Diseño y desarrollo de una plataforma de gestión de gimnasios

Memoria de Proyecto Final de Máster

Máster universitario de Desarrollo de Sitios y Aplicaciones Web

Autor: Ignacio Vicent Salvador

Consultor: Carlos Caballero González

Profesor: César Pablo Córcoles Briongos

03 de enero de 2022

Créditos/Copyright

Dedicatoria/Cita

A mi familia y a mi pareja por apoyarme a seguir estudiando, por estar ahí cuando lo he necesitado y animarme a no tirar la toalla en los momentos más duros. Sin ellos, este trabajo no hubiera sido posible.

Resumen

El objetivo de este trabajo es el diseño, desarrollo y documentación de una aplicación web para gestionar gimnasios. Los usuarios, de forma autónoma, podrán consultar sus rutinas de entrenamiento, anotar el tiempo que pasan en el gimnasio, consultar estadísticas y modificar sus datos personales.

La aplicación se hará con un diseño responsive, por lo que se podrá adaptar a cualquier tamaño de pantalla, facilitando a los usuarios de los gimnasios el uso de esta aplicación, ya sea por ordenador, tablet o móvil.

Para el desarrollo de la aplicación se ha utilizado el framework de ExpressJS para el back-end y Angular para el front-end, además se ha utilizado HTML5 y CSS3 para la maquetación.

Palabras clave:
Aplicación web, Angular, ExpressJS, gimnasio, gestión de gimnasios, rutinas de entrenamiento

Abstract

The objective of this work is the design, development and documentation of a web application to manage gyms. Users, autonomously, will be able to consult their training routines, note the time they spend in the gym, consult statistics and modify their personal data.

The application will be made with a responsive design, so it can be adapted to any screen size, making it easier for gym users to use this application, either by computer, tablet or mobile.

For the development of the application, the ExpressJS framework has been used for the back-end and Angular for the front-end, in addition, HTML5 and CSS3 have been used for the layout.

Keywords:
Web application, Angular, ExpressJS, gym, gym management, workout routines

Índice

Figuras y tablas

Lista de imágenes, tablas, gráficos, etc., numeradas, con títulos y las páginas en las que aparecen.

Índice de figuras

Índice de tablas

1. Introducción/Prefacio

1.1 Contexto y justificación del trabajo

Para ciertos gimnasios con recursos limitados, disponer de una herramienta para gestionar su gimnasio puede parecer imposible. Por eso, en esta memoria, se ha querido plasmar el proyecto de una aplicación web que solucione este problema. **GymFit**, que es como hemos llamado a la herramienta de código abierto que vamos a desarrollar, está pensada para que pueda funcionar en cualquier ordenador y ofrecer tanto a la empresa como a los usuarios todas las funcionalidades más importantes de forma online.

Existen varios tipos de gimnasios en España, los hay que son franquicias, gimnasios de élite para profesionales, gimnasios públicos y pequeños gimnasios de barrio. **GymFit** se centra en este último grupo, los pequeños gimnasios de barrio.

La funcionalidad principal de nuestra herramienta es la gestión de gimnasios para los empleados y los usuarios. El acceso a la aplicación web se realiza mediante usuario y contraseña, y según el rol de éste, el sistema mostrará una vista u otra. Existen tres roles principales, por un lado, tenemos el rol de usuario que permite registrar el tiempo que pasan en el gimnasio y visualizar su rutina de entrenamiento. Por otro lado, tenemos el rol de monitor, que puede ver los usuarios que tiene asignados y crearle las rutinas. Finalmente, tenemos el rol de administrador, que se encarga de crear las cuentas de monitor, los ejercicios que se pueden realizar en el gimnasio, visualizar las incidencias y las diferentes estadísticas.

Se pueden encontrar en el mercado diferentes aplicaciones web relacionadas con la gestión de gimnasios, pero éstas o bien son muy caras o bien ofrecen demasiadas funcionalidades que no se utilizarán nunca. Por eso hemos querido ofrecer **GymFit** centrándonos solo en lo más importante de la gestión de gimnasios.

1.2 Objetivos del trabajo

1.2.1 Principales

Objetivos generales del proyecto:

- Diseñar, analizar y desarrollar una aplicación web que permita la gestión de gimnasios de forma sencilla. En el mercado existen multitud de aplicaciones web para gestionar gimnasios, algunas de ellas las analizaremos más adelante, pero todas se centran en un usuario experto, por eso el objetivo es crear una aplicación sencilla para que todos los usuarios puedan utilizarla.
- Unificar los datos en una única plataforma centralizada. Hasta ahora existen diferentes plataformas en los gimnasios (reservas, invitados, socios) que tienen diferentes interfaces y no interoperan entre sí, el objetivo es tener una única plataforma desde la que se encuentren todos los datos, facilitando la creación de futuras características.

