****

**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 8](#_Toc167895553)

[2. Objetivos 8](#_Toc167895554)

[3. Resumen de la solución propuesta 9](#_Toc167895555)

[3.1. Solución propuesta 9](#_Toc167895556)

[3.2 Metodología empleada 9](#_Toc167895557)

[4. Planificación y seguimiento 10](#_Toc167895558)

[4.1 Planificación inicial 10](#_Toc167895559)

[4.2 Seguimiento 11](#_Toc167895560)

[5. Arquitectura 12](#_Toc167895561)

[5.1 Front-End 12](#_Toc167895562)

[5.2 Back-End 12](#_Toc167895563)

[6. Tecnologías e integración de productos de terceros 12](#_Toc167895564)

[6.1 Tecnologías 12](#_Toc167895565)

[6.2. Herramientas 14](#_Toc167895566)

[6.3 Librerías 15](#_Toc167895567)

[7. Especificación y análisis de requisitos 16](#_Toc167895568)

[8. Diseño del software 19](#_Toc167895569)

[8.1. Vista estática 19](#_Toc167895570)

[8.2. Vista dinámica 19](#_Toc167895571)

[9. Gestión de datos e información 19](#_Toc167895572)

[10. Pruebas 19](#_Toc167895573)

[11.Manual de usuario 24](#_Toc167895574)

[11.1 Requisitos mínimos 24](#_Toc167895575)

[11.2 Instalación 24](#_Toc167895576)

[11.3 Funcionamiento 25](#_Toc167895577)

[12. Principales aportaciones 33](#_Toc167895578)

[13. Conclusiones 33](#_Toc167895579)

[14. Vías de trabajo futuro 33](#_Toc167895580)

[ENLACES 34](#_Toc167895581)

# 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. 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 convertir estas aplicaciones en complicadas de entender y muchas veces hasta excedentes para el usuario

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, 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

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

Imagen X: 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 créate vite@latest** [] y a **npm i** 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. Las arquitecturas empleadas por cada parte serán explicadas en los siguientes apartados.

## 

## 5.1 Front-End

## 5.2 Back-End

El intercambio de datos entre el Front-End y la base de datos se hará mediante la una API que exponer recursos que pueden ser consumidos mediante URLs y que siguen una serie de reglas

Diagrama

Descripción generada automáticamente

Imagen x: Arquitectura Back-End

En la imagen x podemos ver la arquitectura del Back-End. Con NodeJS se crea un servidor web básico el cual está encargado de recibir las peticiones http al puerto 1234. Sin embargo, al ser necesario un manejo específico de cada tipo de petición HTTP (GET, POST, PUT y DELETE) 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, framework que nos permite la gestión de estas peticiones de manera sencilla e independiente, a demás de así evitar la implementación de una solución ya existente.

La información almacenada en las peticiones HTTP entrantes determinan qué código será ejecutado en la aplicación. Esta información se comparte, modifica y tratan 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 decidió 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:** En esta capa están definidas 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 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 de bases de datos racionales.

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.

***PNPM***

Es un 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 tener los paquetes por separado.

PNPM 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.

***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***

XAMPP es una 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.

## 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.

Esta ha sido empleado 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.

# 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 editar y borrar sesiones * 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 |

***HU05 – Gestión de usuarios***

|  |  |
| --- | --- |
| **Como** | Administrador |
| **Quiero** | Poder modificar los roles que identifican a cada usuario así como las contraseñas |
| **Para** | Poder gestionar las acciones de los usuarios |
| **Pruebas de aceptación** | * 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. |

***HU06 – Consulta de clases***

|  |  |
| --- | --- |
| **Como** | Cliente |
| **Quiero** | Buscar clases |
| **Para** | Poder ver las diferentes clases a las que puedo inscribirme |
| **Pruebas de aceptación** | * Podrá realizar una búsqueda por nombre |

***HU07 – Validación Cliente***

|  |  |
| --- | --- |
| **Como** | Administrador |
| **Quiero** | Validar a los clientes |
| **Para** | Que los clientes puedan acceder a las funcionalidades |
| **Pruebas de aceptación** | * Se debe poder validar cada cliente * Se debe poder quitar la validación para cada cliente |

***HU08 – Duplicación Sesión***

|  |  |
| --- | --- |
| **Como** | Monitor |
| **Quiero** | Duplicar una sesión |
| **Para** | Repetir sesiones sin tener que crearlas completamente a mano |
| **Pruebas de aceptación** | * Se debe poder duplicar una sesión, siendo esta creada en una número de días ,determinados por el usuario, posteriores a la fecha de la sesión |

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 se expone el diseño software de la aplicación web con un enfoque

simplificado, a través dos perspectivas del sistema, una estática para entender la

estructura que lo forma, y otra dinámica para entender cómo se comporta.

