SISTEMA DE REGISTRO DE ENTRENAMIENTOS

Práctica 5: Despliegue de un sistema de información Web con persistencia de datos utilizando contenedores Docker

---

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.

Objetivo y alcance funcional final de la aplicación

Para elegir cuál iba a ser nuestro objetivo buscamos una necesidad real en nuestras vidas y que tenga un impacto en las vidas de las personas. A su vez, queríamos que estuviese relacionado con algo que tuviésemos en común los tres miembros del grupo. Finalmente decidimos que íbamos a buscar algo relacionado con la salud y un estilo de vida saludable.

Una vida saludable conlleva realizar buenas prácticas alimentarias, un buen descanso, tener una buena salud mental, etc. Pero, sin duda, uno de los pilares fundamentales para alcanzar el bienestar es el deporte.

El deporte es una actividad que en ocasiones puede resultar dura y exigente, pero que a la vez termina siendo gratificante. Una de las cosas que más satisfactorias resultan es el poder ver los resultados y la mejora día tras día gracias al esfuerzo realizado. Establecer unos objetivos y ver cómo se cumplen o cómo se progresa hacia ellos sirve de gran ayuda para tener constancia.

Muchas personas que se proponen metas, ya sea bajar o subir de peso, ganar fuerza o resistencia, terminan abandonando sus objetivos muchas veces por ver el ejercicio físico como una actividad frustrante. El poder ver gráficamente tu evolución puede evitar esta desmotivación y vigilar todos esos cambios durante el proceso. Un claro ejemplo de ello es el ejercicio en el gimnasio. Puede resultar difícil controlar el peso que se levanta, la cantidad de repeticiones y series que se hace en un determinado ejercicio, incluso la evolución del peso corporal a lo largo del tiempo o según el tipo de ejercicio realizado. Sin embargo, si se anota la evolución, llega a ser gratificante poder observar mejoras en tu fuerza, y al fin y al cabo, en tus propósitos.

Además, en actividades de este estilo, las personas tienden a compartir sus experiencias, contarlas e incluso practicarlas junto con otras. El compartir una pasión, un proceso, también es una manera de realizar deporte disfrutándolo y con mayor ilusión.

Esta aplicación quiere, entre otras cosas, ayudar al usuario a que las ganas y la motivación por la actividad física pesen más sobre el rechazo y la frustración. Por ello, se ofrecerá al usuario la capacidad de almacenar y visualizar en forma de métricas, gráficos y estadísticas datos relativos a su salud, y principalmente, a su entrenamiento deportivo.

Consistirá en que el usuario pueda meter sus datos mientras realiza ejercicio o bien una vez acabado el mismo. Los datos irán desde la cantidad de peso, series y repeticiones con las que se maneja en un ejercicio de fuerza, el tiempo de entrenamiento por día, el ritmo en una carrera continua, el peso corporal del usuario,, horas de sueño, etc.

A su vez, se proporcionará un sistema en el que será posible la interacción entre usuarios mediante mensajes. Estos mensajes podrán ser tanto mensajes de texto convencionales como gráficos o resúmenes de entrenamientos que éste haya realizado.

El objetivo de nuestra aplicación es :

- Proporcionar un espacio en el que los usuarios sean capaces de controlar sus actividades deportivas y métricas sobre su estado físico mediante gráficos y estadísticas que reflejen los distintos datos que los usuarios vayan introduciendo.
- Motivar y promover la actividad física haciendo ésta más amena y divertida al poder explorar un sistema web dedicado exclusivamente al usuario.
- Permitir la comunicación entre usuarios con el fin de que sea un aliciente para que más y más personas se sumen a llevar un estilo de vida saludable.

Nuestro sistema consta de dos perfiles de usuarios; el usuario principal y el administrador del sistema.

El usuario principal es toda aquella persona que quiera registrar su estado de salud, independientemente de su edad, estado de salud o conocimientos sobre las correctas prácticas para tener un estilo de vida saludable. Sus funcionalidades son:

- **Registro e inicio de sesión**: Los usuarios podrán crearse una cuenta en la aplicación web donde poder guardar su perfil de entrenamientos personalizado y sus estadísticas.
- **Creación de rutinas**: Los usuarios podrán crear rutinas personalizadas con los ejercicios que el sistema ofrece, las rutinas contendrán información sobre los ejercicios que lo componen e información relevante para la realización de cada ejercicio. En el caso de un ejercicio de levantamiento de peso, podría ser el número de series, las repeticiones por cada serie y el peso a utilizar. Además podrán utilizar las rutinas que hay predefinidas en la aplicación web y adaptar el peso, las series y las repeticiones de cada usuario.

- **Registro de entrenamiento a tiempo real**: Los usuarios podrán marcar los ejercicios que van realizando durante el entrenamiento elegido para que la ejecución de un entrenamiento sea más guiada y sencilla, por ejemplo: durante un ejercicio de fuerza se irá mostrando el número de series y repeticiones realizadas de modo que no hará falta recordarlo.
- **Preparación de los entrenamientos:** Los usuarios podrán seguir rutinas de ejercicios personalizadas u otras ya existentes para poder preparar con antelación las sesiones de entrenamiento y llevar un seguimiento aún más potente.
- **Seguimiento a otros usuarios**: Los usuarios podrán añadir como amigos a otros usuarios. Entonces el usuario podrá compartir con otros usuarios los entrenamientos que realiza y a su vez podrá ver los entrenamientos de sus amigos.
- **Visualización de estadísticas y evolución**: Los usuarios tendrán acceso a unas estadísticas creadas a partir de sus entrenamientos realizados anteriormente con el objetivo de seguir el progreso a lo largo del tiempo y poder ajustar nuevos objetivos.

El segundo rol es el administrador del sistema que se encarga del mantenimiento y actualización de ejercicios (Press Banca, sentadilla…) y crear, eliminar o modificar rutinas de entrenamientos. Sus funcionalidades son:

- **Inicio de sesión como administrador**: El administrador tendrá unas credenciales que le permitirá entrar a una cuenta con privilegios para poder administrar y realizar el mantenimiento de la aplicación web.
- **Gestión de ejercicios**: El administrador del sistema podrá añadir, eliminar o modificar los ejercicios de la base de datos del sistema. Esto garantizará que los usuarios tengan acceso a una amplia variedad de ejercicios para realizar sus entrenamientos
- **Gestión de rutinas**: El administrador del sistema podrá añadir, eliminar o editar las rutinas de la base de datos del sistema. Esto facilita la elección de rutinas adecuadas para diferentes objetivos y niveles de condición física.

