
Universidade de Vigo

ESCOLA SUPERIOR DE ENXEÑARÍA INFORMÁTICA

Memoria do Traballo de Fin de Grao que presenta

D. Jorge Alcalde Piñeiro

para a obtención do Título de Graduado en Enxeñaría Informática



Xullo, 2024

Traballo de Fin de Grao Nº: OVFKBJE

Titor/a: Daniel González Peña

Área de coñecemento: Linguaxes e Sistemas Informáticos

Departamento: Informática

Agradecimientos

A mi madre y a mi padre por ayudarme y apoyarme, permitirme estudiar y viajar.

Gracias a mi abuela por cuidarme y criarme y pasar tanto tiempo conmigo ayudándome a crecer hasta convertirme en la persona que soy hoy.

Gracias a mi mejor amigo y a mi pareja por escuchar mis problemas.

Gracias a todas las personas que aportaron en mi crecimiento y en que sea lo que soy hoy en día.

Índice de contenidos

| | |
|--|----|
| 1. Introducción | 6 |
| 2. Objetivos | 6 |
| 3. Resumen de la solución propuesta | 7 |
| 3.1. Solución propuesta..... | 7 |
| 3.2 Metodología empleada | 7 |
| 4. Planificación y seguimiento..... | 8 |
| 4.1 Planificación inicial | 8 |
| 4.2 Seguimiento..... | 9 |
| 5. Arquitectura | 11 |
| 5.1 Front-End..... | 11 |
| 5.2 Back-End | 13 |
| 6. Tecnologías e integración de productos de terceros..... | 14 |
| 6.1 Tecnologías | 14 |
| 6.2. Herramientas..... | 16 |
| 6.3 Librerías..... | 17 |
| 7. Especificación y análisis de requisitos | 18 |
| 8. Diseño del software..... | 21 |
| 8.1. Vista estática | 21 |
| 8.2. Vista dinámica | 25 |
| 9. Gestión de datos e información | 25 |
| 10. Pruebas..... | 28 |
| 11. Manual de usuario | 33 |
| 11.1 Requisitos mínimos | 33 |
| 11.2 Instalación | 33 |
| 11.3 Funcionamiento | 34 |
| 12. Principales aportaciones | 42 |
| 13. Conclusiones..... | 43 |
| 14. Vías de trabajo futuro..... | 43 |
| 15. Referencias | 45 |

Índice de ilustraciones

| | |
|---|----|
| Ilustración 1: DIAGRAMA DE GANTT INICIAL | 9 |
| Ilustración 2: DIAGRAMA DE GANTT FINAL..... | 10 |
| Ilustración 3: ARQUITECTURA CLIENTE-SERVIDOR..... | 11 |
| Ilustración 4:ARQUITECTURA FRONT-END | 12 |
| Ilustración 5:ARQUITECTURA BACK-END..... | 13 |
| Ilustración 6: ESTRUCTURA DE ARCHIVOS | 22 |
| Ilustración 7: DIAGRAMA CLASES DEL BACK-END | 22 |
| Ilustración 8: DIAGRAMA DE CLASES DEL FRONT-END..... | 24 |
| Ilustración 9: DIAGRAMA DE SECUENCIA DEL SISTEMA..... | 25 |
| Ilustración 10: MODELO ENTIDAD/RELACIÓN EXTENDIDO | 26 |
| Ilustración 11: SECCIÓN DE INICIO | 35 |
| Ilustración 12: FORMULARIO DE INICIO DE SESIÓN | 35 |
| Ilustración 13: FORMULARIO DE REGISTRO | 36 |
| Ilustración 14: PÁGINA DE INICIO DE ADMINISTRADOR | 36 |
| Ilustración 15: PÁGINA DE GESTIÓN DE USUARIOS..... | 37 |
| Ilustración 16: PÁGINA DE GESTIÓN DE CLIENTES..... | 37 |
| Ilustración 17: PÁGINA DE GESTIÓN DE CLASES PROPIAS | 38 |
| Ilustración 18: PÁGINA DE GESTIÓN DE CLASES GESTIONADAS..... | 38 |
| Ilustración 19: PÁGINA DE GESTIÓN DE SESIONES DE CLASES PROPIAS | 39 |
| Ilustración 20: PÁGINA DE GESTIÓN DE CLASES GESTIONADAS..... | 39 |
| Ilustración 21: PÁGINA DE GESTIÓN DE CLIENTES INSCRITOS..... | 40 |
| Ilustración 22: PÁGINA DE INICIO DE CLIENTES..... | 40 |
| Ilustración 23: PÁGINA DE BÚSQUEDA DE CLASES..... | 41 |
| Ilustración 24: PÁGINA DE GESTIÓN DE SESIONES COMO CLIENTE | 41 |
| Ilustración 25: PÁGINA DE CLASES INSCRITAS | 42 |

Índice de tablas

| | |
|-------------------------------------|----|
| Tabla 1: PLANIFICACIÓN INICIAL..... | 9 |
| Tabla 2: PLANIFICACIÓN FINAL..... | 10 |
| Tabla 3: API | 28 |

1. Introducción

En la era digital actual, la tecnología se ha integrado de manera significativa en diversos aspectos de nuestra vida cotidiana, incluyendo la gestión de nuestras actividades y de las diferentes clases y eventos a los que asistimos.

La digitalización también alcanzó los gimnasios, abarcando la gestión de los accesos, las inscripciones y el manejo de las diferentes actividades y recursos que ofrece un gimnasio. El problema reside en que, con el avance de los años, las aplicaciones que en un principio fueron pequeñas e intuitivas, acabaron siendo grandes aplicaciones cuyo tamaño y cantidad tan diversa de funcionalidades termina por convertirlas en complicadas de entender y, muchas veces, hasta excedentes para el usuario. Estas presentan una alta curva de aprendizaje. A su vez, la gran cantidad de funcionalidades hacen que para un usuario la elección de qué aplicación usar sea muchas veces abrumadora ante la inmensa cantidad de opciones.

En este Trabajo Fin de Grado (TFG) se propone la creación de una aplicación web que facilite el acceso de los diferentes usuarios a las clases y, dentro de estas, sesiones que ofrece un gimnasio y que, a su vez, facilite la creación de las diferentes clases y sesiones por los monitores y la gestión de estas por su parte. Así se crearía una aplicación sencilla y simple que se centra en el funcionamiento mínimo de un gimnasio para lograr una mayor simpleza y sencillez y comodidad de los usuarios al momento de emplear esta.

2. Objetivos

El objetivo principal de este TFG es la creación de una aplicación web orientada a la gestión de clases en el gimnasio. Se deberán implementar las siguientes funciones:

- Gestión de usuarios: se debe generar un sistema de creación de usuarios, edición de usuarios, validación de clientes y eliminación de usuarios, todo esto desde el punto de vista de un administrador.
- Gestión de clases: desde el punto de vista de un instructor, se debe poder crear, modificar y eliminar clases, así como poder acceder a aquellas clases en las que gestionan por lo menos una sesión. Por parte del cliente, este debe poder buscar clases y acceder de a aquellas clases a la que esté, por lo menos, inscrito a una sesión.
- Gestión de sesiones: por parte del instructor, este deberá poder, sobre las clases que él creó, crear sesiones, editar estas, duplicar sesiones y borrarlas. Por parte de las sesiones que gestiona, podrá acceder al listado de clientes que están inscritos y gestionar el estado de su asistencia. Como cliente, podrá inscribirse y desinscribirse a estas sesiones.

3. Resumen de la solución propuesta

3.1. Solución propuesta

La solución propuesta ha sido realizada para el cumplimiento de los objetivos anteriormente enumerados mediante la realización de una aplicación web con las siguientes tecnologías:

- **Front-End:** se utilizará React para crear un SPA basada en Componentes web y Tailwind CSS para la parte de UX/UI.
- **Back-End:** se utilizará el entorno de ejecución JavaScript *NodeJS* junto con el framework *Express* para la simplificación de la creación de una *API Rest* de forma más simple y sencilla. Para la persistencia de datos, ha sido elegida una base de datos relacional *MySQL* utilizando *Prisma*, un ORM, para la gestión de esta.

Estas tecnologías han sido seleccionadas como resultado de la motivación de mejorar mis conocimientos en JavaScript, así como adentrarme en la creación completa de APIs Rest y en el empleo de ORMs. Por estos motivos se seleccionaron una combinación de tecnologías que están en auge actualmente en la creación de páginas web y que marcarán una base para el futuro crecimiento y trayectoria profesional.

3.2 Metodología empleada

Respecto a la metodología empleada en este proyecto, se decidió la utilización de la metodología ágil Scrum. Esta decisión fue hecha debido a que esta metodología se centra en aumentar el tiempo de trabajo sobre la aplicación, reduciendo al mínimo las tareas de documentación que no aportan valor al desarrollo, así como también nos aporta una gran flexibilidad y adaptación ante los requisitos cambiantes.

El objetivo principal de Scrum es permitir un desarrollo de software más ágil, flexible y centrado en el valor del cliente. Para conseguir esto, Scrum se basa en un modelo iterativo e incremental, dirigido por una lista de requisitos basados en las necesidades del cliente. Para poder comprender el funcionamiento de Scrum, se deben tener en cuenta una serie de conceptos fundamentales de Scrum: iteraciones y los roles del equipo.

Las iteraciones en Scrum se llaman sprints y se caracterizan por tener una duración fija, que suele rondar entre 1 a 4 semanas; y por tener una planificación previa a la realización del sprint, donde se tendrá una reunión previa al comiendo de este para decidir qué elementos se tratarán en este y una reunión al final del sprint para mostrar una demostración del resultado del trabajo.

Por otro lado, tenemos los diferentes roles por los que está formado el equipo: **Product Owner**, **Scrum Master** y **Equipo de Desarrollo**.

- **Product Owner:** es el responsable del producto final. Es el encargado de decidir qué se va a desarrollar, cuándo y en qué orden para obtener un producto final que le aporte el mayor valor posible al cliente.
- **Scrum Master:** lidera el equipo de scrum y mantiene a los miembros enfocados en el seguimiento de los principios y las prácticas de scrum, ayudando en la creación y

seguimiento del desarrollo, solucionando aquellos problemas provenientes de la productividad y rendimiento.

- **Equipo de Desarrollo:** encargados de determinar cómo producir lo solicitado por el Product Owner.

En este TFG, los integrantes del equipo Scrum son:

- **Tutor:** realiza el papel de Scrum Master.
- **Alumno:** realiza el papel de Product Owner y de Equipo de Desarrollo.

Los artefactos que serán manejados durante todo el desarrollo serán los siguientes:

- **Backlog de producto:** lista de tareas, ordenada en función a su prioridad, que contiene todos los requisitos del proyecto y que otorgan una vista general del proyecto y la evolución de este.
- **Backlog de sprint:** documento que almacena los requisitos a completar en el siguiente sprint. Habitualmente se subdivide en tareas.
- **Incremento del producto:** unión de todos los elementos del Product Backlog completados durante el sprint presente y el valor de los incrementos de todos los anteriores.

4. Planificación y seguimiento

En los siguientes puntos de este apartado se mostrarán la planificación que se hizo inicialmente y la planificación que finalmente se ejecutó. Estas planificaciones se mostrarán haciendo uso de Tablas y Diagramas de Gantt.

4.1 Planificación inicial

La planificación original del proyecto ha sido realizada teniendo en cuenta que la duración máxima que se le puede dedicar a la creación del TFG es de 300h y los horarios del alumno. Para cumplir esto, se han planificado 8 sprints, todos con duración de 2 semanas, con excepción del primer sprint, siendo este la investigación de las tecnologías y una mayor duración debido a que esta se realizará durante el primer cuatrimestre; el segundo sprint, que tiene una duración de 3 semanas y el último, sprint que tiene una duración de 1 semana. Cada sprint tendrá una duración de 40h, con excepción del primer, segundo y último sprint que duran 10 h más el primero, 20h más el segundo y 20h menos el último.

| Sprint | Fecha inicio | Fecha fin | Duración |
|--|--------------|------------|----------|
| #1. Spike Tecnológico | 25/09/2023 | 11/02/2024 | 139d-50h |
| #2. Documentación inicial, Landing page y Login + registro | 12/02/2024 | 03/03/2024 | 21d-60h |
| | | | |

Gestión web de un gimnasio

| | | | |
|-----------------------------------|------------|------------|-----------|
| #3. Funcionalidades administrador | 4/03/2024 | 17/03/2024 | 14d-30h |
| #4. Gestión clases monitor | 18/03/2024 | 31/03/2024 | 14d-30h |
| #5. Gestión cliente | 1/04/2024 | 14/04/2024 | 14d-30h |
| #6. Gestión asistencia | 15/04/2024 | 28/04/2024 | 14d-40h |
| #7. Búsqueda clases | 29/04/2024 | 12/05/2024 | 14d-40h |
| #8. Documentación final | 13/05/2024 | 19/05/2024 | 7d-20h |
| | | | |
| Total | 25/09/2023 | 19/05/2024 | 237d-300h |

