Universidad de Valladolid

ESCUELA DE INGENIERÍA INFORMÁTICA DE SEGOVIA

Grado en Ingeniería Informática de Servicios y Aplicaciones

Aplicación web para el seguimiento anual de multicultivos

Alumno: Laura Sastre Sacristán

Tutor: Fernando Díaz Gómez

Aplicación web para el seguimiento anual de multicultivos

Laura Sastre Sacristán

Agradecimientos

Quiero expresar mi más sincero agradecimiento a todas las personas que me han acompañado y apoyado en este camino. A mi familia, por su apoyo incondicional, por estar siempre a mi lado y por creer en mí incluso en los momentos más difíciles.

A mis amigos, por su comprensión, por sus palabras de ánimo y por estar ahí cuando más los necesitaba, haciendo este camino mucho más llevadero.

También quiero expresar mi gratitud a mi tutor, Fernando, por su apoyo y orientación en cada paso de este proyecto. Su ayuda ha sido crucial para alcanzar este logro.

Resumen

Hoy en día, la tecnología agrícola ha hecho grandes avances. Desde el comienzo de la agricultura, la gente ha estado buscando formas de facilitar y acelerar el trabajo de los agricultores en los campos con herramientas que permitan avanzar en los métodos de trabajo, haciendo más eficientes las explotaciones, es por ello que la demanda de servicios que permitan el seguimiento de los cultivos ha aumentado.

El objetivo es desarrollar una aplicación web que permita a los usuarios agricultores mantener al día la información de sus cultivos (los cultivos en sí, los terrenos donde se cultivan y las labores realizadas en cada cultivo), así como una estimación de los costes.

Las tecnologías empleadas para el desarrollo de la aplicación web serán utilizando MySQL y PHP en el lado servidor y HTML, CSS y Javascript en el lado cliente. La aplicación deber ser utilizable en dispositivos móviles. Para su desarrollo se hará utilizando la metodología iterativo-incremental que permite el crecimiento progresivo del producto.

Palabras claves: Aplicación web, agricultura, cultivos.

Abstract

Nowadays, agricultural technology has made great strides. Since the beginning of agriculture, people have been looking for ways to facilitate and speed up the work of farmers in the fields with tools that allow progress in working methods, making farms more efficient, which is why the demand for services that allow monitoring of crops has increased.

The objective is to develop a web application that allows farmer users to keep up-to-date with the information on their crops (the crops themselves, the land where they are grown and the work carried out on each crop), as well as a cost estimate.

The technologies used for the development of the web application will be using MySQL and PHP on the server side and HTML, CSS and Javascript on the client side. The application must be usable on mobile devices. For its development it will be done using the iterative-incremental methodology that allows the progressive growth of the product.

Palabras claves: Web application, agriculture, crops.

Índice general

Índice de figuras

Índice de tablas

Parte I

Memoria del Proyecto

Capítulo 1

Descripción del proyecto

1.1. Introducción

La agricultura es una actividad fundamental del sector primario, es una actividad humana que implica el cultivo de plantas con el objetivo de producir alimentos, medicamentos y otros productos útiles para el consumo humano y animal. La agricultura es esencial para el ser humano y es una de las actividades económicas más antiguas de la historia, ya que ha permitido a la población obtener los alimentos y recursos necesarios para su supervivencia, ya que los productos agrícolas son consumidos directamente o procesados en la industria para obtener alimentos, materiales textiles, químicos o manufacturados y se ha desarrollado y evolucionado a lo largo de miles de años.

En sus inicios, la agricultura se basaba en la recolección de plantas. No obstante, con el tiempo, los humanos comenzaron a cultivar sus propias cosechas. Este proceso se inició hace unos 10,000 años en varias partes del mundo, incluyendo Mesopotamia, China, África y América del Sur.

En la actualidad, en estos últimos años, el sector agrícola se ha convertido en un asunto cada vez más relevante, dado el crecimiento de población mundial, enfrentándose así a numerosos desafíos, derivado de la necesidad de garantizar la seguridad alimentaria y medioambiental en el mundo.Por lo tanto, la tecnología es una herramienta clave en su desarrollo porque juega un papel fundamental en la búsqueda de soluciones innovadoras que permitan mejorar la eficiencia y sostenibilidad de la agricultura.

La agricultura desempeña un papel fundamental en la economía de España, en gran parte por la amplia gama de cultivos que abarca, como frutas, verduras, cereales, aceitunas y vinos, favorecida por la variada geografía y los diversos climas del país. Por otro lado, esta actividad está ganando popularidad por sus numerosos beneficios, que incluyen la mejora en la calidad del suelo, la diversificación de la producción y la disminución de los riesgos asociados a la pérdida de cosechas.

No obstante, varios factores están afectando la agricultura a nivel mundial, como el cambio climático, la escasez de agua y la presión por reducir los costos de producción, lo que obliga a buscar nuevas herramientas y tecnologías para aumentar la productividad y competitividad en el sector agrícola. La agricultura de precisión y la automatización en la gestión de maquinaria son ejemplos claros de cómo la tecnología está transformando el panorama agrícola.

Esta evolución tecnológica está teniendo un impacto significativo en la forma en que los agricultores gestionan sus cultivos y en la productividad de la actividad agrícola. Las nuevas tecnologías permiten a los agricultores recopilar y analizar datos sobre la condición meteorológica, la calidad del suelo y el estado de los cultivos en tiempo real, lo que les permite tomar decisiones más informadas y eficientes.

Es por ello que la realización de una aplicación web para el seguimiento de multicultivos, puede ser una solución innovadora para mejorar la gestión de este tipo de agricultura. Esta aplicación web puede proporcionar información útil sobre cada cultivo, incluyendo la planificación de sus actividades agrícolas y permitiendo una evaluación detallada de la rentabilidad de la producción agrícola.

En un mundo en constante evolución, es importante seguir explorando nuevas herramientas y tecnologías que permitan a los agricultores maximizar su productividad y reducir el impacto ambiental de sus prácticas agrícolas, asegurando así la supervivencia del sector agrícola a largo plazo.

1.2. Motivación y objetivos del trabajo

La creación de una aplicación para el seguimiento de multicultivos puede ser una solución innovadora y efectiva para ayudar a los agricultores a gestionar sus cultivos de manera más eficiente y sostenible. Una de las mayores preocupaciones de muchos agricultores consiste en que, no están familiarizados con las tecnologías avanzadas, lo que puede dificultar su adopción y uso. No tener un software, amigable y sencillo de usar, no permitirá obtener una mayor eficiencia por parte los agricultores.

Por esa razón, la creación de la aplicación con una interfaz sencilla y amigable permitiría a los agricultores acceder y utilizar la información de manera más rápida y eficiente, reduciría el tiempo y el costo de capacitación, y generaría confianza en el uso de la tecnología en su trabajo diario. Por lo tanto, es esencial que la interfaz de la aplicación esté diseñada teniendo en cuenta las necesidades y habilidades de los agricultores.

En esta línea de trabajo se identifican varios objetivos:

- **OBJ-01**: Desarrollar una aplicación web eficiente, que cuente con una interfaz de usuario atractiva, intuitiva y fácil de usar, y que permita una gestión y seguimiento adecuados de los diferentes cultivos.

- **OBJ-02**: Contar con funcionalidades que permitan a los usuarios analizar y visualizar los datos relacionados con cada uno de los cultivos, como la información específica de las parcelas, sus tratamientos y costes.
- **OBJ-03**: Debe ser diseñado de tal manera que pueda manejar un gran volumen de datos, ya que se espera que se generen grandes cantidades de información relacionada con los diferentes cultivos. Además, la solución debe ser fácilmente escalable, para poder agregar nuevas funcionalidades y características a medida que se requieran en el futuro.

1.3. Alcance

El alcance de esta aplicación abarca la creación de una plataforma web con una variedad de funciones diseñadas para optimizar la gestión y el análisis de cultivos, tratamientos y costos. Esta plataforma permitirá a los usuarios mejorar su experiencia mediante herramientas avanzadas y fáciles de usar. A continuación, se incluye un árbol de características que describe las principales funcionalidades que ofrecerá la aplicación.

1.3.1. Árbol de características

El árbol de características es una representación visual de las funcionalidades principales que tendrá la aplicación. Cada rama funcional se describe a continuación:

- **Gestión de usuarios**: La gestión de usuarios incluye funcionalidades como el registro de nuevos usuarios, inicio y cierre de sesión, así como la edición y eliminación del perfil. Permite a los usuarios administrar su información personal y gestionar su acceso a la plataforma de manera eficiente y segura. Esta sección garantiza que los usuarios tengan control total sobre sus cuentas y sus datos.
- **Gestión de parcelas**: Se centra en el manejo de toda la información relacionada con las parcelas agrícolas. Incluye el registro, edición y eliminación de parcelas, así como la visualización de detalles específicos. Esta rama también incluye la visualización de los tratamientos aplicados a una parcela específica.
- **Gestión de tratamientos**: Permite a los usuarios registrar, editar y eliminar tratamientos específicos aplicados a las parcelas. Esta funcionalidad incluye la visualización de una lista completa de tratamientos y los detalles de cada uno. Facilita la administración precisa y detallada de los tratamientos, ayudando a los usuarios a mantener un seguimiento de las actividades realizadas en sus cultivos.
- **Gestión de costes**: La gestión de costes abarca la administración de los costes asociados a las explotaciones agrícolas, incluyendo el registro, edición y eliminación de costes. Permite la estimación de los costes por tipo, por año y por temporada. Esta

rama es crucial para la planificación financiera y el control de gastos, proporcionando a los usuarios herramientas para gestionar eficazmente los recursos económicos de sus explotaciones.

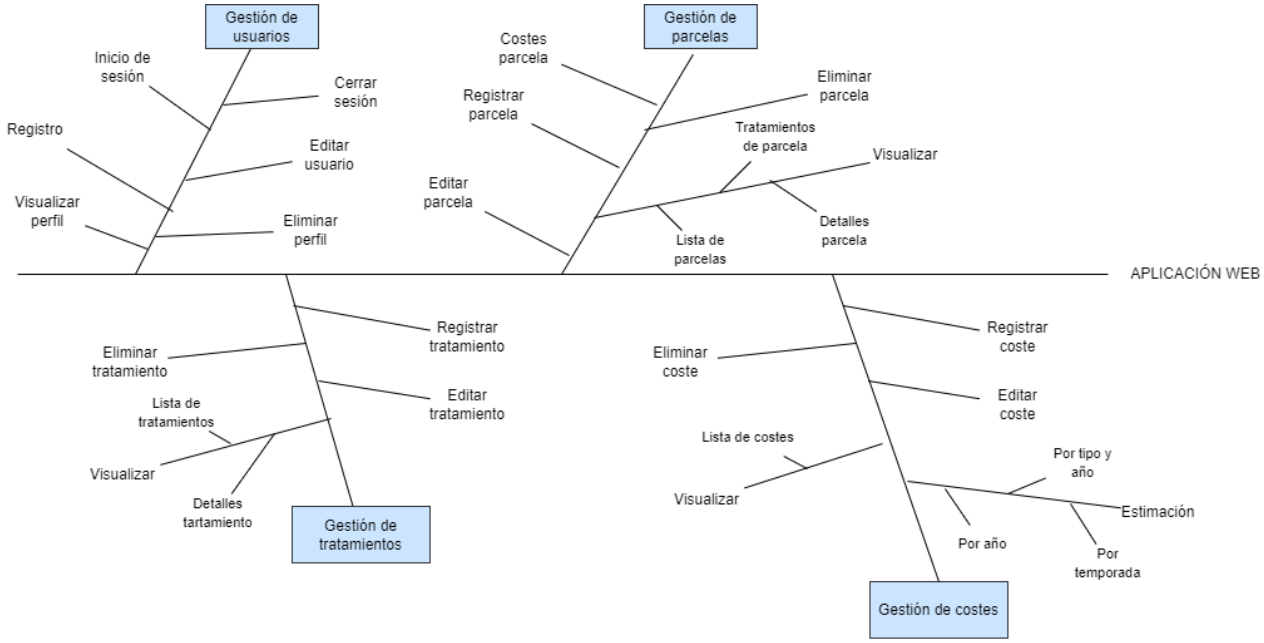


Figura 1.1: Árbol de características

1.3.2. Restricciones y limitaciones

Existen limitaciones en cuanto a la conectividad en áreas rurales donde se utilizará mayormente la aplicación. Es necesario tener en cuenta las restricciones de recursos y tiempo para el desarrollo e implementación de la plataforma web. Dado que este proyecto se realiza en el contexto del Trabajo de Fin de Grado (TFG), la dedicación estimada es de 12 créditos ECTS, lo que se traduce en un total de entre 300 y 360 horas de trabajo.

1.4. Estructura de la memoria

En esta sección se explica la organización del documento con el fin de hacer su lectura más sencilla. A continuación se detallan los capítulos que se incluyen:

- **Capítulo 1 - Descripción del proyecto**: En este capítulo se abordan los aspectos fundamentales del proyecto, como la motivación, los objetivos, el alcance, las restricciones y el tecnológico, también aparece la estructura de la memoria.

- **Capítulo 2 - Entorno de la aplicación**: Este capítulo se enfoca en el estado del arte, dónde se analiza el nivel de conocimiento actual en la materia, y las herramientas utilizadas para el proyecto.
- **Capítulo 3 - Planificación y presupuestos**: En este capítulo se conoce la metodología de trabajo empleada durante el desarrollo del proyecto, junto con la planificación temporal y la estimación de los costos necesarios para llevarlo a cabo.
- **Capítulo 4 - Análisis del sistema**: En este capítulo se presenta un análisis detallado del sistema, donde se describen los actores involucrados, se especifican los requisitos tanto funcionales como no funcionales, y se documentan los casos de uso del sistema. También se incluye un análisis de los requisitos de información (modelo Entidad-Relación, modelo relacional y diccionario de datos.
- **Capítulo 5 - Diseño del sistema**: En este capítulo se describe la arquitectura general, incluyendo tanto la arquitectura lógica como la física, los diagramas de secuencia que representan la interacción entre los componentes del sistema para diferentes casos de uso, y también, se detalla el diseño de la interfaz de usuario, proporcionando una visión clara de la estructura y el funcionamiento del sistema desde una perspectiva técnica.
- **Capítulo 6 - Implemetación**: En este capítulo se describe el proceso de implementación del sistema, detallando la estructura del código y su organización en módulos. Se explican las principales partes del código, incluyendo las funciones y métodos clave.
- **Capítulo 7 - Pruebas**: En este capítulo se presentan las pruebas del sistema, incluyendo las de caja negra y caja blanca. Las pruebas de caja negra verificando que el sistema funcione según los requisitos sin examinar su código interno, mientras que las pruebas de caja blanca analizan el código para asegurar una lógica y ejecución correctas.
- **Capítulo 8 - Conclusiones y trabajo futuro**: En este capítulo se explican las conclusiones del proyecto, destacando los resultados obtenidos y su alineación con los objetivos planteados.Por otro lado, se describen posibles mejoras y trabajo futuro para ampliar sus funcionalidades.
- **Capítulo 9 - Manual de usuario**: En este capítulo se detallan las instrucciones necesarias para el uso del sistema, se explica de manera clara cómo acceder a las distintas funcionalidades, con el objetivo de garantizar una experiencia de usuario intuitiva.

Capítulo 2

Entorno de la aplicación

2.1. Estado del arte

El seguimiento de multicultivos es un área de investigación en constante evolución debido a la necesidad de optimizar los procesos de cultivo y la producción de alimentos en un contexto de creciente demanda y cambio climático. En este sentido, explorando el entorno, podemos ver que existen varias tecnologías y metodologías que podrían ser relevantes para el desarrollo de una aplicación web que realice el seguimiento de multicultivos.

2.1.1. Descripción de trabajos relacionados

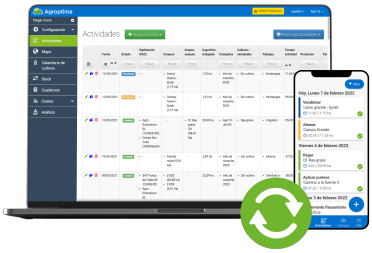
Para lograr los objetivos propuestos en el proyecto de manera satisfactoria, debemos analizar las herramientas que están siendo utilizadas por los agricultores y que están presentes en el mercado.

A continuación, se presentan algunas de las tecnologías y enfoques que podrían ser relevantes para la realización del proyecto:

- **Agroptima**:

  Agroptima es una plataforma [15] de seguimiento de multicultivos agrícola que ayuda a los agricultores a gestionar y optimizar sus operaciones diarias, registrando y monitoreando información sobre sus cultivos. La plataforma ofrece una variedad de herramientas para planificar y gestionar las labores de cultivo, como la siembra, el riego y la fertilización, al funcionar con un almacenamiento de datos en la nube, el usuario puede consultar y modificar toda su información almacenada. Además, proporciona información y análisis meteorológicos en tiempo real para ayudar a los agricultores a tomar decisiones informadas en relación a la gestión de sus cultivos. La plataforma es accesible desde cualquier dispositivo móvil o de escritorio, es decir, consta de una App móvil y una cuenta Web.

Para utilizar esta web se requiere de una cuota anual donde el precio es personalizado y se calcula en función de aspectos como hectáreas, usuarios o funcionalidades.



Agroptima

Figura 2.1: Aplicación agroptima

- **Agrodata**:

  Agrodata es una plataforma tecnológica [9] que ofrece soluciones de gestión y análisis de datos para el sector agrícola, la cual contiene una serie de herramientas para la gestión de cultivos, la toma de decisiones y la optimización de la producción agrícola.

  Ofrece múltiples soluciones entre las que se encuentran el monitoreo de los cultivos, la planificación de siembras y cosechas, la gestión de costos, la identificación de plagas y enfermedades, y la evaluación de la calidad del suelo. Todas estas soluciones están basadas en la recolección y análisis de datos en tiempo real permitiendo a los agricultores optimizar su producción.

  Es accesible tanto en dispositivos móviles como en ordenadores y ofrece servicios de consultoría para la implementación y uso efectivo de sus herramientas tecnológicas.

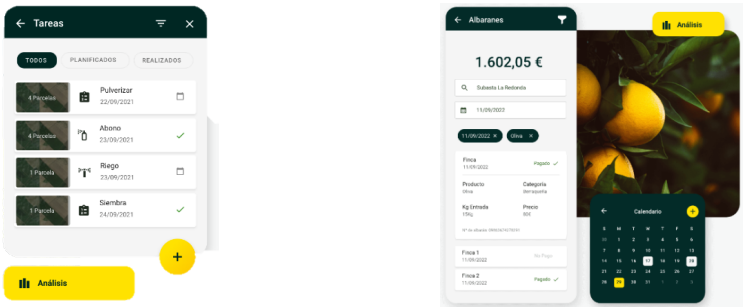


Figura 2.2: Aplicación argrodata

- **VISUAL APP - VisualNACert**:

  Es una aplicación [8] que permite consultar y registrar todas las actividades y tener disponible la plataforma de conectividad con satélites e información agroclimática.

  Ofrece múltiples servicios como catálogo de parcelas y su relación con SIGPAC (Sistema de Información Geográfica de Parcelas Agrícolas), seguimiento satelital de los cultivos con información precisa de lo que ocurre en el campo, gestión de tratamientos, riego y fertilización, trazabilidad y gestión de costes, mano de obra y maquinaria.

  Esta aplicación es accesible tanto para dispositivos moviles (iOS y Andorid).

visual

Figura 2.3: Aplicación Visual App

2.1.2. Discusión

En esta sección vamos a hacer una comparación de los distintos trabajos que hemos investigado y resumido. Así podremos ver las diferencias y similitudes entre ellos de una manera más completa.

Características	Agroptima	Agrodata	Visual App	Mi aplicación
Registro de actividades	✓	✓	✓	✓
Registro de tareas	✓	✓	✓	✓
Control de insumos	✓	x	x	x
Gestión de maquinaria	✓	x	x	x
Análisis de datos	✓	✓	✓	✓
Generación de informes	✓	✓	✓	✓
Interfaz intuitiva	✓	✓	✓	✓
Compatibilidad	✓	✓	✓	✓

Tabla 2.1: Tabla comparativa

La finalidad es desarrollar una aplicación agrícola que combine las mejores características de las aplicaciones existentes, como Agroptima, Agrodata y Visual App, pero con un enfoque centrado en la simplicidad y la usabilidad. Reconocemos que las aplicaciones agrícolas actuales pueden presentar cierta complejidad en su uso, lo cual puede dificultar su utilización y aprovechamiento por parte de los agricultores y profesionales del campo.

Con este proyecto, se busca simplificar la experiencia del usuario y eliminar cualquier complejidad innecesaria. Queriendo ofrecer una interfaz intuitiva y fácil de usar, donde los usuarios puedan realizar el registro de sus actividades, tareas y llevar un cuaderno de campo de manera sencilla y eficiente.

2.2. Herramientas utilizadas

A lo largo del proceso de este proyecto se han utilizado diversas herramientas, tecnologías, plataformas de desarrollo, lenguajes de programación y marcos de trabajo. A continuación, se presenta una lista completa de todas ellas:

Creación y Gestión de Documentos y Diagramas

- Overleaf: es una plataforma en línea [12] que facilita la creación, edición y publicación de documentos científicos, permitiendo a varias personas colaborar en tiempo real. Se utiliza el sistema de composición de textos LaTeX para garantizar la calidad de la presentación del documento.

Figura 2.4: Logo LaTeX

- Draw.io: es una herramienta en línea gratuita para la creación y edición de diagramas y gráficos. Ofrece una interfaz intuitiva y permite la integración con diversas plataformas de almacenamiento en la nube para facilitar el acceso y la gestión de los diagramas.

Figura 2.5: Logo Draw.io

- Balsamiq: es una herramienta de diseño de interfaces de usuario que facilita la creación de wireframes de manera rápida y sencilla [2]. Está orientada a la creación de prototipos de baja fidelidad, permitiendo a los diseñadores y equipos de desarrollo visualizar y comunicar la estructura básica de una aplicación o sitio web. Balsamiq ofrece una amplia biblioteca de componentes predefinidos y opciones de personalización, lo que facilita la iteración rápida y la colaboración entre equipos.

Figura 2.6: Logo Balsamiq

- Microsoft Project: herramienta que facilita la planificación, ejecución y seguimiento de proyectos de manera efectiva y profesional. Diseñado para gestionar proyectos de cualquier tamaño y complejidad, Microsoft Project permite la creación de cronogramas detallados, la asignación de recursos y la monitorización del progreso.

Figura 2.7: Logo Microsoft Project

Editor de Código Fuente y Entorno de Desarrollo

- Visual Studio Code: es un editor de código fuente gratuito y de código abierto desarrollado por Microsoft. Ofrece soporte para múltiples lenguajes de programación y herramientas de desarrollo, y es altamente configurable con extensiones. Facilita la escritura y depuración de código ya que integra características avanzadas como control de versiones y herramientas de colaboración en tiempo real.

Figura 2.8: Logo Visual Studio Code

Lado cliente (*Frontend*)

- HTML: es el estándar fundamental para la creación de páginas web y aplicaciones web. HTML proporciona la estructura básica de los documentos web mediante el uso de una serie de elementos y etiquetas. Estos elementos permiten definir y organizar el contenido de una página web, como texto, imágenes, enlaces y otros recursos.

  Características:

  - **Facilidad de aprendizaje y uso**: HTML es conocido por ser sencillo de aprender y utilizar, lo que lo convierte en el punto de partida ideal para cualquier persona que desee introducirse en el desarrollo web.
  - **Compatibilidad universal**: Casi todos los navegadores web soportan HTML, lo que asegura que los documentos sean accesibles en una amplia variedad de dispositivos y navegadores web.
  - **Estructura clara**: Utiliza etiquetas y elementos que definen la estructura de las páginas web, haciendo que el contenido sea fácilmente entendible tanto para las personas como para las máquinas.

