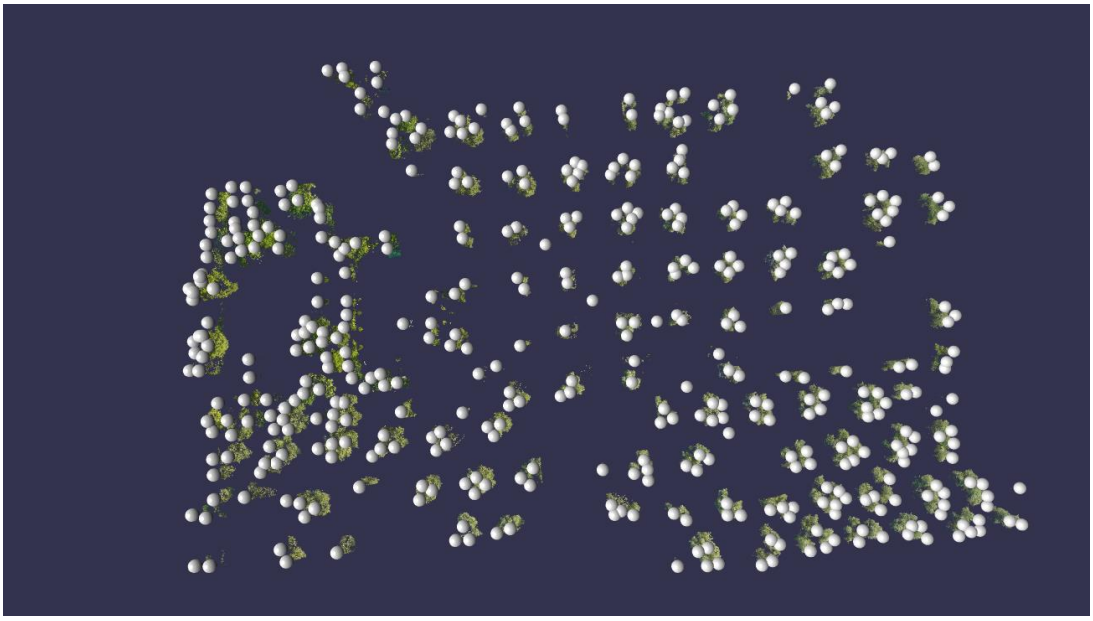


Ilustración 138: Proceso de clustering con umbral de distancia 2 (autor).

En la imagen se pueden observar los centroides de lo que serían los clústeres (esferas) y los puntos con tonos verdes extraídos de la nube. Como se puede observar el umbral resulta siendo pequeño pues para un olivo se presentan varios centroides por ende se prosiguió a aumentar el umbral a un tamaño de cinco.

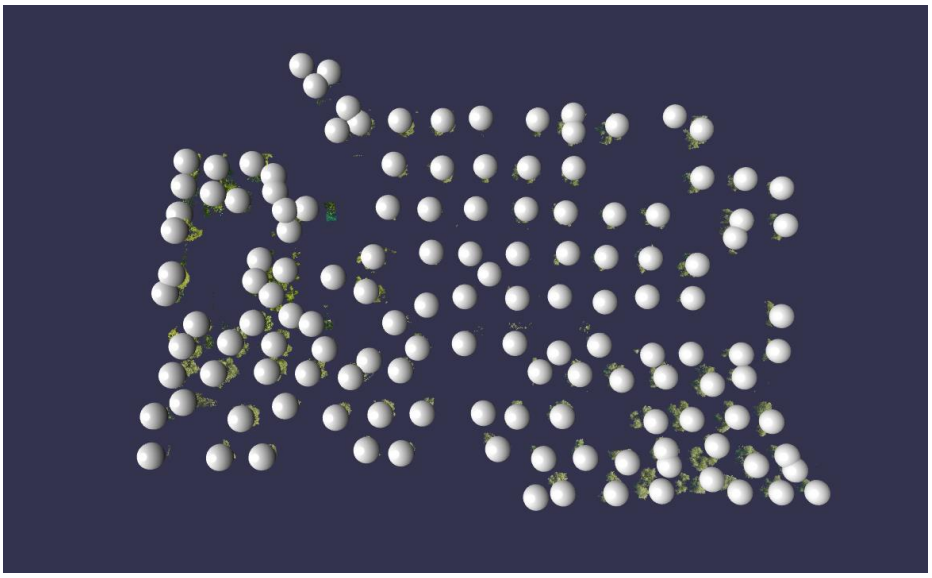


Ilustración 139: Proceso de clustering con umbral de distancia 5 (autor).

En este caso se puede observar una nube y unos centroides mucho más individualizados y colocados cada uno en su olivo o zona de puntos verdes aislada ya que cabe destacar que la segmentación por color puede albergar algunos fallos de clasificación al no estar perfeccionada.

Finalmente, este umbral fue el considerado como la mejor opción ya que clasificaba bastante correctamente los puntos. El desarrollo de esta funcionalidad se detiene en la identificación de las entidades, en trabajos futuros se podrá extraer información de la detección como máxima altura de un olivo, alturas medias, posiciones exactas…

Para conseguir finalmente el desarrollo adecuado de esta funcionalidad así como la inclinación de la textura que se vio en la Sección 5.5.4 era necesario alinear nube y textura una tarea que había generado problemas debido a la gran diferencia entre nube y textura que podemos observar en la siguiente ilustración, la paliación de esta problemática se verá en el siguiente apartado.

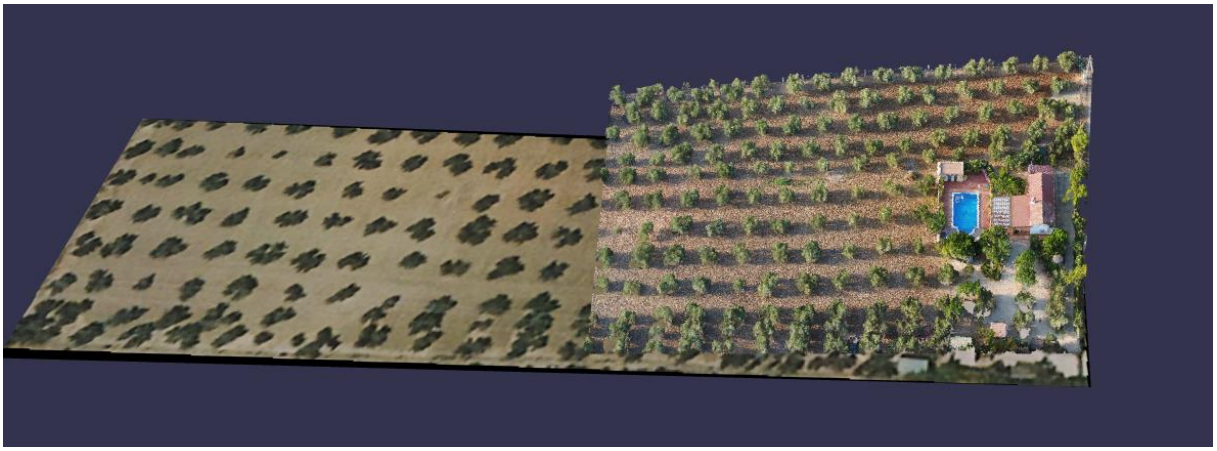


Ilustración 140: Combinación de textura y nube originales (autor).

5.6.4.3 Alineación nube - plano

Como pudimos ver en la Ilustración 140 la diferencia entre nube y textura es notoria, la textura ha sido creada con los vértices originales extraídos del SigPac para una proyección dada, por otro lado la nube es el resultado del vuelo de un dron sobre el terreno. En una primera visualización de la zona podemos encontrar varios problemas que habría que solucionar.

En primer lugar, la nube de puntos no es el resultado de un vuelo completo sobre una parcela exacta, esta dispone de menor información a lo largo de la parcela lo que nos indica que el vuelo se cortaba antes de llegar al final y dispone de mayor información a lo ancho por lo que sobrevolaba más allá de los límites de una única parcela al mismo tiempo

pasando por dos. Con el objetivo de paliar estos problemas la falta de información a lo largo no tendrá solución, pero el exceso a lo ancho se podrá solventar realizando un recorte a la nube.

Para conseguir esto emplearemos la herramienta de Cloud Compare que manejamos en secciones previas, en este caso emplearemos la herramienta **Segment** ubicada en el menú de navegación **Edit**, esta herramienta nos permite realizar una selección poligonal de la parcela para recortarla como vemos en la Ilustración 141, esta herramienta además nos permite quedarnos con el contenido interno o externo a la selección, en este caso no quedaremos con el contenido interno de la selección generada a mano y que busca ser lo más parecida posible a la textura obtenida del PNOA mediante el proceso de la Sección 5.3.5.3. Una vez disponemos del recorte y lo aplicamos volveremos a exportar la nube en el formato TXT para procesarlo de nuevo en la aplicación.

Ilustración 141: Selección poligonal del recorte de la nube (autor).

Una vez dentro de la aplicación ya tenemos dos modelos que disponen de una cantidad de información equiparada en un sentido lo que nos permite igualarlos. En primer lugar, es necesario realizar un ajuste a los puntos de la nube a lo ancho realizando un escalado para igualarlo a la textura. Una vez realizado este escalado aplicaremos uno proporcional a lo largo consiguiendo equiparar todo lo posible esta dimensión de la que no tenemos datos exactos al ser un nivel de información diferente.

Una vez escalada la nube en ambos sentidos solo faltaría desplazarla para ubicarla justo encima, para esto desplazaremos todos los puntos hacia la esquina coincidente aplicando una traslación constante a todos los puntos. El resultado final de estas

operaciones consigue un solapamiento de modelos no exacto pero sí bastante aceptable para las operaciones que se buscan realizar.

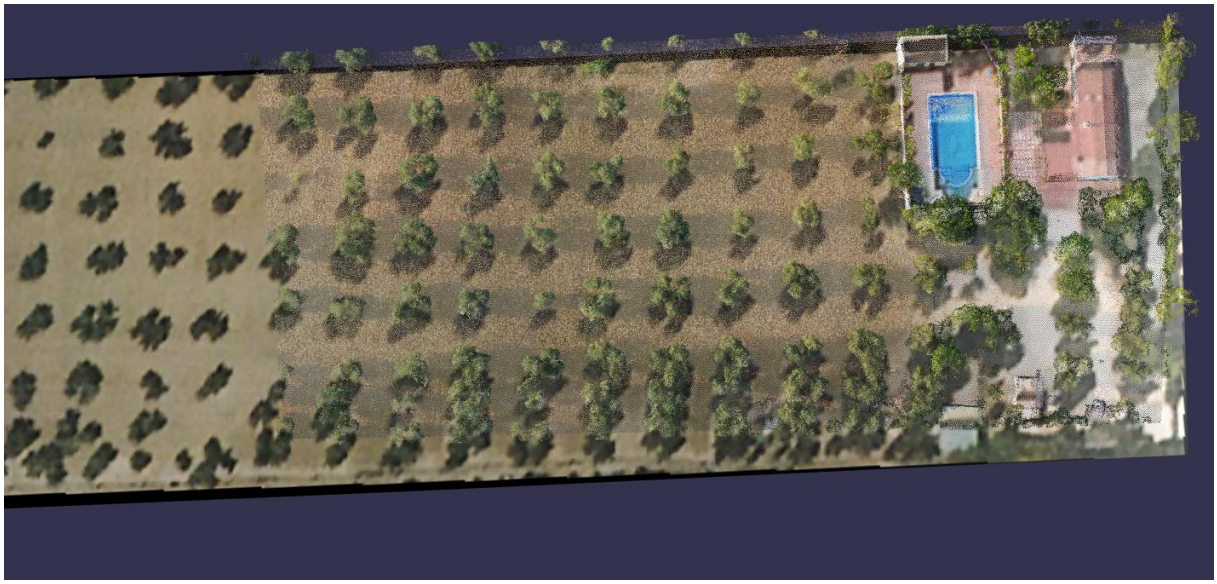


Ilustración 142: Superposición nube de puntos y textura (autor).

Finalmente cabe recalcar que lo ideal para el proyecto es contar con una nube de puntos coincidente exactamente con la parcela, pero esto es algo complicado de conseguir debido a los posibles fallos de dron o planes de vuelo con distintos objetivos, en caso de no contar con este tipo de nube ajustada a los datos SigPac se deberá tratar de manipular todo lo posible con procesos como los vistos en esta sección para conseguir ubicarla en la misma posición.

5.6.5 Gestión y desarrollo de perfiles

Una vez todas las funcionalidades principales del prototipo se encontraban finalizadas se decidió desarrollar una interfaz dedicada al control de datos importantes y delicados de la aplicación entre los que se ubican la información de los usuarios y las parcelas con el fin de poder establecer una herramienta visual y cómoda de gestión de los mismos. Esta zona se decidió ubicar en la sección del perfil de usuario para tener un acceso global.

Las funcionalidades a las que se podrá acceder según el nivel de usuario serán diferentes y seguirán un patrón lógico tal que:

- Los técnicos de parcelas no tendrán acceso en esta sección más que a su información personal, así como la posibilidad de editar esta.

- Los propietarios o administradores declarados de parcelas tendrán acceso al mismo nivel de información que un técnico, pero añadiéndole el permiso de acceso a su listado de parcelas (en caso de disponer de más de una) así como a la posibilidad de editar información de esta.

- Los administradores del sistema serán el escalafón más elevado de la cadena con acceso a toda la información del mismo, así como capacidad de editar, borrar o añadir esto se debe a que son los encargados de que todo funcione en todo momento y que el sistema se mantenga estable.

Una vez definido quien podrá acceder a que funcionalidades se desarrollarán estas y el proceso de implementación de las mismas. Como hemos comentado podemos dividir la sección perfil en tres funcionalidades.

