

Memoria del proyecto

**Desarrollo de una plataforma o aplicación web para
la gestión de un gimnasio**

Trabajo fin de grado
Grado en Ingeniería Informática



**VNiVERSiDAD
DE SALAMANCA**

Julio de 2021

Autor

Alberto Martín Peralejo

Tutores

**André Filipe Sales Mendes
Gabriel Villarrubia González
Juan Francisco de Paz Santana**

Certificado de los tutores

D. André Sales Mendes, D. Juan Francisco de Paz Santana y D. Gabriel Villarrubia González, profesores del Departamento de Informática y Automática de la Universidad de Salamanca.

HACEN CONSTAR: Que el trabajo titulado “Desarrollo de una plataforma o aplicación web para la gestión de un gimnasio” ha sido realizado por D. Alberto Martín Peralejo, con el número de documento 70942032G y constituye la memoria del trabajo realizado para la superación de la asignatura Trabajo de Fin de Grado de la Titulación Grado de Ingeniería Informática de esta Universidad

Y para que así conste a todos los efectos oportunos.

En Salamanca, a de julio de 2021

D. André Filipe Sales
Mendes

D. Juan Francisco de Paz
Santana

D. Gabriel Villaruibia
Gónzalez

Resumen

Este proyecto realizado como Trabajo Fin de Grado consiste en el diseño, desarrollo e implementación de una aplicación web para la gestión de un gimnasio o centro deportivo.

En la actualidad, debido a la pandemia causada por el COVID-19, muchos gimnasios y centros deportivos han tenido que controlar sus aforos o incluso cerrar sus puertas para evitar aglomeraciones de personas y posibles contagios.

El objetivo principal del proyecto es la realización de una aplicación que permita a los clientes de estos centros poder asistir a los mismos con un control total del aforo. Además, los administradores de estos gimnasios tendrán acceso a una herramienta que llegue a mucha más gente y con la que podrán gestionar con eficiencia su negocio.

El desarrollo de la aplicación se divide en dos partes: **Frontend** y **Backend**.

En la parte del Frontend, cuyo objetivo es ser la interfaz del lado del cliente, se ha utilizado JavaScript con Vue.js, un framework MVVM (Model-View-ViewModel) con Licencia MIT. Para el diseño de los componentes de Vue.js usados se ha usado el framework se ha utilizado Vuetify, un framework que sigue las convenciones de la normativa de diseño Material Design, desarrollado por Google.

La aplicación usa un almacén de estados centralizados denominado Store y proporcionado por la librería de gestión del estado denominada Vuex.

En la parte del Backend, cuyo objetivo es el control y almacenamiento de los datos, se ha hecho uso de Node.js, que es un entorno en tiempo de ejecución multiplataforma, de código abierto, para la capa del servidor basado en el lenguaje de programación JavaScript.

Para el almacenamiento de datos se utiliza MongoDB, y para gestión de la gestión de autenticación se usado JWT.

Palabras clave: Gimnasios, aforo, gestión, actividades, deporte, Vue.js, Node.js, MongoDB.

Abstract

This End Degree Project consists in the design, development and implementation of a web application for the management of an sport center.

Currently, due to COVID-19 pandemic, many gyms and sport centers have had to control their capacity or even close to avoid crowds of people and possible infections.

The main objective of the project is the realization of a web applications that allow the clients of these centers access to theirs facilities. Furthermore, the administrators of these centers could manage them with efficiency.

The development of the application divides in two parts: **Frontend** and **Backend**.

At the Frontend, whose main objective is being the client's side interface, it has been used JavaScript with Vue.js, a MVVM framework (Model-View-ViewModel) with MIT License. For the Vue.js components design it has been used Vuetify, a Material Design framework, developed by Google.

The application also uses a centralized state warehouse called Store. It has been provided by the state management library, Vuex.

At the Backend, whose main objective is the data control and storage, it has been used Node.js, an execution time environment, open source, for the server layer based on Javascript.

For the data storage it has been used MongoDB, and for the authentication management it has been used JWT.

Palabras clave: Gym, capacity, management, activities, sport, Vue.js, Node.js, MongoDB.

Índice general

1. Introducción	1
2. Estudio de mercado	3
2.1. Basic-fit	3
2.2. TIMP - App de reservas	4
3. Objetivos	5
3.1. Objetivos del sistema	5
3.2. Objetivos personales	5
4. Conceptos teóricos	7
4.1. Backend y Frontend	7
4.2. JWT	7
4.3. NoSQL	7
4.4. REST API	8
4.5. Herramientas CASE	9
5. Técnicas y herramientas	10
5.1. Técnicas y herramientas usadas en el Backend	10
5.1.1. Javascript	10
5.1.2. JSON	10
5.1.3. Node.js	10
5.1.4. NPM	11
5.1.5. MongoDB	11
5.1.6. Express	12
5.1.7. Mongoose	12
5.1.8. Node-cron	12
5.1.9. Bcrypt	13
5.1.10. Nodemailer	13
5.1.11. Handlebars	13
5.1.12. Multer	13
5.1.13. JWT	13
5.1.14. Visual Studio Code	13
5.1.15. Postman	14
5.2. Técnicas y herramientas usadas en el Frontend	14
5.2.1. Vue.js	14
5.2.1.1. Vuex	15
5.2.1.2. Vuetify	15
5.2.1.3. VueRouter	15
5.2.1.4. VueSocialSharing	15
5.2.1.5. Vuelidate - VeeValidate	16
5.2.1.6. Vue-2 GoogleMaps	16
5.2.1.7. VueFunctionalCalendar	16

5.2.2.	HTML	16
5.2.3.	CSS	17
5.2.4.	Axios	17
5.2.5.	Sweetalert	17
5.2.6.	PayPal	17
5.3.	Herramientas CASE	17
5.3.1.	REM	17
5.3.2.	Visual Paradigm	18
5.3.3.	Microsoft Proyect	18
5.3.4.	EZEstimate	18
5.4.	Herramientas de control de versiones	18
5.4.1.	Git	18
5.4.2.	GitHub	18
5.5.	Herramientas de despliegue	19
5.5.1.	Git, GitHub y GitHub Actions	19
5.5.2.	PM2	19
5.5.3.	Certbot	19
5.6.	Herramientas de generación de documentación	20
5.6.1.	JSdoc	20
5.6.2.	Vue Styleguidist	20
5.6.3.	Overleaf y L ^A T _E X	20
6.	Aspectos relevantes del desarrollo	21
6.1.	Marco de trabajo	21
6.2.	Estimación del esfuerzo	22
6.3.	Planificación temporal	24
6.4.	Especificación de requisitos	25
6.4.1.	Participantes	25
6.4.2.	Objetivos del sistema	25
6.4.3.	Requisitos de información	26
6.4.4.	Requisitos de restricción de información	26
6.4.5.	Requisitos funcionales	26
6.4.6.	Requisitos no funcionales	29
6.5.	Análisis de requisitos	29
6.5.1.	Modelo de dominio	29
6.5.2.	Realización de casos de uso - análisis	30
6.5.3.	Clases de análisis	30
6.5.4.	Vista arquitectónica del modelo de análisis	31
6.6.	Diseño del sistema	31
6.6.1.	Patrones arquitectónicos	31
6.6.1.1.	Modelo-vista-modelo de vista (MVVM)	31
6.6.1.2.	Singleton	32
6.6.1.3.	State management	32
6.6.2.	Subsistemas de diseño	33
6.6.3.	Clases de diseño	33
6.6.4.	Vista arquitectónica del modelo de diseño	35
6.6.5.	Realización de casos de uso - diseño	35

6.6.6. Diseño de bases de datos	37
6.6.7. Modelo de despliegue	37
6.7. Implementación	39
6.7.1. Frontend	39
6.7.2. Backend	40
6.7.3. Hosting	41
6.7.4. Documentación técnica	41
6.8. Pruebas	41
6.9. Funcionalidad de la aplicación	41
6.9.1. Usuario no logueado	41
6.9.2. Usuario logueado	43
6.9.3. Administrador	48
6.9.4. Sistema	50
6.9.5. Notificaciones	51
7. Conclusiones y líneas de trabajo futuras	52
7.1. Conclusiones	52
7.2. Líneas de trabajo futuras	53
8. Referencias	55

Índice de figuras

2.1. Basic-fit, paso 4	3
2.2. TIMP	4
4.1. Arquitectura API RESTful	9
5.1. Patrón State Management	15
6.1. Proceso unificado	22
6.2. Estimación	23
6.3. Tareas, primera parte	24
6.4. Diagrama de Gantt, primera parte	25
6.5. Diagrama de paquetes	26
6.6. Jerarquía de los actores	27
6.7. Diagrama de casos de uso, paquete gestión de pádel	27
6.8. UC-0038 Reservar hora pádel	28
6.9. Modelo de dominio	29
6.10. Diagrama de secuencia UC-0038 Reservar hora pádel	30
6.11. Patrón MVVM	31
6.12. Patrón Singleton	32
6.13. Subsistemas de diseño	33
6.14. Frontend - Modelo de Vista	34
6.15. Backend - Controladores	34
6.16. Vista arquitectónica	35
6.17. Diagrama de secuencia UC-0033 Reservar hora gimnasio	36
6.18. Base de datos	37
6.19. Diagrama de despliegue	38
6.20. Boceto de la vista del perfil	39
6.21. Vista final del perfil	40
6.22. Pantalla Iniciar sesión	42
6.23. Navegador de Usuario no logueado	43
6.24. Navegador usuario logueado	44
6.25. Pantalla Actividades	44
6.26. Pantalla Actividad Kick Boxing 1	45
6.27. Pantalla Actividad Kick Boxing 2	45
6.28. Pantalla Actividad Kick Boxing 3	46
6.29. Pantalla Actividad Kick Boxing 4	46
6.30. Pantalla Gimnasio (sólo una parte)	47
6.31. Pantalla Día Gimnasio 1 (sólo primera parte)	47
6.32. Pantalla Cancelar reservas Gimnasio	48
6.33. Navegador Administrador	48
6.34. Pantalla Administración	49
6.35. Pantalla Ver usuario	49
6.36. Pantalla Crear actividad	50
6.37. Notificación apuntarse a una actividad	51

7.1. Balizas BLE	53
7.2. Sugerencia registro	53
7.3. Aplicación móvil	54

1. Introducción

El mundo actual se ve afectado por la pandemia causada por el virus COVID-19 y, aunque la situación actual ha mejorado respecto al año 2020, se sigue necesitando estar alerta para que la situación no empeore.

Debido a esta pandemia el mundo se ha tenido que enfrentar a varias adversidades, como el confinamiento, lo que ha provocado que negocios como los centros deportivos o gimnasios se vean obligados a cerrar sus puertas o a controlar los clientes que acuden a los mismos.

El objetivo de este proyecto fin de grado es proporcionar una herramienta que ayude a la gestión de estos centros y que proporcione a los clientes un sistema de reservas con el que se asegura que se cumplen los aforos establecidos.

En este documento se explicarán los aspectos más importantes en el desarrollo del proyecto. La estructura de este documento es la siguiente:

- **Estudio de mercado:** se muestran aplicaciones similares ya existentes en el mercado.
- **Objetivos:** se describen los objetivos que se quieren lograr con la realización del proyecto.
- **Conceptos teóricos:** se detallarán los concepto necesarios para la correcta compresión del proyecto.
- **Técnicas y herramientas:** se hace un repaso a través de las tecnologías y aplicaciones utilizadas para el desarrollo del proyecto.
- **Aspectos relevantes del desarrollo:** recoge los aspectos más importantes en el desarrollo y despliegue de la aplicación.
- **Conclusiones y líneas de trabajo futuras:** se comentarán que conclusiones obtenidas tras la realización del proyecto y posible trabajo futuro sobre la aplicación.
- **Bibliografía.**

Este documento se ve complementado por los siguientes anexos:

- **Anexo I - Plan de proyecto:** realización de la estimación del esfuerzo y la planificación temporal del proyecto.
- **Anexo II - Especificación de requisitos:** alberga la especificación de requisitos del sistema.
- **Anexo III - Análisis de requisitos:** documentación sobre la fase de análisis de los requisitos.

- **Anexo IV - Diseño del sistema software:** recoge la documentación de la fase de diseño del sistema.
- **Anexo V - Documentación técnica:** documento que facilita la comprensión del código desarrollado.
- **Anexo VI - Manual de usuario:** por último, este anexo tiene la finalidad de ser una guía para el usuario de la interacción con el sistema.

2. Estudio de mercado

Para el desarrollo de este proyecto se ha realizado un estudio de mercado con el objetivo de encontrar aplicaciones similares. Esto ha proporcionado conocimientos sobre las necesidades de los usuarios.

Se han encontrado varias aplicaciones, a continuación se muestran algunas.

2.1. Basic-fit

Esta aplicación nació con el mismo objetivo que el proyecto desarrollado, controlar los aforos en un gimnasio. Su funcionamiento se divide en varios pasos, en la figura 2.1 podemos ver el paso 4.

- Paso 1: Seleccionar el centro donde se entrena.
- Paso 2: Seleccionar el día para entrenar, sólo permite 7 días de antelación.
- Paso 3: Seleccionar la duración del entrenamiento. Esto se hace en intervalos de 15 minutos con un tiempo máximo de 90 minutos.
- Paso 4: Confirmar reserva.

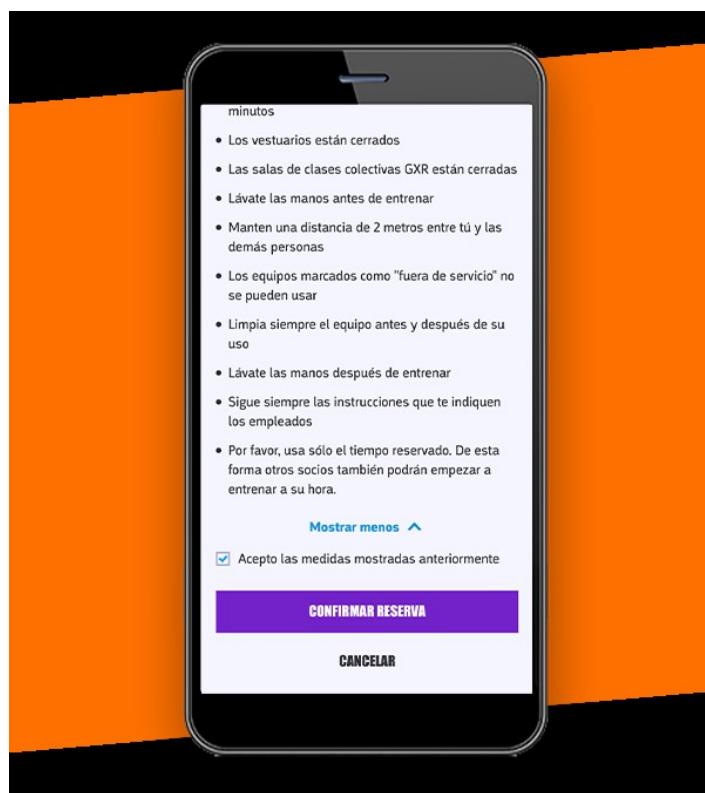


Figura 2.1: Basic-fit, paso 4

2.2. TIMP - App de reservas

Esta aplicación permite hacer reservas de diferentes actividades. Los clientes de esta aplicación tendrán una vista global de los horarios, podrán revisar sus próximas reservas, ver el estado de sus bonos o estar al corriente de sus pagos pendientes. En la figura 2.2 podemos ver un ejemplo de esta aplicación.

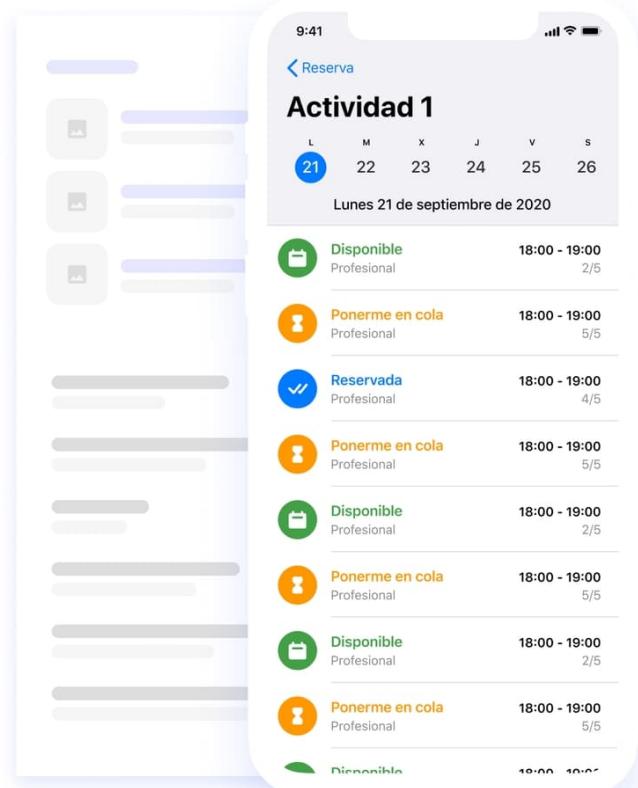


Figura 2.2: TIMP

3. Objetivos

En esta sección se detallarán los objetivo que debe cumplir el sistema para desarrollar el Trabajo de Fin de Grado. Se expondrán los objetivos del sistema en primer lugar y en segundo lugar, los objetivos personales que se pretenden alcanzar.

3.1. Objetivos del sistema

El objetivo principal del sistema es el control de aforos en gimnasios y centros deportivos. Esto se hará mediante un sistema de reservas.

Los objetivos del sistema se pueden ver más detalladamente en el siguiente listado:

- **Gestión de usuarios:** El sistema deberá manejar los roles de Usuario (clientes) y administradores (gestionan el sistema). Los administradores podrán borrar, controlar usuarios y gestionar actividades.

Además, para acceder al sistema es necesario registrarse y autentificarse.