## 8.1. Vista estática

## 

## 8.2. Vista dinámica

# 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 que ya ha sido previamente 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 una base de datos.
* **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 su uso. Con Prisma las entidades y relaciones se crean mediante esquemas de JavaScript y mediante 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, migración que guarda los cambios producidos. Esto lo que produce es un historial de cambios que podrán ser reproducidos en aquel proyecto que se crea pertinente.

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

Diagrama

Descripción generada automáticamente

Imagen x: Modelo de base de datos

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, se considera importante mostrar una tabla informativa con las URI de los recursos REST, compuesta por una breve descripción, los métodos HTTP aceptados por cada URL, parámetros y el objeto devuelto.

|  |  |  |  |
| --- | --- | --- | --- |
| **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/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 | Deleted |
| /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 | {} |
| /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 x: API

# 10. Pruebas

Con el fin de garantizar el buen funcionamiento de la aplicación, a medida que esta se fue desarrollando, se realizaron un gran número de pruebas funcionales, siempre teniendo en cuanta los criterios de aceptación previamente definidos en cada una de las historias de usuario y comprobando si estos efectivamente se cumplían de manera exitosa. Este tipo de prueba, se centra únicamente en las salidas generadas en respuesta a las entradas seleccionadas y a las condiciones de ejecución.

***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 registra y se le redirige a la página de login para que inicie sesión |
| **Evaluación** | Superada correctamente |

|  |  |
| --- | --- |
| **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, siendo redirigido a una parte u otra en función del rol que tenga |
| **Evaluación** | Superada correctamente |

***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** | Superada correctamente |

|  |  |
| --- | --- |
| **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** | Superada correctamente |

|  |  |
| --- | --- |
| **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** | Superada correctamente |

***HU03 – Gestión de sesiones***

|  |  |
| --- | --- |
| **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 sesiones |
| **Evaluación** | Superada correctamente |

|  |  |
| --- | --- |
| **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 muestran la sesión recién editada junto con el resto de sesiones |
| **Evaluación** | Superada correctamente |

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

|  |  |
| --- | --- |
| **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** | Superada correctamente |

|  |  |
| --- | --- |
| **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** | Superada correctamente |

***HU04 – 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** | Superada correctamente |

|  |  |
| --- | --- |
| **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** | Superada correctamente |

***HU05 – 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 usuarios |
| **Evaluación** | Superada correctamente |

|  |  |
| --- | --- |
| **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 usuarios |
| **Evaluación** | Superada correctamente |

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

***HU06 – 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** | Superada correctamente |

***HU07 – 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** | Superada correctamente |

|  |  |
| --- | --- |
| **Funcionalidad** | **Desvalidación 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 desvalidación y se mostrará la tabla de nuevo |
| **Evaluación** | Superada correctamente |

***HU08 – 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 a partir de hoy 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** | Superada correctamente |

# 11.Manual de usuario

En este apartado se explicará de manera detallada todos los pasos necesarios para la realización instalación y puesta en funcionamiento de la aplicación, así como un manual de uso para el correcto uso de esta

## 11.1 Requisitos mínimos

A continuación se detallan los requisitos mínimos para la ejecución de la aplicación por parte del desarrollador y los requisitos para la utilización de la aplicación por parte del usuario

**Modo desarrollador**

**--preguntar a dani cómo poner los requisitos mínimos de hardware**

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.

**Modo usuario**

El único requisito para los usuarios es disponer de un navegador como puede ser Firefox, Google Chrome o Opera, aquellos en los que ha sido probada la aplicación durante el desarrollo.

## 11.2 Instalación

**Modo desarrollador**

Para la puesta en funcionamiento de la aplicación, será necesario levantar por una 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**.

Para el Back-End necesitaremos acceder a **.\back** 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 Back-End usaremos el comando: **pnpm run dev**.

**Modo 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

Con el fin de entender mejor el funcionamiento de la aplicación, a continuación se listan las secciones y sus respectivas funcionalidades , siendo estas en las que está dividida la aplicación.