5.6.5.1 Edición de datos personales

Esta página será accesible por todos los usuarios, siendo además la más simple, en esta será posible realizar un cambio de los datos personales almacenados en la base de datos, desde la foto de perfil hasta el nombre, usuario…

Para esto se emplea como se comentó en la sección de integración de la base de datos el lenguaje de programación PHP junto con AJAX y JavaScript básico.

Se realizarán por ende consultas SQL del tipo **Update** donde se actualizarán los campos que el usuario requiera.

La parte más compleja del sistema se encuentra en lo relativo al cambio de foto de perfil, por defecto a todos los usuarios se les asigna una foto estándar sin imagen cuya ruta se almacena en el campo correspondiente, sin embargo, cuando se realiza un cambio de esta no valdrá solo con actualizar el valor del nombre del archivo con el nuevo, sino que será necesario guardar el fichero de la nueva foto en el servidor o carpeta de la aplicación correspondiente.

Para conseguir esto se ha recurrido igualmente al lenguaje PHP pero esta vez ligado a un elemento **form** de HTML el cual al enviar la nueva foto realizara el siguiente proceso:

1. Se selecciona una nueva foto y se introduce en el input de tipo **file.**

2. Mediante un script se comprueba la extensión del fichero para asegurar que sea una imagen.

3. Se presiona el botón enviar de tipo **submit** que envía el formulario a la dirección correspondiente (el script php).

4. Se inicia el script php que recibe por URL el id del usuario correspondiente mediante la consulta al array $_GET.

5. Se consulta a la base de datos la foto de perfil del usuario

6. Si la foto no es la que se encontraba por defecto se elimina de los directorios

7. Se inserta la nueva foto en el sistema de directorios mediante la función **move_uploaded_file(nombreFichero, nuevaRuta).**

8. Se actualiza en la base de datos la ruta de la foto de perfil del usuario.

9. Finalmente se redirige la aplicación a la página del usuario.

Para facilitar el proceso completo se emplea la variable global de PHP $_FILES que es un array asociativo que almacena datos de los ficheros almacenados en un input de tipo file. Este array almacena la ruta temporal del fichero subido, el nombre del fichero…

5.6.5.2 Listado de parcelas

Esta sección es la más completa de las tres presentadas en esta tarea ya que conlleva edición y mantenimiento de instancias importantes en la base de datos, así como movimiento de ficheros entre usuario y servidor que es una operación delicada como hemos visto en el apartado anterior.

Esta sección tendrá diferentes funciones en función de si en ella entra un Administrador o un Propietario, si el que entra es el segundo se le presentarán sus parcelas y se le indica de los tres tipos de modelos posibles (ortofoto y nube de puntos) cuales tiene disponibles cada una de sus parcelas, en función de esto podrá editar mediante un formulario que le permitirá subir archivos para o bien introducir nuevos en la carpeta de recursos o para sustituir uno antiguo erróneo u obsoleto.

Además de esto se le presenta la oportunidad de en el caso de existir ficheros para alguno de los modelos poder descargarse una copia de las versiones disponibles entre los recursos.

En el caso de ser un Administrador en lugar de listar todas las parcelas se listarán únicamente aquellas que dispongan de algún tipo de modelo presente entre los recursos y

se permitirá realizar las mismas operaciones que se le permitían al propietario. En adición a esto además tendrá la posibilidad de directamente insertar modelos para una parcela que no disponía de ninguno y por ende no aparecía listada.

Una vez conocidas las funcionalidades presentes en esta ventana se debe explicar a continuación como se han desarrollado las mismas.

En primer lugar, la subida o edición de modelos se ha resuelto aplicando el mismo proceso que se aplicó en el cambio de foto. En este caso tenemos un elemento **form** de HTML que contiene dentro de sí tres inputs de tipo **file,** cuando el usuario clique la opción de guardar los datos se realiza una llamada a un script PHP en el cual se aplica en dos ocasiones la función **move_upload_file,** esta función tiene la ventaja de que en caso que el valor del archivo este vacío no se realizara el proceso permitiendo aunar los 2 modelos en un mismo script y no tener por qué subir siempre todos.

Debido a la inserción de esta funcionalidad que trabaja con ficheros de gran dimensionalidad será necesario alterar el fichero de configuración php.ini que establece valores máximos de transacción de ficheros en valores muy bajos. Se establece un umbral de 100 MB al atributo **upload_max_filesize** y **post_max_size.**

Por otro lado, para ofrecer la posibilidad al usuario de la descarga se ha empleado el lenguaje JavaScript principalmente. Primero se realiza una consulta a la base de datos para determinar que modelos se deben ofrecer para ser descargados y a estos se le aplica una función onClick en la cual al clicar se genera un elemento **a** de HTML al que se le aplica una dirección correspondiente a la ubicación del archivo a descargar. Finalmente se simula un clic a este botón y se elimina para que no interfiera en el aspecto visual.

Por último, las operaciones restantes serán de edición básica de datos que se realiza con Ajax y consulta SQL de tipo Update o Insert para un caso que requiera de añadir un nuevo modelo.

5.6.5.3 Listado de usuarios

Esta acción es exclusiva de administradores y está pensada para poder gestionar a todos los usuarios fácilmente. En esta sección se le otorgan tres posibilidades a los administradores como son añadir un usuario con un formulario en blanco a rellenar, editar valores de un usuario con un formulario rellenado con los valores actuales y eliminar a un usuario directamente de la aplicación.

El proceso para realizar todas estas acciones es muy básico pues consisten exclusivamente en consultas SQL a la base de datos mediante PHP. Según la acción a emplear esta se realiza mediante un Update, Insert o Delete.

Ante la prevención de un posible desarrollo futuro de la aplicación con un número elevado de usuarios y parcelas a analizar se ha desarrollado una interfaz dinámica que no muestra todos los usuarios de golpe si no que trabaja con AJAX para mostrar los valores de diez en diez y poder ir hacia delante o atrás conforme convenga.

El diseño de todas las interfaces y formularios se mostró en los diagramas realizados previamente y el resultado final podrá verse en el recorrido final realizado en la última iteración.

5.6.6 Resumen trabajo realizado

Finalizada la quinta iteración esta fue utilizada para aplicar parte de las tareas que se plantearon como soluciones a otras tareas previas. En la versión nueva entregada a los usuarios estos podían visualizar la voxelización realizada así como realizar nuevos recortes de manera más optimizada.

Se dispuso también de una interfaz de usuario para un mejor manejo en la escena que los usuarios agradecieron y vieron bastante bien integrada para facilitar los procesos. En cuanto al proceso de detección se observó una gran mejora en la detección lo cual satisfacía las necesidades actuales. Todo esto gracias también al proceso de integración de la información de nube y plano texturizado que permitirá subsanar ciertos errores.

Por último se les dio paso a acceder al nuevo sistema de perfiles y usuarios que permitían el manejo y edición de la información de forma más accesible para cualquier usuario sin necesidad de entrar a fondo en el sistema.

5.7 Sexta Iteración

Esta iteración se corresponde con la última del desarrollo de la aplicación, por ende, su trabajo se encuentra ligado al refinamiento de la aplicación, así como a su puesto a punto para poder ser utilizada y entregada a los usuarios correspondientes. En esta se realiza una limpieza de código para obtener pequeñas mejoras de rendimiento, se introducirán aspectos de conexión con el usuario para facilitar su navegación e introducirán una capa más de información mediante la introducción de datos gráficos.

Finalmente, el trabajo concluirá con la integración de la aplicación en un servidor web externo con el objetivo de que esta se encuentre siempre accesible.

5.7.1 Creación de representaciones gráficas

Para la representación de datos históricos se ha empleado el uso de gráficas en nuestra aplicación web, estas se encuentran disponibles para cada parcela de forma individualizada y representan los tipos de información que son de utilidad ya que pueden llegar a explicar mediante su análisis el rendimiento del terreno concreto.

Para su desarrollo se ha empleado el uso de las tablas definidas en la sección como de ámbito temporal ya que una de sus claves primarias es una fecha. Además como herramienta se ha usado la librería de JavaScript ChartJS [27] que nos permite desarrollar una gran gama de gráficas de todo tipo de forma asequible.

Para el desarrollo de la página han colocado tantos elementos **canvas** HTML como gráficas queremos colocar en la página y se han ejecutado los scripts básicos de creación de las mismas. En el prototipo inicial se la aplicación se han generado cuatro gráficas de líneas con los datos divididos temporalmente por semanas que es como se almacenarán en la base de datos.

Una vez se carga la página se realizarán cuatro llamadas a la creación de gráficas donde cada una de esta consulta a la base de datos las fechas y datos correspondientes para la parcela y la tabla deseadas. Una vez disponemos de los datos presentamos un número limitado de estos por pantalla, con el fin de otorgar libertad al usuario se desarrolla un elemento **input range** de HTML que permite variar el número de semanas a mostrar en los datos hasta un mínimo de 4. Las instancias que se irán mostrando siempre serán las últimas.

En cuanto a la configuración de las gráficas los parámetros a ajustar para cada una de estas en el código son:

- Labels: Son los valores del eje X, en nuestro caso fechas.
- Datasets: Son los datos a corresponder con los labels.
    - Label: Nombre del dataset.
    - Data: Vector con los datos asociados.
- Type: Tipo de gráfica puede ser de tipo ‘line’, ‘bar’…

- Options: Opciones extra para el ajuste de las gráficas.
    - Scales: Se corresponde con los atributos manejables de los ejes.
        - Type: El tipo de cada eje en el caso del X será ‘time’.
        - Time: En el caso del tipo time se determina la unidad ‘day’, ‘week’… así como el rango de salto entre valores del eje.

5.7.1.1 Edición históricos

Una vez tenemos los datos históricos presentados e integrados en la aplicación se debe integrar un sistema que permita a los usuarios correspondientes editar este tipo de datos. Los usuarios que podrán disponer de esta funcionalidad serán los Administradores para todas las parcelas con este tipo de datos y los Propietarios para sus propias parcelas.

Esta sección se desarrolla en la página donde se editaban y presentaban los modelos descritos en la Sección 5.6.5.2 , en este caso se añadirá a la tabla de parcela un botón para mostrar los históricos de cada parcela individualmente. Al entrar en este se nos muestra una gráfica con los datos históricos más recientes de la parcela y se da la opción de movernos por la tabla y que la gráfica vaya mostrando los datos (Ilustración 143). Además, se introducen dos botones internos de edición y adición de los valores correspondientes permitiendo alterar los resultados de estos cambiando también sus valores en la base de datos. Por último, se introduce un sistema de carga de ficheros por CSV para poder insertar una gran cantidad de datos de un solo golpe. Para la realización de esto en primer lugar se ofrece la posibilidad de descargar el estándar de CSV que maneja la aplicación, en caso de disponer de este solo será necesario rellenarlo y subirlos para cargar los datos.

Una vez este se sube a la página al recibir exclusivamente ficheros con un formato concreto este será procesado recibiendo las fechas y los valores para estas y almacenándolos en la base de datos mediante el uso de AJAX y PHP de forma que más adelante se puedan mostrar en las gráficas y tablas de la aplicación.

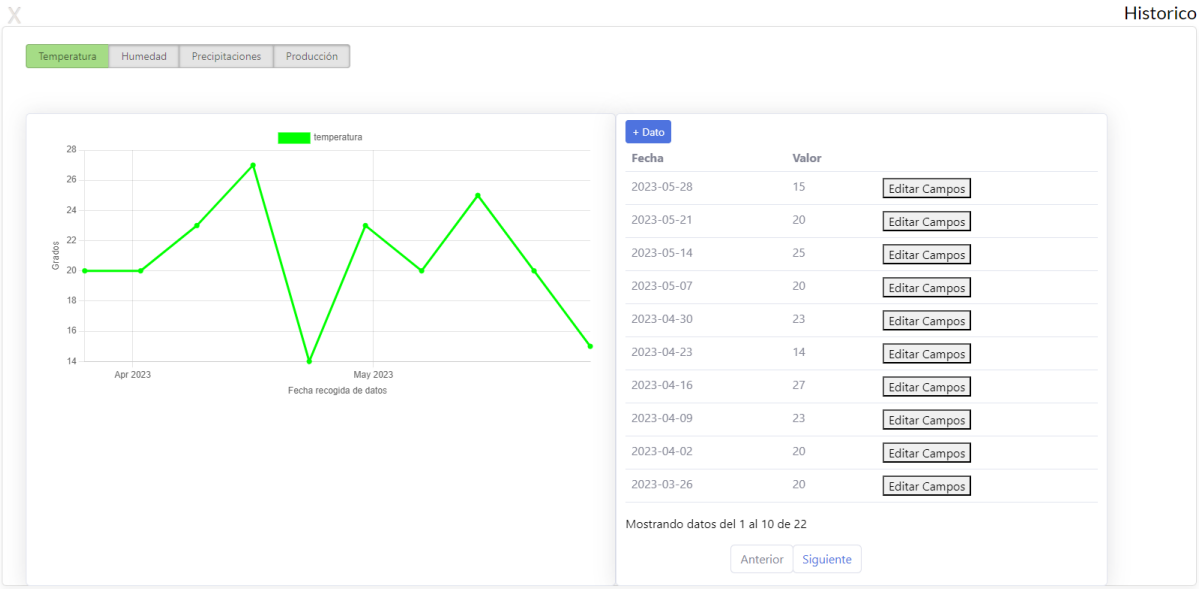


Ilustración 143: Imagen de la ventana de edición de datos históricos (autor).

5.7.2 Mejoras en la exposición al usuario

Una vez la interfaz se encuentra terminada se deben realizar operación e integración de código con la vista puesta en tratar de facilitar la experiencia al usuario, informándole en situaciones dudosas del porqué de cada cosa o que procesos se encuentran corriendo por detrás en el servidor.

El desarrollo de este apartado será especialmente importante en su uso en dispositivos móviles o de menor capacidad de rendimiento donde ciertos procesos pueden llegar a parecer bloqueantes o sin avance por ende es necesario establecer un mínimo de comunicación con el usuario.