- **Gestión de datos de usuario:** El sistema deberá permitir a los usuarios del sistema modificar sus datos.
- **Gestión de matrículas:** El sistema deberá permitir al usuario seleccionar su tipo de matrícula, pagarla y actuar dependiendo del tipo de la misma.
- **Gestión de actividades:** El sistema deberá permitir a los administradores del sistema crear y borrar actividades y a los clientes apuntarse, desapuntarse y dar su opinión de estas.

Además se podrá compartir en diferentes redes.

- **Gestión de reseñas:** El sistema deberá permitir a los usuarios crear reseñas, borrar las suyas y dar like o dislike a las reseñas propias o de otros usuarios.
- **Gestión de gimnasio y pádel:** El sistema deberá permitir al usuario gestionar sus reservas en el gimnasio, tanto realizarlas como cancelarlas.

Además los administradores podrán registrar máquinas en el gimnasio.

3.2. Objetivos personales

A parte de ser Trabajo de Fin de Grado un requisito indispensable para obtener el Título de Grado, los objetivos personales que me han decantado por desarrollar este proyecto han sido tres.

El primer objetivo es el de desarrollar un servicio informático completo: desde la concepción de la idea, hasta la obtención de un producto funcional y la realización de

pruebas, permitiendo poner a prueba todo lo aprendido durante la carrera sin centrarme exclusivamente en una disciplina determinada.

Realizar un sistema al 100 % supone aplicar el conocimiento adquirido sobre programación orientada a objetos e ingeniería software y comprender qué problemas resuelven las herramientas que nos proporcionan estas disciplinas.

El segundo de los objetivos es el interés que he tenido durante toda la carrera por el desarrollo web pero que nunca había llegado a aprender.

El tercer objetivo es el de crear una aplicación que ayude a los gimnasios y centros deportivos a abrir sus puertas, dado que yo soy una persona físicamente activa y la pandemia suspuso el cierre de los mismos.

4. Conceptos teóricos

En esta sección se procederá a explicar aquellos conceptos que necesitan ser detallados para conseguir una fácil compresión del trabajo.

4.1. Backend y Frontend

El **backend** es la parte del desarrollo web que controla la lógica de una página web y su correcto funcionamiento. Se trata todas las acciones que suceden en una aplicación web pero que son invisibles para el usuario como, por ejemplo, la comunicación con el servidor o la actualización de elementos de la base de datos.

El **frontend** es la parte del sitio web con la que el interacciona el usuario. Se suele llamar “el lado del cliente”.

4.2. JWT

JSON Web Token (JWT) es un estándar abierto (RFC-7519) basado en JSON para crear un token que sirva para enviar datos entre aplicaciones o servicios y certificar que sean válidos, correctos y seguros. Su caso más común de utilización es para controlar la autenticación en aplicaciones móviles o web.

4.3. NoSQL

Las Bases de Datos NoSQL (“Not Only SQL”) pertenecen al modelo no relacional. Las principales características y ventajas de este tipo son:

- Su nombre, No Sólo SQL.
- Generalmente, su arquitectura es distribuida, almacenándose la información en más de una máquina del sistema. Por lo que los sistemas que las soportan tienen una mayor escalabilidad horizontal (a más nodos más rendimiento) y también mayor tolerancia a fallos en los distintos nodos.
- Los datos no tienen que almacenarse en tablas.
- Son más eficientes en el procesamiento de los datos que las BBDD relacionales.
- Usan lo que se conoce como consistencia eventual, esto consiste en que los cambios realizados en los datos serán replicados a todos los nodos del sistema, lo cual aumenta el rendimiento de estos sistemas en contraposición a las propiedades ACID de las BBDD relacionales (“Atomicity, Consistency, Isolation and Durability” – Atomicidad, Consistencia/Integridad, Aislamiento y Durabilidad).

4.4. REST API

REST (Representational State Transfer- Transferencia de Estado Representacional), definido por Roy Fielding, es cualquier interfaz entre sistemas que use HTTP para obtener datos o generar operaciones sobre esos datos en todos los posibles formatos, como XML y JSON.

Características:

- **Protocolo cliente/servidor sin estado:** cada petición HTTP alberga la información necesaria para ejecutarla, lo que permite que ni cliente ni servidor necesiten recordar ningún estado previo.
- **Operaciones:** POST (crear), GET (leer y consultar), PUT (editar) y DELETE (eliminar).
- **Los objetos en REST siempre se manejan desde la URI.**
- **Interfaz uniforme:** para la transferencia de datos en un sistema REST, este ejecuta acciones concretas (POST, GET, PUT y DELETE) sobre los recursos, siempre y cuando estén identificados con una URI. Esto simplifica la existencia de una interfaz uniforme que sistematiza el proceso con la información.
- **Sistema de capas:** arquitectura jerárquica entre los componentes. Cada una de estas capas lleva a cabo una funcionalidad dentro del sistema REST.
- **Uso de hipermedios:** hipermedia es una extensión del concepto de hipertexto. La hipermedia permite que el usuario puede navegar por el conjunto de objetos a través de enlaces HTML. En el caso de una API REST, el concepto de hipermedia explica la capacidad de una interfaz de desarrollo de aplicaciones de proporcionar al cliente y al usuario los enlaces adecuados para ejecutar acciones concretas sobre los datos.

Ventajas:

- **Separación entre el cliente y el servidor.**
- **Fiabilidad, visibilidad y escalabilidad.**
- **La API REST siempre es independiente del tipo de plataformas o lenguajes.**

En la figura 4.1 podemos observar su arquitectura:

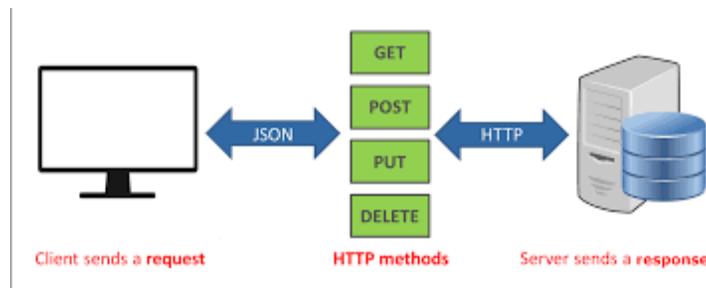


Figura 4.1: Arquitectura API RESTful

4.5. Herramientas CASE

Las herramientas CASE (Computer Aided Software Engineering, Ingeniería de Software Asistida por Computadora) son aplicaciones informáticas o programas informáticos destinadas a aumentar el balance en el desarrollo de software de calidad reduciendo el costo de las mismas en términos de tiempo y de dinero.

5. Técnicas y herramientas

En esta sección se detallarán las técnicas, herramientas, frameworks, lenguajes, etc, utilizados en el desarrollo del proyecto completo.

Se hará una separación de técnicas y herramientas utilizadas en el Backend de las del Frontend, así como de las herramientas usadas para la documentación, las herramientas usadas para el despliegue y las herramientas CASE.

5.1. Técnicas y herramientas usadas en el Backend

5.1.1. Javascript

JavaScript (JS abreviado) es un lenguaje de programación ligero, interpretado, débilmente tipado, dinámico, dirigido por eventos, basado en prototipos, imperativo y orientado a objetos con funciones de primera clase.

El uso más conocido de JS es el desarrollo de páginas web, en el lado del cliente, junto con HTML y CSS.

El estándar para JavaScript es ECMAScript.

5.1.2. JSON

JSON, cuyo nombre corresponde a las siglas JavaScript Object Notation o Notación de Objetos de JavaScript, es un formato ligero de intercambio de datos, que resulta sencillo de leer y escribir para los programadores y simple de interpretar y generar para las máquinas.

5.1.3. Node.js

Node.js es un entorno JavaScript, multiplataforma que nos permite ejecutar en el servidor, de manera asíncrona, con una arquitectura orientada a eventos y basado en el motor V8 desarrollado por The Chromium Project, de Google.

Node.js funciona como un modelo de evaluación en un único hilo de ejecución, con entradas y salidas asíncronas que pueden ejecutarse de forma concurrente sin aumentar los costes por el cambio de contexto.

Se decidió usar Node.js debido a la cantidad de módulos con una funcionalidad probada y robusta que ofrece, cada uno de ellos mantenido por desarrolladores independientes.

5.1.4. NPM

NPM (Node Package Manager), es el gestor de paquetes por defecto para Node.js, un entorno de ejecución para JavaScript, bajo Artistic License 2.0.

NPM proporciona acceso a cientos o miles de paquetes reutilizables. Tiene además la mejor en su clase resolución de dependencias y puede usarse para automatizar la mayor parte de la cadena de herramientas de compilación.

Esta herramienta funciona de dos formas:

- Como un repositorio ampliamente utilizado para la publicación de proyectos Node.js de código abierto. En esta plataforma cualquiera puede publicar y compartir herramientas escritas en JavaScript.
- Como una herramienta de línea de comandos que ayuda a interactuar con plataformas en línea, como navegadores y servidores. Esto ayuda a instalar y desinstalar paquetes, gestión de versiones y gestión de dependencias necesarias para ejecutar un proyecto.

5.1.5. MongoDB

MongoDB es un sistema de base de datos NoSQL, orientado a documentos y de código abierto. En lugar de guardar los datos en tablas, MongoDB guarda estructuras de datos BSON (una especificación similar a JSON) con un esquema dinámico, haciendo que la integración de los datos en ciertas aplicaciones sea más fácil y rápida.

MongoDB es una base de datos adecuada para su uso en producción y con múltiples usos. Sus características principales son:

- Consultas ad hoc: MongoDB soporta la búsqueda por campos, consultas de rangos y expresiones regulares.
- Indexación: un documento de MongoDB puede tener sus campos indexados.
- Replicación: tiene un tipo de replicación primario-secundario, es decir, el primario tiene permisos para ejecutar comandos de escritura y lectura. El secundario duplica los datos del primario y solo tiene permisos para ejecutar comandos de lectura (copia de seguridad).
- Balanceo de carga: mediante el concepto de “sharding” se puede determinar la distribución de los datos en una colección.
- Almacenamiento de archivos: MongoDB puede ser usado como sistema de archivos.
- Agregación: MongoDB proporciona un framework de agregación que permite hacer operaciones similares al “GROUP BY” de SQL.
- Ejecución de JS en el lado del servidor: MongoDB posee la capacidad de realizar consultas utilizando JavaScript, resultando en que estas sean enviadas directamente a la base de datos para ser ejecutadas.

La finalidad de la utilización de MongoDB en el proyecto es de poder manejar datos del sistema y las relaciones entre ellos de una forma más sencilla y comprensible.

5.1.6. Express

Express es una infraestructura de aplicaciones web Node.js que proporciona una serie de características para estas aplicaciones.

Proporciona mecanismos para:

- Escritura de manejadores de peticiones con diferentes verbos HTTP en diferentes caminos URL, (rutas).
- Integración con motores de renderización de “vistas” para generar respuestas mediante la introducción de datos en plantillas.
- Establecimiento de ajustes de aplicaciones web como qué puerto usar para conectar.
- Añadir procesamiento de peticiones “middleware” adicional en cualquier punto de manejo de la petición.

5.1.7. Mongoose

Mongoose es una librería para Node.js que permite escribir consultas para una base de datos de MongoDB, con características como validaciones, construcción de consultas o *queries*, middlewares, conversión de tipos, etc..

Mongoose es un ODM, Object Document Mapper, esto quiere decir que aunque en la base de datos tenemos documentos, estos se representan como objetos (como esquemas tipados) cuando usamos Mongoose. La parte de *mapper* hace referencia a como un documento es representado por un objeto, y otro documento por otro objeto diferente.

Los ODM son similares a los ORM, que usamos en bases de datos relacionales, su objetivo como el de Mongoose es el de proveer una capa adicional de abstracción entre nuestro código y el motor de la base de datos, esta abstracción permite que librerías como Mongoose extiendan la funcionalidad del motor de la base de datos y nos permiten trabajar de manera más productiva y expresiva con la base de datos.

Mongoose se ha utilizado para la definición de los esquemas que se corresponden con los modelos definidos para el sistema.

5.1.8. Node-cron

El módulo node-cron es un pequeño programador de tareas en Javascript puro para Node.js basado en GNU crontab . Este módulo le permite programar tareas en Node.js utilizando la sintaxis crontab completa.

5.1.9. Bcrypt

Bcrypt es una función de hashing de contraseñas diseñado por Niels Provos y David Maxieres, basado en el cifrado de Blowfish. Se utiliza por defecto en sistemas OpenBSD y algunas distribuciones Linux y SUSE. Incorpora un valor llamado salt, que es un número aleatorio que se usará para generar el hash asociado a la contraseña, el salt se guardará junto a ella en la base de datos. Así se evita que dos passwords iguales generen el mismo hash y los problemas derivados de esto.

Se ha usado la librería *bcrypt* para el almacenamiento de las contraseñas de los usuarios, dado que se almacenen encriptadas.

5.1.10. Nodemailer

Nodemailer es un módulo para Node.js que permite el envío sencillo un email a un servidor SMTP en formato texto o HTML.

Se ha usado junto con el punto posterior para el envío de emails cuando los usuarios solicitan resetear su contraseña.

5.1.11. Handlebars

Handlebars sirve para generar HTML a partir de objetos con datos en formato JSON. Handlebars es sistema de plantillas en Javascript que permite crear y formatear código HTML sencillamente.

5.1.12. Multer

Multer es un middleware para Express y Node.js que ayuda a manipular el *multipart/form-data* cuando los usuarios registrados en la aplicación suben sus fotos.

5.1.13. JWT

JSON Web Token (JWT) es un estándar abierto (RFC-7519) basado en JSON para generar un token cuyo propósito sea enviar datos entre aplicaciones o servicios y garantizar que sean válidos, correctos y seguros.

Se ha usado JWT para manejar la autenticación en la aplicación web.

5.1.14. Visual Studio Code

Visual Studio Code es un editor de código fuente que permite trabajar con diversos lenguajes de programación, admite gestionar tus propios atajos de teclado, etc.

Es gratuito, de código abierto y proporciona una utilidad para descargar y manejar extensiones con las que podemos personalizar esta herramienta.

Visual Studio Code presenta diferentes extensiones que otorgan infinidad de ayudas, como colorear tabulaciones o recomendaciones de autocompletado. También hay extensiones que ayudan con respecto al lenguaje de programación que vayamos a utilizar, como por ejemplo para Python, C / C++, JavaScript, etc.

5.1.15. Postman

Postman es una extensión del navegador Google Chrome, que permite el envío de peticiones HTTP REST sin necesidad de desarrollar un cliente.

Se ha usado para realizar pruebas sobre el servicio REST del servidor de forma que se pudiese comprobar el correcto funcionamiento de la API.

5.2. Técnicas y herramientas usadas en el Frontend

5.2.1. Vue.js

Vue.js es un framework web de Javascript, de código abierto y con una arquitectura Model-View-ViewModel (MVVM), en español Modelo-Vista-ModelodeVista.

La librería central de Vue.js se centra en la capa de visualización, y es fácil de usar e integrar con otras librerías o proyectos ya existentes. Por otro lado, Vue es capaz de impulsar Single-Page Applications cuando se usa junto con herramientas modernas y otras librerías de apoyo.

Vue.js permite crear páginas web de forma rápida, sencilla, estructurada y estética. Usa el mismo gestor de paquetes que Node.js, NPM, comentado en el apartado anterior.

Una aplicación desarrollada con Vue.js está formada por componentes. Estos componentes se desglosan en tres partes:

- Template: Aquí se implementa el código para la estructuración del componente. Utiliza el lenguaje HTML.
- Script: Aquí se desarrolla el código que da la funcionalidad al componente. Usa Javascript.
- Style: Aquí se implanta el diseño del componente. Usa CSS.

La elección del Framework, Vue.js, como framework web para el desarrollo de la aplicación ha sido determinada por la facilidad de aprendizaje del mismo, ya que se partía desde 0 en cuanto a conocimiento en desarrollo web.

5.2.1.1. Vuex

Vuex es una librería para gestión del estado (State Management) de aplicaciones Vue.js. Sirve como un almacén centralizado para todos los componentes de una aplicación Vue, con reglas que garantizan que el estado se puede cambiar de manera predecible. En la figura 5.1 podemos observar el diagrama del patrón arquitectónico State Management.

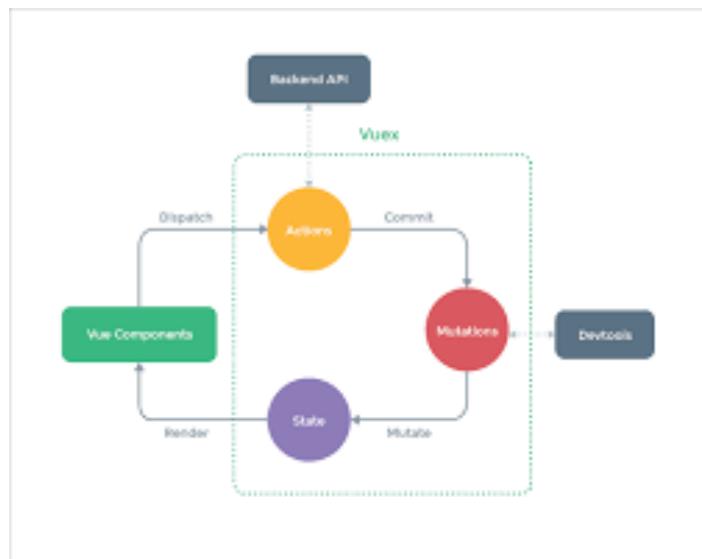


Figura 5.1: Patrón State Management

5.2.1.2. Vuetify

Vuetify es un framework que combina la potencia del popular Vue.js con la estética de Material Design. Permite acelerar el desarrollo de aplicaciones web complejas, incorporando una gran cantidad de componentes estéticos.

5.2.1.3. VueRouter

En Vue, vue-router es la biblioteca de enrutamiento oficial del lado del cliente que proporciona las herramientas necesarias para asignar los componentes de una aplicación a diferentes rutas de URL del navegador.

