****

**E**scola **S**uperior de **E**nxeñaría **I**nformática

|  |
| --- |
| Memoria do Traballo de Fin de Grao que presenta  **D. Jorge Alcalde Piñeiro**  para a obtención do Título de Graduado en Enxeñaría Informática  **Título do Traballo de Fin de Grado** |

|  |  |
| --- | --- |
|  | , 2024  **Traballo de Fin de Grao Nº**:    **Titor/a:** Daniel González Peña  **Área de coñecemento:**  **Departamento:** |

**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 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 5](#_Toc158475096)

[2. Objetivos 5](#_Toc158475097)

[3. Resumen de la solución propuesta 5](#_Toc158475098)

[3.1. Solución propuesta 5](#_Toc158475099)

[3.2 Metodología empleada 6](#_Toc158475100)

[4. Planificación y seguimiento 6](#_Toc158475101)

[4.1 Planificación inicial 6](#_Toc158475102)

[4.2 Seguimiento 6](#_Toc158475103)

[5. Arquitectura 6](#_Toc158475104)

[5.1 Front-End 6](#_Toc158475105)

[5.2 Back-End 6](#_Toc158475106)

[6. Tecnologías e integración de productos de terceros 6](#_Toc158475107)

[6.1 Tecnologías 6](#_Toc158475108)

[6.2. Herramientas 7](#_Toc158475109)

[6.3 Librerías 8](#_Toc158475110)

[7. Especificación y análisis de requisitos 9](#_Toc158475111)

[8. Diseño del software 11](#_Toc158475112)

[8.1. Vista estática 11](#_Toc158475113)

[8.2. Vista dinámica 11](#_Toc158475114)

[9. Gestión de datos e información 11](#_Toc158475115)

[10. Pruebas 11](#_Toc158475116)

[11.Manual de usuario 11](#_Toc158475117)

[12. Principales aportaciones 11](#_Toc158475118)

[13. Conclusiones 11](#_Toc158475119)

# 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, digitalizando la gestión de los accesos a los gimnasios, las inscripciones a estes y la gestión de las diferentes actividades y recursos que ofrece un gimnasio

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 que ofrece un gimnasio y que a su vez, facilite la creación de las diferentes clases por los monitores y de la gestión de estas por su parte.

# 2. Objetivos

Este TFG tiene como objetivo principal la creación de una aplicación web orientada a la gestión de clases en el gimnasio. Concretamente, deberá implementar las siguientes funciones:

* Registro y acceso de usuarios.
* Creación de clases.
* Edición de clases.
* Borrado de clases.
* Inscribirse a clases.
* Desinscribirse a clases.
* Gestión de roles de los usuarios.
* Gestión de la asistencia de los usuarios.
* Interfaz sencilla, intuitiva y adaptable a dispositivos móviles.

# 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.

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 la gestión de base de datos. 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 en base a las ventajas que nos aporta Scrum:

* Aumenta el tiempo de trabajo sobre la aplicación, reduciendo las tareas y la documentación exhaustiva que no aporta valor al proceso de desarrollo.
* 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 sprint para decidir qué elementos se tratarán en el sprint y una reunión al final del sprint para mostrar una demostración del resultado del sprint.

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.

Para terminar, tenemos diferentes artefactos que se manejan durante el desarrollo del proyecto:

* **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:** resultado de cada sprint, una versión parcialmente utilizable o publicable del producto.

# 4. Planificación y seguimiento

En este apartado se muestra una comparativa entra la planificación original del proyecto y su seguimiento, viendo la diferencia entre los tiempos originales y el resultado final. Para tener una muestra más fácil de seguir, se hará 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 no puede exceder las 300h y la disponibilidad horaria del alumno. Para cumplir esto, se han planificado 7 sprints, todos con duración de 2 semanas, con excepción del primer 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 y último sprint que duran 20 h más el primero y 20h menos el último, dejando así un margen de 20h para posibles retrasos.