* Sección de Inicio (Página de inicio)
* Sección de Autenticación (Autenticación y Registro del usuario)
* Sección de Administración (Gestión del administrador de los usuarios)
* Sección de Instrucción (Gestión de clases y sesiones por parte del instrucción)
* 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 caos de estar autenticados ya, al pulsar en “Iniciar Sesión” seremos llevados a la zona pertinente al rol del usuario.

Captura de pantalla de un celular

Descripción generada automáticamenteIlustración x. 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 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

Interfaz de usuario gráfica

Descripción generada automáticamente

Ilustración x. Formulario de inicio de sesión

Interfaz de usuario gráfica

Descripción generada automáticamente

Ilustración x. 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.

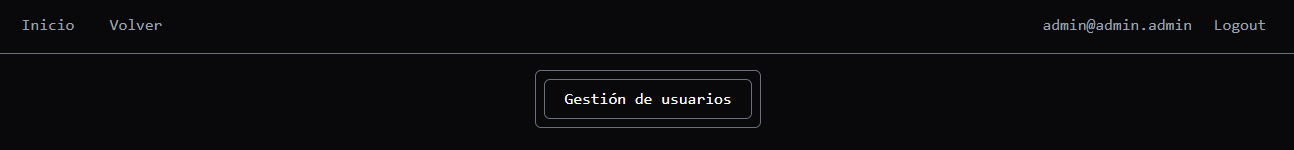


Ilustración x. Página de inicio de administrador

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.

Captura de pantalla de un celular

Descripción generada automáticamente

Ilustración x. 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.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Ilustración x. 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.

Captura de pantalla de un celular

Descripción generada automáticamente

Ilustración x. 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.

Captura de pantalla de un celular con texto e imagen

Descripción generada automáticamente

Ilustración x. 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 así como se mostrarán de forma 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.

Captura de pantalla de un celular

Descripción generada automáticamente

Ilustración x. 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.

Captura de pantalla de un celular

Descripción generada automáticamente

Ilustración x. Página de gestión de sesiones 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.

Captura de pantalla de un celular

Descripción generada automáticamente

Ilustración x. Página de gestión de clientes inscritos

**Sección de Cliente**

Si el rol con el que se registra 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.



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.

Captura de pantalla de un celular

Descripción generada automáticamente

Ilustración x. 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.

Captura de pantalla de un celular

Descripción generada automáticamente

Ilustración x. Página de gestión de sesiones como cliente

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.

Captura de pantalla de un video juego

Descripción generada automáticamente

Ilustración x. Página de clases inscritas

# 12. Principales aportaciones

* La aplicación está protegida mediante un sistema de autenticación.
* La aplicación es adaptable a dispositivos móviles.
* La aplicación permite una gestión simple y sencilla de clases y sesiones de un gimnasio.
* La aplicación permite ver de forma sencilla y simple las clases a las que estás inscrito y a las que debes asistir
* Una forma de gestión de las clases de un gimnasio de forma reducida y simple

# 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, TailwindCSS 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

Como posibles vías de trabajo futuro para la aplicación, se enlistan las siguientes ampliaciones:

**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 androidy 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.

# 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://openwebinars.net/blog/que-es-mysql/> mysql

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

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

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

<https://developer.android.com/studio/intro?hl=es-419> android studio

https://www.mytaskpanel.com/lenguaje-de-programacion-kotlin/#:~:text=¿Qué%20es%20Kotlin%3F,utilizar%20para%20desarrollar%20aplicaciones%20Android. Kotlin

<https://www.luisllamas.es/que-es-pnpm/> pnpm

<https://www.skysnag.com/es/blog/what-is-bcrypt/> bcrypt

<https://www.techtarget.com/searchsecurity/definition/Blowfish> cifrado blowfish

<https://www.npmjs.com/package/zod#introduction> zod

<https://v4.mui.com/es/components/tooltips/> tooltip

<https://github.com/angus-c/just?tab=readme-ov-file#read-books> just

[5 Reglas básicas para crear tus endpoints (ascodecodigo.com)](https://blog.ascodecodigo.com/reglas-restfull/) reglas para los endpoints

<https://www.thoughtworks.com/es-es/radar/tools/excalidraw#:~:text=Excalidraw%20es%20una%20herramienta%20de,de%20crear%20y%20compartir%20diagramas>. Excalidraw

<https://www.nettix.com.pe/blog/web-blog/que-es-xampp-y-como-puedo-usarlo> xampp