5.2.1.4. VueSocialSharing

VueSocialSharing es una librería para el Framework Vue.js cuyo objetivo es el de compartir datos en diferentes redes sociales.

5.2.1.5. Vuelidate - VeeValidate

Vuelidate y VeeValidate son ambas librerías que se usan para validar datos en formularios de componentes diseñados con el Framework Vue.js.

5.2.1.6. Vue-2 GoogleMaps

Esta librería permite sirviéndose de la API de Google Maps, la utilización de mapas, marcadores y barras de búsqueda de localización en el proyecto. También permite otras muchas funcionalidades derivadas del uso de estos elementos, como obtener coordenadas de localizaciones, autocompletar búsquedas de dirección, etc.

Su uso en la aplicación es mínimo y dedicado a mostrar la ubicación del centro.

5.2.1.7. VueFunctionalCalendar

Esta librería ha sido usada como DatePicker.

Un DatePicker es un selector de fecha que permite al usuario seleccionar una fecha de un calendario y / o hora de un rango de tiempo.

En este caso se ha usado para acceder a días concretos en las reservas de gimnasio y pádel.

5.2.2. HTML

HTML (Lenguaje de Marcas de Hipertexto, del inglés HyperText Markup Language) es el componente más básico de la Web. Define el significado y la estructura del contenido web. Además de HTML, generalmente se utilizan otras tecnologías para describir la apariencia/presentación de una página web (CSS) o la funcionalidad/comportamiento (JavaScript).

“Hipertexto” hace referencia a los enlaces que conectan páginas web entre sí, ya sea dentro de un único sitio web o entre sitios web. Los enlaces son un aspecto fundamental de la Web. Al subir contenido a Internet y vincularlo a las páginas creadas por otras personas, te conviertes en un participante activo en la «*World Wide Web*» (Red Informática Mundial).

HTML utiliza “marcas” para etiquetar texto, imágenes y otro contenido para mostrarlo en un navegador Web.

Un elemento HTML se distingue de otro texto en un documento mediante “etiqueta”.

5.2.3. CSS

Hojas de Estilo en Cascada (del inglés Cascading Style Sheets) o CSS es el lenguaje de estilos utilizado para describir la presentación de documentos HTML o XML. CSS describe como debe ser renderizado el elemento estructurado en la pantalla, en papel, en el habla o en otros medios.

CSS es uno de los lenguajes base de la Open Web y posee una especificación estandarizada por parte del W3C.

5.2.4. Axios

Axios es una librería JavaScript que puede ejecutarse en el navegador y que permite hacer simples y sencillas las operaciones como cliente HTTP, por lo que podremos configurar y realizar solicitudes a un servidor y recibiremos respuestas fáciles de procesar.

5.2.5. Sweetalert

Sweetalert es una librería de Javascript que permite mostrar alertas de una forma mucho mas estética.

Su uso reside en notificar al usuario en diferentes acciones.

5.2.6. PayPal

Se usó la API de PayPal para procesar los pagos de las matrículas de los usuarios en la aplicación.

La API de actualización instantánea de PayPal es una función de Pagos Exprés que permite a los clientes ver opciones de envío, opciones de seguro y los totales de impuestos al comienzo del proceso de pago.

5.3. Herramientas CASE

5.3.1. REM

REM (REquirements Management) es una herramienta experimental gratuita de Gestión de Requisitos diseñada para soportar la fase de Ingeniería de Requisitos de un proyecto de desarrollo software de acuerdo con la metodología definida en la Tesis Doctoral “Un Entorno Metodológico de Ingeniería de Requisitos para Sistemas de Información”, presentada por Amador Durán en septiembre de 2000.

5.3.2. Visual Paradigm

Visual Paradigm es una herramienta UML que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue, permite crear todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación.

Se ha usado para la generación de estos diagramas en la documentación.

5.3.3. Microsoft Project

Microsoft Project es un software diseñado por Microsoft y usado por millones de colaboradores, administradores y jefes de proyectos. Sirve para evaluar tareas y las secuencias en las que deben producirse, con el objetivo de estimar la duración del proyecto.

Se ha usado para la planificación temporal del proyecto.

5.3.4. EZEstimate

EZEstimate es una herramienta que permite realizar la estimación del esfuerzo de un proyecto.

5.4. Herramientas de control de versiones

5.4.1. Git

Git es un software de control de versiones diseñado por Linus Torvalds, pensando en la eficiencia, la confiabilidad y compatibilidad del mantenimiento de versiones de aplicaciones cuando estas tienen un gran número de archivos de código fuente.

Es utilizado para controlar los cambios que se llevan a cabo entre varios equipos, y es una herramienta de software libre bajo la licencia GNU v2. 37.

5.4.2. GitHub

GitHub es una plataforma online de desarrollo de código en modo colaborativo, que trabaja con el control de versiones Git.

5.5. Herramientas de despliegue

5.5.1. Git, GitHub y GitHub Actions

Git y GitHub han sido definidos en los puntos 5.4.1 y 5.4.2 respectivamente.

Github Actions es el cliente de integración continua ofrecido por el sitio web GitHub, que permite establecer una serie de instrucciones para realizar un despliegue, conocidas como workflow.

Estas serán ejecutadas cuando se dé una acción sobre una o un conjunto de ramas del repositorio, configurable en el propio workflow.

5.5.2. PM2

Se trata de un gestor de procesos *daemon* del sistema de GNU/Linux que permite gestionar y manejar de una forma bastante simple y sencilla aplicaciones y servicios a través de una interfaz CLI (Command-Line Interface en inglés). Su principal función es lanzar aplicaciones de Node.js en segundo plano, aunque también sirve para lanzar procesos realizando otro tipo de tareas como la ejecución de scripts o código en otros lenguajes de programación como Python.

Se trata de una librería gratuita que se suele utilizar en el desarrollo de aplicaciones web capaz de aguantar cantidades enormes de tráfico con un consumo de recursos reducido y con herramientas que permiten una monitorización remota de las aplicaciones.

PM2 puede controlar un conjunto de procesos listados, que se arrancarán nuevamente en caso de error, manteniéndose encendidos mientras la máquina permanezca encendida, lo que significa que en el caso que uno de ellos se termine por cualquier motivo, si se lanza una excepción de error en el programa que haga que el proceso finalice, PM2 lo iniciará de nuevo de forma automática.

También permite otras herramientas como la gestión de logs, el proceso de observación de archivos para rearanque automático cuando el código de la aplicación cambia, las herramientas de monitorización, las utilidades de despliegue mediante un fichero JSON, etc.

5.5.3. Certbot

Utilizado para habilitar HTTPS en un sitio web, es necesario tener un certificado (tipo de archivo) de una autoridad de certificación (su abreviatura es AC o CA). Entre los diversos ACs que existen, *Let's Encrypt* es una CA con un alto reconocimiento. Con él, se podrá usar el software que usa el protocolo ACME, este suele ser ejecutado en el servidor web para realizar dicha acción de forma más simple y sencilla.

Certbot es un cliente de automatización fácil de usar que puede obtener e implementar certificados SSL / TLS para un servidor web. Fue desarrollado por EFF y otras empresas como un cliente de *Let's Encrypt*, y anteriormente se conocía como “*Let's Encrypt Official Client*” o “*Let's Encrypt Python Client*”. Certbot también colaborará con cualquier otra CA que apoye el acuerdo ACME.

Se recomienda que la mayoría de las personas con derechos de acceso a shell utilicen el cliente de ACME Certbot para realizar la emisión e instalación automática de certificados sin tiempo de inactividad, además de proporcionar un modo experto para aquellos administradores que no necesiten configuración automática. Es fácil de usar, se ejecuta en muchos sistemas operativos y tiene una extensa y detallada documentación. En el caso del sistema de **GBApp**, Certbot ha sido utilizado para agregar e instalar los certificados SSL / TLS para el nombre de dominio del servidor adjunto de una forma fácil y flexible.

5.6. Herramientas de generación de documentación

5.6.1. JSdoc

JSDoc es una sintaxis para agregar documentación de la API al código fuente de JavaScript. JSDoc analiza el código Javascript y automáticamente genera una página HTML con la documentación en ella.

JSDoc se ha usado durante el desarrollo del Anexo V - Documentación técnica.

5.6.2. Vue Styleguidist

Vue Styleguidist es lo mismo que JSDoc pero para componentes del framework web Vue.js

Vue Styleguidist se ha usado durante el desarrollo del Anexo V - Documentación técnica.

5.6.3. Overleaf y L^AT_EX

Overleaf es un sitio web para escribir documentos en latex. Compila código latex de manera automática, mostrando los resultados de manera simultánea. No necesita de la instalación de paquetes. Posee una gran variedad de plantillas disponibles para editar.

L^AT_EX es un sistema de composición de textos, orientado especialmente a la creación de libros, documentos científicos y técnicos que contengan fórmulas matemáticas.

Se ha usado L^AT_EX para la documentación de la memoria y los anexos.

6. Aspectos relevantes del desarrollo

En esta sección se explican las partes más importantes y representativas de las distintas fases de desarrollo del proyecto.

6.1. Marco de trabajo

Durante el desarrollo del proyecto se ha seguido el Proyecto Unificado como marco de trabajo, en la figura 6.1 se observa un diagrama de este proceso, las características de este proceso son las siguientes:

- **Iterativo e incremental:** compuesto por 4 fases (que a su vez se componen por varias iteraciones cada una) explicadas posteriormente. El objetivo de las iteraciones es un incremento del producto desarrollado que añade o mejora las funcionalidades del sistema que se está desarrollando.
- **Dirigido por casos de uso:** los casos de uso son utilizados capturar los requisitos funcionales del sistema. Estos están presentes en todas las fases del proyecto.
- **Centrado en la arquitectura:** no existe un modelo único que cubra todos los aspectos del sistema, existen múltiples modelos y vistas que definen la arquitectura del software.

Las cuatro fases del proceso unificado son:

- **Inicio:** en esta fase se define el modelo de negocio.
- **Elaboración:** en esta fase se obtiene una visión refinada del proyecto que a realizar, definiendo la aplicación, lo que provoca la aparición de nuevos requisitos y obliga a reajustar las estimaciones.
- **Construcción:** en esta fase se lleva a cabo la construcción de proyecto.
- **Transición:** fase final del proyecto en la que debe tener la calidad para ser entregado al cliente.

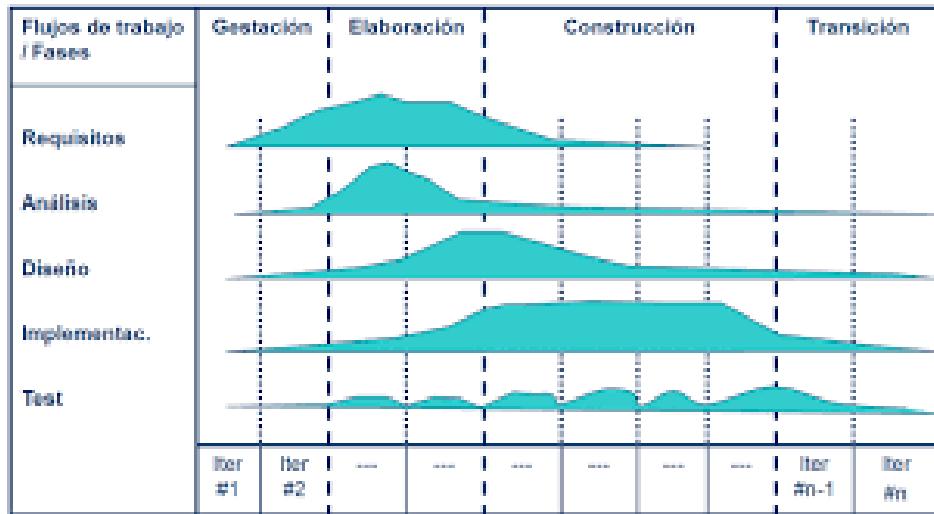


Figura 6.1: Proceso unificado

6.2. Estimación del esfuerzo

La estimación ha sido de las primeras tareas realizadas en el proyecto, con la estimación se puede hacer una idea de la duración total del mismo lo que es de ayuda para una posterior planificación temporal.

Para esta estimación se ha usado a métrica UCP (Use Case Points) o puntos de caso de uso. Esta métrica considera actores, escenarios y factores técnicos y de entorno.

Para conocer la estimación se ha hecho uso de la herramienta EZEstimate, en la figura 6.2 se puede observar los resultados.

Module	Summary																																																																																															
Gestión de Usuarios <input type="button" value="Add Module"/> <input type="button" value="Delete"/>	Total Modules 7 <input type="button" value="Excel Report"/> <input type="button" value="Generate Report"/> Use cases Simple 50 Average 7 Complex 0 Actors Simple 1 Average 0 Complex 3																																																																																															
Add Actor / Use case																																																																																																
Actor / Use case Name	Select Type Complexity																																																																																															
<input type="text"/>	<input type="button" value="Add"/>																																																																																															
Tech / Env Factors																																																																																																
<input type="button" value="Set Tech Factor"/> <input type="button" value="Set Env Factors"/>																																																																																																
Estimation Summary																																																																																																
UAW	10																																																																																															
UUCW	320																																																																																															
UUPC = UAW + UUCW	330																																																																																															
TFactor	41																																																																																															
EFactor	20																																																																																															
TCF = 0.6 + (.01*TFactor)	1,01																																																																																															
EF = 1.4 + (-0.03*EFactor)	0,8																																																																																															
UCP = UUCP*TCT*EF	266,64																																																																																															
Total Effort@ 3 Hrs/UCP	799,92																																																																																															
Use case / Actor List (Double click to delete)																																																																																																
<table border="1"> <thead> <tr> <th>ID</th> <th>Module</th> <th>Type</th> <th>Name</th> <th>Complexity</th> </tr> </thead> <tbody> <tr><td>1</td><td>Gestión de Usu...</td><td>Actor</td><td>Usuario Loguea...</td><td>Complex</td></tr> <tr><td>10</td><td>Gestión de Usu...</td><td>Usecase</td><td>Ver Información...</td><td>Simple</td></tr> <tr><td>11</td><td>Gestión de Acti...</td><td>Usecase</td><td>Listar Actividades</td><td>Simple</td></tr> <tr><td>12</td><td>Gestión de Acti...</td><td>Usecase</td><td>Buscar Activida...</td><td>Simple</td></tr> <tr><td>13</td><td>Gestión de Acti...</td><td>Usecase</td><td>Ver Actividad</td><td>Simple</td></tr> <tr><td>14</td><td>Gestión de Acti...</td><td>Usecase</td><td>Apuntar Actividad</td><td>Average</td></tr> <tr><td>15</td><td>Gestión de Acti...</td><td>Usecase</td><td>Desapuntar Act...</td><td>Simple</td></tr> <tr><td>16</td><td>Gestión de Acti...</td><td>Usecase</td><td>Escribir Reseña</td><td>Simple</td></tr> <tr><td>17</td><td>Gestión de Acti...</td><td>Usecase</td><td>Borrar Reseña</td><td>Simple</td></tr> <tr><td>18</td><td>Gestión de Acti...</td><td>Usecase</td><td>Votar Reseña</td><td>Simple</td></tr> <tr><td>19</td><td>Gestión de Acti...</td><td>Usecase</td><td>Compartir Redes</td><td>Simple</td></tr> <tr><td>2</td><td>Gestión de Usu...</td><td>Actor</td><td>Usuario Adminis...</td><td>Complex</td></tr> <tr><td>20</td><td>Gestión de Acti...</td><td>Usecase</td><td>Listar Reseñas ...</td><td>Simple</td></tr> <tr><td>21</td><td>Gestión de Acti...</td><td>Usecase</td><td>Borrar Actividad</td><td>Simple</td></tr> <tr><td>22</td><td>Gestión de Acti...</td><td>Usecase</td><td>Subir Imágenes...</td><td>Simple</td></tr> <tr><td>23</td><td>Gestión de Acti...</td><td>Usecase</td><td>Borrar Imagen ...</td><td>Simple</td></tr> <tr><td>24</td><td>Gestión de Acti...</td><td>Usecase</td><td>Crear Actividad</td><td>Average</td></tr> <tr><td>25</td><td>Gestión Datos</td><td>Usecase</td><td>Listar Datos Per</td><td>Simple</td></tr> </tbody> </table>		ID	Module	Type	Name	Complexity	1	Gestión de Usu...	Actor	Usuario Loguea...	Complex	10	Gestión de Usu...	Usecase	Ver Información...	Simple	11	Gestión de Acti...	Usecase	Listar Actividades	Simple	12	Gestión de Acti...	Usecase	Buscar Activida...	Simple	13	Gestión de Acti...	Usecase	Ver Actividad	Simple	14	Gestión de Acti...	Usecase	Apuntar Actividad	Average	15	Gestión de Acti...	Usecase	Desapuntar Act...	Simple	16	Gestión de Acti...	Usecase	Escribir Reseña	Simple	17	Gestión de Acti...	Usecase	Borrar Reseña	Simple	18	Gestión de Acti...	Usecase	Votar Reseña	Simple	19	Gestión de Acti...	Usecase	Compartir Redes	Simple	2	Gestión de Usu...	Actor	Usuario Adminis...	Complex	20	Gestión de Acti...	Usecase	Listar Reseñas ...	Simple	21	Gestión de Acti...	Usecase	Borrar Actividad	Simple	22	Gestión de Acti...	Usecase	Subir Imágenes...	Simple	23	Gestión de Acti...	Usecase	Borrar Imagen ...	Simple	24	Gestión de Acti...	Usecase	Crear Actividad	Average	25	Gestión Datos	Usecase	Listar Datos Per	Simple
ID	Module	Type	Name	Complexity																																																																																												
1	Gestión de Usu...	Actor	Usuario Loguea...	Complex																																																																																												
10	Gestión de Usu...	Usecase	Ver Información...	Simple																																																																																												
11	Gestión de Acti...	Usecase	Listar Actividades	Simple																																																																																												
12	Gestión de Acti...	Usecase	Buscar Activida...	Simple																																																																																												
13	Gestión de Acti...	Usecase	Ver Actividad	Simple																																																																																												
14	Gestión de Acti...	Usecase	Apuntar Actividad	Average																																																																																												
15	Gestión de Acti...	Usecase	Desapuntar Act...	Simple																																																																																												
16	Gestión de Acti...	Usecase	Escribir Reseña	Simple																																																																																												
17	Gestión de Acti...	Usecase	Borrar Reseña	Simple																																																																																												
18	Gestión de Acti...	Usecase	Votar Reseña	Simple																																																																																												
19	Gestión de Acti...	Usecase	Compartir Redes	Simple																																																																																												
2	Gestión de Usu...	Actor	Usuario Adminis...	Complex																																																																																												
20	Gestión de Acti...	Usecase	Listar Reseñas ...	Simple																																																																																												
21	Gestión de Acti...	Usecase	Borrar Actividad	Simple																																																																																												
22	Gestión de Acti...	Usecase	Subir Imágenes...	Simple																																																																																												
23	Gestión de Acti...	Usecase	Borrar Imagen ...	Simple																																																																																												
24	Gestión de Acti...	Usecase	Crear Actividad	Average																																																																																												
25	Gestión Datos	Usecase	Listar Datos Per	Simple																																																																																												

Figura 6.2: Estimación

6.3. Planificación temporal

Tras conocer la estimación de coste y esfuerzo se procedió a realizar la planificación temporal. Con el uso de la planificación temporal se podrá conocer las tareas necesarias para el desarrollo completo del proyecto y el tiempo necesario para completar cada una de ellas. Combinando el tiempo estimado de cada una de las tareas se podrá conocer la duración total del proyecto.