- Conseguir que la aplicación sea responsive para que se pueda utilizar desde cualquier dispositivo. Muchas de las aplicaciones que usan los gimnasios son sólo aplicaciones de escritorio, por lo que los usuarios no pueden interactuar si no están enfrente del ordenador, nuestro objetivo es poder crear una aplicación responsive para que puedan utilizar la aplicación desde cualquier lugar.
- Digitalizar la información relativa a las fichas de usuarios y monitores. Algunos de los gimnasios, todavía utilizan las fichas de registro en papel y no disponen de un sistema digitalizado, el objetivo es digitalizar toda esa información para que se pueda consultar desde cualquier lugar.
- Documentar las fases del desarrollo de una aplicación web, desde la idea del proyecto, pasando por planificación, análisis del mercado, diseño y desarrollo. Se quiere realizar una documentación de todo el proceso, no solo del código, por lo que cualquier empresa o persona pueda ver la idea del proyecto, como lo hemos planificado para llevarlo a cabo, en caso de que esa persona decida desarrollar alguna funcionalidad podrá basarse en lo que ya tenemos documentado, que se pueda ver el análisis de mercado que hemos realizado, por si en un futuro es necesario realizar uno nuevo y el diseño que se ha elegido para la aplicación.
- Realizar un proyecto con licencia de código abierto. La gran mayoría de aplicaciones que ofrecen soluciones a los gimnasios son de código cerrado y de pago, por lo que si quieres que implementen alguna funcionalidad hay que hablar con la empresa y lo más seguro que te cobren por ello, por eso queremos ofrecer una solución de código abierto, para que cada empresa pueda disponer del código y modificar o mejorar lo que necesite.

1.2.2 Secundarios

Objetivos adicionales que enriquecen el TF:

- Al ser un proyecto de código abierto, se puede disponer del código fuente para que cualquiera pueda ampliar su funcionalidad. La mayoría de productos que se ofertan, no te muestran el código, por lo que no sabes cómo está implementado o si es un sistema robusto, nosotros queremos que las empresas puedan ver el código para que vean que nuestra aplicación sigue buenas prácticas, es robusta y en caso de que necesiten añadir alguna funcionalidad, puedan implementarla sin ningún problema.
- Profundizar en el conocimiento de las tecnologías para el desarrollo de una aplicación web, sobre todo en Angular. Para ello vamos a realizar un par de cursos y tutoriales, además de leer algunos capítulos de los libros que tenemos disponibles, así podemos realizar una mejor implementación del código y estructurarlo de una forma mucho más eficiente y organizada.
- Diseñar una interfaz responsive que permite adaptarse a cualquier dispositivo. Algunas de las soluciones que se ofrecen en el mercado son solo para ordenadores, que dependen del sistema operativo, nosotros queremos que la aplicación sea responsive para que todos los usuarios puedan utilizarla, independientemente del sistema operativo y de donde se encuentren.

1.3 Enfoque y método seguido

La metodología de desarrollo seguida en este proyecto ha sido una metodología ágil, que se caracteriza por un desarrollo incremental con reuniones con el cliente. Las reuniones se realizan con la finalidad de comprobar que el proyecto está avanzando por el camino correcto, además de comentar mejoras.

En este proyecto el desarrollo ha sido llevado a cabo por un único miembro, por lo que los roles principales de *SCRUM* y el de cliente los ha realizado la misma persona.

Se ha elegido un desarrollo ágil porque aporta flexibilidad y eficiencia cuando se trabaja en proyectos complejos, con requisitos cambiantes o poco definidos, mejora la experiencia del cliente al poder ver cada 10 días los avances del proyecto y minimiza los riesgos, ya que si el cliente ve que el proyecto está yendo por un camino equivocado lo puede comunicar al principio del mismo y no una vez esté terminado.