Para conseguir esto se ha optado por la implementación de la muestra en pantalla de avisos de control al usuario donde se indica la situación de éxito, fracaso o espera en la que se encuentra una operación del sistema.

Las principales operaciones para las que se ha desarrollado esto son la escena donde ciertas operaciones pueden llevar más tiempo del deseado como la carga de nubes de puntos y la interfaz de control de datos de la aplicación donde se debe notificar al usuario del éxito o fracaso de los cambios realizados.

A continuación, se muestran algunos de los tipos de mensajes que el usuario podrá observar, estos se han desarrollado puramente en HTML y se juega con su visibilidad en la aplicación mediante el uso de JavaScript.

Ilustración 145: Mensaje informativo estado de la escena (autor).

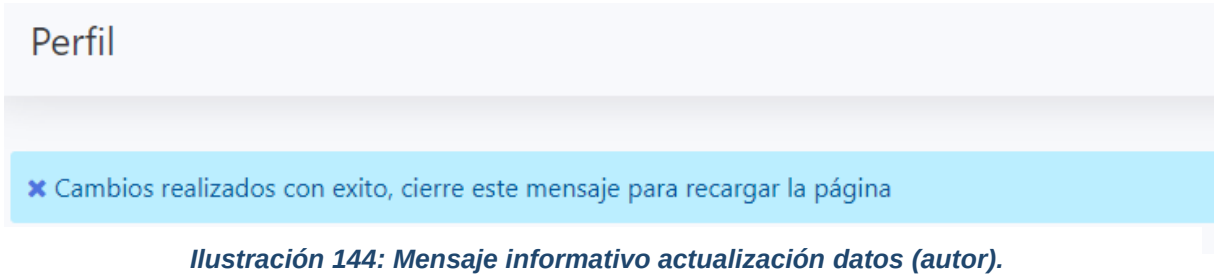


Ilustración 144: Mensaje informativo actualización datos (autor).

5.7.3 Finalización de la interfaz de usuario

Una vez el código y los recursos del proyecto se encuentran finalizados es el momento de realizar las comprobaciones finales de la interfaz de usuario y su navegación, para esto se realizarán múltiples navegaciones por la aplicación con el fin de probar todas las funcionalidades posibles para todos los casos existentes y asegurar el éxito de todas ellas.

Con el fin de mostrar la navegación e interfaces finales de la aplicación se procederá a explicar cada una de las páginas de las que se compone la aplicación web final y las funcionalidades internas a las mismas. No se añadirán capturas completas del código completo de cada una de las páginas ya que a lo largo del desarrollo se ha mostrado el código importante y el resto podrá encontrarse en un repositorio GitHub[29] donde se encuentra almacenado.

5.7.3.1 Inicio de sesión

Para el inicio de sesión se optó por una interfaz sencilla con un formulario central donde solicitar al usuario los datos que posteriormente se validarán. Con un fin más decorativo de opto por insertar de fondo una escena tridimensional como antesala a una visión inicial de lo que el usuario podrá disponer.

Como el renderizado de una escena cualquiera es costoso y esta página debe ser rápida y sencilla se optó por una opción con mayor rendimiento la cuál consistía en generar la primera vez una escena costosa, descargarla completa con la función de BabylonJS

[29] Repositorio del proyecto: https://github.com/PabloLHo/Prototipo-Aplicacion-WebGL

`BABYLON.GLTF2Export.GLBAsync` de esta forma cada vez que se abra la página solo será necesario cargar un fichero para observar toda la escena, esta opción no es válida en la escena final ya que perderíamos acceso los elementos individuales de la misma.

La interfaz final luce tal que:

Ilustración 146: Interfaz final Inicio sesión (autor).

5.7.3.2 Home

Esta página se ubica en nuestra jerarquía del sistema como la página de bienvenida sin embargo nada más entrar al usuario se le presenta la primera gran funcionalidad del sistema como es el mapa interactivo el cual determina qué acción o parcela quiere observar y / o analizar.

Los objetivos planteados alrededor de esta interfaz eran básicos y fáciles de conseguir pues solo se requería de un mapa que hospedara una serie de capas vectoriales con el fin de poder acceder a parcelas individuales. Debido a la rápida consecución de esto se decidió dotar al mismo de una mayor serie de controles para facilitar la experiencia del usuario, el resultado de esto lo pudimos ver en la Ilustración 111. A continuación, se muestra el resultado final de la integración de este en la página.

Ilustración 147: Parte superior página de bienvenida (autor).

En adición a todo esto y con el propósito de poder describir brevemente el sistema y sus capacidades en la parte inferior de la página se muestra información referente al desarrollo, capacidades y en futuro posibles colaboradores y equipo de administración.

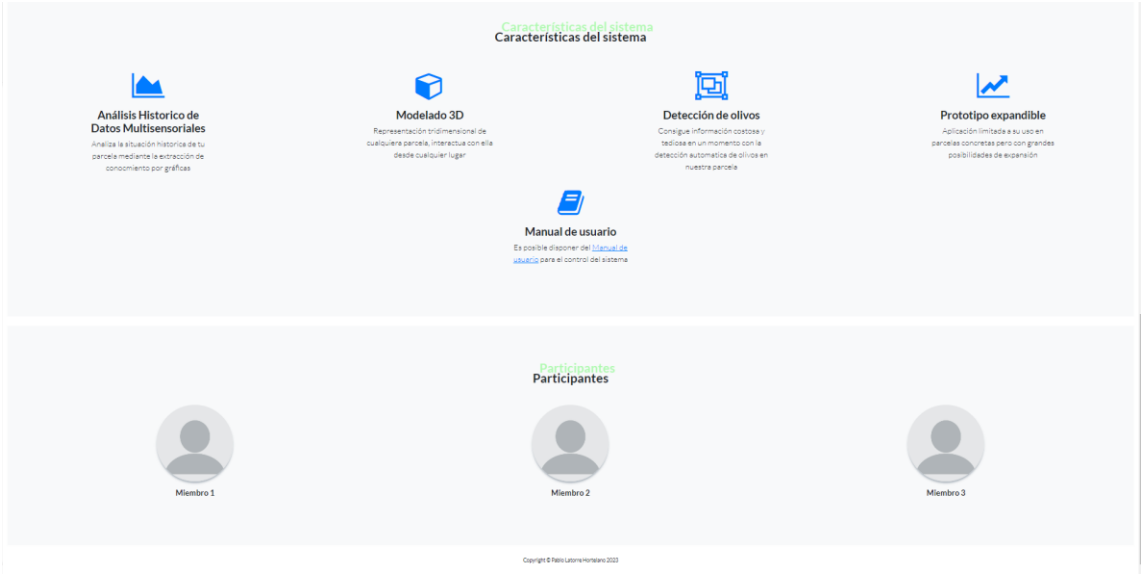


Ilustración 148: Parte inferior página bienvenida (autor).

5.7.3.3 Información parcela

Esta página se centra en la muestra de toda la información no espacial relativa a cada una de las parcelas con el fin de poder obtener información no solo de la interacción con el modelo sino también del análisis de la información ya existente. La página se divide en 3 ventanas como son los datos base de la parcela, los datos en formato SigPac o divididos por zonas y los datos históricos presentados mediante gráficos. El resultado final de estas interfaces el siguiente:

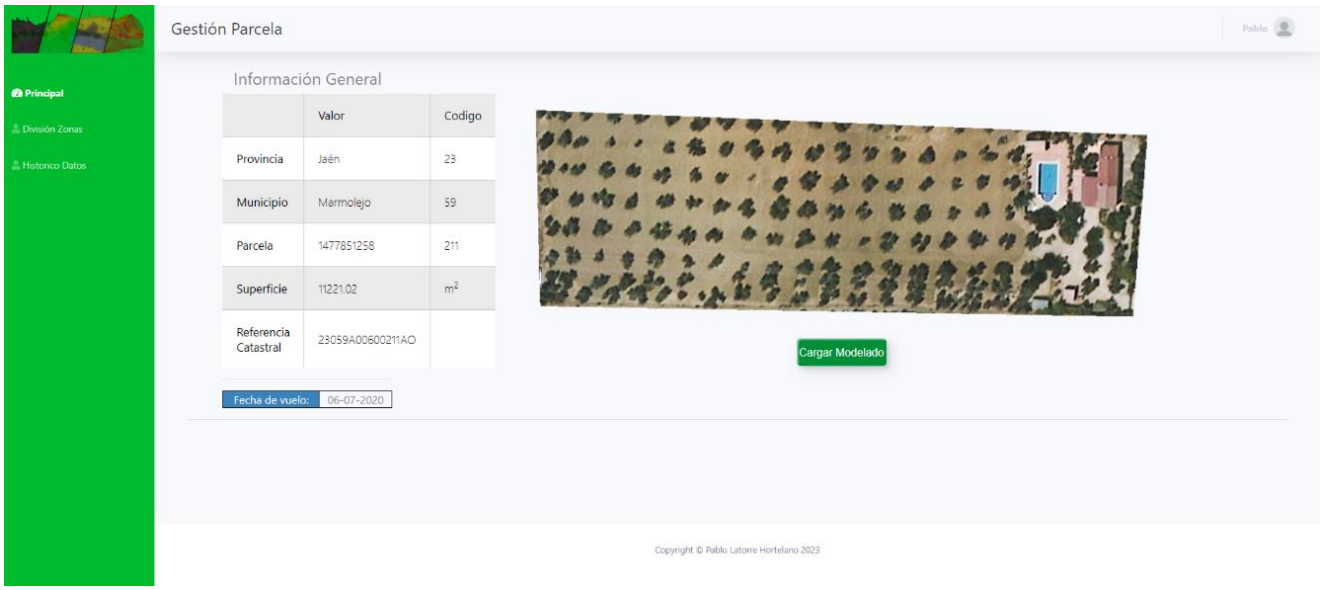


Ilustración 149: Página de datos base de una parcela (autor).

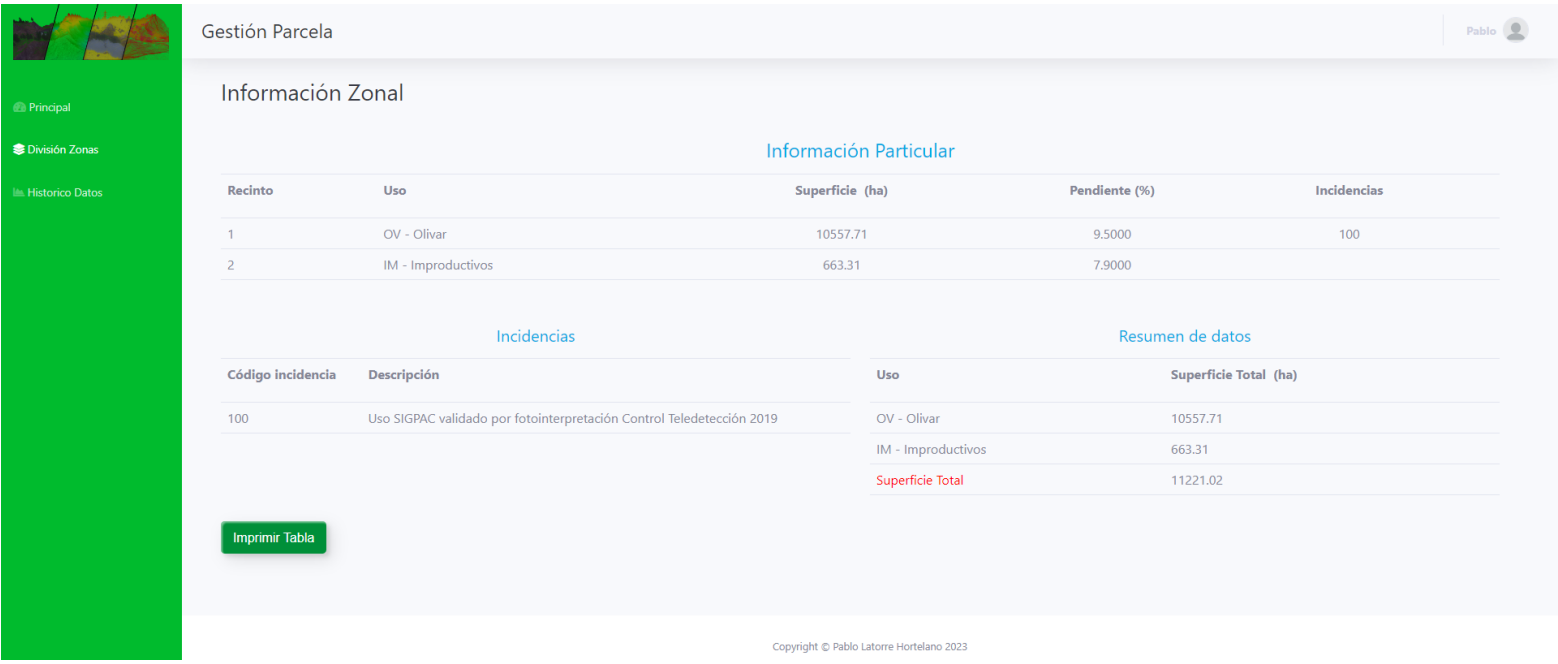


Ilustración 150: Sección datos zonales de una parcela (autor).