Figura 2.9: Logo HTML5

- CSS: es un lenguaje utilizado para definir la apariencia visual de documentos HTML y XML. Permite controlar cómo se visualizan los elementos en las páginas web, incluyendo colores, fuentes, espaciado, y la disposición de los elementos. Es esencial para crear diseños atractivos y experiencias de usuario consistentes en diferentes dispositivos.

  Características:

  - **Control del diseño**: CSS ofrece un control detallado sobre el diseño y estilo de las páginas web. Desde el color y la fuente hasta la disposición de los elementos, hace que todo sea personalizable.
  - **Mejora en la consistencia**: Al utilizar CSS, se puede mantener una consistencia estilística en todo el sitio web, lo que mejora la experiencia del usuario.
  - **Eficiencia en el mantenimiento**: el estilo de múltiples páginas puede ser controlado desde un solo archivo, lo que hace que las actualizaciones y el mantenimiento sean más eficientes y menos propensos a errores.

Figura 2.10: Logo CSS

- Javascript: es un lenguaje de programación interpretado y de alto nivel que se utiliza principalmente para agregar interactividad y dinamismo a las páginas web. Es un componente esencial del desarrollo web moderno junto con HTML y CSS. JavaScript permite crear efectos dinámicos, validar formularios, manejar eventos de usuario y realizar solicitudes de datos de forma asíncrona mediante AJAX.

Características:

- **Interactividad dinámica**: `JavaScript` permite agregar interactividad a las páginas web, como animaciones, validación de formularios y manejo de eventos, mejorando la experiencia del usuario sin recargar la página.
- **Desarrollo del lado del cliente**: Se ejecuta en el navegador del cliente, lo que permite que las páginas web respondan rápidamente a las acciones del usuario, reduciendo la carga en el servidor y mejorando el rendimiento general.
- **Amplia ecosistema y compatibilidad**: `JavaScript` es compatible con todos los navegadores modernos y tiene un ecosistema robusto de bibliotecas y *frameworks*, que facilitan el desarrollo de aplicaciones web complejas y escalables.

Figura 2.11: Logo Javascript

Lado servidor (Backend)

- PHP: lenguaje de programación de código abierto y del lado del servidor que se utiliza para desarrollar aplicaciones web dinámicas e interactivas. Diseñado inicialmente para la creación de páginas web, PHP se ha convertido en uno de los lenguajes más populares para el desarrollo de aplicaciones web gracias a su facilidad de integración con bases de datos y su capacidad para manejar datos de formularios.

  Una de las principales ventajas de PHP es su capacidad para interactuar con bases de datos, especialmente con MySQL, permitiendo construir aplicaciones que gestionan y procesan grandes volúmenes de datos de manera eficiente. También es altamente flexible y se puede combinar con HTML para crear contenido dinámico en el servidor antes de enviarlo al cliente.

  Además, cuenta con una amplia gama de recursos, incluyendo *frameworks* y bibliotecas que simplifican el desarrollo web. Tiene compatibilidad con servidores web como Apache y Nginx facilitando la implementación en diversos entornos.

Figura 2.12: Logo PHP

Base de datos

- PHPMYADMIN: es una herramienta de código abierto que facilita la administración de bases de datos MySQL a través de una interfaz web. Este software ofrece una interfaz gráfica fácil de usar, que simplifica la interacción con MySQL, permitiendo realizar tareas de gestión de bases de datos de manera intuitiva.

  La aplicación web permite realizar operaciones como la creación, modificación y eliminación de bases de datos, tablas y campos con facilidad.

  PHPMYADMIN ofrece una serie de venatjas para la administración de bases de datos. Entre sus principales beneficios, destaca su capacidad para llevar a cabo todas las tareas de gestión de bases de datos de manera eficiente. La herramienta permite a crear y modificar bases de datos, ajustar la estructura de las tablas y ejecutar consultas SQL con facilidad, lo que simplifica el manejo de datos.

  También permite la importación y exportación de bases de datos, una funcionalidad que resulta interesante en procesos de migración de datos y en la realización de copias de seguridad.

Figura 2.13: Logo phpMyAdmin

Capítulo 3

Planificación y presupuestos

3.1. Metodología

En esta sección se realiza un examen detallado del modelo o metodología de desarrollo seleccionada para llevar a cabo el proyecto, así como la planificación y las estimaciones de costos iniciales. También se proporciona información sobre el costo final real del proyecto, teniendo en cuenta todas las circunstancias que influyeron en su desarrollo.

3.1.1. Proceso de desarrollo

Después de examinar diversas metodologías y enfoques empleados en el desarrollo de software, se ha optado por utilizar el modelo iterativo-incremental para este proyecto en particular. Esta metodología combina las ventajas del modelo en cascada y del modelo de prototipos, lo que permite una evolución gradual del proyecto con iteraciones constantes que permiten la retroalimentación y adaptación constante a los requisitos y cambios del entorno.

Para comprender completamente el proceso de desarrollo incremental e iterativo, visualizamos sus dos partes:

- **Iterativa**: La estrategia iterativa implica que las actividades de desarrollo se repiten sistemáticamente en ciclos conocidos como iteraciones. Después de cada iteración, se produce una nueva versión del software hasta que se logra el producto óptimo.

- **Incremental**: La estrategia incremental en el desarrollo de software consiste en dividir el proceso en etapas pequeñas y manejables llamadas incrementos. Cada incremento se construye sobre la versión anterior, de manera que las mejoras se implementan gradualmente, paso a paso.

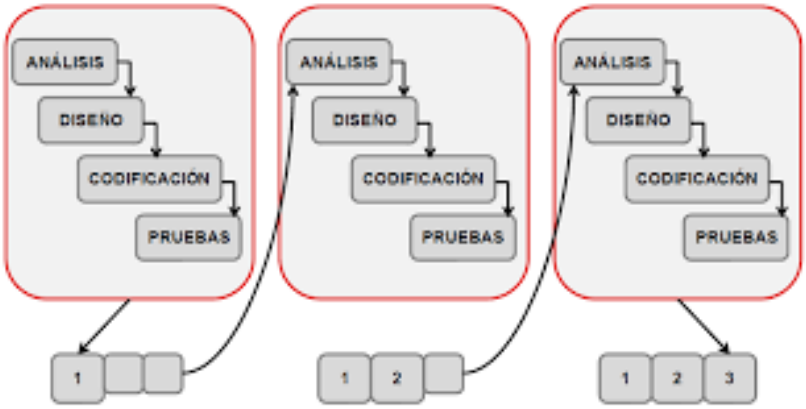


Figura 3.1: Modelo de desarrollo iterativo-incremental

En un proceso de desarrollo iterativo, cada iteración produce una versión completa y funcional del software que sirve como base para la siguiente iteración. Es por eso que cada requisito o característica propuesta para una iteración debe ser completamente desarrollada, probada y documentada, para establecer una base sólida para el siguiente ciclo de desarrollo. Lo que se busca es que en cada iteración los componentes logren evolucionar el producto dependiendo de los completados de las iteraciones antecesoras, agregando más opciones de requisitos y logrando así un mejoramiento mucho más completo. Este enfoque tiene beneficios, aunque también puede haber inconvenientes:

BENEFICIOS

1. Permite adaptarse a los cambios en los requisitos del proyecto, ya que se puede ajustar el enfoque de cada iteración en función de lo que se aprende durante el proceso.

2. Al dividir el proyecto en ciclos, se permite la mejora continua del software, ya que se pueden identificar y corregir errores y problemas a medida que se van descubriendo.

3. Retroalimentación regular: La metodología iterativo-incremental se basa en la retroalimentación constante, lo que permite a los desarrolladores obtener comentarios sobre el software en cada iteración y hacer mejoras en consecuencia.

4. Permite obtener versiones funcionales del software en cada etapa del desarrollo, en lugar de esperar hasta el final para tener una versión completa y funcional del software.

INCONVENIENTES

1. Requiere una planificación cuidadosa: Para que la metodología iterativo-incremental funcione correctamente, se re quiere una planificación cuidadosa de cada iteración, incluyendo los objetivos, el alcance y los recursos necesarios.

2. Implica la participación continua del cliente a lo largo de todo el proyecto

Este proyecto se ha dividido en 6 incremen5os, las cuales se presentan a continuación en orden de desarrollo:

- 1 iteración inicial donde se llevará a cabo un análisis exhaustivo de los requisitos del proyecto.
- 1 iteración para el sistema de gestión de usuarios.
- 1 iteración para el sistema de gestión de parcelas.
- 1 iteración para el sistema de gestión de tratamientos agrícolas aplicados a las parcelas.
- 1 iteración para el sistema de gestión de costes.
- 1 iteración para el sistema de gestión de consultas, que permitirá a los usuarios realizar consultas relacionadas con el proyecto.

3.2. Planificación temporal

La planificación de este proyecto ha tenido en cuenta varios aspectos que tienen un impacto directo en su duración y estimación temporal, como se detalla a continuación:

- El proyecto comenzará el 28 de febrero de 2024 debido a compromisos personales previos, que impidieron iniciarlo antes.
- Se consideran los fines de semana (sábado y domingo) como días no laborables. Además, se establece una jornada laboral de lunes a viernes de 4 horas diarias.
- Se han tomado en cuenta como períodos no laborables para el equipo de trabajo los días festivos y las semanas de vacaciones establecidas en los calendarios laborales a nivel nacional, autonómico y provincial (ver Tabla 3.1).

	Nombre	Comienzo	Fin
1	Jueves Santo	28/03/2024	28/03/2024
2	Viernes Santo	29/03/2024	29/03/2024
1	Día de Castilla y León	23/04/2024	23/04/2024
2	Fiesta del trabajo	01/05/2024	01/05/2024
2	San Juan	25/06/2024	25/06/2024

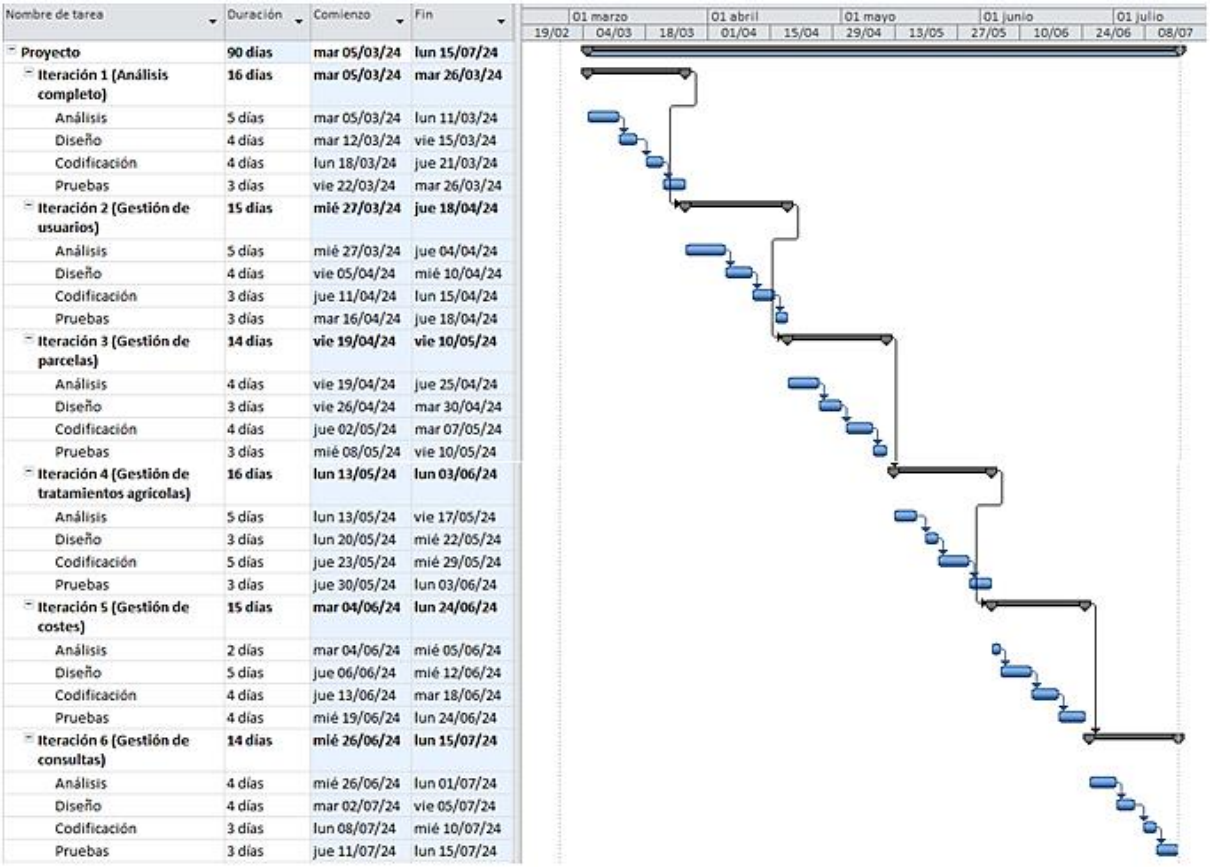
Tabla 3.1: Festividades en el calendario de planificación

- En la estimación temporal, no se ha tenido en cuenta el tiempo destinado a la formación y familiarización de los miembros del equipo de trabajo con las herramientas y tecnologías utilizadas en el proyecto. Se asume que esta capacitación se llevó a cabo previamente al inicio del proyecto, y la experiencia y conocimientos adquiridos se aplican de manera implícita en todas las etapas del desarrollo.

- En las fases iniciales del proyecto, se estima que las tareas de análisis y diseño requieren más tiempo. Esta situación se debe a la metodología empleada en el desarrollo del proyecto, la cual exige un análisis detallado de los requisitos, objetivos y características fundamentales de la herramienta. Esto permite crear los primeros productos que satisfarán las necesidades básicas del cliente. Además, en estas etapas tempranas, el equipo de trabajo adquiere una mayor familiarización con el proyecto y las herramientas utilizadas.

- Durante las etapas intermedias y finales del proyecto, se observa un incremento significativo en la importancia de las tareas de implementación, integración y pruebas. Este aumento se debe a la creciente complejidad en el desarrollo de nuevas funcionalidades, lo cual resulta en la creación de productos más completos que requieren una integración más compleja. Por lo tanto, estos productos demandan pruebas más rigurosas para verificar el cumplimiento de todos los requerimientos establecidos por el cliente.

De acuerdo a las consideraciones mencionadas, se ha realizado una estimación de la duración del proyecto de 90 días laborales, desde el 5 de marzo de 2024 hasta el 15 de julio de 2024. El proyecto se ha dividido en un total de 6 etapas, cada una con una duración aproximada de 15 días. Los detalles específicos de cada etapa, incluyendo las fechas de inicio y finalización, así como la duración estimada en días, se muestran en la tabla 4.2 junto con su correspondiente diagrama de Gantt.



Nombre de tarea	Duración	Comienzo	Fin
Proyecto	**90 días**	**mar 05/03/24**	**lun 15/07/24**
Iteración 1 (Análisis completo)	**16 días**	**mar 05/03/24**	**mar 26/03/24**
Análisis	5 días	mar 05/03/24	lun 11/03/24
Diseño	4 días	mar 12/03/24	vie 15/03/24
Codificación	4 días	lun 18/03/24	jue 21/03/24
Pruebas	3 días	vie 22/03/24	mar 26/03/24
Iteración 2 (Gestión de usuarios)	**15 días**	**mié 27/03/24**	**jue 18/04/24**
Análisis	5 días	mié 27/03/24	jue 04/04/24
Diseño	4 días	vie 05/04/24	mié 10/04/24
Codificación	3 días	jue 11/04/24	lun 15/04/24
Pruebas	3 días	mar 16/04/24	jue 18/04/24
Iteración 3 (Gestión de parcelas)	**14 días**	**vie 19/04/24**	**vie 10/05/24**
Análisis	4 días	vie 19/04/24	jue 25/04/24
Diseño	3 días	vie 26/04/24	mar 30/04/24
Codificación	4 días	jue 02/05/24	mar 07/05/24
Pruebas	3 días	mié 08/05/24	vie 10/05/24
Iteración 4 (Gestión de tratamientos agrícolas)	**16 días**	**lun 13/05/24**	**lun 03/06/24**
Análisis	5 días	lun 13/05/24	vie 17/05/24
Diseño	3 días	lun 20/05/24	mié 22/05/24
Codificación	5 días	jue 23/05/24	mié 29/05/24
Pruebas	3 días	jue 30/05/24	lun 03/06/24
Iteración 5 (Gestión de costes)	**15 días**	**mar 04/06/24**	**lun 24/06/24**
Análisis	2 días	mar 04/06/24	mié 05/06/24
Diseño	5 días	jue 06/06/24	mié 12/06/24
Codificación	4 días	jue 13/06/24	mar 18/06/24
Pruebas	4 días	mié 19/06/24	lun 24/06/24
Iteración 6 (Gestión de consultas)	**14 días**	**mié 26/06/24**	**lun 15/07/24**
Análisis	4 días	mié 26/06/24	lun 01/07/24
Diseño	4 días	mar 02/07/24	vie 05/07/24
Codificación	3 días	lun 08/07/24	mié 10/07/24
Pruebas	3 días	jue 11/07/24	lun 15/07/24

Figura 3.2: Diagrama de Gantt

3.3. Estimación del esfuerzo

En esta etapa, vamos a realizar una primera estimación del proyecto. Para ello, utilizaremos dos métodos ampliamente utilizados en el desarrollo de software: el método COCOMO y el método Albrecht con puntos de función.

Para finalizar, se realizará una explicación detallada y una comparación de los resultados obtenidos en todos los procedimientos utilizados. Esto incluirá una evaluación de la planificación inicial propuesta y cómo se ajusta a la visión temporal y presupuestaria del proyecto.

3.3.1. Estimación por puntos de función (Método Albretch)

La estimación por puntos de función, método propuesto por Albrecht, se basa en la medición funcional de un sistema de software. En lugar de enfocarse en la implementación técnica o en las líneas de código, se centra en las funciones o características del software que brindan valor al usuario.

El método de puntos de función considera varios factores para determinar el tamaño funcional del sistema:

- Entradas de usuario: Se refiere a toda la información que los usuarios ingresan o introducen en el sistema. Estas entradas son necesarias para que el sistema realice las acciones correspondientes.
- Salidas de usuario: Son los resultados o información que el sistema presenta al usuario una vez que ha procesado su solicitud o petición.
- Consultas: Son las peticiones o solicitudes que el usuario realiza al sistema para obtener una respuesta o información específica.
- Ficheros lógicos internos: Son los archivos o estructuras de datos que el sistema utiliza de forma interna para realizar sus operaciones. Estos archivos son exclusivos del sistema y no son accesibles directamente por los usuarios.
- Ficheros lógicos externos: Son los archivos externos que el sistema utiliza para el procesamiento de datos. Estos archivos contienen información relevante para el sistema, pero no son exclusivos de él, es decir, son creados o mantenidos por otros sistemas o usuarios. El sistema los utiliza para obtener o almacenar datos necesarios para su funcionamiento.

Cada una de estas funciones se clasifica y se les asigna un peso según su complejidad y el esfuerzo requerido para implementarlas. Para ello se utilizará la siguiente tabla:

Entradas externas				**Salidas externas**				**Grupos lógicos de datos internos y Grupos lógicos de datos de interfaz**			
	Tipos de datos elementales				**Tipos de datos elementales**				**Tipos de datos elementales**		
Ficheros referenciados	1 a 4	5 a 15	>16	**Ficheros referenciados**	1 a 5	6 a 19	>20	**Tipos de registros**	1 a 19	20 a 50	>51
0 ó 1	B	B	M	0 ó 1	B	B	M	1	B	B	M
2	B	M	A	2 ó 3	B	M	A	2 a 5	B	M	A
3 ó más	M	A	A	4 ó más	M	A	A	6 ó más	M	A	A

Tabla 3.2: Complejidad de funcionalidades

Utilizando la tabla de clasificación de complejidad de funcionalidades, se procede a asignar las funcionalidades a cada uno de los grupos previamente definidos:

Entradas de usuario:

Descripción	Complejidad
Formulario de datos de usuario	Baja
Formulario de parcelas	Baja
Editar información de la parcela	Media
Formulario de costes	Media
Formulario de tratamientos	Baja

Tabla 3.3: Entradas de usuario

Salidas de usuario:

Descripción	Complejidad
Notificación de confirmación de registro.	Baja
Notificación de error en las diferentes gestiones	Media
Visualización de la lista de parcelas y sus detalles.	Baja
Visualización de la información del tratamiento agrícola.	Baja
Visualización de la lista de tratamientos.	Media
Visualización de los costes.	Baja

Tabla 3.4: Salidas de usuario

Consultas:

Descripción	Complejidad
Consulta de listado de parcelas.	Baja
Consulta de parcela por municipio.	Baja
Consulta de tratamientos agrícolas aplicados a un cultivo específico.	Baja
Consulta de tratamientos de parcelas según el tipo y fecha	Media
Consulta de costes por tipo de coste y fecha.	Baja
Consulta de coste por temporada	Media
Consulta de coste por parcela	Media

Tabla 3.5: Consultas

Ficheros lógicos internos:

Descripción	Complejidad
Ficheros de registro de usurios	Baja
Fichero de parcelas	Baja
Fichero de tratamientos agrícolas.	Media
Fichero de costes	Baja

Tabla 3.6: Ficheros lógicos internos

Después de haber definido y evaluado todas las complejidades necesarias del sistema, el siguiente paso es calcular los Puntos de Función Sin Ajustar (PFNA). Para realizar este cálculo, utilizaremos la Tabla 3.7 que se muestra a continuación:

Dominio	Complejidad	Peso por complejidad	Número total de funciones	Total
Entradas de usuario	Baja	x3	3	9
	Media	x4	2	8
	Alta	x6	0	0
Salidas de usuario	Baja	x4	4	16
	Media	x5	2	10
	Alta	x7	0	0
Consultas	Baja	x3	4	12
	Media	x4	3	12
	Alta	x6	0	0
Ficheros lógicos internos	Baja	x7	3	21
	Media	x10	1	10
	Alta	x15	0	0
Ficheros lógicos externos	Baja	x5	1	0
	Media	x7	0	0
	Alta	x10	0	0
Puntos de función sin ajustas (PFNA)				**97**

Tabla 3.7: Puntos de función sin ajustar

Una vez que hemos calculado los puntos de función sin ajustar, procedemos a analizar el Factor de Ajuste (FA). El Factor de Ajuste se determina considerando una serie de factores que evalúan la funcionalidad y la complejidad general del sistema. Para obtener el valor del Factor de Ajuste, es necesario asignar un peso a cada uno de estos factores en una escala del 0 al 5, donde 5 representa el valor más alto y 0 el valor más bajo.

La tabla siguiente muestra la representación de la complejidad de los Factores de Ajuste para este proyecto.

Factor de ajuste	Complejidad
Comunicación de datos	4
Funciones distribuidas	2
Prestaciones	3
Gran uso de la configuración	3
Velocidad de las transacciones	4
Entrada online de datos	3
Diseño para la eficiencia del usuario final	4
Actualización de datos online	1
Complejidad de procesos lógicos internos	3
Reusabilidad del código	3
Facilidad de instalación	1
Facilidad de operación	1
Localizaciones múltiples	1
Facilidad de cambios	3
Total:	**36**

Tabla 3.8: Factores de ajuste

Para calcular el factor de ajuste (FA), se utiliza la fórmula siguiente:

$$FA = 0{,}65 + 0{,}01 \times \text{Total del factor de complejidad} \tag{3.1}$$

Sustituyendo los valores:

$$\begin{aligned} FA &= 0{,}65 + 0{,}01 \times 36 \\ &= 0{,}65 + 0{,}36 \\ &= 1{,}01 \end{aligned}$$

Para calcular los puntos de función ajustados a partir del factor de ajuste y los puntos de función no ajustados, se utiliza la siguiente fórmula:

$$PFA = PFNA \times PFA \tag{3.2}$$

Sustituyendo los valores:

$$\begin{aligned} PFA &= 97 \times 1{,}01 \\ &= 97{,}97 \end{aligned}$$

Para calcular la estimación de la duración, se establece que cada punto de función ajustado tiene un equivalente de 4 horas de trabajo. Esto significa que se asignan 4 horas de esfuerzo por cada punto de función ajustado en el proyecto.

$$\text{Duración del proyecto aproximada} = PFA \times 4h \tag{3.3}$$

Sustituyendo los valores:

$$\begin{aligned} \text{Duración} &= 97{,}97 \times 4 \\ &= 391{,}88 \text{ horas} \end{aligned}$$

3.3.2. Estimación por COCOMO

Este tipo de estimación se fundamenta en el tamaño de líneas de código obtenidos a partir de los valores en la estimación por puntos de función realizada anteriormente utilizando el Método Albretch. Estas mediciones representan el tiempo y esfuerzo dedicados al desarrollo del sistema.