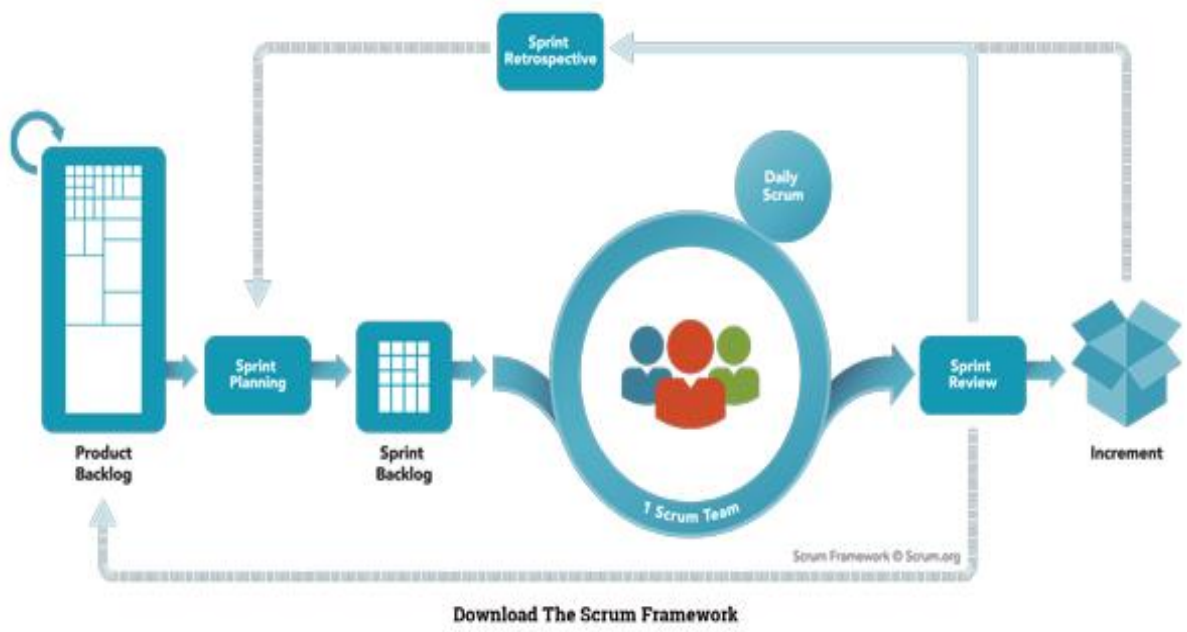


Figura 1: Ciclo de vida SCRUM

En la *figura 1* se muestra un diagrama del ciclo de vida *SCRUM*. Se puede observar las distintas fases de esta metodología, las cuales se explican a continuación:

- El ***product backlog*** es el listado total de tareas que deben desarrollarse para que salga adelante el proyecto. Al tratarse de una metodología ágil, estas tareas pueden modificarse a lo largo del desarrollo.
- El ***sprint backlog*** es el grupo de tareas que se van a desarrollar en el sprint actual. Para decidir qué tareas pasan del *product backlog* al *sprint backlog*, se realiza el *sprint planning.* Cada tarea

tiene un peso, que se asigna en base a la dificultad o prioridad de la tarea. La carga de trabajo de cada *sprint backlog* está planificada en base a la duración de cada sprint.

- El ***sprint*** es el periodo en el cual se desarrollan las tareas incluidas en el *sprint backlog*. Para este proyecto la duración será de 10 días, pero puede tener una duración de entre una y cuatro semanas. El resultado de cada iteración es un producto funcional, el cual, ha incrementado su valor y es lo que va percibiendo el cliente.

El equipo de desarrollo de *SCRUM* está compuesto por los siguientes roles:

- El ***Product Owner*** es quien tiene la visión de negocio, se encarga de hablar con el cliente y prioriza las tareas que se incluyen en cada *sprint backlog*. Solo puede haber un *Product Owner* y este puede ser parte del equipo de desarrollo.
- El ***Scrum Manager*** se encarga de eliminar impedimentos y coordinar al equipo. Vela para que las reglas de *SCRUM* se cumplan.
- El ***Team Scrum*** es el equipo multifuncional y auto-organizado encargado de desarrollar el producto. También se encargan de estimar las tareas del *product backlog*.

Finalmente, en la metodología *SCRUM*, de forma diaria se realizan las *Daily Scrum* que son reuniones entre el equipo de desarrollo y el *Scrum Manager* donde se ponen en común el trabajo realizado durante el último día. Para ello se responden las siguientes tres preguntas:

- ¿Qué hice ayer?
- ¿Qué estoy realizando o tengo pensado realizar hoy?
- ¿Tengo algún impedimento para continuar trabajando?

Estas reuniones facilitan la comunicación y el trabajo en equipo, permitiendo que todos sepan en cada momento en lo que están trabajando el resto de compañeros. Además de que ayuda a facilitar los problemas que puedan surgir durante el desarrollo de la aplicación.

1.4 Planificación del trabajo

Empezamos el proyecto el día 15 de septiembre de 2021 y terminamos el día 3 de enero de 2022, suponiendo que hacemos 4 horas al día, es decir, 20 horas semanales, tenemos un total de 444 horas.

Para el proyecto hemos definido sprints de 10 días, por lo que cada uno tendrá un total de 40 horas, con lo que tendremos un total de 11. En la Tabla 1 se desglosa los diferentes sprints del proyecto, así como las tareas que se han realizado en cada uno de ellos.

En la Tabla 2 se puede ver en detalle las tareas que tiene cada fase del proyecto, donde se detalle el inicio y fin de cada tarea, el número de días requeridos y las dependencias entre tareas.

En la Figura 2 se puede observar el diagrama de Gantt con la estimación del esfuerzo en días para cada fase.

