Escuela Politécnica Superior de Jaén

Grado en ingeniería informática

Universidad de Jaén

Escuela Politécnica Superior
de Jaén

TRABAJO FIN DE GRADO

MIACEITE: PROTOTIPO WEB PARA LA GESTIÓN DEL ACEITE DE OLIVA

Alumno

Álvaro González González

Tutores

Víctor Manuel Rivas Santos
(Departamento de Informática)

José Ignacio Gómez Espínola
(Departamento de Informática)

Septiembre, 2019

Universidad de Jaén

Departamento de Informática

Don Víctor Manuel Rivas Santos y Don José Ignacio Gómez Espínola, tutores del Trabajo Fin de Grado titulado: '**MiAceite: prototipo web para la gestión del aceite de oliva**', que presenta Don Álvaro González González, otorgan el visto bueno para su entrega y defensa en la Escuela Politécnica Superior de Jaén.

Jaén, Septiembre de 2019

El alumno:

Álvaro González González

Los tutores:

RIVAS SANTOS VICTOR MANUEL - 29798183G

Firmado digitalmente por RIVAS SANTOS VICTOR MANUEL - 29798183G
Fecha: 2019.09.04 18:47:39 +02'00'

GOMEZ ESPINOLA JOSE IGNACIO - 24277832K

Firmado digitalmente por GOMEZ ESPINOLA JOSE IGNACIO - 24277832K
Nombre de reconocimiento (DN): c=ES, serialNumber=24277832K, sn=GOMEZ ESPINOLA, givenName=JOSE IGNACIO, cn=GOMEZ ESPINOLA JOSE IGNACIO - 24277832K
Fecha: 2019.09.04 18:27:20 +02'00'

Víctor Manuel Rivas Santos

José Ignacio Gómez Espínola

Agradecimientos

Mis agradecimientos a todos los que han colaborado a lo largo de la realización de este Trabajo Fin de Grado. Agradecer de forma expresa a mis dos tutores, que han mostrado su afán por la enseñanza y me han ayudado a lo largo del desarrollo de este proyecto.

Agraceder también a mis amigos y compañeros con los que he compartido horas y horas de estudio, prácticas y otros buenos momentos a lo largo de toda la carrera.

En especial, agradecer a mi familia y a mi pareja por la ayuda ofrecida, así como el esfuerzo que ha conllevado que yo haya podido llegar hasta el final de esta etapa.

Índice de contenido

Índice de ilustraciones

Índice de tablas

ÍNDICE DE TABLAS

1 - INTRODUCCIÓN

1.1 - Introducción al proyecto

El presente Trabajo Fin de Grado trata sobre el desarrollo de un prototipo de una aplicación web destinada al mundo del olivar.

A través de esta aplicación se ofrecerán una serie de servicios a los usuarios para facilitarles tareas administrativas y de gestión.
Para ello, se desarrollará una parte de administración o back-end a través de la cual los usuarios podrán administrar sus datos y otra de visualización de la interfaz o front-end de los servicios ofrecidos al resto de usuarios.

Para qué necesidades son las que actualmente tienen los usuarios y saber cuáles de ellas son las que más les preocupan, se obtendrán datos de encuestas realizadas a usuarios relacionados con el sector oleícola.

El desarrollo de la aplicación web será llevado a cabo con Java, en concreto con el framework Spring. Para la parte visual de la aplicación se utilizará la biblioteca de diseño de código abierto Bootstrap.

1.2 - Motivación

Tras haber podido vivir la experiencia de cómo se desarrolla la recolección de la aceituna para la producción de aceite durante algunos años de forma directa, al ser un joven jiennense y vivir en un pequeño pueblo de interior de la provincia, nació la inquietud de facilitar ciertos servicios a las personas que tienen menos recursos.

La idea de este proyecto surge para combatir las distintas problemáticas que los pequeños agricultores, e incluso algunos de mayor envergadura, sufren durante cada campaña de recogida de aceituna. Estos problemas pueden variar desde el momento en el que el agricultor empieza a buscar trabajadores para este periodo de tiempo, hasta la dificultad de dejar en vigor los documentos necesarios para su contratación.

Todo ello se debe a que la mayoría de pequeños agricultores son de edad avanzada a los que no llegan las tecnologías y pertenecen a pequeños pueblos de interior de ciudades andaluzas, por lo que necesitan buscar a trabajadores fuera de ellos. A esto se suma que muchos de los trabajadores que se ofrecen para su contratación son inmigrantes, personas desempleadas o jóvenes que han abandonado los estudios, todos ellos sin muchos recursos para desplazarse a buscar trabajo a estos pequeños pueblos o incluso sin recursos tecnológicos para poder encontrar ofertas de trabajo en distintos portales de empleo, debido a que este sector está alejado de las nuevas tecnologías, por lo que uno de los principales objetivos es reducir esta brecha digital establecida.

Debido a lo comentado anteriormente, nace la idea de crear la app web “MiAppceite”, en la que el usuario podrá encontrar un portal de empleo, un sistema de gestión de fincas personal, una plataforma de ofertas de cursos online para la obtención de los distintos permisos necesarios para la realización de las labores agrarias, una sección de noticias del sector y enlaces a distintas tiendas de material agrícola.

A través de esta plataforma se concienciará al usuario de la necesidad de un buen uso del agua, de herbicidas y abonos con el fin de lograr un olivar sano y

sostenible a la vez que se mejora el rendimiento, calidad y producción del aceite. Resumen de las principales deficiencias identificadas

1.3 - Objetivos

1.3.1 - Objetivo general

El objetivo general del proyecto es el desarrollo de un prototipo de aplicación web destinada al sector oleícola a través de la cual el trabajo agrícola deje de ser una actividad laboriosa y anticuada, y pase a ser el mejor medio de vida posible. Como misión social, se quiere facilitar el acceso a puestos de trabajo a todas las personas con recursos más bajos, colaborando con instituciones públicas y asociaciones a través de las cuáles ofrecer las ofertas de empleo registradas en el portal.

1.3.2 - Objetivos específicos

Para poder desarrollar el objetivo general establecido, se deberán de superar los siguientes puntos:

1. Estudiar las necesidades actuales de los usuarios potenciales de la aplicación.
2. Validar los requisitos obtenidos de los usuarios tras los resultados obtenidos.
3. Desarrollar el requisito más valorado por los usuarios.
4. Establecer una metodología de Ingeniería del Software para el desarrollo del prototipo.

2 - ESTADO DEL ARTE

2.1 - Introducción

El mundo de la agricultura ya se está viendo involucrado en el uso de las nuevas tecnologías para la optimización y mejora de sus procesos, y es que la clave para esta mejora, son los datos. Es por eso que la agricultura está llamada a ser partícipe de la llamada revolución digital y pasar a denominarse de forma característica en este campo como “Revolución digital agraria”.

La competitividad actual en un mercado globalizado hace que nuestra agricultura esté casi obligada a sumarse a este cambio. La resiliencia y un sector agrícola inteligente harán que podamos conseguir adaptarnos a las nuevas condiciones climáticas, que podamos adaptarnos a las exigencias del consumidor que cada vez está más concienciado con la sostenibilidad del medio ambiente y, aspecto muy importante en nuestra tierra, Andalucía, mantener y fortalecer el tejido social y económico de las zonas rurales.

Para llevar a cabo esta adopción en el medio rural, los pasos deberán darse poco a poco, y es que es necesario que el agricultor sea el protagonista de estos cambios para que la transformación sea lo más efectiva posible y duradera. Pero lo más importante es lograr que el sector agrario sea llamativo para las nuevas generaciones y poder dar ese relevo generacional que nos de la posibilidad de dar continuidad a la principal actividad económica de Andalucía. [2]

2.2 - Soluciones para la agro-tecnología

2.2.1 - BoosterAgro

Proyecto destinado a la aplicación de la inteligencia artificial en la mejora de las predicciones climáticas en la agricultura. La predicción climática se centra mayormente en predecir las condiciones meteorológicas en un tiempo futuro en concreto. Ya existen diversos mecanismos para llevar a cabo estas predicciones que tan necesarias son para compañías de seguros, empresas agroindustriales, agricultores, bancos, etc. Pero en este caso, **BoosterAgro** está destinada a aplicar la

inteligencia artificial en el proceso de predicción del tiempo a través de un proceso de grabación y del comportamiento de data histórica pasada, con la finalidad de ayudar en la toma de decisiones a los actores en la cadena del agro. [3]

Ilustración 1: Logo BoosterAgro

2.2.2 - Nanny

Proyecto destinado a conseguir mejores cosechas y reducir el riesgo de cultivos hidropónicos. Hacen uso de un dispositivo conectado y una app. El dispositivo está formado por sensores que están en constante vigilancia del cultivo. A través de estos sensores, la app es capaz de enviar alertas al móvil cuando las plantas están en peligro, e incluso, la app te indica qué se debe llevar a cabo para evitar que las plantas sigan dañándose.

A través de la hidroponía se podrá lograr el cultivo y jardines verticales, cosechar en el desierto, en casa, en el espacio, ahorrar un 90% de agua y, llegar a multiplicar la producción gracias a ella. [4]

Ilustración 2: Logo Nanny

2.2.3 - Cropti

Esta aplicación está destinada a la ayuda de técnicos de cooperativas y agricultores a gestionar el cuaderno de explotación. Tiene gran utilidad, debido a que permite a los agricultores rellenar y descargar de forma fácil el modelo oficial de MAGRAMA. También permite la exportación de los datos, ordenar por campaña, cultivos, parcelas, etc. Con el objetivo de cumplir con la normativa que se aplica a las explotaciones agrícolas con ayudas PAC. Esta normativa establece el marco de actuación para poder conseguir un uso sostenible de los productos fitosanitarios aplicados en una explotación agrícola. En él se indica que los agricultores deben registrar las labores agrarias en un documento llamado “Cuaderno de explotación”. Es desde aquí desde donde se comienza el proceso de trazabilidad de los cultivos, con la finalidad de conseguir unos productos de mayor calidad y seguros para los consumidores, garantizando a su vez la competitividad y viabilidad económica de las explotaciones. [5]

Ilustración 3: Captura web Cropti

2.2.4 - Agroguía

Sistema de guiado agrícola GPS que está diseño especialmente para la realización de las labores de abonado y la aplicación de herbicidas. Este sistema permite a los usuarios llevar a cabo tratamientos sin solapar ni dejar faltas, manteniendo así la distancia a la anterior pasada con el fin de ahorrar dinero, tiempo y trabajo.

Este sistema está diseñado para que el usuario pueda usarlo de forma fácil, pueda marcar las zonas tratadas, las que faltan aún por tratar y las zonas solapadas.

Permite también medir el área de las parcelas, calcular la distancia entre dos puntos, calcular la velocidad y área tratada, etc. Permite la exportación de trabajos al PC.

El sistema consiste en una pantalla que se instala en la cabina del tractor que, mediante un recetor GPS de alta precisión, muestra en la pantalla el mapa donde se pueden ver las zonas por las que ya se ha pasado y que, por tanto, están tratadas. [6]

Ilustración 4: Kit agroguía

2.3 - Por qué desarrollar una solución para la agro-tecnología

Debido al mundo que nos espera en unos años con más habitantes, globalizado, con recursos más limitados y urbano, entra en desafío el desarrollo de soluciones que sean capaces de resolver estos problemas.

Se estima que para el año 2050 deberemos de ser capaces de proveer alimentos a una población global que rondará los 10.000 millones de personas, pero con la pecularidad de que, tendremos la mitad de los recursos con los que contamos hoy día. Entre ellos cabe mencionar:

- Menos agua
- Menos tierra cultivable

- Menos agroquímicos autorizados y disponibles
- Menos recursos pesqueros
- Un gran cambio climático

Ante esta problemática, los productores agrícolas y agropecuarios deberán de incrementar la productividad de sus cultivos a través de nuevas tecnologías digitales en la producción agrícola y ganadera. Es por eso, que deben de ser la innovación y la sostenibilidad las principales características que encabecen este nuevo desafío de alimentar al mundo en el año 2050.

A través del desarrollo y generación de procesos flexibles, automatizados, conectados e inteligentes seremos capaces de dar respuesta a la demanda de un mercado exigente bajo criterios de sostenibilidad, transparencia y personalización.

Somos protagonistas de la emergencia de innovaciones disruptivas que sean capaces de poner en duda los sistemas establecidos y responsables de crear nuevos modelos de negocio, modelos de negocio con unos usos y costumbres desconocidos hasta ahora por el consumidor. [7]

3 - HERRAMIENTAS Y FRAMEWORKS

3.1 - Frameworks utilizados

3.1.1 - Spring Framework

Spring Framework [8] es un framework de desarrollo de código abierto que ofrece un modelo de programación completo y una configuración basada en Java para aplicaciones empresariales.

Spring lleva a cabo una gestión de configuración basada en componentes, JavaBeans, y aplica el principio de Inversión de Control (IoC), específicamente utilizando la inyección de dependencias (DI) para manejar las relaciones entre objetos. Esto hace que se de un bajo acoplamiento y una alta cohesión, mejorando así la reutilización y mantención de los componentes.

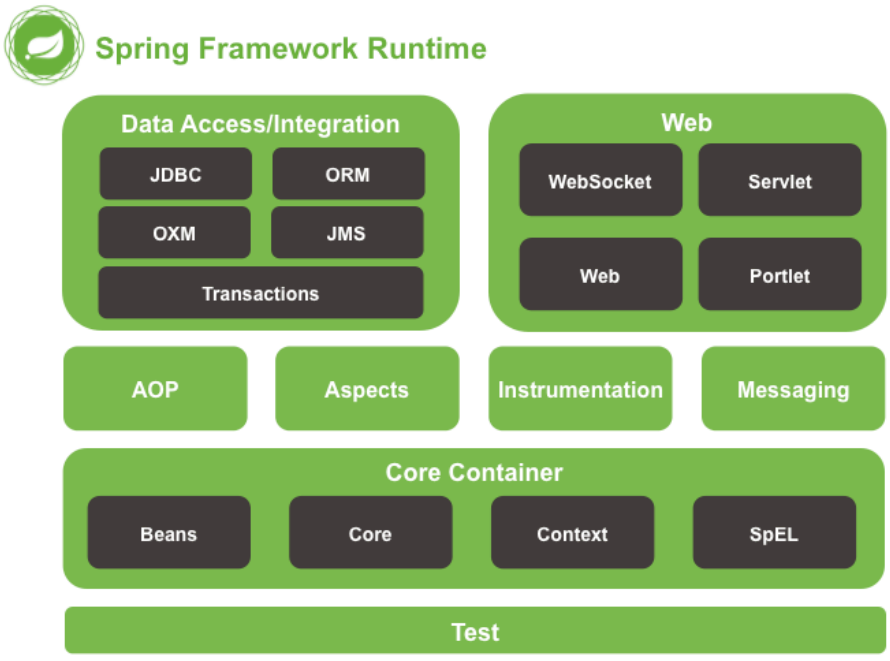

Ilustración 5: Spring Framework Runtime

3.1.2 - Spring Boot

Spring Boot está diseñado para facilitar la puesta en marcha de un proyecto basado en Spring. Con Spring Boot se busca una configuración inicial mínima para poder lanzar el proyecto.

3.1.3 - ORM y persistencia

Spring ofrece una alta abstracción sobre la API de JDBC, así como también ofrece una serie de plantillas para la integración con frameworks de persistencia como Hibernate, JPA, etc.

También ofrece soporte a la lógica de negocio de la aplicación, específicamente a través de clases de acceso a datos, conocidas como DAO.

Otra de las características de la persistencia en Spring es la de los componentes encargados de la gestión de transacciones de base de datos.

3.1.4 - Spring MVC

Con Spring también podemos hacer uso de una arquitectura Modelo Vista Controlador (MVC), que es uno de los principales componentes y tecnologías actualmente.

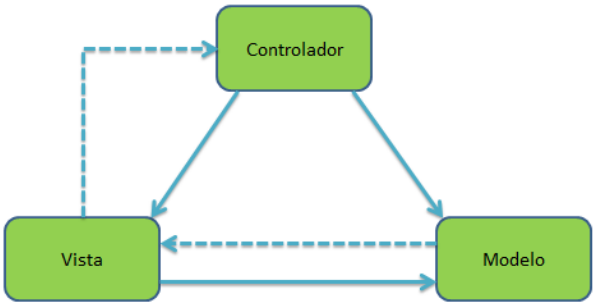

Ilustración 6: Spring MVC

Spring soporta varias tecnologías para la generación de las vistas, entre ellas:

1. JSP
2. Thymeleaf
3. FreeMarker
4. POI

3.1.5 - Bootstrap

Biblioteca destinada al diseño de sitios web y aplicaciones, centrada sólo en el front-end de los proyectos en los que es utilizada. Es una biblioteca multiplataforma y con un conjunto de herramientas de código abierto.

Pueden ser utilizadas distintas plantillas de diseño con menús de navegación, botones, formularios, así como una gran cantidad de componentes basados en HTML y CSS. También pueden encontrarse extensiones adicionales basadas en JavaScript, concretamente en la librería jQuery de JavaScript.

Para este proyecto ha sido utilizado la versión 4.3.1 de Bootstrap [9].

3.1.6 - Thymeleaf

Es un moderno motor de plantillas Java del lado del servidor para entornos web. El principal objetivo que busca Thymeleaf es el de aportar elegantes plantillas, permitiendo así el *natural templating*, que permite que tanto el trabajo de maquetación del proyecto y la parte de programación puedan realizarse a la vez haciendo uso de atributos en lugar de etiquetas.

Permite una integración completa con Spring, framework que hemos elegido para el desarrollo del proyecto. Aún así, también puede ser utilizado con otros frameworks como Play, Java EE 8 MVC…

Para este proyecto ha sido utilizado la versión 3.0.11 de Thymeleaf. [10]

3.1.7 - jQuery

Biblioteca multiplataforma de JavaScript que ayuda la simplificación de la manera en que se interactúa con los documentos HTML del proyecto, permite manipular el árbol DOM, controlar eventos, agregar interacciones con AJAX, crear animaciones, etc.

Es un software de código abierto y libre que ofrece funcionalidades basadas en JavaScript. Su sintaxis facilita la navegación por el documento debido a la forma en la que está diseñado.

Para este proyecto ha sido utilizado la versión 3.2.1 de jQuery. [11]

3.2 - Herramientas utilizadas

3.2.1 - Eclipse

Eclipse [12] es una plataforma de software que está compuesta por una serie de herramientas de programación open source. Eclipse puede combinar fácilmente el soporte de idiomas y otras funciones en cualquiera de los paquetes predeterminados. Con Eclipse Marketplace también se permite una personalización y extensión virtual ilimitadas.

Ilustración 7: Logo Eclipse

3.2.2 - MySQLWorkbench

MySQL Workbench [13] es una herramienta visual unificada destinada a desarrolladores, administradores de bases de datos y arquitectos de bases de datos. MySQL Workbench está disponible para las plataformas Linux, Windows y Mac OS X.

MySQL Workbench proporciona modelado de datos, desarrollo de SQL y herramientas de administración integrales para la administración de usuarios, copia de seguridad, configuración del servidor, etc.

Ilustración 8: Logo MySQL Workbench

3.2.3 - Visual Paradigm

Visual Paradigm [14] es una herramienta CASE (Computer Aided Software Engineering – Ingeniería de Software Asistida por Computadora). Ofrece un conjunto de soluciones para el desarrollo de software, desde la planificación, hasta el desarrollo y documentación, pasando por el análisis y diseño del proyecto.

Algunas de las soluciones que ofrece son:

1. Tabla de decisiones
2. Diagrama de secuencia
3. Casos de uso
4. Diagrama de entidad relación
5. UML
6. Etc.

Disponible en varios e idiomas y para varias plataformas, como son Windows, Linux y Mac OS X.

Ilustración 9: Logo Visual Paradigm

3.2.4 - Formularios de Google

Formularios de Google [15] facilita a los usuarios una serie de servicios para que éstos puedan planificar eventos, gestionar inscripciones, recopilar información, etc.

Permite crear encuestas personalizadas ofreciendo al usuario distintos temas y estilos, así como plantillas prediseñadas que pueden ser seleccionadas para ser utilizadas en las encuestas deseadas.

Las respuestas se recogen de forma automática y pueden verse los resultados en tiempo real en distintas gráficas. Los datos almacenados también pueden ser exportados a distintos formatos.

Ilustración 10: Logo Google Forms

4 - INGENIERÍA DEL SOFTWARE

4.1 - Introducción

En este apartado se desarrollará la parte de ingeniería de requisitos, rama de la Ingeniería del Software que trabaja con objetivos del mundo real, servicios requeridos, restricciones, etc. Con el propósito de especificar el comportamiento del sistema y hacer que los requisitos se encuentren perfectamente verificados y validados antes de alcanzar la fase de diseño en el proyecto. [16]

Se diferenciará entre dos tipos de requisitos, funcionales y no funcionales:

Requisitos funcionales: servicios proporcionados por el sistema, cómo debe de actuar el sistema ante una entrada particular y cómo se debe comportar ante ciertas situaciones.

Requisitos no funcionales: restricciones relacionadas con los servicios ofrecidos por el sistema, relacionadas con la calidad del mismo.