Tabla 1: PLANIFICACIÓN INICIAL

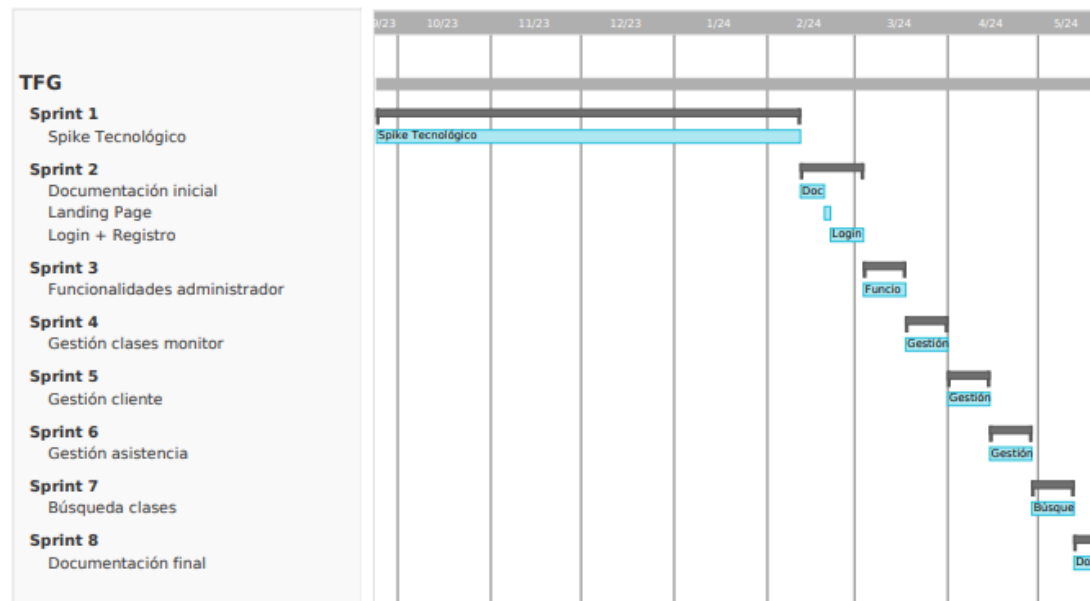


Ilustración 1: DIAGRAMA DE GANTT INICIAL

4.2 Seguimiento

Pese a la intención de una dedicación más corta a la investigación, el resultado final fue una investigación mucho más extensa. A demás, debido a la distribución horaria sin saber exactamente las horas que tendría disponible exactamente el alumno y ser realizadas en base a especulaciones, esto tuvo como resultado que se redujeran el número de sprints y de horas

Gestión web de un gimnasio

dedicadas a cada uno. Por otra parte, al haber realizado una mayor investigación al inicio, esto repercutió de forma muy positiva en la realización del código, hecho que agilizó mucho la producción de este. El último sprint tiene una gran diferencia de días con respecto al resto debió al parón como resultado de la realización de exámenes y demás trabajos.

| Sprint | Fecha inicio | Fecha fin | Duración |
|--|--------------|------------|-----------|
| #1. Spike Tecnológico | 25/09/2023 | 11/02/2024 | 139d-90h |
| #2. Documentación inicial, Landing + autenticación, gestión de usuarios, gestión de clases | 12/02/2024 | 23/02/2024 | 12d-30h |
| #3. Revisión de clases, gestión de sesiones, gestión de sesiones gestionadas por el instructor | 24/02/2024 | 15/03/2024 | 21d-40h |
| #4. Inscripción y búsqueda de clases como cliente, Gestión asistencia clientes | 16/03/2024 | 5/04/2024 | 21d-40h |
| #5. Remodelación visual, duplicación de sesiones y Documentación final | 6/04/2024 | 16/06/2024 | 71d-70h |
| | | | |
| Total | 25/09/2023 | 16/06/2024 | 265d-270h |

Tabla 2: PLANIFICACIÓN FINAL

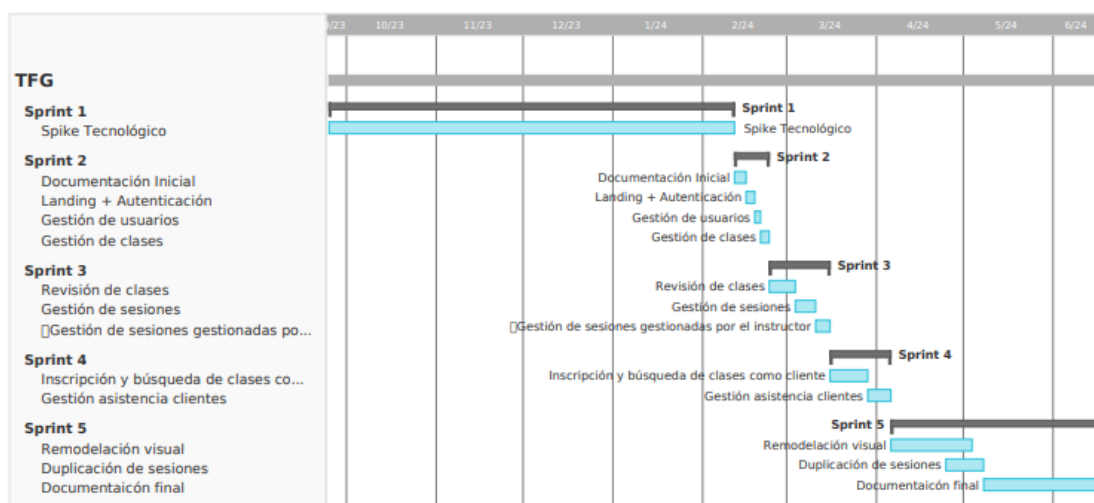


Ilustración 2: DIAGRAMA DE GANTT FINAL

5. Arquitectura

Para la arquitectura se ha empleado un modelo Cliente/Servidor. En esta arquitectura, conformada por un cliente, un servidor y una base de datos; el cliente, que sería el navegador, realiza peticiones a un servidor, el cual se comunica con la base de datos y devuelve el resultado de procesar la petición. Para esto utiliza las tecnologías anteriormente mencionadas.

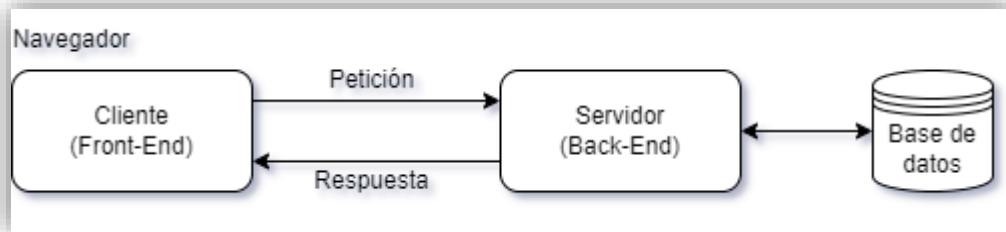


Ilustración 3: ARQUITECTURA CLIENTE-SERVIDOR

Esta arquitectura es la arquitectura general del proyecto, tanto el Front-End como el Back-End tienen su propia arquitectura que se adapta mejor a los requisitos y necesidades de cada parte. Gracias a **pnpm create vite@latest** pudimos crear la aplicación de React de forma automática, instalando una configuración por defecto junto con las dependencias necesarias para la creación de una aplicación de React, lo que nos permitió simplificar el proceso de creación de la aplicación y de la arquitectura. Para el Back-End con **npm i** pudimos crear, al igual que en el Front-End con React, un esqueleto base de NodeJS con las dependencias mínimas para poder empezar a desarrollar. Las arquitecturas empleadas por cada parte serán explicadas en los siguientes apartados.

5.1 Front-End

React es una librería de JavaScript enfocada al desarrollo de aplicaciones móviles y web. React está basada en componentes, entendidos como piezas de código encapsuladas que combinan tanto la lógica de negocio como la interfaz de usuario en una sola unidad reutilizable. La esencia fundamental detrás de los componentes es descomponer una aplicación en partes más pequeñas y manejables, simplificando así la construcción y el mantenimiento del código.

Al fusionar la lógica de negocio y la interfaz de usuario en un único componente, se genera una entidad funcional y autónoma que puede emplearse fácilmente en distintas secciones de la aplicación o, incluso, en aplicaciones completamente diferentes. Este enfoque modular facilita la reutilización del código y contribuye a una arquitectura más escalable y mantenible.

Los componentes pueden ser divididos en dos tipos, “Dumb Components”, caracterizados por no tener un estado interno y sólo estar encargados de renderizar algo visual; y los “Smart Components”, que contienen lógica de negocio y operaciones y, a su vez, también son encargados de la gestión del estado.

Al ser los componentes independientes, esto nos permite poder tener el mismo componente múltiples veces por pantalla con estados internos diferentes cada uno.

La manipulación del DOM es un tema para tener en cuenta a la hora de desarrollar una aplicación web. React destaca en su gestión del entorno y del manejo de los estados de los componentes. Este carga los componentes de forma dinámica, reservando la memoria necesaria para el funcionamiento de cada componente mientras estos están siendo mostrados, de forma que el estado interno de cada componente será preservado mientras se rendericen. React, una vez dejan de mostrarse estos, recupera el espacio en memoria usado para ese componente y, por ende, perdiendo ese estado interno. A demás, React se encarga de gestionar las actualizaciones de los diferentes elementos del DOM, ya que este construye árboles de renderizado, marcando así una relación de parentesco ente el contenedor padre de los componentes y aquellos que viven dentro de este, de forma que en cuanto el padre desaparezca los hijos también lo harán.

Otra de las ventajas que aporta React son los hooks. Los hooks son funciones que te permiten acoplar el estado de React y el ciclo de vida. Estos son muy útiles para la gestión de los estados internos de un componente.

A medida que crece la aplicación, la gestión del estado de esta aumentará en su complejidad. Para reducir esto se ha empleado el hook useContext. Este te permite leer y suscribirte a un contexto desde tu componente, de forma que podremos crear diferentes contextos en la aplicación y después usarlos con este hook. Los contextos a su vez también son hooks de React. Estos son definidos en un archivo y para poder acceder a estos deberemos envolver aquellos componentes que queramos que puedan acceder al contexto con un proveedor. Esto nos permitirá no sólo compartir estados entre componentes, sino que también nos permitirá restringir el acceso al contexto a aquellos componentes que creamos pertinentes.

Además de lo anterior, también se usan otros elementos:

- **RouterProvider:** gestiona las rutas de la aplicación.
- **CustomHooks:** hooks propios que gestionan gran parte de la lógica de la aplicación, así como las llamadas a los diferentes servicios para ponerse en contacto con el Back-End.

Finalmente, para la realización de las peticiones HTTP se emplea la API Fetch, que permite la realización de llamadas AJAX (Asynchronous JavaScript y XML) simples con JavaScript.

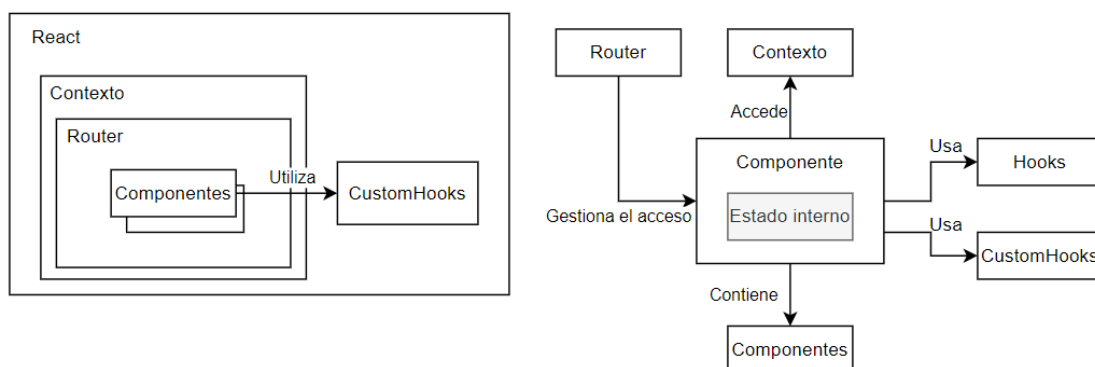


Ilustración 4:ARQUITECTURA FRONT-END

En esta ilustración podemos ver por un lado cómo sería la estructura, siendo que React el contenedor padre, seguido del contexto y el Router que envuelven a los diferentes componentes, que a su vez usan los CustomHooks. Por otro lado, tenemos la vista de cómo

funcionaría un componente, gestionando el Router el acceso al componente y este pudiendo acceder al contexto, contener otros componentes, usar los propios hooks de React y, por otro lado, los CustomHooks, siendo estos quienes ya sea directamente ellos y llamando a una función, harían las peticiones HTTP.

5.2 Back-End

El intercambio de datos entre el Front-End y la base de datos se hará mediante una API que expondrá recursos que pueden ser consumidos mediante URLs.