Como establece el Proceso Unificado, se basó en el siguiente esquema durante la realización del proyecto, llevando a cabo las siguientes etapas a lo largo de las fases y sus iteraciones:

- **Modelado de negocio:** investigación y búsqueda de recursos para disponer de los conocimientos necesarios para la realización del proyecto.
- **Requisitos:** se fijarán objetivos y requisitos necesarios para el proyecto.
- **Análisis:** se llevará a cabo un análisis exhaustivo de los requisitos de la etapa anterior.
- **Diseño:** Se determinará cómo serán y funcionarán todos los componentes del sistema.
- **Implementación:** se implementará el Frontend y el Backend.
- **Pruebas:** se realizarán las pruebas unitarias, de integración y completas del sistema.

En las siguientes figuras se muestra la parte inicio del diagrama de Gantt, así como las tareas, esto se hizo mediante la herramienta Microsoft Project.

■ Planificación GBApp	93 días?	lun 22/02/21	mié 30/06/21	
■ Inicio	17 días?	lun 22/02/21	mar 16/03/21	
■ Iteración 1	9 días?	lun 22/02/21	jue 04/03/21	
▷ Modelado de Negocio	1 día?	lun 22/02/21	lun 22/02/21	
■ Requisitos	3 días?	mar 23/02/21	jue 25/02/21	
Establecimiento de los Objetivos	1 día?	mar 23/02/21	mar 23/02/21	5 Alberto Martín
Definición de actores	0,5 días?	mié 24/02/21	mié 24/02/21	7 Alberto Martín[25%]
Establecimiento de los Requisitos funcionales	2 días?	mié 24/02/21	jue 25/02/21	7 Alberto Martín[50%]
Establecimiento de los Requisitos no funcionales	1 día?	mié 24/02/21	mié 24/02/21	7 Alberto Martín[25%]
■ Análisis	4 días?	vie 26/02/21	mié 03/03/21	
Modelo de dominio	1 día?	vie 26/02/21	vie 26/02/21	9 Alberto Martín
Estudio de mercado	3 días	lun 01/03/21	mié 03/03/21	12 Alberto Martín[50%]
Análisis de tecnologías	2 días	lun 01/03/21	mar 02/03/21	12 Alberto Martín[50%]
■ Diseño	1 día?	jue 04/03/21	jue 04/03/21	
Planificación del diseño	1 día?	jue 04/03/21	jue 04/03/21	13 Alberto Martín
<Hit Iteración 1>	0 días	jue 04/03/21	jue 04/03/21	3
■ Iteración 2	8 días?	vie 05/03/21	mar 16/03/21	17
■ Modelado de negocio	1 día?	vie 05/03/21	vie 05/03/21	
Planificación temporal	0,5 días?	vie 05/03/21	vie 05/03/21	Alberto Martín
Estimación del esfuerzo	0,5 días?	vie 05/03/21	vie 05/03/21	20 Alberto Martín
■ Requisitos	2 días?	lun 08/03/21	mar 09/03/21	
Establecimiento de requisitos de información	1 día?	lun 08/03/21	lun 08/03/21	21:20 Alberto Martín
Definición del modelo de casos de uso	1 día?	mar 09/03/21	mar 09/03/21	23 Alberto Martín

Figura 6.3: Tareas, primera parte

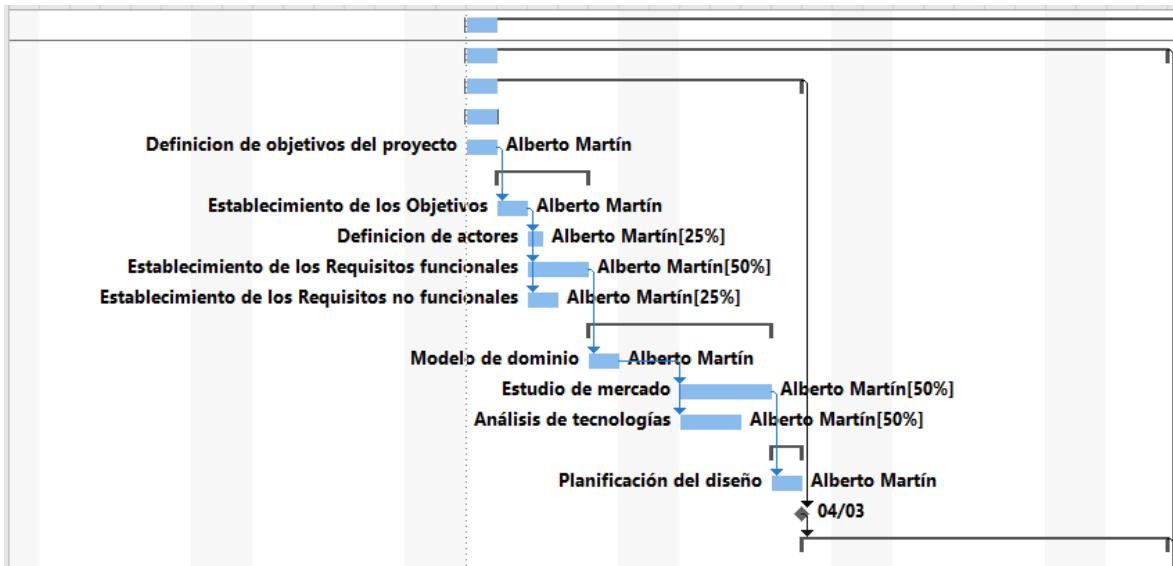


Figura 6.4: Diagrama de Gantt, primera parte

Si se desea conocer más información sobre la estimación del esfuerzo y la planificación temporal se puede consultar el *Anexo I - Plan de proyecto*.

6.4. Especificación de requisitos

En esta fase se han especificado los participantes, recogido los objetivos y catalogados los requisitos necesarios del sistema. Todo esto teniendo en cuenta la metodología de Durán y Bernárdez.

Se han utilizado las herramientas REM y Visual Paradigm. Si se quiere saber más se puede consultar el *Anexo II - Especificación de requisitos*.

La sección sigue la siguiente estructura:

6.4.1. Participantes

Se listan los participantes en este proyecto.

6.4.2. Objetivos del sistema

Los objetivos definidos son los siguientes:

- Gestión de usuarios.
- Gestión de datos de usuario.
- Gestión de matrículas.

- Gestión de actividades.
- Gestión de reseñas.
- Gestión de gimnasio y pádel.

6.4.3. Requisitos de información

Los requisitos de información indican los datos que el sistema debe almacenar.

6.4.4. Requisitos de restricción de información

Los requisitos de restricción de información reflejan las condiciones que debe cumplir la información del sistema.

6.4.5. Requisitos funcionales

Los requisitos funcionales definen como se comporta el sistema en determinadas situaciones. Lo primero que se ha realizado ha sido un diagrama de paquetes.

Previo a esto se realizó un diagrama de paquetes, en la figura 6.5 se observa el mismo.

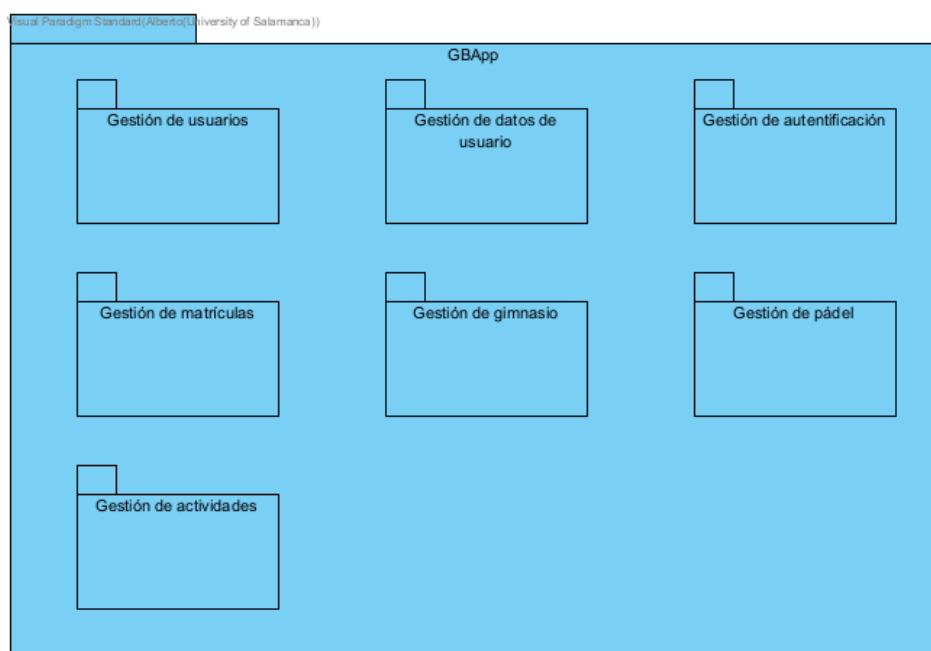


Figura 6.5: Diagrama de paquetes

Y posteriormente se definieron los actores que actúan con el sistema. En la figura 6.6 se observa su jerarquía:

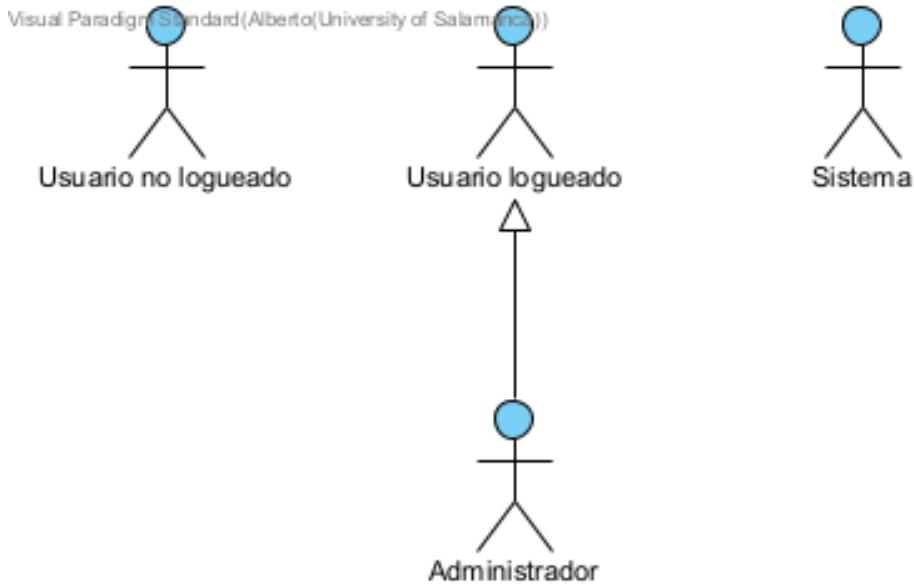


Figura 6.6: Jerarquía de los actores

Por último, se definen los casos de uso del sistema. Se han realizado diagramas de casos de uso por paquetes y, posteriormente, se ha especificado individualmente todos los casos de uso.

A continuación, en la figura 6.7 se muestra un ejemplo:

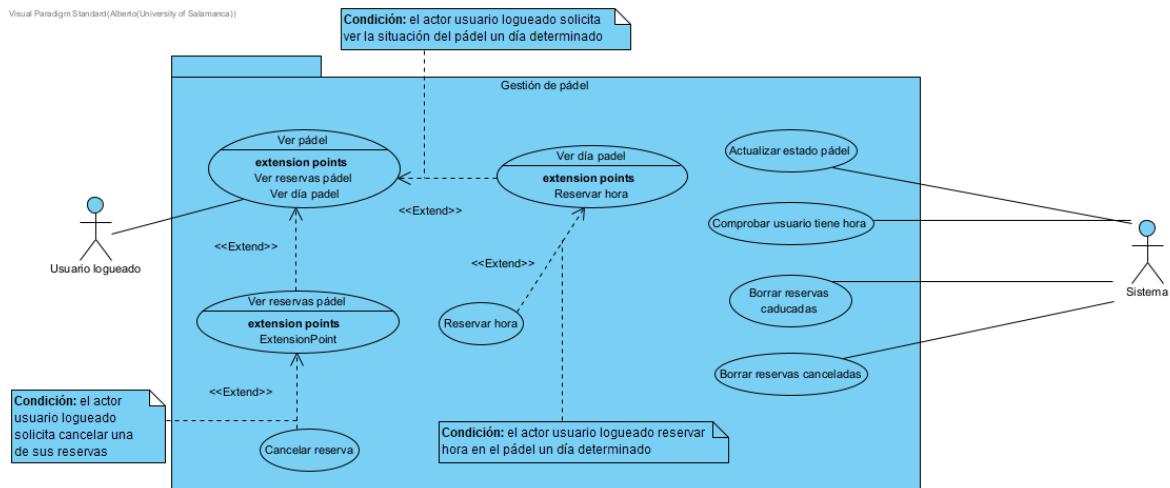


Figura 6.7: Diagrama de casos de uso, paquete gestión de pádel

En la figura 6.8 se observa la tabla de uno de los casos de uso.

UC-0038	Reservar hora pádel	
Versión	1.0	
Autores	<ul style="list-style-type: none"> • Alberto Martín Peralejo 	
Fuentes	<ul style="list-style-type: none"> • André Filipe Sales Mendes • Gabriel Villarrubia González • Juan Francisco De Paz Santana 	
Dependencias	<ul style="list-style-type: none"> • [OBJ-0005] Gestión de gimnasio y pádel • [IRQ-0009] Información sobre las reservas de pádel 	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando <i>el usuario solicite hacer una reserva en una hora en el pádel</i> .	
Precondición	Estar en la vista del día en concreto.	
Secuencia normal	Paso	Acción
	1	El sistema <i>le muestra al usuario una lista de las horas disponibles para reservar</i> .
	2	El actor Usuario logeado (ACT-0002) <i>solicita en reservar en una hora concreta</i> .
	3	El sistema <i>comprueba la reserva</i> .
	4	El sistema <i>registra la reserva, notifica al usuario y finaliza el caso de uso</i> .
Postcondición		
Excepciones	Paso	Acción
	1	Si <i>el día en concreto es incorrecto (de un mes diferente al actual o una fecha anterior a la actual)</i> , el sistema <i>bloquea las reservas</i> , a continuación este caso de uso <i>queda sin efecto</i>
	2	Si <i>la hora no está disponible</i> , el sistema <i>bloquea el botón</i> , a continuación este caso de uso <i>queda sin efecto</i>
	3	Si <i>la fecha es incorrecta</i> , el sistema <i>notifica al usuario</i> , a continuación este caso de uso <i>queda sin efecto</i>
	3	Si <i>el usuario no ha pagado su matrícula</i> , el sistema <i>notifica al usuario</i> , a continuación este caso de uso <i>queda sin efecto</i>
	3	Si <i>la hora ya está reservada</i> , el sistema <i>bloquea el botón y notifica al usuario</i> , a continuación este caso de uso <i>queda sin efecto</i>
	3	Si <i>si el usuario ya había reservado en el gimnasio o en el pádel a esa hora</i> , el sistema <i>notifica al usuario</i> , a continuación este caso de uso <i>queda sin efecto</i>
Rendimiento	Paso	Tiempo máximo
	-	-
Frecuencia esperada		
Importancia	vital	
Urgencia	inmediatamente	
Estado	validado	
Estabilidad	alta	
Comentarios	Ninguno	

Figura 6.8: UC-0038 Reservar hora pádel

6.4.6. Requisitos no funcionales

Los requisitos no funcionales definen restricciones que se deben cumplir en el diseño e implementación.

6.5. Análisis de requisitos

Esta fase consistirá en la realización de un análisis exhaustivo de los requisitos del apartado anterior. Se divide en los apartados que se muestran a continuación.

Más información referente a esta fase puede encontrarse en el *Anexo III - Análisis de requisitos*. Para la realización de este documento se ha hecho uso de la herramienta Visual Paradigm.

6.5.1. Modelo de dominio

Un modelo de dominio en la resolución de problemas e ingeniería de software, es un modelo conceptual de todos los temas relacionados con un problema específico. En él se describen las distintas entidades, sus atributos, papeles y relaciones, además de las restricciones que rigen el dominio del problema.