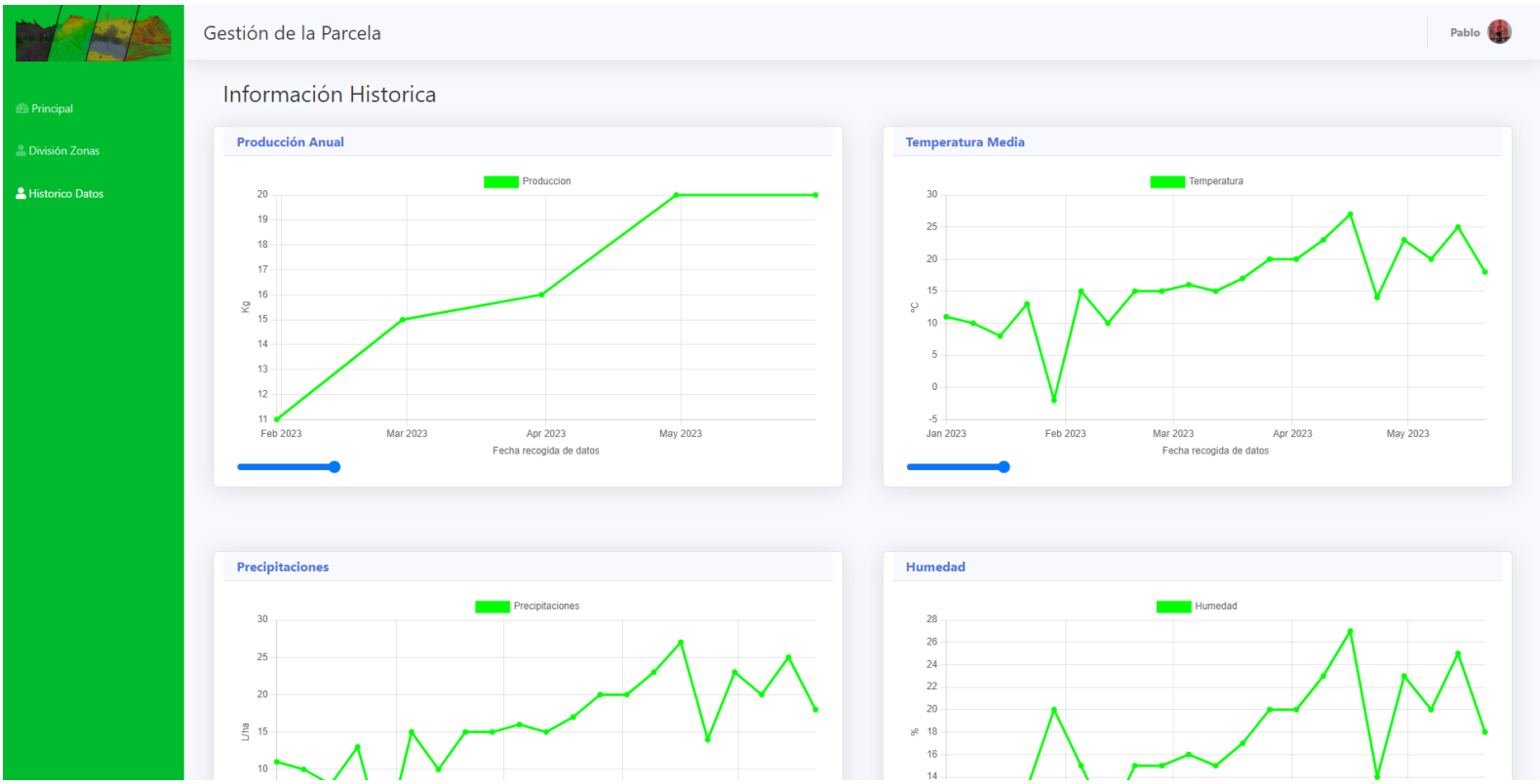


Ilustración 151: Sección datos históricos de una parcela (autor).

5.7.3.4 Escena

La página de la escena es uno de los focos principales e interfaces más importantes de la aplicación, en esta se encuentran el groso más importante de las funcionalidades del sistema.

Los objetivos que se plantearon definían que la escena debía ser clara y fácil de aprender para el usuario, debía disponer de un rendimiento aceptable en múltiples tipos de dispositivos y debía tener un manejo dinámico y eficaz.

Todo esto se consiguió con la integración de una interfaz de usuario clásica con las opciones básicas para poder acceder a toda la escena desde ella, el rendimiento se optimizo mediante la reducción de los modelos más complejos (nubes de puntos) con la intención de que pudiera funcionar en cualquier sistema.

Finalmente, se le añadieron todas las funcionalidades de las que goza la misma como son recorte de nubes de puntos, detección de entidades y múltiples modelos para su visualización. A continuación, se mostrarán ilustraciones de múltiples posibles estados de la interfaz final.

Ilustración 153: Escena con la nube recién cargada (autor).

Ilustración 152: Selección activa (autor).

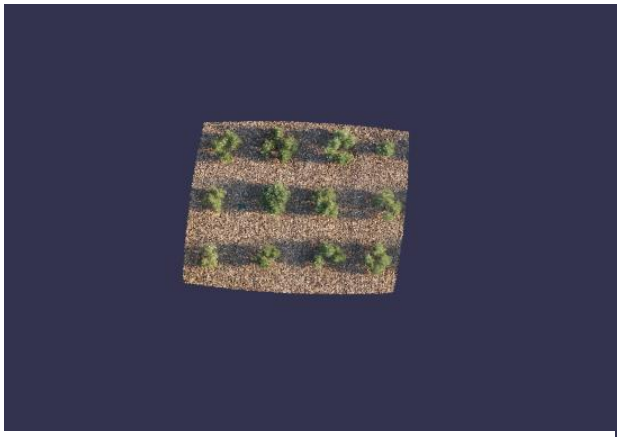


Ilustración 156: Recorte de la selección (autor).

Ilustración 154: Identificación de olivos en textura (autor).

Ilustración 155: Edición de la detección (autor).

5.7.3.5 Perfil

Esta página en los inicios del proyecto no se planteaba como una necesidad del mismo, sin embargo, su desarrollo ha sido de gran utilidad para facilitar la gestión de la información.

Lo que se esperaba de esta, por ende, estaba claro y era poder editar la base de datos y el sistema de archivos fácilmente con una interfaz clara y concisa. Esto se consiguió con la división de la misma en tres secciones como fueron datos personales, modelos y usuarios.

El resultado final como podremos ver a continuación se conforma de 3 ventanas principales junto con varios formularios o ventanas de diálogo que nos permiten acceder a las funcionalidades deseadas.

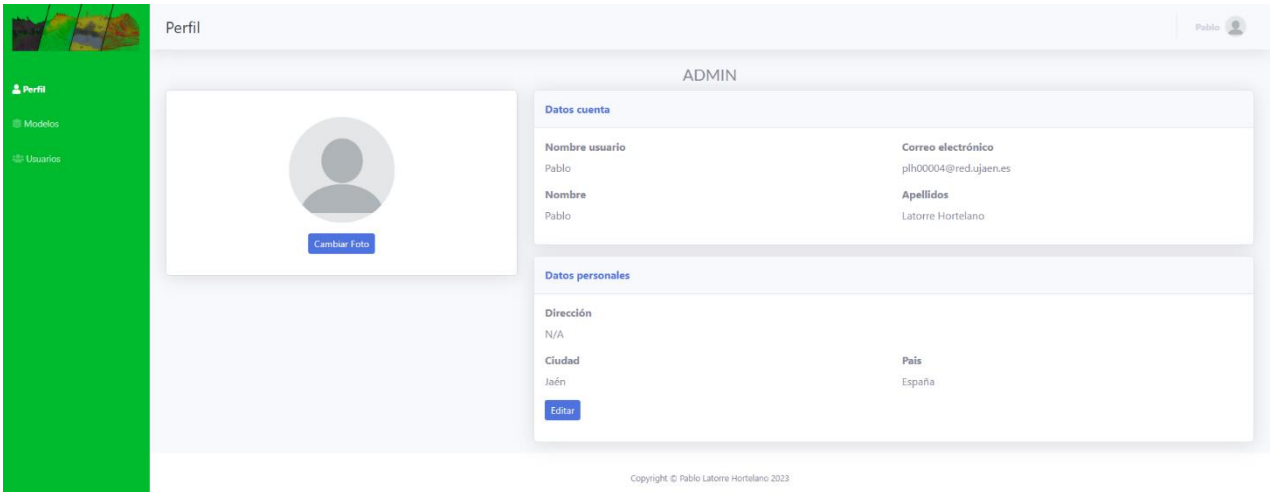


Ilustración 158: Página para muestra y edición de los datos personales (autor).

Ilustración 157: Página de perfil con el listado de las parcelas (autor).

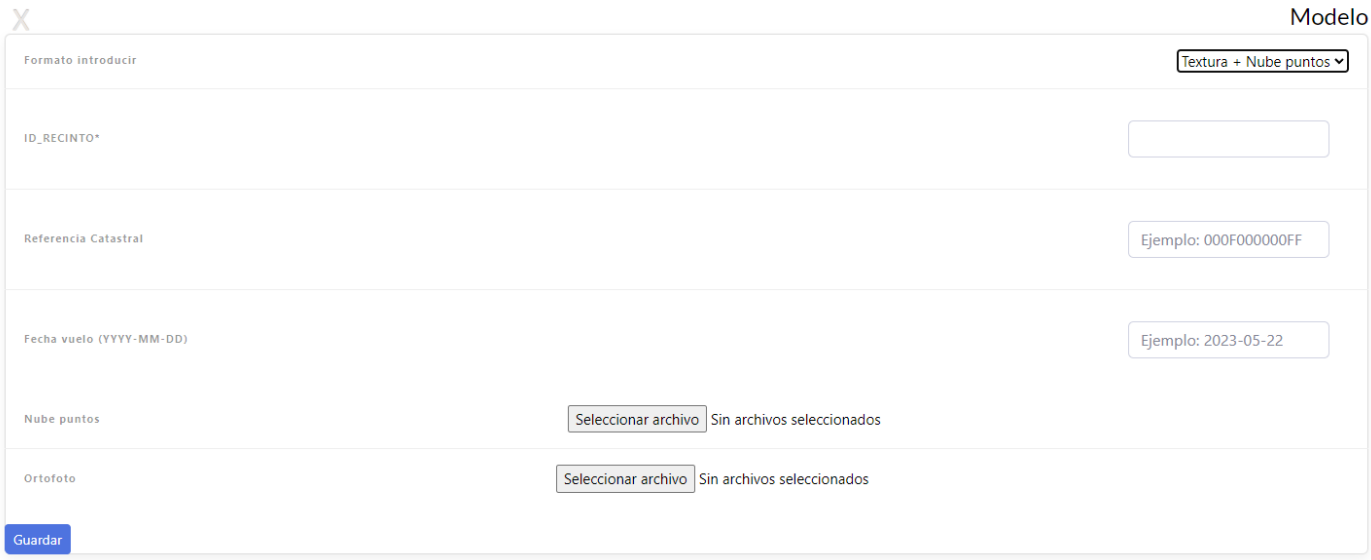


Ilustración 160: Formulario para la adición de modelos a una parcela que no dispone de ninguno (autor).

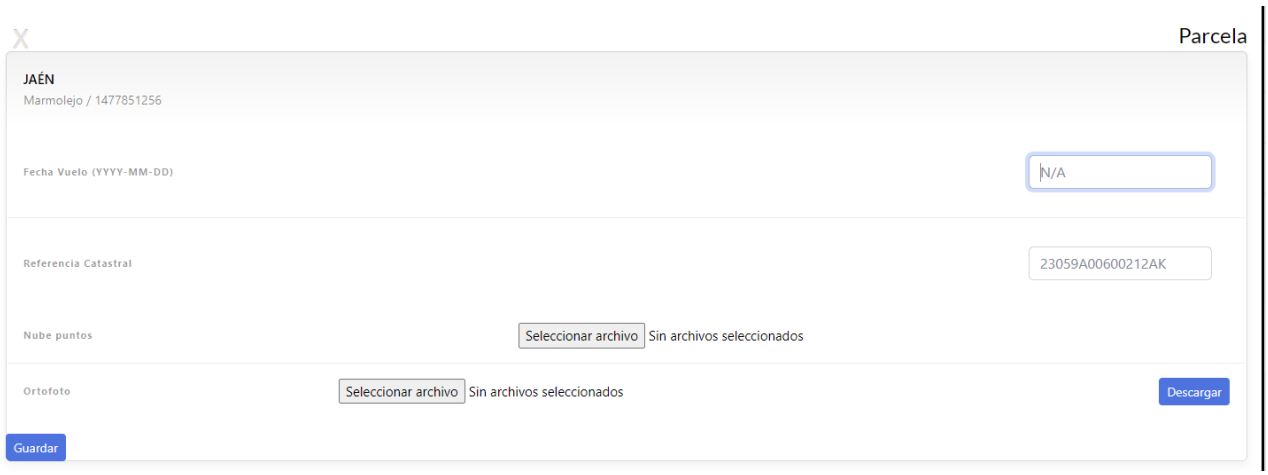


Ilustración 159: Formulario para edición modelos de una parcela (autor).