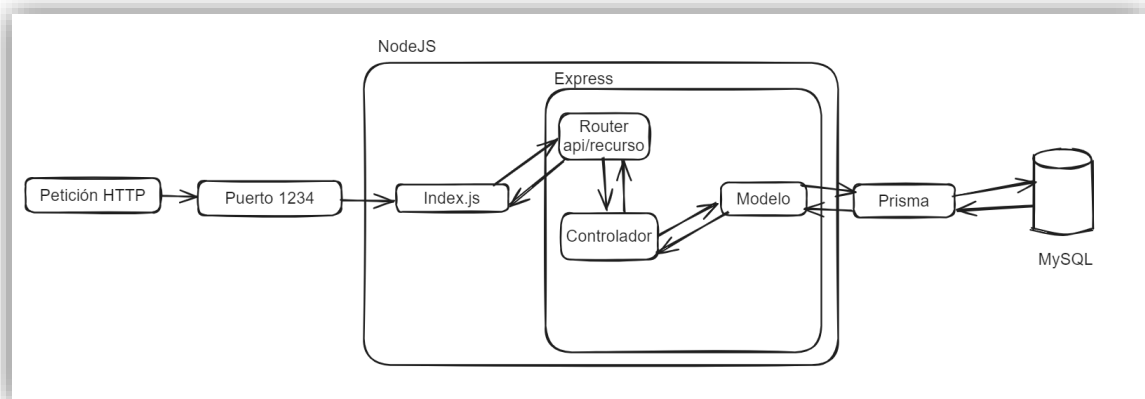


Ilustración 5: ARQUITECTURA BACK-END

En la ilustración 5 podemos ver la arquitectura del Back-End. Con NodeJS se crea un servidor web el cual está encargado de recibir las peticiones HTTP al puerto 1234. Como es necesario un manejo más específico de cada tipo de petición HTTP y realizar una gestión independiente de las peticiones por medio de diferentes URLs, la funcionalidad de NodeJS se complementa con el uso del framework Express, el cual nos permite la gestión de estas peticiones de manera sencilla e independiente, además de, así, evitar que tengamos que implementar una solución ya existente.

El código que será ejecutado en la aplicación será determinado por la información de las solicitudes HTTP entrantes. Esta información se comparte, modifica y trata en el camino entre las distintas capas de la arquitectura que componen la aplicación hasta, finalmente, devolver una respuesta. Para la conexión con la base de datos, ha sido decidido el uso de Prisma, un ORM que simplifica al máximo la gestión y mantenimiento de la base de datos, y, así, reducir al mínimo los problemas derivados de esto.

- **Router:** capa donde se definen las diferentes URLs que componen la API. Ante la llegada de una petición, esta será analizada tanto por el método como por la propia URL a la que quiere acceder y, en función de esto, se decidirá el controlador encargado de gestionar esta petición.
- **Controlador:** capa contenedora de la lógica sobre cómo responder a una petición y que trabaja directamente con los modelos.

- **Modelos:** capa encargada de la gestión de la información y de la realización de la lógica de la aplicación más cercana a la información.
- **Prisma:** capa intermediara entre los modelos y la base de datos encargada de la gestión de las entidades de la base de datos, así como del mantenimiento de la integridad estructural de esta. En esta capa se define la estructura de la base de datos.

6. Tecnologías e integración de productos de terceros

En este apartado se explicarán las tecnologías, librerías, herramientas y productos de terceros utilizados para el desarrollo de este proyecto.

6.1 Tecnologías

JavaScript

Lenguaje de programación de alto nivel, interpretado y orientado a objetos, utilizado principalmente para el desarrollo de aplicaciones web, pensado para agregar potencial de iteración y dinamismo a las páginas web.

En el proyecto, este ha sido empleado tanto en la realización del Front-End como en la creación del Back-End.

MySQL

Sistema de administración de bases de datos relacionales creado por Oracle.

MySQL ha sido empleado para el almacenamiento de los datos de la aplicación.

NodeJS

Entorno de tiempo de ejecución de JavaScript de código abierto. Este es asíncrono y está ejecutado en el motor de tiempo de ejecución JavaScript V8. NodeJS está basado en eventos y se emplea para el desarrollo de aplicaciones web con E/S de datos constantes. NodeJS ha sido diseñado para optimizar el rendimiento y escalabilidad en aplicaciones web.

NodeJS fue utilizado para la creación del servidor backend de la aplicación.

NPM

Gestor de paquetes de NodeJS, herramienta por defecto para la instalación y gestión de estos paquetes para JavaScript. NPM a su vez también es el repositorio de paquetes más grandes que existe.

NPM fue utilizado para la instalación de las dependencias de la aplicación.

PNPM

Gestor de paquetes de NodeJS que, a diferencia de otras herramientas de gestión, este utiliza almacenamiento centralizado y enlaza los paquetes a través de hard links, en vez de tenerlos por separado.

PNPM fue utilizado para la instalación de las dependencias de la aplicación.

Express

Entorno de trabajo para aplicaciones web para Node.js, de código abierto y con licencia MIT. Se utiliza para desarrollar aplicaciones web y APIs. Permite una configuración de rutas para una respuesta completa ante una petición HTTP.

Express ha sido empleado para la creación del Back-End.

Prisma

Es un ORM, una plataforma que permite una abstracción sobre una base de datos y que permite la gestión de esta de forma simplificada y clara al dar un modelo de datos claro y fácil. La interacción con la base de datos es realizada mediante un lenguaje de modelado específico de Prisma.

Prisma ha sido empleado para la gestión de la base de datos, así como su creación.

Tailwind CSS

Framework CSS de bajo nivel altamente personalizable que permite un desarrollo ágil, basado en clases de utilidad que se pueden aplicar de forma sencilla al código HTML. Tailwind CSS se encarga de la traducción automática de sus clases a CSS puro.

Tailwind CSS ha sido empleado en la creación del apartado visual del Front-End.

6.2. Herramientas

Visual Estudio Code

Editor de código fuente desarrollado por Microsoft para Windows, Linux, macOS y web. Es software libre y multiplataforma. Cuenta con soporte para la depuración y un control integrado de Git. A su vez, también cuenta con un sinfín de extensiones que amplían las funcionalidades de este.

Visual Estudio Code ha sido utilizado como el editor de código durante la creación de la aplicación.

Git

Software de control de versiones, 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. Su uso se centra en registrar los cambios de código realizados en local y poder compartirlo.

Git fue usado como sistema de control de versiones de la aplicación.

GitHub

Herramienta web ampliamente utilizada que emplea Git para el control de versiones. Esta nos permite el almacenamiento de repositorios de código, así como la descarga de estos, su actualización y una revisión del historial de cambios que se han ido sucediendo.

GitHub fue utilizado como sistema de control principal de versiones de la aplicación.

GitHub Copilot

Herramienta de inteligencia artificial basada en la nube y desarrollada por GitHub y OpenAI como asistente para los usuarios mediante el autocompletado de código.

GitHub Copilot ha sido utilizado durante el desarrollo de la aplicación para la agilización del desarrollo.

Draw.io

Software de dibujo gráfico multiplataforma que permite el diseño de gráficos y diagramas de forma simple y gratuita.

Draw.io ha sido utilizado para la creación de gráficos y diagramas de la documentación de este proyecto.

Excalidraw

Herramienta de dibujo online que permite la creación de gráficos y diagramas de forma simple, concisa y gratuita, además de proporciona una forma rápida de crear y compartir diagramas.

Excalidraw ha sido utilizado para la creación de gráficos y diagramas de la documentación de este proyecto.

XAMPP

Herramienta de desarrollo que incluye diferentes softwares libres, como Linux, Apache, MySQL o Pearl.

XAMPP ha sido utilizado para la creación de un servidor local encargado de almacenar la base de datos MySQL.

TeamGantt

Software de gestión de proyectos de código abierto. Se centra en la programación de proyectos de equipos pequeños.

TeamGantt ha sido empleado para la realización de los diagramas de Gantt.

6.3 Librerías

React

Librería JavaScript de código abierto diseñada para crear interfaces de usuario con el objetivo de facilitar el desarrollo de aplicaciones en una sola página. Es mantenido por Meta y la comunidad de software libre.

Esta ha sido empleada para el desarrollo del Front-End.

Bcrypt

Librería para la aplicación de funciones hash a contraseñas y derivación de claves para contraseñas basadas en el cifrado Blowfish. La función de derivaciones de claves que usa esta es lenta, lo que dificulta la obtención de la contraseña por la fuerza bruta.

Bcrypt ha sido empleada para la encriptación de las contraseñas de los usuarios.

Zod

Librería de validación y declaración de esquemas de TypeScript, recogiendo los esquemas una amplia gama de posibilidades, desde una cadena de texto a un objeto entero. Esta librería permite la verificación de los tipos de un objeto.

Zod ha sido empleada para la verificación de los datos recibidos en el Back-End previa a la utilización de estos.

React-tooltips

Librería de creación de tooltips, elementos encargados de mostrar texto informativo sobre un elemento.

React-tooltips ha sido empleado para inserción de nombres de iconos en el Front-End.

Just

Librería de componentes funcionales de React.

Just ha sido empleado para la utilización de un debounce en la barra de búsqueda de clases.

JsonWebToken

Librería para la implementación y gestión de tokens de sesión JWT.

JsonWebToken ha sido empleada para el control de los tokens de sesiones.

7. Especificación y análisis de requisitos

En este apartado se mostrarán primero los diferentes roles de la aplicación, los requisitos funcionales en forma de historias de usuarios y los requisitos no funcionales. Esto fue creado a partir de los objetivos previamente indicados.

Por una parte, los roles que han sido identificados serían los siguientes:

- **Administrador:** usuario encargado de la gestión de los diferentes usuarios de la aplicación, es decir, gestionar qué usuarios actúan como monitores o como clientes, además de poder validar los clientes. Este usuario no tendrá acceso al resto de las funcionalidades de la aplicación.
- **Monitor:** usuario encargado de la gestión de las clases (creación, edición y borrado de estas), así como la gestión de asistencia de las clases. Este usuario no podrá acceder a las funcionalidades del cliente.
- **Cliente:** este usuario podrá inscribirse y desinscribirse a las diferentes clases. No podrá acceder a las funcionalidades del monitor.

Por otra parte, las historias de usuario están compuestas por un nombre breve, la descripción de la funcionalidad y las diferentes pruebas de aceptación que se deben cumplir para que se considere correcto su implementación.

HU01 – Autenticación de Usuario

| | |
|------------------------------|---|
| Como | Usuario |
| Quiero | Registrarme e iniciar sesión |
| Para | Poder hacer uso de las funcionalidades de la aplicación |
| Pruebas de aceptación | <ul style="list-style-type: none">• Se debe poder realizar el registro introduciendo un correo y una contraseña• Si los datos son correctos, el usuario es redirigido a la página de inicio de sesión• Una vez registrado el usuario, debe poder iniciar sesión con un correo y contraseña para poder acceder a la parte privada de la aplicación |

HU02 – Gestión de clases

| | |
|------------------------------|---|
| Como | Monitor |
| Quiero | Poder añadir, modificar y eliminar clases creadas por mí mismo |
| Para | Poder gestionar las diferentes clases que imparto |
| Pruebas de aceptación | <ul style="list-style-type: none">• Se debe poder añadir, modificar y eliminar clases creadas por el monitor• En el caso de borrado, se deben eliminar las inscripciones de todos los clientes que estuvieran inscritos a esta• A la hora de realizar una modificación, el aforo no puede ser inferior al número total de clientes inscritos a la clase |

HU03 – Gestión de sesiones como creador

| | |
|------------------------------|--|
| Como | Monitor |
| Quiero | Poder gestionar las sesiones de las clases que creo |
| Para | Poder gestionar las diferentes sesiones |
| Pruebas de aceptación | <ul style="list-style-type: none">• Se debe poder añadir sesiones para cada clase, indicando el día y la hora• Se debe poder editar y borrar sesiones• Se debe poder marcar la asistencia y la justificación de las faltas de asistencia |

HU04 – Gestión de sesiones como instructor

| | |
|---------------|--|
| Como | Monitor |
| Quiero | Poder gestionar las sesiones en las que estoy asignado como instructor |
| Para | Poder gestionar las diferentes sesiones |

| | |
|------------------------------|---|
| Pruebas de aceptación | <ul style="list-style-type: none">• Se debe poder marcar la asistencia y la justificación de las faltas de asistencia |
|------------------------------|---|

HU05 – Gestión de inscripciones

| | |
|------------------------------|--|
| Como | Cliente |
| Quiero | Inscribirme y desinscribirme |
| Para | Poder asistir a las diferentes clases a las que esté inscrito |
| Pruebas de aceptación | <ul style="list-style-type: none">• Se debe poder inscribir y desinscribir a las diferentes clases |

HU06 – Gestión de usuarios

| | |
|------------------------------|---|
| Como | Administrador |
| Quiero | Poder modificar los roles que identifican a cada usuario, así como las contraseñas, la eliminación de usuarios y creación de nuevos |
| Para | Poder gestionar las acciones de los usuarios |
| Pruebas de aceptación | <ul style="list-style-type: none">• Se debe poder modificar los roles y contraseñas de cada usuario en el sistema a administrador, monitor y cliente.• Al realizar las modificaciones, en caso de que tenga alguna clase creada o clase a la que esté inscrito y cambia de rol de cliente a instructor o administrador o de instructor a cliente o administrador, se eliminarán todas las anteriores• Se debe poder eliminar usuarios. Todos aquellos elementos que dependan de este serán borrados• Se debe poder crear usuarios. |

HU07 – Consulta de clases

| | |
|------------------------------|--|
| Como | Cliente |
| Quiero | Buscar clases |
| Para | Poder ver las diferentes clases a las que puedo inscribirme |
| Pruebas de aceptación | <ul style="list-style-type: none">• Podrá realizar una búsqueda por nombre |

HU08– Validación Cliente

| | |
|------------------------------|---|
| Como | Administrador |
| Quiero | Validar a los clientes |
| Para | Que los clientes puedan acceder a las funcionalidades |
| Pruebas de aceptación | <ul style="list-style-type: none">• Se debe poder validar cada cliente• Se debe poder quitar la validación para cada cliente |

HU09 – Duplicación Sesión

| | |
|------------------------------|---|
| Como | Monitor |
| Quiero | Duplicar una sesión |
| Para | Repetir sesiones sin tener que crearlas completamente a mano |
| Pruebas de aceptación | <ul style="list-style-type: none">• Se debe poder duplicar una sesión, siendo esta creada en un número de días determinados por el usuario, posteriores a la fecha de la sesión |

Respecto a los requisitos no funcionales, la interfaz debe ser intuitiva, sencilla y cómoda al usuario, haciendo foco en la utilidad y facilidad de uso frente a un mayor nivel visual pero que afecte a la usabilidad de la aplicación.

