SISTEMA DE REGISTRO DE ENTRENAMIENTOS

Práctica 4: Diseño, desarrollo e instalación de un sistema de información Web con persistencia de datos

---

Pablo Ernesto Angusto Delgado 842255
José Miguel Florentín Domingo 839899
Miguel Aréjula Aísa 850068
Curso 2023-2024

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

Universidad Zaragoza 1542

Índice

Resumen

Cuando se nos planteó este proyecto vimos una oportunidad para buscar una solución a un problema. Tras realizar un estudio sobre diferentes de ellos decidimos tratar de ayudar a la gente que le gusta hacer deporte o quiere empezar a tener un estilo de vida más saludable. Para encontrar la mejor solución realizamos una lluvia de ideas, en la cual cada uno de los miembros propuso al menos dos problemas que había encontrado y su solución correspondiente. Tras exponerlos, nos dimos cuenta que los tres habíamos coincidido en uno: *La dificultad de tener un registro de tus entrenamientos y hábitos saludables.*

Concordamos en que para mantener un estilo de vida saludable es indispensable poder ver la mejoría de los resultados logrados a lo largo del tiempo. Así, encontramos la idea final de nuestro proyecto. Una página web en la cual los usuarios podrán realizar un registro de sus entrenamientos y hábitos saludables. Siendo capaces de visualizar sus datos de una forma visual mediante gráficas. Mientras pensábamos las distintas funcionalidades del sistema se nos ocurrieron nuevas ideas que aportan mucho a nuestro sistema como: realizar el registro del entrenamiento en tiempo real o compartir tus entrenamientos con tus amigos.

En está práctica se ha implementado el sistema definido en las anteriores prácticas. Partiendo del diseño hecho en **Figma**, se ha llevado a cabo la interfaz de usuario con **React**, que permite con **HTML** diseñar los elementos básicos de la web y añadirle una mayor funcionalidad con el uso de **Javascript**. Todo esto a la vez que se trabaja en el framework de *express* con **Node** con el fin de gestionar la base de datos que contiene la información del sistema.

Metodología

En la primera sesión creamos un repositorio en GitHub y lo compartimos para tener todos los integrantes del grupo acceso a él. A continuación generamos un proyecto vite en el cual desarrollaremos nuestro sistema. Añadimos varias dependencias y librerías que sabíamos que íbamos a necesitar en un futuro. También creamos un contenedor para trabajar todos en el mismo sistema operativo, ya que cada uno de los miembros tenía uno distinto. Para su ejecución hemos usado *Docker*. De esta forma hemos conseguido evitarnos muchos problemas a la hora de descargar las librerías y dependencias. Al final de esta sesión dividimos las tareas a realizar por cada miembro del equipo. Asignando los distintos wireframes diseñados en una de las prácticas anteriores.

Tras acabar los esquemas organizamos una reunión para poner en común el trabajo realizado y realizar las funciones encargadas de interactuar con la base de datos. Primero de todo, planteamos las funciones a implementar, es decir, crear un usuario, eliminar un usuario, etc. Y una vez que teníamos todo plantemos comenzamos a realizarlas.

Dificultades Encontradas

Era la primera vez de algún integrante que trabajaba con esta tecnología y el resto de integrantes tenían unos conocimientos muy básicos de esta. Esto provocó que los primeros días de trabajo nos costará más avanzar e invertimos una gran cantidad de tiempo aprendiendo nuevas cosas sobre la tecnología. Además de depurar constantemente en busca de errores.

Otro gran problema que hemos tenido ha sido el uso correcto de GitHub. Era la primera vez que lo utilizabamos para un gran proyecto en el cual los integrantes modifican los mismos ficheros. En un primer momento utilizabamos la herramienta *Live Share* de *Visual Studio Code.* Sin embargo, nos dimos cuenta que esto no era la forma correcta de trabajar, ya que siempre haciamos *Pulls* a la rama *main*. Provocandonos muchos conflictos en los ficheros. Por lo tanto decidimos informarnos y aprender sobre el correcto uso de esta herramienta. A partir de este momento comenzamos a usar las ramas. Cada funcionalidad era realizada en una rama distinta, provocando un *workflow* correcto y sin ningun problema. Además no se permitía hacer un *pull* directamente a la rama *main*. Era obligatorio realizar un *pull request* y fuese otro miembro del grupo el que comprobase el correcto funcionamiento de esta para confirmar el *pull request.* Otra condición que estipulamos, fue que todo el código que se encontrase en la rama main debía de compilar y funcionar correctamente.