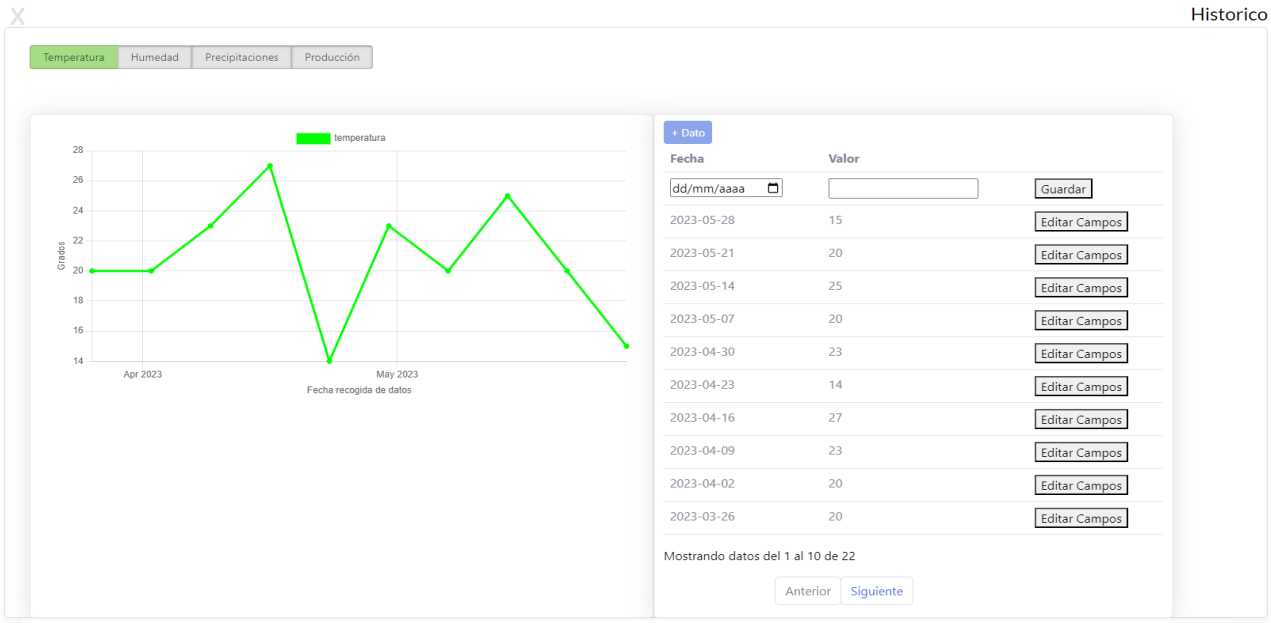


Ilustración 161: Ventana asociada a la edición y adición de históricos (autor).

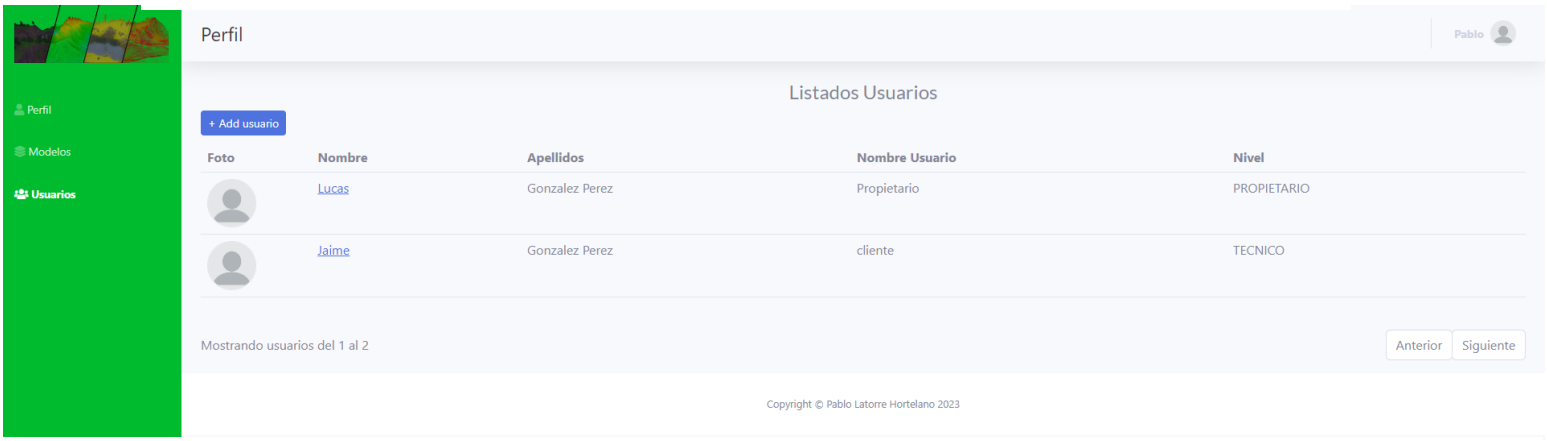


Ilustración 163: Página perfil con los listados de usuarios (autor).

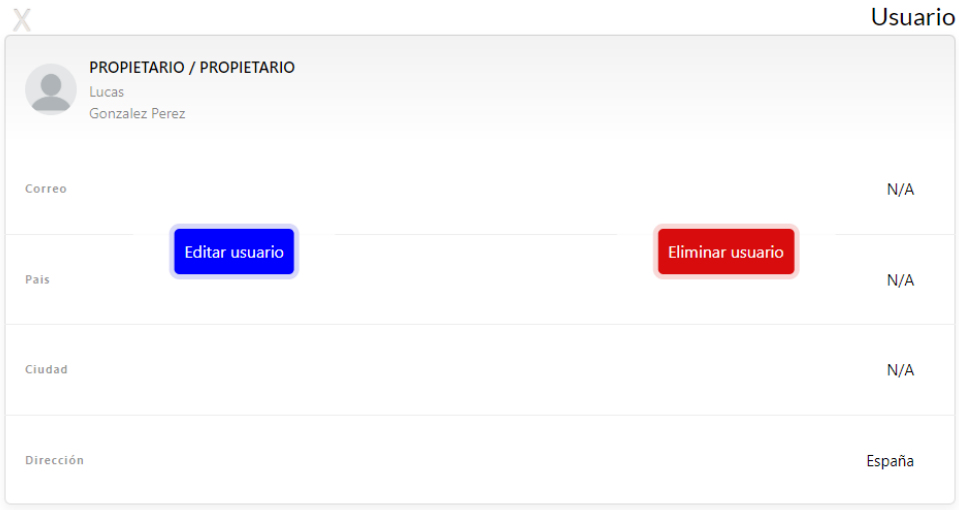


Ilustración 162: Ventana de dialogo para visualización datos de un usuario (autor).

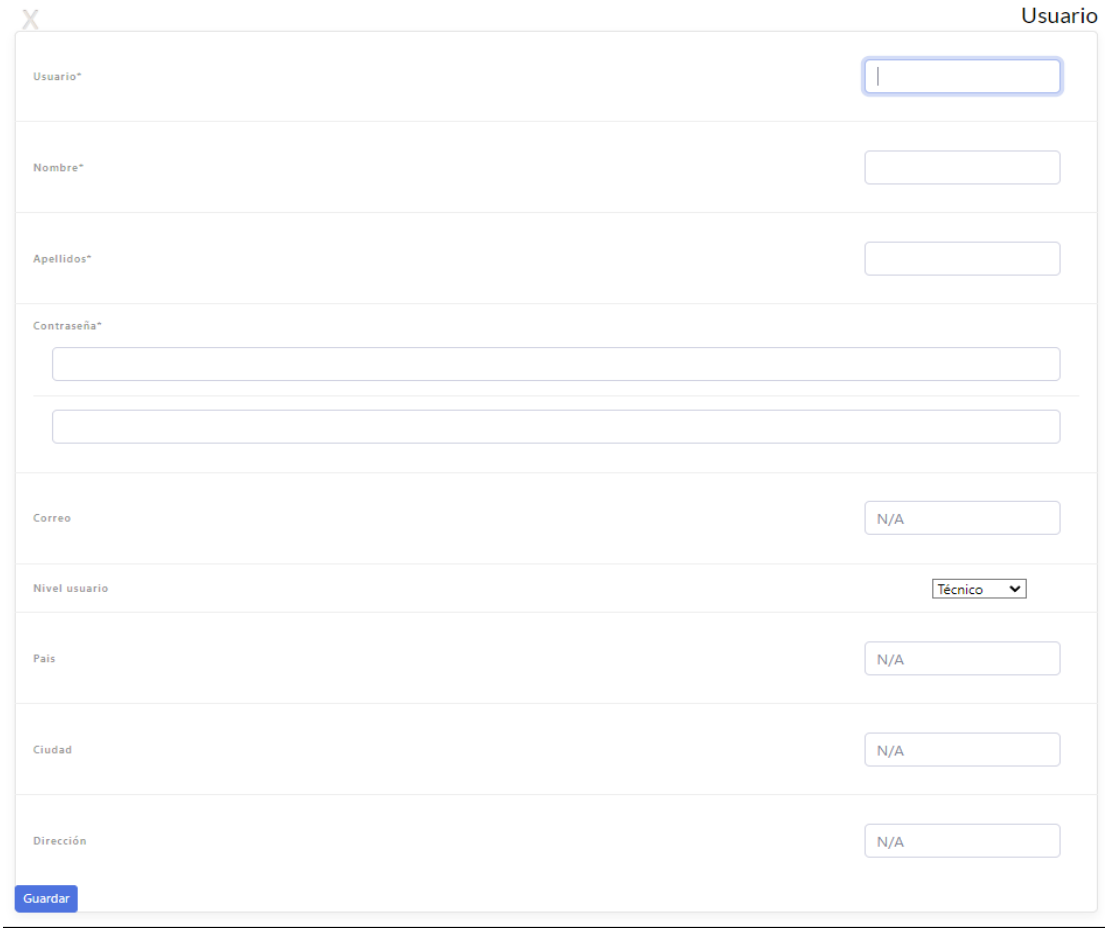


Ilustración 164: Formulario de adición de un nuevo usuario al sistema (autor).

5.7.4 Integración aplicación en un servidor web

Con todo el trabajo ya finalizado solo falta la tarea de otorgar acceso a los usuarios a través de la red y en cualquier momento que se requiera de su uso, para esto se optó por integrar la aplicación en una máquina virtual de un dispositivo que se encuentra siempre conectado.

La máquina virtual que se creó disponía de las siguientes características:

- Sistema operativo: Debian 11
- 2 CPU
- 2 GB de RAM

Una vez se dispone de esta creada trabajaremos con esta vía SSH mediante comandos por el terminal. Para disponer de acceso a esta vía red será necesario instalar apache, así como un sistema de gestión de base de datos para poder almacenar la nuestra.

La instalación de apache en el terminal de Linux no es una tarea compleja se debe ejecutar en el terminal el comando ***sudo apt install apache2***, en ocasiones además de esto es recomendable previamente ejecutar el comando ***sudo apt Update*** para actualizar los paquetes locales. Con ambas ejecuciones terminadas y por temas de seguridad será necesario ajustar el firewall permitiendo el acceso por apache con el comando ***sudo ufw allow 'Apache'.***

Una vez realizados estos pasos disponemos de tres comandos básicos para controlar el servidor como son:

- sudo systemctl stop apache2: Detiene el servidor web.
- sudo systemctl start apache2: Inicia el servidor web.
- sudo systemctl restart apache2: Reinicia el servidor web.

Una vez terminado el proceso de instalación de apache tendremos que introducir el código y todos los recursos en un directorio que creemos para acceder desde este vía URL. Lo podremos encontrar en la ruta **var/www/<Directorio creado>** una vez estemos en el directorio podremos copiar todos los archivos necesarios.

El siguiente paso es instalar el sistema de gestión de base de datos, en nuestro caso el elegido es MariaDB, para instalarlo deberemos ejecutar el comando ***sudo apt install mariadb-server.*** Una vez instalado en el sistema deberemos proceder con la importación de la base de datos. Previamente hemos exportado esta desde nuestro equipo local gracias a la funcionalidad de phpMyAdmin que nos deja conseguirlo con facilidad.

Para conseguir importarla deberemos ejecutar la siguiente secuencia de comandos

MariaDB -u root -p // Nos permite abrir la base de datos

CREATE DATABASE nombre //Crea una nueva base de datos en el sistema

MariaDB -u usuario -p nombre < bbdd.sql //Esta es la instrucción de importación donde se define el usuario que lo realiza, el nombre de la base de datos y el path donde se ubica el fichero SQL a importar.

Con estas instrucciones la base de datos se debe encontrar disponible y simplemente tenemos que tener cuidado de que las credenciales de acceso a esta coincidan con las del script PHP de conexión.

Llegados a este punto disponemos de los recursos y la base de datos por lo que exclusivamente necesitamos de la instalación de Geoserver y algún cambio en los permisos de directorios concretos.

Como se comentó en fases previas del desarrollo el servidor tendrá funcionalidades en las que tendrá que subir fichero y almacenarlos, para esto es necesario dotar al usuario de ciertos permisos de escritura en los directorios donde se procederá a almacenar la información. Para conseguir esto se utilizó el comando ***chmod*** de Linux para editar los permisos de las carpetas.

Los directorios a editar son los que almacenan modelos y las fotos de usuario, por defecto estos directorios tienen los permisos 755 donde el creador tiene permisos de lectura, escritura y ejecución y el resto de usuarios disponen de permisos de lectura y ejecución. Para conseguir que las funcionalidades funcionen correctamente se ejecutara por tanto el comando con la siguiente estructura:

Chmod +757 <path directorio>

Con estos cambios realizados y tras probar que funciona correctamente solo faltaría la instalación del Geoserver para poder acceder a las funcionalidades que demandan su servicio. El proceso de instalación se comentó previamente en la sección que se trabajó en local con el mismo y sigue el mismo funcionamiento.

5.7.5 Resumen trabajo realizado

Esta última iteración se planteó para la solución de posibles errores encontrados durante el desarrollo así como el refinamiento de la aplicación. Como ejecutable o versión entregada se dispone ya del prototipo final en el cual los usuarios podrán probar todas las funcionalidades desde el mapa interactivo, gestión de usuarios y datos hasta todo lo relacionado con la propia escena como recorte, detección de olivos en textura y nube o movimientos libres por la escena.

Por último se realiza integración de la aplicación en un servidor web mediante una máquina virtual con el fin de poder acceder desde la red a la página con un dominio propio. Desde este servicio no se encuentran disponibles todas las funcionalidades pero es posible acceder a gran parte de ellas.

5.8 Pruebas finales con usuarios

Por último, en esta sección se entra en las pruebas finales realizadas al sistema con gente diferente del propio desarrollador de la aplicación con el fin de obtener valoraciones y opiniones sobre su manejo y facilidad de uso. Estas personas se encontrarán acostumbradas al manejo de tecnologías y conocen el fin de este proyecto.

Usuario	Perfil	Edad
Usuario 1	Informático	35
Usuario 2	Usuario aleatorio	22
Usuario 3	Propietario agrícola	45

Tabla 29: Perfiles usuarios de las pruebas de usabilidad.