Planteamiento del sistema y storyboard de la aplicación

Para hacer el sistema más navegable, siempre que aparezca el logo de nuestra web, el pulsar sobre él redirigirá al usuario a la página principal, es decir, al home del usuario.

Usuario

Registrarse e iniciar sesión

Cuando un usuario nuevo entra en la aplicación deberá crearse una cuenta para poder utilizar nuestra aplicación web. Para ello pulsará sobre *Registrarse* y le llevará a varias pantalla donde introducirá los datos necesarios y tendrá la capacidad de navegar libremente entre ellas, siempre que para avanzar haya rellenado los campos. Una vez rellenada la información en estas pantallas, la tarea acabará cuando se pulse el botón de *Registrarse* en la última pantalla. Cuando un usuario que ya tiene una cuenta quiere acceder a la aplicación web, introducirá los datos necesarios en la primera pantalla y si son correctos iniciará sesión cuando pulse el botón *Iniciar sesión*.

Crear rutina personalizada

Un usuario puede crear rutinas de ejercicios para realizarlas más adelante, para ello desde la página principal desplegará el menú, accederá a *Mis rutinas* y podrá elegir desde el icono "*+*" entrar en la pantalla de creación de una rutina. Una vez ahí, el usuario dará nombre a la rutina y desde el botón "*+*" se le redirigirá a una pantalla en la que podrá añadir ejercicios pulsando sobre ellos. Cuando haya terminado de añadir ejercicios le dará al botón *Añadir* y guardará la rutina creada.

Realizar un entrenamiento en tiempo real

Cuando el usuario vaya a realizar una rutina en el gimnasio, desplegará el menú principal y elegirá la opción de mis rutinas para acceder a las plantillas de entrenamientos que tiene por defecto o que él mismo ha creado. El usuario elegirá una rutina y empezará el entrenamiento, mientras vaya realizando los ejercicios que le indique la rutina, el usuario irá apuntando las series que haga. Cuando el usuario termine de realizar los ejercicios de la rutina pulsará el botón *Terminar* y le aparecerá información sobre el tiempo que ha realizado deporte y si quiere compartir el entrenamiento con sus amigos. Si el usuario quiere compartirlo pulsa el botón Sí, el botón No en caso contrario.

Seguir a amigos y ver sus entrenamientos

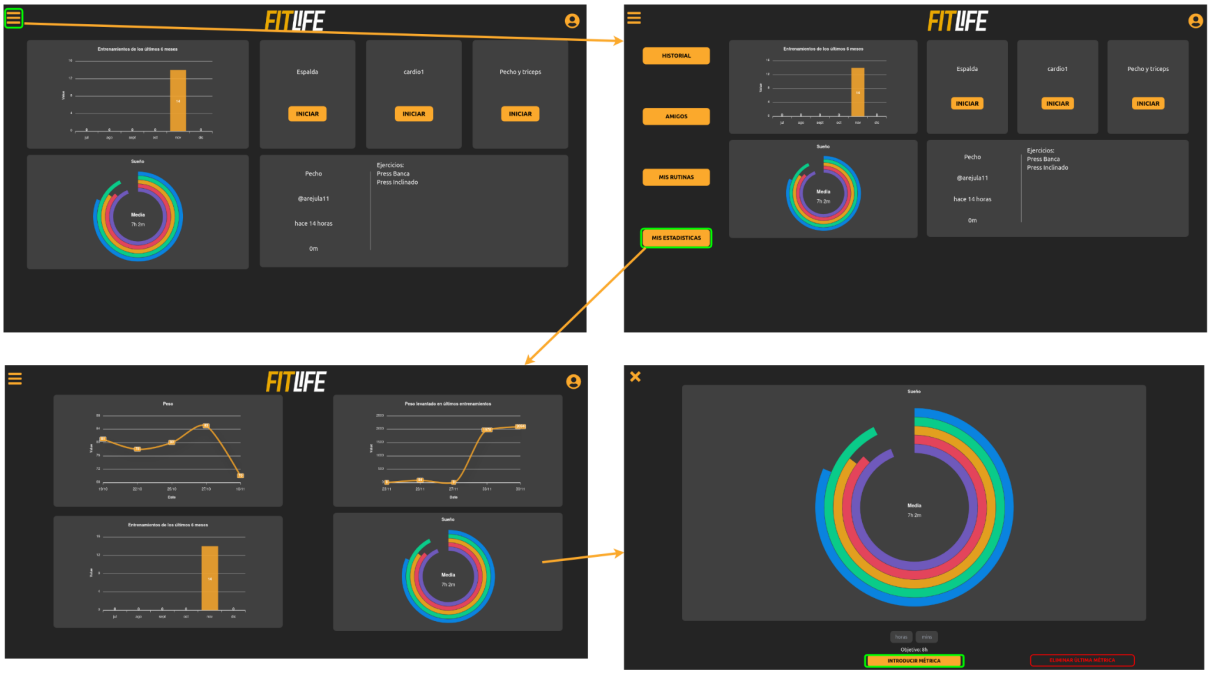
El usuario podrá seguir a sus amigos y ver los entrenamientos que realizan, para ello el usuario deberá desplegar el menú principal y elegir la opción de *Amigos* así podrá ver los últimos entrenamientos de todos sus amigos. Además para poder ver entrenamientos de sus amigos tendrá que seguirlos. El usuario deberá clicar en el icono de búsqueda. La nueva pantalla muestra una lista con todos los amigos que dispone. Para dejar de seguir un amigo, pulsar sobre la "X" correspondiente al amigo a eliminar.

Para añadir un nuevo amigo, buscarlo a partir de su correo o nombre de usuario en el buscador situado en la parte superior de la pantalla. Una vez encuentres al amigo a añadir, pulsar sobre el botón "Añadir" correspondiente.

Aceptar un amigo

Cuando un usuario añade a un amigo, no se comienzan a seguir instantáneamente. Antes, el usuario al cual le han solicitado, debe aceptar que quiere ser amigo. Para ello, el usuario deberá desplegar el menú principal y elegir la opción de *Amigos*. Pulsará sobre el icono de búsqueda y a continuación sobre *Solicitudes de Amistad*. Allí encontrará, si tiene, las solicitudes pendientes y para aceptarlas pulsará sobre *Aceptar* de aquellas que desee añadir.