Para poder representar el modelo de dominio del sistema, se ha utilizado un diagrama de clases como el que se observa en la figura 6.9:

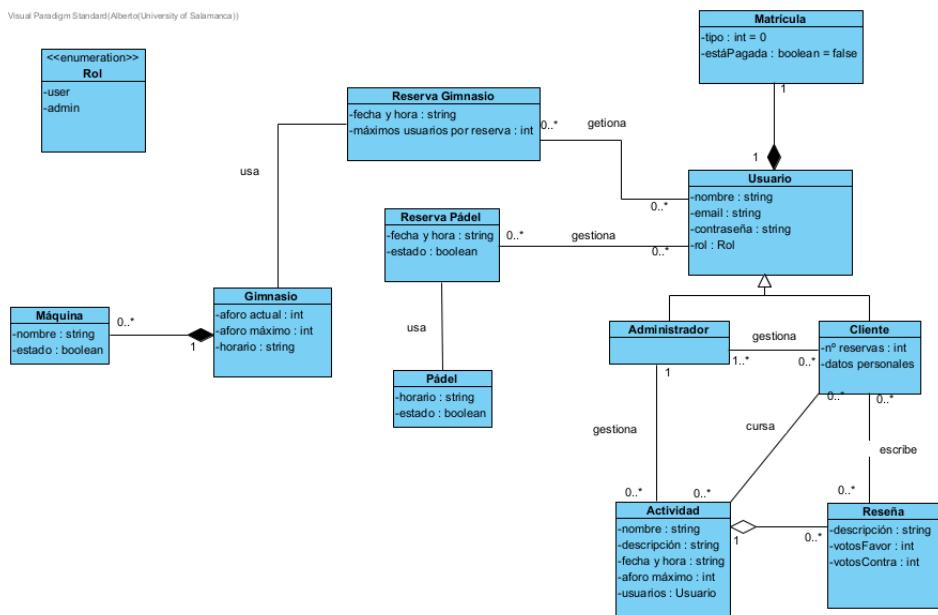


Figura 6.9: Modelo de dominio

6.5.2. Realización de casos de uso - análisis

En este apartado se han detallado de forma generalizada los pasos que debería tener un caso de uso usando diagramas de secuencia. Se puede observar en la figura 6.10 un ejemplo de estos:

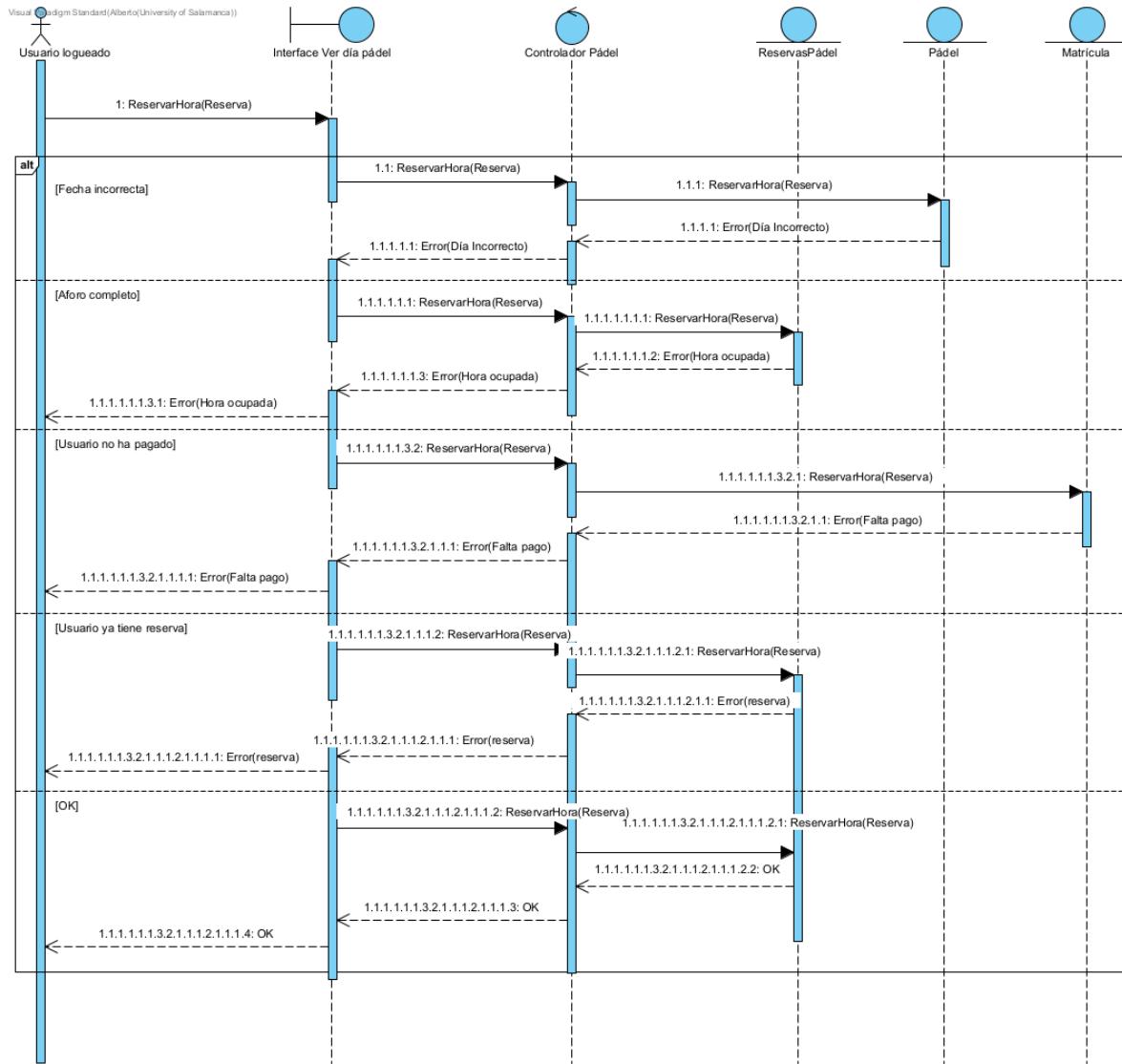


Figura 6.10: Diagrama de secuencia UC-0038 Reservar hora pádel

6.5.3. Clases de análisis

En esta sección se han realizado los diagramas de comunicación que relacionaban los elementos surgidos durante el desarrollo de los diagramas de secuencia del punto anterior.

6.5.4. Vista arquitectónica del modelo de análisis

En la vista de arquitectura del modelo de análisis se han catalogado las clases resultantes en los puntos anteriores y han sido organizadas según el patrón MVC (Modelo Vista Controlador); aunque finalmente, como se observará en la fase de diseño del sistema, se ha acabado decantando por el patrón MVVM.

6.6. Diseño del sistema

Este apartado se centra en el dominio de la solución. Por lo tanto los nombres de clases, métodos y atributos que se mostrarán a lo largo de este documento serán una aproximación cercana a la que nos encontramos en la implementación, es decir, se trata de una aproximación cercana a la implementación.

Para la generación de los diagramas mostrados en este documento se ha utilizado la herramienta UML Visual Paradigm.

La información detallada de la fase de diseño del sistema se puede encontrar en el documento **Anexo IV – Diseño del Sistema Software**.

6.6.1. Patrones arquitectónicos

6.6.1.1. Modelo-vista-modelo de vista (MVVM)

MVVM se deriva del patrón clásico MVC (Modelo-Vista-Controlador). El patrón modelo–vista–modelo de vista (en inglés, model–view–viewmodel, abreviado MVVM) se caracteriza por tratar de desacoplar lo máximo posible la interfaz de usuario de la lógica de la aplicación, facilitando las pruebas, mantenimiento y la escalabilidad de los proyectos.

En la figura 6.11 se puede observar este patrón.

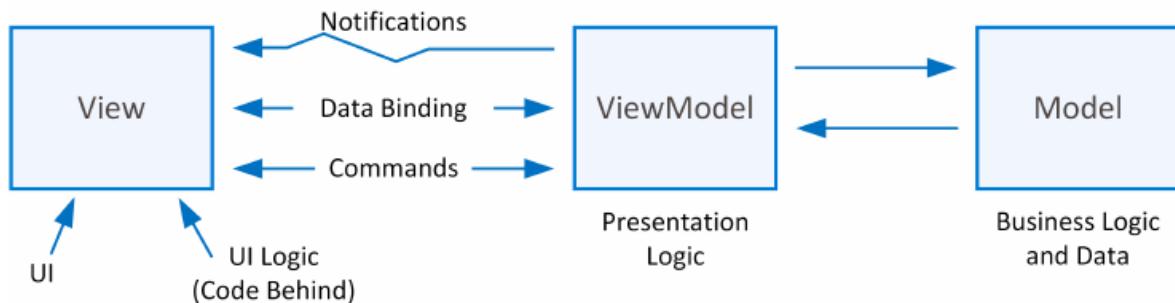


Figura 6.11: Patrón MVVM

Estos son los tres componentes que forman el patrón MVVM:

- **Modelo:** Representa la capa de datos y/o la lógica de negocio, también denominado como el objeto del dominio. El modelo contiene la información, pero nunca las acciones o servicios que la manipulan. En ningún caso tiene dependencia alguna con la vista.
- **Vista:** La misión de la vista es representar la información a través de los elementos visuales que la componen. Las vistas en MVVM son activas, contienen comportamientos, eventos y enlaces a datos que, en cierta manera, necesitan tener conocimiento del modelo subyacente.
- **Modelo de vista:** El modelo de vista es un actor intermediario entre el modelo y la vista, contiene toda la lógica de presentación y se comporta como una abstracción de la interfaz. La comunicación entre la vista y el viewmodel se realiza por medio los enlaces de datos (binders).

El modelo de vista también se encarga de realizar la comunicación entre los modelos y las vistas.

6.6.1.2. Singleton

Singleton es un patrón de diseño creacional que nos permite asegurarnos de que una clase tenga una única instancia, a la vez que proporciona un punto de acceso global a dicha instancia. En la figura 6.12 se puede observar este patrón.

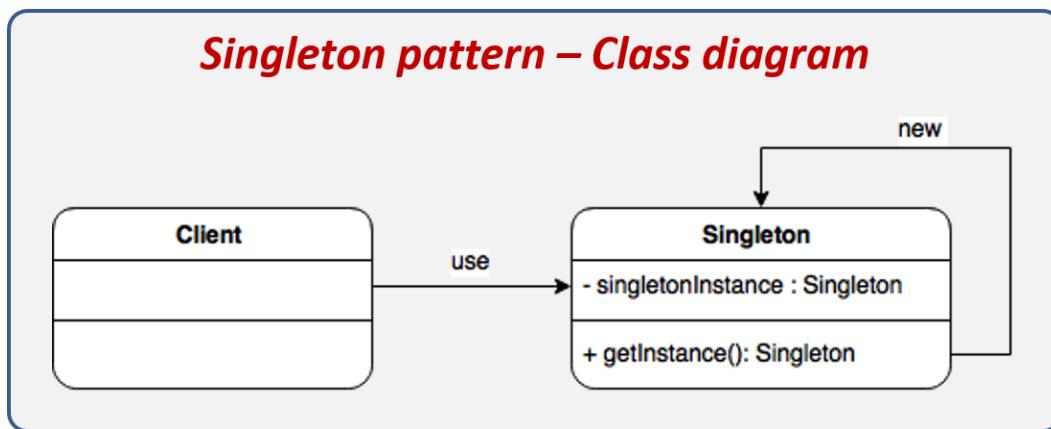


Figura 6.12: Patrón Singleton

Se utiliza de manera conjunta junto al patrón State management a través de la extensión Vuex de Vue.js. La instancia del objeto singleton creado es obtenida por los componentes a través del DOM.

6.6.1.3. State management

Patrón de administración del estado o State management es un patrón de comportamiento que permite establecer el acceso a objetos a través de Stores, de tal modo

que distintos componentes puedan acceder a un estado compartido y así evitar información duplicada o tener que utilizar el árbol de componentes para la transmisión de datos.

Se ha utilizado este patrón a través de la extensión Vuex para vue.js, diseñada para este propósito. Diseñando un único store que carga la información necesaria de la base de datos.

6.6.2. Subsistemas de diseño

En el siguiente diagrama se ha hecho un desglose del sistema en subpaquetes de la aplicación. El diagrama también muestra la relación entre ellos. En el apartado 2.2 del *Anexo IV - Diseño del sistema software* se definen estos subpaquetes. Aquí, los podemos observar en la siguiente figura 6.13:

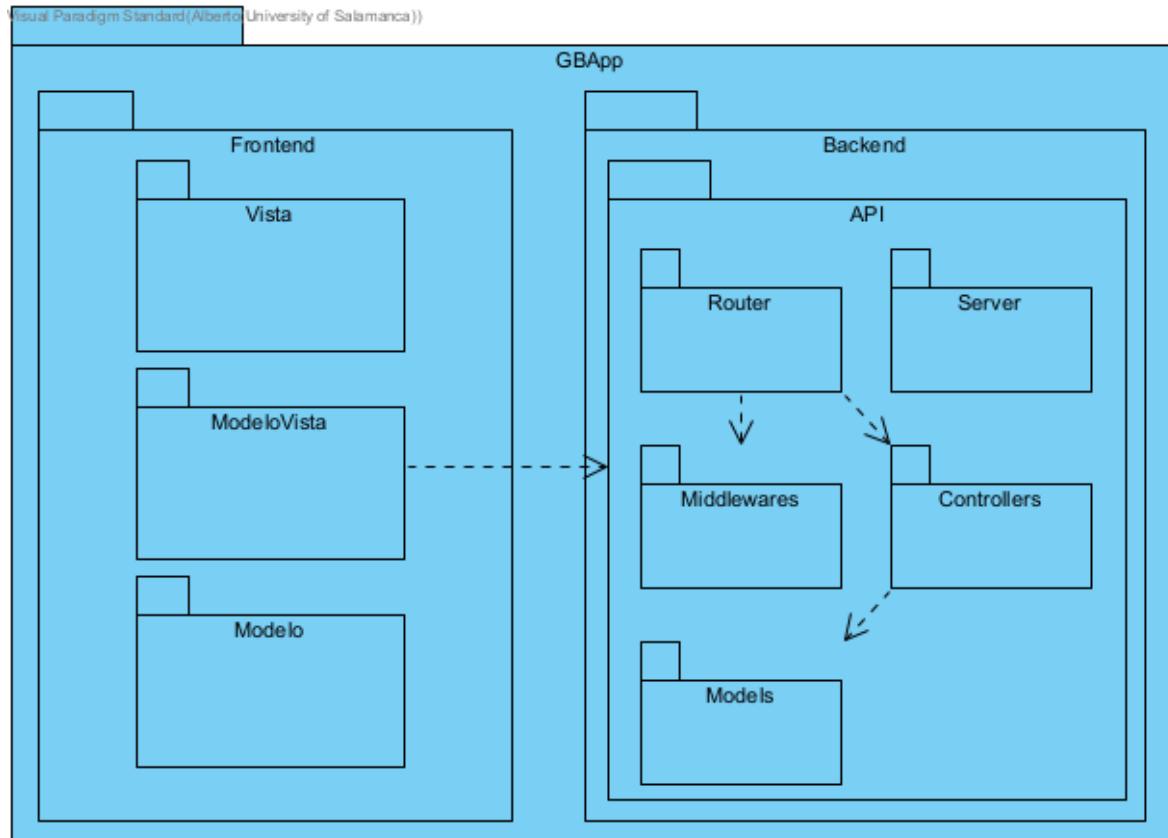


Figura 6.13: Subsistemas de diseño

6.6.3. Clases de diseño

En esta sección se definen todos los contenidos de cada capa del patrón MVVM, así como las clases de cada paquete y los métodos de dicha clase.

En la figura 6.14 se puede observar el Modelo de Vista del Frontend.

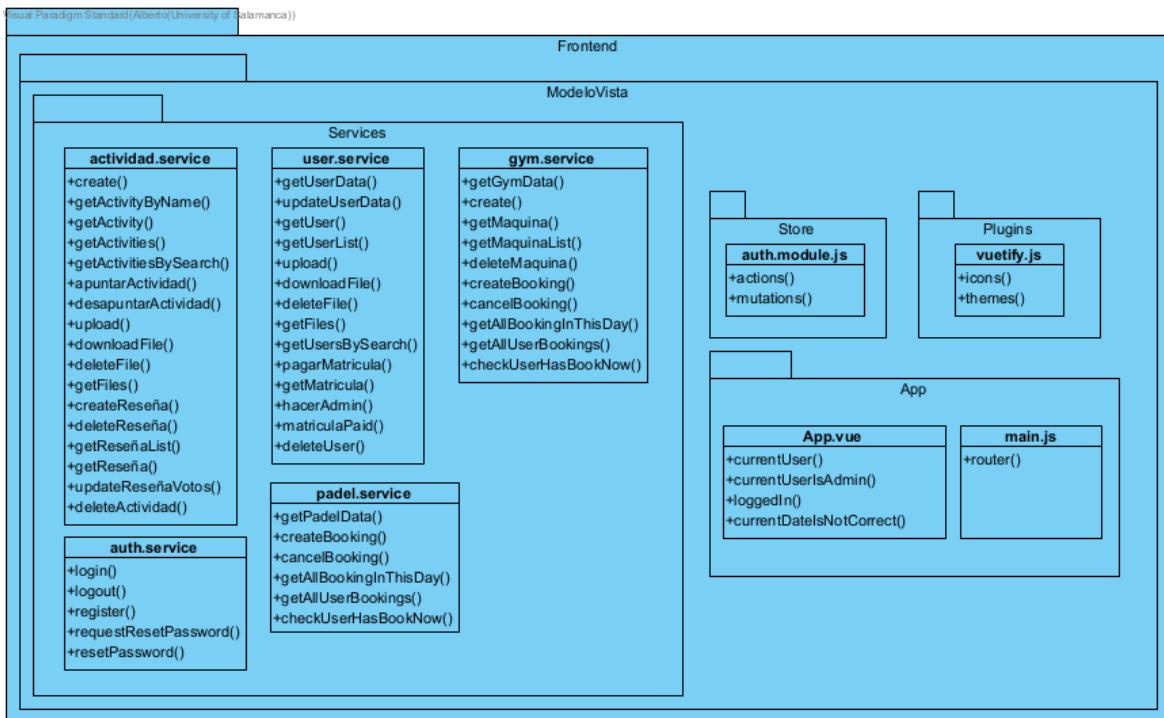


Figura 6.14: Frontend - Modelo de Vista

En la figura 6.15 se puede observar los controladores del Backend.