Las pruebas se dividirán en una serie de tareas donde se le encomendará una tarea al usuario que tendrá que resolver por su propia cuenta a fin de demostrar la usabilidad del servicio, el desarrollador en este caso tendrá simplemente que observar y anotar los posibles comentarios realizados por el usuario a fin de considerarlos a la postre para posibles cambios o mejorar. Las tareas de las que consta esta prueba son las siguientes:

1. Navegación por el mapa y selección de una parcela para su investigación
2. Búsqueda y recolección de datos concretos de la parcela previamente seleccionada.
3. Visualizar una gráfica en grande, reducir los datos visualizados de temperatura a 4 semanas y finalmente entrar a la escena
4. Recorte porción de parcela
5. Detección de entidades en la escena
6. Subida y descarga de un modelo
7. Borrado y reinserción de un usuario
8. Inserción de múltiples datos históricos

A continuación, se comentarán los resultados tras estas pruebas notificando los consejos y comentarios realizados por estos usuarios, en cada una de las pruebas que haya podido suponer un reto, aquellas sin nada a destacar se omitirán en la tabla.

Tareas	Usuario 1
Tarea 2: Información	Pequeños despistes en la navegación por la página
Tarea 3: Operaciones	Poco intuitivo el desplegar una gráfica

Tabla 30: Comentarios pruebas usabilidad del sistema Usuario 1.

Tareas	Usuario 2
Tarea 1: Mapa	Se siente al inicio un poco perdido sobre la utilidad de cada opción del mapa, búsqueda por campos más amplios
Tarea 2: Información	Poca comprensión acerca de cierta terminología
Tarea 3: Operaciones	Poco intuitivo el desplegar una gráfica
Tarea 6: Edición modelos	El usuario no suele cerrar por su cuenta los avisos
Tarea 8: Subida CSV	Sin costumbre es un proceso tedioso

Tabla 31: Comentarios pruebas usabilidad del sistema Usuario 2.

Tareas	Usuario 3
Tarea 1: Mapa	No considera intuitivo tener que desplegar el mismo las capas, búsqueda por campos más amplios
Tarea 3: Operaciones	Poco intuitivo el desplegar una gráfica
Tarea 4: Recorte	No cierra los avisos manualmente y se pierde los nuevos
Tarea 6: Edición modelos	Falta de información sobre formatos a subir

Tabla 32: Comentarios pruebas usabilidad del sistema Usuario 3.

Tras la realización de estas pruebas se han obtenido las conclusiones y feedback que podemos encontrar en la tabla superior, por lo general la prueba ha sido bastante satisfactoria pues gran parte de la problemática encontrada por los usuarios dispone de solución en nuestra aplicación y todos han sido capaces de interactuar con el sistema con soltura.

A continuación, se mostrarán los cambios realizados con el fin de mejorar los aspectos comentados por los usuarios.

- En la sección inicial donde el mapa se ha insertado una leyenda superior (Ilustración 165) con el fin de obtener una mayor comprensión sobre cada botón disponible en la interfaz.

Ilustración 165: Leyenda sobre los botones mapa interactivo (autor).

- Adición a la búsqueda del mapa de una búsqueda por referencias catastrales además de por municipios. Despliegue automático de ciertas capas de información que no supongan una carga excesiva para el sistema.

- Se han añadido más mensajes de notificación de acciones a la interfaz para indicarle cuando realice una acción si es correcta o debe proceder con algo antes, actualización de la eliminación automática de avisos cuando se solventa el problema y se observa un cambio de intenciones en el usuario.

- Aumento de indicaciones a la hora de rellenar datos o subir archivos limitando estos a archivos del formato adecuado así como indicando previamente el formato esperado.

- Simplificación de la terminología usada y agrupamiento de la información en campos comunes para ahorrar tiempo y posibles pérdidas en la navegación.

Capítulo 6: CONCLUSIONES Y TRABAJOS FUTUROS

En este capítulo se tratarán las conclusiones extraídas del trabajo, así como aquellas posibles mejoras que pudieran aplicarse. Este hecho dotaría a la aplicación de un mayor número de funcionalidades así como mejoraría en gran medida el rendimiento de la misma.

6.1 Conclusiones

En este proyecto enmarcado dentro las limitaciones de un Trabajo de Fin de Grado se ha desarrollado una aplicación web que cumple con los principales objetivos y requisitos planteados al inicio del trabajo. Esta aplicación es de vital importancia para facilitar el flujo de trabajo de las personas que trabajan en el sector del olivar y de manera más concreta de las personas encargadas del mantenimiento y toma de decisiones. Fijándonos en los objetivos que se marcaron a lo largo del trabajo se ha conseguido:

- **Desarrollar una aplicación capaz de gestionar gran parte de la información intrínseca de las parcelas de olivar**, así como poder manipular y visualizar aspectos difícilmente accesibles cumpliendo con las siguientes características
    - Construcción de una aplicación que permite acceder a los datos de las parcelas desde cualquier lugar necesitando exclusivamente de una red Wifi.
    - Permite administrar un volumen grande de parcelas y poder consultar cualquier tipo de información necesaria.
    - Permite interaccionar directamente con modelos tridimensionales pudiendo extraer conocimiento visual de trabajar con ellas.
    - Permite extraer información desconocida y almacenarla en una base de datos para poder trabajar a posteriori con la misma.
- Acercarnos poco a poco a lo que es la digitalización completa de procesos de análisis con un volumen de datos difícilmente manejable. Además, se consigue con esto ir en la dirección de desarrollo de la agricultura de precisión.

Por ende, tras observar que los requisitos obligatorios que se marcaron al inicio se han cumplido y que los resultados son satisfactorios para el usuario se puede concluir que

tanto el proyecto como la metodología que se ha seguido cumplen con el propósito marcado y esto motiva a continuar mejorando la aplicación en trabajo futuros.

6.2 Trabajo futuro

El trabajo realizado pese a ser bastante completo y cumplir los requisitos obligatorios expuestos por el usuario no es perfecto. Existen aspectos susceptibles de mejorar y con los que aumentar la funcionalidad.

El principal aspecto a mejorar se encuentra relacionado con los algoritmos de detección de olivar, ya que este no era uno de los requisitos fundamentales del proyecto no se ha llegado a profundizar en exceso en el script desarrollado obteniendo un algoritmo con unos resultados aceptables pero mejorables. Durante la realización del mismo se han empleado algoritmos de procesamiento de imágenes (OpenCV) y agrupamiento o clustering. Sin embargo, en ocasiones trabajar exclusivamente con detección de contornos no es lo más adecuado pues como hemos visto existen parcelas con infraestructuras y esto puede llevar a confusión. Sería de interés probar técnicas de clasificación y detección por color [28], así como aplicar otras técnicas de clasificación y agrupamiento más avanzadas o complejas consiguiendo un script que mediante la combinación de técnicas pueda llegar a disponer de una tasa de detección muy elevada, así como de una considerable reducción de los falsos positivos.

Ligado a esto se podría disponer de funcionalidades más precisas que las desarrolladas previamente. En cuanto al enriquecimiento de información que se propone entre la textura y la nube de puntos es posible mejorar los algoritmos de detección y clustering implementados, así como permitir al usuario realizar estas operaciones con porciones más pequeñas de la parcela. Un ejemplo de esto puede ser mediante el recorte no solo de la nube sino de la textura a la par, todo esto gracias al solapamiento de ambos modelos. Este como se comentó en su momento es uno de los principales problemas encontrados durante el desarrollo del proyecto y otro punto a mejorar de cara al futuro de la aplicación.

Otro aspecto a mejorar para un mayor enriquecimiento de la información es la dotación de las texturas de un mayor nivel de precisión en cuanto a la simulación de la altura [29] mediante mapas de sombras para obtener un modelo tridimensional más realista. Esta operación no ha sido del todo exitosa en el desarrollo de la escena ya que por el nivel de detalle disponible los resaltes de altura no son lo suficientemente notables para disponer de una representación fiel con la que extraer datos de elevación exactos y precisos así como la

problemática de coincidencia espacial de nube y textura que impedía desarrollar con éxito la recreación de la inclinación del terreno. Con el trabajo de capas de tipo ráster a un menor nivel de detalle, así como con el contexto de la elevación de la zona de alrededor y no exclusivamente en base a la propia parcela se podría llegar a disponer de una representación mucho más afín a lo que observamos en la realidad. Otra solución más sencilla y a un menor nivel de detalle sería la de mejorar el sistema actual con una mejor coincidencia de modelos como se comentó previamente.

Un aspecto que no se ha tratado en este trabajo pero que pudiera ser de interés es la integración en tiempo real de los datos de ciertas parcelas [30], mediante la sensorización de las mismas se podría obtener datos importantes en tiempo real pudiendo procesarlos para enriquecer la base de datos, esta pasaría a ser del tipo espacial-temporal consiguiendo de esta manera acércanos todo los posible a la recreación de un digital twin [31] con las parcelas que así lo requieran. De manera inicial se podría disponer de sensores del tipo de estaciones meteorológicas para datos del clima, para detección de plagas u otros fines que nos permitieran disponer de históricos constantes con los que tratar de predecir rendimientos futuros en función a los históricos de rendimiento de cada parcela

Finalmente, y para completar la aplicación se podría indagar también en el campo del aprendizaje automático [32] y la minería de datos con el fin de aprender de los datos disponibles y poder tratar de adelantarnos a posibles problemas que surjan estableciendo patrones o reglas de asociación. En caso de disponer de avances como los que proponen los digital twin en los que nuestra aplicación se encuentra en constante comunicación con la parcela y disponemos de datos constantes mediante herramientas y algoritmos de aprendizaje automático sería posible predecir a largo plazo que tratamientos o medidas serían recomendables para orientar el rendimiento de nuestra parcela hacia el espacio temporal pasado donde el rendimiento fue elevado.

Capítulo 7: DEFINICIONES Y ABREVIATURAS

7.1 Terminología

Término	Definición
Ortofotografía	Presentación fotográfica de la orográfica terrestre con todos los elementos a la misma escala, se consigue con imágenes aéreas con el objetivo de obtener una proyección ortogonal.
Agricultura de Precisión	Gestión de la información agrícola con el objetivo de obtener un mayor rendimiento mediante la integración de la tecnología
Teledetección	Técnica de adquisición de datos de la superficie terrestre desde sensores disponibles en plataformas espaciales
Umbralización	Técnica de segmentación simple que permite separar pixeles de una imagen en escala de grises en función de un umbral
Clustering	Tarea de agrupar objetos por similitud o cercanía en sus características, es una tarea de minería de datos exploratoria
Bounding Box	Caja capaz de envolver una primitiva u objeto geométrico en una primitiva más simple con la que se puede trabajar
Burndown	Representación gráfica del trabajo que queda por hacer en un proyecto en el tiempo restante
Ráster	Representación de una imagen digitalizada en forma de cuadricula o matriz de pixeles donde cada celda tiene un valor indicativo de información del mundo real como altura, temperatura…
Voxelización	Proceso de discretización de modelos geométricos utilizado para obtener ventajas computacionales. Es una representación aproximada de la geometría
Mesh	Representación poligonal de la geometría generada mediante un sistema de vértices posicionados en el espacio de coordenadas. El formato más habitual es la malla de triángulos

Pseudo-Mercator	Proyección cartográfica ampliamente utilizada para representar la Tierra en mapas digitales y navegación en línea.

7.2 Abreviaturas

Abreviatura	Término desarrollado	Definición
UAVs	Unmanned Aerial Vehicles	Vehículo aéreo no tripulado, comúnmente conocido como dron es una aeronave manejada remotamente
LiDAR	Ligth Detection and Ranging	Dispositivo con la capacidad de determinar la distancia desde un emisor a un objeto mediante un haz láser expulsado
SigPac	Sistema de Información Geográfica de Parcelas Agrícolas Comunitarias	Sistema de información público del Gobierno de España empleada para la identificación de parcelas de olivar declaradas
SWIR	Short Wavelength InfraRed	Acrónimo con el que se designa a la parte del espectro electromagnético del infrarrojo de onda corta. Se encuentra justo después del NIR y se sitúa en el intervalo 1000 – 3000 nanómetros
NIR	Near InfraRed	Acrónimo con el que se designa a la parte del espectro electromagnético ubicado en el intervalo 800-2500 nanómetros justo por encima del rango visible de luz.
WMS	Web Map Service	Servicio de mapas con datos referenciados espacialmente de forma dinámica mediante información geográfica
WFS	Web Feature Service	Servicio de mapas con el objetivo de proporcionar información relativa a entidades almacenadas en coberturas vectoriales
CSW	Catalogue Service for the Web	Estándar para la recuperación, captura y consulta de metadatos de los datos referenciados espacialmente
PNOA	Plan Nacional de Ortofotografía Aérea	Proyecto con el objetivo de obtención de productos fotogramétricos del territorio nacional para obtención de ortofotografías y modelos digitales del terreno

SIG	Sistema de Información Geográfica	Conjunto de herramientas que permite el almacenamiento, manipulación, análisis y otras funcionalidades con datos espaciales procedentes del mundo real.
IDE	Integrated Development Environment	Aplicación utilizada para ayudar en el desarrollo de creación de software
DEM	Digital Elevation Model	Representación ráster de una superficie continua que generalmente hace referencia a la superficie de la tierra
CASE	Computer-aided software engineering	Aplicaciones informáticas destinadas a aumentar el balance en el desarrollo de software reduciendo coste temporal y monetario
UML	Unified Modeling Lenguage	Lenguaje de modelado de sistemas de software más utilizado en la actualidad. Lenguaje gráfico para visualizar, especificar y construir un sistema
COCOMO	Constructive Cost Model	Modelo matemático utilizado para estimar coste software de un proyecto
AEMET	Agencia Estatal de Meteorología	Agencia adscrita al Gobierno de España con el objeto del desarrollo, implantación y prestación de servicios meteorológicos del estado.
IGN	Instituto Geográfico Nacional	Dependencia científica y técnica del Registro Nacional que se ocupa de la cartografía del país.
DERA	Datos Espaciales de Referencia de Andalucía	Repositorio de bases cartográficas de diferente naturaleza geométrica referidas al territorio andaluz.
PLY	Polygon File Format	Formato de archivo diseñado para almacenar datos tridimensionales de escáneres 3D
TIFF	Tagged Image File Format	Formato de archivo informático para almacenamiento de imágenes de mapa de bits
DBSCAN	Density Based Spatial Clustering of Applications with Noise	Algoritmo de agrupamiento espacial basado en densidad de aplicaciones con ruido

JSON	JavaScript Object Notation	Formato de texto que forma parte del sistema JavaScript y que deriva de sus sintaxis. Desarrollado para facilitar análisis sintáctico por su estructuración
RGB-A	Red Green Blue Alpha	Sistema de definición de color basado en su descomposición en los colores primario de la luz y un valor de trasparencia
MDT	Modelo Digital del Terreno	Estructura numérica de datos que representa la distribución espacial de una variable cuantitativa y continua

Capítulo 8: BIBLIOGRAFÍA

[1] F. J. Pierce y P. Nowak, «Aspects of Precisión Agriculture,» de *Advances in Agronomy*, vol. 67, D. L. Sparks, Ed., Academic Press, 1999, pp. 1-85.