8. Diseño del software

En este apartado se expondrá el diseño de la aplicación web desde dos puntos de vista, la vista estática y la vista dinámica. La vista estática se encarga de mostrar la estructura que forma la aplicación, mientras que la vista dinámica explica cómo se comporta la aplicación. Estas dos vistas se expondrán de forma simplificada.

8.1. Vista estática

Para la organización del desarrollo de una aplicación, uno de sus puntos fundamentales es la organización de los archivos que componen esta. La estructura de archivos elegida sería la siguiente:

Gestión web de un gimnasio

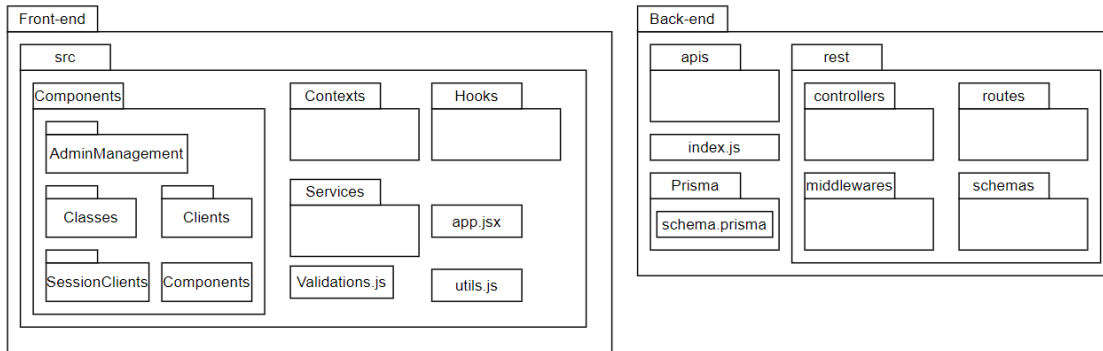


Ilustración 6: ESTRUCTURA DE ARCHIVOS

En la parte del Front-End, la distribución ha sido elegida para tener una mayor separación de los diferentes componentes de la aplicación y permitir la mayor reutilización posible de estos, además de poder extraer toda la lógica posible de los componentes encargados de mostrar algún elemento visual. En Components se almacenan todos los componentes visuales, habiéndose hecho pequeñas agrupaciones en aquellos componentes altamente relacionados. En la parte del Back-End, la distribución mostrada ha sido seleccionada para tener una mayor separación de funciones, así como para poder relacionar de forma sencilla qué funcionalidades puede tener cada archivo.

En la siguiente imagen podemos observar el diagrama de clases del Back-End. En este podemos las clases de las que nacen todas son Router, Controller y Model. Estos tres interactúan entre sí. En el caso del Router, este emplea las funciones del Controller y, a su vez, este usa las funciones del Model.

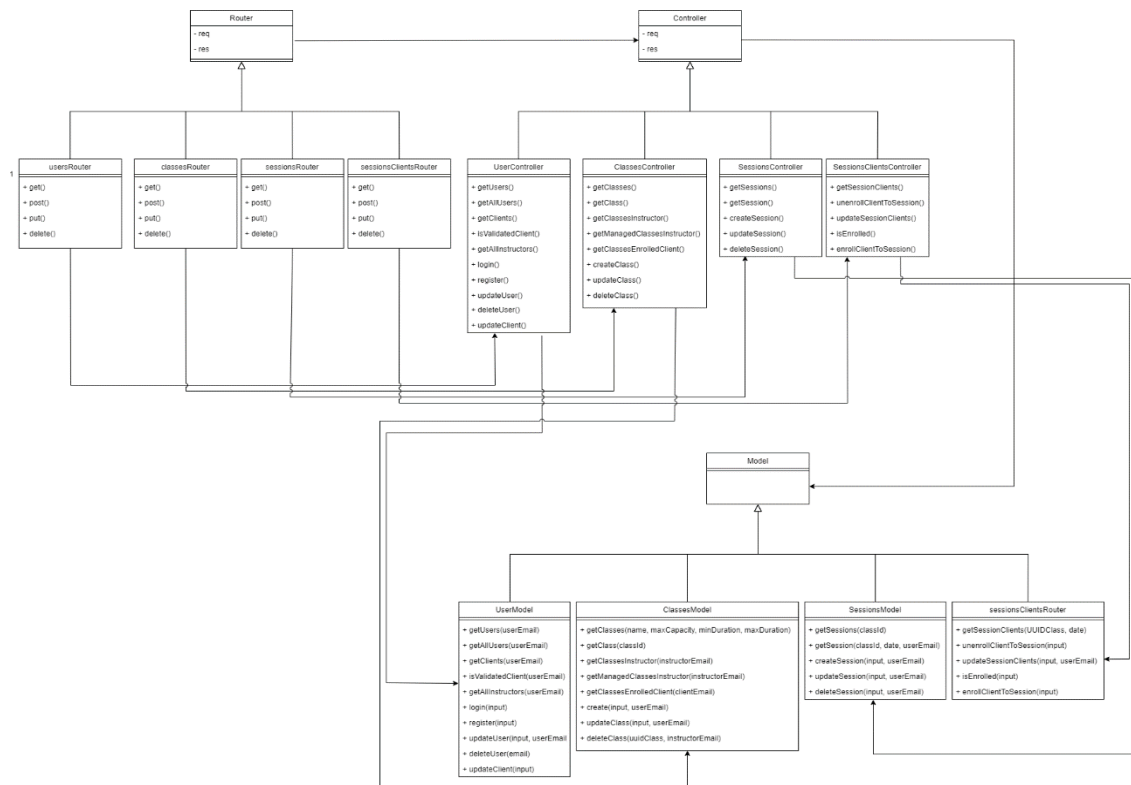


Ilustración 7: DIAGRAMA CLASES DEL BACK-END

En el caso del Front-End, al tener un tamaño muy superior al del Back-End, se ha decidido la creación de un diagrama de clases para la gestión de clases. Este diagrama de clases está compuesto por 4 entidades principales de las que heredan el resto:

- **Context:** guardan el estado de la aplicación que se comparte entre los diferentes componentes.
- **Component:** encargado de mostrar la parte visual de la aplicación y de tener la lógica mínima necesaria para su correcto funcionamiento.
- **CustomHook:** encargado de realizar la lógica de negocio de la aplicación.
- **Service:** realiza las peticiones HTTP.

A demás de las clases anteriores, cada implementación de las clases tiene sus propios métodos y atributos específicos.

Gestión web de un gimnasio

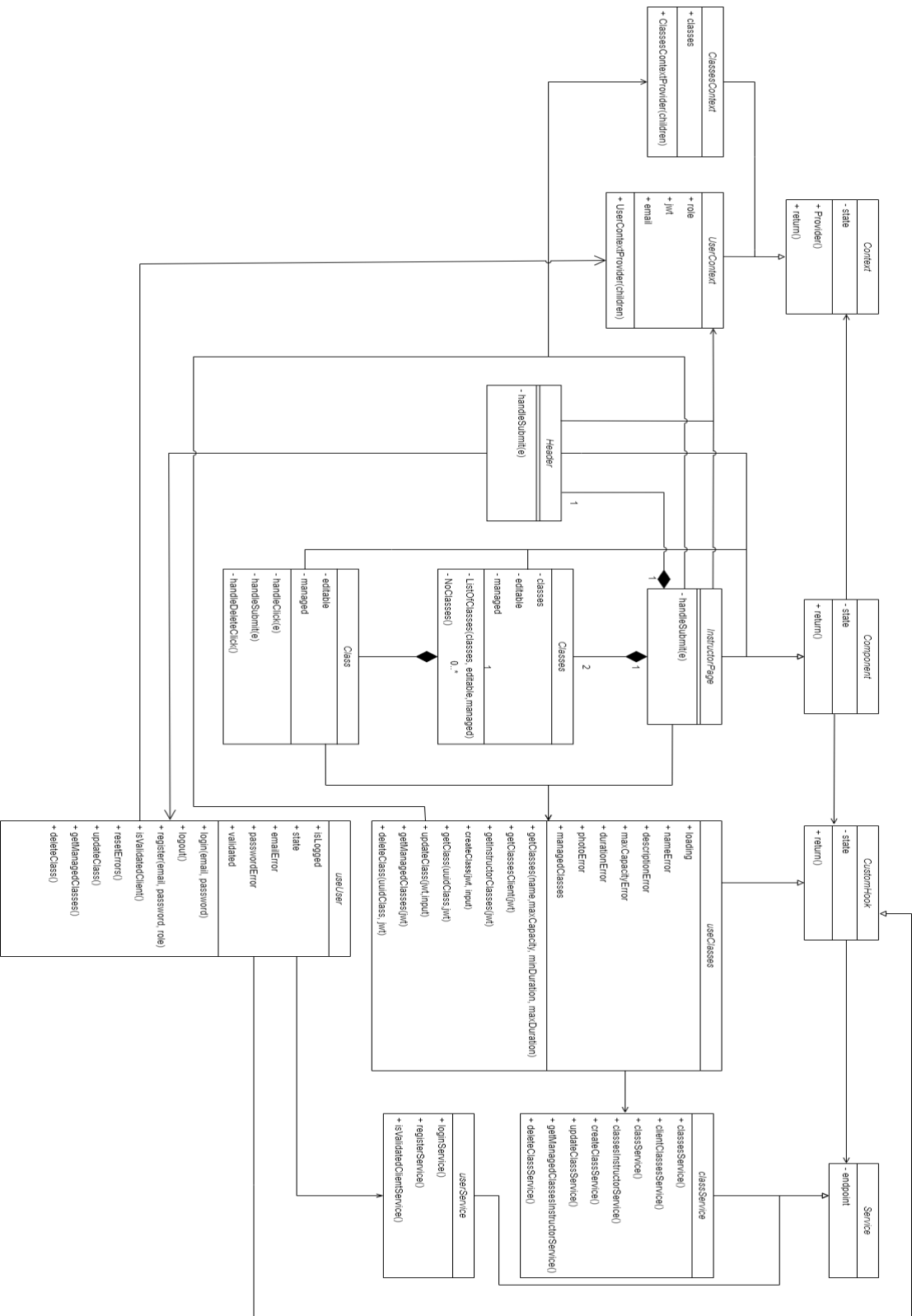


Ilustración 8: DIAGRAMA DE CLASES DEL FRONT-END

Lo que podemos observar en la ilustración es la relación de la página de instructores con el resto de los componentes. Esta está siendo renderizada por React y, a su vez, esta almacena diferentes instancias del resto de componentes, siendo, en este caso, de Header y de Classes que, a su vez, también contiene dos tipos de componentes, renderizándose NoClasses si no hubiera clases y ListOfClasses si hubiera clases, siendo este componente una lista de instancias del componente Class.

8.2. Vista dinámica

Debido a la gran cantidad de acciones que compone la aplicación, se decidió la realización de un único diagrama de secuencias que ejemplifique de manera genérica cómo se comportan las acciones, el flujo de estas y cómo reaccionan estas con el sistema.

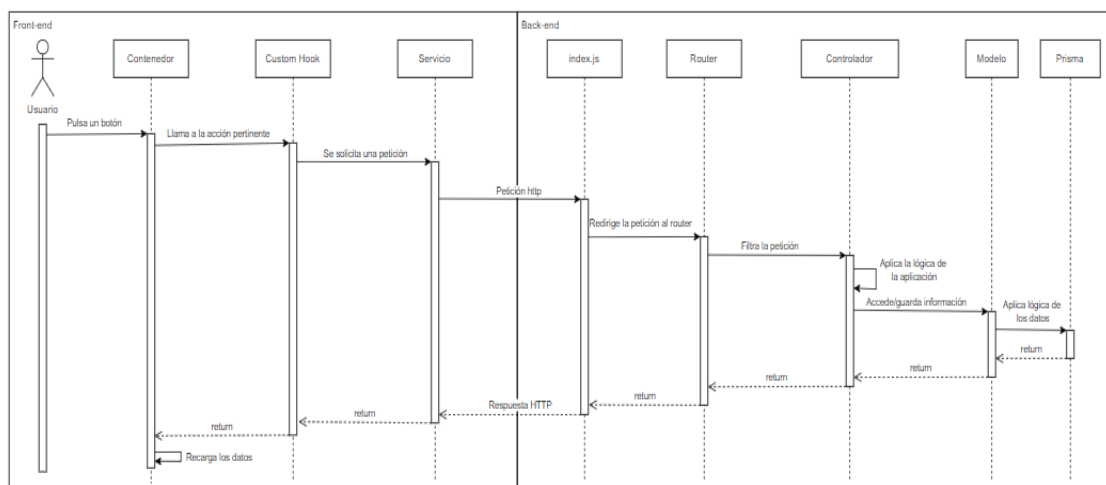


Ilustración 9: DIAGRAMA DE SECUENCIA DEL SISTEMA

9. Gestión de datos e información

Para la gestión de los datos de la aplicación ha sido decidido el empleo de MySQL, una base de datos relacional, debido a su previo empleo en la carrera y además el alumno ha profundizado aún más en las prácticas curriculares realizadas. Las bases de datos relacionales están formadas por tres elementos fundamentales:

- **Entidades:** personas, objetos o conceptos de los que trata una base de datos, siendo los elementos mínimos necesarios para la construcción de la misma.
- **Atributos:** describen las propiedades que posee cada entidad.
- **Relaciones:** vínculos creados entre distintas entidades.