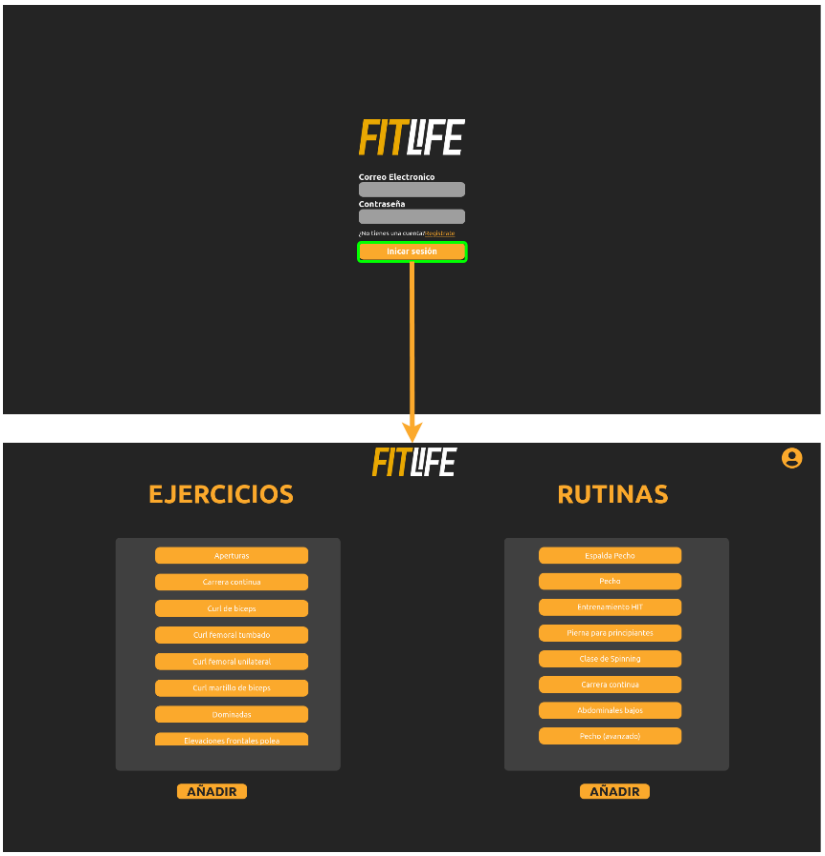
Ver tus estadísticas e introducir nuevas métricas



El usuario podrá visualizar sus estadísticas sobre él y sobre su evolución en sus entrenamientos desplegando el menú principal y eligiendo la opción de *Mis estadísticas*. Una vez en la página de las estadísticas podrá seleccionar cualquiera de las estadísticas que se le mostrarán y ver más información sobre ella. Además, rellenando los campos y eligiendo la opción de *Introducir nueva métrica* la información introducida en el formulario se añadirá a la estadística.

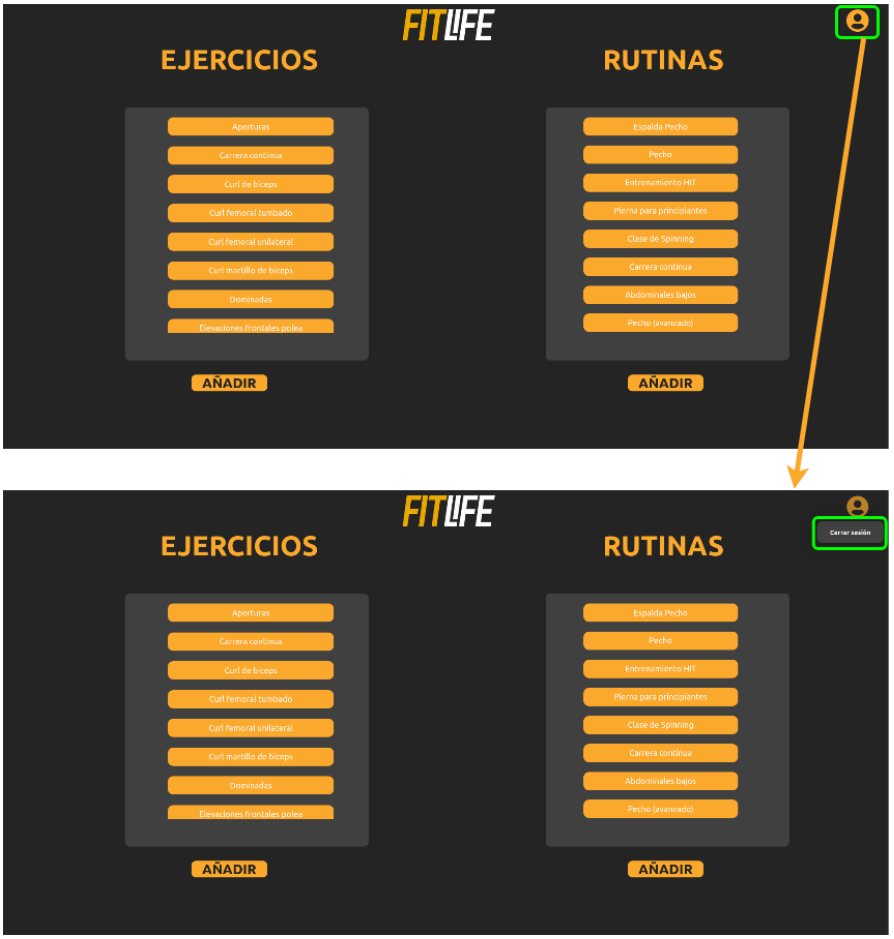
Administrador

Iniciar sesión



El administrador introduce las credenciales que se le han proporcionado y accede a su pantalla *home* al pulsar en el botón *Iniciar sesión.*

Cerrar sesión



El administrador para cerrar sesión debe encontrarse en su pantalla *home* y pulsar en el botón de su perfil en la esquina superior derecha. Se desplegará un botón que al pulsar cerrará la sesión.