[2] R. Gebbers y V. I. Adamchuk, «Precision Agriculture and Food Security,» *Science,* vol. 327, pp. 828-831, 12 Febrero 2010.

[3] V. Moysiadis, V. Vitsas , P. Sarigiannidis y A. Khelifi, «Smart Farming in Europe,» *Computer Science Review,* vol. 39, p. 100345, 2021.

[4] P. Shanmugapriya, S. Rathika, T. Ramesh y P. Janaki, «Applications of remote sensing in agriculture-A Review,» vol. 8, nº 01, pp. 2270-2283, 2019.

[5] M. Wójtowicz, A. Wójtowicz y J. Piekarczyk, «Application of remote sensing methods in agriculture,» *Communications in biometry and crop science,* vol. 11, nº 1, pp. 31-50, 2016.

[6] A. Khan, U. Khan, M. Waleed, A. Khan, T. Kamal, S. N. Marwat, M. Maqsood y F. Aadil, «Remote Sensing: An Automated Methodology for Olive Tree Detection and Counting in Satellite Images,» *IEEE Access,* vol. 6, pp. 77816-77828, 2018.

[7] R. H. Güting, «An introduction to spatial database systems,» *The VLDB Journal,* vol. 3, nº 4, pp. 357-399, 1994.

[8] S. Shekhar, S. Chawla, S. Ravada, A. Fetterer, X. Liu y C.-T. Lu, «Spatial databases-accomplishments and research needs,» *IEEE Transactions on Knowledge and Data Engineering,* vol. 11, nº 1, pp. 45-55, 1999.

[9] A. Aquino, J. M. Ponce, B. Millán Prior, D. Tejada-Guzmán y J. M. Andújar-Márquez, «Identificación y conteo de aceitunas en imágenes digitales tomadas en el olivar mediante morfología matemática y redes neuronales convolucionales,» de *XL Jornadas de Automática*, 2019.

[10] A. Álvarez, C. Lasa y R. de las Heras, Métodos Ágiles. Scrum, Kanban, Lean, Anaya Multimedia, 2017.

[11] M. Bermejo, El Kanban, Barcelona: Barcelona, España: UOC, 2011.

[12] K. Schwaber y J. Sutherland, La guía definitiva de Scrum: las reglas del juego, 2020.

[13] S. Laoyan, «Scrumban: lo mejor de dos metodologías ágiles,» Asana, 31 11 2022. [En línea]. Available: https://asana.com/es/resources/scrumban. [Último acceso: 01 05 2023].

[14] B. W. Boehm, «Software Engineering Economics,» *IEEE Transactions on Software Engineering,* Vols. %1 de %2SE-10, nº 1, pp. 4-21, 1984.

[15] L. Xiaosong, L. Shushi, C. Wenjun y F. Songjiang, «The Application of Risk Matrix to Software Project Risk Management,» de *International Forum on Information Technology and Applications*, 2009.

[16] «Ministerio para la transición ecológica y el reto demográfico,» Gobierno de España, [En línea]. Available: https://www.miteco.gob.es/es/calidad-y-evaluacion-

ambiental/temas/productos-quimicos/fitosanitarios/. [Último acceso: 14 12 2022].

[17] A. y. M. A. Ministerio de Agricultura y Pesa, Junio 2017. [En línea]. Available: https://www.mapa.gob.es/images/es/170612_propuestarenovacion_panuspff_tcm30-381264.pdf. [Último acceso: 15 12 2022].

[18] «Leaflet,» [En línea]. Available: https://leafletjs.com/reference.html. [Último acceso: 22 3 2023].

[19] «OpenLayers,» [En línea]. Available: https://openlayers.org/. [Último acceso: 20 4 2023].

[20] Wikipedia, «Wikipedia,» 2020. [En línea]. Available: https://es.wikipedia.org/w/index.php?title=Algoritmo_radial&oldid=128862125. [Último acceso: 27 5 2023].

[21] S. Fang y H. Chen, «Hardware accelerated voxelization,» *Computers & Graphics,* vol. 24, nº 3, pp. 433-442, 2000.

[22] «DAT GUI,» [En línea]. Available: https://github.com/dataarts/dat.gui. [Último acceso: 13 4 2023].

[23] Wikipedia, «Algoritmo de Canny,» 2020. [En línea]. Available: https://es.wikipedia.org/w/index.php?title=Algoritmo_de_Canny&oldid=131215822. [Último acceso: 18 2 2023].

[24] «OpenCV: Canny Edge Detection,» OpenCV, [En línea]. Available: https://docs.opencv.org/3.4/da/d22/tutorial_py_canny.html. [Último acceso: 18 2 2023].

[25] Wikipedia, «DBSCAN,» 2023. [En línea]. Available: https://en.wikipedia.org/wiki/DBSCAN. [Último acceso: 20 2 2023].

[26] J. Amanatides y A. Woo, «A fast voxel traversal algorithm for ray tracing.,» de *Eurographics*, 1987.

[27] «ChartJS,» [En línea]. Available: https://www.chartjs.org/docs/latest/.

[28] J. L. Hernández, G. García-Mateos, J. M. González-Esquiva, D. Escarabajal-Henarejos, A. Ruiz-Canales y J. M. Molina-Martínez, «Optimal color space selection method for plant/soil segmentation in agriculture,» *Computers and Electronics in Agriculture,* vol. 122, pp. 124-132, 2016.

[29] W.-S. Shin y N. Baek, «Editing LiDAR-Based Terrains with Height and Texture Maps,» de *2016 International Conference on Information Science and Security (ICISS)*, 2016.

[30] N. R. Kitchen, «Emerging technologies for real-time and integrated agriculture decisions,» *Computers and Electronics in Agriculture,* vol. 61, nº 1, pp. 1-3, 2008.

[31] M. Singh, E. Fuenmayor, E. P. Hinchy, Y. Qiao, N. Murray y D. Devine, «Digital Twin: Origin to Future,» *Applied System Innovation,* vol. 4, nº 2, 2021.

[32] K. G. Liakos, P. Busato, D. Moshou, S. Pearson y D. Bochtis, «Machine Learning in Agriculture: A Review,» *Sensors,* vol. 18, nº 8, 2018.

Capítulo 9: APÉNDICES

Para este capítulo se han dejado las secciones necesarias para poder comprender este trabajo correctamente en su completitud pero que no encajaban en capítulos anteriores, por este motivo se incluirán a continuación.

9.1 Guía original del Trabajo Fin de Título

Este Trabajo de Fin de Grado fue asignado y publicado durante el transcurso del primer cuatrimestre del curso académico 2022/2023 por la Universidad de Jaén para el grado de Ingeniería Informática. La guía desarrollada para el mismo no ha sido modificada desde su publicación[30].

9.2 Instalación y configuración del sistema

Al tratarse de una aplicación web y como se ha expuesto a lo largo del desarrollo esta se encuentra disponible en un servidor web y por tanto no sería necesaria una descarga o instalación de la misma, sin embargo, ya que como se expuso en la Sección 1.6.2 se realiza la entrega de la última versión de la aplicación con todo el código y archivos necesarios, por ende siguiendo las explicaciones de esta documentación será posible realizar una instalación local del sistema y disponer de el así como de todas sus funcionalidades.

9.3 Manuales de usuario

Con el fin de facilitar la primera experiencia de los usuarios con la aplicación en este trabajo se ha incluido el desarrollo de un manual de usuario que indique a los usuarios como ser capaces de acceder a todas las funcionalidades del sistema, así como poder emplear estas de la manera correcta.

En la Sección 5.3.3 se especificó el diagrama de los posibles recorridos que un usuario puede tomar, en este manual se especificará a continuación dentro de las distintas interfaces cuales son las opciones posibles.

1. **Inicio sesión.** Como se ha ido viendo a lo largo del desarrollo de este documento para que un usuario pueda disponer de los servicios de la aplicación debe encontrarse en primer lugar registrado en la aplicación. Si

[30] http://eps-anterior.ujaen.es/TFGtemporal/mostrarTFG.php?id=3802

este se encuentra registrado deberá introducir su nombre de usuario (1) y contraseña (2) para finalmente aplicar o bien el botón de login (3) o pulsar la tecla enter para formalizar el formulario.

Ilustración 166: Inicio sesión (autor).

2. **Interacción con el mapa.** Una vez que el usuario haya iniciado sesión se la planteará la opción de interactuar con el mapa. El funcionamiento de este es sencillo, pero dispone de múltiples controles que pudieran liar un poco al usuario. En primer lugar, observaremos un mapa base con la cartografía de España además de un resalto de la zona de Andalucía que es de donde se disponen datos en la actualidad. Mediante los controles de zoom (1) (2) podremos manejar el mapa, así como con la rueda del ratón o los dedos en caso de estar en un dispositivo móvil, se presenta también la posibilidad de poner el mapa en pantalla completa (3) y realizar una búsqueda del municipio que deseemos para hacer zoom a este (4).

   Finalmente se destacarán el control de capas que nos permite desplegarlo (5) y activar unas capas u otras del mapa conforme lo deseemos. Las capas base y de limitaciones territoriales (6) (7) se irán modificando con el zoom del mapa y al aumentarlo al nivel 15 se podrá automáticamente el mapa satelital, sin embargo, tenemos la opción de cambiar la vista manualmente hasta que volvamos a cambiar el zoom de la escena. Cuando nos encontramos seleccionando una parcela con la capa activa será necesario por último clicar

en el botón de activar selección (8) y posteriormente clicar en la parcela deseada.

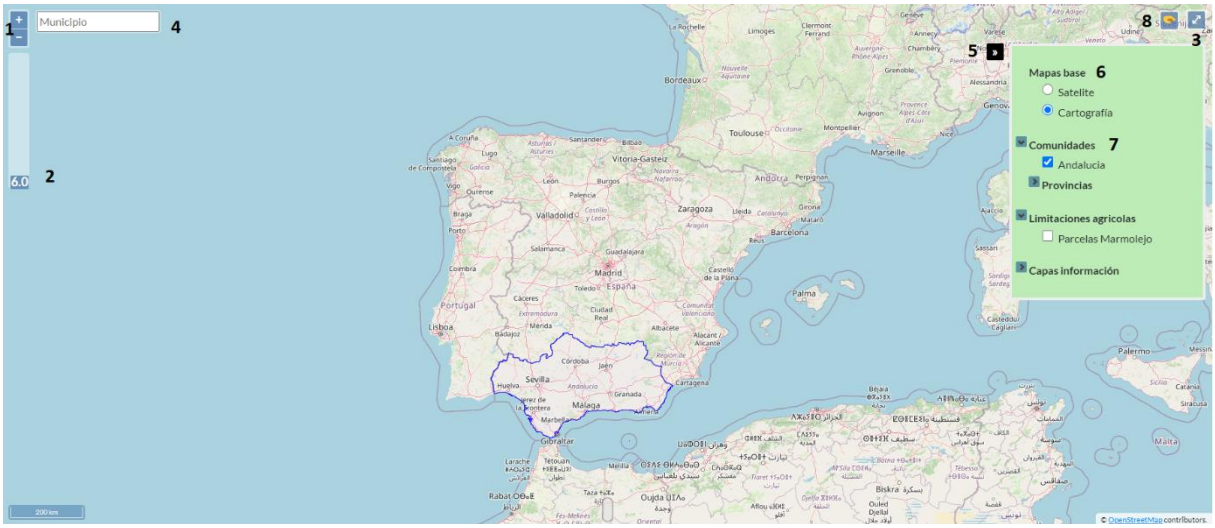


Ilustración 167: Mapa interactivo (autor).

3. **Información parcela.** Esta página es muy simple y no dispone de muchas funcionalidades que puedan llegar a liar al usuario o dificultar si experiencia. En primer lugar, disponemos de datos básicos de la parcela, así como imágenes para su representación. Lo único destacable de la primera ventana será el botón para movernos a la escena (1).