Antes de comenzar con el cálculo, es importante conocer las diferentes tipologías de COCOMO que se pueden aplicar en función de las características específicas del proyecto que se desea llevar a cabo. Estas tipologías incluyen:

- Modelo Orgánico: Se aplica a proyectos de pequeño tamaño, con baja complejidad y requisitos bien definidos. Estos proyectos suelen ser desarrollados por equipos pequeños.
- Modelo Empotrado: Se utiliza en proyectos de gran tamaño, con alta complejidad y requisitos muy estrictos. Estos proyectos suelen requerir una gran cantidad de recursos y un equipo multidisciplinario.
- Modelo Semiempotrado: Este modelo se sitúa entre los dos anteriores y se aplica a proyectos con características intermedias. Los equipos de desarrollo se coordinan para llevar a cabo el desarrollo de manera eficiente.

Después de analizar cada uno de los modelos disponibles, he llegado a la conclusión de que el modelo semiempotrado es el más adecuado para este proyecto. Aunque el proyecto no es de gran escala, su complejidad puede presentar desafíos significativos.

Para realizar la estimación utilizando el modelo COCOMO, es necesario estimar el número de líneas de código que se requerirán en el proyecto. En este caso, utilizaremos el lenguaje de programación PHP para el desarrollo de la aplicación.

Según la equivalencia establecida, en promedio se necesitan aproximadamente 53 líneas de código por cada punto de función en el lenguaje PHP. Utilizando esta relación, podemos estimar el número total de líneas de código para el proyecto de la siguiente manera:

$$LDC = PFA \times \text{estimación de líneas} \tag{3.4}$$

Sustituyendo los valores:

$$\begin{aligned} \text{LDC} &= 97,97 \times 53 \\ &= 5192,41 \end{aligned}$$

A continuación, se presentan las 15 características que se utilizan como factores de esfuerzo en el modelo COCOMO para determinar la estimación de esfuerzo en función de las líneas de código:

	Valor de los factores					
Factores	**Muy bajo**	**Bajo**	**Medio**	**Alto**	**Muy alto**	**Extra**
Fiabilidad requerida	0.75	0.88	1.00	1.15	1.4	
Tamaño de la base de datos		0.94	1.00	1.08	1.16	
Complejidad del software	0.7.	0.85	1.00	1.15	1.30	1.65
Restricciones de tiempo de ejecución			1.00	1.11	1.30	1.66
Restricciones de memoria			1.00	1.06	1.21	1.56
Volatilidad del hardware		0.87	1.00	1.15	1.30	
Restricciones de tiempo de respuesta		0.87	1.00	1.07		
Calidad de los analistas	1.46	1.19	1.00	0.86	0.71	
Experiencia con el tipo de aplicación	1.29	1.13	1.00	0.91	0.82	
Experiencia con el hardware	1.21	1.10	1.00	0.90		
Experiencia con el lenguaje de programación	1.14	1.07	1.00	0.95		
Calidad de los programadores	1.42	1.17	1.00	0.86	0.70	
Técnicas modernas de programación	1.24	1.10	1.00	0.91	0.82	
Empleo de herramientas	1.24	1.10	1.00	0.91	0.83	
Restricciones a la duración del proyecto	1.23	1.08	1.00	1.04	1.10	

Tabla 3.9: Factores de ajuste de COCOMO

El valor de ajuste correspondiente a cada una de las características se encuentra reflejado en la Tabla 3.10

Fiabilidad Requerida	Alto	1,15
Tamaño de la base de datos	Medio	1,00
Complejidad del software	Alto	1,15
Restricciones en tiempo de ejecución	Medio	1,00
Restricciones de memoria	Medio	1,00
Volatilidad del Hardware	Bajo	0,87
Restricciones de tiempo de respuesta	Alto	1,07
Calidad de los analistas	Medio	1,00
Experiencia con el tipo de aplicación	Medio	1,00
Experiencia con el hardware	Alto	0,90
Experiencia con el lenguaje de programación	Medio	1,00
Calidad de los programadores	Medio	1,00
Técnicas modernas de programación	Alto	0,91
Empleo de herramientas	Alto	0,91
Restricciones a la duración del proyecto	Medio	1,00

Tabla 3.10: Calificación factores de ajuste

Para calcular el valor "mxütilizado en la estimación del esfuerzo del proyecto (E) y el tiempo de desarrollo (T_{DEV}), se multiplican los valores correspondientes de la tabla entre sí. Estos valores se obtienen a partir de las constantes propuestas en la Tabla 3.11, que dependen del modelo de COCOMO elegido.

Modelo	**a**	**b**	**c**	**d**
Orgánico	2,40	1,05	2,50	0,38
Semi- Empotrado	3,00	1,12	2,50	0,35
Empotrado	3,60	1,20	2,50	0,32

Tabla 3.11: Ponderaciones de COCOMO para los distintos modelos

$$\begin{aligned} m_x &= 1,15 \times 1,00 \times 1,15 \times 1,00 \times 1,00 \times 0,87 \times 1,07 \times 1,00 \times 1,00 \times 0,90 \\ &\quad \times 1,00 \times 1,00 \times 0,91 \times 0,91 \times 1,00 \\ &= 0,917 \end{aligned}$$

$$Esfuerzo(E) = a \times (KLDC)^b \times m_x \tag{3.5}$$

Sustituyendo los valores:

$$E = 3 \times 5,192^{1,12} \times 0,917$$
$$= 17,40 \ \ personas/mes$$

$$\text{Tiempo de desarrollo } T_{DEV} = c \times E^d \tag{3.6}$$

Sustituyendo los valores:

$$T_{DEV} = 2,50 \times 17,40^{0,35}$$
$$= 6,79 \ \ meses$$

Una vez que hemos obtenido el esfuerzo y el tiempo de desarrollo del proyecto, calculamos el esfuerzo nominal. El esfuerzo nominal determina el número aproximado de personas necesarias para llevar a cabo el proyecto en un período de casi 7 meses.

$$N = E/T_{DEV} \tag{3.7}$$

Sustituyendo los valores:

$$N = 2,50 \times 17,40/6,79$$
$$= 2,56 \ \cong \ 3 \ \ personas$$

Según el cálculo del esfuerzo nominal, se requerirían aproximadamente 3 personas para completar el proyecto en el período estimado aproximado de 7 meses.

3.3.3. Comparativa de estimaciones

Una vez obtenidas las estimaciones utilizando los métodos de puntos de función y COCOMO, podemos concluir que el método de puntos de función proporciona una estimación más realista para este proyecto en comparación con el método COCOMO.

La estimación del método COCOMO resulta excesiva, lo cual se debe a que COCOMO está diseñado principalmente para lenguajes de programación de bajo nivel y, por lo tanto, sobreestima la duración del proyecto en este caso particular.

3.4. Presupuesto económico

Una vez realizados todos los análisis y estimaciones iniciales del proyecto, llega el momento de calcular los presupuestos tanto los iniciales como los posteriores del desarrollo del proyecto.

Para llevar a cabo esta tarea, se ha elaborado un presupuesto, desglosado en tres categorías: hardware, software y recursos humanos (RRHH). Es importante tener en cuenta

que para calcular el costo real asociado al uso de hardware y software se debe utilizar una fórmula que considere la frecuencia de uso, el tiempo empleado y el costo mensual de estas herramientas. A continuación, se presenta esta fórmula, junto con las unidades de medida correspondientes:

$$\text{Coste por mes } (\frac{€}{\text{mes}}) = \frac{\text{Coste total } €}{\text{Vida útil (meses)}} \tag{3.8}$$

$$\text{Coste real por mes } (\frac{€}{\text{mes}}) = \text{Coste por mes } (\frac{€}{\text{mes}}) \times \text{Porcentaje de uso} \tag{3.9}$$

$$\text{Coste real } € = \text{Coste real por mes } (\frac{€}{\text{mes}}) \times \text{Tiempo de uso (meses)} \tag{3.10}$$

3.4.1. Presupuesto inicial del proyecto

Obtenida la estimación más precisa para el proyecto, en esta etapa se realiza el cálculo de los presupuestos utilizando la planificación inicial y la estimación basada en puntos de función utilizando el Método de Albrecht.

Hardware

Para llevar a cabo el proyecto, se utilizará un ordenador personal de gama media con las siguientes especificaciones: procesador i5-5200U, sistema operativo de 64 bits, 8 GB de RAM y 1 TB de memoria HDD, además de un ratón con 1000DPI, una pantalla de 27" y un dispositivo móvil.

Debido a estos factores, el coste total relacionado con el hardware es de 5'99 euros y se divide en función de las especificaciones que se detallan en la Tabla 3.12

Componente	Coste total (€)	Vida útil (meses)	Uso (%)	Uso (meses)	Coste Real (€)
Ordenador portátil	711,59	5 ·12 = 60	80	4	37,95
Ratón	17,98	3 ·12 = 36	70	4	1,39
Pantalla	179,99	5 ·12 = 72	60	4	6,00
Dispositivo móvil	399,00	4 ·12 = 48	20	4	6,65
				Total:	51,99

Tabla 3.12: Total inicial costes hardware

Software

No hay coste alguno asociado al software ya que todas las herramientas utilizadas tienen una licencia gratuita para estudiantes, tal y como se detalla en la Tabla 3.13

Herramienta	Coste (€)	Uso (%)	Uso (meses)	Coste Real (€)
Overleaf	0	50	4	0,00
Notepad ++	0	30	4	0,00
Visual Studio Code	0	60	4	0,00
StarUML	0	30	4	0,00
Draw.io	0	20	4	0,00
Microsoft 365	5,60€/mes	30	4	0,00
			Total:	0,00

Tabla 3.13: Total inicial costes software

Recursos humanos

Estos costos corresponden a todos los gastos relacionados con el personal que participa en las diversas etapas de planificación, tal como se indica en el diagrama de Gantt presentado anteriormente. Dado que este proyecto solo cuenta con un recurso humano que desempeña un papel activo en todas las fases del proyecto, es decir, el autor, este último asumirá diferentes roles según la fase y la tarea que se esté llevando a cabo. Esto se hace para proporcionar una estimación más precisa y realista de los costos involucrados. Los roles que el autor asume en el proyecto son los siguientes:

- **Analista**: persona responsable de convertir las necesidades y requerimientos del cliente en especificaciones técnicas para el equipo de desarrollo. También identifica riesgos y oportunidades de mejora.
- **Diseñador UX/UI**: persona responsable de crear una experiencia de usuario satisfactoria al diseñar interfaces de usuario y diseñar la interacción del usuario con el producto. Se asegura de que el diseño del producto sea coherente con la marca y la identidad visual de la empresa.
- **Desarrollador**: persona responsable de escribir el código y construir la funcionalidad del producto.
- **Tester**: persona responsable de garantizar la calidad del producto final mediante la identificación de errores o defectos en el software, documentando cualquier error encontrado. También realiza pruebas de rendimiento y de seguridad para garantizar que el producto sea seguro y funcione bien en diferentes situaciones y condiciones.

Se trabajará a media jornada, es decir, veinte horas semanales. Teniendo en cuenta que disponemos de noventa días laborables, esto resulta en un total de 360 horas de trabajo para el desarrollo de la aplicación.

En la Universidad, el Trabajo de Fin de Grado está calculado como 12 ECTS (Créditos), donde cada ECTS equivale a un rango de 25-30 horas. Por lo tanto, el rango total requerido oscila entre 300 y 360 horas.

Dado que este proyecto está estimado en 360 horas, se encuentra dentro del rango estipulado por la Universidad, cumpliendo así con los requisitos académicos establecidos.

Por esta razón, su sueldo bruto será la suma del sueldo bruto obtenido por cada rol adoptado, es decir, 5357,30€, como indica el total de la Tabla 3.14

Rol	Sueldo (€/hora)	Tiempo (horas)	Total (€)
Analista	16,35	100	1635,00
Diseñador UX/UI	14,42	75	1081,50
Desarrollador	14,90	145	2160,00
Tester	12,02	40	480,80
		Total:	5357,30

Tabla 3.14: Total inicial costes personal

Es esencial tener en cuenta el costo adicional de registrar a esta persona en la Seguridad Social. Este costo se calcula como el 30,9 % del salario bruto, que es 0,309 multiplicado por la suma de los salarios, resultando en un total de 0,309 * 5357,30 = 1655,40€. Este porcentaje se aplica debido a que el contrato es indefinido. En consecuencia, el costo total relacionado con el personal asciende a 7012,70€, desglosado conforme a lo indicado en la Ecuación 3.11.

$$\text{Presupuesto RRHH} = 5357{,}30\ \text{(Sueldo bruto)} \times 1655{,}4\ \text{(Cotizaciones)} \tag{3.11}$$

Presupuesto total inicial

Según este estudio, se estima que el coste total del proyecto es de 6296,99€. De esta cantidad, 51,99€ corresponden al uso de recursos hardware y 7012,70€ corresponden a los costes de personal, tal como se detalla en la Tabla 3.15.

Coste real del proyecto (fase inicial)			
Hardware (€)	Software (€)	RRHH (€)	Total (€)
51.99	0,00	7012,70	7064,69

Tabla 3.15: Presupuesto total inicial

3.4.2. Presupuesto final del proyecto

Tras acabar el proyecto, se lleva a cabo un análisis del costo real teniendo en cuenta factores que han afectado la planificación inicial durante su ejecución.

La fecha de finalización del proyecto se ha extendido hasta el 17 de septiembre de 2024, lo que representa un retraso de 47 días laborables en comparación con la fecha prevista originalmente del 15 de julio de 2024. Los principales factores que han contribuido a este retraso incluyen:

- Experiencia limitada en la planificación del proyecto.
- Compatibilidad con las responsabilidades laborales.
- Aumento de una semana debido a problemas en la gestión de los tratamientos específicos.
- Mayor número de horas dedicadas a la revisión de la documentación.

Este retraso de 47 días ha generado un aumento de aproximadamente 188 horas, aunque los costes iniciales de software no han sufrido cambios debido a que son herramientas de software libre o con licencia de estudiante, sí ha modificado el presupuesto hardware y de recursos humanos.

El presupuesto de hardware presentado en la Tabla 3.12 ha sido actualizado, reflejando los cambios mostrados en la Tabla 3.16. De esta manera, el coste total real asociado al hardware asciende a 78€. Esto resulta en una diferencia de 26,01€ entre el costo planificado y el costo real.

Componente	Coste total (€)	Vida útil (meses)	Uso (%)	Uso (meses)	Coste Real (€)
Ordenador portátil	711,59	5 ·12 = 60	80	6	56,93
Ratón	17,98	3 ·12 = 36	70	6	2,09
Pantalla	179,99	5 ·12 = 72	60	6	9,00
Dispositivo móvil	399,00	4 ·12 = 48	20	6	9,98
				Total:	78,00

Tabla 3.16: Total final costes hardware

Para evaluar con mayor precisión la diferencia de costes en el presupuesto de RRHH, se considerará como se ha mencionado anteriormente, que el estudiante ha aumentado 188 horas. Por lo tanto, la tabla de presupuesto de RRHH (Tabla 3.14) ha sido ajustada como se muestra en la Tabla 3.17.

Como resultado, el sueldo bruto será la suma de los ingresos brutos correspondientes a cada rol desempeñado, lo que equivale a 8151,20 €, según se detalla en la Tabla 3.17.

Rol	Sueldo (€/hora)	Tiempo (horas)	Total (€)
Analista	16,35	152	2481,20
Diseñador UX/UI	14,42	114	1643,88
Desarrollador	14,90	221	3292,90
Tester	12,02	61	733,22
		Total:	8151,20

Tabla 3.17: Total final costes personal

Es esencial tener en cuenta el costo adicional de registrar a esta persona en la Seguridad Social. Este costo se calcula como el 30,9 % del salario bruto, que es 0,309 multiplicado por la suma de los salarios, resultando en un total de 0,309 * 8151,20 = 2519,02€. Este porcentaje se aplica debido a que el contrato es indefinido. En consecuencia, el costo total relacionado con el personal asciende a 10670,22€, desglosado conforme a lo indicado en la Ecuación 3.12.

$$\text{Presupuesto real RRHH} = 8151{,}20\ \text{(Sueldo bruto)} + 2519{,}02\ \text{(Cotizaciones)} \quad (3.12)$$

Por lo tanto, la diferencia entre lo planificado y lo realmente obtenido es de 10670,22€ menos 7012,70 € lo que resulta en 3657,52€

Presupuesto total final

De acuerdo a estos nuevos datos, se calcula que el coste real total del proyecto asciende a 10748,22€. De esta suma, 78,00€ están asociados al uso de recursos de hardware, mientras que 10670,22€ corresponden a los gastos de personal, como se detalla en la Tabla 3.18.

Coste real del proyecto (fase inicial)			
Hardware (€)	Software (€)	RRHH (€)	Total (€)
78,00	0,00	10670,22	10748,22

Tabla 3.18: Presupuesto total final

Por lo tanto, en términos generales, la diferencia entre el coste estimado y el coste real es de 10670,22€ - 7064,69€ lo que da una diferencia de 3605,53€.

Parte II

Documentación técnica

Capítulo 4

Análisis del sistema

4.1. Descripción de actores

Un actor en el contexto de un sistema se define como cualquier agente externo que interactúa con dicho sistema, y puede ser tanto una persona como una máquina.

En nuestro proyecto, contaremos con un único actor:

- **Usuario**: Este usuario tiene acceso a las funcionalidades del sistema, se puede registrar, gesitionar parcelas y los datos relativos a ellas, permitiéndole consultar y utilizar la información relevante pero sin poder realizar tareas de administración.

4.2. Requisitos de usuario

En esta parte, se detallarán los requisitos de usuario, entendidos como las acciones que un actor puede realizar dentro del sistema. Es decir, se trata de la descripción de las capacidades y funciones que el usuario puede ejecutar. A continuación, en la tabla 4.1 se enumeran los requisitos de usuario del sistema:

Requisito	Descripción
RU-01	El usuario puede añadir parcelas
RU-02	El usuario puede visualizar un listado de todas las parcelas
RU-03	El usuario puede ver detalles de cada parcela
RU-04	El usuario puede eliminar las parcelas
RU-05	El usuario puede modificar los datos de la parcela
RU-06	El usuario puede filtrar las parcelas por el término

Requisito	Descripción
RU-07	El usuario puede añadir tratamientos específicos a cada parcela
RU-08	El usuario puede iniciar sesión en la aplicación
RU-09	El usuario puede visualizar un listado con todos los tratamientos específicos
RU-10	El usuario puede filtrar el listado de tratamientos específicos por tipo de tratamiento y por año
RU-11	El usuario puede acceder a los detalles del tratamiento específico
RU-12	El usuario puede modificar los detalles del tratamiento específico
RU-13	El usuario puede añadir diferentes tipos de costes relacionados con los tratamientos
RU-14	El usuario puede ver el gasto total
RU-15	El usuario puede ver el gasto total por distintos años, temporadas y tipo
RU-16	El usuario puede ver una estimación del gasto por parcela en relación a su superficie
RU-17	El usuario puede visualizar un listado de todos los costes
RU-18	El usuario puede filtrar los costes por año y por tipo de coste
RU-19	El usuario puede modificar los datos de cada coste específico
RU-20	El usuario puede eliminar el coste
RU-21	El usuario puede visualizar su información de perfil
RU-22	El usuario puede modificar su información de perfil
RU-23	El usuario puede eliminar su perfil
RU-24	El usuario puede cerrar sesión
RU-25	El usuario puede ver la información de la política de privacidad y cookies
RU-25	El usuario puede crearse una cuenta y quedar registrado

Tabla 4.1: Requisitos de usuario

4.3. Casos de uso

En el ámbito de la Ingeniería de Software, los casos de uso se emplean como una estrategia eficaz para delinear los requisitos funcionales de un sistema desde el punto de

vista de los usuarios. Un caso de uso detalla cómo un usuario (actor) interactúa con el sistema, describiendo, paso a paso, el proceso que tiene lugar durante esa interacción. Esta técnica es esencial para establecer y entender claramente cómo debería comportarse el sistema, asegurando que se satisfagan las demandas y expectativas de quienes lo utilizan.

En la subsección 4.3.1 se presentará el diagrama completo de casos de uso, mientras que en la subsección 4.3.2 se detallarán las tablas de casos de uso específicos.

4.3.1. Diagrama de Casos de Uso

Una vez completado el resumen de los requisitos del usuario, se va a mostrar una especificación más detallada de estos requerimientos. Para lograr una representación más clara y comprensible de las funcionalidades y las interacciones esperadas, emplearemos el diagrama de casos de uso. Estos diagramas proporcionan una representación gráfica de los diferentes escenarios en los que los usuarios interactúan con el sistema, permitiendo así una interpretación más precisa y completa de las posibles acciones y comportamientos del usuario.

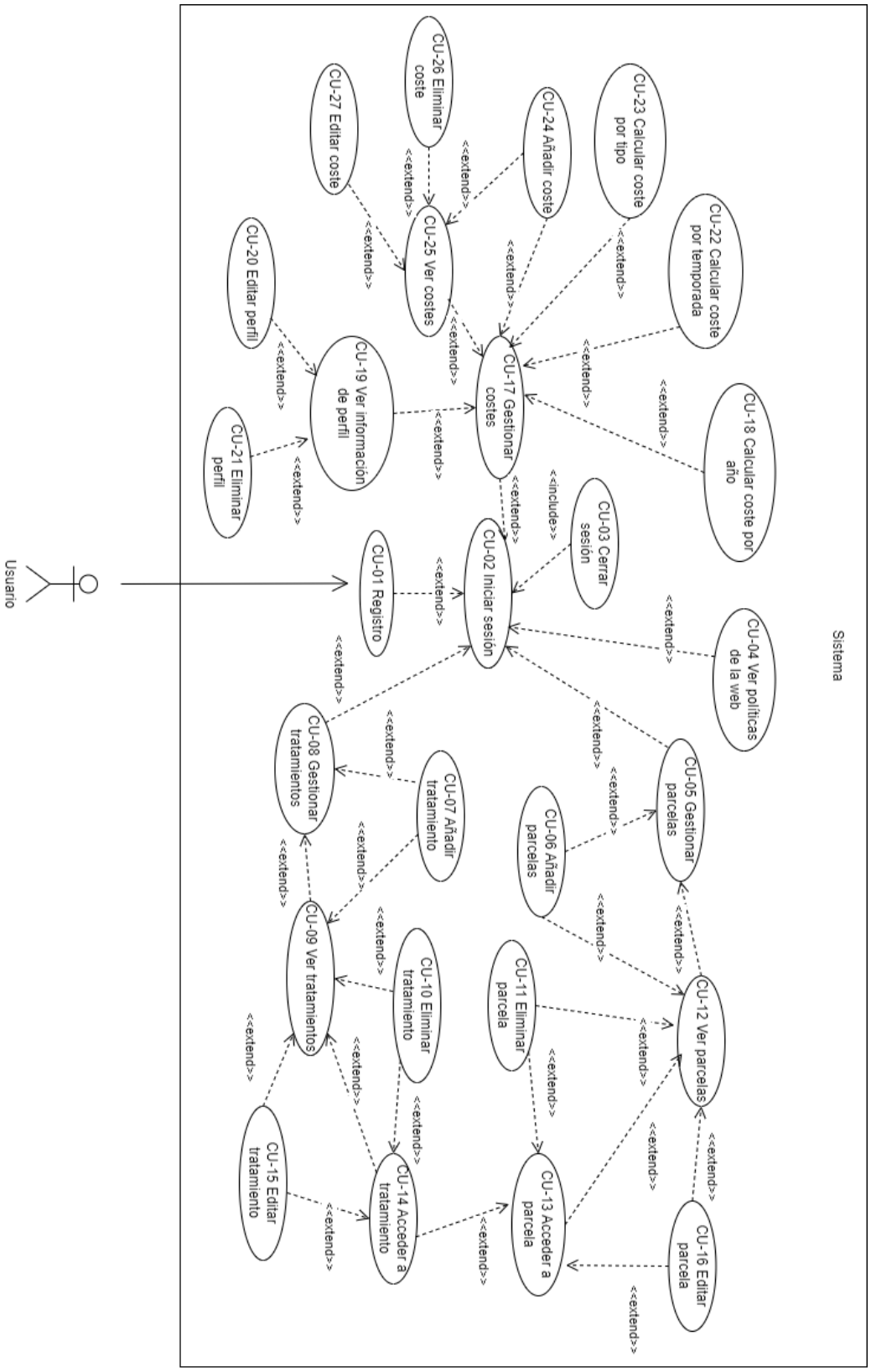


Figura 4.1: Diagrama de Casos de Uso

4.3.2. Especificacion de casos de uso

A continuación, se presentan varios casos de uso específicos para el sistema, cada uno reflejando una función particular desde la perspectiva del usuario. He seleccionado solo los más relevantes para mantener la claridad y evitar la sobrecarga de información, ofreciendo un análisis detallado de actores, condiciones previas, flujo de eventos y resultados.