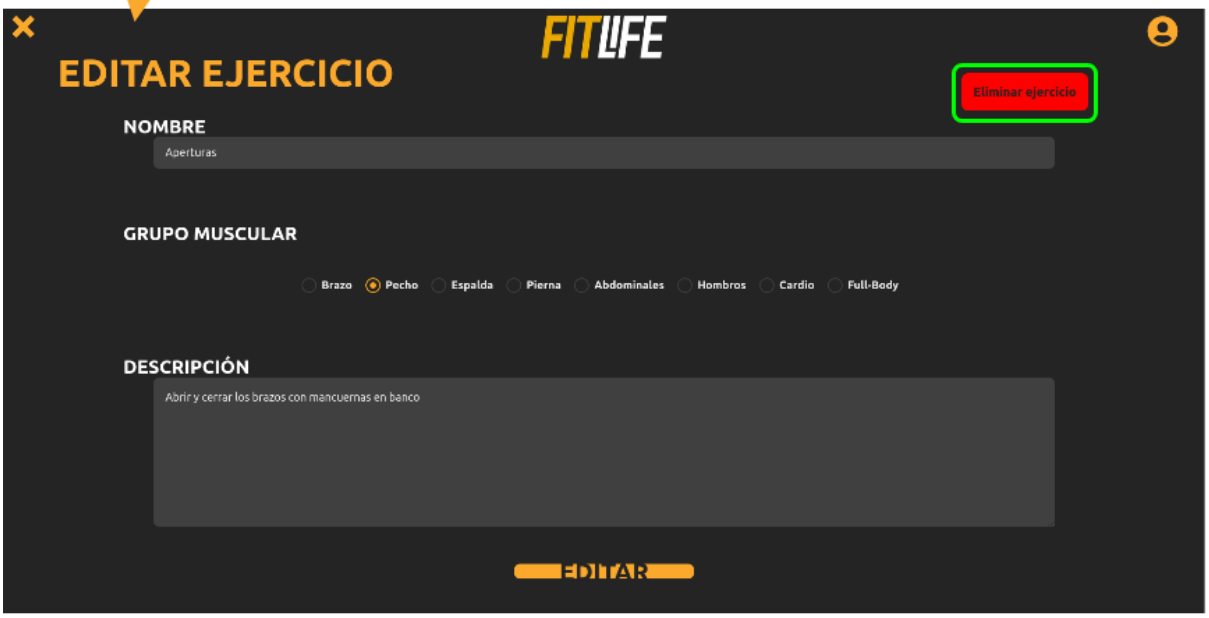
Añadir un ejercicio

Una de las principales funciones del administrador es gestionar los ejercicios disponibles para los usuarios. Por tanto, una funcionalidad del sistema es la de añadir nuevos ejercicios. Para realizarlo, el administrador debe pulsar en el botón de *Añadir* correspondiente a los ejercicios. Se abrirá una nueva página en la cual podrá añadir el nombre del ejercicio, seleccionar el grupo muscular que trabaja el ejercicio y una pequeña descripción del ejercicio; en la cual podrá añadir más detalle sobre los grupos musculares implicados y la explicación de la correcta ejecución del ejercicio. Para terminar de crear el ejercicio pulsar sobre *Añadir*.

Editar un ejercicio

El administrador también es capaz de editar los ejercicios previamente creados. Para realizar esta tarea el administrador debe pulsar en el ejercicio que quiere modificar, por ejemplo, el ejercicio *Aperturas* y se abrirá la pantalla correspondiente a ese ejercicio. Ahí podrá modificar el nombre, los grupos musculares y la descripción. Una vez haya acabado pulsa en el botón de *Editar.*

Eliminar un ejercicio



Para eliminar uno de los ejercicios que se encuentran en la lista, el administrador debe pulsar sobre el ejercicio correspondiente y posteriormente en el botón de *Eliminar ejercicio*.

Añadir una rutina

Al añadir una rutina, el administrador tendrá que pulsar el botón "*+*" para entrar en la pantalla de creación de una rutina. Una vez ahí, dará nombre a la rutina en el cuadro *"Nombre de la rutina"* y desde el botón "*+*" se le redirigirá a una pantalla en la que podrá añadir ejercicios pulsando sobre ellos. Cuando haya terminado de añadir ejercicios le dará al botón *Añadir* y guardará la rutina creada.