|  |  |  |  |
| --- | --- | --- | --- |
| Sprint | Fecha  inicio | Fecha  fin | Duración |
| #1  Documentación inicial  Landing page  Login + registro | 12/02/2024 | 03/03/2024 | 21d-60h |
| #2  Funcionalidades administrador | 4/03/2024 | 17/03/2024 | 14d-40h |
| #3  Gestión clases monitor | 18-03/2024 | 31/03/2024 | 14d-40h |
| #4  Gestión cliente | 1/04/2024 | 14/04/2024 | 14d-40h |
| #5  Gestión asistencia | 15/04/2024 | 28/04/2024 | 14d-40h |
| #6  Búsqueda clases | 29/04/2024 | 12/05/2024 | 14d-40h |
| #7  Documentación final | 13/05/2024 | 19/05/2024 | 7d-20h |
|  | | | |
| Total | 12/02/2024 | 19/05/2024 | 98d-280h |

TABLA 1: PLANIFICACIÓN INICIAL

Gráfico, Gráfico en cascada

Descripción generada automáticamente

Ilustración 1: Diagrama de Gantt inicial

## 4.2 Seguimiento

# 5. Arquitectura

Para la arquitectura se ha decidido el uso de aquella en la que se basan todas las aplicaciones Web: 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 una petición. Para esto utiliza las tecnologías mencionadas en el apartado [3.1].

Diagrama

Descripción generada automáticamente

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 créate 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 le proceso de creación de la aplicación y de la arquitectura. Las arquitecturas empleadas por cada parte serán explicadas en los siguientes apartados.

## 5.1 Front-End

## 5.2 Back-End

# 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 interpretado, dialecto del estándar ECMAScript. Actualmente está siendo usado el en 97% de las páginas web existentes. Se define como orientado a objetos, ​ basado en prototipos, imperativo, débilmente tipado y dinámico.

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

***MySQL***

Es un sistema de administración relacional de bases de datos.

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

***NodeJS***

Es un entorno en tiempo de ejecución multiplataforma, de código abierto, para la capa del servidor basado en el lenguaje de programación JavaScript, asíncrono basado en el motor V8 de Google. 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***

Es un 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

***Express***

Es un 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 la configuración de middlewares para responder ante solicitudes HTTP y define una tabla de rutas para poder acceder a los diferentes datos del servidor.

Express ha sido empleado para la creación de el 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 realiza mediante un lenguaje de modelado específico de Prisma.

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

## 6.2. Herramientas

***Visual Estudio Code***

Editor de código fuente desarrollado por Microsoft para Windows, Linux, macOS y web. Incluye soporte para la depuración, control integrado de Git, resaltado de sintaxis, finalización inteligente de código, fragmentos y refactorización de código.

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 compartirlo y organizarlo con las diferentes personas que trabajan.

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

***GitHub***

Herramienta web que complementa a Git y que permite almacenar y gestionar las actualizaciones de los diferentes proyectos que se decidan almacenar en su web.

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

***GitHub Copilot***

Es una herramienta de inteligencia artificial basada en la nuve 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

## 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 Facebook y la comunidad de software libre. No es un framework y por lo tanto no dicta un marco de trabajo como puede ser un Modelo-Vista-Controlador

Bcrypt

Zod

React-tooltips

## just-debounce-it

https://www.npmjs.com/package/just-debounce-it

# 7. Especificación y análisis de requisitos

En este apartado se exponen los diferentes roles identificados, las distintas historias de usuario identificadas y realizadas durante el proyecto que componen al Backlog del producto y por último los requisitos no funcionales de este. Esto fue generado a partid de los objetivos ya descritos en la sección de Objetivos.

Por una parte, los roles identificados son 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. 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 y marcar la asistencia a las diferentes clases en las que se encuentra inscrito. No podrá acceder a las funcionalidades del monitor.

Por otra parte, la estructura de las historias de usuario es la siguiente:

* Nombre breve.
* Descripción de la funcionalidad.
* Criterios de aceptación o lista de requisitos que tiene que cumplir la historia para que se considere como completa

***HU01 – Autenticación de Usuario***

|  |  |
| --- | --- |
| **Como** | Usuario |
| **Quiero** | Registrarme e iniciar sesión |
| **Para** | Poder hacer uso de las funcionalidades de la aplicación |
| **Pruebas de aceptación** | * Se debe poder realizar el registro iniciando sesión con un correo y una contraseña (Preguntar cómo me recomienda él hacer el sistema de inicio de sesión * 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** | * 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** | Monitor |
| **Quiero** | Poder añadir y gestionar sesiones a las clases que gestiono |
| **Para** | Poder gestionar las diferentes sesiones |
| **Pruebas de aceptación** | * Se debe poder añadir sesiones para cada clase, indicando el día y la hora * Se debe poder marcar la asistencia y la justificación de las faltas de asistencia |