CU-01	Registro
Actor	Usuario
Requisitos asociados	RU-25
Descripción	El usuario navega a la página de registro, ingresa su información personal y crea una cuenta en la plataforma.
Precondiciones	PRE-1. El usuario no debe tener una cuenta registrada.
Postcondiciones	POST-1. Se crea y almacena la cuenta del usuario en la base de datos del sistema.
Flujo normal	▪ FN1. El usuario accede a la página de registro. ▪ FN2. El usuario ingresa su información personal. ▪ FN3. El sistema valida los datos del usuario. ▪ FN4. El sistema registra la cuenta y notifica que el registro fue exitoso.
Flujos alternativo	▪ FA1. Si los datos son incorrectos o están incompletos, se le solicita que realice las correcciones necesarias.
Excepciones	▪ E1. El correo electrónico ya existe en el sistema. ▪ E2. Campos obligatorios no completados. ▪ E3. El sistema no está disponible temporalmente.
Prioridad	Alta

Tabla 4.2: Caso de Uso: CU-01 Registro

CU-02	Iniciar sesión
Actor	Usuario
Requisitos asociados	RU-08, RU-24
Descripción	El usuario ingresa sus credenciales correo y contraseña) para acceder al sistema.
Precondiciones	PRE-1. El usuario debe estar registrado en el sistema.
Postcondiciones	POST-1. El usuario accede a su perfil y puede utilizar las funcionalidades del sistema.
Flujo normal	▪ FN1. El usuario navega a la página de inicio de sesión. ▪ FN2. El usuario introduce su correo y contraseña. ▪ FN3. El sistema valida las credenciales. ▪ FN4. El sistema permite el acceso y redirige al usuario a la página principal.
Flujo alternativo 1	▪ FA1. Si las credenciales son incorrectas, se muestra un mensaje de error y se solicita al usuario que intente nuevamente.
Excepciones	▪ E1. El sistema está temporalmente fuera de servicio.
Prioridad	Alta

Tabla 4.3: Caso de Uso: CU-02 Iniciar sesión

CU-05	Gestionar parcelas
Actor	Usuario
Requisitos asociados	RU-01, RU-02, RU-03, RU-04, RU-05, RU-06
Descripción	El usuario puede acceder a las opciones de añadir, editar, eliminar y ver detalles de las parcelas registradas en el sistema.
Precondiciones	PRE-1. El usuario debe estar autenticado.
Postcondiciones	POST-1. Las parcelas gestionadas se actualizan correctamente en la base de datos.
Flujo normal	▪ FN1. El usuario accede a la sección de gestión de parcelas. ▪ FN2. El usuario selecciona la acción deseada (añadir, editar, eliminar o ver detalles). ▪ FN3. El sistema procesa la acción solicitada y muestra la confirmación.
Flujo alternativo	▪ FA1. Si el usuario intenta eliminar una parcela o cancelar alguna acción, se solicita una confirmación antes de proceder o vuelve a la página anterior.
Excepciones	▪ E1. El sistema no puede conectar con la base de datos.
Prioridad	Alta

Tabla 4.4: Caso de Uso: CU-05 Gestionar parcelas

CU-17	Gestionar costes
Actor	Usuario
Requisitos asociados	RU-13, RU-14, RU-15, RU-16, RU-17, RU-18, RU-19, RU-20
Descripción	El usuario puede acceder a añadir, editar, eliminar y consultar costes asociados a las parcelas.
Precondiciones	PRE-1. El usuario debe estar autenticado.
Postcondiciones	POST-1. Los costes gestionados se reflejan correctamente en la base de datos.
Flujo normal	▪ FN1. El usuario accede a la sección de gestión de costes. ▪ FN2. El usuario selecciona la acción deseada (añadir, editar, eliminar o consultar). ▪ FN3. El sistema ejecuta la acción solicitada y muestra una confirmación.
Flujo alternativo	▪ FA1. Si el usuario intenta eliminar un coste o cancelar alguna acción, se solicita una confirmación antes de proceder o vuelve a la página anterior.
Excepciones	▪ E1. El sistema no puede recuperar los datos solicitados.
Prioridad	Alta

Tabla 4.5: Caso de Uso: CU-17 Gestionar costes

CU-09	Ver tratamientos
Actor	Usuario
Requisitos asociados	RU-09, RU-10, RU-11
Descripción	El usuario puede consultar los tratamientos registrados, incluyendo detalles específicos sobre cada uno.
Precondiciones	PRE-1. El usuario debe estar autenticado.
Postcondiciones	POST-1. La información de los tratamientos se muestra correctamente en la interfaz.
Flujo normal	▪ FN1. El usuario accede a la sección de tratamientos. ▪ FN2. El usuario selecciona un tratamiento para ver sus detalles. ▪ FN3. El sistema muestra la información del tratamiento seleccionado.
Flujo alternativo	▪ FA2. Si no hay tratamientos registrados, se muestra un mensaje indicando la falta de datos.
Excepciones	▪ E1. El sistema no puede recuperar la información de los tratamientos de la base de datos.
Prioridad	Media

Tabla 4.6: Caso de Uso: CU-09 Ver tratamientos

4.4. Requisitos funcionales

A continuación se detallan los requisitos funcionales del sistema, que especifican las funcionalidades para satisfacer las necesidades del usuario. Estos requisitos garantizan el cumplimiento de las operaciones clave y de la interacción con el sistema.

Requisito	Descripción
RF-01	El sistema debe permitir al usuario añadir nuevas parcelas a través de una interfaz de entrada de datos.
RF-02	El sistema debe validar la información de la parcela antes de guardarla en la base de datos.
RF-03	El sistema debe proporcionar al usuario un listado actualizado de todas las parcelas existentes en el sistema.
RF-04	El sistema debe permitir al usuario ordenar el listado de parcelas por orden alfabético.
RF-05	El sistema debe permitir al usuario ver los detalles completos de cada parcela al seleccionarla del listado.
RF-06	El sistema debe mostrar información detallada de la parcela, como el propietario, el polígono, el término...
RF-07	El sistema debe permitir al usuario eliminar parcelas seleccionadas de la base de datos.
RF-08	El sistema debe solicitar confirmación antes de proceder con la eliminación de una parcela.
RF-09	El sistema debe permitir al usuario modificar los datos de una parcela existente.
RF-10	El sistema debe validar los cambios realizados en los datos de la parcela antes de actualizarlos.
RF-11	El sistema debe permitir al usuario filtrar las parcelas mediante un término de búsqueda específico.
RF-12	El sistema debe mostrar únicamente las parcelas que coincidan con el término de búsqueda ingresado.
RF-13	El sistema debe permitir al usuario añadir tratamientos específicos a una parcela seleccionada.
RF-14	El sistema debe permitir al usuario especificar detalles adicionales sobre el tratamiento añadido, como fecha y tipo.

Requisito	Descripción
RF-15	El sistema debe permitir al usuario iniciar sesión utilizando sus credenciales de acceso.
RF-16	El sistema debe validar las credenciales del usuario y permitir el acceso solo si son correctas.
RF-17	El sistema debe proporcionar al usuario un listado de todos los tratamientos específicos registrados.
RF-18	El sistema debe ordenar el listado de tratamientos por fecha.
RF-19	El sistema debe permitir al usuario filtrar la lista de tratamientos por tipo de tratamiento.
RF-20	El sistema debe permitir al usuario filtrar la lista de tratamientos por año.
RF-21	El sistema debe permitir al usuario acceder a la información detallada de un tratamiento específico.
RF-22	El sistema debe permitir al usuario modificar los detalles de un tratamiento específico existente.
RF-23	El sistema debe validar los cambios realizados en los detalles del tratamiento antes de actualizar la base de datos.
RF-24	El sistema debe permitir al usuario añadir diferentes tipos de costes asociados a tratamientos.
RF-25	El sistema debe permitir al usuario visualizar el gasto total acumulado en tratamientos.
RF-26	El sistema debe calcular y mostrar el gasto total basado en los registros de coste actuales.
RF-27	El sistema debe permitir al usuario consultar el gasto total y poder filtrarlo por años, temporadas y tipo de coste.
RF-28	El sistema debe ofrecer herramientas para visualizar gráficos o informes de gasto por diferentes categorías.
RF-29	El sistema debe mostrar la estimación de gasto en relación con el área de cada parcela.
RF-30	El sistema debe permitir al usuario visualizar un listado de todos los costes registrados.
RF-31	El sistema debe ordenar el listado de costes por fecha.
RF-32	El sistema debe permitir al usuario filtrar los costes por año.

Requisito	Descripción
RF-33	El sistema debe permitir al usuario filtrar los costes por tipo de coste.
RF-34	El sistema debe permitir al usuario modificar la información de cada coste específico.
RF-35	El sistema debe validar los cambios realizados en la información del coste antes de actualizarla.
RF-36	El sistema debe permitir al usuario eliminar un coste específico de la base de datos.
RF-37	El sistema debe solicitar confirmación antes de proceder con la eliminación de un coste.
RF-38	El sistema debe permitir al usuario visualizar su información de perfil, incluyendo nombre y datos de contacto.
RF-39	El sistema debe ofrecer la opción de editar la información del perfil del usuario.

Tabla 4.7: Requisitos Funcionales

4.5. Requisitos no funcionales

A continuación se presentan los requisitos no funcionales, que definen los atributos de calidad del sistema, como la fiabilidad, el rendimiento y la usabilidad. Estos requisitos aseguran que el sistema no solo cumpla con las funcionalidades, sino que también ofrezca una experiencia de usuario óptima y eficiente.

Requisito	Descripción	Tipo
NF-01	El sistema debe responder a las solicitudes del usuario en menos de 3 segundos bajo condiciones normales de carga.	Rendimiento
NF-02	El sistema debe ser capaz de manejar simultáneamente hasta 5,000 usuarios activos sin experimentar una degradación significativa en el rendimiento.	Escalabilidad
NF-03	Todos los datos sensibles del usuario deben ser cifrados.	Seguridad

Requisito	Descripción	Tipo
NF-04	El sistema debe realizar copias de seguridad automáticas cada 24 horas y conservar las copias de seguridad durante al menos 30 días.	Fiabilidad
NF-05	La disponibilidad del sistema debe ser del 99.9 % al año, excluyendo el tiempo de mantenimiento programado.	Fiabilidad
NF-06	La interfaz de usuario debe cumplir con las pautas de accesibilidad WCAG.	Usabilidad
NF-07	El sistema debe adaptarse a diferentes tipos de pantalla de ordenador.	Accesibilidad
NF-08	La aplicación debe ser compatible con las versiones más recientes de los navegadores web más utilizados (Chrome, Firefox, Safari, Edge).	Compatibilidad
NF-09	El sistema debe permitir actualizaciones y cambios en la configuración del sistema sin necesidad de reiniciar el servidor.	Mantenibilidad
NF-10	El sistema debe ser capaz de manejar adecuadamente datos incorrectos o no proporcionados por el usuario, estableciendo valores predeterminados cuando sea necesario o informando al usuario sobre el error, sin que ello cause fallos en la operación del sistema.	Robustez
NF-11	El sistema debe permitir la recuperación de datos en caso de fallo, con una pérdida de datos máxima de 10 minutos.	Recuperación
NF-12	El código de la aplicación será altamente reutilizable en otros sistemas.	Reusabilidad

Tabla 4.8: Requisitos No Funcionales y Atributos de Calidad

Restricciones

Se enumeran las restricciones técnicas que establecen los límites y requisitos operativos del sistema. Estas garantizan que el desarrollo cumpla con los estándares tecnológicos y herramientas específicas.

Restricción	Descripción
R-01	El sistema debe ser desarrollado utilizando PHP para el backend, JavaScript para la lógica del lado del cliente, y debe cumplir con los estándares de HTML5 y CSS3 para el diseño de la interfaz.
R-02	La base de datos debe ser gestionada a través de phpMyAdmin.
R-03	El sistema debe garantizar que las operaciones de la base de datos, incluyendo consultas y actualizaciones, sean realizadas de forma eficiente y segura para evitar problemas de rendimiento y vulnerabilidades.
R-04	La aplicación debe ser accesible y funcional en dispositivos móviles y de escritorio, asegurando una experiencia de usuario óptima en diferentes tamaños de pantalla y resoluciones.

Tabla 4.9: Restricciones Técnicas del Sistema

4.6. Requisitos de información

Los requisitos de información desempeñan un papel fundamental, ya que la correcta recopilación, almacenamiento y utilización de los datos son elementos esenciales para el funcionamiento efectivo de la aplicación.

Esta sección tiene como objetivo sentar las bases para la correcta gestión de la información en el desarrollo de la aplicación web, asegurando que los requisitos de información se cumplan de manera efectiva y contribuyendo al éxito del proyecto.

En la tabla 4.10 se definen los datos requeridos por la aplicación.

Id	Descripción
RI-01	El sistema se encargará de almacenar la información de todos los usuarios que se registren en la aplicación.
RI-02	El sistema se encargará de almacenar la información de todas las parcelas proporcionadas por el usuario registrado.
RI-03	El sistema se encargará de almacenar la información referida a los costes de relacionados con los tratamientos agrícola.
RI-04	EL sistema se encargará de almacenar información de los distintos tratamientos agrícolas realizados en las parcelas.

Tabla 4.10: Requisitos de información

4.6.1. Introducción al dominio de aplicación

En el contexto de nuestra aplicación web de gestión de multicultivos, es fundamental comprender las entidades y conceptos clave que manejamos para ofrecer una solución efectiva y adaptada a las necesidades del sector agrícola.

La aplicación gestiona varias entidades esenciales para la administración de explotaciones agrícolas. Estas incluyen:

- **Usuarios**: Representan a las personas que interactúan con la aplicación. Cada usuario está asociado a una explotación específica, de forma que varios usuarios pueden estar asociados a la misma explotación.
- **Explotaciones**: Una explotación es una unidad de gestión agrícola que puede consistir en una o varias parcelas. El objetivo es centralizar la información y el control sobre todas las actividades agrícolas realizadas en esa área.
- **Parcelas**: Son las unidades de terreno dentro de una explotación. Cada parcela está detalladamente identificada por su ubicación geográfica y características específicas, como superficie y propietario. La gestión de parcelas permite un seguimiento preciso de las actividades y costes asociados a cada área cultivada.
- **Tratamiento**: Define los tipos de tratamientos que se pueden aplicar en las parcelas, en este cado pueden ser abonado, fitosanitarios, labores de arado, siembra y recolección.
- **Tratamientos Específicos**: Son cada uno de los tratamientos que es realizado a cada parcela durante una temporada determinada.
- **Temporadas**: En el ámbito agrícola, una temporada se refiere a un periodo de tiempo durante el cual se desarrollan ciertas actividades o ciclos de cultivo. En lugar de alinearse con el calendario anual, una temporada puede coincidir con periodos específicos que se ajusten a los ciclos de cultivo o condiciones climáticas.
- **Costes**: Por un lado, los costes se utilizan para calcular los gastos genearados durante los tratamientos en base al año, temporada y tipo de gasto. Por otro lado, la aplicación permite estimar los gastos por parcela, en función de los tratamientos, en relación a su superficie, facilitando un análisis detallado de los costos asociados a cada área cultivada.

4.6.2. Modelo Entidad-Relación

Después de haber indicado los requisitos de información, se presenta, a continuación, el modelo entidad-relación, el cual permite una representación clara y organizada de la estructura de la información.

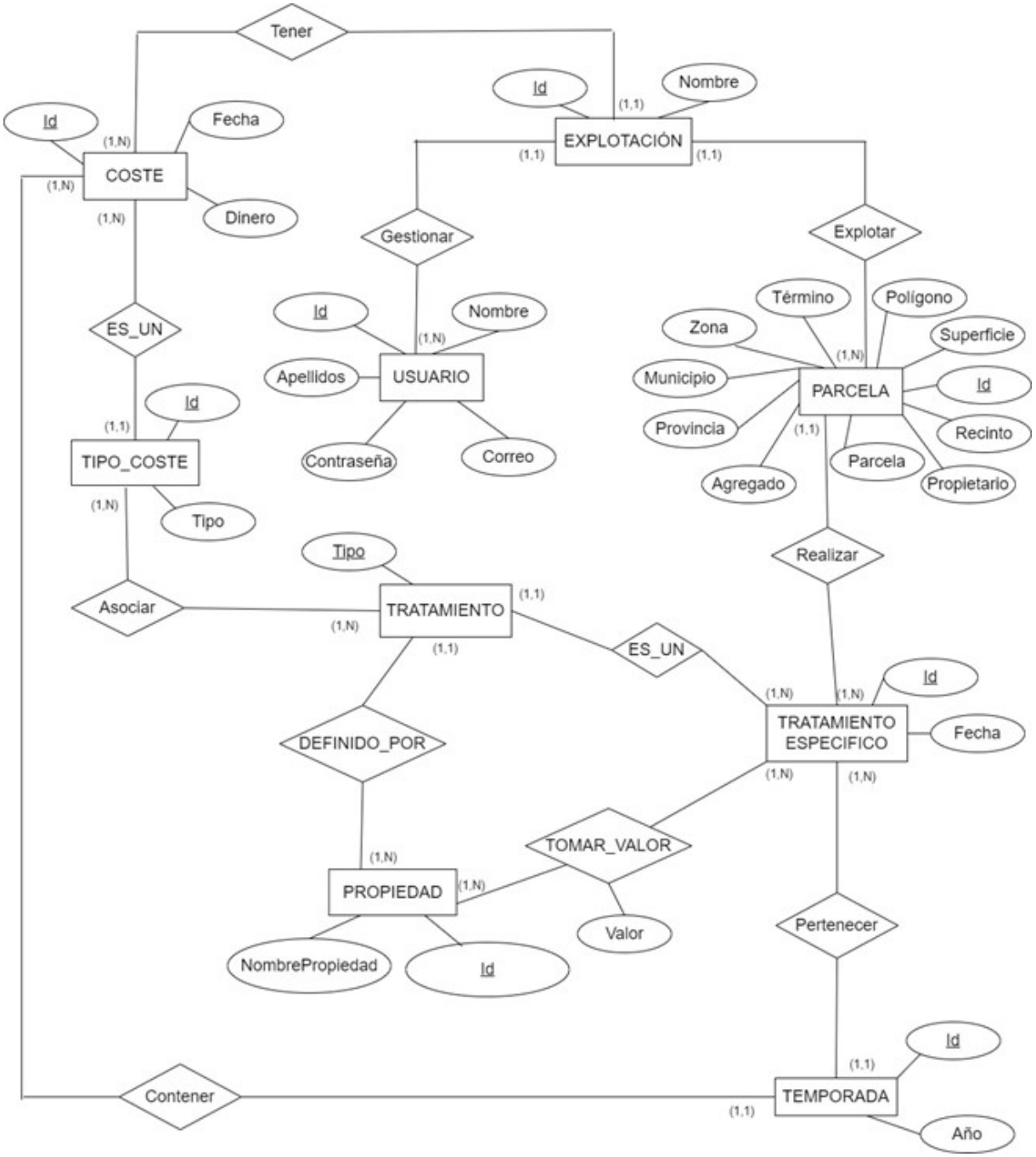


Figura 4.2: Modelo Entidad-Relación

4.6.3. Diccionario de datos

Una vez que se han definido los requisitos y se ha creado el modelo entidad-relación, se lleva a cabo un análisis detallado tanto de las entidades como de las relaciones y sus atributos.

Entidades

En esta fase se lleva a cabo un análisis minucioso de los atributos que definen cada entidad. Estos atributos son identificados, analizados y vinculados de forma única a su entidad correspondiente, asegurando así una descripción precisa y coherente de la información.