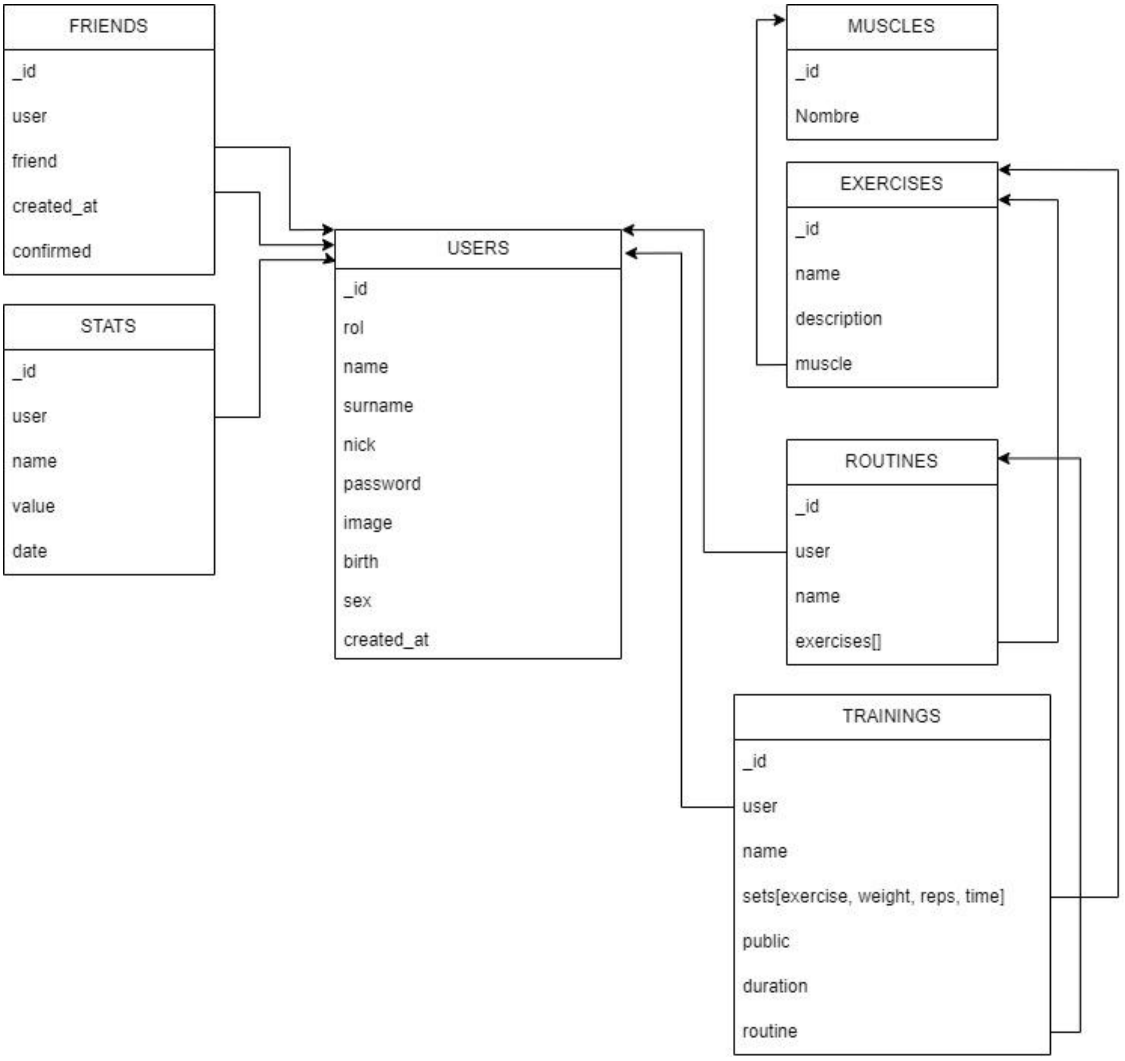
Editar una rutina

Para editar una rutina, el administrador pulsará una de las rutinas que se encuentren en la lista de rutinas ya existentes. Podrá cambiar el nombre de la rutina en el cuadro de texto con dicho nombre. También podrá añadir nuevos ejercicios pulsando en el icono “+” abriéndose así una pantalla en la que puede elegir el ejercicio a añadir. Si quiere eliminar un ejercicio, pulsará en la “x” correspondiente al ejercicio a borrar de la rutina. Para finalizar los cambios pulsar sobre *Guardar*.

Eliminar una rutina

Para eliminar una de las rutinas que se encuentran en la lista, el administrador debe pulsar en el botón de la rutina correspondiente que desea borrar. Entonces, pulsará sobre el botón *Eliminar*.

Modelo de datos del sistema



Esta imagen representa el esquema de nuestra Base de Datos. En ella se almacena toda la información necesaria para poder luego desarrollar el sistema y gestionar todos esos aspectos que nos preocupan y queremos gestionar. A continuación, se describe este modelo a grandes rasgos. Para ver los atributos de las tablas junto con su significado ver el siguiente apartado Creación de la Base de Datos.

Nuestra web gira y está dedicada al **Usuario**, y es por ello que esta tabla es esencial en la BD. Contiene aspectos relacionados con datos personales del usuario.

Uno de los objetivos de la aplicación es, como ya se ha dicho, el de proporcionar **Estadísticas** sobre la actividad física y la salud a la persona que utilice el sistema. Toda estadística posee, por tanto, un usuario asociado al que pertenece su valor.

Obviamente, al ser una web de actividad física existen **Entrenamientos** con una número determinado de diferentes **Ejercicios** que trabajan ciertos **Músculos**.

Los entrenamientos a su vez pueden ir guiados por **Rutinas** de ejercicios que existen para ayudar al usuario a trabajar ciertos aspectos físicos que éste desee fortalecer.

Por último, otro de los aspectos que nos interesan en el sistema es el poder compartir con tus **Amigos** tus resultados y entrenamientos. Esta tabla representa pares de amigos.

Creación de la Base de Datos

- **Ejercicios**

```
const {Schema, model} = require("mongoose")

const exerciseSchema = Schema({
        // Nombre del ejercicio
        name: {
                type: String,
                required: true
        },
        // Descripcion del ejercicio
        description: String,
        // Musculo que trabaja
        muscle: {
                type: Schema.ObjectId,
                ref: "Muscle"
        }
})
module.exports = model("Exercise", exerciseSchema, "exercises")
```

- **name:** representa el nombre del ejercicio. Es un parámetro de tipo String con valor obligatorio.
- **description:** se trata de una descripción del ejercicio en texto libre. Es un parámetro de tipo String.
- **muscle:** es el músculo principalmente trabajado en el ejercicio. Es un parámetro que hace referencia a una instancia de la tabla *muscleSchema*.

- **Amigos**

```
const {Schema, model} = require("mongoose")

const friendSchema = Schema({
        // Usuario
        user: {
                type: Schema.ObjectId,
                ref: "User"
        },
        // Usuario amigo
        friend: {
                type: Schema.ObjectId,
                ref: "User"
        },
        // Cuando empezaron a seguirse
        created_at: {
                type: Date,
                default: Date.now
        } ,
        confirmed: {
            type: Boolean,
            default: false
          }

})
module.exports = model("Friend", friendSchema, "friends")
```

- **user:** es el primero de los dos amigos que se siguen en la aplicación y que forman el par de esta "amistad". Se trata de una referencia a una instancia de la tabla *userSchema*.
- **friend:** es el segundo de los dos amigos que se siguen en la aplicación y que forman el par de esta "amistad". Se trata de una referencia a una instancia de la tabla *userSchema*.
- **created_at:** se trata de la fecha en la que empezaron a seguirse los dos usuarios. Es un parámetro de tipo Date que por defecto tendrá la fecha del momento en el que se introduce la relación entre usuarios en el sistema.
- **confirmed:** es una variable para saber si *friend* ha aceptado la solicitud a *user*. Por defecto será *false*, se convertirá en *true* cuando friend acepte la solicitud de amistad de user y pasarán a ser amigos y poder ver su información.

- **Músculos**

```
const {Schema, model} = require("mongoose")

const muscleSchema = Schema({
        // Nombre del músculo
        name: {
                type: String,
                required: true
        }
})
module.exports = model("Muscle", muscleSchema, "muscles")
```

- **name:** representa el nombre del músculo. Es un parámetro de tipo String con valor obligatorio.

- **Rutinas**

```
const {Schema, model} = require("mongoose")

const rutineSchema = Schema({
        // Nombre de la rutina
        name: {
                type: String,
                required: true
        },
        // Usuario que hace la rutina
        user: {
                type: Schema.ObjectId,
                ref: "User"
        },
        // Lista de ejercicios que realiza la rutina
        exercises: [{
                type: Schema.ObjectId,
                ref: "Exercise"
        }]
})
module.exports = model("Rutine", rutineSchema, "rutines")
```

- **name:** representa el nombre del ejercicio. Es un parámetro de tipo String con valor obligatorio.
- **user:** representa al usuario que creó la rutina. Se trata de una referencia a una instancia de la tabla *userSchema*.
- **exercises:** representa la lista de ejercicios que posee la rutina. Se trata de una lista que contiene referencias a instancias de la tabla *exerciseSchema*.