Sprint	Fecha de inicio	Fecha de fin	Tareas realizadas
Sprint 1	15/09/2021	24/09/2021	Búsqueda y organización información Definir el proyecto con el consultor Introducción, contexto y justificación Objetivos del trabajo Enfoque y método seguido
Sprint 2	25/09/2021	04/10/2021	Planificación del trabajo Elaboración del diagrama de flujo Elaboración de wireframes Diseño de prototipos
Sprint 3	05/10/2021	14/10/2021	Diseño de prototipos Diseño de la base de datos Redacción y corrección memoria
Sprint 4	15/10/2021	24/10/2021	Redacción y corrección memoria Configuración del espacio de trabajo Programación de la aplicación
Sprint 5	25/10/2021	03/11/2021	Programación de la aplicación
Sprint 6	04/11/2021	13/11/2021	Programación de la aplicación
Sprint 7	14/11/2021	23/11/2021	Programación de la aplicación Maquetación y estilos
Sprint 8	24/11/2021	03/12/2021	Maquetación y estilos Pruebas de funcionamiento Corrección de errores Elaboración video aplicación Documentar desarrollo aplicación
Sprint 9	04/12/2021	13/12/2021	Documentar desarrollo aplicación Programación de la aplicación
Sprint 10	14/12/2021	23/12/2021	Programación de la aplicación Documentación y revisión del proyecto
Sprint 11	24/12/2021	03/01/2022	Documentación y revisión del proyecto Elaboración presentación

Tabla 1: Desglose de los sprints realizados durante el proyecto

	Nombre de la tarea	Duración	Inicio	Fin	Predecesores
1	Inicio del proyecto	14 días	15/09/2021	28/09/2021	
	1.1 Búsqueda y organización de la información	4 días			
	1.2 Definir el proyecto y consensuar con el consultor	2 día			1.1
	1.3 Introducción, contexto y justificación	2 días			1.2
	1.4 Objetivos del trabajo	1 día			1.3
	1.5 Enfoque y método seguido	1 día			1.4
	1.6 Planificación del trabajo	4 días			1.5
2	Diseño de la aplicación	18 días	29/09/2021	16/10/2021	
	2.1 Elaboración del diagrama de flujo	1 día			1
	2.2 Elaboración de wireframes	2 días			2.1
	2.3 Diseño de prototipos	5 días			2.2
	2.4 Diseño de la base de datos	2 días			2.3
	2.5 Redacción y corrección memoria	8 días			2.4
3	Desarrollo de la aplicación	50 días	17/10/2021	05/12/2021	
	3.1 Instalación y configuración del espacio de trabajo	1 día			
	3.2 Programación de la aplicación	33 días			3.1
	3.3 Maquetación y estilos	6 días			
	3.4 Pruebas de funcionamiento	2 días			3.2
	3.5 Corrección de errores	2 días			3.4
	3.6 Elaboración video funcionamiento aplicación	1 día			3.5
	3.7 Documentar el desarrollo aplicación	5 días			
4	Fase final del proyecto	29 días	06/12/2021	03/01/2022	
	4.1 Programación de la aplicación	15 días			
	4.2 Documentación y revisión del proyecto	11 días			4.1
	4.3 Elaboración presentación	3 días			4.2
	Total	111 días / 444 horas			