Antes de comenzar, resaltar lo que Frederick Brooks señaló sobre la ingeniería de requisitos: *"La parte más difícil en la construcción de un sistema software es precisamente decidir qué construir. Ninguna otra parte del proyecto es tan difícil como establecer de forma detallada los requisitos técnicos. Ninguna otra parte del trabajo realizado afecta tanto al resultado final, si se hace de forma incorrecta. En ninguna otra parte es tan difícil de rectificar los errores cometidos."*

4.2 - Definición de requisitos

En primer lugar, se indicarán de forma breve una serie de requisitos iniciales que deberá incluir el sistema, los cuáles serán detallados de forma más exhaustiva posteriormente, así como evaluados por una serie de usuarios que serán encuestados para la validación de cada uno de los requisitos antes de comenzar con su desarrollo.

La encuesta de evaluación de requisitos será llevada a cabo entre distintos usuarios potenciales, siendo cada uno de estos de distinta edad, competencias tecnológicas y conocimientos agrarios, con el fin de obtener un resultado final más positivo debido a la diversidad de puntos de vista.

Objetivos

1. El sistema constará de dos tipos de usuarios, usuario registrado y usuario no registrado.
    a. El usuario registrado podrá acceder a una serie de recursos adicionales del sistema además de los disponibles para el usuario no registrado.

   b. El usuario no registrado solo podrá acceder a los recursos del sistema que serán de carácter informativo.

2. El sistema ofrecerá un apartado de ofertas laborales.
3. El sistema deberá tener un diseño intuitivo y responsive.
4. El sistema deberá tener espacios reservados para banners publicitarios.
5. El sistema contará con una página de noticias relacionadas con el sector oleícola.
6. El sistema contará con una página de catálogo de cursos disponibles.
7. Las funcionalidades del usuario registrado serán:
   a. El usuario registrado podrá activar la opción de gestionar fincas.
      i. Si la opción gestionar fincas está activada, el usuario tendrá acceso a la página de gestión personal de fincas.
      ii. El usuario podrá añadir parcelas.
      iii. El portal deberá ofrecer el sistema SIGPAC para la búsqueda de parcelas.
      iv. El usuario podrá añadir información relacionada de cada parcela.
      v. El usuario podrá añadir trabajadores.

Definición detallada de los requisitos del sistema

Como una primera versión de definición de requisitos del sistema, se han detallado una serie de aspectos que he creído personalmente que pueden ser interesantes para los usuarios a los que está destinada esta aplicación.

***Versión 0.0**. Versión inicial previa a la primera validación de usuario.*

Requisitos funcionales

Requisito 1: Portal de empleo.

El sistema deberá ofrecer un portal de empleo en el que los **usuarios registrados** podrán añadir ofertas de trabajo.

Requisito 2: Página de noticias.

El sistema deberá ofrecer un apartado exclusivo para noticias relacionadas con el sector.

Requisito 3: Plataforma e-learning.

El sistema ofrecerá al usuario un catálogo de cursos disponibles a los que poder matricularse.

Requisito 4: Sistema de gestión de fincas.

El sistema ofrecerá al usuario un sistema de gestión de fincas personal que será de acceso privado.

Requisitos no funcionales

Requisito 1: Interfaz de usuario y diseño responsive.

El sistema deberá ofrecer al usuario un diseño intuitivo en el que cada usuario que acceda sea capaz de comprender su total funcionamiento.

El sistema deberá ofrecer al usuario un sistema adaptable a cualquier dispositivo, ya sean smartphones, tablets, laptops o desktops.

El sistema deberá ofrecer en cada una de las páginas que visiten los usuarios no registrados espacios reservados a posibles banners publicitarios.

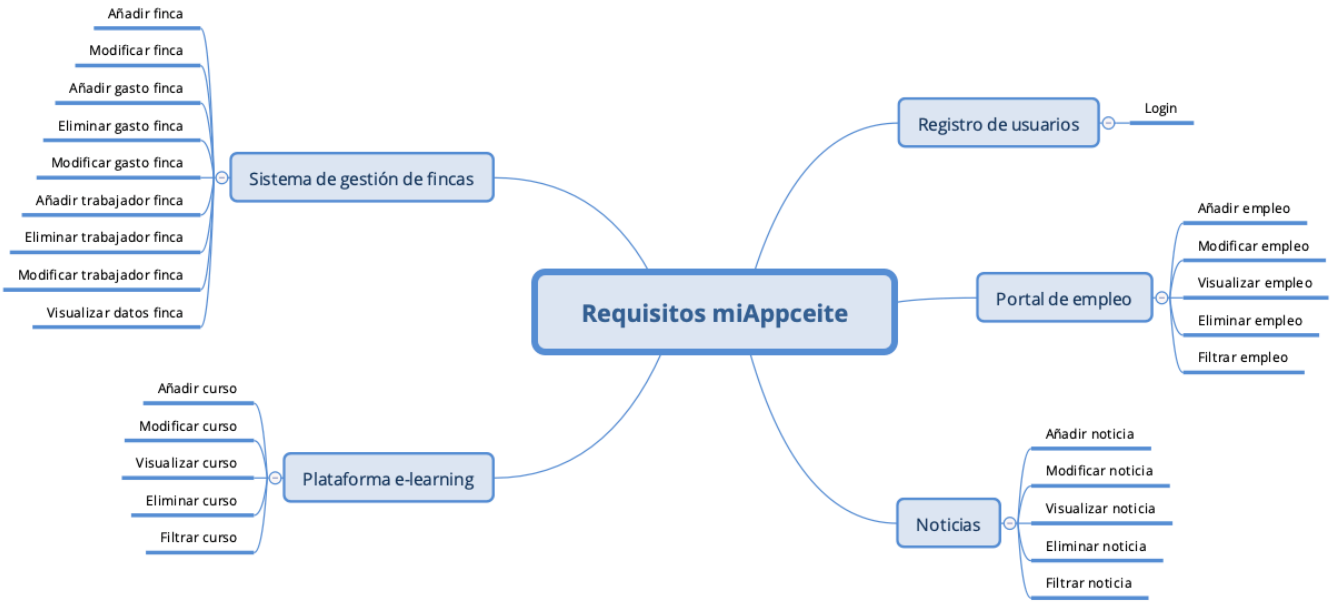

Ilustración 11: Mapa mental requisitos MiAceite

4.3 - Planificación

Atendiendo al período docente establecido por la Universidad de Jaén durante el segundo cuatrimestre (28/01/2019 al 17/05/2019), se ha basado la planificación de horas respecto a los días de período lectivo de entre los meses de enero a mayo de 2019 y el período no lectivo de entre los meses de mayo y agosto (18/05/2019 al 30/08/2019, exceptuando 15 y 16 de agosto), con un total de 292 horas, contando con que se trabajarán unas 2 horas diarias, durante los 73 días de período lectivo que marca el calendario.

Tras la finalización del periodo lectivo, se llevará a cabo la realización del proyecto entre los meses de mayo y agosto (18/05/2019 al 30/08/2019, exceptuando 15 y 16 de agosto), de cara a la última semana de entrega posible del proyecto.

Día de comienzo del proyecto: 28/01/2019.
Día de fin de proyecto: 30/08/2019.

Total de días de duración del proyecto: 146 días.
Total de horas diarias trabajadas: 2 horas.
Total de horas trabajadas durante el proyecto: 292 horas.

Entregas	Versión	Validación	Duración
Entrega 0	V 0.0	*Versión inicial previa a la primera validación de usuario.*	7 días
Entrega 1	V 1.0	*Validación general del proyecto*	10 días
Entrega 2	V 2.0	*Validación de PMV v1.0*	30 días

Tabla 1: Planificación inicial

Total de días para las entregas 0,1 y 2 es: 7 días + 10 días + 30 días = 47 días.

Añadir que la documentación se irá haciendo en paralelo a la vez que se va desarrollando el proyecto.

En la planificación general se añadirá el resto de partes del proyecto que ocuparán también parte del tiempo del desarrollo del proyecto, como es el análisis del proyecto, la búsqueda de herramientas, documentación, búsqueda bibliográfica, etc. Hasta ocupar el total del tiempo estimado del proyecto.

4.3.1 - Metodologías Lean y Scrum. Modelo Lean-Scrum

Puesto que el producto a desarrollar está destinado a un sector en el que no hay un gran uso de las nuevas tecnologías, se seguirá un modelo Lean, donde se irán desarrollando prototipos de la aplicación que a su vez serán validados por un conjunto de usuarios, con el fin de ir mejorando el producto final y haciendo así que este sea de la forma que al usuario le sea más útil y familiar.

Esta metodología de desarrollo hace que se pueda comenzar desde un prototipo inicial básico en el que los requisitos establecidos inicialmente puedan ir cambiando en función de la necesidad del usuario. Debido a ello, se irán desarrollando una serie de prototipos, conocidos en esta metodología como **PMV** (Producto Mínimo Viable). [17]

Para la mejora iterativa del producto, será necesario que los usuarios a los que está dirigida la aplicación final estén involucrados en el desarrollo de la misma, con el fin de estudiar sus hipótesis en base al producto desarrollado, puesto que su valoración, opiniones y sugerencias son la base para que el producto final sea lo que el usuario quiera, necesite o esté dispuesto a pagar por él.

En la metodología **Lean** se aceptan que los requisitos cambien durante la vida del proyecto, incluso en las etapas tardías del proyecto. Se tiene en cuenta la mejora continua del software y los cambios en las especificaciones indicados por el cliente que van siendo desarrollados en breves plazos, normalmente inferiores a dos semanas.

Durante el desarrollo del proyecto, tanto los responsables del negocio como los desarrolladores trabajan de forma conjunta.

Junto a la metodología Lean, será utilizada la metodología ágil para procesos de Tecnologías de la Información denominada **Scrum.** [18]

Scrum favorece al desarrollo de proyectos caracterizados por su alta complejidad, haciéndolos que se puedan satisfacer las necesidades de una manera más asequible gracias a la planificación y a la asignación de tareas.

Cuando se cuenta con un equipo de desarrolladores favorece la realización de tareas de una manera simultánea. Entre los diferentes miembros del equipo que forman la metología Scrum encontramos:

1. Product Owner: miembro del equipo que está en contacto con el cliente, más concretamente es quien representa al cliente cuando se debe decidir los requisitos del proyecto.
2. Scrum Master: su papel es fundamental dentro del equipo, entre sus tareas está el conseguir una buena coordinación dentro del equipo, garantizando a su vez el cumplimiento de los objetivos.
3. Equipo Scrum: resto de los miembros del equipo que se encargan del dearrollo del proyecto

En mi caso al ser yo la única persona que forma parte del equipo no se va a favorecer la realización simultánea del proyecto, pero si una mejor estructuración y planificación. La persona que va a tener los tres papeles esenciales dentro de la metodología voy a ser yo, realizando las partes estipuladas.

Las partes de la metodología que he hecho referencia anteriormente pero que no he explicado detenidamente son **Product Backlog**, **Sprint Backlog** y **Sprint**.

Para el **Product Backlog** es tarea del Product Owner quien a través de la experiencia, encuestas y entrevistas extraje las historias de usuario que se requieren.

Como pase seguido se establece el **Sprint Backlog** donde se especifican las diferentes tareas a realizar y que miembros del equipo se encargará de cada una, en mi caso, yo soy quien debo realizar todas de manera completa.

Por último, se realiza el **Sprint**, periodo de tiempo que abarca entre una y cinco semanas aproximadamente, donde a través de entregas parciales comprobaremos la realización de las tareas declaradas en el Sprint Backlog.

Para comprobar lo anterior se realizan una serie de reuniones:

- Reunión Sprint: Se definen el tiempo de las distintas etapas del Sprint, la reunión se lleva a cabo con la presencia del Scrum Master y Product Owner.
- Daily Scrum: también llamada reunión diaria, donde al comienzo del día se establece los objetivos diarios con el fin de mejorar y conseguir la mejor productividad posible
- Retrospectiva: es la última reunión que tiene el proyecto donde se analiza y evalúan tanto los problemas surgidos a lo largo del desarrollo del producto y su resolución de manera eficiente.

4.4 - Análisis

4.4.1 - Análisis de requisitos

Tras la realización de una encuesta a un conjunto de 30 usuarios para la validación de requisitos inicialmente establecidos, se han obtenido los siguientes datos:

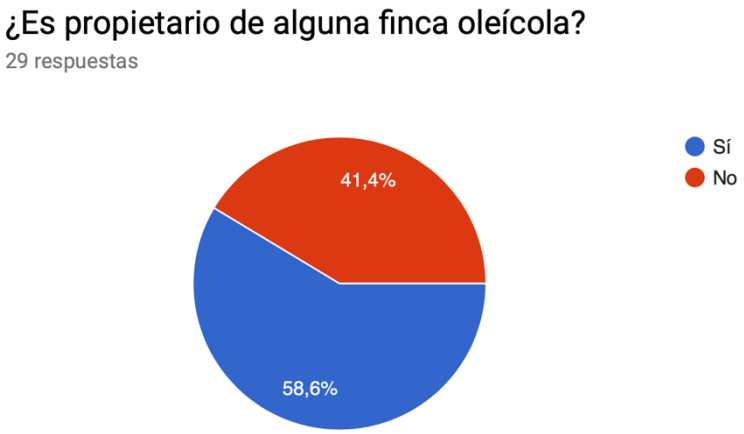

Ilustración 12: Pregunta 1 encuesta análisis de requisitos

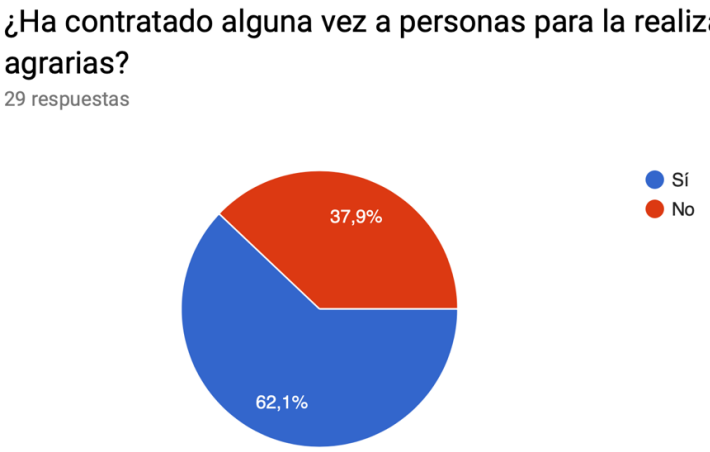

Ilustración 13: Pregunta 2 encuesta análisis de requisitos

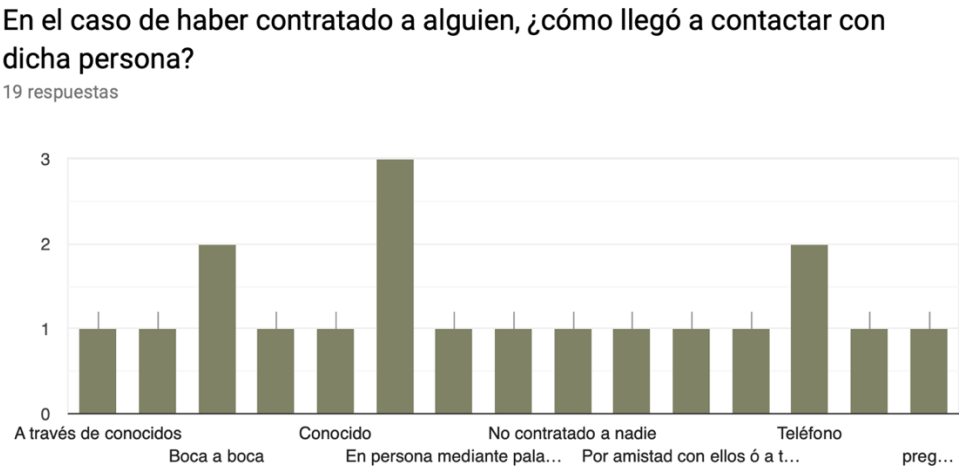

Ilustración 14: Pregunta 3 encuesta análisis de requisitos

Ilustración 15: Pregunta 4 encuesta análisis de requisitos

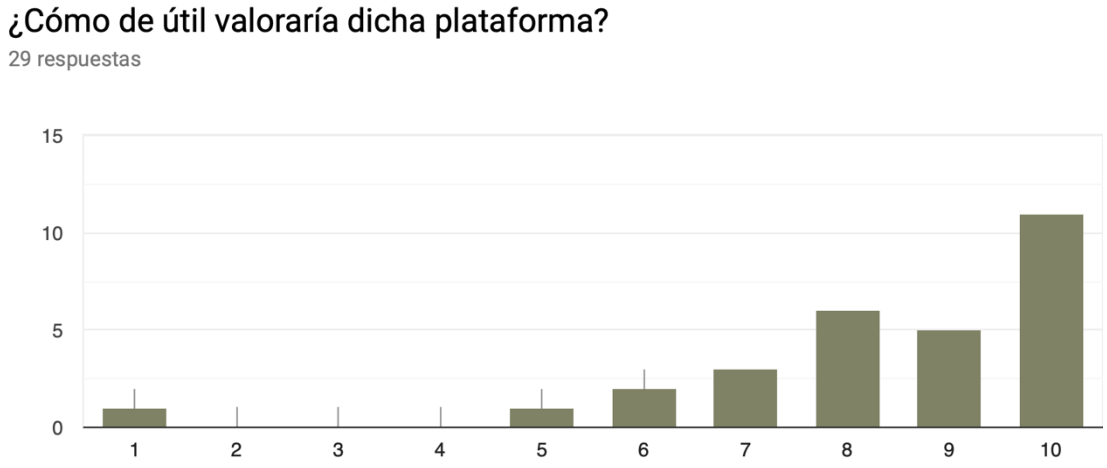

Ilustración 16: Pregunta 5 encuesta análisis de requisitos

En caso de haber valorado con menos de 5 la pregunta anterior, comente por qué:

2 respuestas

N/A
No creo que sea necesario la plataforma

Ilustración 17: Pregunta 6 encuesta análisis de requisitos

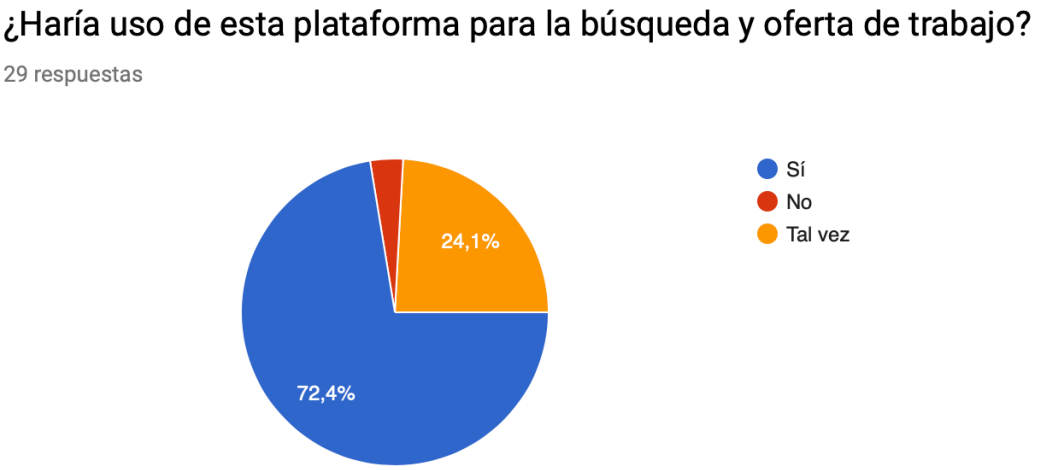

Ilustración 18: Pregunta 7 encuesta análisis de requisitos

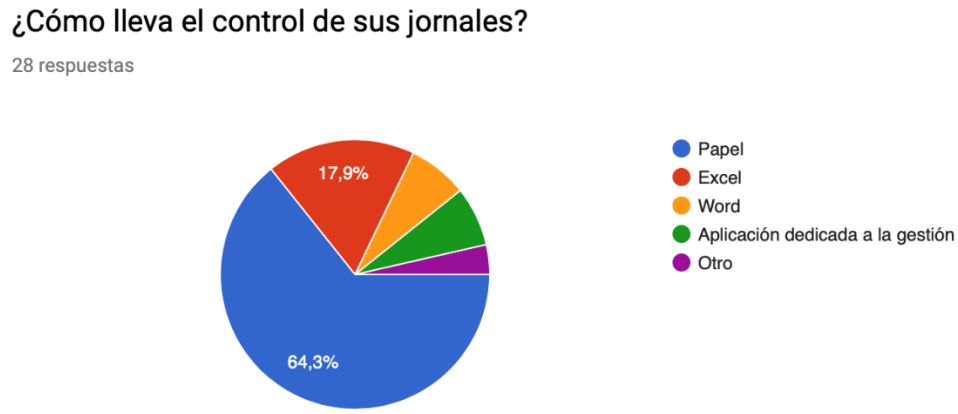

Ilustración 19: Pregunta 8 encuesta análisis de requisitos

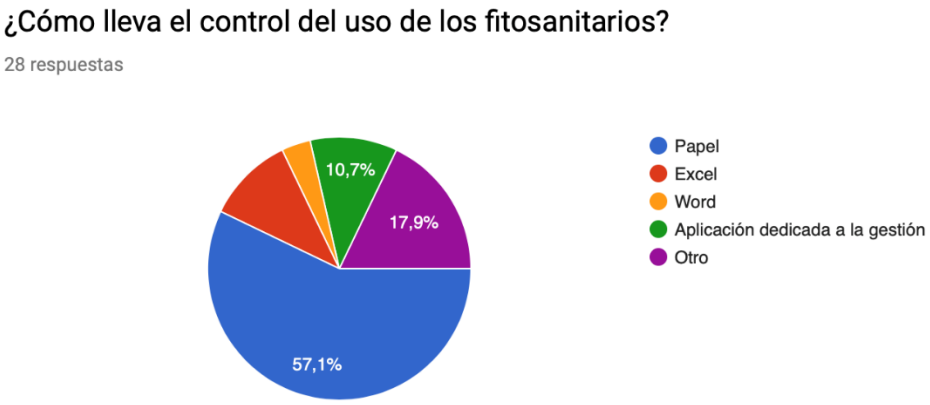

Ilustración 20: Pregunta 9 encuesta análisis de requisitos

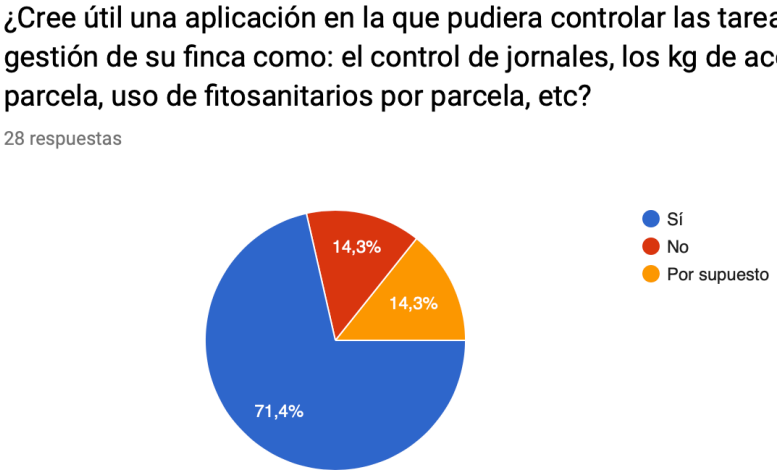

Ilustración 21: Pregunta 10 encuesta análisis de requisitos

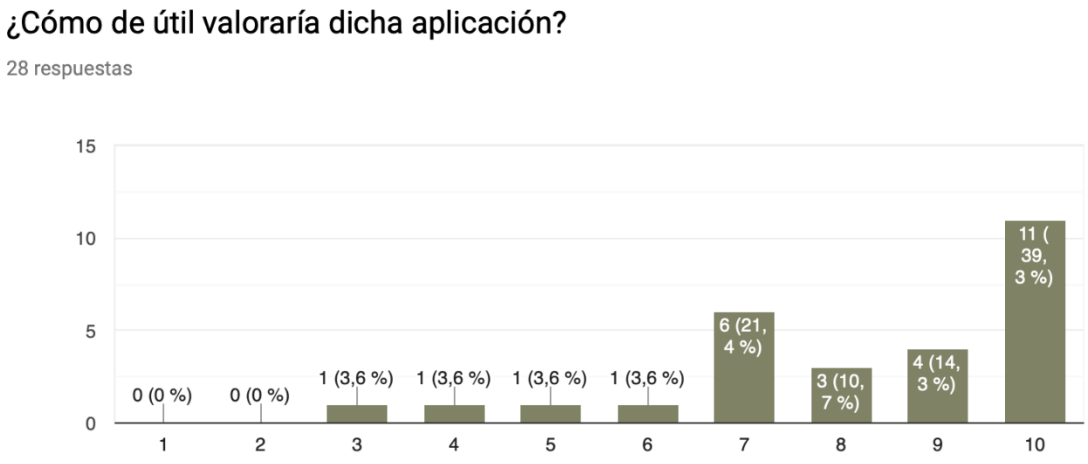

Ilustración 22: Pregunta 11 encuesta análisis de requisitos

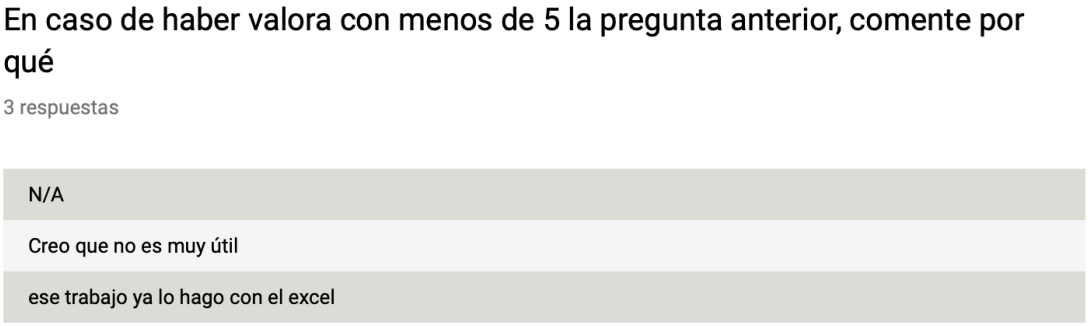

Ilustración 23: Pregunta 12 encuesta análisis de requisitos

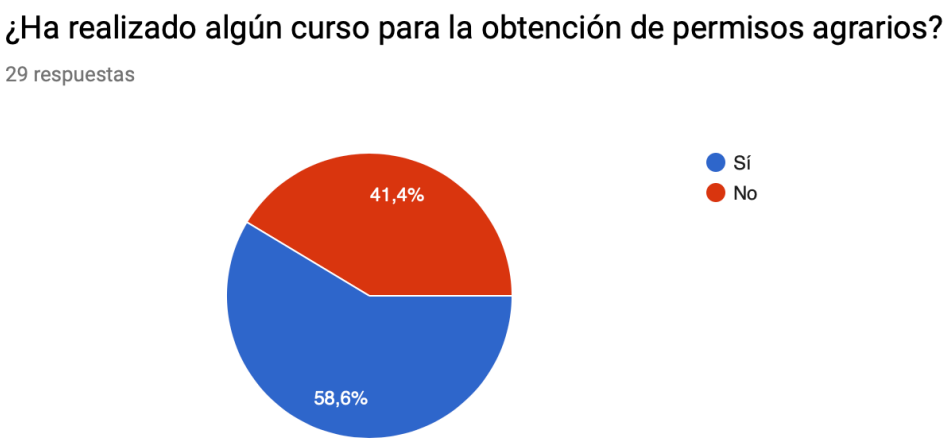

Ilustración 24: Pregunta 13 encuesta análisis de requisitos

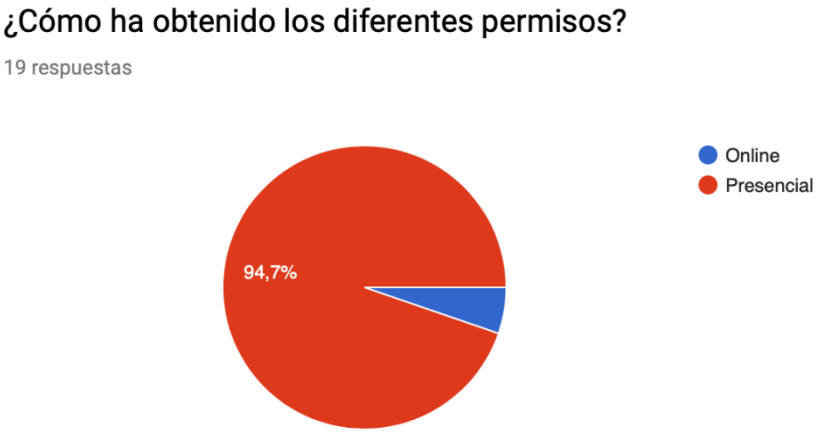

Ilustración 25: Pregunta 14 encuesta análisis de requisitos

¿Cree útil la posibilidad de realizar cursos online para la obtención de los distintos carnets necesarios para las tareas agrarias?

29 respuestas

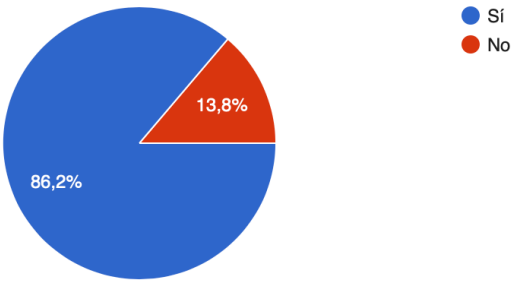

Ilustración 26: Pregunta 15 encuesta análisis de requisitos

En caso de haber votado si en la pregunta anterior, valore su utilidad

25 respuestas

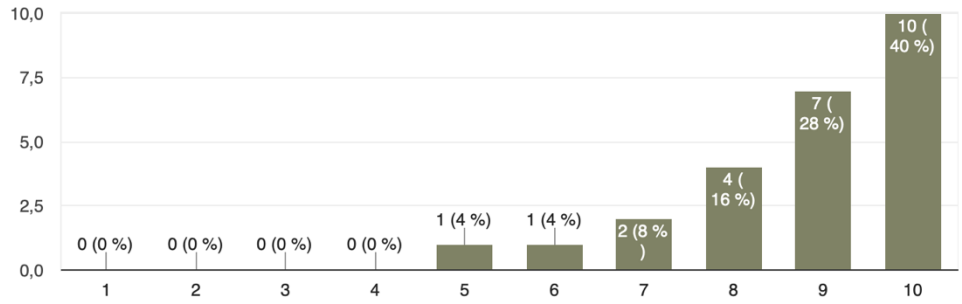

Ilustración 27: Pregunta 16 encuesta análisis de requisitos

En caso de haber valorado con menos de 5 la pregunta anterior, comente por qué:

1 respuesta

N/A

Ilustración 28: Pregunta 17 encuesta análisis de requisitos

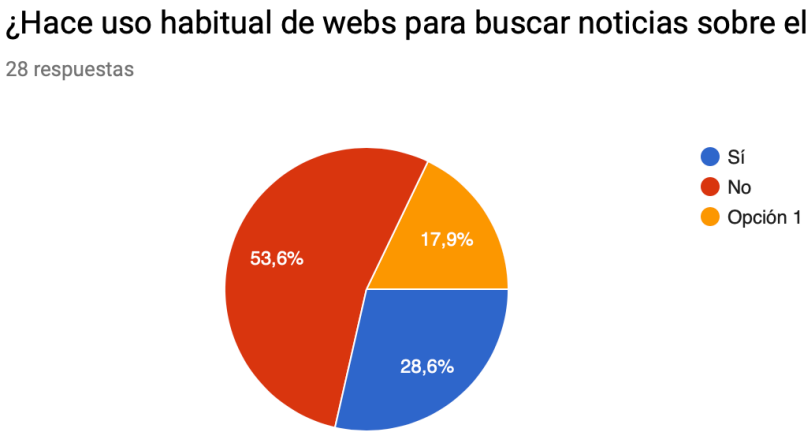

Ilustración 29: Pregunta 18 encuesta análisis de requisitos

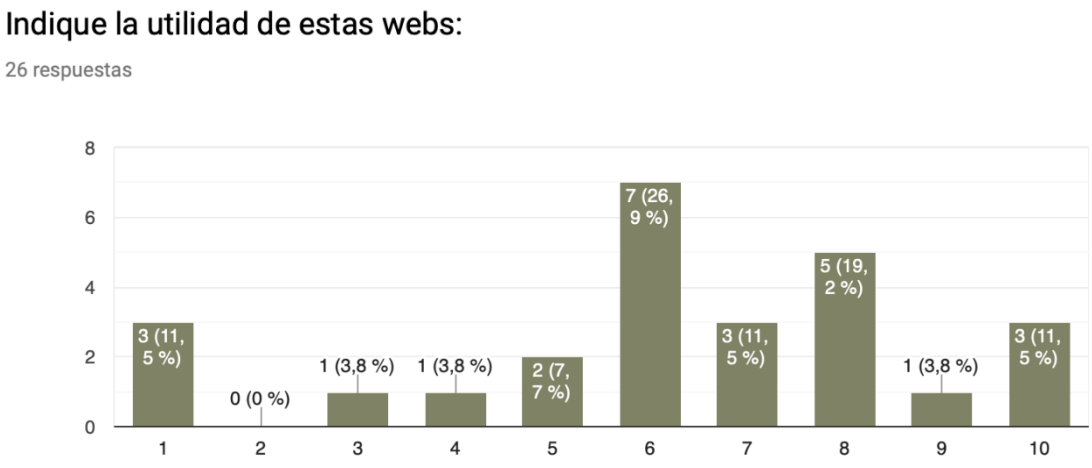

Ilustración 30: Pregunta 19 encuesta análisis de requisitos

¿Le gustaría que fuese notificado a su móvil de todas las noticias relacionadas con el sector?

29 respuestas

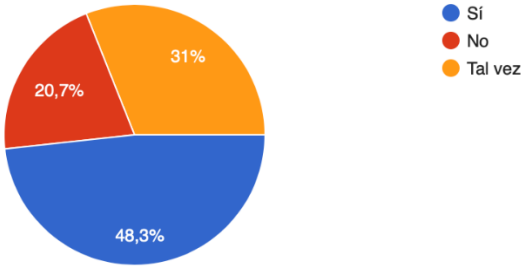

Ilustración 31: Pregunta 20 encuesta análisis de requisitos

Una vez propuestas esta serie de ideas, ¿le gustaría tener una web con todo el contenido comentado?

30 respuestas

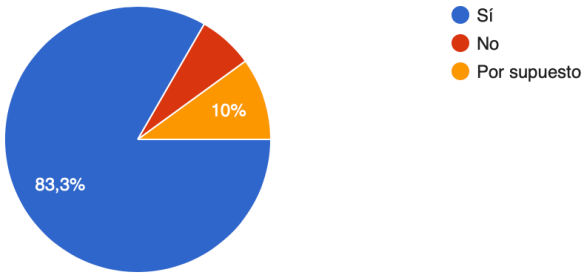

Ilustración 32: Pregunta 21 encuesta análisis de requisitos

En el caso de haber contestado "No" en la pergunta anterior, comente por qué:

2 respuestas

N/A
Solo me interesa buscar trabajo relacionado

Ilustración 33: Pregunta 22 encuesta análisis de requisitos

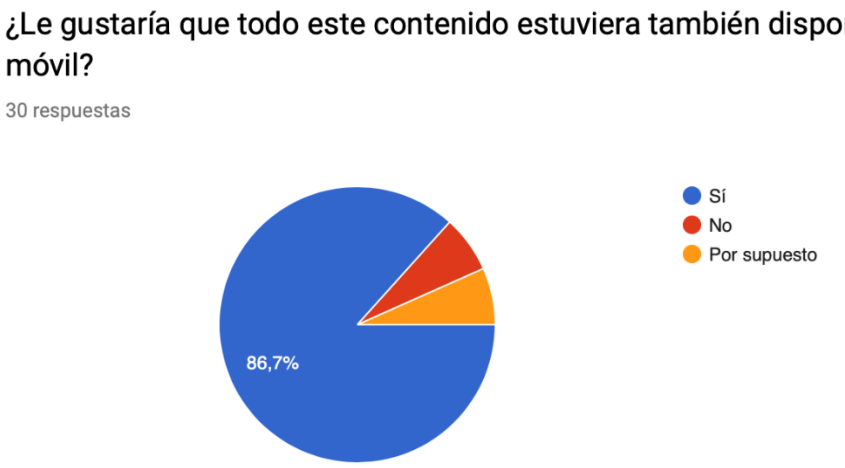

Ilustración 34: Pregunta 23 encuesta análisis de requisitos

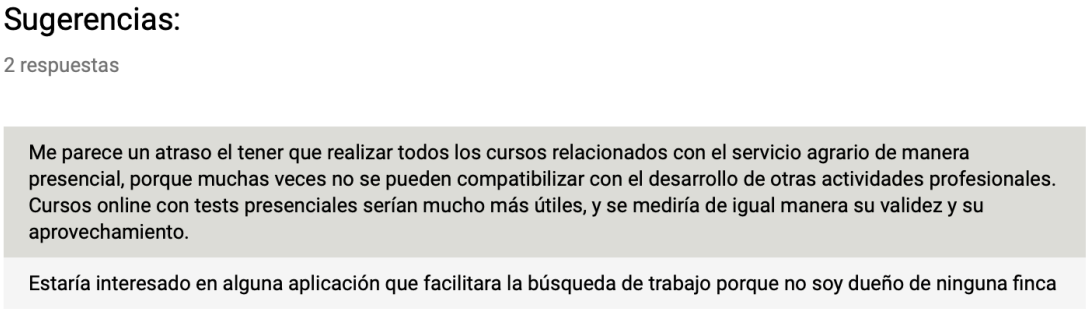

Ilustración 35: Pregunta 24 encuesta análisis de requisitos

Para obtener el punto de vista desde la parte técnica de cada uno de los requisitos detallados inicialmente, se ha realizado un análisis de requisitos en base a una conversación con María del Carmen Calatrava Moreno, doctora en informática y consultora en Technopolis Group.

La conversación tuvo lugar el día 9 de marzo de 2019 a través de medios de comunicación online.

Durante la conversación se han expuesto los requisitos detallados en la versión 0 para su análisis.

La valoración de cada uno de ellos por parte de María del Carmen (MC) ha sido:

Requisitos funcionales

Requisito funcional 1: Portal de empleo.

El sistema deberá ofrecer un portal de empleo en el que los **usuarios registrados** podrán añadir ofertas de trabajo.

MC: *"Habría que hacer un análisis de la necesidad que tienen los usuarios finales (tanto trabajadores, como empresarios) de dicha funcionalidad. Puede ser que ya tengan mecanismos efectivos para esta finalidad (ej. el boca a boca, la oficina de empleo, etc.). También hay que considerar que algunos trabajadores pueden ser inmigrantes en situación irregular, y puede que no quieran o deban tener un perfil público en una aplicación."*

Durante la conversación también se consideró el hecho de que puede que haya personas que no estén familiarizadas con las TIC.

Debido a la valoración de este requisito por parte de María del Carmen, en este punto, se ha pensado en colaborar con telecentros como Guadalinfo, ya que, estos centros, son de acceso gratuito y están situados en localidades de menos de 20.000 habitantes, en Entidades Locales Autónomas y en las barriadas urbanas menos favorecidas. Desde allí, todas las personas que no

tengan los recursos TIC necesarios para acceder a la búsqueda de trabajo podrán ser informados del proceso necesario para acceder a la aplicación y buscar empleo.

También se ha pensado en colaborar con asociaciones agrarias como UPA, COAG, ASAJA, etc. Para que puedan dar a conocer la aplicación entre los usuarios potenciales.

Requisito funcional 2: Página de noticias.

El sistema deberá ofrecer un apartado exclusivo para noticias relacionadas con el sector.

MC: *"Quizás sea un requisito que deberías de volver a estudiar, ya que hay muchas páginas y portales destinados a dar toda la información relacionada con el mundo del olivar. Sería interesante estudiar cómo tu página de noticias se puede diferenciar del resto."*

Debido a la valoración de este requisito por parte de María del Carmen, se ha optado por hacer un estudio sobre las páginas existentes y lanzar una encuesta para valorar los nuevos requisitos que se obtendrán del estudio previamente comentado.

Requisito funcional 3: Plataforma e-learning.

El sistema ofrecerá al usuario un catálogo de cursos disponibles a los que poder matricularse.

MC: *"Referido a este punto, creo que es el que veo menos viable para su desarrollo. Creo que los trabajadores agrícolas no son grandes consumidores de cursos online. Es cierto que hay cursos que deben realizar para obtener cierta acreditación (ej. aplicador de productos fitosanitarios) o para obtener subvenciones (ej. cursos de iniciación a la agricultura para jóvenes agricultores). Sin embargo, esos cursos están organizados por gobiernos regionales. Por ejemplo, la Junta de Andalucía organiza esos cursos ella misma*

sin delegar a empresas externas. Los organiza a través del IFAPA y tienen su propia plataforma de e-learning - de hecho, acaban de actualizarla."

Debido a la valoración de este requisito por parte de María del Carmen, se ha optado por suprimir el desarrollo del mismo, ya que se profundizará en un desarrollo más exhaustivo del resto de requisitos. Por lo tanto, sólo se incluirá un apartado con menciones a páginas ya existentes.

Requisito funcional 4: Sistema de gestión de fincas.

El sistema ofrecerá al usuario un sistema de gestión de fincas personal que será de acceso privado.

MC: *"Este requisito me parece interesante. Actualmente no hay muchas aplicaciones destinadas a ello, y aun existiendo algunas, seguro que podrás mejorar muchas de sus funcionalidades, o incluso, personalizarlas a gusto del usuario. [..] Como observación, creo que deberías de hacer un estudio previo de las ya existentes y ver qué ofrecen. Una vez estudiadas cada una de las aplicaciones y sus funcionalidades, decide qué puedes hacer nuevo, mejorar o personalizar."*

Debido a la valoración de este requisito por parte de María del Carmen, se ha decidido seguir con el desarrollo de este requisito.

Requisitos no funcionales

Requisito no funcional 1: Interfaz de usuario y diseño responsive.

a) El sistema deberá ofrecer al usuario un diseño intuitivo en el que cada usuario que acceda sea capaz de comprender su total funcionamiento.

MC: "*No sólo eso, sino que además debe de requerir pocos golpes de ratón (o de dedo) para introducir información. Por ejemplo, un trabajador utilizando la herramienta de gestión se puede cansar de introducir información si eso le lleva demasiado tiempo."*

b) El sistema deberá ofrecer al usuario un sistema adaptable a cualquier dispositivo, ya sean smartphones, tablets, laptops o desktops.