- **Estadísticas**

```
const {Schema, model} = require("mongoose")

const statSchema = Schema({
        // Nombre estadística
        name: {
                type: String,
                required: true
        },
        // Usuario a la que pertenece la estadística
        user: {
                type: Schema.ObjectId,
                ref: "User"
        },
        // Valor
        value: {
                type: Number,
                required: true
        },
        // Fecha en la que se recopiló
        date: {
                type: Date,
                default: Date.now
        }
})

module.exports = model("Stat", statSchema, "stats")
```

- **name:** representa el nombre de la estadística. Es un parámetro de tipo String con valor obligatorio.
- **user:** representa al usuario que creó la estadística. Se trata de una referencia a una instancia de la tabla *userSchema*.
- **value:** se trata del valor numérico de la estadística. Es un parámetro de tipo Number con valor obligatorio.
- **date:** representa la fecha en la que se recopiló el dato almacenado. Es un parámetro de tipo Date que por defecto tendrá la fecha del momento en el que se introduce la estadística en el sistema.

- **Entrenamientos**

```
const {Schema, model} = require("mongoose")

const trainingSchema = Schema({
        // nombre del entrenamiento, por defecto "Entrenamiento"
        // Si se sigue una rutina su nombre será el de la rutina
        name: {
                type: String,
                default: "Entrenamiento"
        },
        // usuario que la  realaliza
        user: {
                type: Schema.ObjectId,
                ref: "User"
        },
        // series
        sets: [{
                exercise:{
                        type: Schema.ObjectId,
                        ref: "Exercise"
                },
                weight: Number,
                reps: Number
        }],
        // si va a poder ser vista por sus amigos
        public:{
                type: Boolean,
                required: true
        } ,
        duration:{
                type: Number,
                required: true
        } ,
        // fecha creacion
        created_at: {
                type: Date,
                default: Date.now
        },
         routine:{
                type: Schema.ObjectId,
                ref: "Rutine"

           }
})
```

module.exports = **model**("Training", trainingSchema, "trainings")

- **name:** representa el nombre de la estadística. Es un parámetro de tipo String con valor obligatorio.
- **user:** representa al usuario que realizó el entrenamiento. Se trata de una referencia a una instancia de la tabla *userSchema*.
- **sets:** es una lista cuyas componentes están compuestas por:
  - **exercise:** representa un ejercicio realizado en el entrenamiento. Se trata de una referencia a una instancia de la tabla *exerciseSchema*.
  - **weight:** representa el peso levantado o empleado para realizar el ejercicio, ya sea de pesas, mancuernas, placas de una máquina, etc. Es un parámetro de tipo Number.
  - **reps:** representa el número de repeticiones hechas en la serie con ese peso en el ejercicio. Es un parámetro de tipo Number.
- **public:** representa si el entrenamiento será visible o no a los amigos del usuario que realiza la actividad. Es un parámetro de tipo Boolean que por defecto vale true.
- **duration:** es una variable que representa los minutos que ha durado el entrenamiento
- **created_at:** se trata de la fecha en la que se realizó el entrenamiento. Es un parámetro de tipo Date que por defecto tendrá la fecha del momento en el que se introduce el entrenamiento en el sistema.
- **routine:** representa la rutina a la que corresponde el entrenamiento. Se trata de una referencia a una instancia de la tabla *rutineSchema*.

- **Usuarios**

```
const {Schema, model} = require("mongoose")

const userSchema = Schema({
        // Nombre
        name: {
                type: String,
                required: true
        },
        // Apellidos
        surname: {
                type: String,
                required: true
        },
        // Nombre Usuario
        nick: {
                type: String,
                required: true,
                unique: true
        },
        // correo electrónico
        email: {
                type: String,
                required: true,
                unique:true
        },
        // Contraseña cuenta
        password: {
                type: String,
                required: true
        },
        // foto perfil
        image: {
                type: String,
                default: "default.png"
        },
        // Fecha nacimiento
        birth: Date,
        // Sexo
        sex: String,
        // Fecha creación cuenta
        created_at: {
                type: Date,
                default: Date.now
```

```
        },
        // Usuario o admin
        rol: {
                type: String,
                default: "usuario"
        }

})
module.exports = model("User", userSchema, "users")
```

- **name:** es el nombre del usuario. Es un parámetro de tipo String con valor obligatorio.
- **surname:** son los apellidos del usuario. Es un parámetro de tipo String con valor obligatorio.
- **nick:** es el nickname del usuario, es decir, el apodo del usuario dentro de la aplicación. Es un parámetro de tipo String con valor obligatorio y valor único dentro de la BD.
- **email:** es el correo electrónico del usuario. Es un parámetro de tipo String con valor obligatorio y valor único dentro de la BD.
- **password:** es la contraseña del usuario en la aplicación. Es un parámetro de tipo String con valor obligatorio.
- **image:** es el nombre del archivo que contiene la foto de perfil del usuario. Es un parámetro de tipo String con valor por defecto “default.png”.
- **birth:** es la fecha de nacimiento del usuario. Es un parámetro de tipo Date.
- **sex:** es el sexo del usuario. Es un parámetro de tipo String.
- **created_at:** se trata de la fecha en la que se creó la cuenta. Es un parámetro de tipo Date que por defecto tendrá la fecha del momento en el que se crea el usuario en el sistema.
- **rol:** es el tipo de usuario dentro del sistema: “usuario” o “admin”. Es un parámetro de tipo String que por defecto tiene el valor “usuario”.

Funcionalidad

- **Ejercicio:** Hemos realizado varias funcionalidades para poder gestionar los ejercicios que se guardan en la base de datos:
  - *add*: añade un ejercicio a la base de datos con un nombre, descripción y músculo determinado.
  - *eliminate*: elimina un determinado ejercicio creado con anterioridad en la base de datos.
  - *exercises*: lista los ejercicios que trabajan un músculo determinado.
  - *allExercises*: lista todos los ejercicios de todos los músculos.
  - update: actualiza un determinado ejercicio creado con anterioridad en la base de datos.
  - *exercisesById*: devuelve la información del ejercicio que corresponde con un id determinado.

- **Entrenamiento:** Para gestionar los entrenamientos que realiza un usuario hemos creado varias funciones para interactuar con la base de datos:
  - *add*: añade un entrenamiento con un nombre y opción de visibilidad pública determinada y unas series de ejercicios que incluyen las repeticiones y el peso si es necesario, además de la fecha de creación, por defecto la fecha actual.
  - *eliminate*: que elimina un entrenamiento específico creado con anterioridad.
  - *trainingsUser*: devuelve la información de los entrenamientos de un usuario en específico.
  - *trainings*: devuelve la información de los entrenamientos de tus amigos ordenados por la fecha de realización.
  - *trainingsById:* devuelve la información del entrenamiento que corresponde con un id determinado
  - *trainingsUserLastPage*: lista los últimos 5 entrenamientos de un usuario determinado
  - *training*: devuelve la información de un entrenamiento determinado sin paginación.