Tabla 2: Planificación del trabajo

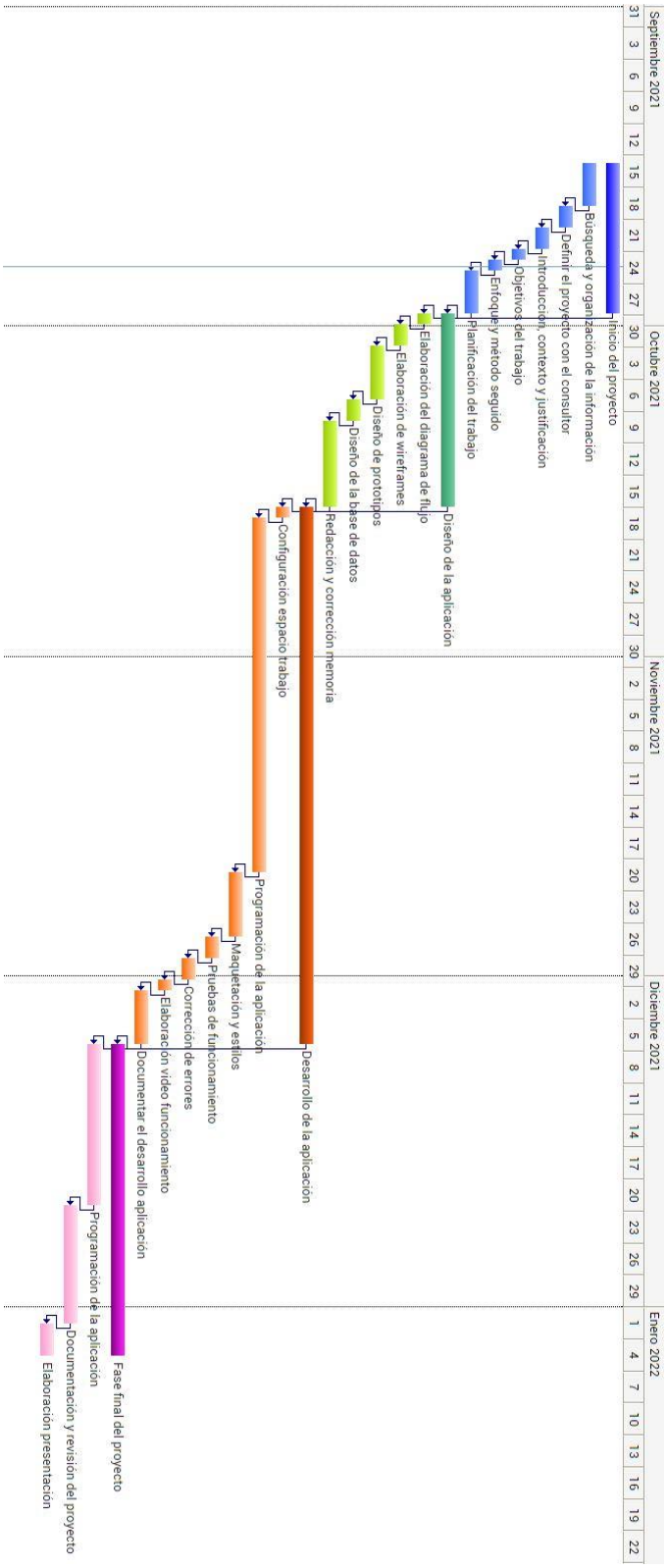


Figura 2: Diagrama de Gantt

1.5 Presupuesto

Suponemos que se trabaja durante 4h cada día, por lo que el proyecto tendrá un total de 111 días que son lo mismo que 444 horas. El coste medio de un programador freelance en España es de 20€/hora. Por lo que tenemos un coste total de programación de 8.880€.

Para la herramienta de programación, se ha elegido Visual Studio Code, que es de licencia gratuita, por lo que no se ha incrementado el precio. El resto de librerías utilizadas han sido de código abierto, por lo que no supone ningún coste adicional para el proyecto.

El programador necesitó un curso de formación que costó 300€, lo que supone un total de 9.180€.

El hardware utilizado ha sido un monitor, teclado, ratón y torre, lo que supone un precio estimado total de 600€. El hardware tiene una duración de 5 años (1825 días), por lo que el uso dado en el proyecto es de solo 111 días, lo que supone un coste de 36,5€.

Para poder poner el proyecto en producción, se necesita un servidor web dedicado, comparando precios y calidades se nos queda por 50€/mes, lo que supone 600€/año. Además, debemos tener en cuenta el dominio de gymfit.es, gymfit.com y gymfit.org, lo que supone un incremento de 10€/año por cada uno. Lo que nos haría un total de 630€.

Sumando todos los gastos anteriores tenemos un coste de 9.846,5€. Al trabajar desde casa, no tenemos que contar gastos por local o coworking, pero sí añadir los gastos de luz y agua, por lo que hay que incrementar un 10% el presupuesto para tener en cuenta esos gastos, lo que supone 984,65€. El total sería 10.831,15€.

EQUIPO HUMANO			
Servicio	**Tiempo**	**Horas**	**Coste**
Definición de la idea principal	6 días	24 horas	480€
Inicio del proyecto	5 días	20 horas	400€
Diseño de la aplicación	20 días	80 horas	1.600€
Desarrollo de la aplicación	76 días	304 horas	6.080 €
Documentación	4 días	16 horas	320 €
TOTAL	111 días	444 horas	**8.880€**

Tabla 3: Presupuesto equipo humano

EQUIPAMIENTO TÉCNICO			
Recurso	**Tipo de pago**	**Precio/unidad**	**Coste**
Servidor web dedicado	mensual	50€	600€
Dominios (.es, .com y .org)	anual	10€	30€
Hardware (pantalla, ratón, teclado, torre)	único	36,5€	36,5€
Cursos de formación	único	300€	300€
Costes indirectos (luz, agua, etc)	único	10% total	984,65€
TOTAL			**10.831,15€**

Tabla 4: Presupuesto equipamiento técnico

2. Contenidos

La aplicación web tiene una estructura típica de una página web, formada por cabecera, cuerpo y pie de página.

- **Cabecera**. Contiene el Logo de la aplicación web en la parte izquierda y los distintos apartados en la parte derecha. Dependiendo de si entramos como usuario como monitor los apartados pueden variar.
    - **Usuario**. El usuario tendrá los siguientes apartados:
        - **Perfil**.
        - **Rutinas**.
        - **Estadísticas**.
        - **Incidencias**.
    - **Monitor**. El monitor dispone de los siguientes apartados:
        - **Perfil**.
        - **Usuarios**.
        - **Rutinas**.
        - **Incidencias**.
- **Cuerpo**. Es donde se muestra toda la información de la aplicación. En algunos apartados puede que la pantalla esté dividida en dos zonas, una para visualizar los listados y otra para la interacción.
- **Pie de página**. Dispondrá de todos los enlaces necesarios para cumplir con las normativas vigentes, como puede ser el aviso legal, política de privacidad, condiciones de uso y cookies. Además, se visualizará el logo de la aplicación.