MC: *"Totalmente, además estaría bien que el sistema pudiera estar operativo incluso sin conexión a internet (en la medida de lo posible), ya que hay zonas del campo con mala cobertura."*

c) El sistema deberá ofrecer en cada una de las páginas que visiten los usuarios no registrados espacios reservados a posibles banners publicitarios.

MC: *"Dependiendo de cuál sea tu modelo de negocio. Hacer negocio con la publicidad es complicado porque prácticamente todas las otras páginas ya incluyen publicidad, pero si encuentras un modelo de negocio publicitario que funcione, deberías de aprovecharlo."*

Versión 1.0. *Validación general del proyecto.*

Requisitos funcionales

Requisito 1: Registro de usuarios.

El sistema deberá ofrecer la opción de registro de usuario. La verificación de registro se realizará a través de un correo de confirmación. Una vez registrado, el usuario podrá acceder a los recursos adicionales disponibles para este tipo de usuarios.

El usuario no registrado solo podrá acceder a los recursos básicos del sistema:

1. Portal de empleo.
2. Noticias.
3. Enlaces a cursos online.

Subrequisitos:

- Registro
- Inicio de sesión
- Editar cuenta
- Ver cuenta
- Cerrar sesión
- Eliminar usuario

Requisito 2: Portal de empleo.

El sistema deberá ofrecer un portal de empleo en el que los **usuarios registrados** podrán añadir ofertas de trabajo en las que se deberá incluir:

4. Tipo de trabajo.
5. Descripción.
6. Teléfono de contacto.
7. Localización.

Una vez añadida la oferta de trabajo, el sistema la catalogará por tipo de trabajo, localización y fecha de inclusión en el sistema.

El sistema deberá ofrecer la opción de eliminar la oferta al **usuario registrado** que la haya publicado.

El sistema ofrecerá filtros para la búsqueda de ofertas de empleo para los usuarios (**registrados o no registrados**) que deseen catalogarlas por *tipo de trabajo, localización y fecha de inclusión en el sistema.*

Subrequisitos:

- Añadir empleo
- Modificar empleo
- Visualizar empleo
- Eliminar empleo
- Filtrar empleo

Requisito 3: Página de noticias.

El sistema deberá ofrecer un apartado exclusivo para noticias relacionadas con el sector.

A este apartado podrán acceder tanto **usuarios registrados** como **no registrados**.

Requisito 4: Plataforma e-learning.

El sistema ofrecerá al usuario tanto **registrado** como **no registrado** un catálogo de enlaces a cursos disponibles.

Requisito 5: Sistema de gestión de fincas.

El sistema deberá de ofrecer al **usuario registrado** la opción de activar la gestión de fincas.

El sistema deberá de ofrecer al **usuario registrado** la opción de añadir fincas.

El sistema deberá de ofrecer al **usuario registrado** la opción de añadir fincas a través del sistema SIGPAC.

El sistema deberá ofrecer al **usuario registrado** la opción de añadir gastos a cada finca.

El sistema deberá ofrecer al **usuario registrado** la opción de añadir trabajadores a cada finca del sistema.

El sistema deberá ofrecer al **usuario registrado** la opción de eliminar trabajadores a cada finca del sistema.

El sistema deberá ofrecer al usuario un resumen detallado de cada finca atendiendo a la información asignada a cada una de ellas.

Requisitos no funcionales

Requisito 1: Interfaz de usuario y diseño responsive.

El sistema deberá ofrecer al usuario un diseño intuitivo en el que cada usuario que acceda sea capaz de comprender su total funcionamiento.

El sistema deberá ofrecer al usuario un sistema adaptable a cualquier dispositivo, ya sean smartphones, tablets, laptops o desktops.

El sistema deberá ofrecer en cada una de las páginas que visiten los usuarios no registrados espacios reservados a posibles banners publicitarios.

4.4.2 - Revisión de requisitos

Debido a la metodología que se está utilizará, la metodología Lean, se han tomado una serie de decisiones para desarrollar el siguiente producto mínimo viable. Con ello, se busca que éste sea un producto de valor para el usuario, es decir, que el producto desarrollado cumpla con las expectativas del usuario de la forma más rápida posible.

Tras los datos obtenidos de las encuestas realizadas, así como las conclusiones obtenidas tras la entrevista realizada a María del Carmen Calatrava, se ha decidido realizar el desarrollo **del Requisito funcional 2: Portal de empleo.**

Se ha tomado esta decisión debido a que en las encuestas ha sido el punto de los requisitos expuestos en las encuestas qué más útil ven los usuarios. El **79,3%** de los usuarios ha contestado que si cree interesante una plataforma que ofrezca estos servicios.

Por otro lado, para el desarrollo de este primer Producto Mínimo Viable, también se desarrollará el **Requisito funcional 1: Registro de usuarios** y el **Requisito no funcional 1: Interfaz de usuario y diseño responsive.**

Se ha decidido desarrollar **Requisito funcional 1**, debido a que aporta funcionalidad básica para el funcionamiento del **Requisito funcional 2: Portal de empleo.**

En cuanto al **Requisito no funcional 1: Interfaz de usuario y diseño responsive,** será desarrollado para el primer producto mínimo viable puesto que para los usuarios que probarán este PMV necesitan del diseño intuitivo y de su correcta funcionalidad en versión mobile.

Se ha tomado esta serie de decisiones debido a que, tras los resultados obtenidos en las encuestas, habría que investigar más a fondo las necesidades del usuario. Por ejemplo, para el **Requisito funcional 5: Sistema de gestión de fincas,** el 71,4% de los usuarios encuestados han contestado que creen útil esta parte.

Debido a la complejidad de desarrollo de este requisito, sería interesante que al menos un 85% de los usuarios encuestados estuviera interesado en los servicios de este requisito, puesto que es un requisito que necesita muchas horas de desarrollo y debe de ser bien analizado y validado por los usuarios.

Los requisitos obtenidos para el desarrollo serán los siguientes:

Requisito	Sub requisito
Registro de usuario	Registro
	Inicio de sesión
	Editar cuenta

	Cerrar sesión
	Ver cuenta
	Eliminar usuario
Portal de empleo	Añadir empleo
	Modificar
	Visualizar
	Eliminar empleo
	Filtrar empleo
Diseño	Desktop
	Responsive

Tabla 2: Requisitos para versión 1

4.4.3 - Planificación para el desarrollo de la versión 1.0

En nuestra planificación se extrajeron una serie de requisitos fundamentales para el desarrollo de nuestro proyecto, ahora como gestor del proyecto debo estimar los recursos necesarios para la ejecución del proyecto.

Como usamos la metodología ágil de SCRUM debemos definir historias de usuario y estimar las mismas. Las historias de usuario las vamos a obtener en función de los requisitos.

Historias de usuario	Estimación	Prioridad
Registro	8	C
Inicio de sesión	3	C
Editar cuenta	2	S
Cerrar sesión	5	S
Ver cuenta	1	W
Eliminar usuario	2	W
Añadir empleo	8	M
Modificar	5	S
Visualizar	2	C
Eliminar empleo	3	W

Filtrar empleo	3	W
Instalación IDE	5	M
Estudio framework	8	M
Desktop	3	C
Responsive	2	S

Tabla 3: Planificación versión 1

Para la estimación de las historias de usuario se ha tenido en cuenta la serie de Fibonacci.

En total poseemos de **60 puntos de historia**, los cuales se van a desarrollar en **4 Sprints** de **15 puntos de historia**. El proyecto será desarrollado como se comentó con anterioridad, en 2 horas diarias.

Como conclusión obtenemos que por cada sprint emplearemos 30 horas y el reparto de las mismas llevadas a cabo por mí, en rol de Product Owner quedando de la siguiente manera:

Sprint 1	Sprint 2	Sprint 3	Sprint 4
Estudio framework (8)	Añadir empleo (6)	Eliminar empleo (1)	Editar cuenta (2)
Instalación IDE (5)	Modificar (5)	Filtrar empleo (3)	Cerrar sesión (5)
Añadir empleo (2)	Visualizar (2)	Registro usuario (8)	Ver cuenta (1)
	Eliminar empleo (2)	Inicio de sesión (3)	Eliminar usuario (2)
			Desktop (3)
			Responsive (2)

Tabla 4: Sprints versión 1

4.4.4 - Análisis del sistema

En esta sección se desarrollará la fase de análisis del sistema, fase de análisis de la Ingeniería del Software que hace uso del lenguaje de modelado de software conocido como UML [19] (Unified Modeling Language). Con el uso de UML se ha desarrollado el modelo de la aplicación a través del cual se pueden identificar los distintos componentes del sistema.

Modelo de dominio

En esta parte se han identificado qué clases formarán parte del modelo de la parte del servidor de la aplicación de forma general a través de la representación del diagrama de modelo de dominio.

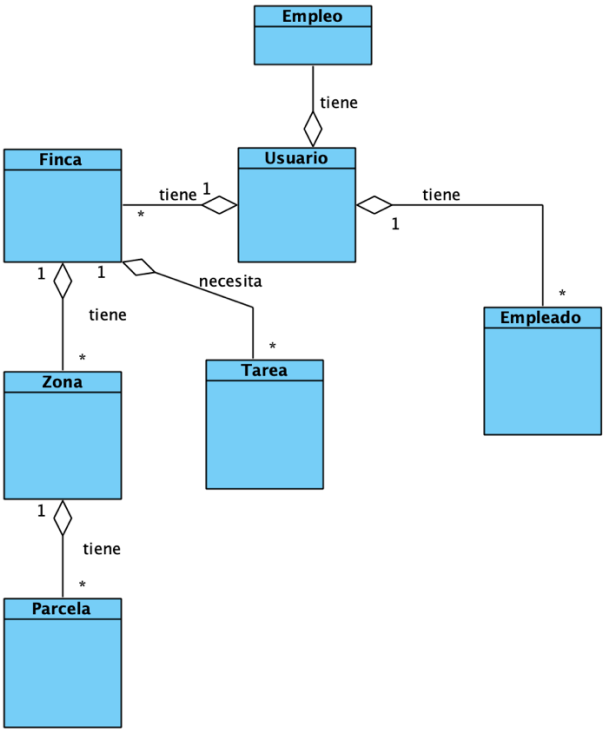

Ilustración 36: Diagrama de modelo de dominio

Las clases representadas en este diagrama representan las clases que conformarán el sistema con su nombre, relaciones entre ellas, la multiplicidad y navegabilidad de las relaciones comentadas.

En la fase de diseño se profundizará más en qué clases se definirán de forma definitiva junto con sus relaciones, multiplicidad y navegabilidad.

La clase Usuario contiene las relaciones con las clases Empleo, Finca y Empleado, ya que éste puede crear empleos en la zona de administración, registrar fincas y registrar empleados. Como se puede observar en el diagrama, el usuario puede tener de uno a muchos empleos registrados, fincas registradas y empleados registrados. En cambio, un empleo, finca o empleado, sólo pueden pertenecer a un usuario específico, como se puede ver en la navegabilidad de las relaciones.

La clase Finca contiene las relaciones de Zona y Tarea, puesto que una finca puede estar compuesta de distintas zonas y pueden registrarse varias tareas en ésta. Como hemos comentado anteriormente, debido a la navegabilidad entre dichas clases, finca puede tener de una a muchas zonas y tareas, pero una zona y una tarea específicas sólo pueden pertenecer a una finca en concreto.

En cuanto a la clase Parcela, está relacionada con la clase Zona debido a que una zona puede estar compuesta por distintas parcelas y a su vez, cada parcela sólo puede estar asignada a una zona en concreto.

Como hemos comentado anteriormente, en la fase de diseño se desarrollarán estas clases, relaciones, navegabilidad y multiplicidad de formas más detallada.

4.5 - Diseño

4.5.1 - Introducción

El diseño de la aplicación será desarrollado en base al patrón Modelo-Vista-Controlador (MVC). Este patrón de arquitectura software está caracterizado por separar la capa de datos, la capa de interfaz de usuario y la capa de modelo de negocio en tres componentes. [20]

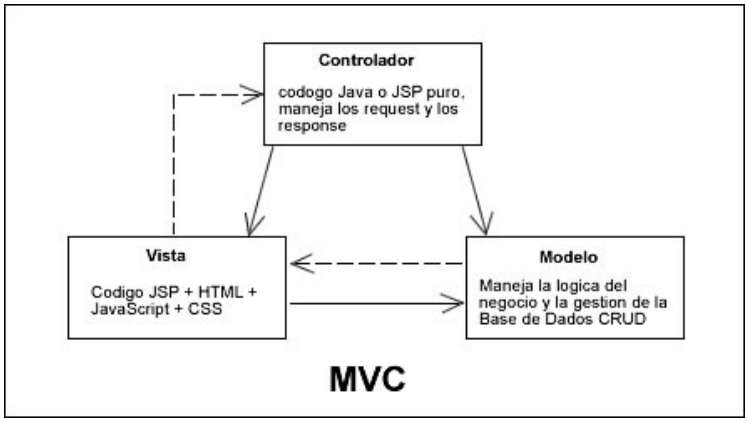

Ilustración 37: Modelo MVC

1. **Modelo**: la capa de modelo de negocio define la lógica de negocio de la aplicación, es decir, se encarga de controlar los mecanismos de acceso a datos y su modificación. Es la capa encargada de trabajar con los datos de forma directa.

2. **Vista**: la capa de la vista se encarga de almacenar el código de la aplicación destinado a la representación visual de los datos, es decir, define las interfaces de usuario.

3. **Controlador**: proporciona acceso a la lógica de negocio de la aplicación. Normalmente, un controlador delega el proceso de lógica de negocio a un conjunto de componentes de servicios. Estos servicios, acceden a base de datos a través de la interfaz DAO (Data Access Object).

   Los controladores reciben parámetros del usuario y son convertidos en un objeto del modelo, poblando en sus atributos los datos enviados como resultado de la lógica de negocio.

   Usamos la anotación @Controller para definir los controladores en Spring.

4.5.2 - Diagramas de clases

En esta sección se especifica el diagrama de clases del sistema de forma más detallada.

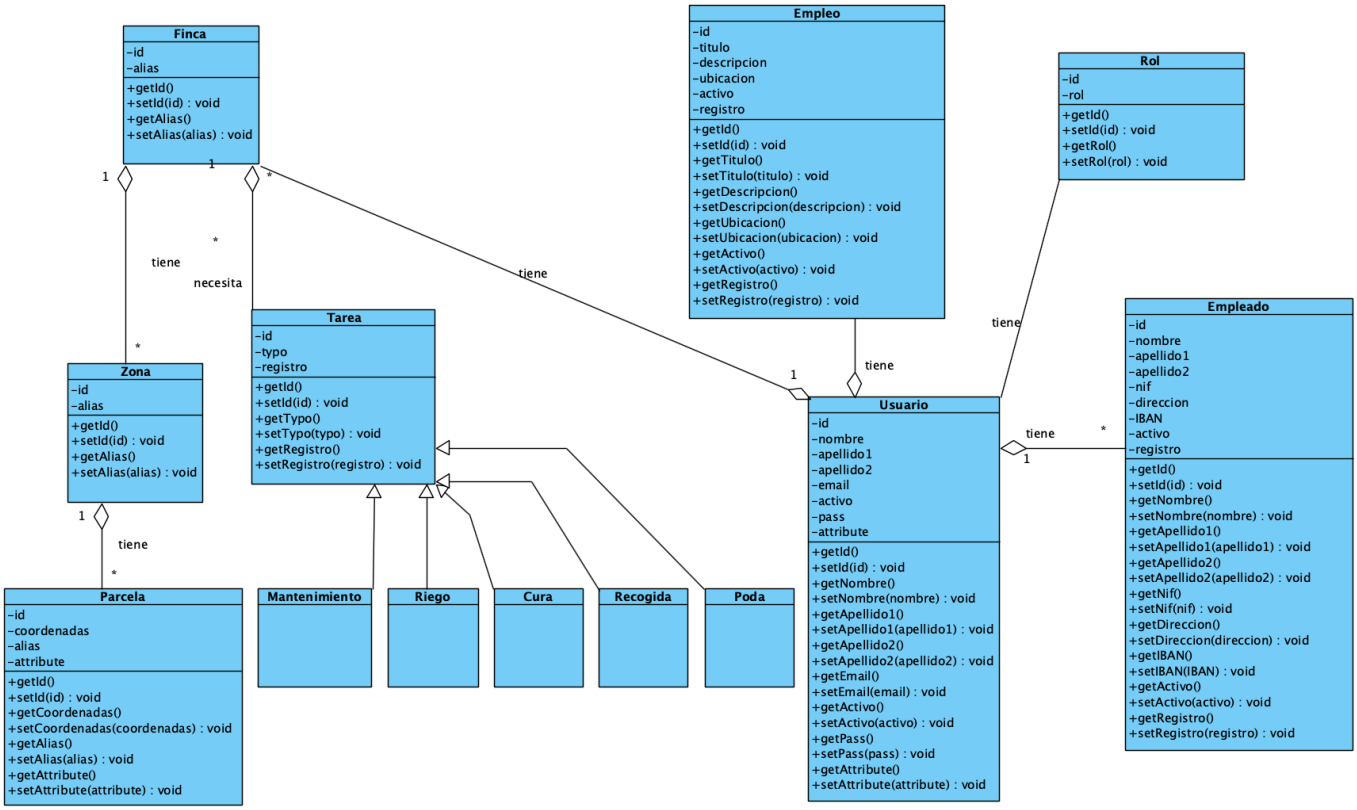

Ilustración 38: Diagrama de clases

Como se ha adelantado anteriormente, en este diagrama se muestran las clases con sus atributos y algunos de sus métodos. También se han añadido clases adicionales que en un análisis previo no se habían tenido en cuenta como necesarias para el sistema pero que, tras un estudio más exhaustivo, se han considerado necesarias para el correcto funcionamiento del sistema.

Como podemos ver, un ejemplo de una clase necesaria para el correcto funcionamiento del sistema es la clase Rol, puesto que para poder acceder a los diferentes servicios del sistema es necesario registrar al usuario con un rol distinto.

4.5.3 - Diagramas de casos de uso

En esta sección se especifican algunos de los distintos casos de uso del sistema. Estos casos de uso tienen la función de reflejar el comportamiento del sistema desde el punto de vista del usuario, reflejando así, las distintas funcionalidades que ofrece el sistema.