***HU04 – Gestión de inscripciones***

|  |  |
| --- | --- |
| **Como** | Cliente |
| **Quiero** | Inscribirme y desinscribirme |
| **Para** | Poder asistir a las diferentes clases a las que esté inscrito |
| **Pruebas de aceptación** | * Se debe poder inscribir y desinscribir a las diferentes clases * Debe poder marcar asistencia en las clases |

***HU05 – Gestión de usuarios***

|  |  |
| --- | --- |
| **Como** | Administrador |
| **Quiero** | Poder modificar los roles que identifican a cada usuario |
| **Para** | Poder gestionar las acciones de los usuarios |
| **Pruebas de aceptación** | * Se debe poder modificar los roles 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 estea inscrito, se eliminarán todas las anteriores |

***HU06 – Consulta de clases***

|  |  |
| --- | --- |
| **Como** | Cliente |
| **Quiero** | Buscar clases |
| **Para** | Poder ver las diferentes clases a las que puedo inscribirme |
| **Pruebas de aceptación** | * A la hora de visualizar los resultados, se deben enseñar todos las a las que no está inscrito * Podrá realizar una búsqueda por nombre |

***HU07 – Asistencia a clases***

|  |  |
| --- | --- |
| **Como** | Monitor |
| **Quiero** | Poder marcar la asistencia a las clases |
| **Para** | No marcar a los clientes que asistan |
|  |  |
| **Pruebas de aceptación** | * Se debe poder marcar a los clientes que asistan a clase * Al encontrarse una falta de un cliente de forma repetida, este usuario debe ser penalizado |

***HU08 – Justificación falta a clases***

|  |  |
| --- | --- |
| **Como** | Monitor |
| **Quiero** | Poder justificar la falta de asistencia a las clases |
| **Para** | No contabilizar la falta a la hora de sancionar |
|  |  |
| **Pruebas de aceptación** | * Se debe poder marcar si es justificada o no una falta de asistencia. * Al contabilizarse las faltas para realizar sanciones, no se debe poder contabilizar las faltas justificadas |

***HU09 – Cambio Monitor***

|  |  |
| --- | --- |
| **Como** | Administrador |
| **Quiero** | Cambiar el monitor de una clase |
| **Para** | Poder asignar el representante de la clase a otro monitor |
| **Pruebas de aceptación** | * Se debe poder cambiar el monitor asociado a una clase |

En cuanto a requisitos no funcionales, se destaca únicamente la Usabilidad. La interfaz se ha diseñado para que sea lo más cómoda e intuitiva para el usuario, fácil de navegar en ella y optimizada para el uso en dispositivos móviles.

# 8. Diseño del software

En este apartado veremos el diseño de la aplicación tanto desde una vista estática como desde una vista dinámica

## 8.1. Vista estática

## 8.2. Vista dinámica

# 9. Gestión de datos e información

|  |  |  |  |
| --- | --- | --- | --- |
| URL | Descripción | M.HTTP | Objeto JSON |
| /users | Devuelve todos los usuarios | get | [{},{},…] |
| /users/all | Devuelve todos los administradores,clientes e instructores | Get | {admins:[],clients:[],instructors[]} |
| /users/login | Inicia sesión | Post | {email,password, } |
| /users/register | Registra a un usuario | Post | {email,password,role} |
| /users/:email | Elimina un usuario | Delete | {} |
|  |  |  |  |
|  |  |  |  |

# 10. Pruebas

# 11.Manual de usuario

# 12. Principales aportaciones

# 13. Conclusiones

# ENLACES

<https://www.plainconcepts.com/es/product-owner/> (PRODUCT OWNER SCRUM)

<https://asana.com/es/resources/product-owner> (PRODUCT OWNER SCRUM)

<https://asana.com/es/resources/scrum-master> (scrum master)

<https://www2.deloitte.com/es/es/pages/technology/articles/roles-y-responsabilidades-scrum.html> (equipo de desarrollo)

<https://asana.com/es/resources/product-backlog>

<https://asana.com/es/resources/sprint-backlog>

Vite, «Getting Started» .Available: https://vitejs.dev/guide/