ID	E01				
Nombre	USUARIO				
Definición	La entidad USUARIO representa a cada una delas personas que se registran en la aplicación				
Atributos					
ID	Nombre	Definición	Tipo	Único	Nulo
E01.01	Id	Identificador autoincremental	Int	Si	No
E01.02	Nombre	Nombre de usuario con el que se registra	Varchar	No	No
E01.03	Apellido	Apellidos del usuario que se registra	Varchar	No	No
E01.04	Correo	Dirección de correo electrónico asociada a cada usuario	Varchar	Si	No
E01.05	Contraseña	Contraseña de inicio de sesión a la aplicación	Varchar	No	No
Identificadores					
ID	Nombre Atributo				
E01.01	Id				

Tabla 4.11: Entidad “USUARIO”

ID	E02				
Nombre	PARCELA				
Definición	La entidad PARCELA representa cada uno de los cultivos de cada usuario				
Atributos					
ID	Nombre	Definición	Tipo	Único	Nulo
E02.01	Id	Identificador autoincremental	Int	Si	No
E02.02	Provincia	Referencia numérica SIGPAC de la provincia	Int	No	No
E02.03	Municipio	Referencia numérica SIGPAC del municipio	Int	No	No
E02.04	Agregado	Referencia numérica SIGPAC del agregado	Int	No	No
E02.05	Zona	Referencia numérica SIGPAC de la zona	Int	No	No
E02.06	Término	Nombre de agregado o municipio	Varchar	No	No
E02.07	Polígono	Referencia numérica SIGPAC del polígono	Int	No	No
E02.08	Parcela	Referencia numérica SIGPAC de la parcela	Int	No	No
E02.09	Recinto	Referencia numérica SIGPAC del recinto	Int	No	No
E02.10	Superficie	Cabida de la parcela expresada en hectáreas	Float	No	No
E02.11	Propietario	Nombre del propietario	Varchar	No	Si
Identificadores					
ID	Nombre Atributo				
E02.01	Id				

Tabla 4.12: Entidad “PARCELA”

ID	E03				
Nombre	TRATAMIENTO_ESPECÍFICO				
Definición	La entidad TRATAMIENTO_ESPECÍFICO representa la tarea que se realiza a cada parcela				
Atributos					
ID	Nombre	Definición	Tipo	Único	Nulo
E03.01	Id	Identificador autoincremental	Int	Si	No
E03.02	Fecha	Momento en el que se realiza el tratamiento	Varchar	No	No

Identificadores	
ID	Nombre Atributo
E03.01	Id

Tabla 4.13: Entidad “TRATAMIENTO_ESPECÍFICO”

ID	E04				
Nombre	TRATAMIENTO				
Definición	La entidad TRATAMIENTO representa cada uno de los tipos de tareas que hay para las parcelas.				
Atributos					
ID	Nombre	Definición	Tipo	Único	Nulo
E04.01	Tipo	Distintos tipos de tratamientos	Varchar	Si	No

Identificadores	
ID	Nombre Atributo
E04.01	Tipo

Tabla 4.14: Entidad "TRATAMIENTO"

ID	E05				
Nombre	PROPIEDAD				
Definición	La entidad PROPIEDAD representa cada una de la características de cada tipo de tratamiento.				
Atributos					
ID	Nombre	Definición	Tipo	Único	Nulo
E05.01	NombrePropiedad	Cada una de las características	Varchar	Si	No

Identificadores	
ID	Nombre Atributo
E05.01	NombrePropiedad

Tabla 4.15: Entidad "PROPIEDAD"

ID	E06				
Nombre	TEMPORADA				
Definición	La entidad TEMPORADA representa el periodo de tiempo del tratamiento de las parcelas				
Atributos					
ID	Nombre	Definición	Tipo	Único	Nulo
E06.01	Id	Identificador autoincremental	Int	Si	No
E06.02	Año	Año de la temporada de las parcelas	Varchar	No	No
Identificadores					
ID	Nombre Atributo				
E06.01	Id				

Tabla 4.16: Entidad “TEMPORADA”

ID	E07				
Nombre	COSTE				
Definición	La entidad COSTE representa cada uno de los gastos en relación a las parcelas				
Atributos					
ID	Nombre	Definición	Tipo	Único	Nulo
E07.01	Id	Identificador autoincremental	Int	Si	No
E07.02	Fecha	Momento del gasto	Date	No	No
E07.03	Concepto	Descripción del gasto de dinero	Varchar	No	No
E07.04	Dinero	Cantidad de dinero gastada	Float	No	No
Identificadores					
ID	Nombre Atributo				
E07.01	Id				

Tabla 4.17: Entidad ”COSTE”

ID	E08				
Nombre	EXPLOTACIÓN				
Definición	La entidad EXPLOTACIÓN representa la agrupación de parcelas de varios usuarios				
Atributos					
ID	Nombre	Definición	Tipo	Único	Nulo
E08.01	Id	Identificador autoincremental	Int	Si	No
E08.02	Nombre	Designación de la explotación	Varchar	No	No

Identificadores	
ID	Nombre Atributo
E08.01	Id

Tabla 4.18: Entidad “EXPLOTACIÓN”

ID	E09				
Nombre	TIPO_COSTE				
Definición	La entidad TIPO_COSTE representa clase de coste que tiene los tratamientos				
Atributos					
ID	Nombre	Definición	Tipo	Único	Nulo
E09.01	Id	Identificador autoincremental	Int	Si	No
E09.02	Tipo	Clase de coste	Varchar	No	No

Identificadores	
ID	Nombre Atributo
E09.01	Id

Tabla 4.19: Entidad “TIPO_COSTE”

Relaciones

En esta fase se realiza un análisis exhaustivo de las relaciones entre las entidades del sistema. Se identifican y estudian las interacciones, estableciendo conexiones precisas y claras entre las entidades involucradas. El objetivo es representar de manera coherente y comprensible las relaciones y flujos de información en el sistema.

ID	R01		
Nombre	GESTIONAR		
Definición	La relación GESTIONAR modela como los usuarios pueden gestionar explotaciones		
Entidades			
ID	Nombre Entidad	Participación	Cardinalidad
E01	USUARIO	1	N
E02	EXPLOTACIÓN	1	1
Atributos			
ID	Nombre Atributo		

Tabla 4.20: Relación “GESTIONAR”

ID	R02		
Nombre	REALIZAR		
Definición	La relación REALIZAR modela como a las parcelas se las realiza tratamientos concretos		
Entidades			
ID	Nombre Entidad	Participación	Cardinalidad
E02	PARCELA	1	1
E03	TRATAMIENTO_ESPECÍFICO	1	N
Atributos			
ID	Nombre Atributo		

Tabla 4.21: Relación “REALIZAR”

ID	R03		
Nombre	PERTENECER		
Definición	La relación PERTENECER modela como los tratamientos de las parcelas están asociados a una temporada		
Entidades			
ID	Nombre Entidad	Participación	Cardinalidad
E03	TRATAMIENTO	1	N
E09	TEMPORADA	1	1
Atributos			
ID	Nombre Atributo		

Tabla 4.22: Relación “PERTENECER”

ID	R04		
Nombre	CONTENER		
Definición	La relación CONTENER modela como los gastos se agrupan por temporadas		
Entidades			
ID	Nombre Entidad	Participación	Cardinalidad
E09	TEMPORADA	1	1
E10	COSTE	1	N
Atributos			
ID	Nombre Atributo		

Tabla 4.23: Relación “CONTENER”

ID	R05		
Nombre	ES_UN		
Definición	La relación ES_UN modela como un tratamiento concreto forma parte de un tratamiento general		
Entidades			
ID	Nombre Entidad	Participación	Cardinalidad
E03	TRATAMIENTO_ESPECÍFICO	1	N
E04	TRATAMIENTO	1	1
Atributos			
ID	Nombre Atributo		

Tabla 4.24: Relación “ES_UN”

ID	R06		
Nombre	DEFINIDO_POR		
Definición	La relación DEFINIDO_POR modela como un tratamiento tiene distintas características		
Entidades			
ID	Nombre Entidad	Participación	Cardinalidad
E04	TRATAMIENTO	1	1
E05	PROPIEDAD	1	N
Atributos			
ID	Nombre Atributo		

Tabla 4.25: Relación “DEFINIDO_POR”

ID	R07		
Nombre	TOMAR_VALOR		
Definición	La relación TOMAR_VALOR modela como cada propiedad tiene un valor		
Entidades			
ID	Nombre Entidad	Participación	Cardinalidad
E05	PROPIEDAD	1	N
E03	TRATAMIENTO_ESPECÍFICO	1	N
Atributos			
ID	Nombre Atributo		
R07.01	Valor		

Tabla 4.26: Relación "TOMAR_VALOR"

ID	R09		
Nombre	ASOCIAR		
Definición	La relación ASOCIAR que los tipos de coste son para distintos tratamientos y viceversa		
Entidades			
ID	Nombre Entidad	Participación	Cardinalidad
E09	TIPO_COSTE	1	N
E04	TRATAMIENTO	1	N
Atributos			
ID	Nombre Atributo		

Tabla 4.27: Relación "EXPLOTAR"

ID	R08		
Nombre	EXPLOTAR		
Definición	La relación EXPLOTAR modela que varias pareclas son de una explotacion		
Entidades			
ID	Nombre Entidad	Participación	Cardinalidad
E08	EXPLOTACIÓN	1	1
E02	PARCELA	1	N
Atributos			
ID	Nombre Atributo		

Tabla 4.28: Relación “EXPLOTAR”

ID	R09		
Nombre	ES_UN		
Definición	La relación ES_UN representa que los costes están asociados a diferentes tipos.		
Entidades			
ID	Nombre Entidad	Participación	Cardinalidad
E07	COSTE	1	N
E09	TIPO_COSTE	1	1
Atributos			
ID	Nombre Atributo		

Tabla 4.29: Relación “ES_UN”

ID	R10		
Nombre	TENER		
Definición	La relación TENER representa que la explotación genera varios tipos de costes		
Entidades			
ID	Nombre Entidad	Participación	Cardinalidad
E08	EXPLOTACIÓN	1	1
E07	COSTES	1	N
Atributos			
ID	Nombre Atributo		

Tabla 4.30: Relación “TENER”

Capítulo 5

Diseño del sistema

5.1. Arquitectura

La arquitectura de software es un concepto clave en el desarrollo de sistemas, ya que define la estructura y organización fundamental de un sistema, así como las relaciones entre sus componentes. A lo largo del tiempo, se ha debatido mucho sobre la definición precisa de la arquitectura, pero una visión compartida por desarrolladores expertos es que la arquitectura se centra en "las cosas importantes", es decir, aquellos aspectos del diseño que son cruciales para el éxito del sistema. Esta perspectiva resalta la importancia de identificar y gestionar los elementos arquitectónicos que, si se descuidan, pueden llevar a problemas graves en el futuro.

Una arquitectura bien diseñada no solo facilita la comprensión y mantenimiento del software, sino que también mejora la agilidad en la entrega de nuevas funcionalidades. Aunque los usuarios finales no suelen percibir directamente la calidad de la arquitectura, su impacto es crucial para garantizar que el sistema sea escalable y adaptable a lo largo del tiempo. La atención a una buena arquitectura permite evitar problemas futuros, asegurando que el desarrollo del software no se vea obstaculizado por decisiones de diseño deficientes. Por ello, invertir en una arquitectura sólida es fundamental para el éxito a largo plazo del proyecto.

5.1.1. Arquitectura lógica

La arquitectura lógica se refiere a la estructura y organización de los componentes lógicos que conforman un sistema, así como las interacciones y relaciones que estos establecen entre sí dentro de la aplicación. Este enfoque permite visualizar cómo los diferentes módulos o servicios se comunican y cooperan para cumplir con las funcionalidades del sistema. La arquitectura lógica del sistema que nos ocupa se encuentra detallada en la figura 5.1, proporcionando una visión clara de su diseño interno y sus conexiones clave.

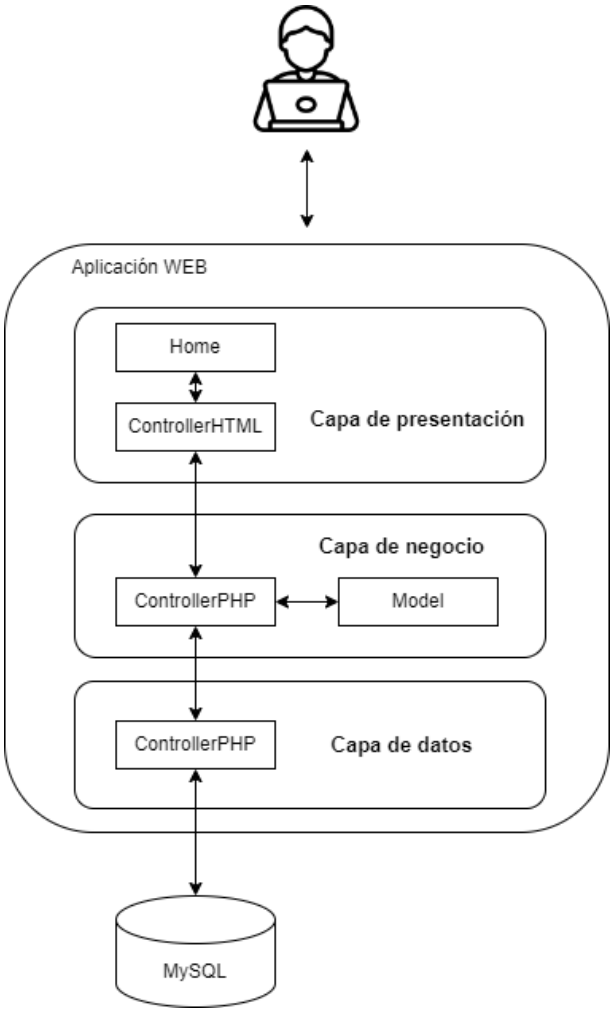


Figura 5.1: Arquitectura lógica

La arquitectura lógica de la aplicación web se compone de tres capas separadas, cada una de las cuales realiza tareas específicas y se interrelaciona con las demás para garantizar el funcionamiento adecuado del sistema:

- **Capa de presentación:** El usuario final interactúa con esta capa. Está compuesto por vistas, es decir, páginas y elementos visuales que el usuario ve y usa, y un controlador JavaScript que le da a las vistas funcionalidad dinámica. Este controlador funciona como un puente entre la capa de negocio y la de presentación, lo que permite que los datos del usuario sean procesados correctamente. Además, en esta capa se incluyen los archivos CSS pertinentes que se encargan de establecer el estilo y el aspecto visual de la interfaz.
- **Capa de negocio:** Toda la lógica de negocio de la aplicación, que se encuentra en el backend, se aloja en esta capa. Esta capa, implementada en PHP, se encarga de procesar y administrar los datos de la capa de presentación. Su función principal

es interpretar las solicitudes del usuario, realizar las operaciones necesarias sobre los datos y tomar decisiones en función de la lógica predeterminada. Es el centro del sistema, donde se crean y realizan las reglas y procedimientos que dictan el comportamiento de la aplicación.

- **Capa de datos:** La principal responsabilidad de esta capa final es la interacción con la base de datos. Su trabajo consiste en intermediar entre la capa de negocio y la base de datos, supervisando las operaciones de lectura y escritura de datos. De esta manera, garantiza que la capa de negocio pueda acceder a los datos necesarios para realizar sus operaciones y almacenar los resultados de manera segura y eficiente.

Tras completar el análisis, se puede determinar una arquitectura que guarda una considerable similitud con el modelo de arquitectura modelo-vista-controlador (MVC). En la figura 5.2, se presentan los componentes que conforman esta estructura:

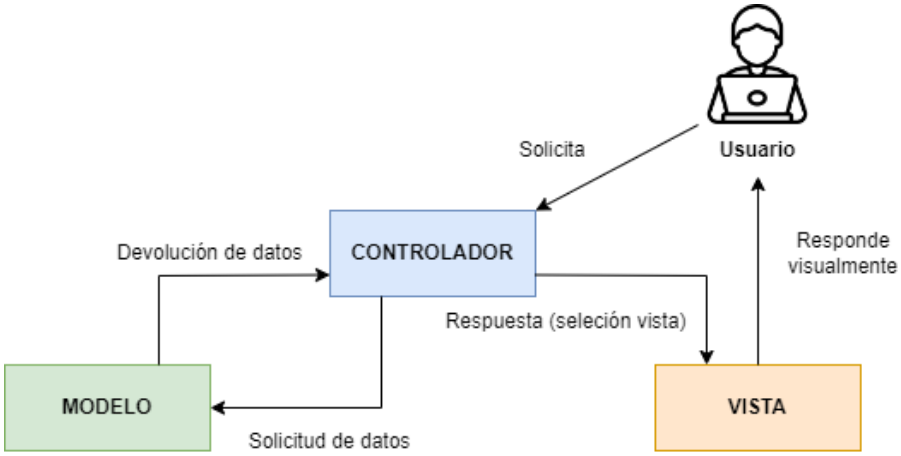


Figura 5.2: Patrón MVC (Modelo-Vista-Controlador)

- **Modelo:** Es el componente que gestiona los datos y la lógica de la aplicación. Se encarga de la interacción con la base de datos y realiza las operaciones necesarias para manejar la información.
- **Vista:** Representa la interfaz de usuario y es responsable de mostrar los datos al usuario. Está formada por las vistas y los archivos CSS que definen la apariencia de la aplicación.
- **Controlador:** Actúa como intermediario entre el modelo y la vista. Procesa las entradas del usuario y actualiza el modelo o la vista según sea necesario, facilitando la comunicación entre ambos.

5.1.2. Arquitectura física

La arquitectura física representa los componentes tangibles necesarios para implementar la lógica de una aplicación, así como las interrelaciones entre ellos. En el caso de esta

aplicación, la arquitectura física ilustra cómo un cliente, utilizando cualquier dispositivo, puede acceder a la aplicación a través de un navegador, estableciendo comunicación con el servidor web donde está alojada. Este servidor web, a su vez, se conecta con el servidor de base de datos donde se almacenan los datos de la aplicación.

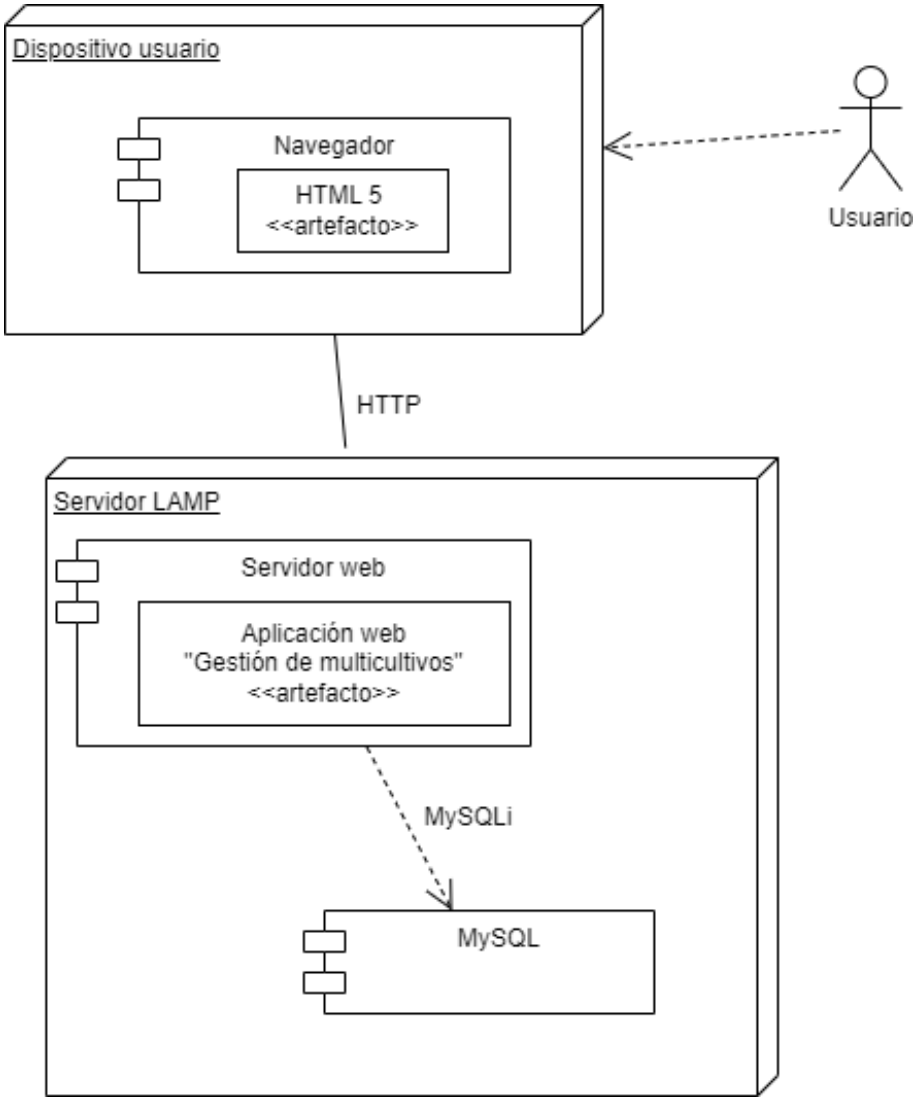


Figura 5.3: Arquitectura física

5.2. Modelo Relacional

Para estructurar y organizar la información en la base de datos, utilizaremos el modelo relacional, que define los datos y establece las relaciones entre ellos de manera coherente.

USUARIO (Id, Nombre, Apellidos, Correo, Contraseña, IdExplotacion)
FK: IdExplotacion → **EXPLOTACION** (Id)

PARCELA (Id, Provincia, Municipio, Agregado, Zona, Término, Polígono, Parcela, Recinto, Superficie, Propietario, IdExplotacion)
FK: IdExplotacion → **EXPLOTACION** (Id)

EXPLOTACION (Id, Nombre)

TRATAMIENTO_ESPECIFICO (Id, Fecha, Tipo, IdParcela, IdAño)
FK: Tipo → **TRATAMIENTO** (Tipo)
FK: IdParcela → **PARCELA** (Id)
FK: IdAño → **TEMPORADA** (Id)

TRATAMIENTO (Tipo)

PROPIEDAD (Id, NombrePropiedad, Tipo)
FK: Tipo → **TRATAMIENTO** (Tipo)

TOMAR_VALOR (IdTratamiento, IdPropiedad, Valor)
FK: IdTratamiento → **TRATAMIENTO_ESPECIFICO** (Id)
FK: IdPropiedad → **PROPIEDAD** (Id)

TEMPORADA (Id, Año)

COSTE (Id, Fecha, Dinero, IdExplotacion, IdTipoCoste, IdAño)
FK: IdExplotacion → **EXPLOTACION** (Id)
FK: IdTipoCoste → **TIPO_COSTE** (Id)
FK: IdAño → **TEMPORADA** (Id)

TIPO_COSTE (Id, Tipo)

ASOCIAR (IdTratamiento, IdTipoCoste)
FK: IdTipoCoste → **TIPO_COSTE** (Id)
FK: IdTratamiento → **TRATAMIENTO** (Tipo)

Restricciones de Integridad

Estas restricciones son necesarias para asegurar que las relaciones entre las tablas se mantengan consistentes durante las operaciones de borrado y actualización. En la aplicación, se ha configurado las restricciones de integridad para que utilicen la opción CASCADE.

- **Borrado:** Cuando se elimina un registro en una tabla, con la opción CASCADE configurada, cuando se borra un registro en una tabla principal, todos los registros relacionados en las tablas secundarias se eliminan automáticamente. Esto garantiza que no queden registros huérfanos y que la base de datos mantenga su integridad referencial.
- **Actualización:** Al modificar el valor de una clave primaria en una tabla principal, todos los registros en las tablas secundarias que referencian ese valor se actualizan automáticamente con ese nuevo valor.

5.3. Diagramas de secuencia

Para definir los flujos adecuados de las operaciones dentro de la aplicación, empleamos diagramas de secuencia. Estos diagramas detallan la serie de interacciones que ilustran el orden en que los objetos colaboran entre sí.

5.3.1. Estereotipos de clases de línea

Se hace uso de tres estereotipos de clase de línea principales en los diagramas de secuencia. Con ellos, se representan diferentes responsabilidades y roles del sistema.

- **Entidades (*Entity*)**: Las entidades son responsables de almacenar los datos utilizados por el sistema y de realizar los cálculos pertinentes sobre estos. Cada entidad representa un concepto significativo dentro del dominio del problema y se encarga de conservar la identidad de los datos persistentes.
- **Objeto de Frontera (*Boundary*)**: Estos objetos se encargan de modelar todas las interacciones entre el sistema y los actores externos. Cualquier comunicación entre el sistema y un actor externo debe pasar por un objeto de frontera, ya sea para gestionar datos en una interfaz gráfica de usuario o para interactuar con una API web.
- **Controlador**: Los controladores facilitan la comunicación entre las interfaces y las entidades, evitando que las entidades expongan su lógica directamente a los actores externos, reciben las solicitudes de los límites, determinan cómo procesarlas y manipulan el estado de la aplicación.

5.3.2. Diagrama de secuencia de "Inicio de sesión"

El diagrama 5.4 muestra el proceso de inicio de sesión de un usuario en el sistema. Este proceso se inicia cuando el usuario accede a la pantalla de inicio de sesión, lo que provoca que se cargue el formulario correspondiente en `FormularioLogin.php`, como se muestra en el paso 1 del diagrama, a continuación, el usuario completa los campos necesarios, como su nombre de usuario y contraseña, y pulsa el botón de envío, lo que genera el evento descrito en el paso 4.

En este punto, se llama a la función `ValidarUsuario()`, que es responsable de verificar que los datos introducidos tienen el formato correcto. Esta validación puede tener dos resultados, si las credenciales son incorrectas (`ValidarUsuario()`: KO), el sistema responde con un mensaje de error (paso 6 en el diagrama), informando al usuario que el inicio de sesión ha fallado. En el caso contrario, si las credenciales son correctas (`ValidarUsuario()`: OK), el proceso continúa en el paso 6 del diagrama, donde se solicita al Controlador usuario.php que inicie la sesión del usuario.