Sistema

Ilustración 39: Diagrama de caso de uso Sistema

Buscar empleo

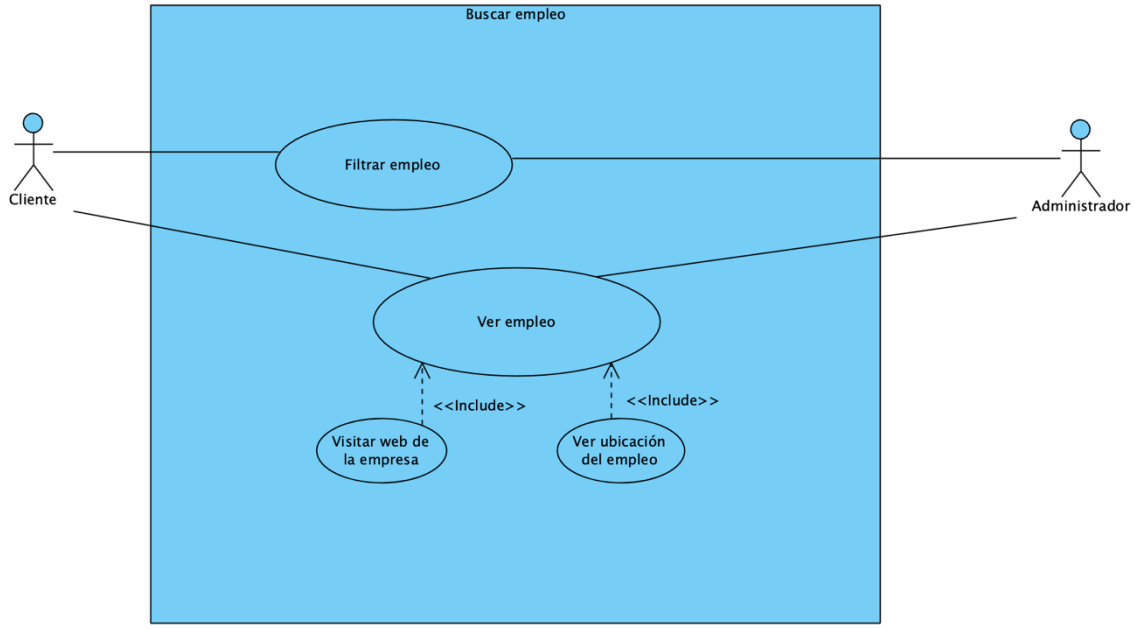

Ilustración 40: Diagrama de caso de uso Buscar empleo

Ver noticias

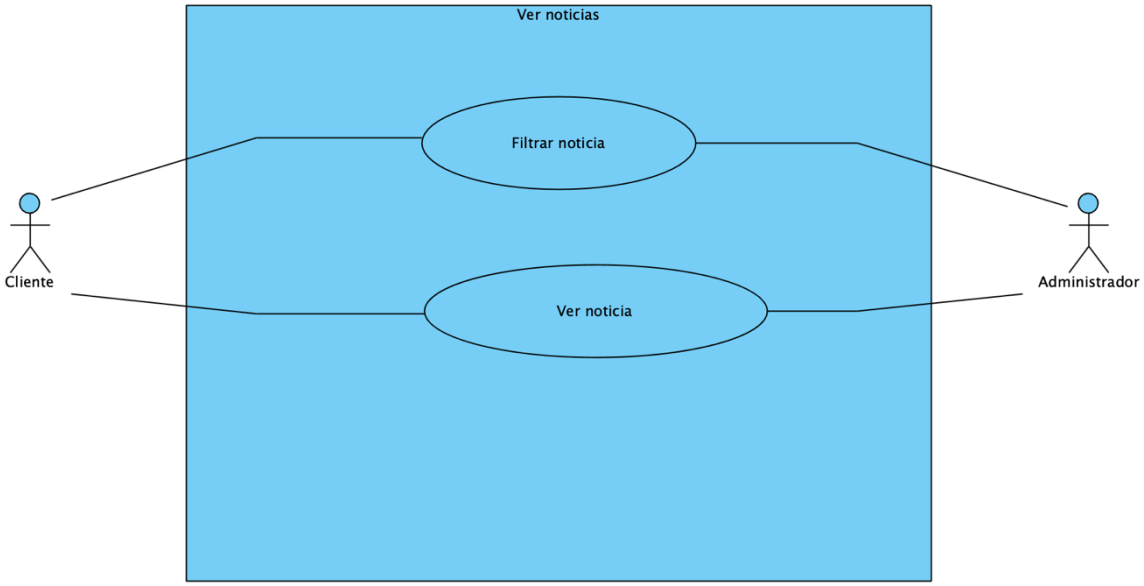

Ilustración 41: Diagrama de caso de uso Ver noticias

Buscar curso

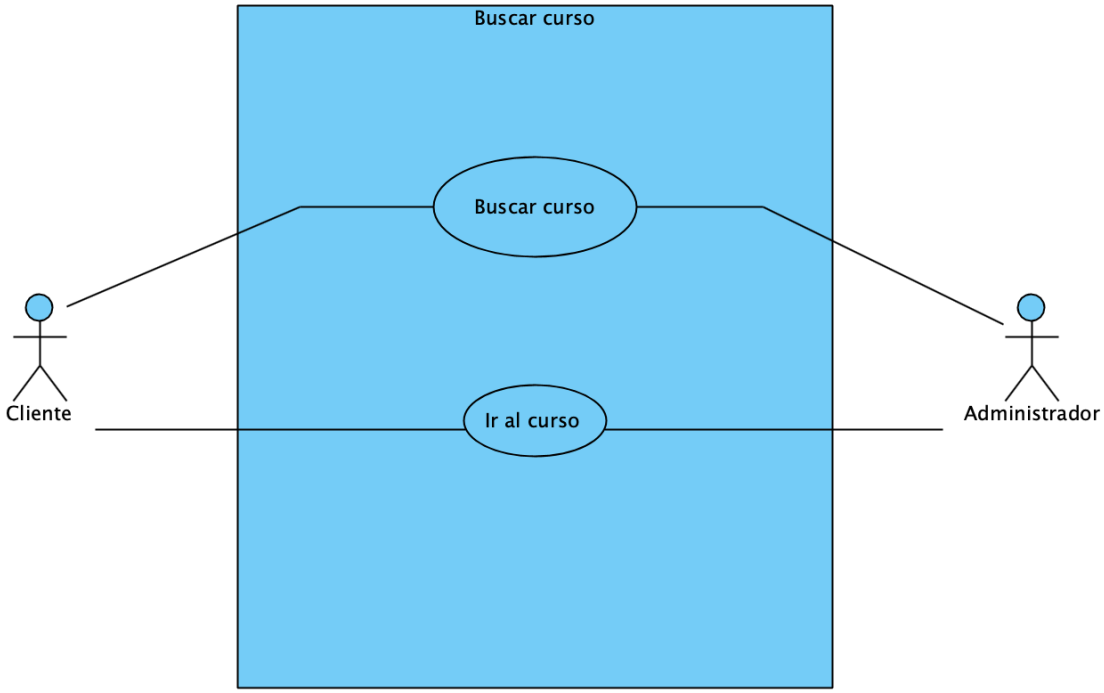

Ilustración 42: Diagrama de caso de uso Buscar curso

Gestionar fincas

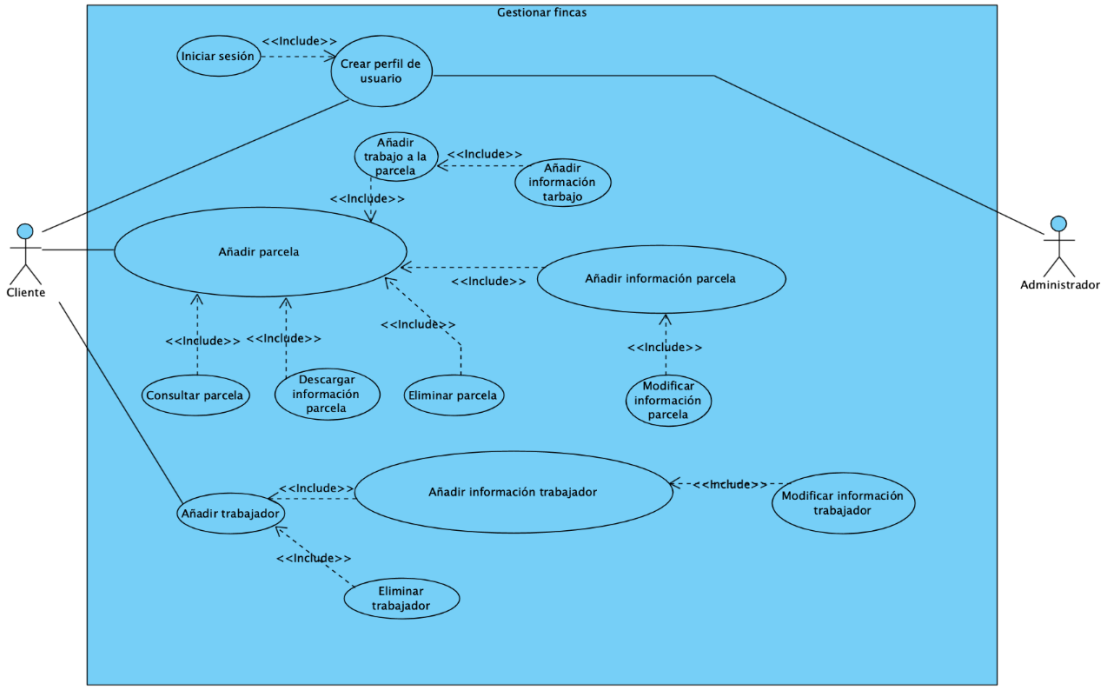

Ilustración 43: Diagrama de caso de uso Gestionar fincas

Administrar oferta de empleo

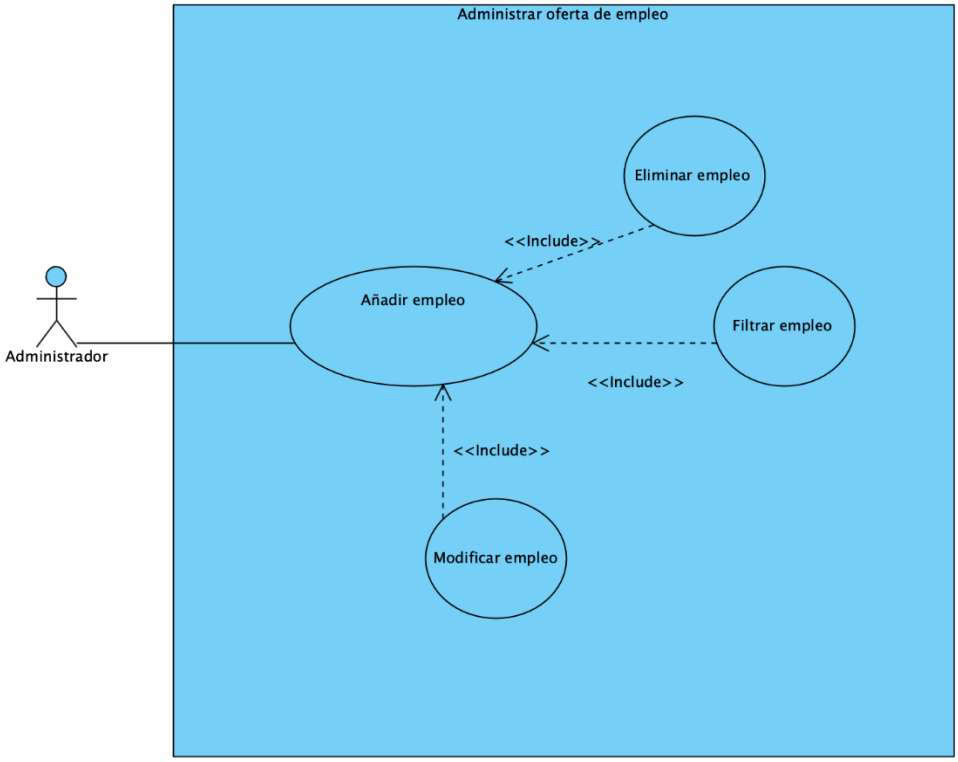

Ilustración 44: Diagrama de caso de uso Administrar oferta de empleo

Administrar noticia

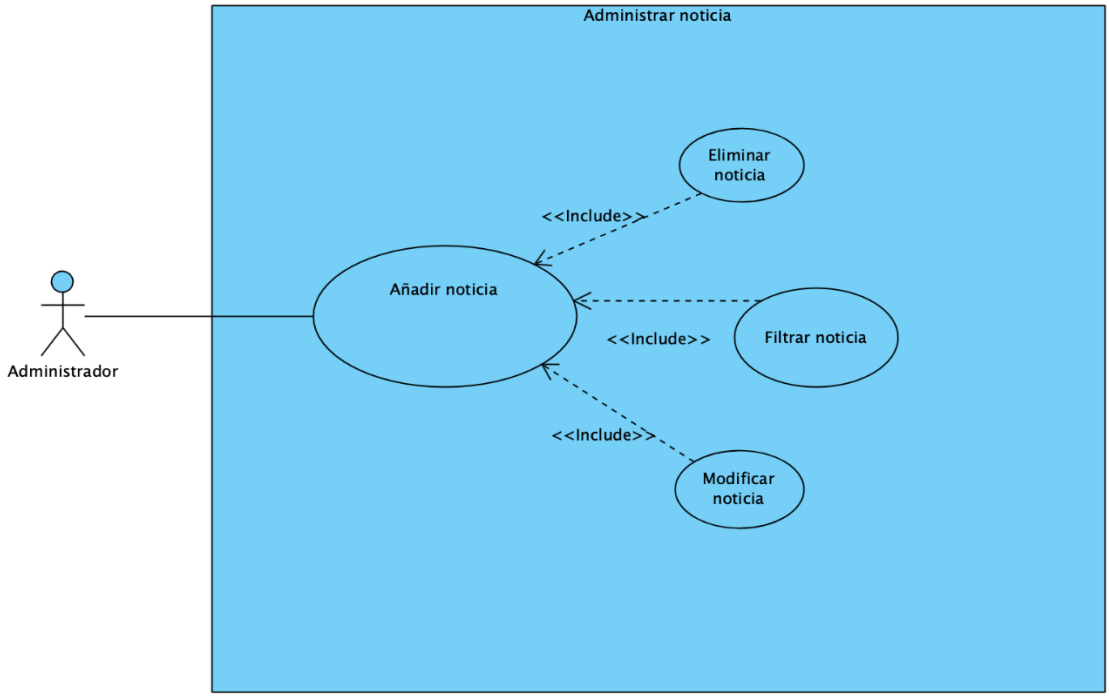

Ilustración 45: Diagrama de caso de uso Administrar noticia

Administrar curso

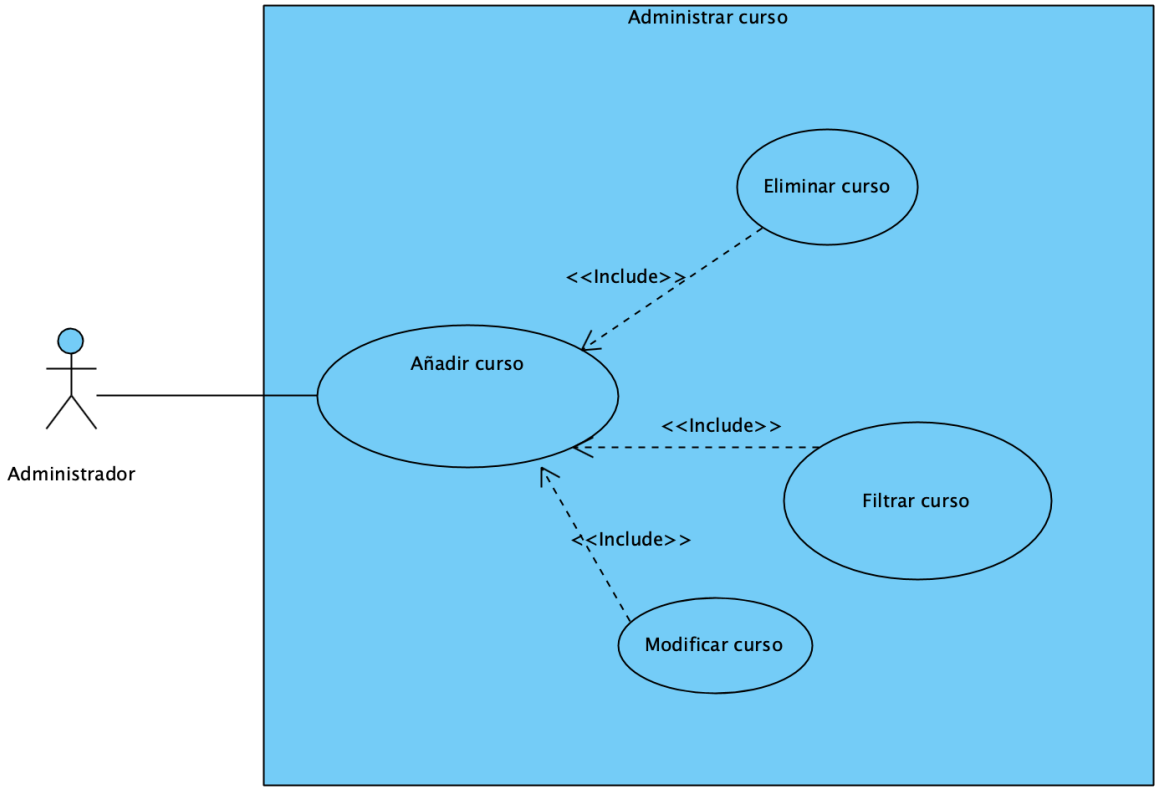

Ilustración 46: Diagrama de caso de uso Administrar curso

4.5.4 - Diagramas de secuencia

En este apartado se representan los diagramas de secuencia del sistema. Para el diseño de éstos ha sido tenido en cuenta el patrón MVC. Estos diagramas muestran la comunicación real entre las tres principales partes del sistema, el modelo, la vista y el controlador.

Indicar que sólo se han desarrollado alguno de lo diagramas con más importancia debido a la cantidad de diagramas existentes en el sistema.