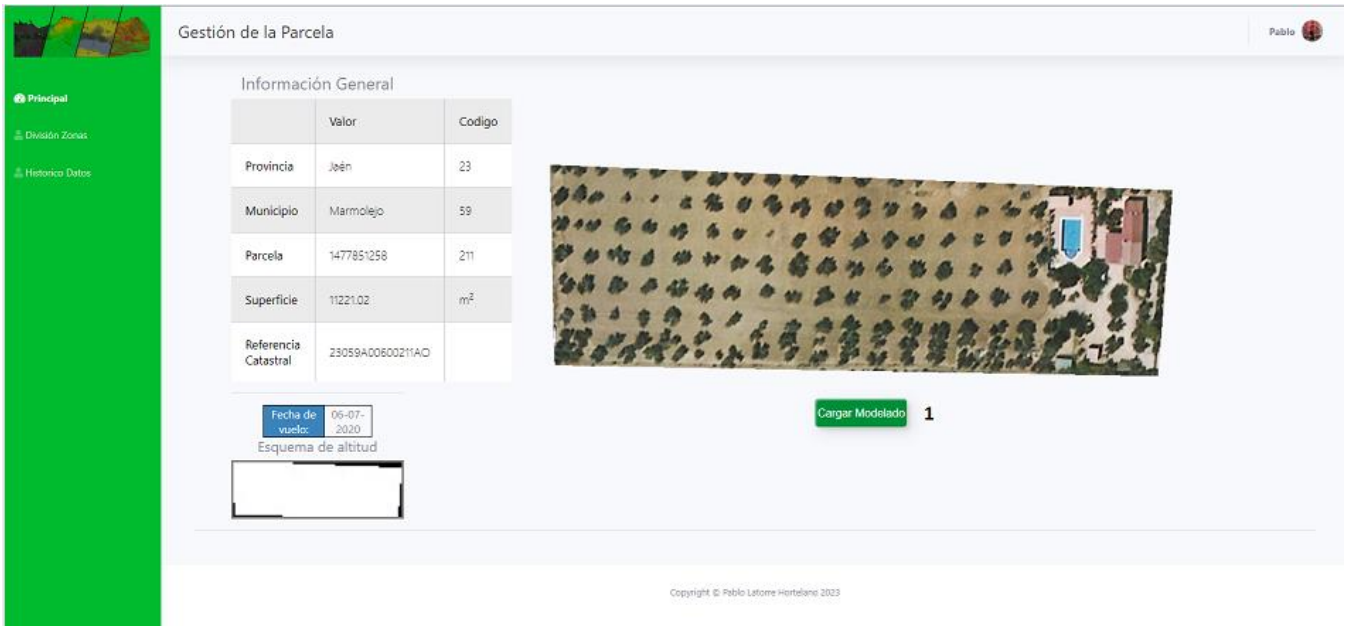


Ilustración 168: Página principal información (autor).

Si seleccionamos la opción de división zonas de la barra lateral (1) nos moveremos a una nueva ventana con información más concreta de la parcela. Además se dispone de la opción de imprimir en formato PDF las tablas mostradas

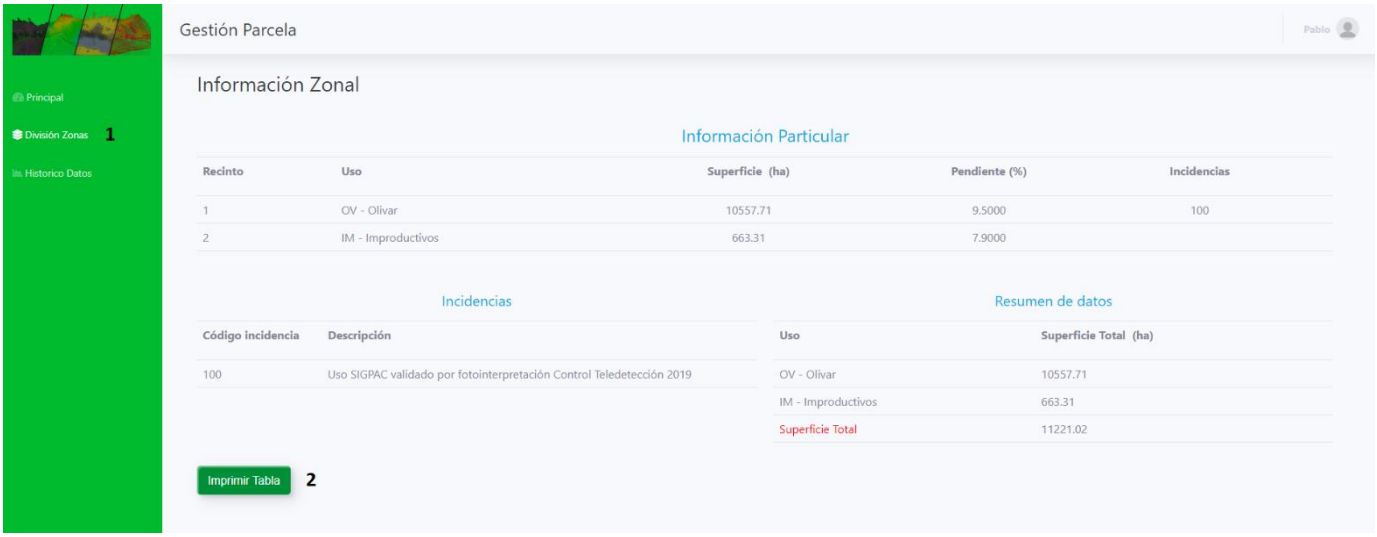


Ilustración 169: Página información zonal (autor).

Por último, se dispone de los datos históricos en formato gráfico (1) en esta ventana se presentan 4 gráficas con datos puramente visuales, la principal funcionalidad a destacar es que clicando en los nombres (2) de las gráficas se podrán abrir estar con un mayor tamaño, además, con el slider inferior azul (3) podrás determinar cuántos datos observar desde el mínimo de 4 hasta el máximo existente para el aspecto dado.

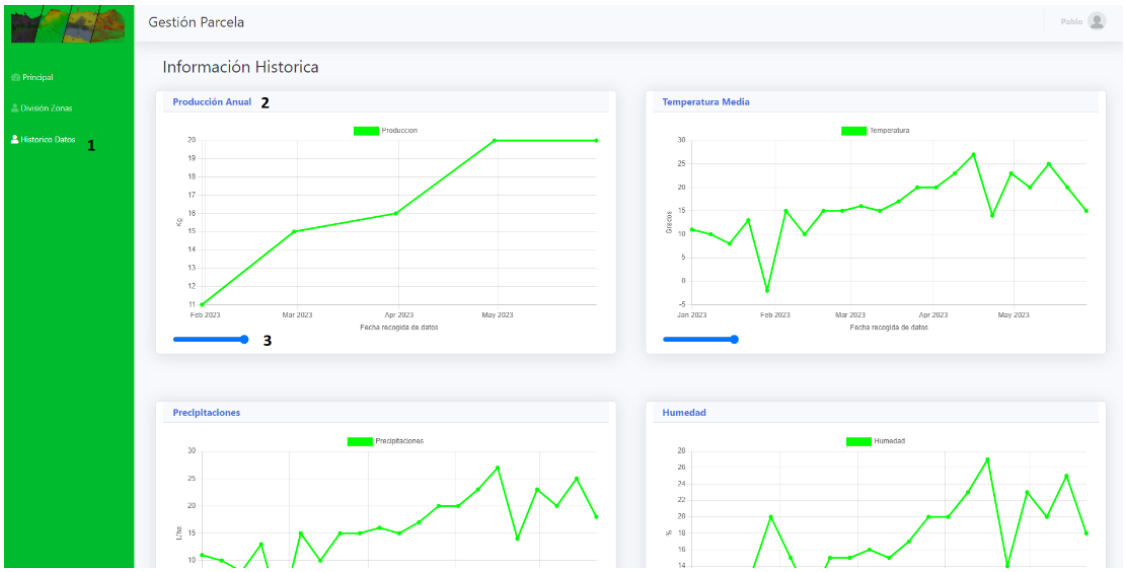


Ilustración 170: Página de históricos (autor).

4. **Escena.** Esta ventana será la que más funcionalidades disponga para su uso, es necesario dividir el proceso en dos categorías como son los controles de teclado y los controles por interfaz de usuario.

   En primer lugar, los controles por teclado se presentan en la siguiente tabla:

Tecla	Descripción
Flecha izquierda	Visión cenital: Mover cámara X negativo Visión perspectiva: Rotación eje Y
Flecha derecha	Visión cenital: Mover cámara X positivo Visión perspectiva: Rotación eje Y
Flecha arriba	Visión cenital: Mover cámara Z negativo Visión perspectiva: Rotación eje X
Flecha abajo	Visión cenital: Mover cámara Z positivo Visión perspectiva: Rotación eje X
W	Visión cenital: Zoom in
S	Visión cenital: Zoom out

Tabla 33: Controles por teclado de la escena.

   En el caso de encontrarnos en un dispositivo móvil es posible manejar la escena y la cámara con la interacción táctil fácilmente.

   Por otro lado, tenemos los controles de la interfaz gráfica que nos permitirá cambiar el tipo de perspectiva de la escena (1), centrar la cámara (2), variar el zoom manualmente, así como mover la cámara (3) y elegir qué tipo de modelo queremos observar (4). Estas funcionalidades son las más básicas en adición a estas luego habrá acciones más concretas que necesitan de mayor procesamiento.

   Para disponer de la selección y el recorte será necesario encontrarnos en perspectiva cenital (1) y marcar el checkbox de selección (5), una vez cumplamos estos requisitos se nos abrirá la sección de recorte (6) y podremos proceder con esta. En caso de realizar la detección (7) se deberá hacer en el modelo de textura o combinatoria de textura y nube para la coherencia visual.

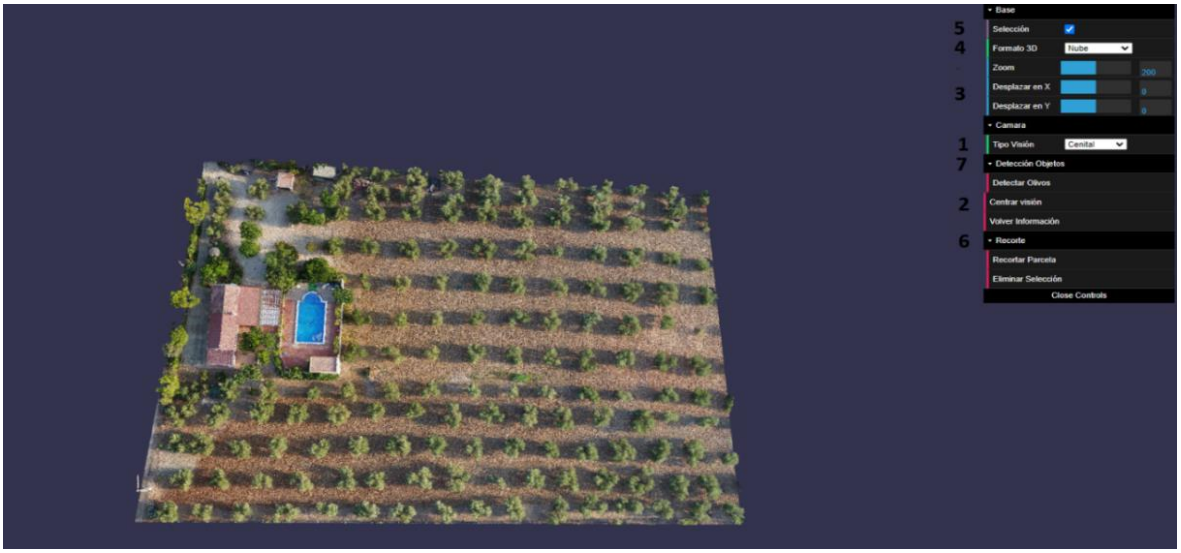


Ilustración 172: Escena (autor).

Fuera de estas funcionalidades y tras aplicar ciertas operaciones podemos encontrar que se desbloquean nuevas funcionalidades de la interfaz como son la reestructuración de la nube de puntos (1) si esta ha sido recortada o la corrección del proceso de detección de olivos (2)(3) así como el proceso de clustering para obtener los olivos para una nube de puntos dada (4) estando estos disponibles cuando se active la detección.

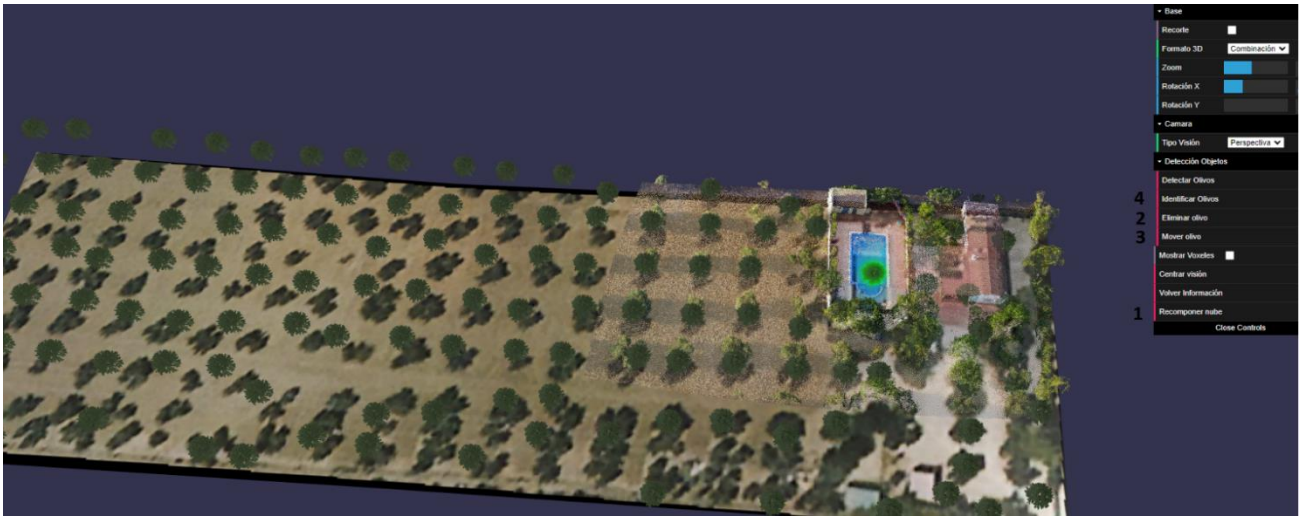


Ilustración 171: Operaciones secundarias escena (autor).