La estructura descrita anteriormente es la que se utilizará a lo largo de todas las pantallas de la aplicación, excepto para la pantalla de login y registro, la cual no dispondrá de cabecera y pie de página.

- **Inicio**. Página donde se mostrará el login para poder acceder a la aplicación. Si estamos en la ruta de usuarios se nos mostrará un link para ir a la página de registro y así poder darnos de alta en la aplicación, si estamos en la ruta de monitores solo se nos mostrará el login.
- **Perfil**. Página donde el usuario o monitor podrá editar su información personal.
- **Rutinas**. Página donde el usuario podrá ver la rutina que tiene asignada y marcar que empieza o termina de hacer la rutina. Si somos monitores veremos el listado de rutinas que tenemos creadas.
- **Usuarios**. Página que solo verán los monitores y que mostrará todos los usuarios que tiene asignados ese monitor, además podrá modificar la rutina que tiene asignada cada usuario.
- **Estadísticas**. Página que solo verán los usuarios y que mostrará distintas estadísticas.
- **Incidencias**. Página donde el usuario o monitor puede informar de algún problema de la aplicación.

3. Arquitectura de la aplicación

La arquitectura de la aplicación web es la de cliente/servidor, en la que el cliente se encargará de realizar peticiones *HTTP* para obtener los datos y el servidor será el encargado de suministrar esa información mediante una *API REST*. En la figura 3 se puede ver el esquema de la arquitectura cliente/servidor.

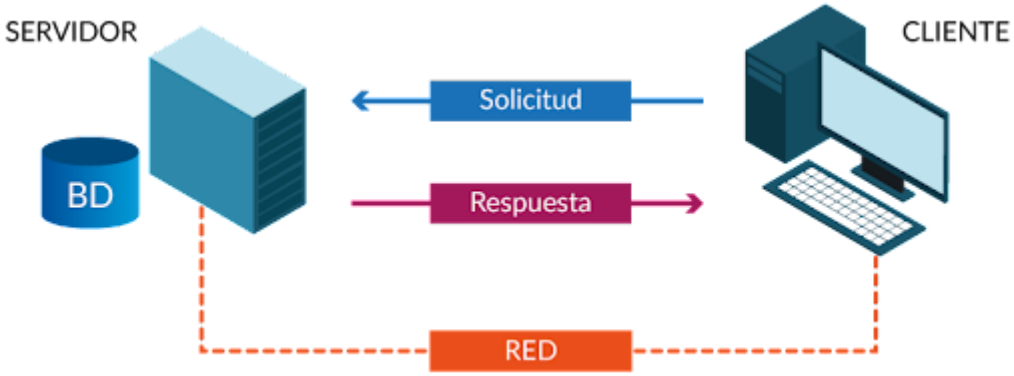


Figura 3: Arquitectura cliente/servidor

En la parte del cliente, es donde tendremos nuestra aplicación en Angular y es la parte que interactúa con el usuario, por lo que tendremos que tener un buen diseño visual para que le resulte agradable y cómodo al usuario. Usaremos el patrón arquitectónico Modelo Vista Controlador, que nos ayuda a desacoplar las clases de la aplicación mediante la implementación de interfaces gráficas de usuario. Las principales ventajas que nos aporta este patrón es que desacopla el modelo de datos de su visualización, por lo que podemos modificar la vista las veces que necesitemos sin tener que modificar el modelo.

En la parte del servidor, es donde tenemos la persistencia de datos, además de toda la lógica de nuestra aplicación. Esta parte el usuario no la ve, pero es muy importante ya que cualquier fallo hará que la web se caiga y el usuario no pueda acceder a ella. En la parte del servidor también tenemos la base de datos, a la cual nos conectaremos para almacenar la información de la aplicación.

3.1 Análisis de la aplicación

A continuación, se muestra el diagrama de los casos de uso del sistema, el cual resulta sumamente útil para analizar, elaborar y comprender el funcionamiento deseado de la aplicación. También nos permite ver, de forma visual, la relación entre los diferentes actores que interactúan con el sistema y los diferentes casos de uso.