Iniciar sesión

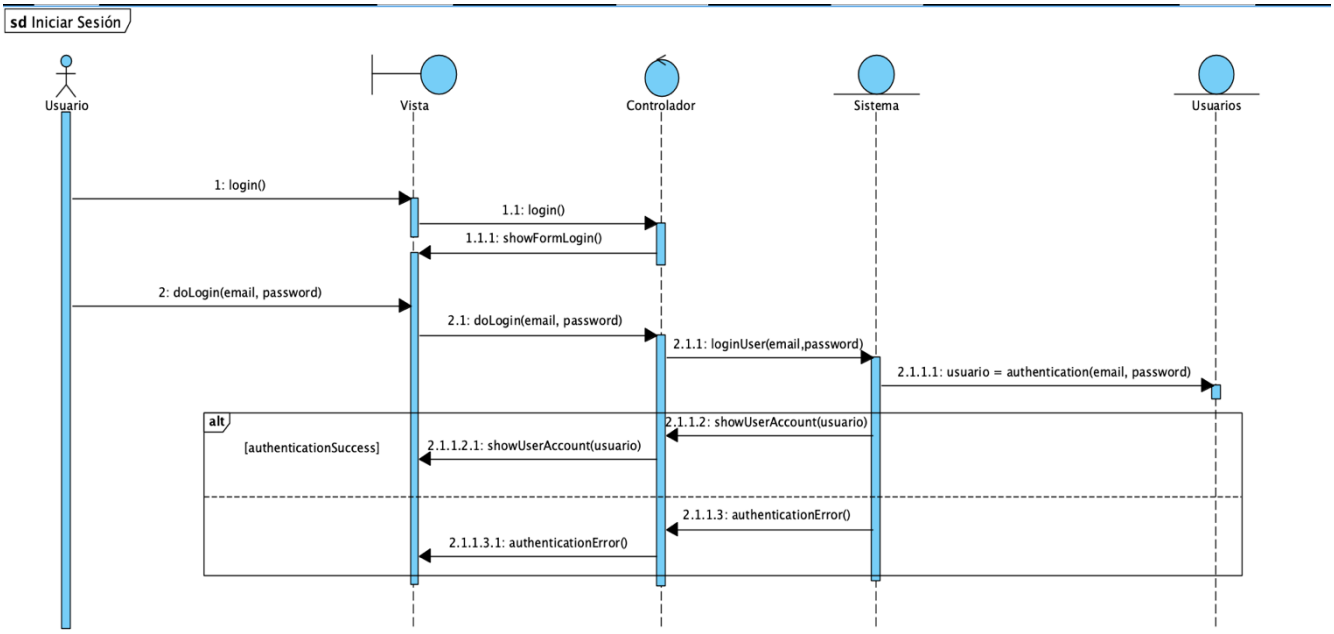

Ilustración 47: Diagrama de secuencia Iniciar sesión

Visualizar pedido

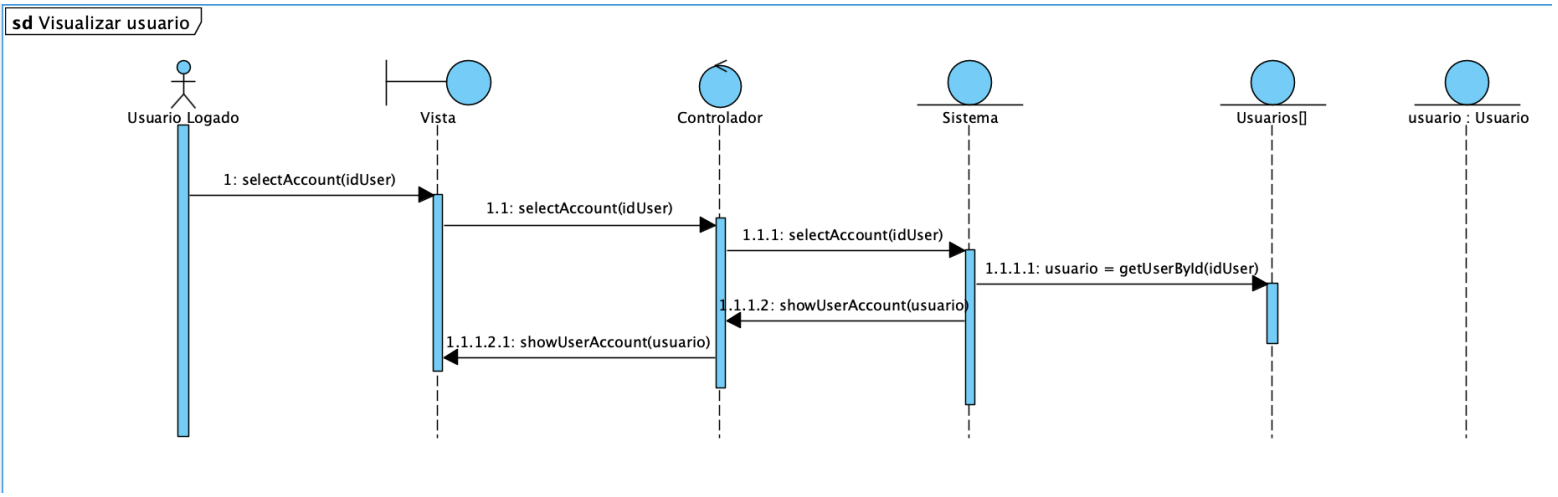

Ilustración 47: Diagrama de secuencia Visualizar pedido

Editar usuario

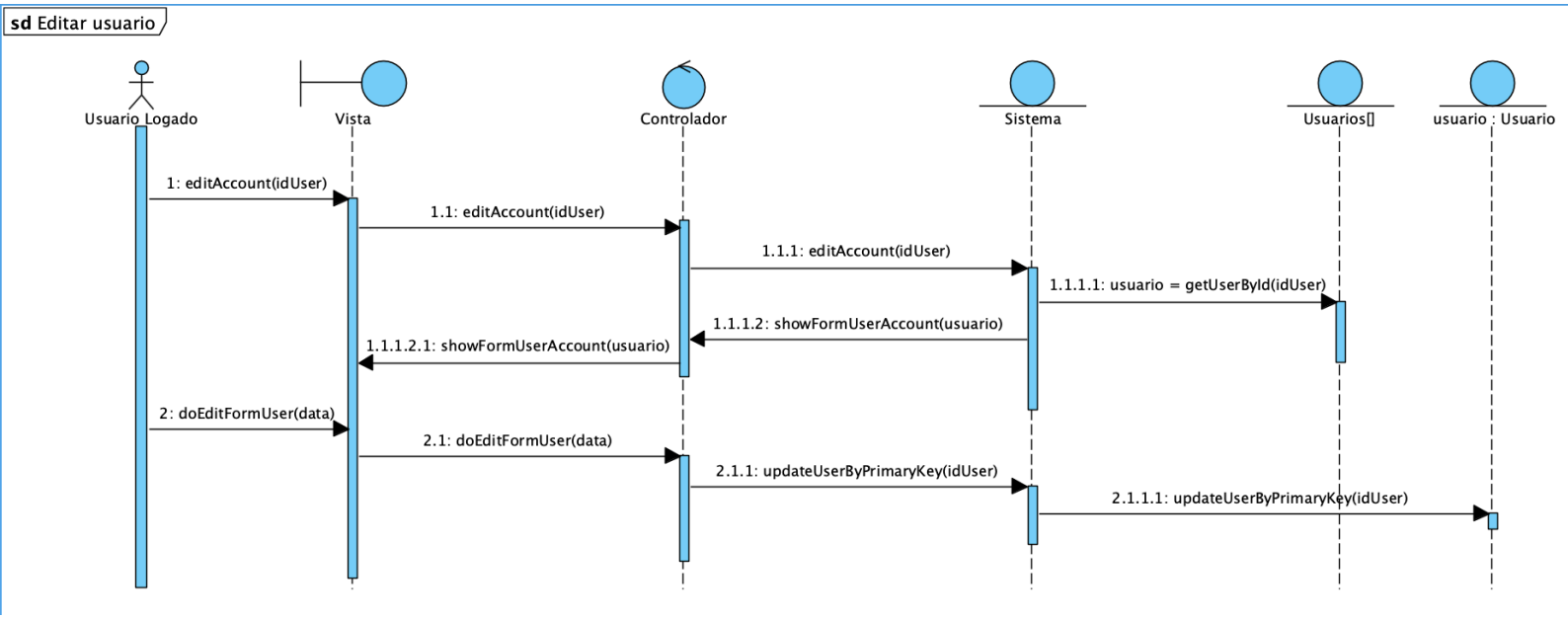

Ilustración 48: Diagrama de secuencia Editar usuario

Añadir empleo

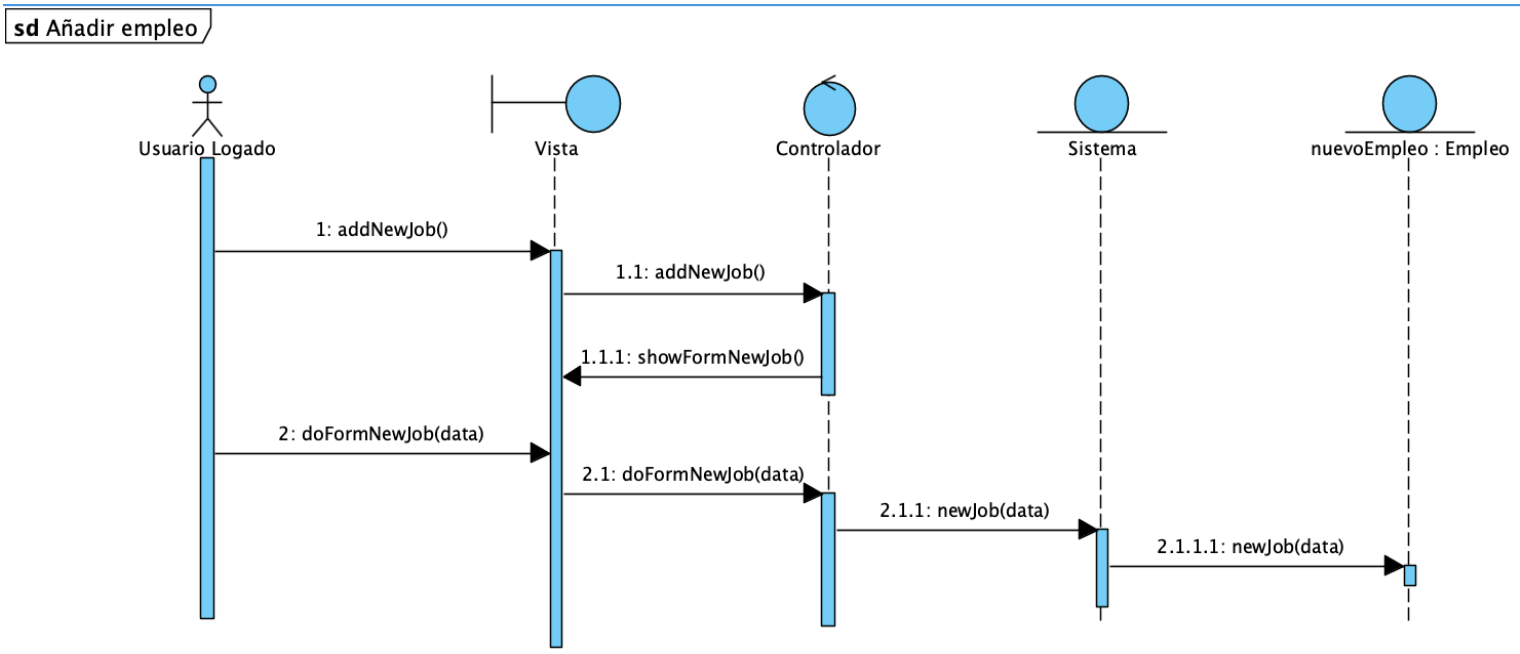

Ilustración 49: Diagrama de secuencia Añadir empleo

4.5.5 - Diseño de interfaz de usuario

Para conseguir una idea previa sobre el diseño de la interfaz de usuario de la aplicación se ha llevado a cabo la realización de wireframes para cada una de las pantallas principales de la aplicación. Se han tenido en cuenta tanto los diseños para la vista de escritorio como para la vista de la versión móvil.

Han sido utilizados dos templates para el desarrollo del diseño de la aplicación. Para el diseño de la parte pública del sistema ha sido elegido el template de bootstrap **Landing Page** [21]. Para la parte privada de la aplicación, es decir, la zona de administración con acceso para los usuarios registrados en el sistema, ha sido seleccionado el template **SB 2 Admin** [22]**,** debido a la variedad de componentes fáciles de integrar y su diseño intuitivo y responsive.

Inicio desktop

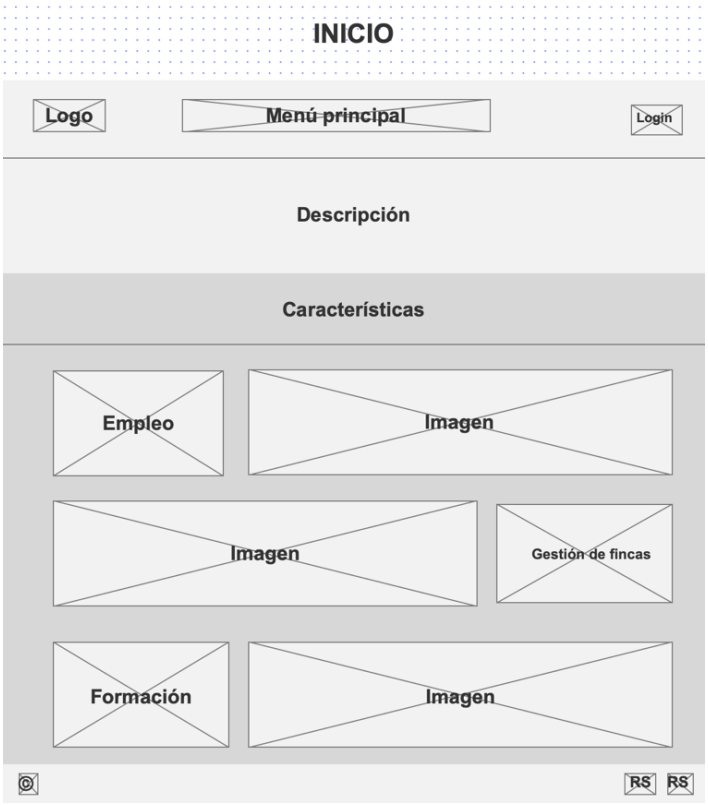

Ilustración 50: Wireframe inicio desktop

Inicio mobile

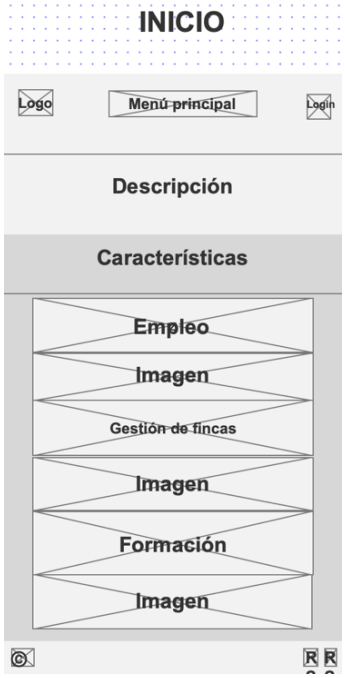

Ilustración 51: Wireframe inicio mobile

Empleo desktop

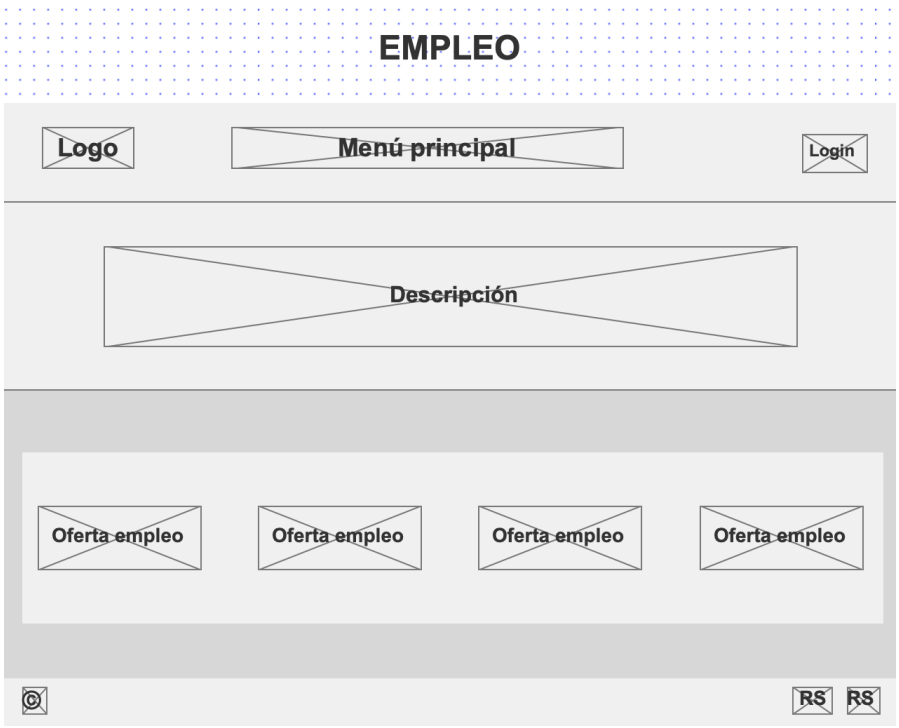

Ilustración 52: Wireframe empleo desktop

Empleo mobile

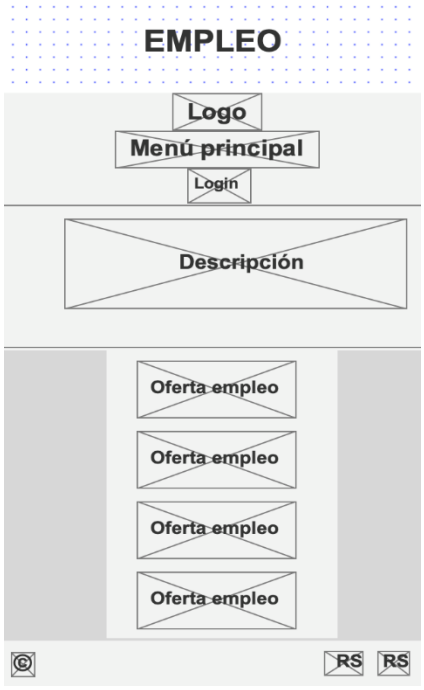

Ilustración 53: Wireframe empleo mobile

Formación desktop

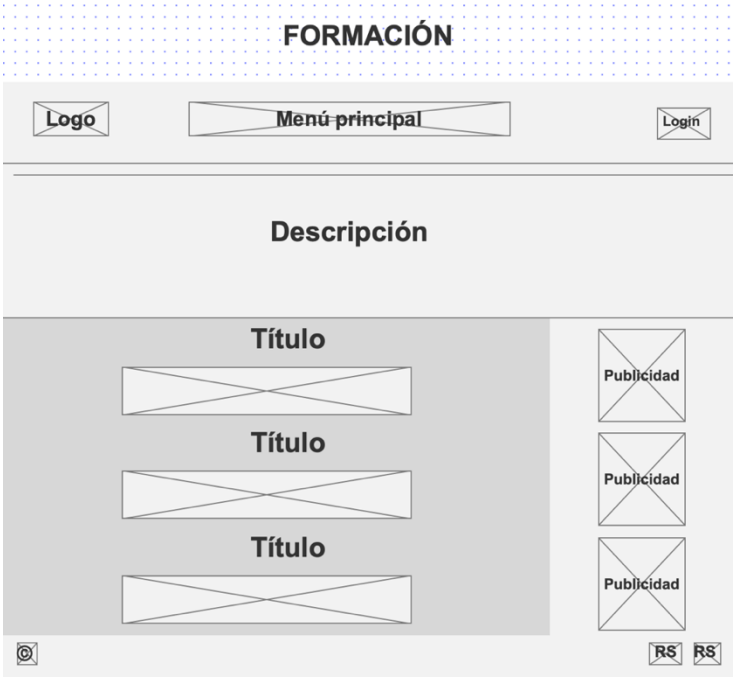

Ilustración 54: Wireframe formación desktop

Formación mobile

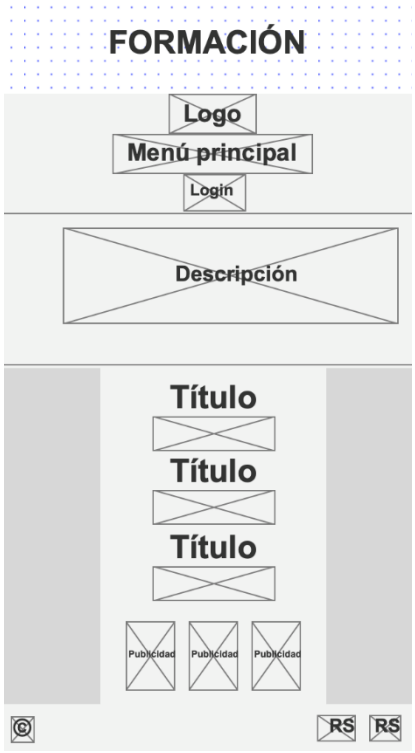

Ilustración 55: Wireframe formación mobile

Noticias desktop

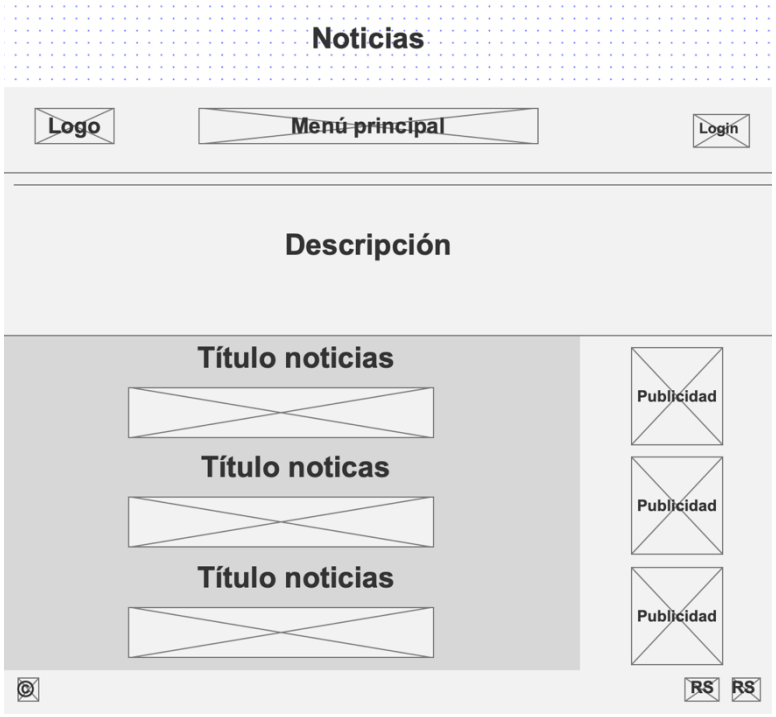

Ilustración 56: Wireframe noticias desktop

Noticias mobile

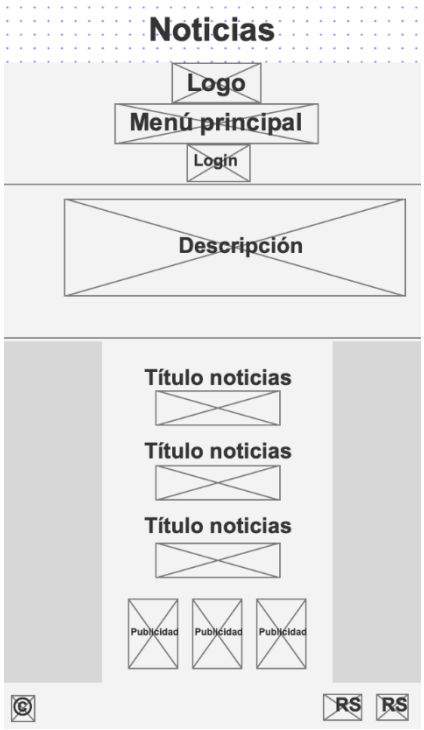

Ilustración 57: Wireframe noticias mobile

Crear empleo desktop

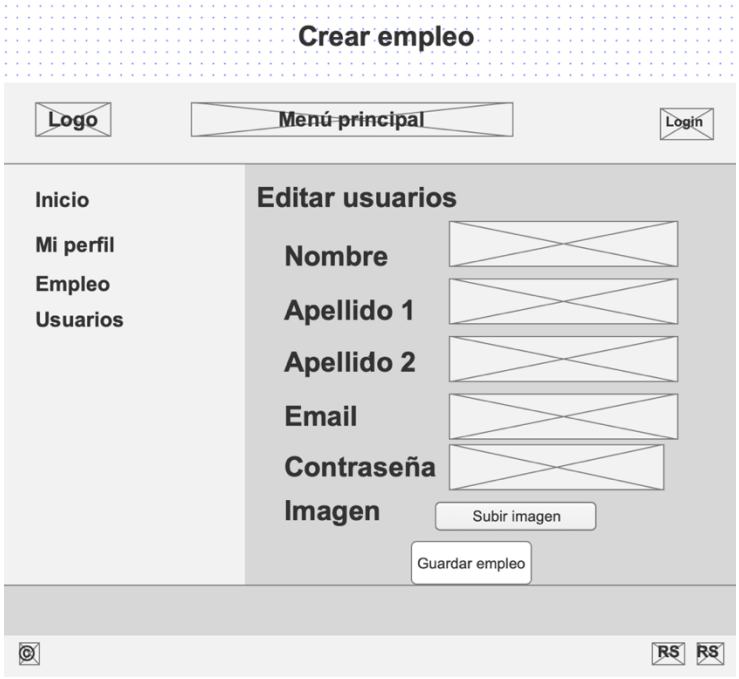

Ilustración 58: Wireframe crear empleo desktop

Crear empleo mobile

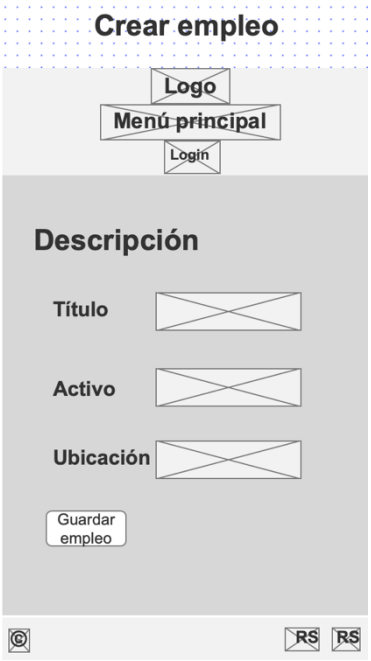

Ilustración 59: Wireframe crear empleo mobile

4.6 - Implementación

4.6.1 - Spring MVC

Spring MVC es un framework de aplicaciones web basado en el modelo Modelo-Vista-Controlador que toma ventajas de los siguientes principios de diseño:

- Inyección de dependencias (DI)
- Orientado al uso de interfaces
- Extenso uso de clases POJO
- Desarrollo testeable

Spring MVC está caracterizado por la clara separación de funciones, como son el Controller, validator, DispatcherServlet, etc. A su vez, cuenta con una configuración robusta pero sencilla de realizar, tanto para las clases del framwork como para las clases propias de la aplicación definidas como JavaBeans.

En cuanto a adaptabilidad, Spring MVC es flexible y poco intrusivo. Cuenta con diferentes formas de definir los métodos en los controladores, diversas firmas de métodos que necesitemos implementar para un escenario determinado, como puede ser el uso de anotaciones en los argumentos como:

1. @RequestParam
2. @RequestHandler
3. @PathVariable

4.6.2 - DispatcherServlet

DispatcherServlet [23] es el encargado de controlar la parte del front de SpringMVC. Coordina las peticiones web (HTTP request) y ciclo de vida. También observa el resquest y aplica el controller apropiado según la url (handler mapping). Es configurado internamente por Spring Boot.