El último gran problema que tuvimos fue la organización. De nuevo provocado por la inexperiencia en trabajos de esta magnitud y la inexperiencia a la hora de determinar el tiempo a dedicar a cada tarea, ha hecho que nos haya costado organizarnos de la forma más óptima y que permita un trabajo más eficiente. Según avanzabamos en el proyecto hemos ido mejorando este aspecto.

Han sido varios los problemas que han surgido en el desarrollo de esta práctica. Sin embargo, nos ha permitido adquirir experiencia con este tipo de trabajos que nos permitiran realizar correctamente los próximos trabajos a lo largo de las siguientes asignaturas y en la vida laboral.

Plan de pruebas

Para las pruebas del sistema, cada miembro del grupo se ha encargado de realizar cada uno tres pruebas de usuario con distintas personas. Así se ha podido comprobar aquellos aspectos que no quedaban tan claros dentro de la interfaz del sistema. Con ello las tareas se han intentado dejar más claras y sencillas de realizar.

Por otro lado, los miembros del grupo hemos realizado en reuniones repetidas pruebas del sistema realizando todos los casos de uso y viendo problemas que han ido surgiendo a lo largo del desarrollo y debatiendo nuestras posturas acerca de temas de debate como lo es la interfaz de usuario.

Estructura

En nuestro proyecto disponemos de dos grandes repositorios:

1. **api:** en él se encuentra todo lo necesario para gestionar la base de datos del sistema. En los distintos subdirectorios se definen aspectos relativos a la información y cómo se almacenan los datos además de ficheros de configuración. Entre los más importantes destacan:
    a. **models:** donde se definen las tablas de la BD.
    b. **controllers**: donde están todas las funciones accesibles y utilizadas en la implementación del sistema, es decir nuestro DAO (Data Access Object).
    c. **routes**: en él están disponibles las rutas necesarias para acceder al DAO.
    d. El resto de archivos y directorios son por lo general ficheros de configuración no tan reseñables en cuanto a implementación.
2. **web:** el directorio contiene todo lo relativo al diseño de la interfaz de usuario y funcionalidad de la página. Dentro del fichero **src**, además de estar el main.jsx y App.jsx que inicializan el sistema, se encuentran una serie de subdirectorios importantes de comentar:
    a. **Components**: aquí se encuentran todos los elementos visuales del sistema junto con su funcionamiento. Por un lado se encuentran las páginas que componen el conjunto de pantallas de la web y por otro algunos componentes adicionales usados dentro de ellas.
    b. **css**: carpeta que contiene los archivos de extensión .css que formatea la visualización de los elementos existentes en el directorio *Components*. A excepción de elementos generales que se encuentran en un archivo .css general (*styles.css*) hay un archivo de estilo para cada pantalla.
    c. **Router**: el router es una herramienta en React que sirve para navegar, en nuestro caso, entre las distintas pantallas. Permite redirecciones desde unas vistas a otras facilitando la implementación dado a su sencillo uso. En él están por tanto incluidas todas las páginas alcanzables para el usuario.

Cumplimiento de los Requisitos Funcionales

La gran mayoría de los requisitos planteados inicialmente han sido realizados, dotando al sistema de una gran funcionalidad y variedad de posibilidades. Aquí todos aquellos:

- **Registro e inicio de sesión.**
- **Creación de entrenamientos**.
- **Registro de entrenamiento a tiempo real**.
- **Preparación de los entrenamientos**.
- **Seguimiento a otros usuarios**.
- **Visualización de estadísticas y evolución**.
- **Inicio de sesión como administrador**.
- **Gestión de ejercicios**.
- **Gestión de entrenamientos**.

Solo uno de ellos no hemos conseguido llegar a implementarlo debido a su dificultad, lo que requeriría disponer de más tiempo para realizarlos. No obstante, dicho requisito fue definido con una importancia baja con respecto al resto y por tanto sería implementado en una futura versión del sistema. Su existencia no supone ninguna falta de operatividad en la web y por tanto se podría decir que es prescindible:

- **Conexión con aplicaciones externas**.

Seguridad del sistema y su mantenimiento

En cuanto a seguridad se refiere, nuestro sistema se encarga de asegurar la identidad de los usuarios. Para ello, se dispone de un servicio de creación y autenticación de usuarios. Si uno de ellos quiere acceder a nuestra página web, deberá registrarse y disponer de una cuenta dentro de nuestra aplicación.