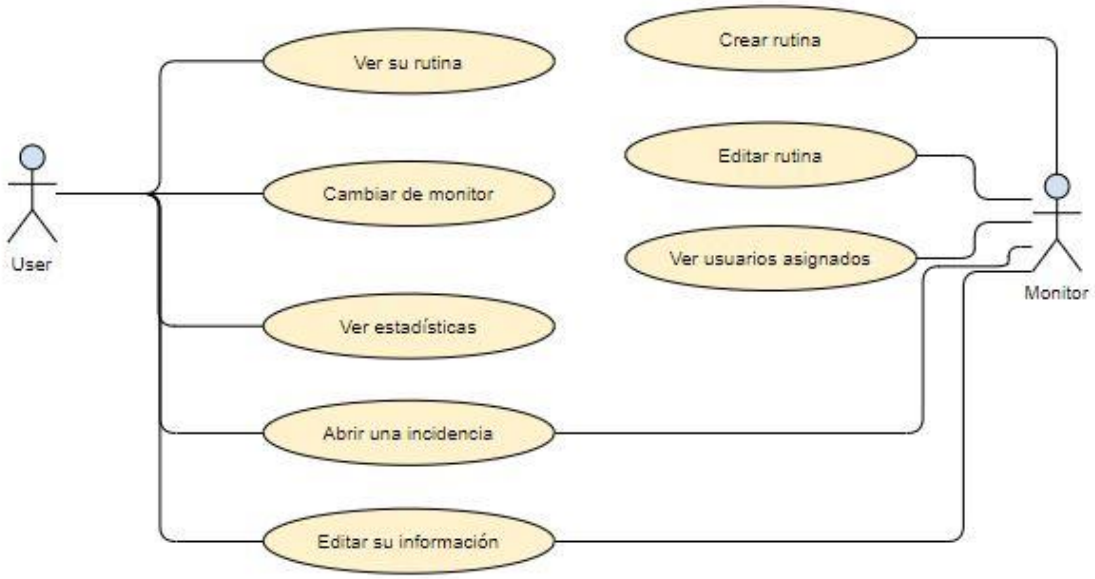


Figura 4: Diagrama de casos de uso de la aplicación

A continuación, se muestra la descripción de los casos de uso. Tanto al usuario como al monitor se les ha llamado usuario en la descripción de los casos de uso, ya que usuario hace referencia a quienes van a utilizar la aplicación, en nuestro caso el usuario y el monitor.

Caso de uso CU01 – Ver su rutina			
Identificador	CU01	**Fecha creación**	16/11/2021
Nombre	Rutinas	**Fecha revisión**	29/11/2021
Autor	Ignacio Vicent	**Fecha aprobación**	29/11/2021
Origen	Petición por parte del usuario	**Versión**	1.0
Descripción: En el panel, se creará una interfaz donde el usuario podrá visualizar su rutina. En ella podrá visualizar cada ejercicio.			
Secuencia de pasos: 1. El usuario accede al apartado Rutinas. 2. El usuario accede al tab “Mis rutinas”. 3. El sistema muestra la rutina con el listado de ejercicios del usuario.			
Excepciones: 1. El usuario no dispone de rutina. 2. El sistema muestra un listado de ejercicios vacío.			

Tabla 5: Caso de uso CU01 – Ver su rutina

Caso de uso CU02 – Cambiar de monitor			
Identificador	CU02	**Fecha creación**	16/11/2021
Nombre	Monitores	**Fecha revisión**	29/11/2021
Autor	Ignacio Vicent	**Fecha aprobación**	29/11/2021
Origen	Petición por parte del usuario	**Versión**	1.0
Descripción: En el panel, se creará una interfaz donde el usuario podrá visualizar su monitor. También podrá ver un listado con todos los monitores, por si desea cambiar de monitor.			
Secuencia de pasos: 1. El usuario accede al apartado Rutinas. 2. El usuario accede al tab “Monitores”. 3. El sistema muestra el monitor que tiene asignado el usuario y un listado de monitores. 4. El usuario selecciona el nuevo monitor.			

5. El usuario confirma que quiere cambiar de monitor.
Excepciones: 1. No hay monitores disponibles o dados de alta. 2. El sistema muestra un listado de monitores vacío.

Tabla 6: Caso de uso CU02 – Cambiar de monitor

Caso de uso CU03 – Ver estadísticas			
Identificador	CU03	**Fecha creación**	16/11/2021
Nombre	Estadísticas	**Fecha revisión**	29/11/2021
Autor	Ignacio Vicent	**Fecha aprobación**	29/11/2021
Origen	Petición por parte del usuario	**Versión**	1.0
Descripción: En el panel, se creará una interfaz donde el usuario podrá visualizar sus estadísticas.			
Secuencia de pasos: 1. El usuario accede al apartado Estadísticas. 2. El sistema muestra las estadísticas al usuario.			
Excepciones: 1. No hay datos para las estadísticas. 2. El sistema muestra un texto “No hay datos”.			

Tabla 7: Caso de uso CU03 – Ver estadísticas

Caso de uso CU04 – Abrir incidencia			
Identificador	CU04	**Fecha creación**	16/11/2021
Nombre	Incidencias	**Fecha revisión**	29/11/2021
Autor	Ignacio Vicent	**Fecha aprobación**	29/11/2021
Origen	Petición por parte del usuario	**Versión**	1.0
Descripción: En el panel, se creará una interfaz donde el usuario podrá rellenar un formulario con la información de la incidencia. Si falta algún campo necesario para el envío, se mostrará dicho error con un mensaje.			
Secuencia de pasos: 1. El usuario accede al apartado Incidencias. 2. El sistema muestra el formulario de abrir incidencia al usuario. 3. El usuario introduce los datos de la incidencia. 4. El sistema valida los datos introducidos. 5. El sistema informa que la incidencia se ha enviado correctamente. 6. El sistema limpia el formulario.			
Excepciones: 1. El sistema encuentra un error validando los datos introducidos. 2. El sistema muestra un texto en rojo con el error.			