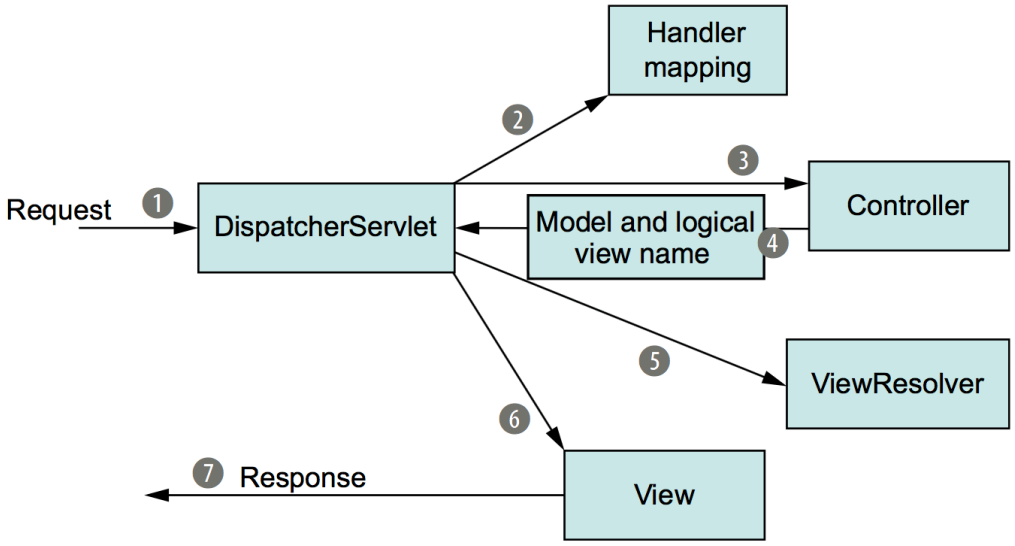

Ilustración 60: DispatcherServlet Spring MVC

4.6.3 - @RequestMapping

@RequestMapping es usada para mapear las URLs hacia los métodos handler de una clase Controller. Puede ser especificada a nivel de clase o a nivel de método.

La URL especificada en la anotación @RequestMapping a nivel de clase se concatena con la URL especificada en la anotación @RequestMapping a nivel de método a través del atributo value. La URL especificada en la anotación @RequestMapping a nivel de método es relativa a la URL especificada a nivel de clase.

4.6.4 - @PathVariable

Usa parámetros anotados para acceder a variables de las plantillas URL o URI template.y extrae los datos desde el request URI. Los valores de los parámetros son convertidos y pasados como argumentos en los métodos handler usando la anotación @PathVariable.

4.6.5 - @RequestParam

Utiliza parámetros anotados para acceder a específicos parámetros del Servlet request y extrae los datos desde request query parameters URL. Los valores de los parámetros son convertidos y pasados como argumentos en los métodos handler usando la anotación @RequestParam.

4.6.6 - Implementación del servidor

En esta sección se detallarán algunas de las partes comentadas sobre los métodos y técnicas necesarias para la implementación del servidor.

Se mostrarán algunos ejemplos obtenidos del código desarrollado para ver de forma más clara cada uno de los aspectos comentados.

La aplicación está basada en Java EE 8. Para su desarrollo se ha utilizado el IDE Eclipse, como se comentado anteriormente, y un proyecto Maven, que ayudará a la gestión y construcción del proyecto. En cuanto a la versión de Spring utilizada, ha sido la versión 5.1.6.

4.6.6.1 - Persistencia

En cuanto a la persistencia de la aplicación, ha sido utilizada JPA [24] (Java Persistence Api), la API de persistencia desarrollada para Java EE, con la que se busca unificar diferentes tecnologías que proveen un mapeo objeto-relacional bajo un único estándar. JPA hace uso de objetos simples como Entitys o POJOs.

JPA ha sido utilizada junto a Hibernate 5.3.9, una de las implementaciones más conocidas de esta API.

Gracias a JPA e Hibernate, podemos autogenerar las tablas definidas como clases Java en una base de datos relacional. Para ello, se ha hecho uso de la anotación @Entity. [25]

Como se puede observar en la imagen inferior, la clase Usuario, ha sido definida como hemos comentado con la anotación @Entity para definirla como clase persistente en base de datos.

```java
@Entity
@Table(name = "usuarios")
public class Usuario implements Serializable {

    private static final long serialVersionUID = -5454445950255018706L;

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @NotEmpty
    private String nombre;

    @NotEmpty
    private String apellido1;

    @NotEmpty
    private String apellido2;

    @NotEmpty
    @Email
    @Column(unique = true)
    private String email;

    @Temporal(TemporalType.DATE)
    private Date registro;

    @NotEmpty
    private Boolean activo;

    @NotEmpty
    private String pass;

    private String foto;

    @OneToMany(mappedBy="titular_id" ,fetch=FetchType.LAZY, cascade=CascadeType.ALL)
    private List<Empleo> empleos;

    @OneToMany(fetch=FetchType.LAZY, cascade=CascadeType.ALL)
    @JoinColumn(name="id_usuario")
    private List<Rol> roles;
```

Ilustración 61: Implementación clase Usuario

También se pueden observar otro tipo de anotaciones como por ejemplo:

- **@Id**: indica qué atributo debe ser la clave primaria de la clase en base de datos. [25]

- **@GeneratedValue**: indica que el campo indicado se autoincrementará al ser generado de forma automática. [25]

- **@NotEmpty**: indica que el valor del atributo indicado no puede ser nulo. [25]

- **@Column**: se utiliza para indicar una propiedad a la columna (atributo de la clase que será mapeado en base de datos) indicada. En este caso, se está indicando que el valor del atributo email debe de ser único. [25]

- **@Temporal**: especifica que el atributo de la clase es una fecha del tipo TemporalType.DATE. [25]

- **@OneToMany:** esta anotación es utilizada para indicar una relación entre dos clases. En este caso, se especifica que la clase Usuario tiene una relación de uno a muchos con la clase Empleo. Esta relación está implementada sobre una lista.

  Los atributos fetch y cascade se utilizan para indicar cómo se quieren cargar los datos a la hora de cargar una instancia de la clase Usuario y para indicar qué hacer con los datos relacionados con usuario (los datos almacenados en la lista empleos) cuando se realice alguna acción sobre la clase Usuario, respectivamente. En este caso, con el uso de fetch = FetchType.LAZY se ha indicado que se realice una carga perezosa de los datos (los datos relacionados sólo se recuperan cuando se demanden). Con el uso de cascade = CascadeType.ALL, se está indicando que los datos se modifiquen, actualicen… todos en cascada. [25]

- **@JoinColumn**: indica la unión (JOIN) de la relación roles a través del atributo id_usuario de la clase Roles. [25]

4.6.6.2 - Controladores

Como se comentó en el apartado **4.5.1. Introducción** del diseño de la aplicación, se ha llevado a acabo la arquitectura Modelo-Vista-Controlador. Por ello, a través de Spring hemos utilizado las características de Spring MVC para llevar a cabo el control de la comunicación de la aplicación a través de los controladores manejados por el Dispatcher Servelt de Spring.

A través de la anotación **@Controller**, hemos identificado cada uno de los controladores del sistema. En la imagen inferior, se puede observar la definición de uno de estos, **EmpleoController**.

```
@Controller
public class EmpleoController {

    @Autowired
    private IEmpleoService empleoService;

    @GetMapping(value = "/private/empleos/ficha/{id}")
    public String ficha(@PathVariable(value = "id") Long id, Map<String, Object> model, RedirectAttributes flash) {

        Empleo empleo = empleoService.findJob(id);
        if (empleo == null) {
            flash.addFlashAttribute("error", "Empleo no encontrado");
            return "redirect:/admin/empleos/lista";
        }

        model.put("empleo", empleo);
        model.put("titulo", "Detalle empleo: " + empleo.getTitulo());
        return "views/admin/empleos/ficha";
    }

    @GetMapping({"/admin/empleos/lista", "/admin/empleos/"})
    public String listarEmpleos(@RequestParam(name="page", defaultValue="0") int page, Model model) {

        Pageable pageRequest = PageRequest.of(page, 5);

        Page<Empleo> empleos = empleoService.findAllJobs(pageRequest);

        PageRender<Empleo> pageRender = new PageRender<Empleo>("/admin/empleos/lista", empleos);

        model.addAttribute("titulo", "Listado de empleos");
        model.addAttribute("empleos", empleos);
        model.addAttribute("page", pageRender);

        return "views/admin/empleos/lista";
    }
```

Ilustración 62: Implementación EmpleoController

Como podemos observar, han sido utilizadas otra serie de anotaciones para definir más elementos necesarios para llevar a cabo el correcto funcionamiento de la aplicación. Algunas de estas anotaciones y su funcionamiento son:

- **@RequestMapping**: ya explicada en el apartado **4.6.3. @RequestMapping.**

- **@GetMapping**: anotación que permite simplificar el manejo del método Get de @RequestMapping.

- **@PathVariable**: ya explicada en el apartado **4.6.4. @PathVariable.**

- **@RequestParam**: ya explicada en el apartado **4.6.5. @RequestParam.**

@Autowired: anotación que indica el uso de la inyección de dependencias de Spring (técnica explicada en el apartado **3.1.1. Spring Framework**).

4.6.6.3 - Spring Security

Para llevar a cabo el inicio de sesión y la autenticación de la aplicación, se ha utilizado **Spring Security 5.1.5** [26]. Para ello, se ha implementado la interfaz de Spring Security **UserDetailsService** a través de nuestro service **JpaUserDetailsService.java,** como se puede ver en la imagen inferior.

```
@Service("jpaUserDetailsService")
public class JpaUserDetailsService implements UserDetailsService{

    @Autowired
    private IUsuarioDao usuarioDao;

    private Logger logger = LoggerFactory.getLogger(JpaUserDetailsService.class);

    @Override
    @Transactional(readOnly=true)
    public UserDetails loadUserByUsername(String email) throws UsernameNotFoundException {

        Usuario usuario = usuarioDao.findByEmail(email);

        if(usuario == null) {
            logger.error("Error en el Login: no existe el usuario '" + email + "' en el sistema!");
            throw new UsernameNotFoundException("Email: " + email + " no existe en el sistema!");
        }

        List<GrantedAuthority> roles = new ArrayList<GrantedAuthority>();

        for(Rol role: usuario.getRoles()) {
            logger.info("Role: ".concat(role.getRol()));
            roles.add(new SimpleGrantedAuthority(role.getRol()));
        }

        if(roles.isEmpty()) {
            logger.error("Error en el Login: Usuario '" + email + "' no tiene roles asignados!");
            throw new UsernameNotFoundException("Error en el Login: usuario '" + email + "' no tiene roles asignados!");
        }

        return new User(usuario.getEmail(), usuario.getPass(), usuario.getActivo(), true, true, true, roles);
    }

}
```

Ilustración 63: Implementación interface UserDetailsService

Como podemos observar, a través de la anotación **@Service**, definimos la clase **JpaUserDetailsService.java** como un **componente** de Spring.

Con la anotación **@Transactional**, definimos el método como de solo lectura, puesto que en el método **UserDetails,** sólo realizamos una consulta a base de datos.

4.6.7 - Implementación del cliente

Para la implementación del lado del cliente, se ha utilizado el motor de plantillas **Thymeleaf** (detallado en el apartado **3.1.6. Thymeleaf**) sobre todo el contexto HTML de la aplicación.

Con el uso de Thymeleaf hemos obtenido un código más limpio y legible que el que se obtiene con el uso de otras tecnologías, como es el caso de JSP.

Una de las grandes ventajas que hemos encontrado a la hora de usar Thymeleaf junto a Spring, es el uso de **Spring Expressions Language** (Spring EL) así como el uso de los **beans** y **bindings results** de Spring para la integración de los formularios.

En la imagen inferior podemos ver un ejemplo del formulario de creación de un empleo en la parte de administración de la aplicación.

```
<div class="container py-4">
    <div class="card bg-dark text-white">
        <div class="card-header" th:text="${titulo}"></div>
        <div class="card-body">

            <form th:action="@{/admin/empleos/form}" th:object="${empleo}" method="post">
                <div class="form-group row">
                    <label for="nombre" class="col-sm-2 col-form-label">Título</label>
                    <div class="col-sm-6">
                        <input type="text" th:field="*{titulo}" class="form-control"
                            th:errorclass="'alert-danger'" />
                            <small class="form-text text-danger"
                            th:if="${#fields.hasErrors('titulo')}" th:errors="*{titulo}"></small>
                    </div>
                </div>
                <div class="form-group row">
                    <label for="activo" class="col-sm-2 col-form-label">Activo</label>
                    <div class="col-sm-6">
                        <input type="text" th:field="*{descripcion}" class="form-control"
                            th:errorclass="'form-control alert-danger'" />
                        <small class="form-text text-danger"
                            th:if="${#fields.hasErrors('activo')}" th:errors="*{activo}"></small>
                    </div>
                </div>
                <div class="form-group row">
                    <label for="apellido1" class="col-sm-2 col-form-label">Ubicación</label>
                    <div class="col-sm-6">
                        <input type="text" th:field="*{ubicacion}" class="form-control"
                            th:errorclass="'form-control alert-danger'" />
                        <small class="form-text text-danger"
                            th:if="${#fields.hasErrors('ubicacion')}" th:errors="*{ubicacion}"></small>
                    </div>
                </div>
                <div class="form-group row">
                    <label for="apellido1" class="col-sm-2 col-form-label">Publicar</label>
                    <div class="col-sm-6">
                        <input type="checkbox" th:field="*{activo}" class="form-control"
                            th:errorclass="'form-control alert-danger'" />
                        <small class="form-text text-danger"
                            th:if="${#fields.hasErrors('activo')}" th:errors="*{activo}"></small>
                    </div>
                </div>
                <div class="form-group row">
                    <div class="col-sm-6">
                        <input type="submit" value="Guardar empleo" class="btn btn-secondary" />
                    </div>
                </div>

            </form>
        </div>
    </div>
</div>
```

Ilustración 64: Código de formulario de creación de empleo

5 - PRUEBAS

En este apartado se van a mostrar el resultado de algunas de las pantallas de la aplicación durante la fase de pruebas del prototipo resultante tras el desarrollo.