En complemento de MySQL, se ha empleado Prisma, un ORM. Este provee una abstracción entre la base de datos y la aplicación. Con Prisma las entidades y relaciones se crean mediante esquemas, que son objetos similares a los de JavaScript y junto lo que se conoce como migraciones, que vendrían a ser las distintas modificaciones que se realizan a la base de datos a partir del esquema previamente creado. Cada vez que se realiza un cambio en el esquema se realiza una migración, siendo esta aquella que guarda los cambios producidos. Esto lo que

produce es un historial de cambios que se aplican cada vez que la aplicación es desplegada, ya sea en modo desarrollo como en modo producción.

En la siguiente ilustración podemos observar el modelo de la base de datos.

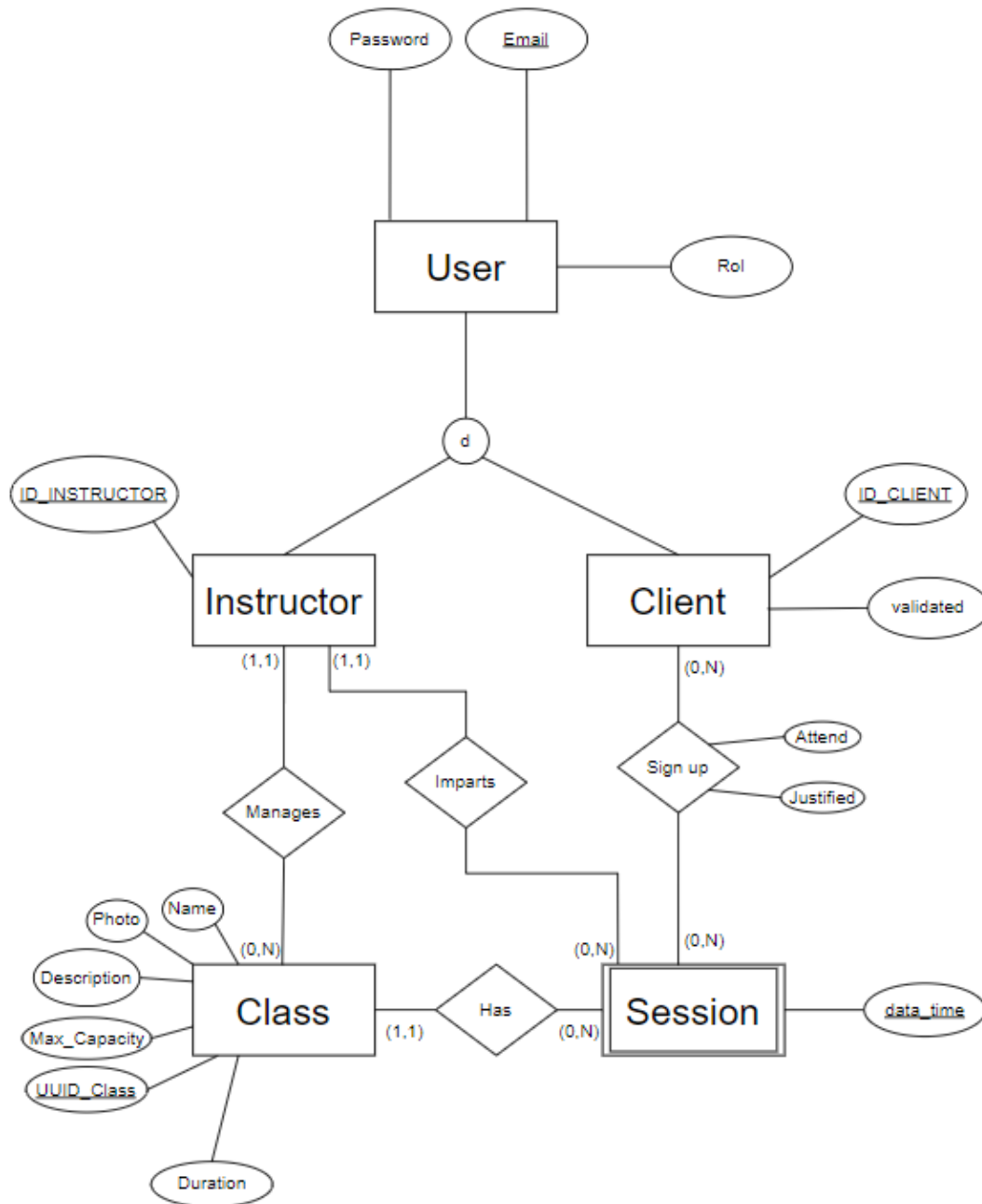


Ilustración 10: MODELO ENTIDAD/RELACIÓN EXTENDIDO

Por otra parte, durante el desarrollo, el alojamiento del MySQL se ha realizado mediante el uso de XAMPP, debido a la simpleza que este nos ofrece para el mantenimiento y la instalación del MySQL.

Para concluir con este punto, se mostrará una tabla con la información de las URLs de los recursos REST, formada por la URL, una descripción, el método HTTP(Get,Post,Put o Delete) y lo que devuelve.

| URL | Descripción | HTTP | Retorno |
|---------------------------|--|--------|--------------------------------------|
| /users/ | Devuelve todos los usuarios | Get | [{}, {}, ...] |
| /users/all | Devuelve todos los administradores, clientes e instructores | Get | {admins:[],clients:[],instructors[]} |
| /users/instructors/ | Devuelve todos los instructores | Get | [{}, {}, ...] |
| /users/clients/ | Devuelve todos los clientes | Get | [{}, {}, ...] |
| /users/clients/validation | Devuelve si el usuario actual está registrado | Get | True False |
| /users/login | Inicia sesión | Post | {jwt,role } |
| /users/register | Registra a un usuario | Post | Created |
| /users/ | Actualizar un usuario | Put | Updated |
| /users/clients/ | Actualizar un cliente | Put | Updated |
| /users/:email | Elimina un usuario | Delete | <u>Deleted</u> |
| /classes/ | Devuelve todas las clases | Get | [{}, {}, ...] |
| /classes/instructor/ | Devuelve todas las clases creadas por un instructor | Get | [{}, {}, ...] |
| /classes/client/ | Devuelve todas las clases a las que está inscrito por lo menos a una sesión el cliente | Get | [{}, {}, ...] |
| /classes/sessions/ | Devuelve todas las clases en las que el instructor gestiona por lo menos una sesión | Get | [{}, {}, ...] |
| /classes/:classId | Devuelve la clase asociada al id dado | Get | {} |
| /classes/ | Inserta una nueva clase | Post | Created |
| /classes/ | Actualiza una clase | Put | Updated |
| /classes/:class_id | Elimina la clase asociada al id dado | Delete | Deleted |
| /sessions/:classId | Devuelve las sesiones asociadas a la clase aportada | Get | [{}, {}, ...] |
| /sessions/:classId/:date | Devuelve la sesión asociada a la clase y la fecha aportadas | Get | {} |

| URL | Descripción | HTTP | Retorno |
|--|---|--------|---------------|
| /sessions/ | Inserta una nueva sesión | Post | Created |
| /sessions/ | Actualiza una sesión | Put | Updated |
| /sessions/ | Elimina una sesión | Delete | Deleted |
| /sessionsClients/:UUIDClass/:date | Devuelve las inscripciones dada una sesión | Get | [{}, {}, ...] |
| /sessionsClients/:UUIDClass/:date | Inscribe un cliente a una sesión | Post | Enrolled |
| /sessionsClients/:UUIDClass/:date/isEnrolled | Devuelve si un cliente está inscrito a una sesión | Post | True False |
| /sessionsClients/:UUIDClass/:date | Actualiza una inscripción | Put | Updated |
| /sessionsClients/:UUIDClass/:date | Elimina la inscripción de un cliente a una sesión | Delete | Deleted |

Tabla 3: API

10. Pruebas

Con el objetivo de garantizar el correcto funcionamiento de la aplicación, se han ido realizando un gran número de pruebas de forma paralela al desarrollo, teniendo como base los criterios de aceptación previamente definidos en cada una de las historias de usuario y verificando el cumplimiento de los requisitos de manera exitosa. Este tipo de pruebas se centran únicamente en las respuestas generadas ante la interacción con la aplicación.

HU01 – Autenticación de Usuario

| Funcionalidad | Registro de Usuario |
|-----------------------|--|
| Precondiciones | El usuario ha introducido el email, la contraseña y la contraseña repetida correctamente |
| Ejecución | El usuario pulsa el botón “Registrarse” |
| Resultado | El usuario es registrado y redirigido a la página de login para que inicie sesión |
| Evaluación | Completada de forma correcta |

| Funcionalidad | Inicio de sesión del Usuario |
|-----------------------|--|
| Precondiciones | El usuario ha introducido el email y la contraseña correctamente |
| Ejecución | El usuario pulsa el botón “Iniciar Sesión” |
| Resultado | En caso de que los datos sean correctos, es redirigido a la parte privada de la aplicación, accediendo a una parte u otra en función del rol que tenga |
| Evaluación | Completada de forma correcta |

HU02 – Gestión de clases

| Funcionalidad | Creación de clases |
|-----------------------|---|
| Precondiciones | El instructor pulsa el botón “Add” |
| Ejecución | El instructor introduce los datos de la clase y pulsa el botón “Crear” |
| Resultado | Se crea la clase, se muestra un mensaje de confirmación y se muestra la nueva clase creada junto con el resto |
| Evaluación | Completada de forma correcta |

| Funcionalidad | Edición de clases |
|-----------------------|---|
| Precondiciones | El instructor pulsa el icono con forma de controles deslizantes |
| Ejecución | El instructor modifica los datos de la clase y pulsa el botón “Guardar” |
| Resultado | Se edita la clase, se muestra un mensaje de confirmación y se muestran los datos de la clase editada junto con el resto |
| Evaluación | Completada de forma correcta |

| Funcionalidad | Borrado de clases |
|-----------------------|--|
| Precondiciones | El instructor pulsa el icono forma de X |
| Ejecución | El instructor confirma la eliminación de la clase |
| Resultado | Se borra la clase, se muestra un mensaje de confirmación y se muestran el resto de las clases. |
| Evaluación | Completada de forma correcta |

HU03 – Gestión de sesiones como creador

| Funcionalidad | Creación de sesiones |
|-----------------------|---|
| Precondiciones | El instructor pulsa el botón “Añadir Sesión” |
| Ejecución | El instructor introduce los datos correctos de la sesión |
| Resultado | Se crea la sesión, se muestra un mensaje de confirmación y se muestran la sesión recién creada junto con el resto de las sesiones |
| Evaluación | Completada de forma correcta |

| Funcionalidad | Edición de sesiones |
|-----------------------|--|
| Precondiciones | El instructor pulsa el icono con forma de controles deslizantes |
| Ejecución | El instructor modifica los datos de la sesión |
| Resultado | Se edita la sesión, se muestra un mensaje de confirmación y se muestra la sesión recién editada junto con el resto de las sesiones |
| Evaluación | Completada de forma correcta |

| Funcionalidad | Borrado de sesiones |
|-----------------------|--|
| Precondiciones | El instructor pulsa el icono con forma de X |
| Ejecución | El instructor confirma la eliminación de la sesión |
| Resultado | Se borra la sesión, se muestra un mensaje de confirmación y se muestran las sesiones |
| Evaluación | Completada de forma correcta |

HU04 – Gestión de sesiones como instructor

| Funcionalidad | Confirmación de asistencia de cliente a sesión |
|-----------------------|---|
| Precondiciones | El instructor pulsa el botón “Ver” de la sesión desde las clases del apartado de clases gestionadas |
| Ejecución | El instructor pulsa el checkbox asociado a la columna “Attend” del cliente |
| Resultado | Se cambia el estado de la casilla y se muestra un mensaje de confirmación |
| Evaluación | Completada de forma correcta |

| Funcionalidad | Confirmación de justificación de una falta de un cliente a una sesión |
|-----------------------|---|
| Precondiciones | El instructor pulsa el botón “Ver” de la sesión desde las clases del apartado de clases gestionadas |
| Ejecución | El instructor pulsa el checkbox asociado a la columna “Justified” del cliente |
| Resultado | Se cambia el estado de la casilla y se muestra un mensaje de confirmación |
| Evaluación | Completada de forma correcta |

HU05 – Gestión de inscripciones

| Funcionalidad | Inscripción a una sesión |
|----------------|---|
| Precondiciones | El cliente debe estar en la página de alguna clase |
| Ejecución | El cliente pulsará el botón “Inscribirse” |
| Resultado | Se inscribirá el cliente a la sesión, se muestra un mensaje de confirmación y pasará a mostrarse el botón de desinscribirse |
| Evaluación | Completada de forma correcta |

| Funcionalidad | Desinscripción a una sesión |
|----------------|---|
| Precondiciones | El cliente debe estar en la página de alguna clase |
| Ejecución | El cliente pulsará el botón “Desinscribirse” |
| Resultado | Se desinscribirá el cliente a la sesión, se muestra un mensaje de confirmación y pasará a mostrarse el botón de inscribirse |
| Evaluación | Completada de forma correcta |