Este controlador, envía una solicitud al Modelo `usuario.php` para verificar las credenciales contra los datos almacenados en la base de datos (paso 7), si las credenciales no coinciden con ningún registro en la base de datos, el sistema devolverá un mensaje de error al usuario (paso 9), indicándole que el inicio de sesión ha fallado. Este mensaje se muestra en la interfaz del usuario en el paso 10. En cambio, si las credenciales son válidas, el sistema establecerá las variables de sesión (paso 8), confirmará el inicio de sesión (paso 9) y redirigirá al usuario a la página principal (paso 10), permitiéndole acceder a las funcionalidades del sistema.

Este flujo de trabajo garantiza que solo los usuarios con credenciales correctas puedan acceder al sistema, proporcionando un mecanismo seguro para la autenticación de usuarios.

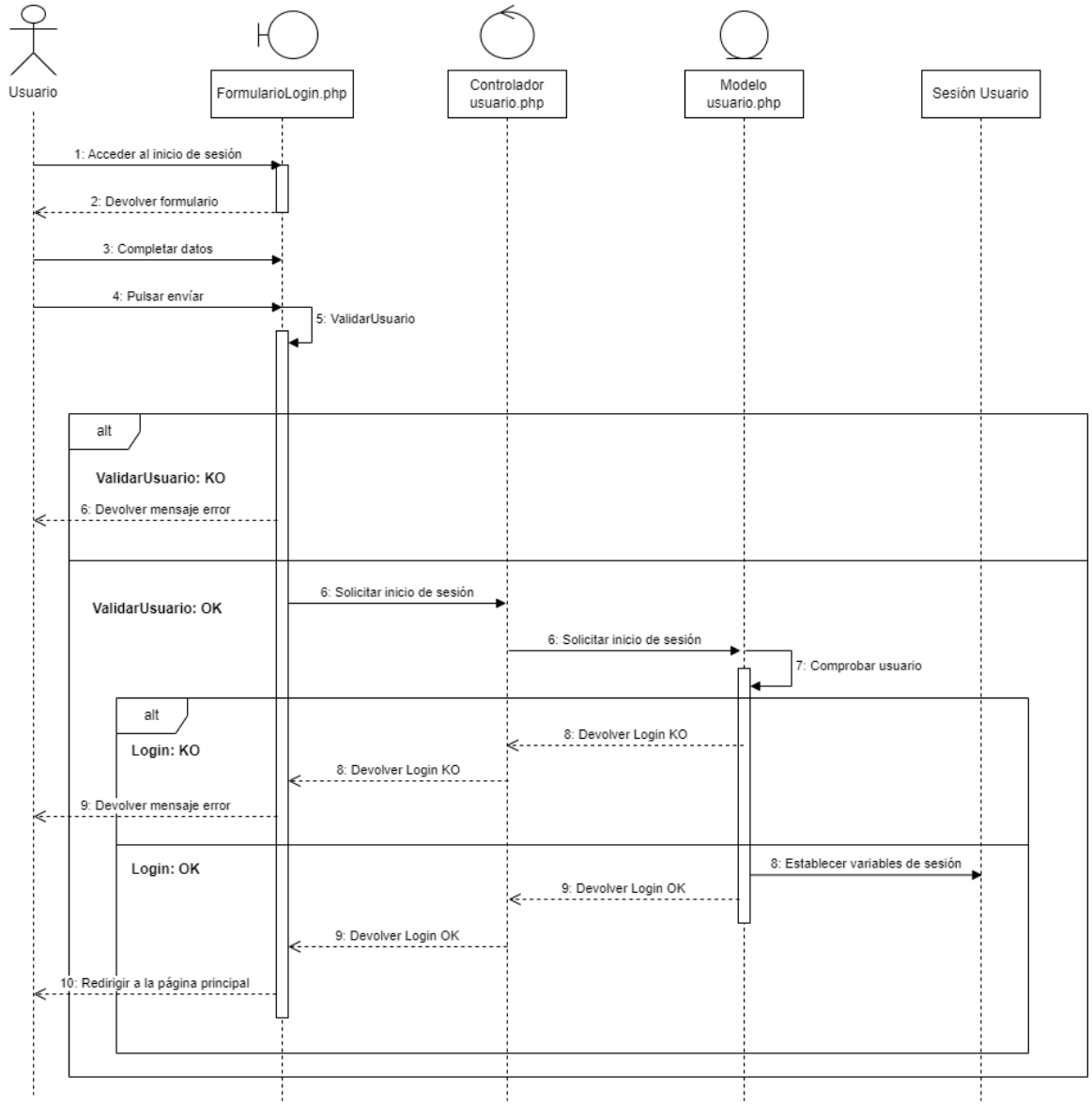


Figura 5.4: Diagrama de secuencia de inicio de sesión

5.3.3. Diagrama de secuencia "Modificar datos parcela"

El el siguiente diagrama 5.5 se muestra el proceso completo para modificar los datos de una parcela en el sistema. Esto se inicia cuando el usuario accede a la sección de parcelas desde la interfaz principal, como se muestra en el paso 1 del diagrama. Al acceder a la sección de parcelas, el usuario puede navegar al listado completo de parcelas pulsando a un bóton (`listParcelas.php`), lo que ocurre en el paso 3.

Al llegar al listado de parcelas, se realiza una solicitud para obtener la información de las parcelas disponibles. Esta solicitud se envía al controlador, que a su vez consulta al modelo (`modelo/parcela.php`) en el paso 5 para recuperar los datos de las parcelas almacenadas en la base de datos.

El modelo responde devolviendo la información requerida y se muestra entonces al usuario la lista de parcelas, en el paso 9, permitiéndole seleccionar la parcela que desea modificar.

Una vez que el usuario selecciona una parcela específica (paso 10), el sistema genera un formulario de modificación que permite al usuario introducir los nuevos datos deseados para la parcela seleccionada (paso 12). Tras esto, envia al controlador la solicitud para la modificación, que a su vez contacta al modelo (`modelo/parcela.php`) para realizar la actualización en la base de datos (paso 14).

Dependiendo del resultado de la operación de modificación, el flujo puede seguir dos caminos:

- Modificación fallida (`ModificarParcela()`: KO): Si la actualización no se realiza correctamente, el sistema devuelve un mensaje de error al usuario.
- Modificación exitosa (`ModificarParcela()`: OK): Si la modificación se realiza con éxito, el sistema redirige al usuario (paso 17) a la lista actualizada de parcelas, donde se muestran los cambios realizados (paso 18).

Este flujo de trabajo asegura que el usuario pueda modificar los datos de las parcelas de manera efectiva, con mecanismos adecuados para manejar tanto los casos de éxito como los de fallo.

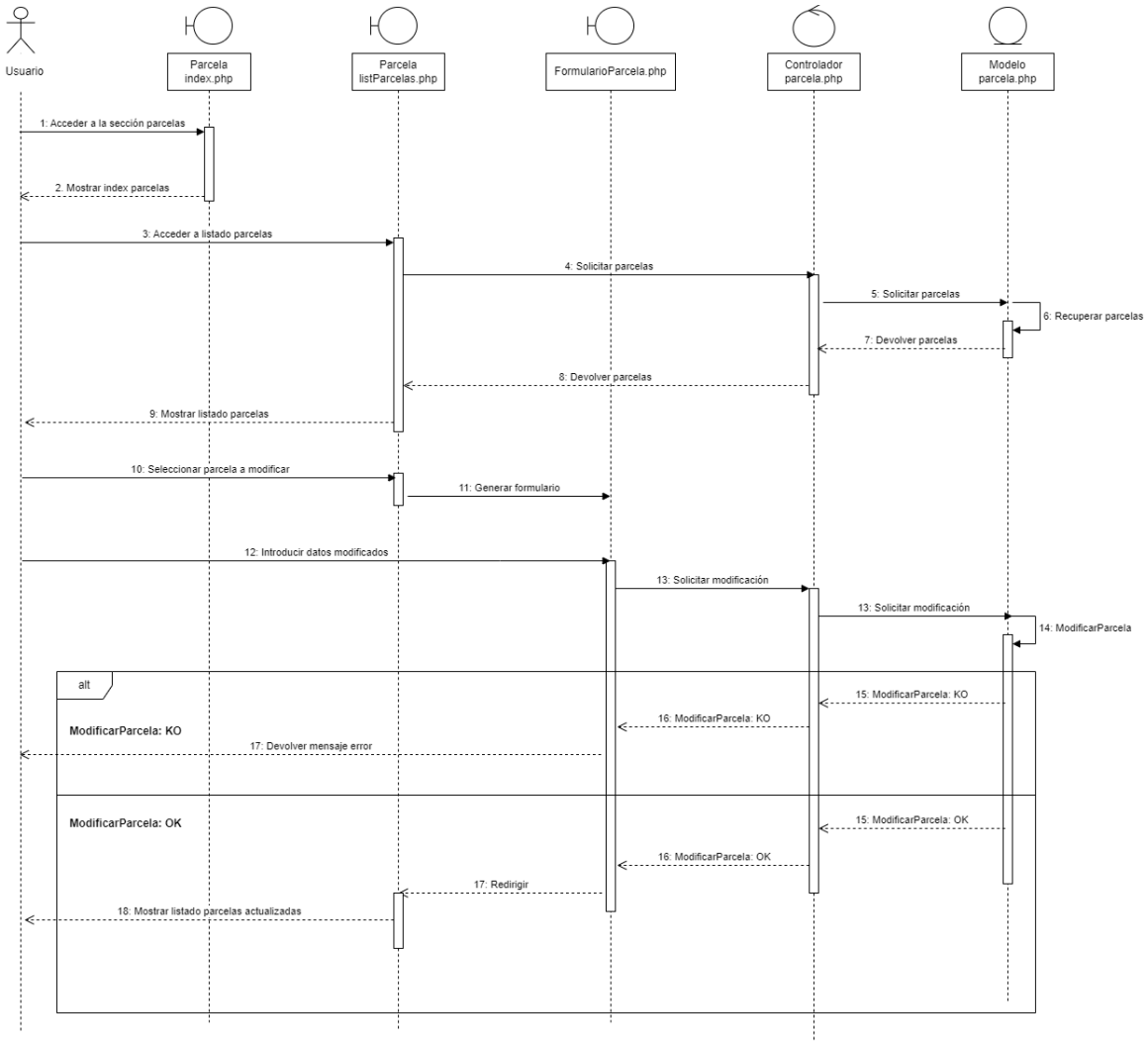
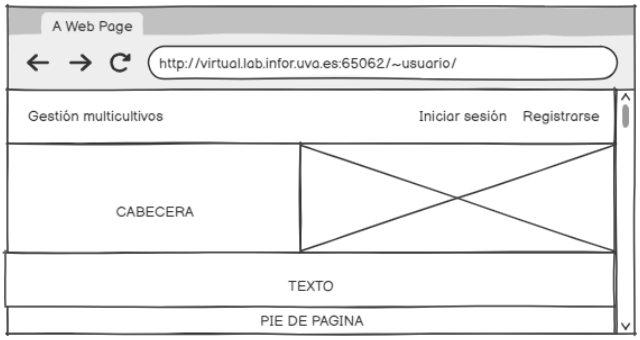
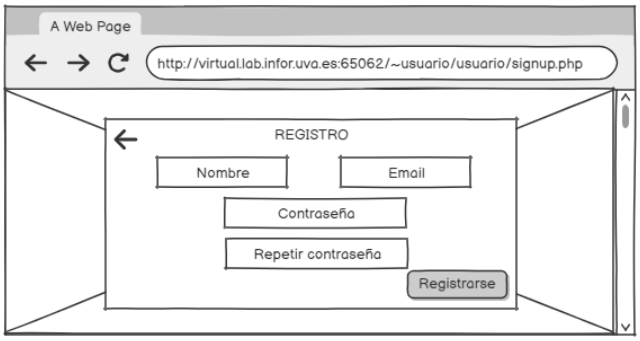


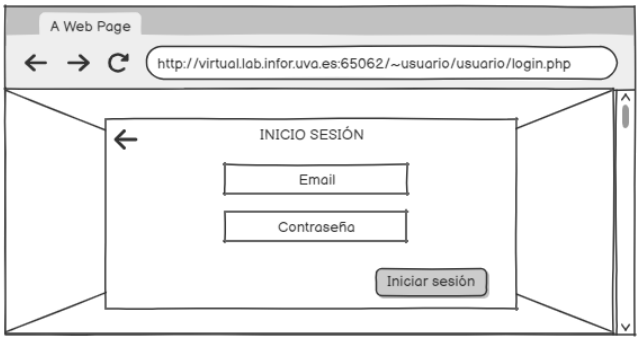
Figura 5.5: Diagrama de secuencia de modificar datos parclea

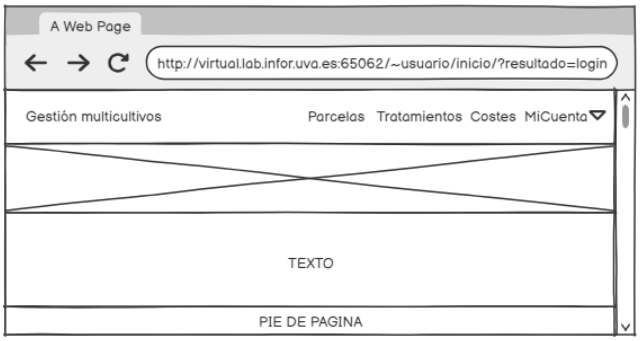
5.4. Diseño de interfaz de usuario

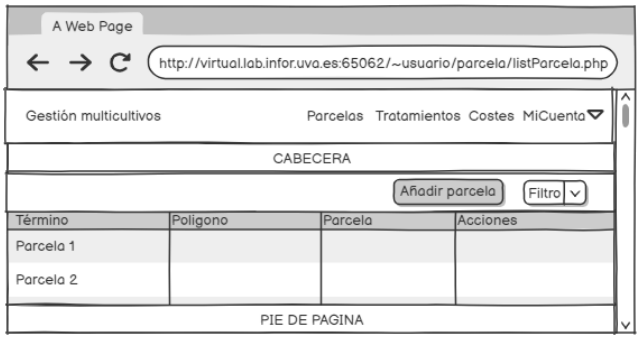
En esta sección se presentarán los diferentes diseños preliminares para cada una de las vistas de la aplicación. Estos diseños reflejan las ideas iniciales del proyecto y, por lo tanto, es posible que experimenten cambios conforme avance el desarrollo y se perfeccionen las interfaces finales. Los diseños aquí expuestos buscan capturar la estructura básica y la funcionalidad prevista para cada vista, sirviendo como una base sobre la cual se irán realizando ajustes y mejoras a lo largo del proceso de desarrollo.

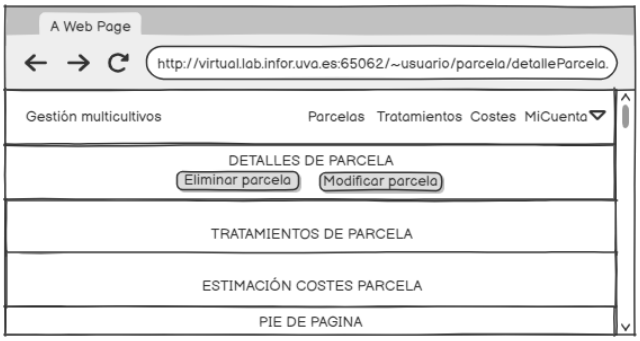
	Pantalla principal
Descripción	Pantalla que se muestra al iniciar la aplicación web
Activación	▪ Iniciar la aplicación web.
Diseño	 Figura 5.6: Interfaz “Pantalla principal”
Eventos	▪ **EV-01**: Acceso al inicio de sesión. ▪ **EV-02**: Acceso al registro de un nuevo usuario.

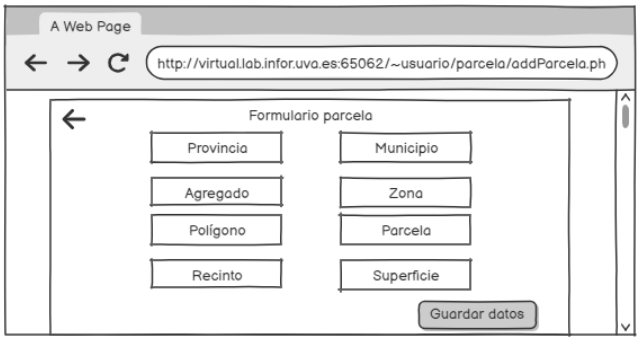
	Registro de usuario
Descripción	Pantalla que permite crear una nueva cuenta de usuario dentro de la aplicación. Además, también se accede a la pantalla de inicio de sesión cuendo el usuario previamente tenga una cuenta ya creada.
Activación	▪ Pulsando el botón de registro de la pantalla principal. ▪ Pulsando el botón de registro desde la pantalla de inicio de sesión.
Diseño	 Figura 5.7: Interfaz “Registro de usuario”
Eventos	▪ **EV-01**: Pulsar al botón para envío de información de usuario. ▪ **EV-02**: Pulsar al botón para ir al inicio de sesión. ▪ **EV-03**: Mensajes informativos sobre la creación del usuario y requerimientos. ▪ **EV-04**: Pulsar el botón para acceder a la pantalla principal.

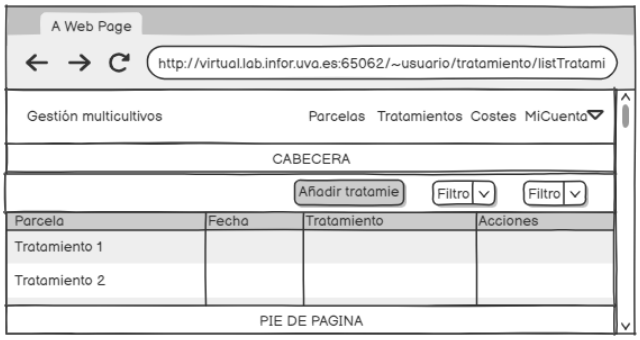
	Inicio de sesión
Descripción	Pantalla que permite al usuario acceder a su cuenta personal previamente creada. También permite el acceso a la pantalla de registro de usuario.
Activación	▪ Pulsando el botón de inicio de sesión de la pantalla principal. ▪ Pulsando el botón de inicio de sesión desde la pantalla de registro.
Diseño	 Figura 5.8: Interfaz “Inicio de sesión”
Eventos	▪ **EV-01**: Pulsar al botón para envío de información de usuario. ▪ **EV-02**: Pulsar al botón para ir al registro. ▪ **EV-03**: Mensajes informativos sobre el inicio de sesión y requerimientos. ▪ **EV-04**: Pulsar el botón para acceder a la pantalla principal.

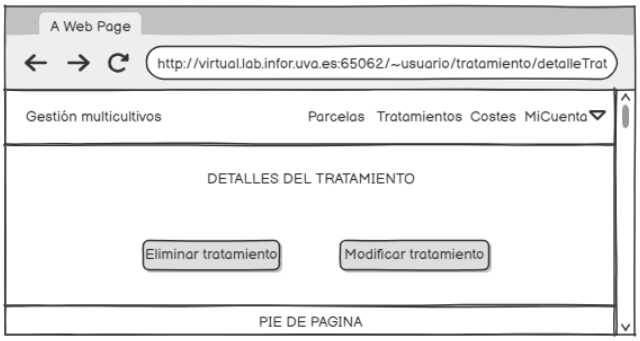
	Home/Pantalla principal
Descripción	Pantalla que se muestra al iniciar sesión de un usuario registrado en la plataforma
Activación	▪ Iniciando sesión en la aplicación
Diseño	 Figura 5.9: Interfaz “Home/Pantalla principal”
Eventos	▪ **EV-01**: Acceso a la gestión de parcelas. ▪ **EV-02**: Acceso a la gestión de tratamientos. ▪ **EV-03**: Acceso a la gestión de costes. ▪ **EV-04**: Acceso a la la información de tu cuenta.

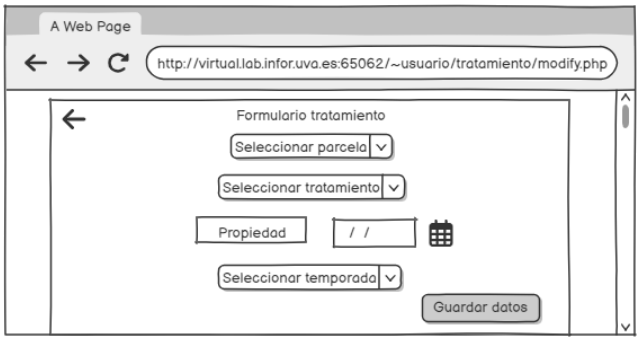
	Listado Parcelas
Descripción	Pantalla dónde se ve un listado de las parcelas registradas en la aplicación.
Activación	▪ Pulsando en ver parcelas dentro de la sección de parcelas desde la home.
Diseño	 Figura 5.10: Interfaz “Listado de parcelas”
Eventos	▪ **EV-01**: Acceso a los detalles de cada parcela. ▪ **EV-02**: Acceso a añadir una nueva parcela. ▪ **EV-03**: Acceso a eliminar las parcela. ▪ **EV-04**: Acceso a modificar las parcelas. ▪ **EV-05**: Acceso a filtrar el listado de parcelas.

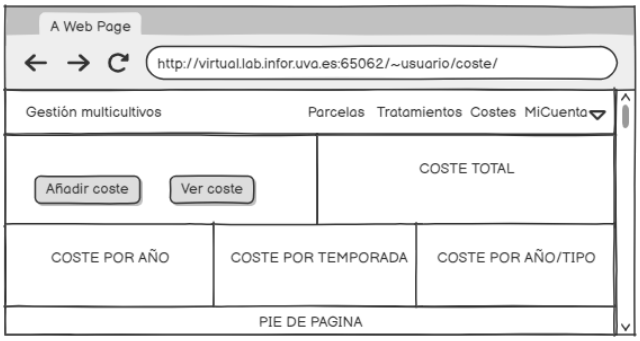
	Detalle Parcelas
Descripción	Pantalla dónde se ve la información especifica de cada parcela, sus tratamientos y la estimación de sus costes.
Activación	▪ Pulsando en la acción de ver detalles de la parcela desde la página del listado de parcelas.
Diseño	 Figura 5.11: Interfaz 'Detalle de parcelas"
Eventos	▪ **EV-01**: Acceso a modificar la parcela. ▪ **EV-02**: Acceso a eliminar la parcela. ▪ **EV-03**: Acceso a toda la información sobre la parcela.

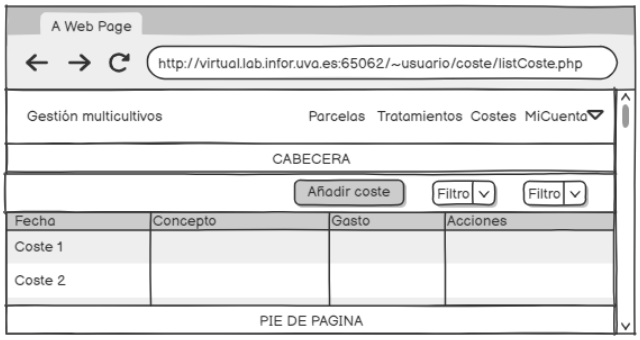
	Formulario Parcelas
Descripción	Pantalla dónde se está el formulario que permite el registro o modificación de una parcela.
Activación	▪ Pulsando en la acción de modificar parcela o en añadir parcela desde la pantalla de listado de parcelas
Diseño	 Figura 5.12: Interfaz “Formulario parcela”
Eventos	▪ **EV-01**: Acceso a campos para introducir los datos de la parcela. ▪ **EV-02**: Acceso a registrar o modificar la parcela. ▪ **EV-03**: Acceso volver a la página anterior.