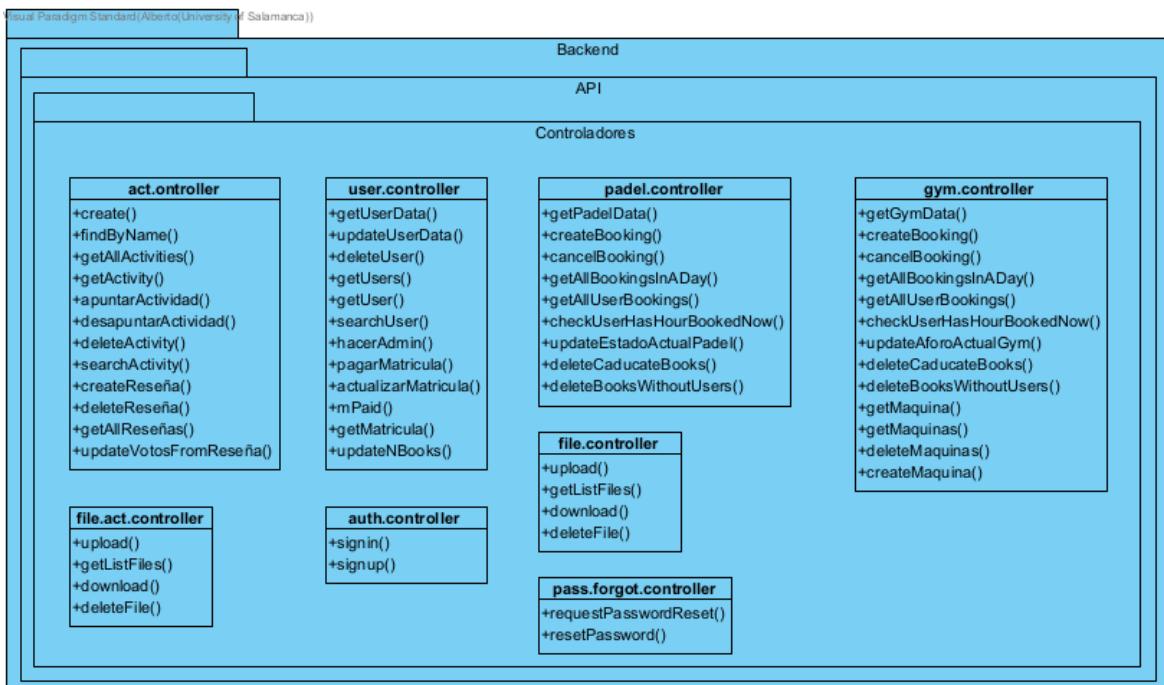


Figura 6.15: Backend - Controladores

6.6.4. Vista arquitectónica del modelo de diseño

En la vista arquitectónica del modelo de diseño se han ordenado los subpaquetes del sistema según la capa del patrón MVVM a la que pertenecen. En la figura 6.16 se ve la vista arquitectónica.

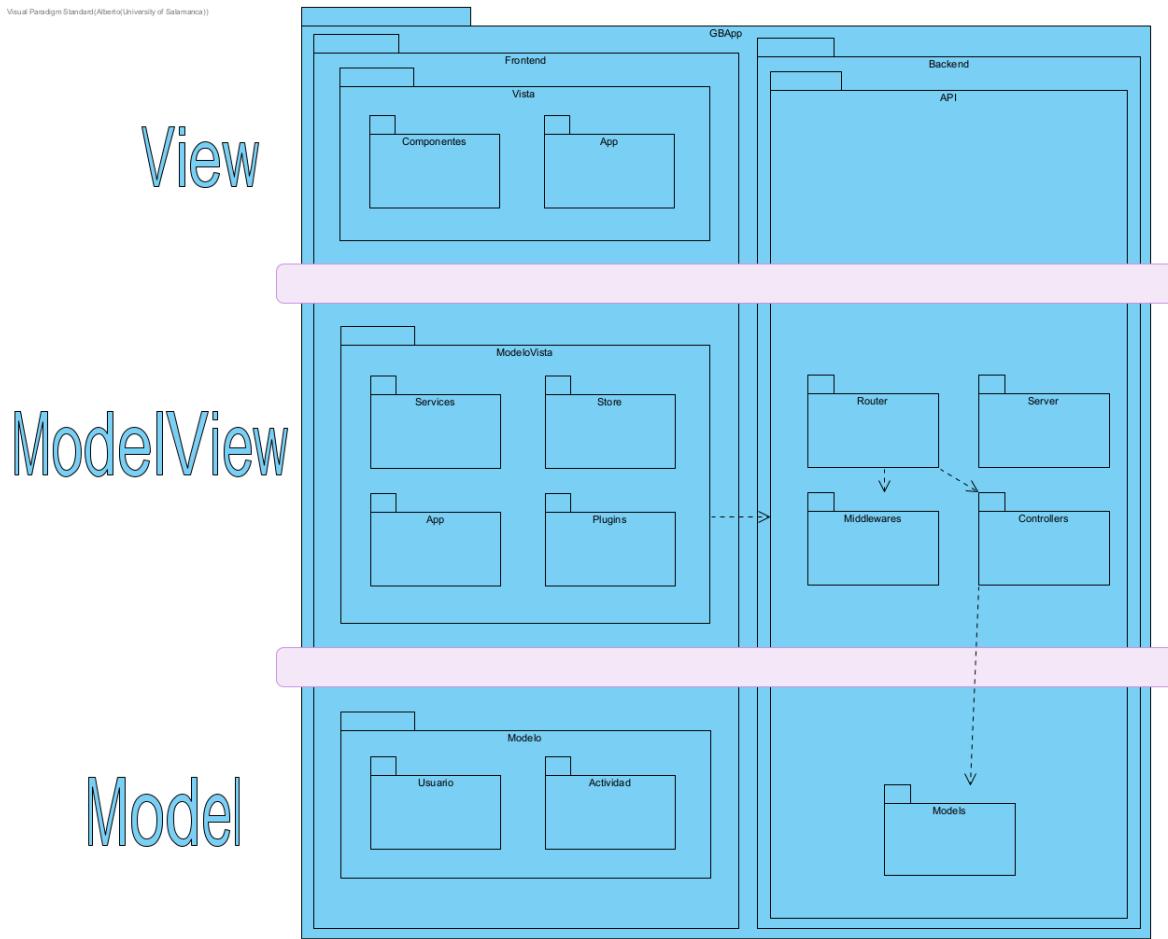


Figura 6.16: Vista arquitectónica

6.6.5. Realización de casos de uso - diseño

La finalidad de esta sección es detallar el intercambio de mensajes entre objetos mediante diagramas de secuencia. Estos diagramas han sido refinados a partir de los diagramas del *Anexo III - Análisis de requisitos*. Los nombres de las Vistas de los diagramas coincidirán con los nombres de las Vistas al momento de la implementación, así como los métodos y demás clases. En la figura 6.17 se puede ver un ejemplo de los diagramas de secuencia.

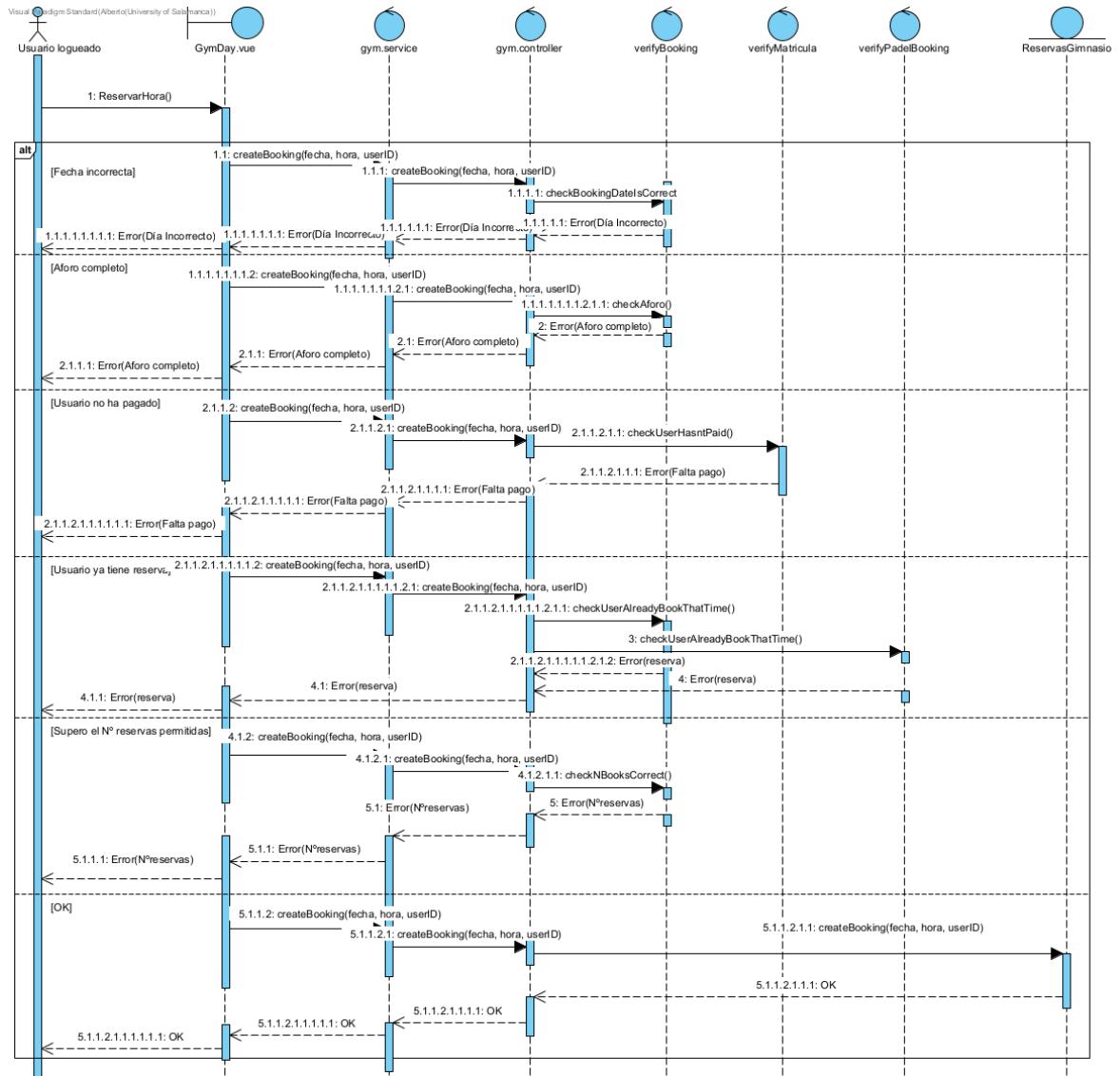


Figura 6.17: Diagrama de secuencia UC-0033 Reservar hora gimnasio

6.6.6. Diseño de bases de datos

Se ha utilizado la base de datos NoSQL, MongoDB. Esta base de datos trabaja en colecciones y estructuras de datos BSON (una especificación similar a JSON) con un esquema dinámico, haciendo que la integración de los datos en ciertas aplicaciones sea más fácil y rápida. En la figura 6.18 se puede observar la base de datos.

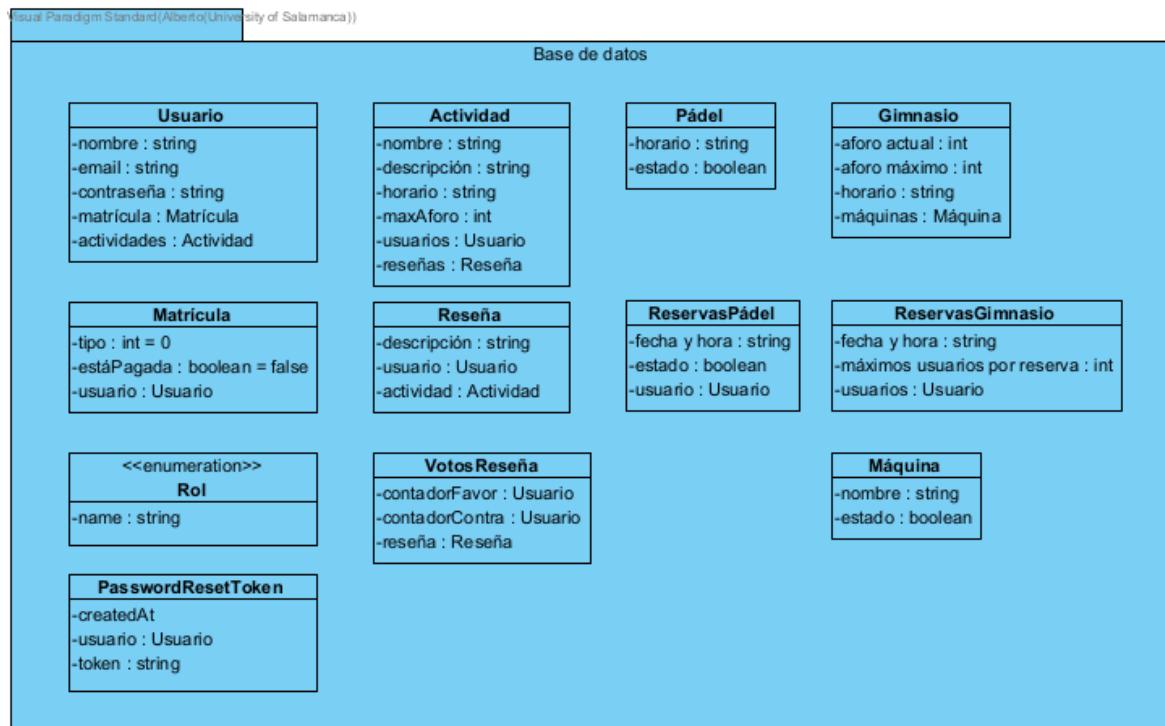


Figura 6.18: Base de datos

En el *Anexo IV - Diseño del sistema* software se explican las relaciones entre las diferentes colecciones.

6.6.7. Modelo de despliegue

El modelo de despliegue muestra el resultado del despliegue de los nodos desarrollados y la comunicación entre ellos. En la figura 6.19 se puede observar el diagrama de despliegue.

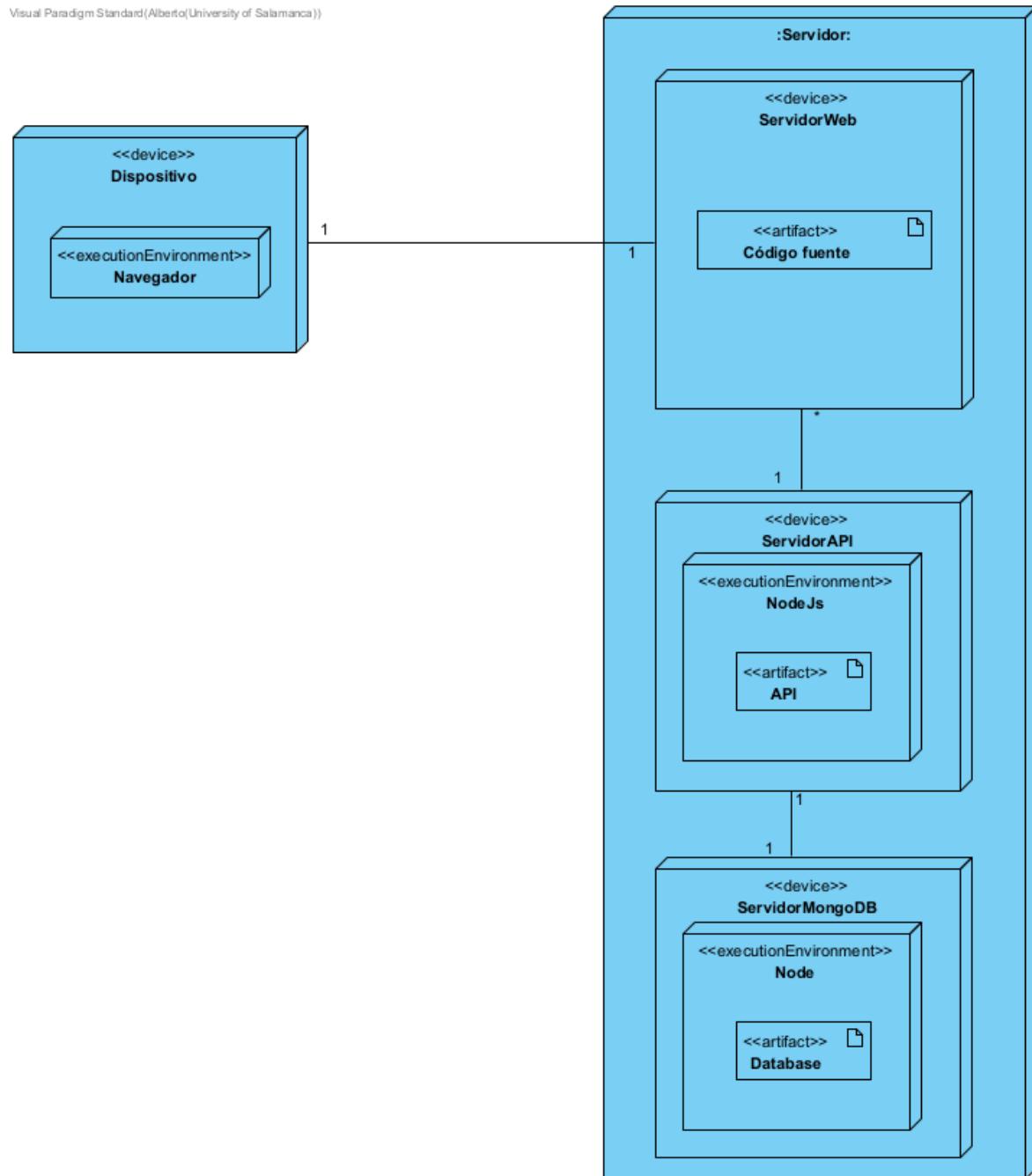


Figura 6.19: Diagrama de despliegue

Hay 2 nodos principales:

- **Dispositivo:** Dispositivo donde se encuentra un navegador. Mediante este navegador se podrá acceder a la aplicación donde los usuarios podrán realizar las distintas acciones del sistema detalladas anteriormente.
- **Servidor:** Que se dividie a su vez en tres nodos:
 - **Servidor Web:** Servidor en el que se almacenan los datos necesarios para desplegar la aplicación web en el cliente web. Se comunicará con el servidor API para acceder a la base de datos y realizar la autenticación de usuarios
 - **Servidor API:** Servidor que, mediante la API, prestará soporte a los nodos dispositivo para el manejo de la información del sistema. Cabe destacar que el entorno de ejecución se trata de Node.js.
 - **Servidor MongoDB:** Servidor que actúa como base de datos y donde se almacenará la información necesaria del sistema. La base de datos será MongoDB.