HU06 – Gestión de usuarios

| Funcionalidad | Borrado de usuarios |
|----------------|---|
| Precondiciones | El administrador pulsa el botón “Eliminar” |
| Ejecución | El administrador confirma la eliminación |
| Resultado | Se eliminará el usuario y todo elemento en la base de datos que tengan alguna relación con este, se mostrará un mensaje de verificación y se mostrarán el resto de los usuarios |
| Evaluación | Completada de forma correcta |

| Funcionalidad | Edición de usuarios |
|----------------|--|
| Precondiciones | El administrador pulsa el botón “Editar” |
| Ejecución | El administrador modifica los datos que considera pertinentes |
| Resultado | Se editará el usuario, se mostrará un mensaje de verificación y se mostrarán el usuario actualizado junto con el resto de los usuarios |
| Evaluación | Completada de forma correcta |

| Funcionalidad | Creación de usuarios |
|-----------------------|---|
| Precondiciones | El administrador pulsa el botón “Añadir usuario” |
| Ejecución | El administrador introducirá los datos necesarios y pulsará el botón “Añadir” |
| Resultado | Se creará el usuario, se mostrará un mensaje de confirmación y se mostrarán el usuario recién creado junto con el resto de los usuarios |
| Evaluación | Completada de forma correcta |

HU07 – Consulta de clases

| Funcionalidad | Búsqueda de sesiones |
|-----------------------|---|
| Precondiciones | El cliente pulsa el botón “Búsqueda” |
| Ejecución | El cliente introducirá los datos necesarios y pulsará el botón “Buscar” |
| Resultado | Se mostrarán las clases acordes a los filtros introducidos por el cliente |
| Evaluación | Completada de forma correcta |

HU08 – Validación Cliente

| Funcionalidad | Validación de clientes |
|-----------------------|---|
| Precondiciones | El administrador pulsa el botón “Lista de clientes” |
| Ejecución | El administrador pulsa el checkbox que no está activo, correspondiente a la columna “Validado” |
| Resultado | Se cambiará el estado del checkbox, se mostrará un mensaje de confirmación de la validación y se mostrará la tabla de nuevo |
| Evaluación | Completada de forma correcta |

| Funcionalidad | Invalidación de clientes |
|-----------------------|---|
| Precondiciones | El administrador pulsa el botón “Lista de clientes” |
| Ejecución | El administrador pulsa el checkbox activo correspondiente a la columna “Validado” |
| Resultado | Se cambiará el estado del checkbox, se mostrará un mensaje de confirmación de la invalidación y se mostrará la tabla de nuevo |
| Evaluación | Completada de forma correcta |

HU09 – Duplicación Sesión

| Funcionalidad | Duplicación de sesiones |
|-----------------------|---|
| Precondiciones | El instructor pulsa el icono con forma de dos rectángulos superpuestos y un más |
| Ejecución | El instructor introduce el número de días en el que se duplicará la sesión |
| Resultado | Se duplicará la sesión, se mostrará un mensaje de confirmación y se mostrará la sesión recién creada junto con el resto |
| Evaluación | Completada de forma correcta |

11. Manual de usuario

En los siguientes puntos se mostrarán detalladamente los pasos necesarios para la instalación y los requisitos mínimos para su uso, así como un manual de usuario donde se explica cómo se utiliza la aplicación.

11.1 Requisitos mínimos

Desarrollador

Para desarrollar, los requisitos mínimos de hardware serían los mínimos necesarios para tener tanto NodeJS como para un MySQL levantado. A nivel de producción variaría en función del tamaño y el flujo de clientes que tenga la aplicación.

A nivel de software, previamente se debe tener instalado NodeJS en la versión 20.10.0, así como el instalador de paquetes pnpm en la versión 8.12.1 o el instalador de paquetes npm en la versión 10.2.3. Además de esto es necesario tener un MySQL. Se recomienda a nivel de desarrollo el uso de XAMPP.

Usuario

Los usuarios acceden a esta aplicación mediante un navegador, así que el único requisito sería disponer de un navegador, ya sea Firefox, Opera o Google Chrome, navegadores en los que ha sido probada la aplicación.

11.2 Instalación

Desarrollador

Para la puesta en funcionamiento de la aplicación, será necesario levantar por un lado el Front-End y, por otro lado, el Back-End.

Para el Front-End necesitaremos acceder a `.\front\GymClass` e instalar todas las dependencias y paquetes de este. Para ello utilizaremos el comando: **pnpm install**. Una vez instaladas todas las dependencias, para poner en funcionamiento el Front-End, usaremos el comando: **pnpm run dev**. Aquí es posible usar tanto pnpm como npm, pero se recomienda el uso de pnpm por su rapidez.

Para el Back-End, previamente es necesario tener un servidor MySQL en funcionamiento con una base de datos en blanco, junto con un usuario que tenga permisos de administrador. Una vez tengamos esto, necesitaremos cambiar en el archivo `.env` de la carpeta `.\back` la URL de la base de datos. Esta será del siguiente tipo: `mysql://usuario:contraseña@urlDeLaBaseDeDatos/nombreDeLaBaseDeDatos`. Tras todo esto, podemos realizar ya desde la carpeta `.\back` el comando: **pnpm install**, que instalará todas las dependencias necesarias, y **pnpm install prisma**. Tras instalar prisma, habrá que realizar las migraciones de la base de datos. Para ello usaremos **pnpm prisma migrate dev**. Una vez acabadas las migraciones e instaladas todas las dependencias, para poner en funcionamiento el Back-End, usaremos el comando: **pnpm run dev**. Al igual que en el Front-End, se pueden emplear tanto pnpm como npm. Para la ejecución del migrate con npm habrá que emplear **npm prisma migrate dev**.

Usuario

El usuario no necesita realizar ninguna instalación. Este podrá acceder a la aplicación entrando en la URL al navegador. Tal y como está desplegada actualmente la aplicación, debería acceder a la url: <http://localhost:5173/>. El puerto variará en función de si el desarrollador tiene este ocupado o no.

11.3 Funcionamiento

Para lograr una mejor comprensión de la aplicación y de su funcionamiento, en este apartado se mostrarán las diferentes secciones que componen nuestra aplicación.

Las diferentes secciones serían las siguientes:

- **Sección de Inicio:** página de inicio.
- **Sección de Autenticación:** autenticación y registro de usuarios.
- **Sección de Administración:** gestión del administrador de los usuarios.
- **Sección de Instrucción:** gestión de clases y sesiones por parte del instructor.
- **Sección de Cliente:** gestión de inscripciones a sesiones de las diferentes clases.

Sección de Inicio

Lo primero que encontramos al abrir la aplicación sería la Landing Page, página en la que veríamos un número limitado de clases y desde la que podremos acceder a la autenticación y, en caso de que estemos autenticados, podremos acceder a las diferentes clases y funcionalidades, siendo redirigidos a una página u otra en función del rol que tengamos. En caso de estar autenticados ya, al pulsar en “Iniciar Sesión”, seremos llevados a la zona pertinente al rol del usuario.

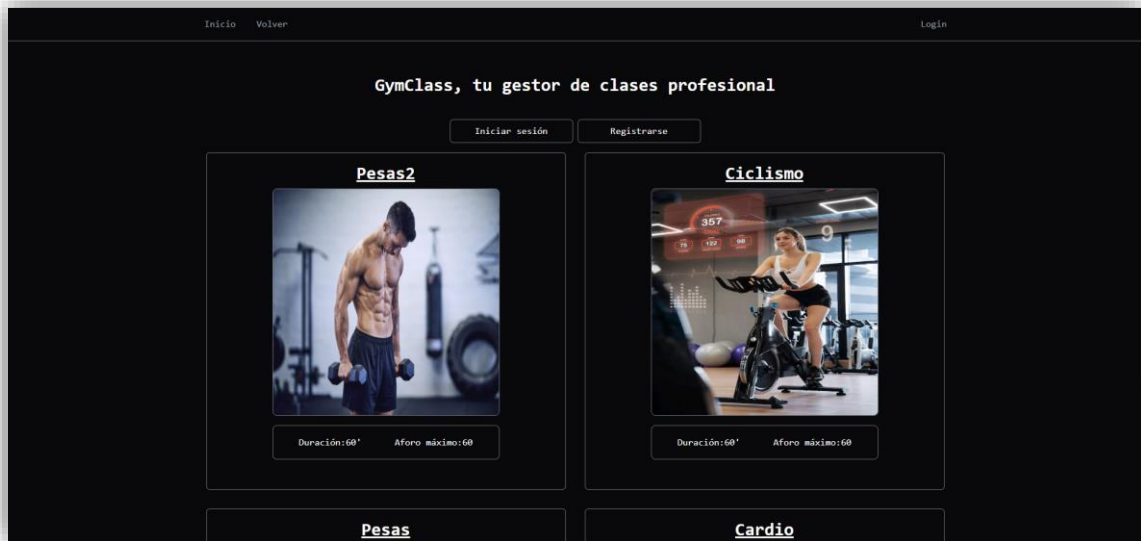


Ilustración 11: SECCIÓN DE INICIO

Sección de Autenticación

Si accedemos a registro o a inicio de sesión, veremos un formulario para iniciar sesión en caso de que se acceda a esta y otro para registrarse en el caso contrario. Si tiene una cuenta puede entrar directamente mediante el inicio de sesión. En caso contrario, será necesario que se registre para poder acceder. Por defecto, mediante este registro será registrado como cliente

Ilustración 12: FORMULARIO DE INICIO DE SESIÓN

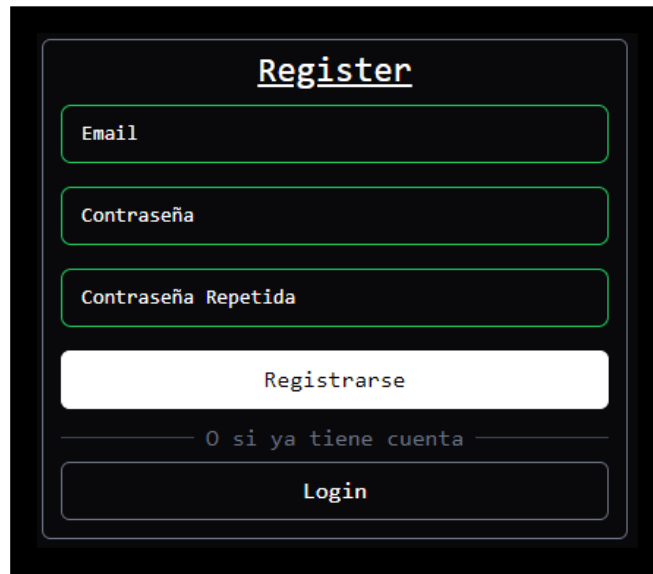
A dark-themed registration form titled "Register" in a light blue font. It features three input fields with light blue borders: "Email", "Contraseña", and "Contraseña Repetida". Below these fields is a light blue button labeled "Registrarse". Underneath the button is a link that says "O si ya tiene cuenta" in a light blue font. At the bottom is a dark button labeled "Login" in a light blue font.

Ilustración 13: FORMULARIO DE REGISTRO

Sección de Administración

Una vez se inicie sesión con una cuenta cuyo rol sea administrador, será redirigido a la página de inicio de los administradores. Desde esta página podrá acceder a la gestión de los usuarios.

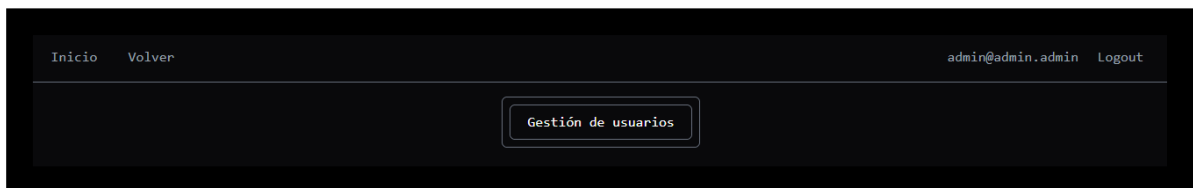
A dark-themed administrator dashboard. At the top, there is a header bar with "Inicio" and "Volver" on the left, and "admin@admin.admin" and "Logout" on the right. Below the header is a large dark area with a single button labeled "Gestión de usuarios" in the center.

Ilustración 14: PÁGINA DE INICIO DE ADMINISTRADOR

Una vez se acceda a la página de gestión de usuarios, se podrá añadir nuevos usuarios, editar y borrar los existentes y acceder a la gestión de los clientes.



Ilustración 15: PÁGINA DE GESTIÓN DE USUARIOS

Una vez se acceda a la lista de clientes, será posible realizar las mismas acciones que con el resto de los usuarios con el añadido de poder validar los clientes.