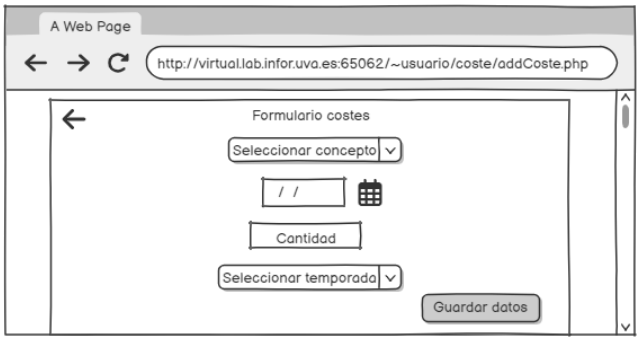
	Listado Tratamientos
Descripción	Pantalla dónde se ve un listado de los tratamientos registrados para las distintas parcelas en la aplicación.
Activación	▪ Pulsando en ver tratamientos dentro de la sección de tratamientos desde la home.
Diseño	 Figura 5.13: Interfaz “Lista de tratamientos”
Eventos	▪ **EV-01**: Acceso a los detalles de cada taratamiento. ▪ **EV-02**: Acceso a añadir un nuevo tratamiento. ▪ **EV-03**: Acceso a eliminar los tratamientos. ▪ **EV-04**: Acceso a modificar los tratamientos. ▪ **EV-05**: Acceso a filtrar el listado de tratamientos.

	Detalle Tratamientos
Descripción	Pantalla dónde se ve la información especifica de cada tratamiento y la parcela correspondiente.
Activación	▪ Pulsando en la acción de ver detalles del tratamiento desde la página del listado de tratamientos.
Diseño	 Figura 5.14: Interfaz “Detalle de tratamiento”
Eventos	▪ **EV-01**: Acceso a modificar el tratamiento. ▪ **EV-02**: Acceso a eliminar el tratamiento. ▪ **EV-03**: Acceso a toda la información sobre el tratamiento.

	Formulario Tratamientos
Descripción	Pantalla dónde se está el formulario que permite el registro o modificación de un tratamiento.
Activación	▪ Pulsando en la acción de modificar tratamiento o en añadir tratamiento desde la pantalla de listado de tratamientos
Diseño	 Figura 5.15: Interfaz “Formulario tratamiento”
Eventos	▪ **EV-01**: Acceso a campos para introducir los datos del tratamiento. ▪ **EV-02**: Acceso a registrar o modificar el tratamiento. ▪ **EV-03**: Acceso volver a la página anterior.

	Estimación de costes
Descripción	Pantalla dónde se calculan los costes totales por año, por temporada y por tipo de coste, además.
Activación	▪ Pulsando en la sección de costes en el menú de la pantalla principal.
Diseño	 Figura 5.16: Interfaz “Estimación costes”
Eventos	▪ **EV-01**: Acceso a añadir un nuevo gasto. ▪ **EV-02**: Acceso a ver el listado de los gastos registrados. ▪ **EV-03**: Acceso a filtros por año, temporada o tipo de coste para la estimación.

	Listado Costes
Descripción	Pantalla dónde se ve un listado de los costes registrados para los distintos tratamientos y parcelas en la aplicación.
Activación	▪ Pulsando en ver costes dentro de la sección de costes desde la home.
Diseño	 Figura 5.17: Interfaz “Lista de costes”
Eventos	▪ **EV-01**: Acceso a añadir un nuevo coste. ▪ **EV-02**: Acceso a eliminar los costes. ▪ **EV-03**: Acceso a modificar los costes. ▪ **EV-04**: Acceso a filtrar el listado de costes.

	Formulario Costes
Descripción	Pantalla dónde se está el formulario que permite el registro o modificación de un coste.
Activación	▪ Pulsando en la acción de modificar coste o en añadir coste desde la pantalla de listado de costes
Diseño	 Figura 5.18: Interfaz “Formulario costes”
Eventos	▪ **EV-01**: Acceso a campos para introducir los datos del coste. ▪ **EV-02**: Acceso a registrar o modificar el coste. ▪ **EV-03**: Acceso volver a la página anterior.

Capítulo 6

Implementación

En este capítulo se mostrarán los aspectos más importantes y relevantes sobre la implementación de la aplicación web. Para facilitar su explicación, el contenido se organizará en diferentes secciones.

6.1. Estructura del proyecto

En este apartado se presentará la estructura del proyecto, explicando la organización de las carpetas y archivos que componen la aplicación. Esta estructura ha sido diseñada para mantener un código ordenado, facilitar el mantenimiento y permitir una escalabilidad eficiente.

Estructura de las carpetas

- `.vscode`: Contiene las configuraciones del espacio de trabajo, en este caso, por ejemplo, el archivo `sftp.json`, el cual nos permite conectarnos a la máquina virtual y mantener los cambios actualizados de forma automática.
- `controladores`: Contiene los archivos PHP que sirven para manejar la lógica de control de flujo, determinando que respuesta enviar al usuario cuando realiza una solicitud, como con las distintas gestiones de parcelas, tratamientos, costes y usuarios.
- `coste`: Contiene los archivos PHP que representan las distintas vistas de la gestión de costes para el usuario, como añadir o modificar coste, listado de costes y su estimación.
- `css`: Contiene los archivos CSS utilizados para dar estilo a las diferentes páginas de la aplicación.
- img: Contiene las imágenes que se utilizan en la aplicación.

- `inc`: Contiene archivos PHP comunes, como por ejemplo pantillas, alertas e inicialización de código. También incluye dos subcarpetas, la primera de `inc/componentes`, dónde encontramos todos los formularios de la aplicación, y la segunda de `inc/modelos`, con los archivos que contienen las funcionalidades relacionadas con consultas a la base de datos.
- `inicio`: Contiene el archivo que representa la vista de la pantalla principal de la aplicación de un usuario registrado.
- js: Contiene los archivos JavaScript que contienen la funcionalidad interactiva del lado cliente.
- parcela: Contiene los archivos PHP que representan las distintas vistas de la gestión de parcelas para el usuario, como añadir o modificar parcelas, el listado e información específica.
- `políticas`: Contiene los archivos PHP que representan las distintas vistas de aviso legal, política de privacidad y de *cookies*.
- `tratamiento`: Contiene los archivos PHP que representan las distintas vistas de la gestión de tratamientos para el usuario, como añadir o modificar tratamientos, el listado e información específica.
- usuario: Contiene los archivos PHP que representan las distintas vistas de la gestión de usuarios, registro, inicio de sesión y la modificación de su información.
- index.php: Archivo que representa la vista de la pantalla principal al entrar en la aplicación.

A continuación, se muestra la estructura de carpetas y archivos del proyecto a bajo nivel, resaltando la organización interna. Esta imagen da una idea clara de cómo están dispuestos los diferentes elementos, facilitando la comprensión del diseño y la escalabilidad de la aplicación.

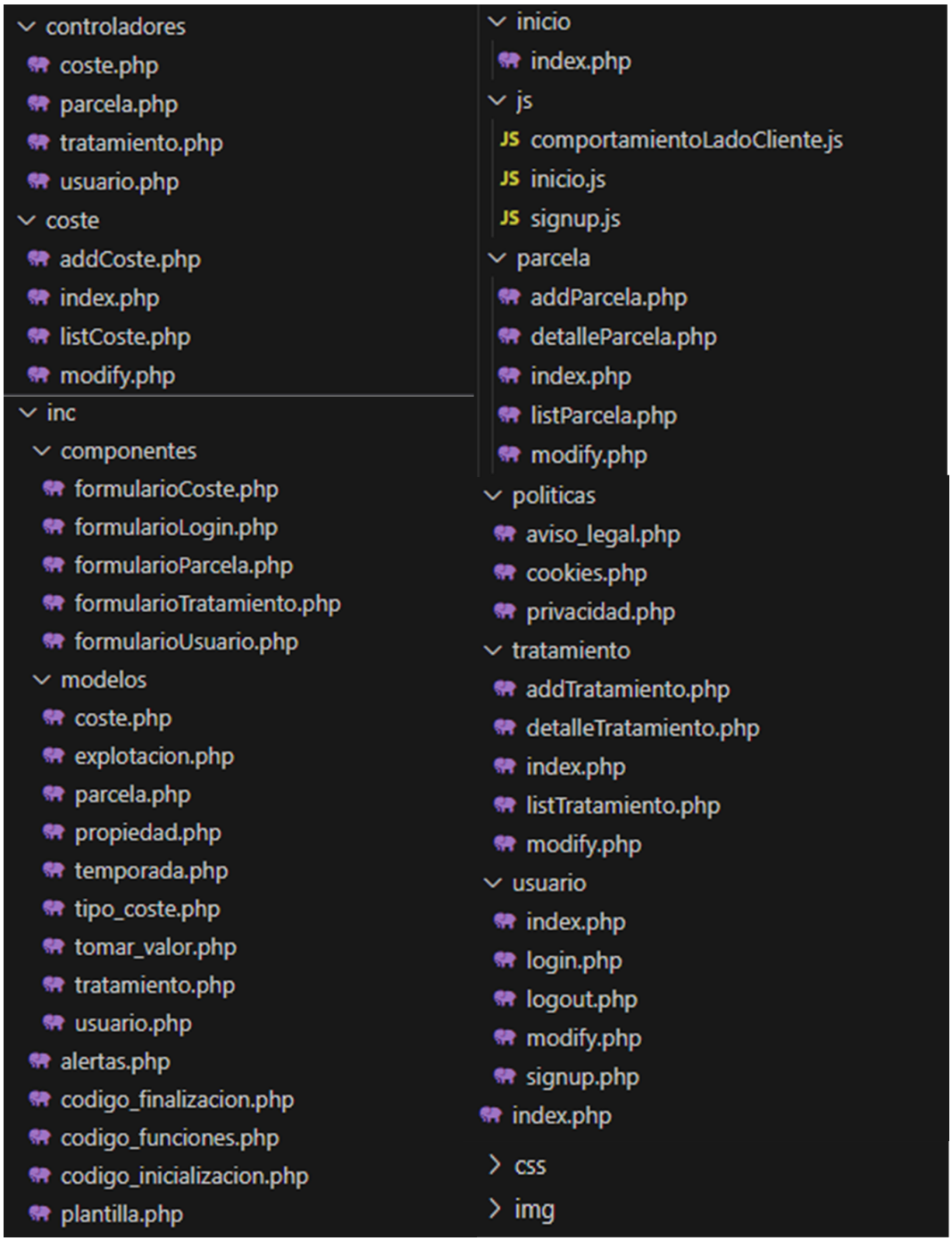


Figura 6.1: Estructura del proyecto

6.2. Detalles de implementación

Esta sección describe los principales aspectos técnicos del proyecto. Los temas incluyen la conexión a la base de datos, la seguridad en la inserción de datos, la gestión de sesiones

y autenticación, así como la implementación de las funcionalidades relacionadas con la gestión de parcelas, tratamientos y costes. Cada uno de estos elementos es importante para garantizar el funcionamiento adecuado, la integridad de los datos y la seguridad de la aplicacion.

6.2.1. Conexión a la base de datos

Se han creado dos archivos para gestionar las conexiones a la base de datos: `codigo_inicializacion.php` y `codigo_finalizacion.php`.

Estos archivos se encuentran al principio y al final de cada script PHP para garantizar una conexión adecuada de la base de datos, un uso eficiente de los recursos del sistema y evitar posibles problemas como pérdidas de memoria.

Inicialización de la conexión

- Primero se configura la sesión y el `buffer` de salida:

  ```
  session_start();
  ob_start();
  ```

- Se definen una serie de constantes que estandarizan los códigos de error utilizados en toda la aplicación que permiten un manejo más claro y uniforme de los errores y resultados en las diferentes operaciones

- Se establece la conexión con la base de datos mediante los siguientes parámetros:

  ```
  $host = ‘‘localhost’’;
  $usuario = ‘‘usuario’’;
  $password = ‘‘******’’;
  $conexionBD = mysqli_connect($host, $usuario, $password)
  or die("No se pudo conectar a la base de datos");
  ```

  Se especifican el *host*, el usuario, y la contraseña para conectarse a la base de datos. Si la conexión falla, se interrumpe la ejecución y se muestra un mensaje de error.

- Una vez establecida la conexión, se selecciona la base de datos en la que se va a trabajar:

  ```
  mysqli_select_db($conexionBD, "myapplication");
  ```

- Finalmente, se incluyen archivos que contienen las funciones comunes y los modelos necesarios para gestionar la lógica

Finalización de la conexión

- Se cierra la conexión establecida con la base de datos, liberando recursos del sistema:

  ```
  mysqli_close($conexionBD);
  ```

- Se vacía el buffer de salida con:

  ```
  ob_end_flush();
  ```

  Esto, asegura que todo el contenido generado durante la ejecución del script se envíe al navegador y luego se desactiva el buffering.

6.2.2. Seguridad en las consultas a la base de datos

Se abordará la seguridad en las consultas de base de datos mostrando cómo proteger la aplicación contra ataques de inyección SQL. Para ello, se utilizan sentencias preparadas.

Más adelante, se explica un ejemplo de una consulta segura para obtener una parcela por su ID.

Ejemplo de consulta segura: `getParcela()`

La función `getParcela()` obtiene los datos de una parcela específica, para ello se utilizan consultan preparadas:

```
function getParcela($conexionBD, $id)
{
    $sentenciaSQL = mysqli_stmt_init($conexionBD);
    $okFlag = mysqli_stmt_prepare($sentenciaSQL, 'SELECT * FROM PARCELA
    WHERE Id=?');
```

- `mysqli_stmt_init()`: Inicializa una sentencia preparada asociada a la conexión con la base de datos.
- `mysqli_stmt_prepare()`: Prepara la consulta SQL. El uso de `?` como marcador de posición evita la inyección de código malicioso, ya que los datos se pasan después de preparar la consulta.

  ```
  if ($okFlag) {
      mysqli_stmt_bind_param($sentenciaSQL, "d", $id);
  ```

- `mysqli_stmt_bind_param()`: Aquí se realiza la vinculación de los parámetros. La `"d"` indica que el tipo de dato a pasar es un número, lo que protege aún más el sistema asegurándose de que solo se pasen datos de tipo correcto.

```
        $okFlag = mysqli_stmt_execute($sentenciaSQL);

        $resultado = mysqli_stmt_get_result($sentenciaSQL);
```

- `mysqli_stmt_execute()`: Ejecuta la consulta preparada. Debido a que los parámetros ya se han vinculado y validado, la consulta se ejecuta de manera segura.
- `mysqli_stmt_get_result()`: Recupera el resultado de la consulta.

```
        if (mysqli_num_rows($resultado) > 0) {
            $fila = mysqli_fetch_assoc($resultado);
        } else {
            $fila = array();
        }
        mysqli_stmt_close($sentenciaSQL);
    }
```

- Verificación de resultados: Se comprueba si la consulta devolvió alguna fila. Si es así, se obtiene una fila de resultados, en caso contrario, se devuelve un array vacío.

```
    if ($okFlag) {
        $contexto["resultado"] = OK;
        $contexto["datos"] = $fila;
        return $contexto;
    } else {
        $contexto["resultado"] = ERROR_CONSULTA_PARCELA;
        return $contexto;
    }
}
```

- Manejo de errores: Si la consulta se ejecuta correctamente, el resultado es devuelto con un código de éxito (`OK`). Si algo falla, se devuelve un código de error específico (`ERROR_CONSULTA_PARCELA`).

6.2.3. Gestión de sesiones y autenticación

Proceso de autenticación (*Login*)

El formulario de *login* permite al usuario ingresar su correo electrónico y contraseña. Los datos se envían al controlador, que al recibir la solicitud de autenticación, recoge los datos enviados por el usuario y los limpia mediante la función `comprobar_entrada2()` para evitar inyecciones o entradas inválidas. Se llama a la función `loginUser()` para validar el *login*.

En caso de éxito, los datos del usuario se almacenan en la sesión (`$_SESSION`), lo que permite autenticar al usuario en el sistema. Si el *login* falla, el controlador envía los errores al formulario para que el usuario los corrija.

```
if ($contexto["resultado"] == OK && count($contexto["datos"]) > 0) {
    $_SESSION["Email"] = $contexto["datos"]["Email"];
    $_SESSION["Id"] = $contexto["datos"]["Id"];
    header("Location: ../inicio/?resultado=login-ok", true, 303);
} else {
    $contexto["errores"]["login"] = "No existe el usuario o la contraseña no es corre
    $_SESSION["contexto"] = $contexto;
    header("Location: ../usuario/login.php?resultado=error", true, 303);
}
```

Autonticación contra la base de datos (Funcion `LoginUser()`)

1. **Validación Inicial:** Primero, la función llama a `validarLogin()` para asegurarse de que los datos recibidos no contienen errores. Si los datos son incorrectos, se devuelve un contexto de error.

   ```
   if (empty($email)) {
       $contexto["errores"]["email"] = "El email es obligatorio";
   } elseif (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
       $contexto["errores"]["email"] = "Formato no válido";
   }

   if (empty($password)) {
       $contexto["errores"]["password"] = "La contraseña es obligatoria";
   }
   ```

2. **Cifrado de Contraseña:** Antes de realizar la consulta a la base de datos, la contraseña se cifra utilizando el algoritmo `md5()` junto con una sal (`SaltForPasswordEncryption`), de esta forma, la contraseña no se almacena ni transmite en texto claro, lo que protege frente a ataques comunes como la interceptación de credenciales en tránsito.

   ```
   $SaltForPasswordEncryption = "*****";
   $password = md5($SaltForPasswordEncryption . $password);
   ```

3. **Consulta a la Base de Datos:** Se ejecuta una consulta `SELECT` para obtener los detalles del usuario, en caso de que las credenciales sean correctas.

4. **Manejo del Resultado:** Si se encuentra un usuario que coincida con el email y la contraseña, se guarda la información en el contexto y se devuelve al controlador. De lo contrario, se devuelve un error indicando que las credenciales no son válidas.

Gestión de sesiones

Una vez que el usuario se autentica correctamente, el controlador guarda datos clave en la sesión, que luego serán utilizados para mantener el estado del usuario a lo largo de la sesión.

- Inicio de Sesión: La sesión se inicia con `session_start()` y se almacenan los datos del usuario en `$_SESSION`, lo que permite identificarlo en futuras interacciones con la aplicación.
- Protección de Rutas: El sistema puede restringir el acceso a ciertas páginas dependiendo de si la sesión está activa.

Cerrar sesión (*Logout*)

El proceso de cierre de sesión se realiza destruyendo todos los datos de la sesión y asegurando que no queden rastros de la sesión activa.

- Destruir la Sesión: Se vacía el *array* `$_SESSION` y se destruye la sesión en el servidor mediante `session_destroy`().
- Eliminar *Cookie* de Sesión: La *cookie* `PHPSESSID` se borra forzando su expiración, asegurando que la sesión no pueda reutilizarse en futuros accesos.
- Redirección: Finalmente, el usuario es redirigido a la página de inicio.

```
$_SESSION = array();
setcookie('PHPSESSID','',time() - 1);
session_destroy();
header('location: ../index.php');
```

6.2.4. Gestión de parcelas, tratamientos y costes

El flujo de gestión de parcelas, tratamientos y costes en una explotación agraria sigue un patrón similar en los tres casos, estructurado en torno a tres componentes principales: vistas, controladores y modelos.

Vistas

- Formularios de creación y edición:
  - Nuevo elemento: Los campos del formulario se inicializan vacíos para permitir al usuario introducir los datos de una nuevo elemento.
  - Edición: Si se está editando un elemento existente, el formulario se llena con la información actual del registro, recuperada mediante una función como, por ejemplo, `getParcela()`.

    - Validación: En ambos casos, el formulario valida la entrada del usuario, mostrando mensajes de error si los datos son incorrectos (por ejemplo, campos obligatorios vacíos o formatos incorrectos).

- Listado y visualización detallada:

    - Listado: se presenta como una tabla donde se pueden ver todas las entradas disponibles. Cada fila de la tabla tiene enlaces para ver detalles, editar o eliminar el registro.
    - Visualización detallada: al seleccionar un elemento para ver detalles específicos, que presenta toda la información relevante del elemento seleccionado..

Controladores

Actúan como intermediario entre la interfaz de usuario (vistas) y los modelos. Maneja las solicitudes del usuario y realiza operaciones sobre los datos.

- **Crear una nuevo elemento**: Si el `ID` es 0 (es decir, se está creando una nueva elemento), el controlador recoge los datos del formulario, los valida y llama a la función, por ejemplo, `addParcela()` del modelo para insertarlos en la base de datos.
- **Modificar un elemento existente**: Si el `ID` es distinto de 0, el controlador usa los datos introducidos para actualizar el elemento correspondiente mediante, por ejemplo, la función `modifyParcela()`.
- **Eliminar una Parcela**: El controlador llama a la función correspondiente para eliminar el registro correspondiente de la base de datos.
- **Listado**: Para mostrar un listado de todos los elementos, el controlador obtiene todos los registros de la base de datos y los pasa a la vista de listado (e.g., `index`).
- **Visualización detallada**: El controlador obtiene los detalles de un único elemento específico y los pasa a la vista de visualización específica, por ejemplo, `detalleParcela.php`).

En todos los casos, se gestiona el contexto del resultado de la llamada a la base de datos en el modelo, para mostrar notificaciones al usuario sobre el éxito o los errores en la operación realizada.

6.3. Modelos

Responsable de la lógica de acceso a la base de datos e interactúa directamente con las tablas que almacenan los datos.

- **Validación de Datos**: Funciones como `validarParcela()` comprueban que los campos obligatorios estén completos y que los datos tengan el formato correcto. Los errores se devuelven para mostrarse en el formulario.
- **Operaciones**: Para mostrar parcelas ya registradas, el modelo utiliza funciones como `getParcela()` o `getParcelas()`, que ejecutan consultas SQL para obtener la información desde la base de datos. Y, añadir, editar o eliminar elementos se utilizan funciones como `addParcela()`, `modifyParcela()` o `deleteParcela()`
- **Paginación y filtros**: Funciones como `getParcelasPaginados()` permiten gestionar la visualización de un gran número de registros de manera ordenada y eficiente.

Capítulo 7

Pruebas

En el desarrollo de software, asegurar la calidad del sistema es importante para garantizar su fiabilidad y rendimiento. Para lograr esto, se deben realizar pruebas exhaustivas que verifiquen si el sistema cumple con los requisitos establecidos, si funciona adecuadamente en distintas condiciones y si está libre de errores. Entre los métodos más comunes para realizar estas evaluaciones se encuentran las pruebas de caja negra y las pruebas de caja blanca. Estos enfoques permiten identificar problemas y asegurar que el sistema opere de manera óptima antes de su implementación final.

7.1. Pruebas de caja blanca

las pruebas de caja blanca se enfocan en analizar la estructura interna del software. En este tipo de pruebas, los evaluadores tienen acceso tanto al código fuente como a la lógica interna, lo que les permite diseñar pruebas específicas que cubran distintas rutas de ejecución y partes del código. Este enfoque es fundamental para asegurar la robustez y la integridad del software, así como para detectar posibles vulnerabilidades y errores lógicos.

En el marco de este proyecto, se han diseñado y ejecutado las siguientes pruebas durante el proceso de codificación:

- **Cobertura Completa del Código**: Se verifica que cada línea del código sea ejecutada durante las pruebas, para identificar cualquier segmento que no sea utilizado en la ejecución normal.
- **Pruebas Unitarias de Funcionalidades**: Se realizaron pruebas unitarias en cada función del código a medida que se desarrollaba, para garantizar que cada componente funcione de manera adecuada por separado.
- **Verificación de Sincronización y Comunicación con la Base de Datos**: Se revisa que no existan fallos en la comunicación entre la aplicación y la base de datos, asegurando un intercambio de información fiable.

