Universidade 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:

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 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

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

Scrum ? Sprints?

4. Planificación y seguimiento

4.1 Planificación inicial
4.2 Seguimiento

5. Arquitectura

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

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 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

HU04 – 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

HU05 – 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

HU06 – 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

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

10. Pruebas

11.Manual de usuario

12. Principales aportaciones

13. Conclusiones