6.7. Implementación

Tras finalizar la fase de diseño se ha comenzado la codificación de la aplicación. En esta fase se han utilizado y puesto en práctica la técnicas y herramientas especificadas en el apartado 5 de este mismo documento.

La implementación se divide en dos partes que se han ido realizando de manera simultánea:

6.7.1. Frontend

Previo a la codificación de los componentes de Vue.js se realizaron unos bocetos usando la herramienta AdobeXD. El objetivo de estos bocetos era tener una idea de como iban a ser las vistas antes de comenzar a realizarlas. Aquí se muestra un ejemplo de ello en la figura 6.20.



Figura 6.20: Boceto de la vista del perfil

Tras la realización de estos bocetos se comenzó el desarrollo del Frontend de la aplicación web, para ello se ha hecho uso de el framework web Vue.js además de todas las herramientas mostradas en el apartado 5.2.

Para el diseño de las vistas se ha hecho uso de Vuetify, un framework que combina la potencia del popular VueJs con la estética de Material Design.

En la figura 6.21 se puede mirar el resultado final de la pantalla del perfil.

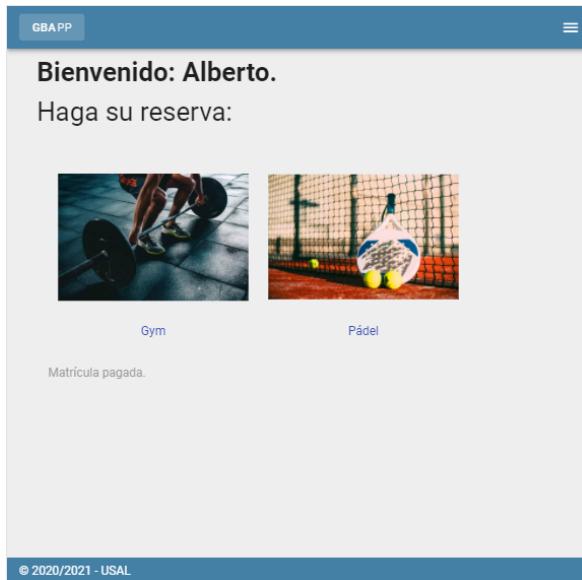


Figura 6.21: Vista final del perfil

Una vez obtenido las herramientas necesarias se comenzó el desarrollo de los diferentes paquetes del sistema. Estos paquetes contienen las vistas necesarias para que los clientes interactúen con el sistema.

Para ver el código se debe acudir al *Anexo V - Documentación técnica*.

6.7.2. Backend

Para el desarrollo del Backend se han hecho uso de todas las herramientas que se detallan en el apartado 5.1.

Cabe destacar que durante las primeras etapas de la implementación se ha usado con frecuencia una de ellas. La herramienta Postman, usada para el testeo.

Se ha desarrollado un API REST mediante el uso de Node.js, la librería express y MongoDB.

Este API REST contiene varios controladores y “middlewares” que responden a las peticiones realizadas desde el Frontend.

Para ver el código se debe acudir al *Anexo V - Documentación técnica*.

6.7.3. Hosting

Durante la mayor parte del proceso de implementación la aplicación se ha desplegado en local por la rapidez de actualización de los cambios en el código sobre la aplicación alojada.

En las últimas etapas de la implementación se ha desplegado la aplicación en un servidor alquilado de la empresa Digital Ocean y la ayuda de las herramientas de despliegue que se detallan en el apartado 5.5.

6.7.4. Documentación técnica

Si se desea conocer más información, se puede consultar el *Anexo V - Documentación técnica*.

6.8. Pruebas

Las pruebas son una parte fundamental en el desarrollo de proyectos puesto que permiten comprobar que la funcionalidad del sistema es correcta y cumple con los objetivos marcados.

Con el fin de probar la aplicación y garantizar su correcto funcionamiento se han realizado pruebas al final y durante el desarrollo, que se clasifican en dos tipos:

- **Pruebas unitarias:** estas pruebas comenzaron una vez finalizado el proceso de implementación del paquete o módulo desarrollado. Se hicieron sobre cada componente desarrollado de forma individual.
- **Pruebas de integración:** cuando acabaron las pruebas unitarias comenzaron las pruebas de integración, en estas se ha comprobado que el funcionamiento de los componentes que forman el sistema sea correcto.

6.9. Funcionalidad de la aplicación

En esta sección se explica de forma resumida la funcionalidad del sistema. Para ello, se detallará teniendo en cuenta los actores del sistema.

Si se desea ver esta información de forma más detallada, se puede consultar el *Anexo VI -Manual de usuario* donde se explican todas las tareas que puede realizar el usuario dentro del sistema.

6.9.1. Usuario no logueado

El usuario no logueado tendrá pocas acciones que realizar, debido a que para acceder a la funcionalidad del sistema se requiere estar registrado en el mismo.

El usuario no logueado podrá:

- **Iniciar sesión.**
- **Registrarse.**
- **Solicitar reseteo de contraseña y resetear contraseña (una vez registrado en el sistema).**
- **Ver información del centro.**
- **Ver ubicación del centro.**
- **Ver matrículas disponibles.**

En la figura 6.22 se muestra la pantalla de Inicio de sesión:

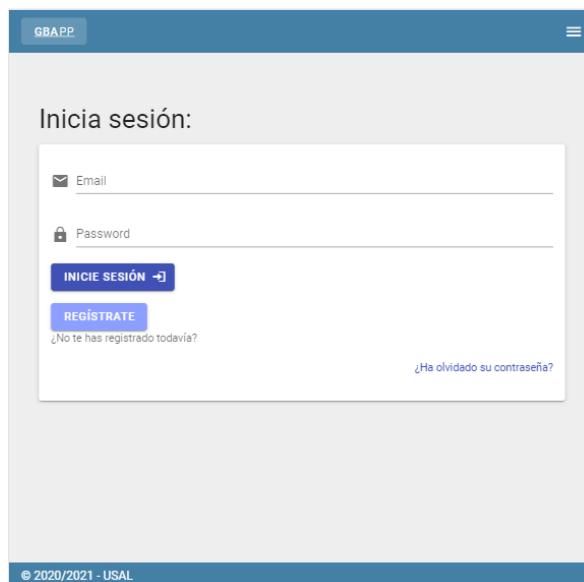


Figura 6.22: Pantalla Iniciar sesión

En esta pantalla se le presenta al usuario un formulario a llenar para iniciar sesión. También tiene acceso a diferentes pantallas si pulsa diferentes botones o accede al navegador.

En la figura 6.23 se muestra el navegador de este actor:

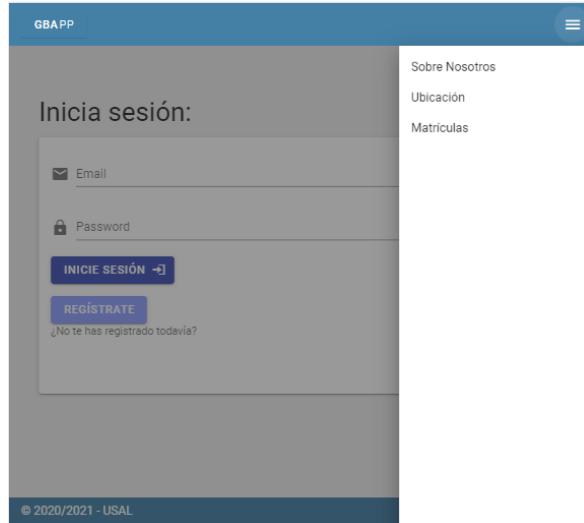


Figura 6.23: Navegador de Usuario no logueado

Este es el navegador disponible cuando el usuario todavía no ha iniciado sesión. A través del mismo podrá acceder a diferentes pantallas.

6.9.2. Usuario logueado

El usuario logueado tendrá acceso a toda la funcionalidad del sistema.

Entre otras cosas podrá:

- **Gestionar su matrícula.**
- **Apuntarse a diferentes actividades acorde a su matrícula y dar opinión sobre ellas.**
- **Gestionar sus datos.**
- **Gestionar sus reservas de gimnasio y pádel.**

A continuación se muestran algunos ejemplos de reservas y actividades, en la figura 6.24 se muestra el navegador de este actor:

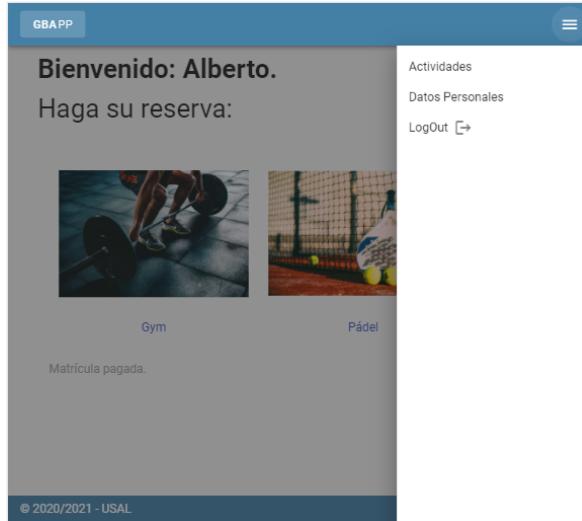


Figura 6.24: Navegador usuario logueado

Cuando el usuario se loguea aparece en la pantalla de perfil. Este es el aspecto del navegador cuando el usuario se loguea. Como se puede observar cambian las opciones y las pantallas a las que acceder.

En la figura 6.25 se muestra la pantalla de Actividades:



Figura 6.25: Pantalla Actividades

Una de las pantallas que se accede desde el navegador es la de actividades. Esta pantalla muestra la lista de actividades registradas en el sistema. El sistema ofrece además un buscador de actividades.

En las figuras 6.26 - 6.29 se muestra la pantalla de la actividad Kick Boxing:



Figura 6.26: Pantalla Actividad Kick Boxing 1

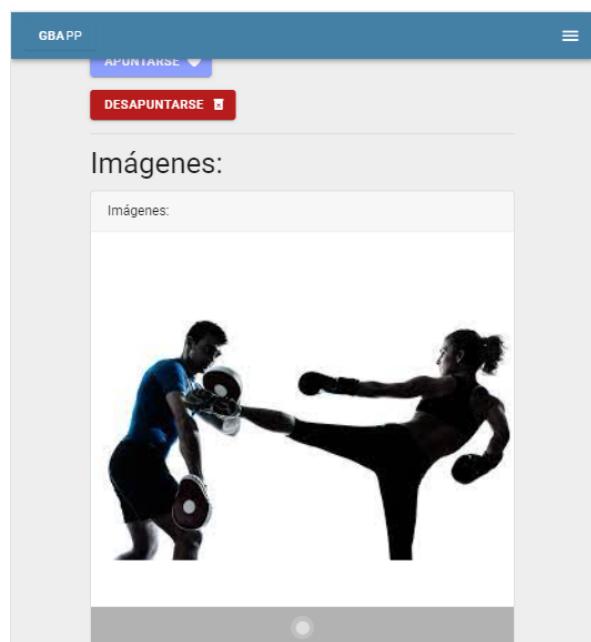


Figura 6.27: Pantalla Actividad Kick Boxing 2

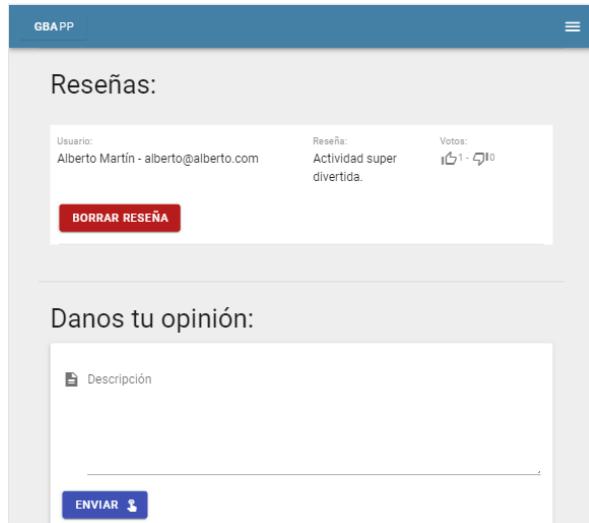


Figura 6.28: Pantalla Actividad Kick Boxing 3

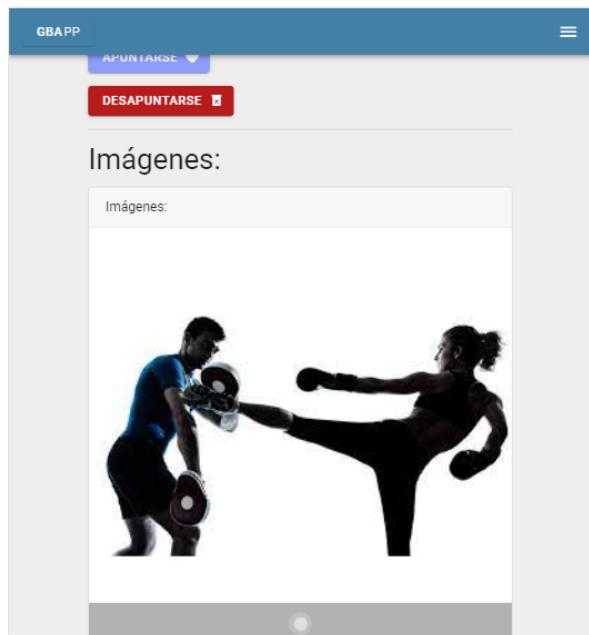


Figura 6.29: Pantalla Actividad Kick Boxing 4

Esta es la pantalla de una actividad, en concreto Kick Boxing. Esta se divide en cuatro partes.

1. La primera parte muestra la descripción de la actividad y los botones para apuntarse o desapuntarse.
2. La segunda muestra las imágenes que los administradores han subido de la actividad.
3. La tercera se compone de una lista de reseñas de usuarios así como de un formulario para crear las mismas.
4. La cuarta trata de unos botones con los que compartir la realización de la acti-

vidad en redes sociales.

En la figura 6.30 se muestra la pantalla del Gimnasio:

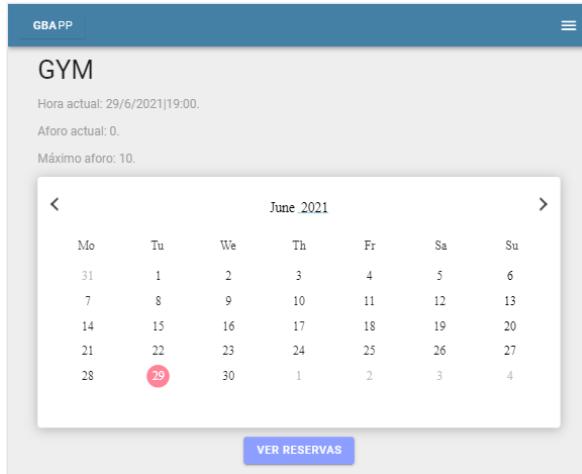


Figura 6.30: Pantalla Gimnasio (sólo una parte)

Esta pantalla es accesible a través del botón del perfil, en ella se muestra la información actual del gimnasio, un calendario con el que poder acceder a un día en el gimnasio, un botón con el que acceder a las reservas de propias del usuario y una lista de máquinas registradas (aquí no se muestra).

En la figura 6.31 se muestra la pantalla de Ver día gimnasio:

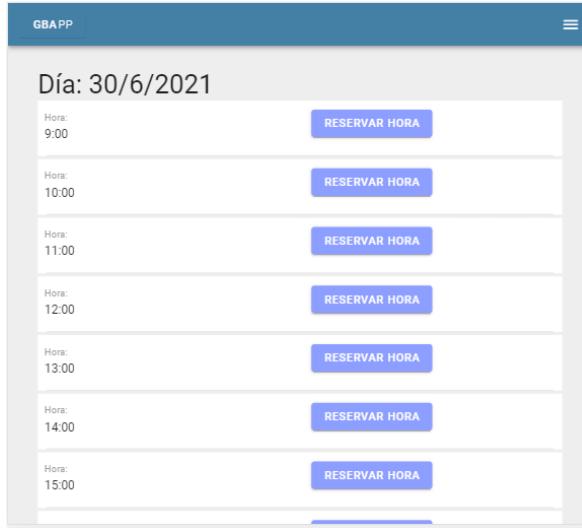


Figura 6.31: Pantalla Día Gimnasio 1 (sólo primera parte)

En esta pantalla se muestra un día normal en el gimnasio. Desde esta pantalla se realizan las reservas.