- **Estadística:** Hemos creado funciones para interactuar con las estadísticas del usuario en la base de datos:
  - *add*: añade un valor y el nombre de la estadística a la que corresponde con la fecha actual.
  - *eliminate*: elimina un valor de una estadística creada con anterioridad.
  - *eliminateLast*: elimina el último valor de una estadística determinada.
  - *stats*: devuelve la información de todos los valores y fecha de las estadísticas con un nombre específico.
  - *trainings:* devuelve información sobre el número de entrenamientos que ha hecho el usuario durante los últimos 6 meses.

- *statsLastFive*: devuelve información sobre los últimos 5 valores de una estadística determinada

- **Amigo :** Tenemos varias funciones para gestionar los amigos de nuestra base de datos:
  - *add*: añade una relación mutua entre dos usuarios como amigos.
  - *eliminate*: elimina una relación de amistad específica creada con anterioridad.
  - *friends*: devuelve la información de los usuarios que tienen una relación como amigos en la base de datos con un usuario determinado.
  - *numFriends*: devuelve el número de amigos que tiene un usuario determinado
  - *requests*: lista las solicitudes de amistad que tiene el usuario
  - *confirm*: acepta la solicitud de amistad de un usuario determinado

- **Músculo:** Para gestionar los grupos musculares que intervienen en un ejercicio de nuestra base de datos hemos creado las siguientes funciones:
  - *add*: crea un nuevo músculo a la base de datos si no existe
  - *eliminate*: se encarga de eliminar un músculo el cual ha tenido que ser creado previamente.
  - *muscles*: muestra todos los músculos de la base de datos.

- **Rutina:** Hemos creado para gestionar las rutinas de nuestra base de datos las siguientes funciones:
  - *add*: añade una rutina nueva con unos ejercicios específicos que tienen que existir en la base de datos.
  - *eliminate*: se encarga de eliminar una rutina en específico creada con anterioridad por el mismo usuario.
  - *update*: actualiza la información de una rutina determinada creada con anterioridad y por el mismo usuario.
  - *rutines*: muestra la información de las rutinas del mismo usuario que la ejecuta y las del administrador.
  - *rutinesUser*: muestra la información de las rutinas de un usuario determinado
  - *rutinesAdmin*: muestra la información de las rutinas del administrador
  - *rutinesScroll*: muestra la información de las rutinas tanto del usuario como del administrador pero sin paginar
  - *routine:* muestra la información de una rutina determinada
  - *favRoutines:* devuelve la información de las 3 rutinas más utilizadas del usuario

- **Usuario:** Para gestionar los usuarios que intervienen en la base de datos hemos creado las siguientes funciones:
  - *register*: crea un usuario que no existía con anterioridad, cabe destacar que guardamos la contraseña encriptada.
  - *login*: inicia sesión de un usuario existente y crea un token para realizar funciones que necesiten esta identificación.
  - *profile*: devuelve la información de un usuario que esté creado con anterioridad.
  - *update* modifica la información de un usuario existente en la base de datos.
  - *upload*: sube una imagen a nuestra base de datos y la asigna al usuario que ha ejecutado la petición.
  - *avatar*: devuelve la foto que está asignada a un usuario cualquiera que esté en la base de datos.
  - *counters*: muestra el número de amigos que tiene un usuario en específico y el número de entrenamientos que ha realizado.
  - *searchUsers*: lista los usuarios que coinciden con la búsqueda realizada y aún no son amigos del usuario y tampoco han enviado una solicitud de amistad al usuario.
  - *searchRequests*: lista los usuarios que coinciden con la búsqueda realizada y han enviado una solicitud de amistad al usuario pero aún no ha sido aceptada.

Diferencias entre la primera versión planteada y el sistema desarrollado

El primer día que comenzamos a realizar el proyecto tuvimos muy claro que este era uno de los primeros grandes proyectos que hemos realizado en la universidad. Además era también de las primeras veces que trabajamos con estas tecnologías. Por lo tanto, sabíamos que gran parte del tiempo lo íbamos a invertir en aprender tanto como organizarse mejor y sobre las tecnologías usadas. Esto hizo que desde el primer momento planteáramos una versión adecuada al tiempo disponible y a los conocimientos. Nuestra versión presentada en la primera práctica es prácticamente idéntica a la versión implementada. Logrando cumplir el primer objetivo que nos propusimos; plantear una idea realista y que seamos capaces de desarrollar.

La única gran diferencia que no se ha llegado a implementar es la conexión de nuestro sistema con aplicaciones externas que permitan la obtención de datos. En la primera versión se propuso esta idea de obtener datos desde otras aplicaciones deportivas de recolección de datos permitiendo que los usuarios pudiesen unificar todos sus datos en un único sistema de información. Sin embargo, este objetivo ha sido imposible de cumplir por la falta de tiempo y sobre todo por la falta de conocimientos. Es cierto, que encontramos ciertas aplicaciones que permiten la conexión a su sistema mediante una API. Pese a ello, no hemos sabido cómo integrarlo en nuestro sistema. Probablemente con más tiempo disponible para invertir en este proyecto, se podría haber logrado.

El otro pequeño detalle que no se ha cumplido del todo es hacer nuestro sistema totalmente *responsive*. Permitiendo su correcta visualización en dispositivos más pequeños como *tablets* o *smartphones.* Si es cierto que se ha logrado que ciertos componentes de nuestros sistema sean *responsive.* Por ejemplo, el texto de nuestro sistema si lo es. Al reducir el tamaño de la pantalla también se disminuye el tamaño del texto. Para ello hemos utilizado el framework de CSS *tailwind.* Sin embargo, nuestra falta de conocimiento al comienzo del desarrollo del sistema ha provocado que para implementar la funcionalidad debemos invertir más tiempo del que disponemos.

Despliegue e instalación de la aplicación

Para realizar el despliegue de nuestro sistema hemos diseñado dos contenedores, en el primer contenedor se despliega el servidor de nuestro sistema, que en este caso se trata únicamente de la base de datos. En el segundo contenedor se aloja el cliente, es decir, la web.