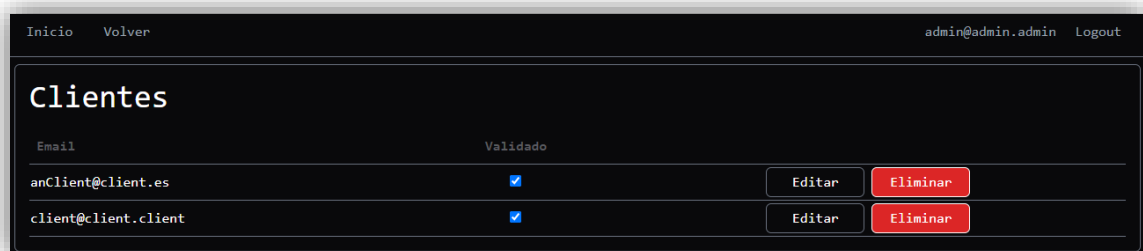


Ilustración 16: PÁGINA DE GESTIÓN DE CLIENTES

Sección de Instrucción

En caso de que se inicie sesión con una cuenta cuyo rol sea instructor, se accederá a la página de inicio de los instructores, página que estará dividida en dos partes. Por un lado, se verán aquellas clases que hayan sido creadas por el usuario actual, así como editar y borrar estas y poder añadir nuevas clases.

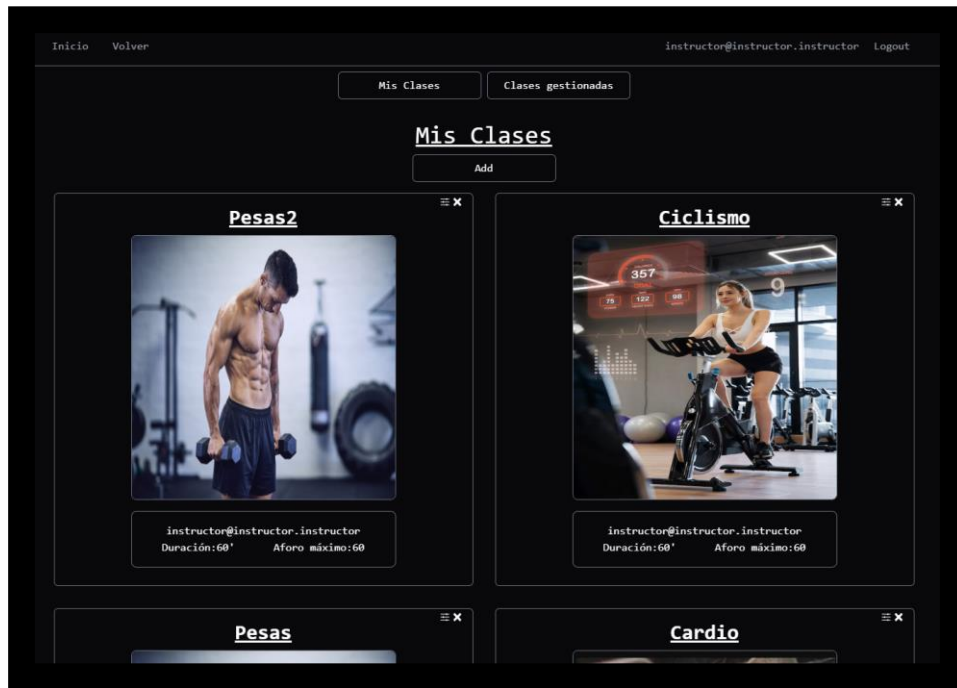


Ilustración 17: PÁGINA DE GESTIÓN DE CLASES PROPIAS

Por otro lado, se puede acceder a aquellas clases en las que existe al menos una sesión en la que el instructor sea el usuario actual.

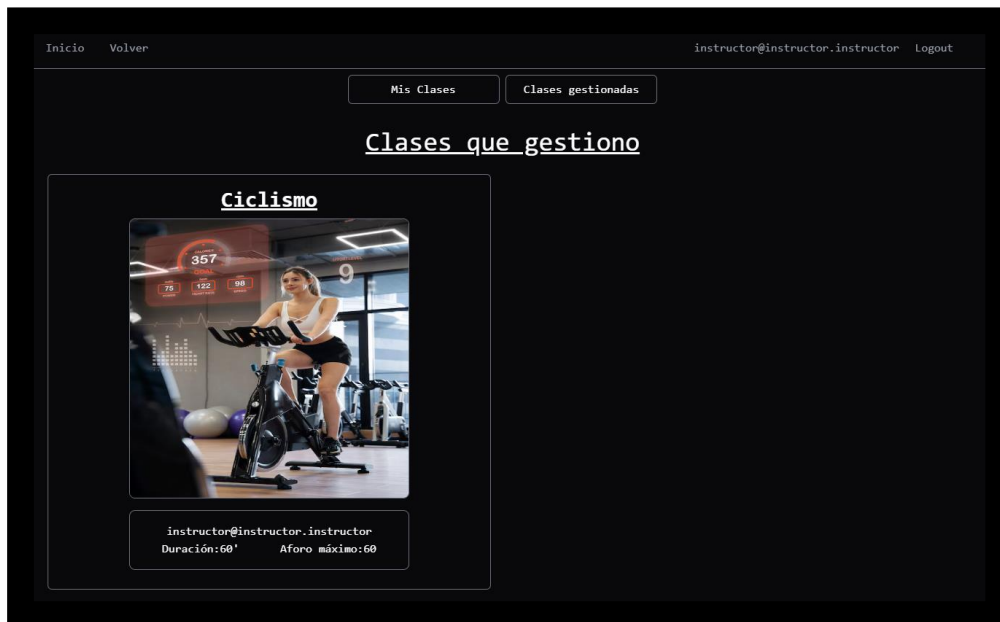


Ilustración 18: PÁGINA DE GESTIÓN DE CLASES GESTIONADAS

Si se accede a las sesiones de una clase propia pulsando el nombre de la clase se accederá a la página de gestión de las sesiones de esta clase, donde se podrá crear, editar, duplicar y eliminar sesiones. Estas estarán separadas en grupos por días de la semana, mostrándose de forma

Gestión web de un gimnasio

predeterminada las sesiones cuya fecha es superior a la actual, preferencia que puede ser modificada pulsando el botón “Sin Impartir”, botón que hará que se muestren todas las sesiones, tanto las que aún no ocurrieron como las ya han sido realizadas.

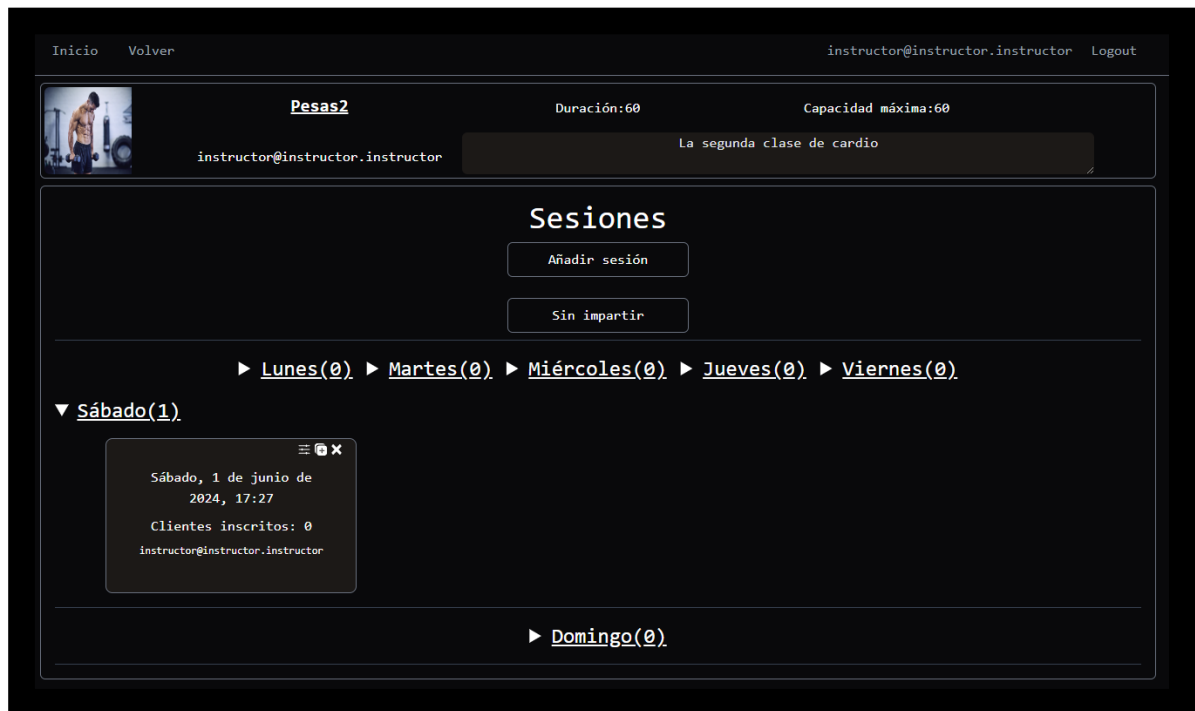


Ilustración 19: PÁGINA DE GESTIÓN DE SESIONES DE CLASES PROPIAS

Si, por otro lado, se entra a las sesiones desde el apartado de clases gestionadas, se podrán ver aquellas sesiones cuyo instructor es el usuario actual, así como se podrá acceder a la lista de clientes inscritos a cada sesión.



Ilustración 20: PÁGINA DE GESTIÓN DE CLASES GESTIONADAS

Una vez se acceda a los clientes inscritos, se verá tanto el email de los clientes como el estado de su asistencia y justificación, así como poder modificar estos dos últimos.

The screenshot shows a web interface for managing registered clients for a specific class. At the top, there are links for 'Inicio' and 'Volver', and the user 'instructor@instructor.instructor' is logged in. The main section is titled 'Ciclismo' and displays 'Duración:60' and 'Capacidad máxima:60'. Below this, the instructor's email 'instructor@instructor.instructor' is shown next to a text input field containing 'Clase de ciclismo intensivo donde se realizará'. A summary bar indicates the date 'Martes, 28 de mayo de 2024, 20:53', 'Clientes inscritos:1', and the instructor's email. Below this is a table with columns 'Email', 'Attend', and 'Justified'. One client is listed with email 'client@client.client', 'Attend' status checked, and 'Justified' status unchecked.

| Email | Attend | Justified |
|----------------------|-------------------------------------|--------------------------|
| client@client.client | <input checked="" type="checkbox"/> | <input type="checkbox"/> |

Ilustración 21: PÁGINA DE GESTIÓN DE CLIENTES INSCRITOS

Sección de Cliente

Si el rol con el que se autentica el usuario es de cliente, será redirigido a la página de inicio de los clientes, página en la que podrá acceder a la búsqueda de clases y a las clases en las que esté registrado a al menos una sesión.

The screenshot shows the 'Inicio de Clientes' page. At the top, there are links for 'Inicio' and 'Volver', and the user 'client@client.client' is logged in. Below the navigation bar, there are two buttons: 'Búsqueda' and 'Clases inscrito'.

Ilustración 22: PÁGINA DE INICIO DE CLIENTES

Ilustración x. Página de gestión de clientes inscritos

Desde la página de Búsqueda se podrá buscar y filtrar las diferentes clases por su nombre, capacidad máxima y duración máxima y mínima. Los filtros serán aplicados al pulsar el botón Buscar, a excepción del nombre, campo por el cual se buscará cada vez que se modifique.

Gestión web de un gimnasio

[Inicio](#) [Volver](#) client@client.client [Logout](#)

Nombre

Pesas

Capacidad Máxima


Duración Mínima

Duración Máxima

50

Buscar

Pesas2




instructor@instructor.instructor

Duración:50' Aforo máximo:60

Ilustración 23: PÁGINA DE BÚSQUEDA DE CLASES

Se podrá acceder a las sesiones de las clases pulsando el nombre de la clase, desde donde será posible la inscripción y desinscripción a las sesiones.

[Inicio](#) [Volver](#) client@client.client [Logout](#)



Pesas2

instructor@instructor.instructor

Duración:50

Capacidad máxima:60

La segunda clase de pesas

Sesiones

► Lunes(0) ► Martes(0) ► Miércoles(0) ► Jueves(0) ► Viernes(0)

▼ Sábado(2)

Sábado, 1 de junio de 2024, 17:27

Cientes inscritos: 1

instructor@instructor.instructor

Desinscribirse

Sábado, 8 de junio de 2024, 18:33

Cientes inscritos: 0

instructor@instructor.instructor

Inscribirse

► Domingo(0)

Ilustración 24: PÁGINA DE GESTIÓN DE SESIONES COMO CLIENTE

Página 41 de 47

En la otra parte, en las clases inscritas, se mostrarán aquellas clases en las que se esté inscrito a al menos una sesión y se podrá acceder a las sesiones de estas pulsando en el nombre, que redirigirá al usuario a la misma página recién mostrada.