5.1 - Parte pública de la aplicación

5.1.1 - Pantalla Inicio

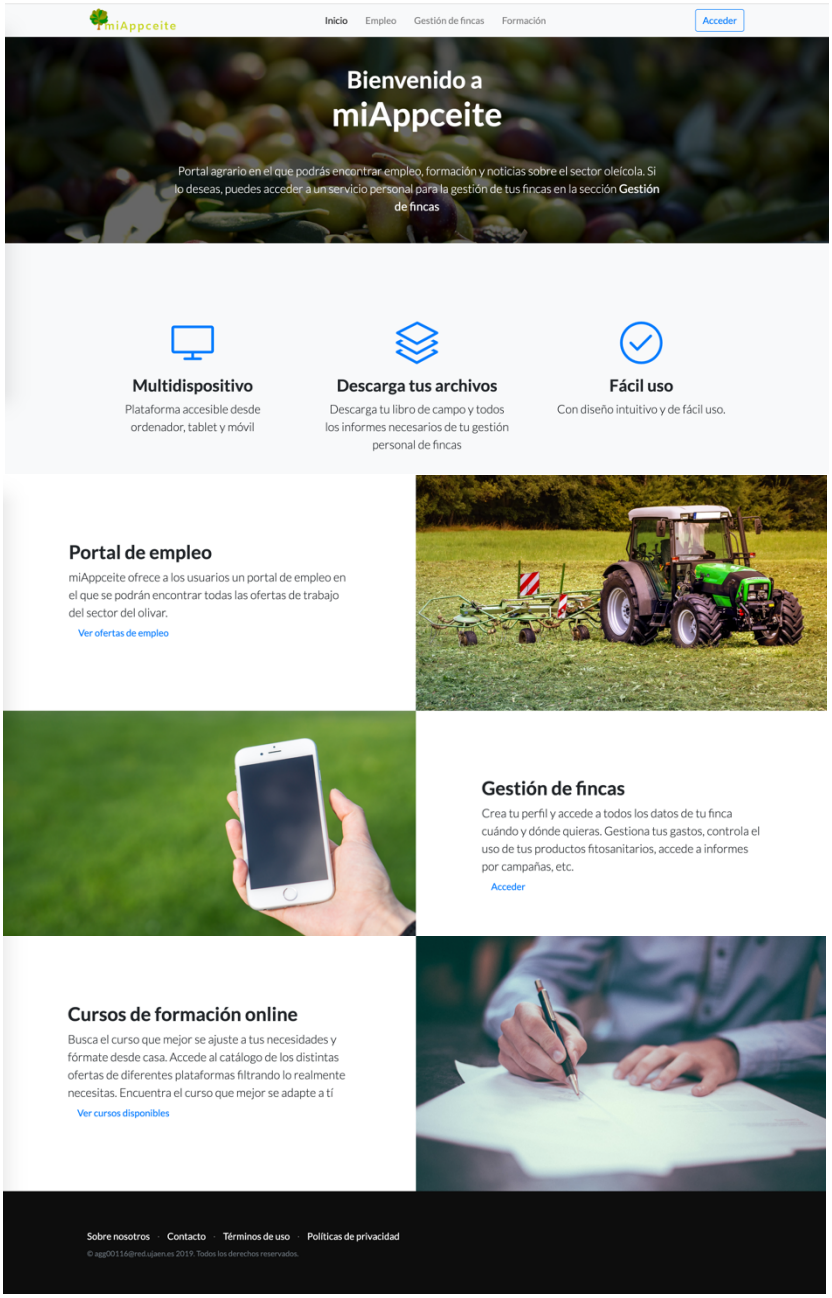

Ilustración 66: Pantalla inicio

5.1.2 - Pantalla Empleo

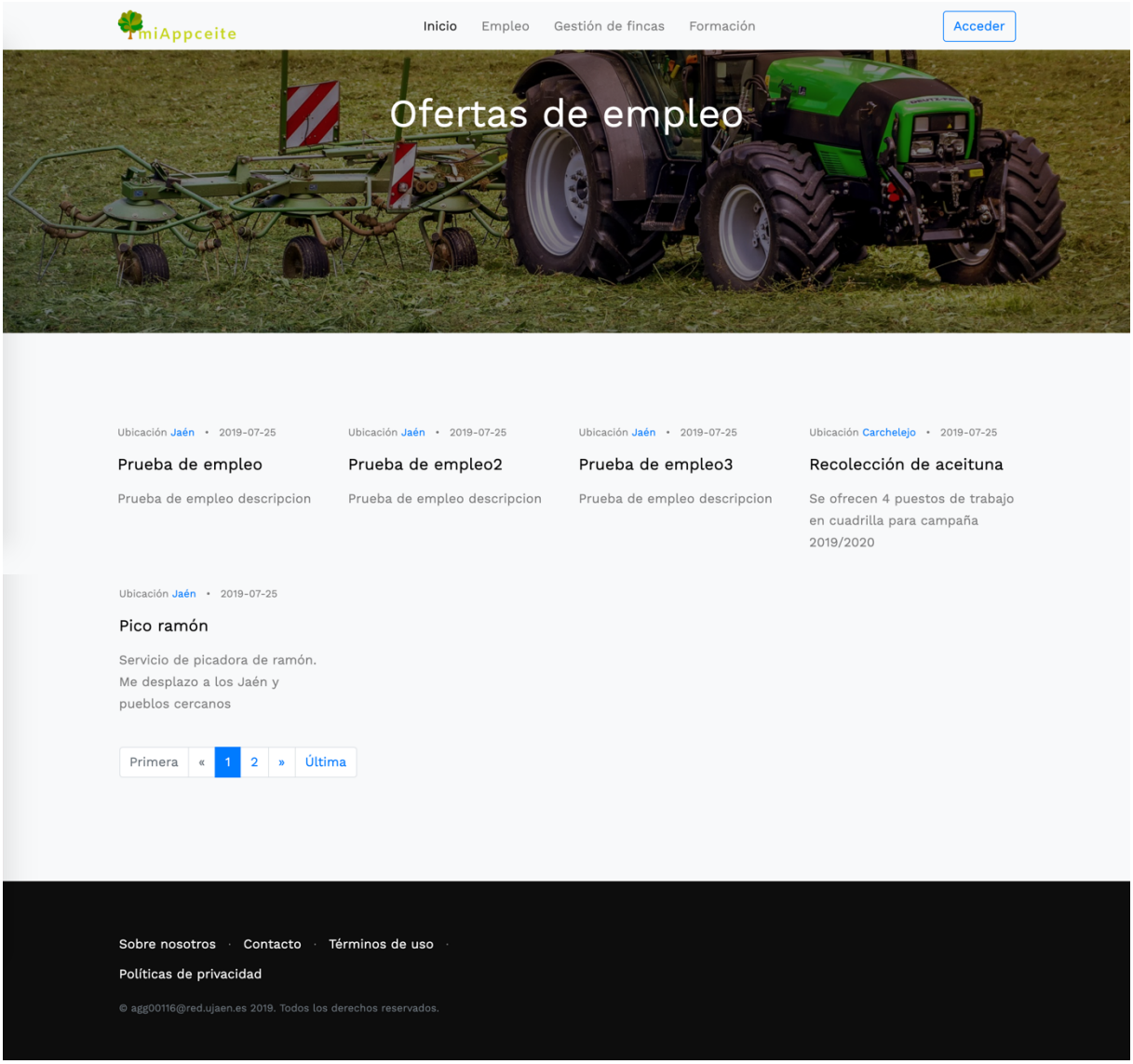

Ilustración 67: Pantalla Empleo

5.1.3 - Pantalla Gestión de Fincas

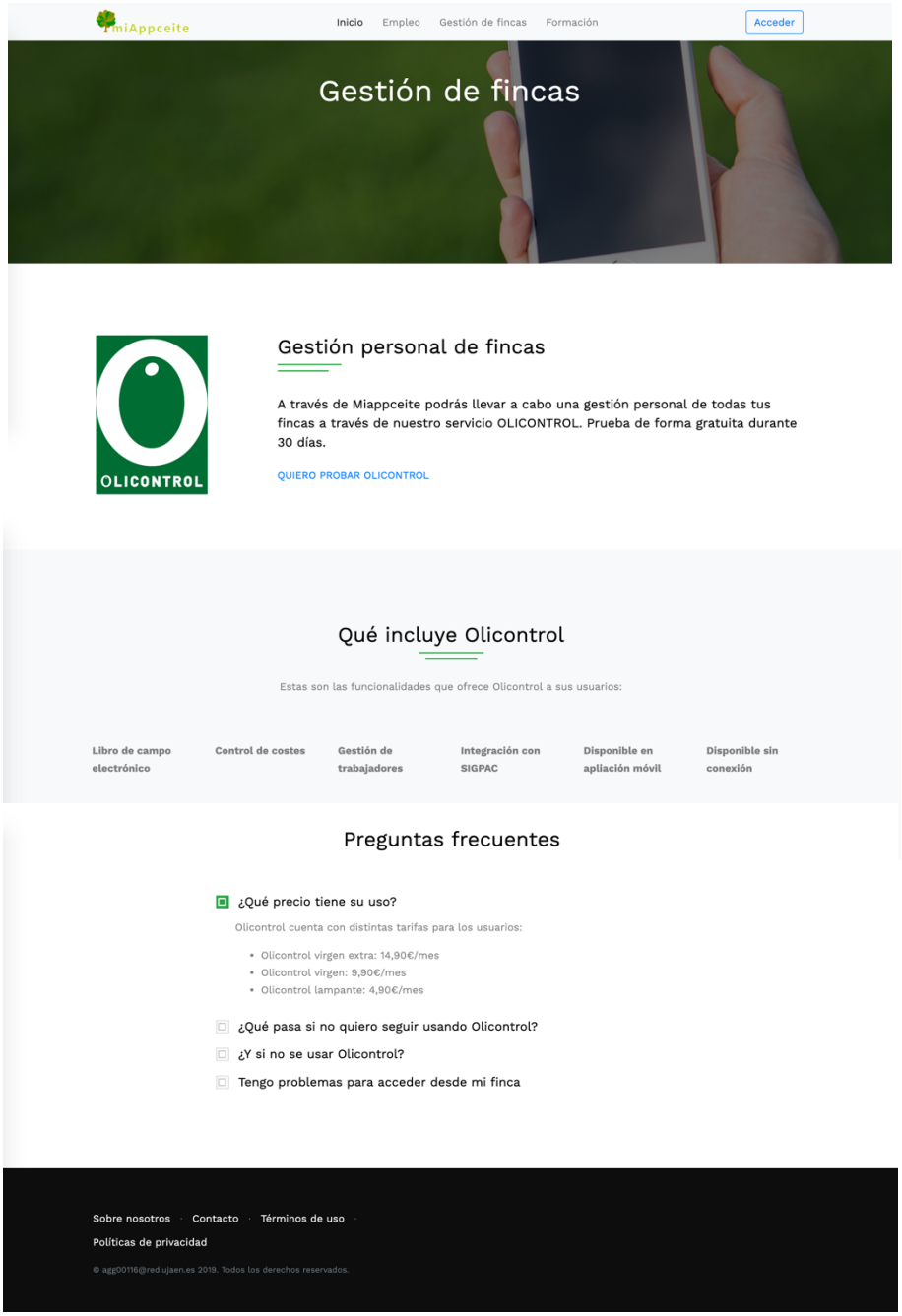

Ilustración 68: Pantalla Gestión de fincas

5.2 - Parte privada de la aplicación

5.2.1 - Pantalla Login

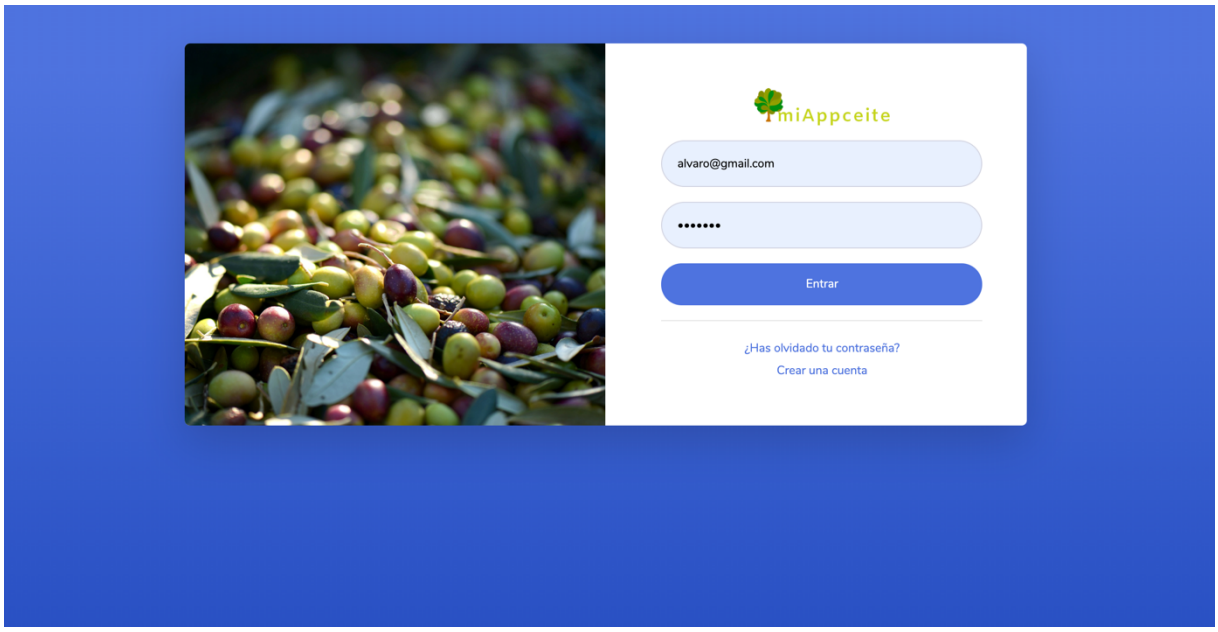

Ilustración 69: Pantalla Login

5.2.2 - Pantalla Inicio de administración

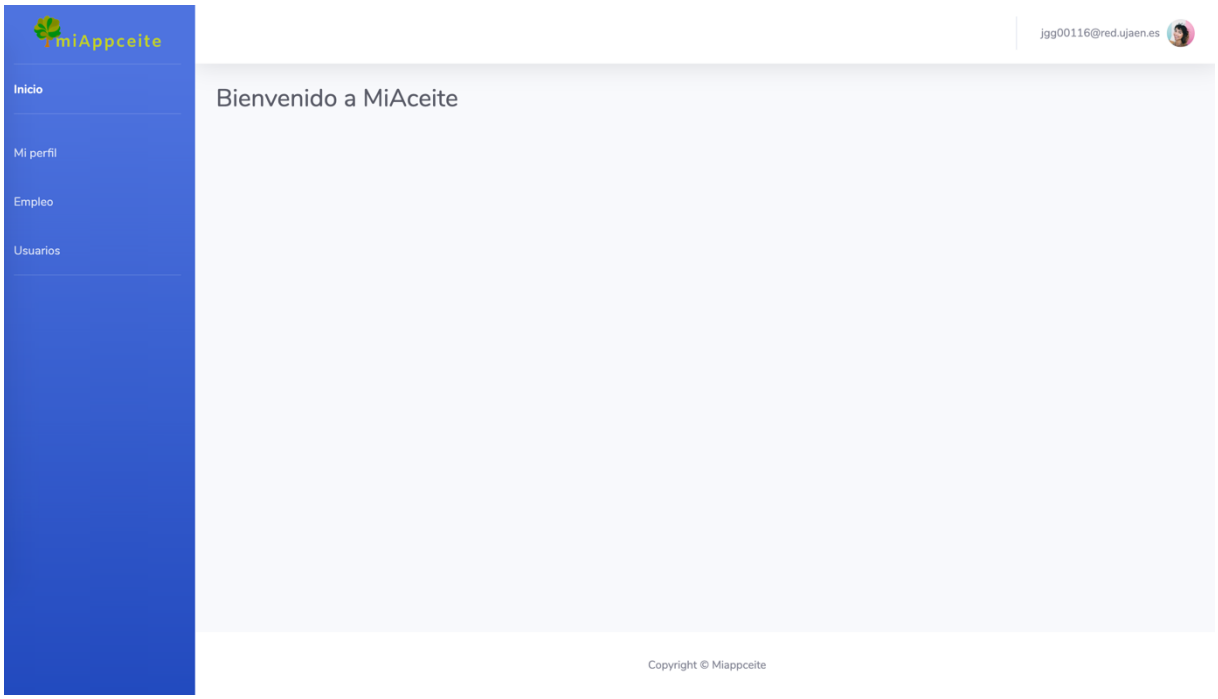

Ilustración 70: Pantalla Dashboard administración

5.2.3 - Pantalla Edición de usuario

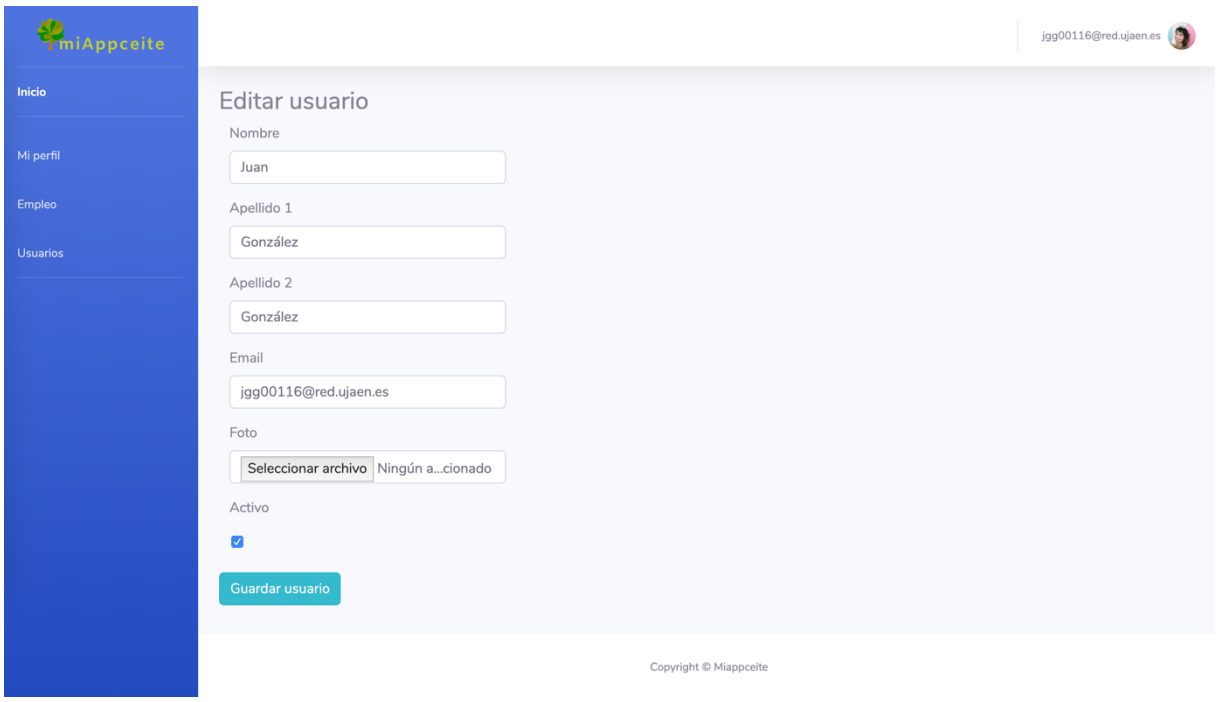

Ilustración 71: Pantalla Edición usuario administración

5.2.4 - Pantalla Listado de empleos

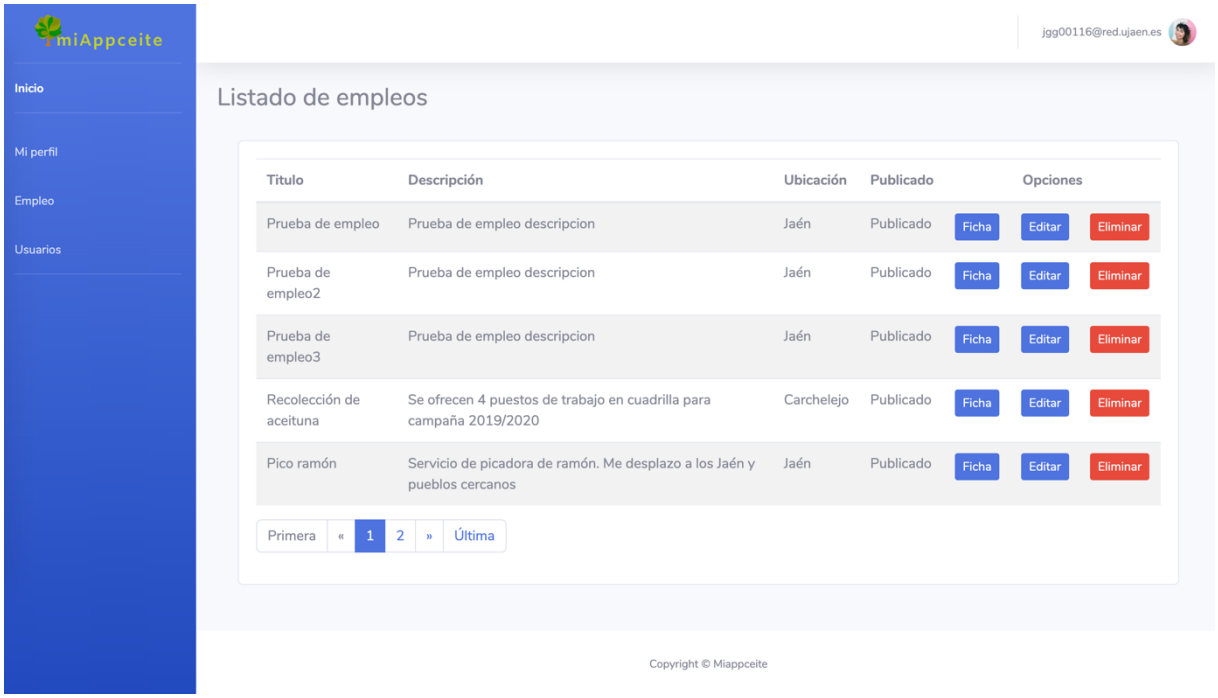

Ilustración 72: Pantalla Listado de empleos administración

6 - CONCLUSIÓN

6.1 - Reflexiones sobre el desarrollo del proyecto

En primer lugar, me gustaría comentar la decisión de mi elección de Java para el desarrollo del proyecto y del framework Spring.

Actualmente, Java es uno de los lenguajes más extendidos para el desarrollo web y de aplicaciones de escritorio. Aún siendo el más extendido, es cierto que Java no es el lenguaje más sencillo y su aprendizaje para manejarlo en todo su potencial es lento, pero gracias a la cantidad de gente que lo utiliza, se puede encontrar ayuda en muchos portales y foros destinados al desarrollo de software.

Esto, unido a la cantidad de código open source disponible para Java, hace que sea un gran atractivo a la hora de tomar una decisión para llevar a cabo un desarrollo software.

Es por eso, que animo a todos los alumnos que no tengan claro qué tecnología usar para el desarrollo del proyecto a usar Java.

En cuanto a Spring, en concreto Spring Boot, es uno de los frameworks más robustos para el desarrollo de aplicaciones Java con el que podemos simplificar la configuración del proyecto y desarrollar un código simplificado y organizado.

A su vez, Spring cuenta con modularidad de componentes, programación orientada a aspectos, productividad, es un framework que ha sido bastante probado, que es seguro y confiable.

Con la elección de Spring, he podido aprender el uso de una nueva tecnología como la que ha sido Thymeleaf, tecnología que me ha enseñado a desenvolverme de forma más fluida en el desarrollo de la parte del cliente.

6.2 - Cumplimiento de los objetivos iniciales

Desde el momento de la especificación de los objetivos establecidos en la propuesta de este TFG, se partía con la búsqueda de una solución software para un campo que actualmente está en pleno desarrollo, como es el mundo agrícola.

Es por eso que el desarrollo de este proyecto necesitaba una parte de análisis previo de la situación actual de los usuarios con el uso de nuevas tecnologías en el sector agrícola.

Fue complicado hacer entender a ciertos usuarios que una solución como esta, les podría facilitar el trabajo en muchas de las ocasiones y fue este uno de los objetivos personales que me marqué para comenzar a cambiar la situación actual de uso de las nuevas tecnologías en este sector.

Con el estudio realizado y los requisitos definidos, se llegó al primer momento de desarrollo del prototipo en el que se decidió realizar la solución más simple posible pensando en las reacciones de los usuarios a los que estaba destinada la aplicación en un futuro.

Tras el desarrollo, ha quedado como resultado final un primer prototipo que puede ser ofrecido a los usuarios para hacer un análisis previo de lo desarrollado y tras analizar las respuestas de los usuarios, pasar al desarrollo de un prototipo más completo.

6.3 - Posibles desarrollos futuros

Para el completo desarrollo de la aplicación hacen falta, principalmente, muchas más horas de análisis juntos a los usuarios y también, muchas horas de desarrollo hasta poder obtener una aplicación completamente funcional para todos sus requisitos definidos.

Tanto de análisis, como de desarrollo funcional, ya hemos comentado que hacen falta muchas horas para poder realizarlos, pero para otros aspectos como son la seguridad de la aplicación, un diseño bien analizado y desarrollado y llegar a muchos más usuarios potenciales, deberán también de tenerse en cuenta para un desarrollo final exitoso.

Personalmente, tengo como propósito poder llevar a cabo el desarrollo total de la aplicación, ya sea solo o con ayuda de amigos y/o compañeros. Pienso que actualmente hacen falta más herramientas de este tipo en el mercado, y más aún, cuando hay un mercado tan grande y tan pocas ofertas ofrecidas a los usuarios.

7 - ANEXO. CONTENIDO CD-ROM

- Carpeta del proyecto: carpeta con el código fuente del proyecto.
- Memoria: memoria del proyecto en formato PDF.
- Documentación necesaria para la entrega formal del TFG:
    - Informe valorable del tutor
    - Autorización de publicación

8 - BIBLIOGRAFÍA

[1] Pressman, R. (2010). *Ingeniería del Software.* McGraw-Hill.

[2] Revolución digital agrícola https://lahuertadigital.es/revolucion-digital-agricola/

[3] Solución agrícola Boosteragro https://boosteragro.com

[4] Solución agrícola Nanny https://futurizable.com/agrotech/

[5] Solución agrícola Cropti https://cropti.com

[6] Solución agrícola Agroguía http://agroguia.es

[7] Tecnología agrícola https://www.bialarblog.com/tecnologia-agricola-agtech-agricultura/

[8] Spring Framework https://docs.spring.io/spring/docs/4.3.x/spring-framework-reference/html/overview.html

[9] Librería Bootstrap https://getbootstrap.com/docs/4.3/getting-started/introduction/

[10] Motor de plantillas Thymeleaf https://www.thymeleaf.org/

[11] Framework jQuery https://api.jquery.com/

[12] Herramienta Eclipse https://www.eclipse.org/ide/

[13] Herramienta MySQL Workbench https://www.mysql.com/products/workbench/

[14] Herramienta Visual Paradigm https://www.visual-paradigm.com/

[15] Herramienta Google Formularios https://www.google.es/intl/es/forms/about/

[16] Apuntes asignatura Ingeniería de Requisitos. Tema 1- Introducción a la Ingeniería de Requisitos.

[17] Lenguaje de modelado software UML y metodología Scrum https://www.progressalean.com/procesos-de-informacion-y-tecnologia-modelo-scrum/

[18] Succeeding with agile: software development using Scrum. M. Cohn – Addison-Wesley, 2010

[19] Lenguaje de modelado software UML https://www.uml.org/

[20] Arquitectura software MVC https://sites.google.com/site/portafoliomatby/m

[21] Template de diseño landing page https://startbootstrap.com/previews/landing-page/

[22] Template de diseño de administración https://startbootstrap.com/themes/sb-admin-2/

[23] Dispatcher Servlet https://stackoverflow.com/questions/2769467/what-is-dispatcher-servlet-in-spring

[24] API de persistencia JPA https://www.oracle.com/technetwork/java/javaee/tech/persistence-jsp-140049.html

[25] Mapeado objeto relacional Hibernate https://docs.jboss.org/hibernate/jpa/2.1/api/javax/persistence/

[26] Framework Spring Security https://spring.io/projects/spring-security