Por otro lado, protegemos la información del mismo al separar dos tipos de usuarios en nuestra web. Los usuarios normales tienen permiso para interactuar del sistema añadiendo información al mismo con datos personales relacionados con su actividad física y su salud. Sin embargo, están limitados a no poder eliminar o editar según qué datos que comprometería la seguridad de la aplicación. Por ello, existe un tipo de usuario administrados con tareas de mayor poder en el sistema y que permite actualizar y realizar el mantenimiento del mismo.

Sus interacciones con la BD conllevan una mayor responsabilidad ya que tienen un rango mayor de interacción. Es por ello que su uso debe ser restringido y que no se puede crear una cuenta de tipo administrador directamente desde la web.

Por último cabe recalcar que al acceder a la api es necesario acceder usando un token para que las operaciones sean aceptadas y denegando así todas aquellas peticiones que no dispongan del mismo.

En cuanto al mantenimiento, lo ya explicado. El usuario administrador será el encargado de realizar cambios o actualizaciones en la web con el fin de mantener una información no obsoleta e introducir ejercicios, rutinas o métricas novedosas que atraigan a los usuarios así como tener la posibilidad de solucionar futuros errores no previstos.

Documentación del sistema

Para poder usar el sistema se ha utilizado un contenedor *Docker*. Aquí una explicación de la configuración necesaria para poder correr el sistema.

El disponer de un *devcontainer* nos ha permitido trabajar con el mismo sistema operativo y configuración evitando problemas de versiones y arquitecturas. Por ello será necesario disponer de *Docker* en nuestra máquina. Por otro lado, recomendamos el uso de *VisualStudio code* como editor de texto ya que este dispone de una extensión para el uso de contenedores con la herramienta seleccionada y facilita considerablemente su uso.

Tras tener instalada la extensión, nos preguntará si queremos reabrir el proyecto en un contenedor, aceptamos y esperamos a que configure lo necesario.

Una vez dentro del contenedor será necesario instalar desde línea de comandos npm. Para ello, ejecutar en la terminal: `npm install`

Una vez hecho esto, abrir la carpeta api y ejecutar la api de la siguiente forma:

```
> cd api
> npm start
```

Tras ello, abrir otra terminal en la que ejecutar estos comandos:

```
> cd web
> npm run dev
```

Pinchar en el link que aparecerá tras ejecutar esto, y entonces estaremos listos para poder ejecutar la aplicación.

Para entrar en la web, en el sistema de login se proporciona la siguiente cuenta para entrar:

- **Usuario:** pabloangusto@gmail.com
- **Contraseña:** contrasenapabload

Accesibilidad y usabilidad del sistema

Con el fin de desarrollar un sistema accesible y con una gran usabilidad que permita llegar a un público más amplio se han llevado a cabo técnicas de desarrollo de interfaces claras y sencillas.

Las pantallas no se encuentran sobrecargadas, es decir, se ha seguido un diseño minimalista y se ha apostado por la inclusión de una gran cantidad de iconos que permite reconocer de una mejor forma las tareas a realizar.

Por otro lado, la elección de los colores ha estado pensada para que tuvieran altos contrastes y no hubiese problemas para aquellas personas con menos capacidades cognitivas o visuales.

La consistencia en el sistema también es un gran aspecto a tener en cuenta a la hora de desarrollar una aplicación. Se necesita que mismos iconos lleven a realizar mismas tareas, pantallas con diseños parecidos tengan funcionalidades parejas, etc.

Bibliografía

MongoDB documentación:
https://www.mongodb.com/docs/manual/reference/ (14/10/23)
Vídeo explicativo de creación de una API con express:
https://youtu.be/YmZE1HXjpd4 (10/10/23)
Documentación oficial de react:
https://es.react.dev/
Documentación oficial de nextui
https://nextui.org/
Documentación oficial de Tailwind
https://tailwindcss.com/
Documentación oficial de apex-charts (herramienta para realizar gráficas)
https://apexcharts.com/docs/react-charts/
Guia básica de CSS
https://www.w3schools.com/css/
Curso sobre GitHub
https://www.coursera.org/learn/introduction-git-github/home/module/1
Curso sobre Github
https://github.com/mouredev/hello-git

Tiempo Empleado

Para el desarrollo de esta práctica el trabajo fue repartido equitativamente, por lo tanto, hemos trabajado prácticamente el mismo tiempo todos los miembros del grupo. Además, ciertas horas han sido de reuniones o trabajo colectivo, ya sea planteando el diseño o la realizando la memoria.

	Pablo Angusto	Jose Miguel Florentín	Miguel Aréjula
Reuniones	18 h	18 h	18 h
Trabajo individual	30 h	30 h	30 h
Trabajo colectivo	15 h	15 h	15 h