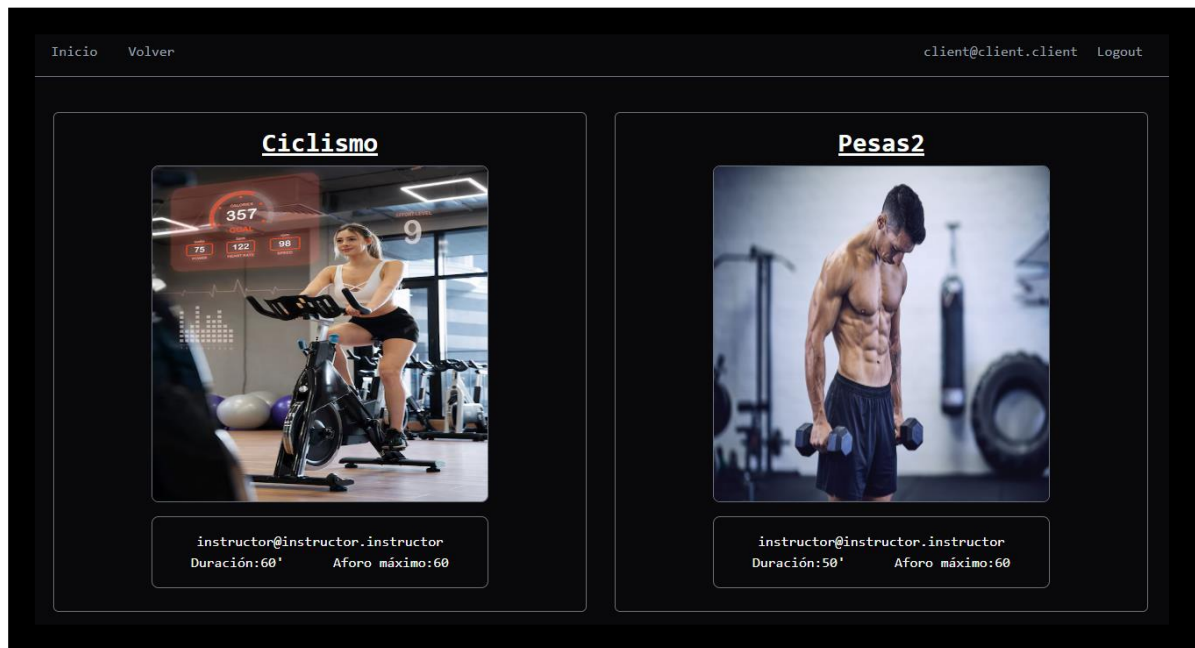


Ilustración 25: PÁGINA DE CLASES INSCRITAS

12. Principales aportaciones

Respecto al objetivo de la gestión de usuarios, se han realizado las siguientes aportaciones:

- Se ha desarrollado una aplicación con un alto nivel de flexibilidad y reutilización gracias al uso de React. La creación, modificación y eliminación de usuarios ha sido creada con un alto nivel de flexibilidad al estar formados por componentes de un menor tamaño que colaboran para lograr el objetivo deseado.
- Gracias al uso de NodeJS y de Express, se ha logrado una gestión simple y centrada de las peticiones de los usuarios.
- Con el uso de Prisma se ha logrado una gestión eficiente de la base de datos

En el siguiente punto, la gestión de clases, se han realizado las siguientes aportaciones:

- Se ha implementado una interfaz de usuario que reacciona a los diferentes tamaños de pantalla gracias al uso de Tailwind CSS. A su vez, se ha reducido al máximo el CSS sobrante gracias a la gestión de Tailwind CSS de la traducción del CSS.
- Gracias a los hooks que nos aporta React, ha sido posible la separación de la mayoría de la lógica de negocio de los componentes a funciones reutilizables.
- Se ha implementado un debouncer que permite la búsqueda a medida que el cliente inserta el nombre de una clase, de forma que el tiempo entre que escribe y se realiza la búsqueda ha sido adaptado al tiempo medio que un usuario tarda en escribir una letra,

permitiendo así que se realicen las búsquedas cuando el tiempo desde que se dejó de escribir es superior al establecido.

Respecto a la gestión de sesiones, se realizaron las siguientes aportaciones:

- Gracias al uso de Zod, se ha implementado un sistema de verificación de los datos que son recibidos en las peticiones HTTP cuyo uso es altamente reutilizable.
- Por medio de uso de las funcionalidades que nos aporta Express y Prisma, se ha implementado un sistema de sanción para aquellos clientes que no accedan a sus clases de forma reiterada.

13. Conclusiones

El objetivo principal de este TFG era estudiar las diferentes tecnologías que se usan en el desarrollo web como programador Full Stack, es decir, desarrollador de tanto Front-End como Back-End. Mi meta principal era aprender y vivir la experiencia de un desarrollador de este tipo, así como aumentar mi conocimiento en JavaScript, base de datos relacionales, CSS, Tailwind CSS y ORMs. Otra de las metas era acercarme a un desarrollo más real de una aplicación y sobre todo en una escala superior a la realizada durante la carrera.

Ha sido una experiencia muy gratificante y llena de procesos de aprendizaje. Por primera vez realicé un estudio de forma autónoma de los diferentes elementos que posteriormente usaría en el TFG y que me permitieron reconocer qué es lo que realmente me apasiona de la programación, lo que me facilitó decidir el rumbo profesional que quiero tomar como programador.

Dentro del aprendizaje, el proceso hizo que se me permitiera reconocer la gran cantidad de información que existe y mejorar en el proceso de filtrado de esta, así como una mejor organización. A demás de esto, me ha permitido cerciorarme de los conocimientos que tengo.

Al trabajar de forma autónoma, me fue posible adaptarme a un diferente modelo de trabajo y poder discernir aquellos métodos de trabajo con una mejor adaptación para mi modelo.

Pese que aún me queda un gran camino por recorrer, gracias a este proyecto me he podido acercar aún más a alcanzar un mayor nivel como programador.

14. Vías de trabajo futuro

Creación de una aplicación móvil a partir de la aplicación

Una de las posibles ampliaciones a realizar sería la creación de una aplicación móvil mediante la utilización del entorno de desarrollo integrado Android Studio, aprovechando su integración con Google y flexibilidad para la creación de aplicaciones Android y usando como lenguaje Kotlin debido a la posibilidad de creación de aplicaciones multiplataforma.

Creación de un perfil de usuario

Se podría ampliar el apartado de la personalización del cliente, añadiendo nuevos campos a la hora de realizar el registro del usuario, añadiendo desde nombre, edad y diferentes ajustes para una mayor personalización, así como una página para poder acceder a estos elementos y que sea posible que el propio usuario pueda modificar sus datos.

Implementación de un modo oscuro y claro

Actualmente existe una única forma de visualización, estando la aplicación enfocada en tonos oscuros. Por esto sería positivo la creación de un modo claro y oscuro, siendo el modo oscuro la visualización actualmente existente y el modo claro una visualización con tonos más claros y opuestos a los del modo actual.

Implementación de una base de datos externa

Para reducir la inmensa cantidad de información que se almacena como resultado de almacenar imágenes y para mejorar la escalabilidad de la aplicación, se podría extraer toda información almacenada y guardarla en una base de datos Oracle de mayor tamaño, pasando a retener la base de datos MySQL una versión reducida de los datos, de modo que, ante peticiones de imágenes, estas se pidan a esta segunda base de datos. A su vez, así se podría realizar migraciones diarias de información desde la base de datos Oracle a la MySQL para actualizar los datos.

15. Referencias

- Angus, C. (2021). *GitHub - angus-c/just: A library of dependency-free JavaScript utilities that do just one thing*. GitHub. <https://github.com/angus-c/just?tab=readme-ov-file#read-books> (recuperado el 4 de junio de 2024)
- Awati, R. (s. f.). *Blowfish*. TechTarget. <https://www.techtarget.com/searchsecurity/definition/Blowfish> (recuperado el 4 de junio de 2024)
- Cortés, D. E. (2023). *Componentes en React.js*. Medium. <https://medium.com/@diego.coder/componentes-en-react-js-9a1444cddc52#:~:text=js%3F-Un%20componente%20en%20React, en%20una%20sola%20unidad%20reutilizable> (recuperado el 4 de junio de 2024)
- Flores, F. (2023, 13 abril). *Qué es Visual Studio Code y qué ventajas ofrece*. OpenWebinars.net. <https://openwebinars.net/blog/que-es-visual-studio-code-y-que-ventajas-ofrece/> (recuperado el 4 de junio de 2024)
- Google I/O (s. f.). *Introducción a Android Studio*. Google for developers. <https://developer.android.com/?hl=es-419> (recuperado el 4 de junio de 2024)
- Holgado, J. L. (2021). *Qué es Tailwind y por qué usarlo*. <https://www.knowmadmood.com/es/blog/que-es-tailwind> (recuperado el 4 de junio de 2024)
- Llamas, L. (2024). *Qué es y cómo usar PNPM*. Luis Llamas. <https://www.luisllamas.es/que-es-pnpm/> (recuperado el 4 de junio de 2024)
- Londoño, P. (2023). *Qué es MySQL, para qué sirve y características principales*. <https://blog.hubspot.es/website/que-es-mysql#:~:text=MySQL%20es%20un%20sistema%20de%20administraci%C3%B3n%20de%20bases,de%20c%C3%B3digo%20abierto%20m%C3%A1s%20utilizada%20en%20el%20mundo>. (recuperado el 4 de junio de 2024)
- MacNeil, C. (2024). *5 responsabilidades clave del Product Owner*. Asana. <https://asana.com/es/resources/product-owner> (recuperado el 4 de junio de 2024)
- Material-UI. (s. f.). *React Tooltip component*. <https://v4.mui.com/es/components/tooltips/> (recuperado el 4 de junio de 2024)
- MDN Web Docs (s. f.). *DOM*. MDN Web Docs. <https://developer.mozilla.org/es/docs/Glossary/DOM> (recuperado el 4 de junio de 2024)
- Nettix (s. f.). *¿Qué es xampp y como puedo usarlo?*. Nettix Perú. <https://www.nettix.com.pe/blog/web-blog/que-es-xampp-y-como-puedo-usarlo> (recuperado el 4 de junio de 2024)
- Npm: zod. (2019). Npm. <https://www.npmjs.com/package/zod#introduction> (recuperado el 4 de junio de 2024)

- OpenProject. (s. f.). *La mejor alternativa a TeamGantt de código abierto - OpenProject*. OpenProject.org. <https://www.openproject.org/es/alternativas-de-software-de-gestion-de-proyectos/mejor-alternativa-teamgantt/#:~:text=TeamGantt%20es%20una%20soluci%C3%B3n%20sencilla,costos%20y%20gesti%C3%B3n%20de%20tiempo> (recuperado el 4 de junio de 2024)
- Plain Concepts. (2022). *Product Owner: ¿Qué es y cuáles son sus funciones en Scrum?* Plain Concepts. <https://www.plainconcepts.com/es/product-owner/> (recuperado el 4 de junio de 2024)
- Raeburn, A. (2024a). *¿Qué es un Scrum Master y cuál es su función?*. Asana. <https://asana.com/es/resources/scrum-master> (recuperado el 4 de junio de 2024)
- Raeburn, A. (2024b). *Qué es product backlog y guía para hacer uno con ejemplo*. Asana. <https://asana.com/es/resources/product-backlog> (recuperado el 4 de junio de 2024)
- React. (s. f.). *Preservar y reiniciar el estado* <https://es.react.dev/learn/preserving-and-resetting-state> (recuperado el 4 de junio de 2024)
- Robledano, A. (2019). *Qué es MySQL: Características y ventajas*. OpenWebinars. <https://openwebinars.net/blog/que-es-mysql/> (recuperado el 4 de junio de 2024)
- Roche, J. (s. f.). *Scrum: roles y responsabilidades: los 3 roles de la metodología Scrum*. Deloitte. <https://www2.deloitte.com/es/es/pages/technology/articles/roles-y-responsabilidades-scrum.html> (recuperado el 4 de junio de 2024)
- Skysnag (2023). *¿Qué es bcrypt?*. El blog de Skysnag. <https://www.skysnag.com/es/blog/what-is-bcrypt/> (recuperado el 4 de junio de 2024)
- Tailwind CSS. (s. f.). *Tailwind CSS*. <https://tailwindcss.com/> (recuperado el 4 de junio de 2024)
- Team Asana (2024). *Backlog: qué es el trabajo pendiente del sprint y ejemplos*. Asana. <https://asana.com/es/resources/sprint-backlog> (recuperado el 4 de junio de 2024)
- Thoughtworks (2022). *Excalidraw*. <https://www.thoughtworks.com/es-es/radar/tools/excalidraw#:~:text=Excalidraw%20es%20una%20herramienta%20de,de%20crear%20y%20compartir%20diagramas> (recuperado el 4 de junio de 2024)
- Torres (2019). *Reglas RestFull: aprendamos a definir correctamente las endpoints de nuestra API y hacerla más clara para el cliente*. Blog AscoDeCodigo. <https://blog.ascodecodigo.com/reglas-restfull/> (recuperado el 4 de junio de 2024)
- Ubah, K. (2023). *Fetch API – Cómo realizar un GET Request y un POST Request en JavaScript*. FreeCodeCamp. Traducido por Fagúndez Federico. [https://www.freecodecamp.org/espanol/news/fetch-api-como-realizar-un-get-request-y-un-post-request-en-javascript/#:~:text=fetch\(\)%20es%20un%20mecanismo,la%20ejecuci%C3%B3n%20de%20otras%20instrucciones](https://www.freecodecamp.org/espanol/news/fetch-api-como-realizar-un-get-request-y-un-post-request-en-javascript/#:~:text=fetch()%20es%20un%20mecanismo,la%20ejecuci%C3%B3n%20de%20otras%20instrucciones) (recuperado el 4 de junio de 2024)
- Valente, O., Rosal, M., Peveri, M. y Etchevarría, J. J. (s. f.). *El lenguaje de programación Kotlin: qué es y para qué sirve*. My Task Panel Consulting. <https://www.mytaskpanel.com/lenguaje-de-programacion->

[kotlin/#:~:text=%C2%BFQu%C3%A9%20es%20Kotlin%3F,utilizar%20para%20desarrollar%20aplicaciones%20Android.%20Kotlin](#) (recuperado el 4 de junio de 2024)

Vite (s. f.). *Getting Started*. Vite. <https://vitejs.dev/guide/> (recuperado el 4 de junio de 2024)