En la figura 6.32 se muestra la pantalla de Cancelar reservas gimnasio:

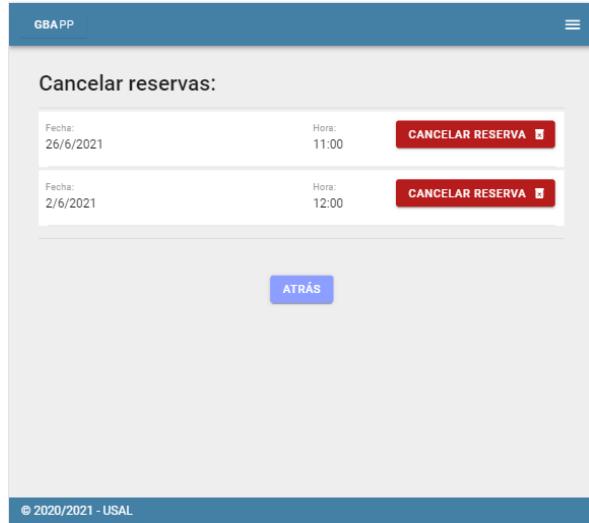


Figura 6.32: Pantalla Cancelar reservas Gimnasio

Por último, esta es la pantalla desde donde los usuarios podrán cancelar las reservas.

6.9.3. Administrador

El usuario administrador será el encargado de la gestión de usuarios y la gestión de actividades.

En la figura 6.33 se muestra el navegador de este actor:

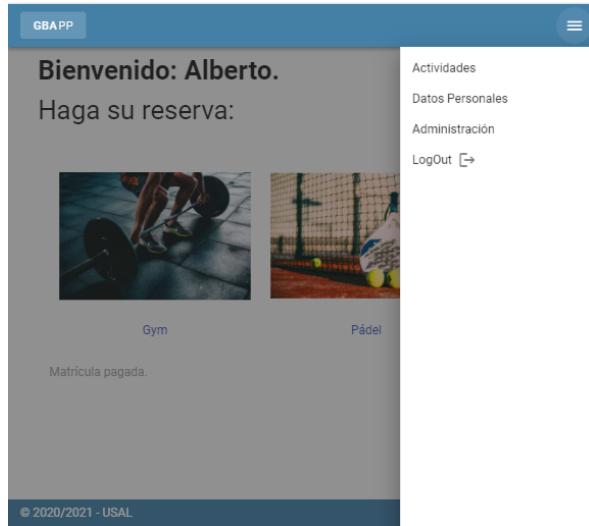


Figura 6.33: Navegador Administrador

Este es el navegador de un usuario administrador una vez logueado. Como se puede observar aparece la opción de administración

En la figura 6.34 se muestra la pantalla de Administración:

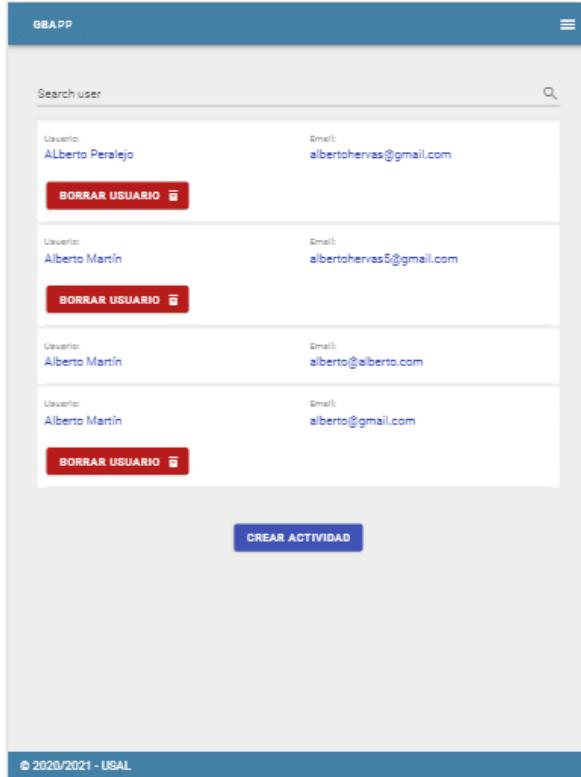


Figura 6.34: Pantalla Administración

Si el usuario pulsa administración en el navegador aparece la siguiente pantalla. En ella podemos ver una lista de usuarios registrados en el sistema, un buscador de usuarios y un botón destinado a crear actividades.

Cabe destacar que si se pulsa el botón de borrar usuario (como también pasa con las actividades) se desplegará un cuadro de dialogo para confirmar la eliminación.

En la figura 6.35 se muestra la pantalla de Ver Usuario:

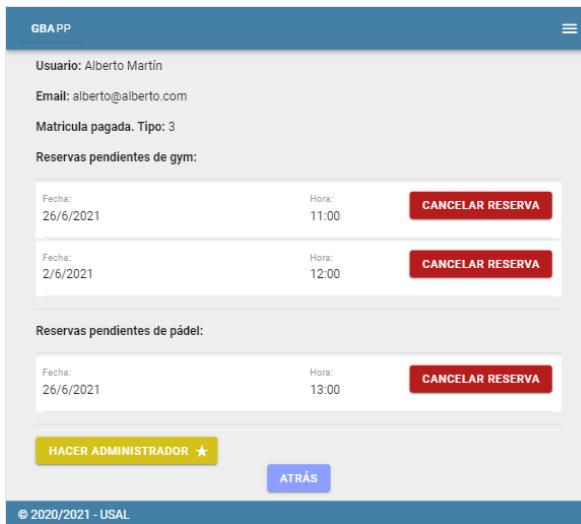


Figura 6.35: Pantalla Ver usuario

Si se accede a la pantalla de ver usuario se desglosará la información del mismo. Esta pantalla ofrece la posibilidad de hacer administrador al usuario que se está viendo así como borrar sus reservas.

En la figura 6.36 se muestra la pantalla de Crear actividad:

The screenshot shows a mobile application interface titled "GBAPP". At the top, there is a blue header bar with the title "Formulario de registro de actividad:". Below the header, there is a white form area with several input fields:

- A text input field labeled "Nombre" (Name) with a person icon.
- A text input field labeled "Descripción" (Description) with a document icon.
- A text input field labeled "Horario" (Time) with a calendar icon.
- A text input field labeled "Días (format: Monday, Tuesday, ..., Sunday)" (Days) with a list icon.
- A text input field labeled "Monitor" with a shield icon.
- A text input field labeled "Aforo máximo" (Maximum capacity) with a person icon.

At the bottom of the form, there are two buttons: a blue "CREAR" button with a user icon and a white "ATRÁS" (Back) button. At the very bottom of the screen, there is a dark blue footer bar with the text "© 2020/2021 - USAL".

Figura 6.36: Pantalla Crear actividad

Si se pulsa el botón de crear actividad se accederá esta pantalla en la que se presenta un formulario para el registro de una actividad.

6.9.4. Sistema

El sistema tiene tareas programadas según el día y la hora de la semana o mes que en el que se encuentra.

Las diferentes tareas son:

- Actualizar las matrículas de los usuarios cada día 1 de mes.
- Actualizar el número de reservas semanales de los usuarios cada domingo.
- Actualizar el estado del gimnasio y pádel cada hora.
- Eliminar las reservas de fechas pasadas y las reservas sin usuarios.

6.9.5. Notificaciones

El sistema se comunica con el usuario a través de notificaciones. Un ejemplo de ello es la figura 6.37:

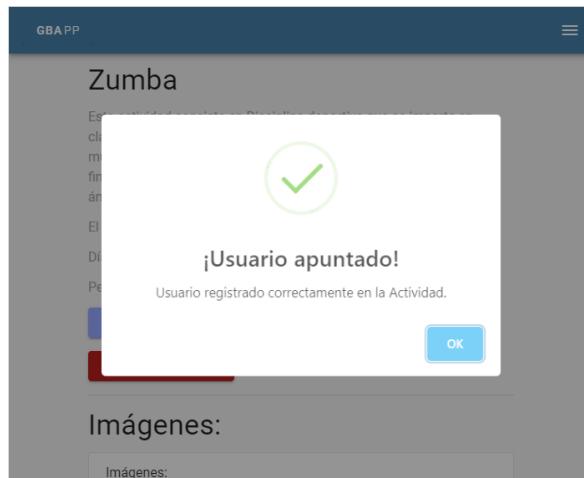


Figura 6.37: Notificación apuntarse a una actividad

7. Conclusiones y líneas de trabajo futuras

En esta sección se recogen tanto las conclusiones obtenidas al finalizar el proyecto como las líneas futuras pensadas como mejora del mismo.

7.1. Conclusiones

Con el proyecto acabado y todos sus objetivos cumplidos se concluye lo siguiente:

En primer lugar, cabe destacar que se ha desarrollado un sistema capaz de cumplir con los objetivos previstos:

- **Gestión de usuarios:** Se ha desarrollado un sistema capaz de manejar los diferentes roles de usuario, clientes y administradores; estos últimos tendrán un control total del sistema manejando a todos los usuarios.
- **Gestión de datos de usuario** El sistema desarrollado ofrece a los usuarios registrados actualizar y modificar sus datos.
- **Gestión de matrícula:** la aplicación ofrece 3 tipos de matrículas que el cliente puede escoger y pagar con un método seguro como lo es PayPal.
- **Gestión de actividades:** el sistema ofrece a los administradores las herramientas necesarias para registrar y borrar actividades. Estas actividades tienen un aforo límite y los clientes se pueden apuntar o desapuntar.
- **Gestión de reseñas:** la aplicación ofrece a los usuarios registrados una forma de dar su opinión de las diferentes actividades creadas.
- **Gestión de gimnasio y pádel:** por último, se ha cumplido el objetivo de ofrecer un sistema de reservas con la finalidad de controlar el aforo del gimnasio o pádel.

En segundo lugar, se ha podido comprobar el esfuerzo que supone el desarrollo de un servicio informático completo cuando se realiza individualmente obteniendo un mayor conocimiento en las tareas de análisis e implementación en el desarrollo de estos servicios o proyectos.

En tercer y último lugar, la aplicación web desarrollada ha supuesto la adquisición de conocimientos en el tema del desarrollo web, que como se explico previamente era un tema de interés personal.

También se ha visto la importancia de conocer distintas herramientas(frameworks, lenguajes o disciplinas), de tal modo que se agilice la elección de estas para cada situación particular.

En resumen, se ha puesto a prueba todos los conocimientos adquiridos en el Grado mediante su utilización en el proyecto.

7.2. Líneas de trabajo futuras

Tras realizar el desarrollo del sistema software, se proponen algunas mejorar o ampliaciones que sería interesante realizar:

- **Balizas BLE:** con el uso de las balizas BLE se podría llevar una trazabilidad de los usuarios que han usado cada maquina ya que el sistema tendrá la capacidad de saber que maquina ha usado cada usuario y el tiempo que la ha usado.

Para ello se instalaría en cada una de las maquinas una baliza BLE que será detectada por la aplicación web que se desarrollaría, de esta manera se podría llevar una trazabilidad al mismo tiempo que se podrían calcular estadísticas de tiempo de uso de maquinas como puede ser el desgaste asociado al uso. En la figura 7.1 se muestra un pequeño diagrama diseñado.



Figura 7.1: Balizas BLE

- **Registro:** otra opción sería el permitir el registro e inicio de sesión con proveedores de servicio como Twitter, Facebook o Google.



Figura 7.2: Sugerencia registro

- **Buzón sugerencias de actividades:** establecer un buzón de sugerencias de actividades donde los usuarios podrían dejar sus preferencias para que los administradores lo tengan en cuenta.
- **Aplicación móvil:** una de las opciones de futuro es el desarrollo de una aplicación móvil. Como el sistema de autenticación se ha desarrollado usando JWT, el cual es compatible para las aplicaciones móviles, se piensa el desarrollo de una aplicación móvil dado que se podría realizar sin necesidad de crear un nuevo servidor, es decir, reutilizando la parte del backend ya creada.



Figura 7.3: Aplicación móvil

- **Handlebars:** mejorar la plantilla de los correos enviados a los usuarios durante el reseteo de contraseña.

8. Referencias

Accedido entre el 22/02/2021 y 30/06/2021.

- Vue.js
<https://es.vuejs.org/v2/guide/#>
- Códigos de estado HTTP
https://es.wikipedia.org/wiki/Anexo:C%C3%B3digos_de_estado_HTTP
- VueRouter
<https://elabismodenull.wordpress.com/2017/05/24/vuejs-conceptos-avanzados-de-vue-router/>
- Backend validator
<https://www.npmjs.com/package/validator>
- Bcrypt
<https://www.npmjs.com/package/bcrypt>
<https://www.izertis.com/es/-/blog/criptacion-de-password-en-nodejs-y-mongodb-bcrypt>
- Vue Google Maps
<http://cristiantorresalfaro.blogspot.com/2020/08/mostrar-ubicacion-en-google-map-vue-js.html>
<https://blog.nubecollectiva.com/como-mostrar-google-maps-con-vue-js-2-6-vue-cli-3-11-parte-1/>
- Axios
<https://styde.net/solicitudes-http-con-axios/>
<https://bezkoder.com/vue-axios-file-upload/>
<https://www.arsys.es/blog/programacion/axios/>
- Autentificación
<https://bezkoder.com/jwt-vue-vuex-authentication/>
<https://bezkoder.com/node-js-mongodb-auth-jwt/>
<https://bezkoder.com/node-express-vue-jwt-auth/>
<https://blog.logrocket.com/implementing-a-secure-password-reset-in-node-js/>

- REST API

<https://bezkoder.com/node-express-mongodb-crud-rest-api/>

https://github.com/audreylamy/Booking_web_app

<https://bezkoder.com/vue-js-crud-app/>

- Local Storage

<https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage>

<https://www.bbvaapimarket.com/es/mundo-api/api-rest-que-es-y-cuales-son-sus-ventajas-en-el-desarrollo-de-proyectos/>

- JWT

https://en.wikipedia.org/wiki/JSON_Web_Token#Standard_fields

<https://www.npmjs.com/package/jsonwebtoken>

- MEVN

<https://bezkoder.com/vue-node-express-mongodb-mevn-crud/>

- Multer

<https://bezkoder.com/node-js-express-file-upload/>

<https://bezkoder.com/vue-multiple-files-upload/>

<https://www.npmjs.com/package/multer>

- MongoDB

<https://bezkoder.com/mongoose-one-to-many-relationship/>

<https://bezkoder.com/mongodb-many-to-many-mongoose/>

<https://code.tutsplus.com/es/articles/an-introduction-to-mongoose-for-mongodb-and-nodejs--cms-29527>

<https://sitiobigdata.com/2017/12/27/mongodb-arquitectura-y-modelo-de-datos/#>

<https://core.ac.uk/download/pdf/44310803.pdf>

- Paypal

<https://www.npmjs.com/package/@paypal/paypal-js>

<https://developer.paypal.com/>

<https://dev.to/galihm/integrate-paypal-smart-payment-buttons-in-rails-part-1-5101>

<https://developer.paypal.com/docs/business/checkout/configure-payments/single-page-app/#vue>

- Node-cron
<https://www.npmjs.com/package/node-cron>
- Vue Functional Calendar
<https://www.npmjs.com/package/vue-functional-calendar>
<https://github.com/ManukMinasyan/vue-functional-calendar?ref=madewithvuejs.com>
<https://vue-functional-calendar.vercel.app/>
- Nodemailer
<https://nodemailer.com/about/>
- Vuetify
<https://vuetifyjs.com/en/>
<https://www.youtube.com/playlist?list=PL4cUxeGkcC9g0MQZfHwKcuB0Yswgb3gA5>
- Iconos
<https://fonts.google.com/icons?selected=Material+Icons>
- VueSocialSharing
<https://nicolasbeauvais.github.io/vue-social-sharing/?path=/story/vuesocialsharing--multiple-share-networks>
- Certbot - <https://certbot.eff.org/>
- Javascript
https://developer.mozilla.org/es/docs/Learn/JavaScript/First_steps/What_is_JavaScript
<https://developer.mozilla.org/es/docs/Web/JavaScript>
- JSON
<https://www.nextu.com/blog/que-es-json/>
- Node.js
<https://nodejs.org/es/>
- Express
<https://expressjs.com/es/>
https://developer.mozilla.org/es/docs/Learn/Server-side/Express_Nodejs/Introduction
- Moongoose
<https://mongoosejs.com/>

<https://codigofacilito.com/articulos/que-es-mongoose>

- Handlebars

<https://desarrolloweb.com/manuales/manual-handlebars.html>

- PM2

<https://pm2.keymetrics.io/>

- Vuex

<https://unpocodejava.com/2019/05/09/que-es-vuex/>

<https://vuex.vuejs.org/>

- Sweetalert

<https://sweetalert.js.org/guides/>

- HTML

<https://developer.mozilla.org/es/docs/Web/HTML>

- CSS

<https://developer.mozilla.org/es/docs/Web/CSS>

- NoSQL

<https://www.unir.net/ingenieria/revista/bases-de-datos-nosql/>

- Herramientas CASE

https://es.wikipedia.org/wiki/Herramienta_CASE

- REM

http://www.lsi.us.es/descargas/descarga_programas.php?id=3

- Visual Paradigm

<https://www.visual-paradigm.com/>

- Microsoft Project

<https://www.microsoft.com/es-es/microsoft-365/project/project-management-software>

- Git

<https://es.wikipedia.org/wiki/Git>

- Digital Ocean

<https://www.digitalocean.com/>

- Vue Styleguidist

<https://vue-styleguidist.github.io/>

- JSdoc
<https://jsdoc.app/>
- Swagger
<https://swagger.io/>
- Postman
<https://www.postman.com/>
- Visual Studio Code
<https://code.visualstudio.com/>
- VeeValidate
<https://vee-validate.logaretm.com/v4/>
- Overleaf y Latex
<https://www.overleaf.com>
<https://www.latex-project.org/get/>
- McConnell, S. - “Desarrollo y gestión de proyectos informáticos”. Mc Graw Hill, 1997.
- Pressman, R. S. - “Ingeniería del Software: Un Enfoque Práctico”. 7^a Edición. McGraw-Hill. 2010.
- Amador Durán Toro, Beatriz Bernárdez Jiménez – “Metodología para la Elicitación de Requisitos de Sistemas Software”.