Para su despliegue es necesario tener instalado *Docker.* Es necesario que su servicio esté activo, en caso que esté sin funcionar hay distintas formas de arrancarlo.

Si nuestro sistema operativo es MacOs la mejor forma es abriendo la aplicación y este servicio se ejecutará automáticamente. Para comprobar que está activo tenemos que ver el icono de docker en la barra superior del menú.

Si nuestro sistema operativo es Linux el mejor método es ejecutar en la terminal el comando :

$ sudo systemctl start docker

Una vez que *docker* está activo podemos desplegar nuestro sistema. En el directorio docker encontramos los siguientes ficheros y subdirectorios:

```
docker
├── despliegue.sh
├── docker-compose.yaml
├── web
│   ├── Dockerfile
│   └── ...
└── api
    ├── Dockerfile
    └── ...
```

El script *despliegue.sh* se encarga de desplegar el sistema. En el fichero *docker-compose.yaml* se encuentran la configuración de los puertos de los contenedores y se indica la ruta del fichero *Dockerfile* con la imagen de cada contenedor. Y finalmente dentro de cada subdirectorio se encuentran los ficheros *Dockerfile* correspondientes a la base de datos y a la página web.

Por lo tanto para desplegar el sistema tenemos que dar permiso de ejecución al script y ejecutarlo, una maner sencilla es ejecutando en el directorio docker los siguientes comandos:

$ chmod u+x despliegue.sh

$ ./despliegue.sh

En el caso que queramos parar y eliminar los contenedores se debe ejecutar el comando:

$ docker compose down

Uso de la aplicación

Para poder comprobar el funcionamiento de la aplicación como usuario, es necesario crear una cuenta o iniciar sesión con una ya existente.

Para registrarse cabe destacar que no se puede repetir un correo electrónico o un nickname de un usuario que ya existe en la aplicación. Una vez registrado, el usuario deberá iniciar sesión.

El usuario podrá navegar libremente y utilizar las funciones de la aplicación. Para poder observar el máximo potencial de la aplicación hemos creado varios usuarios que ya tienen información registrada. Este usuario ya tiene amigos añadidos por lo que podrá ver los entrenamientos de los amigos, tiene rutinas creadas que podrá utilizar para realizar entrenamientos, tiene entrenamientos realizados para ver el historial de entrenamientos realizados y ver estadísticas sobre estos, tiene añadidos manualmente estadísticas de peso y de sueño… Para acceder a la **cuenta de usuario** se deberá iniciar sesión con la siguiente información:

- **Correo electrónico:** pabloangusto@gmail.com
- **Contraseña**: contrasenapabload

Os proporcionamos un segundo usuario para permitiros probar la funcionalidad de amigos y solicitudes de amistad:

- **Correo electrónico:** arejula@gmail.com
- **Contraseña**: arejula

Además, si se quiere comprobar el funcionamiento de la aplicación como administrador se debe iniciar sesión con una cuenta de administrador ya creada porque son cuentas que se tienen que crear manualmente. Para entrar en una **cuenta de administrador** se debe iniciar sesión con la siguiente información:

- **Correo electrónico**: pabloangusto2@gmail.com
- **Contraseña**: 1234567

Cronograma

	Pablo Angusto	Jose Miguel Florentín	Miguel Aréjula
Reuniones	28 h	28 h	28 h
Trabajo individual	42 h	42 h	42 h
Trabajo colectivo	26 h	26 h	26 h

- **Esfuerzo dedicado y dificultades:**

El esfuerzo dedicado a la realización del sistema ha sido notable dada la gran carga de trabajo del semestre y la inexperiencia con el desarrollo de un sistema de información. Era la primera vez de algún integrante que trabajaba con esta tecnología, y el resto tenía 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 sintaxis, estructuras, etc. 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 utilizamos 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 hacíamos *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 ningún 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.

- **Conocimientos adquiridos:**

El amplio esfuerzo requerido nos ha llevado a adquirir muchos nuevos conceptos y metodologías. Al comenzar el curso prácticamente no nos desenvolvíamos en estos entornos y ahora consideramos tener unos conocimientos básicos que nos han permitido manejarnos con solvencia.

Así, hemos aprendido **React**, que requiere tanto **HTML** para diseñar los elementos básicos de la web como **Javascript** para añadirle una mayor funcionalidad. Con el fin de gestionar la base de datos que contiene la información del sistema ha sido necesario el manejarnos con el framework de *express* con **Node**.

Como ya hemos comentado en el punto anterior, también nos hemos familiarizado con el uso de GitHub y algunas de sus funcionalidades para agilizar el trabajo en equipo así como el uso de Docker para que todos los integrantes del grupo pudieran tener el mismo sistema operativo, mismas versiones, etc.

- **Funcionalidades a implementar en un futuro:**

Al principio, antes de empezar con la implementación de nuestra aplicación web, marcamos unos objetivos a realizar que por falta de tiempo o por su dificultad hemos tenido que dejar de lado y también nuevas ideas que se nos han ocurrido durante el desarrollo de nuestra aplicación. Uno de estos casos ha sido la conexión de nuestra web con una aplicación externa mediante su API, como podría ser Strava, para obtener y sincronizar información sobre algunos entrenamiento más específicos de cardio, como correr, y estadísticas de salud. Otra funcionalidad que nos hubiera gustado tener en nuestra aplicación es el almacenamiento y gestión de fotos de perfil para los usuarios, pero no nos pareció algo de gran relevancia dado el objetivo de nuestra aplicación.

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
Documentación oficial de Docker
https://docs.docker.com/

ANEXO

Anexo I : Fichero Dokerfile utilizado durante el desarrollo del sistema

Para el desarrollo del sistema se decidió trabajar en un único sistema operativo, para ello lo ideal es la creación de un contenedor de Ubuntu. El cual permite que todos los integrantes trabejen en las mismas condiciones y no aparezcan problemas de versiones y compatibilidades durante el desarrollo.

```
FROM ubuntu:latest

RUN apt update -y
RUN apt upgrade -y

RUN apt install zsh  git sudo man    -y

RUN apt install curl -y
RUN apt install wget -y
RUN curl -fsSL https://deb.nodesource.com/setup_19.x | sudo -E bash - && sudo apt-get install -y nodejs

RUN useradd --create-home --shell /bin/zsh vscode || true
RUN adduser vscode sudo

RUN echo '%sudo ALL=(ALL) NOPASSWD:ALL' >>/etc/sudoers
RUN echo "vscode:vscode" | chpasswd

COPY ./.zshrc /home/vscode/
```