- **Verificación de Formato de Datos**: Asegura que los datos ingresados y almacenados cumplan con los formatos requeridos, manteniendo la consistencia de la información.
- **Validación de Eliminación de Datos**: Verifica que al eliminar datos desde la interfaz, estos sean completamente removidos de la base de datos sin dejar rastros.
- **Prueba de Actualización de Datos**: Asegura que cualquier modificación realizada en la interfaz de usuario se refleje de manera correcta en la base de datos, manteniendo la integridad de la información.
- **Prueba de Consultas a la Base de Datos**: Se comprueba que las consultas generadas por la aplicación devuelvan los resultados correctos según los parámetros de búsqueda definidos.
- **Gestión de Errores**: Se comprueba que los errores que ocurran en la aplicación sean manejados adecuadamente, mostrando mensajes claros al usuario y ofreciendo posibles soluciones.
- **Prueba de Seguridad contra Inyección SQL**: Se garantiza que la aplicación esté protegida contra intentos de inyección SQL, evitando la ejecución de comandos no autorizados desde la interfaz.

7.2. Pruebas de caja negra

Las pruebas de caja negra se orientan a evaluar la funcionalidad de un sistema sin necesidad de conocer su estructura interna. En este tipo de pruebas, el evaluador no tiene acceso ni al código fuente ni a la lógica interna del software. En cambio, se enfoca en las entradas y salidas del sistema, observando cómo se comporta bajo diferentes condiciones. Este tipo de pruebas son clave para verificar la usabilidad, la interoperabilidad y la seguridad del software.

Estas pruebas se centrarán en aquellas situaciones en las que el usuario interactúe con el sistema, introduciendo datos o filtros que el sistema deberá procesar y evaluar para responder adecuadamente. A continuación, se presentan las pruebas realizadas:

ID	**PCN-01 Registro de usuario**
Objetivo	Comprobar la creación de un nuevo usuario en el sistema
Precondiciones	No exista un usuario previo con las mismas credenciales

Datos de entrada	▪ Nombre ▪ Apellidos ▪ Email ▪ Explotación ▪ Contraseña ▪ Repetir contraseña
Acción esperada	Se almacenará el usuario en el sistema al pulsar el botón de "Registrarse".
Resultado	Correcto

Tabla 7.1: Prueba de caja negra "Registro de usuario"

ID	**PCN-02 Inicio de sesión**
Objetivo	Comprobar los usuario inicien sesión en la aplicación.
Precondiciones	El usuario debe estar previamente registrado en la aplicación.
Datos de entrada	▪ Email ▪ Contraseña
Acción esperada	El usuario se autentica y es redirigido a la página principal.
Resultado	Correcto

Tabla 7.2: Prueba de caja negra "Iniciar sesión"

ID	**PCN-03 Modificar datos perfil de usuario**
Objetivo	Comprobar el usuario puede modificar sus datos personales.
Precondiciones	El usuario debe estar previamente registrado en la aplicación.

Datos de entrada	▪ Email nuevo
Acción esperada	El sistema guarda los nuevos datos.
Resultado	Correcto

Tabla 7.3: Prueba de caja negra “Modificar perfil de usuario"

ID	**PCN-04 Cerrar sesión**
Objetivo	Comprobar el usuario puede cerrar sesión.
Precondiciones	El usuario debe estar autenticado en la aplicación.
Datos de entrada	N/A
Acción esperada	El usuario cierra sesión y se redirige a la pantalla principal.
Resultado	Correcto

Tabla 7.4: Prueba de caja negra “Cerrar sesión"

ID	**PCN-05 Crear parcela**
Objetivo	Comprobar el usuario puede registrar una nueva parcela.
Precondiciones	El usuario debe estar autenticado en la aplicación.

Datos de entrada	▪ Provincia ▪ Municipio ▪ Zona ▪ Término ▪ Propietario ▪ Poligono ▪ Parcela ▪ Recinto ▪ Superficie
Acción esperada	El sistema guarda la parcela en la base de datos.
Resultado	Correcto

Tabla 7.5: Prueba de caja negra “Crear parcela"

ID	**PCN-06 Modificar parcela**
Objetivo	Comprobar el usuario puede modificar una parcela.
Precondiciones	El usuairo debe estar autenticado y la parcela debe haberse registrado previamente.
Datos de entrada	▪ Nueva provincia ▪ Nuevo municipio
Acción esperada	El sistema guarda la nueva información.
Resultado	Correcto

Tabla 7.6: Prueba de caja negra “Modificar parcela"

ID	**PCN-07 Eliminar parcela**
Objetivo	Comprobar el usuario puede eliminar una parcela.
Precondiciones	El usuario debe estar autenticado y la parcela debe haberse registrado previamente.
Datos de entrada	N/A
Acción esperada	Al pulsar el botón “Eliminar”, la parcela se elimina de la aplicación y se actualiza el listado de estas.
Resultado	Correcto

Tabla 7.7: Prueba de caja negra “Eliminar parcela"

ID	**PCN-08 Filtro de parcelas**
Objetivo	Comprobar filtrar las parcelas por el término funciona.
Precondiciones	El usuario debe estar autenticado en la aplicación.
Datos de entrada	▪ Filtrar por el término
Acción esperada	Muestra la lista actualizada con el filtro utilizado.
Resultado	Correcto

Tabla 7.8: Prueba de caja negra “Filtro de parcela"

ID	**PCN-09 Crear tartamiento**
Objetivo	Comprobar el usuario puede registrar un nuevo tartamiento.
Precondiciones	El usuario debe estar autenticado en la aplicación.

Datos de entrada	▪ Parcela ▪ Tratamiento ▪ Propiedades ▪ Fecha ▪ Temporada
Acción esperada	El sistema guarda el tartamiento en la base de datos.
Resultado	Correcto

Tabla 7.9: Prueba de caja negra “Crear tratamiento"

ID	**PCN-10 Modificar tartamiento**
Objetivo	Comprobar el usuario puede modificar un tartamiento.
Precondiciones	El usuario debe estar autenticado y el tartamiento debe haberse registrado previamente.
Datos de entrada	▪ Nueva parcela
Acción esperada	El sistema guarda la nueva información.
Resultado	Correcto

Tabla 7.10: Prueba de caja negra “Modificar tratamiento"

ID	**PCN-11 Eliminar tartamiento**
Objetivo	Comprobar el usuario puede eliminar un tartamiento.
Precondiciones	El usuario debe estar autenticado y el tartamiento debe haberse registrado previamente.
Datos de entrada	N/A

Acción esperada	Al pulsar el botón “Eliminar”, el tartamiento se elimina de la aplicación y se actualiza el listado de estos.
Resultado	Correcto

Tabla 7.11: Prueba de caja negra “ELiminar tratamiento"

ID	**PCN-12 Filtro de tartamientos**
Objetivo	Comprobar filtrar los tartamientos por la fecha y el tipo.
Precondiciones	El usuario debe estar autenticado en la aplicación.
Datos de entrada	▪ Filtrar una fecha y tipo específico
Acción esperada	Muestra la lista actualizada con el filtro utilizado.
Resultado	Correcto

Tabla 7.12: Prueba de caja negra “Filtro de tratamiento"

ID	**PCN-13 Crear costes**
Objetivo	Comprobar el usuario puede registrar un nuevo coste.
Precondiciones	El usuario debe estar autenticado en la aplicación.
Datos de entrada	▪ Concepto ▪ Cantidad ▪ Temporada ▪ Fecha
Acción esperada	El sistema guarda el coste en la base de datos.
Resultado	Correcto

Tabla 7.13: Prueba de caja negra “Crear coste"

ID	**PCN-14 Modificar coste**
Objetivo	Comprobar el usuario puede modificar un coste.
Precondiciones	El usuario debe estar autenticado y el coste debe haberse registrado previamente.
Datos de entrada	▪ Nueva fecha
Acción esperada	El sistema guarda la nueva información.
Resultado	Correcto

Tabla 7.14: Prueba de caja negra “Modificar coste"

ID	**PCN-15 Eliminar coste**
Objetivo	Comprobar el usuario puede eliminar un coste.
Precondiciones	El usuario debe estar autenticado y el coste debe haberse registrado previamente.
Datos de entrada	N/A
Acción esperada	Al pulsar el botón “Eliminar”, el coste se elimina de la aplicación y se actualiza el listado de estos.
Resultado	Correcto

Tabla 7.15: Prueba de caja negra “ELiminar coste"

ID	**PCN-16 Filtro de costes**
Objetivo	Comprobar filtrar los costes por la fecha y el tipo.
Precondiciones	El usuario debe estar autenticado en la aplicación.
Datos de entrada	▪ Filtrar una fecha y tipo específico
Acción esperada	Muestra la lista actualizada con el filtro utilizado.

Resultado	Correcto

Tabla 7.16: Prueba de caja negra “Filtro de costes"

ID	**PCN-17 Filtro de estimación de costes**
Objetivo	Comprobar calculo de estimación de costes al filtrar los costes por la fecha , temporada y tipo.
Precondiciones	El usuario debe estar autenticado en la aplicación.
Datos de entrada	▪ Filtrar una fecha, temporada y tipo específico
Acción esperada	Muestra el cáculo del gasto acorde al filtro elegido.
Resultado	Correcto

Tabla 7.17: Prueba de caja negra “Filtro de estimación de costes"

Capítulo 8

Conclusiones y trabajo futuro

8.1. Conclusiones

Al finalizar con este proyecto, puedo decir que mis objetivos se lograron y desarrollé una aplicación web eficiente y escalable para el manejo de una variedad de cultivos. Durante este proceso, aprendí la importancia de combinar una interfaz atractiva con funcionalidades que agreguen valor real a los usuarios, como la visualización de datos y el manejo de grandes cantidades de información. Estas características se implementan para permitir a los agricultores gestionar sus cultivos de forma más eficaz y precisa, que es uno de los objetivos principales.

Además del cumplimiento de los objetivos técnicos, he adquirido un conocimiento mucho más profundo de lo que implica el desarrollo de una aplicación completa, desde la planificación y diseño de la base de datos hasta la implementación de la lógica de negocio. Aunque ya tenía cierta experiencia previa con `PHP`, este proyecto me ha permitido profundizar mucho más en su uso, especialmente en la integración con `JavaScript` para crear una experiencia de usuario dinámica.

Por otro lado, he mejorado mis habilidades en la gestión de proyectos y la resolución de problemas complejos, utilizando una metodología iterativo-incremental. Esta metodología permitió avanzar de manera progresiva, ajustando el desarrollo según las necesidades. Cada obstáculo, ya sea al integrar nuevas funcionalidades o al optimizar el rendimiento, se convirtió en una oportunidad de aprendizaje y mejora continua.

En resumen, este proyecto ha sido una experiencia enriquecedora, ya que combina tecnología con utilidad práctica en un sector tan relevante como la agricultura. Ha facilitado la aplicación de conocimientos técnicos mientras se desarrollaba una solución útil para la gestión eficiente de cultivos, lo que me ha permitido crecer tanto técnica como personalmente.

8.2. Trabajo futuro

Aunque este proyecto ha cumplido con los objetivos planteados y ha demostrado ser una herramienta valiosa para la gestión de cultivos, siempre existen oportunidades de mejoras. Dado que se trata de un trabajo de fin de grado con un límite de tiempo y recursos, la aplicación web ha sido desarrollada dentro de estos parámetros. No obstante, hay áreas en las que se puede expandir y mejorar. Entre estas áreas se incluyen la incorporación de nuevas funcionalidades y la actualización del diseño para una experiencia de usuario aún más refinada. A continuación, se presentan algunos ejemplos de posibles mejoras que podrían elevar aún más la utilidad y el atractivo de la aplicación.

- Filtros Avanzados: Ampliar los filtros disponibles para los listados de parcelas, tratamientos y costes, permitiendo búsquedas más específicas y combinaciones de filtros múltiples, esto facilitaria una navegación más eficiente y una localización de lo que queramos buscar más rápida.
- Exportación de Datos: Incorporar la funcionalidad de exportar datos a formatos como `Excel` o `CSV` para facilitar el análisis fuera de la aplicación y permitir a los usuarios generar informes personalizados, realizar cálculos adicionales y compartir datos.
- Informes: Permitir la visualización de gráficos o informes detallados sobre el gasto acumulado y estimaciones, ofrecería a los usuarios una comprensión más clara de sus costos y tendencias.
- Automatización de Tareas: Implementar funcionalidades de automatización que permitan a los usuarios configurar recordatorios y tareas, como programar tratamientos periódicos para parcelas o generar informes automáticamente en intervalos definidos.
- Mejoras técnicas: Explorar el uso de *frameworks* modernos en `JavaScript` y otras tecnologías emergentes para facilitar futuras actualizaciones, integrando nuevas funcionalidades de manera más ágil y optimizando el rendimiento de la aplicación.

Parte III

Manuales de la Aplicación

Capítulo 9

Manual de Usuario

El propósito de este capítulo es proporcionar una guía clara y detallada para los usuarios sobre cómo utilizar las principales funciones de la aplicación.

9.1. Manual de Usuario

Para empezar a utilizar cualquier aplicación, hay que familiarizarse con su funcionamiento. Este manual del usuario proporciona instrucciones detalladas sobre cómo interactuar con el sistema, divididas en las principales funcionalidades de este proyecto:

1. Registro e inicio de sesión
2. Gestión de parcelas
3. Gestión de tratamientos
4. Gestión de costes

9.1.1. Registro e inicio de sesión

Una vez iniciada la aplicación, se mostrará en la pantalla principal dos opciones, donde el usuario podrá iniciar sesión o, si aún no tiene una cuenta, registrarse.

Registro de usuario

- Completar todos los campos del formulario de registro. La contraseña debe contener al menos una letra mayúscula, una letra minúscula, un número y tener una longitud mínima de 8 caracteres.
- Pulsar el botón "Registrarse" para finalizar el registro.

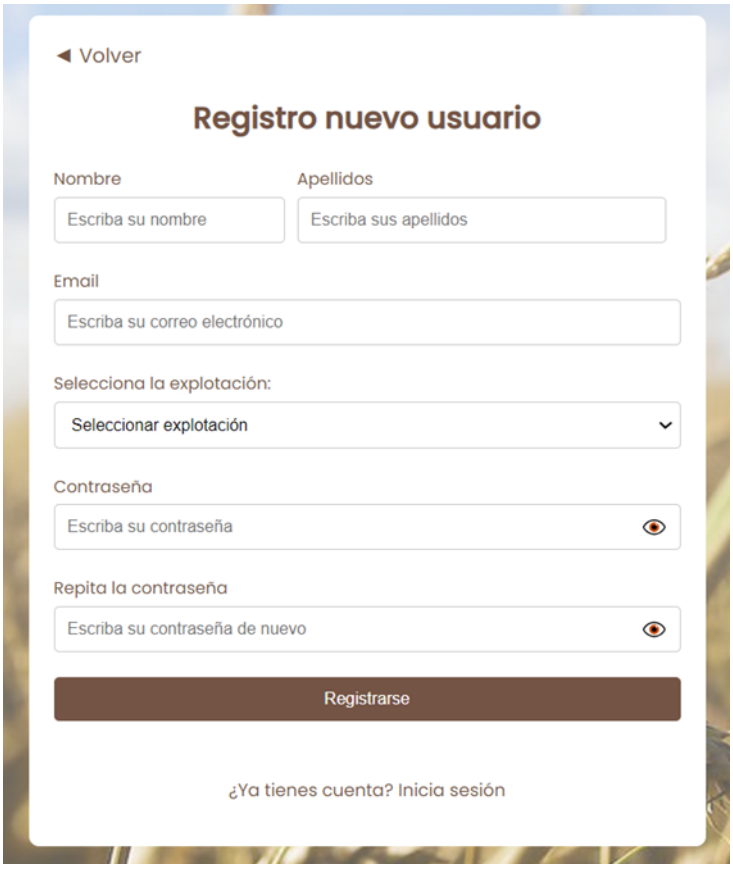


Figura 9.1: Captura de aplicación “Registro de usuario”

Inicio de sesión

- Introducir su correo electrónico y contraseña.
- Se puede hacer visible la contraseña utilizando el botón correspondiente.
- Pulsar el botón “Entrar”.
- Si las credenciales son incorrectas, aparecerá un mensaje de error, en caso contrario, se accederá exitosamente al sistema.

Figura 9.2: Captura de aplicación “Inicio de sesión”

9.1.2. Gestión de parcelas

Una vez dentro del sistema, el usuario puede gestionar parcelas pulsando en “Parcelas”, en el menú superior de la pantalla principal:

Listado de parcelas

Para ver el listado de las parcelas registradas, se pulsa en el botón “Ver parcelas”. Y veremos un listado como el siguiente:

Figura 9.3: Captura de aplicación “Listado parcelas”

Como se puede ver en la Figura 9.3 a partir de este listado, se puede filtrar las parcelas, se selecciona el término en el campo superior y automáticamente se actualizará el listado. Por otro lado también permite:

Crear o modificar parcela

- Pulsar en el botón "Añadir parcela", si lo que queremos es registrar una nueva, o pulsar dentro de la columna "Acciones" en "Editar" en la parcela en la que queremos cambiar datos.
- Introducir los campos del formulario.
- Pulsar el botón "Registar" o el botón "Guardar cambios".
- Si las credenciales son incorrectas, aparecerá un mensaje de error, en caso contrario, se completará la acción con éxito.

Figura 9.4: Captura de aplicación "Añadir/Editar parcela"

Ver detalles parcela

- Pulsar el botón "Consulta" en la parcela que queramos ver todos los detalles.
- La página mostrará todos los datos de la parcela, los tratamientos que tiene en el año actual y sus costes estimados.

Eliminar parcela

- Pulsar el botón "Eliminar" en la parcela que queramos eliminar.
- Aparecerá un aviso de alerta para confirmar la eliminación.
- Pulsar el botón "Aceptar" para eliminarlo permanentemente o "Cancelar" para parar la eliminación.

9.1.3. Gestión de tratamientos

Siguiendo de forma parecida a las parcelas, el usuario puede gestionar tratamientos pulsando en "Tratamientos", en el menú superior de la pantalla principal:

Listado de tratamientos

Para ver el listado de los tratamientos registradas, se pulsa en el botón "Ver tratamientos". Y veremos un listado como el siguiente:

Figura 9.5: Captura de aplicación "Listado tratamientos"

Como podemos ver, a partir del listado, se puede filtrar los tartamientos, se selecciona la fecha y el tipo en el campo superior y automáticamente se actualizará el listado. Por otro lado también se permite:

Crear o modificar tratamiento

- Pulsar en el botón "Añadir tratamiento", si lo que queremos es registrar un nuevo tratamiento, o pulsar dentro de la columna "Acciones" en "Editar" en el tratamiento en la que queremos cambiar datos.

- Introducir los campos del formulario.

- Pulsar el botón "Registar" o el botón "Guardar cambios".

- Si los datos son incorrectas, aparecerá un mensaje de error, en caso contrario, se completará la acción con éxito.

Figura 9.6: Captura de aplicación “Añadir/Editar tratamiento”

Ver detalles tratamiento

- Pulsar el botón “Consulta” en el tratamiento que queramos ver todos los detalles.
- La página mostrará todos los datos del tratamiento y de la parcela a la que corresponde.

Eliminar tratamiento

- Pulsar el botón “Eliminar” en el tratamiento que queramos eliminar.
- Aparecerá un aviso de alerta para confirmar la eliminación.
- Pulsar el botón “Aceptar” para eliminarlo permanentemente o “Cancelar” para parar la eliminación.

9.1.4. Gestión de costes

El usuario puede gestionar costes pulsando en “Costes”, en el menú superior de la pantalla principal:

Estimación de costes

Una vez hemos entrado en la sección de costes, podemos ver distintos cáculos de estimaciones de gatos, dependiendo de diferentes parámetros que podemos modificar eligiendo la fecha, la temporada o el tipo de gasto:

Figura 9.7: Captura de aplicación “Estimación costes”

Crear o modificar coste

Para crear o editar los datos de un coste, entramos en “Ver costes” desde la pantalla de estimaciones, aquí podremos ver un listado de todos los costes.

- Pulsar en el botón “Añadir coste”, para registrar un nuevo coste, o pulsar dentro de la columna “Acciones” en “Editar” para cambiar datos de uno concreto.
- Introducir los campos del formulario.
- Pulsar el botón “Registar” o el botón “Guardar cambios”.
- Si los datos son incorrectas, aparecerá un mensaje de error, en caso contrario, se completará la acción con éxito.

Figura 9.8: Captura de aplicación “Añadir/Editar coste”

Eliminar coste

- Pulsar el botón “Eliminar” en el coste que queramos eliminar.
- Aparecerá un aviso de alerta para confirmar la eliminación.
- Pulsar el botón “Aceptar” para eliminarlo permanentemente o “Cancelar” para parar la eliminación.

Webgrafía

[1] *A.Reto. Pruebas de Caja Negra vs. Pruebas de Caja Blanca, Diferencias Clave.* Visitado el 19 de agosto de 2024. 2024, 19 junio. URL: https://reto.com.mx/pruebas-de-caja-negra-vs-pruebas-de-caja-blanca-difere/.

[2] *Balsamiq: Fast, focused wireframing for teams and individuals*. Visitado el 8 de agosto de 2024. 2024. URL: https://www.balsamiq.com/.

[3] *Cómo instalar el servidor web Apache en Ubuntu 20.04 | DigitalOcean. (s. f.).* Visitado el 3 de noviembre de 2023. 2024. URL: https://www.digitalocean.com/community/tutorials/how-to-install-the-apache-web-server-on-ubuntu-20-04-es.

[4] *Cómo instalar phpMyAdmin con Apache en Ubuntu 20.04 - Tecnolitas. Tecnolitas.* Visitado el 15 de noviembre de 2023. 2022, 30 agosto. URL: https://tecnolitas.com/blog/instalar-phpmyadmin-con-apache/.

[5] *Entity-Boundary-Control – Ratón de biblioteca.* Visitado el 12 de agosto de 2024. 2021, 25 enero. URL: https://www.ratondbiblioteca.com/ebc.

[6] *Function Point Languages Table*. Visitado el 15 de abril de 2024. Quantitative Software Management. Agosto 2015. URL: http://www.qsm.com/resources/function-point-languages-table.

[7] *Glassdoor. Sueldos de la empresa.* Visitado el 5 de mayo de 2024. 2024. URL: https://www.glassdoor.es/Sueldos/index.html.

[8] *Home - VisualNACert.* Visitado el 2 de abril de 2024. 2024, 17 Julio. URL: https://visualnacert.com/language/esp/.

[9] *Inicio. (s. f.). AgroData.* Visitado el 2 de abril de 2024. 2024. URL: https://www.agro-data.es/.

[10] *Javier ¿Cuánto cuesta contratar un trabajador? Infoautonomos.* Visitado el 27 de marzo de 2024. 2022, 5 diciembre. URL: https://www.infoautonomos.com/blog/cuanto-cuesta-contratar-un-trabajador/.

[11] *Jesús. Ventajas y Desventajas de HTML y CSS: Explorando Sus Beneficios e Inconvenientes. Tutoriales Dongee.* Visitado el 7 de mayo de 2024. 2023, 25 diciembre. URL: https://www.dongee.com/tutoriales/ventajas-y-desventajas-de-html-y-css/#ventajas-de-html.

[12] *LaTeX, Evolved.* Visitado el 22 de marzo de 2024. 2023. URL: https://www.overleaf.com/.

[13] *Medina, M. Qué es phpMyAdmin y cómo usarlo. Hostinet.* Visitado el 19 de abril de 2024. 2023, 19 julio. URL: https://www.hostinet.com/formacion/bases-de-datos/que-es-phpmyadmin-y-como-usarlo/.

[14] *Software Architecture Guide.* Visitado el 20 de julio de 2024. 2019, 1 agosto. URL: https://www.infoautonomos.com/blog/cuanto-cuesta-contratar-un-trabajador/.

[15] *Software y Aplicación de gestión agrícola fácil de usar - Agroptima.* Visitado el 2 de abril de 2024. 2024. URL: https://www.agroptima.com/.

[16] *Tutorial de diagramas de despliegue. (s. f.). Lucidchart.* Visitado el 4 de junio de 2024. 2024. URL: https://www.lucidchart.com/pages/es/tutorial-de-diagramas-de-despliegue.