Tabla 8: Caso de uso CU04 – Abrir incidencia

Caso de uso CU05 – Editar información			
Identificador	CU05	**Fecha creación**	16/11/2021
Nombre	Perfil	**Fecha revisión**	29/11/2021
Autor	Ignacio Vicent	**Fecha aprobación**	29/11/2021
Origen	Petición por parte del usuario	**Versión**	1.0
Descripción: En el panel, se creará una interfaz donde el usuario podrá editar un formulario con su información. Si falta algún campo necesario para el envío, se mostrará dicho error con un mensaje.			
Secuencia de pasos: 1. El usuario accede al apartado Perfil. 2. El sistema muestra el formulario con los datos del usuario. 3. El usuario modifica los datos. 4. El sistema valida los datos introducidos. 5. El sistema informa que los cambios se han guardado correctamente.			
Excepciones: 1. El sistema encuentra un error validando los datos introducidos. 2. El sistema muestra un texto en rojo con el error.			

Tabla 9: Caso de uso CU05 – Editar información

Caso de uso CU06 – Crear rutina			
Identificador	CU06	**Fecha creación**	16/11/2021
Nombre	Rutinas	**Fecha revisión**	29/11/2021
Autor	Ignacio Vicent	**Fecha aprobación**	29/11/2021
Origen	Petición por parte del usuario	**Versión**	1.0
Descripción: En el panel, se creará una interfaz donde el usuario podrá crear una rutina añadiendo los ejercicios.			
Secuencia de pasos: 1. El usuario accede al apartado Rutinas. 2. El sistema muestra un listado con las rutinas creadas y un botón para crear una. 3. El usuario pincha en el botón “Crear rutina”. 4. El sistema muestra un listado vacío con los ejercicios seleccionados, un listado con los ejercicios disponibles y un botón de añadir. 5. El usuario selecciona el ejercicio del listado de ejercicios disponibles y pincha en el botón de añadir. 6. El sistema añade el ejercicio al listado de ejercicios seleccionados.			
Excepciones: 1. El sistema no dispone de ejercicios. 2. El sistema muestra un listado vacío de ejercicios.			

Tabla 10: Caso de uso CU06 – Crear rutina

Caso de uso CU07 – Editar rutina			
Identificador	CU07	**Fecha creación**	16/11/2021
Nombre	Rutinas	**Fecha revisión**	29/11/2021
Autor	Ignacio Vicent	**Fecha aprobación**	29/11/2021
Origen	Petición por parte del usuario	**Versión**	1.0
Descripción: En el panel, se creará una interfaz donde el usuario podrá editar una rutina añadiendo o eliminado los ejercicios.			
Secuencia de pasos: 1. El usuario accede al apartado Rutinas. 2. El sistema muestra un listado con las rutinas creadas y un botón para crear una. 3. El usuario selecciona la rutina a editar. 4. El sistema muestra un listado con los ejercicios seleccionados, un listado con los ejercicios disponibles y un botón de añadir. 5. El usuario selecciona el ejercicio del listado de ejercicios seleccionados y pincha en el botón de eliminar. 6. El sistema elimina el ejercicio del listado de ejercicios seleccionados. 7. El usuario selecciona el ejercicio del listado de ejercicios disponibles y pincha en el botón de añadir. 8. El sistema añade el ejercicio al listado de ejercicios seleccionados.			
Excepciones: 1. El sistema no dispone de ejercicios. 2. El sistema muestra un listado vacío de ejercicios.			

Tabla 11: Caso de uso CU07 – Editar rutina

Caso de uso CU08 – Ver usuarios asignados			
Identificador	CU08	**Fecha creación**	16/11/2021
Nombre	Usuarios	**Fecha revisión**	29/11/2021
Autor	Ignacio Vicent	**Fecha aprobación**	29/11/2021
Origen	Petición por parte del usuario	**Versión**	1.0
Descripción: En el panel, se creará una interfaz donde el usuario podrá visualizar todos los usuarios que tiene asignados.			
Secuencia de pasos: 1. El usuario accede al apartado Usuarios. 2. El sistema muestra un listado con los usuarios asignados.			
Excepciones: 1. El usuario no tiene usuarios asignados.			

2. El sistema muestra un listado vacío de usuarios.

Tabla 12: Caso de uso CU08 – Ver usuarios asignados

3.2 Diagrama de navegación

En la figura 4 podemos observar las páginas por las que puede navegar el cliente de nuestra aplicación web. Los distintos colores representan los niveles de la aplicación, para llegar al nivel de más abajo, como por ejemplo a “Mis rutinas”, debemos pasar por los distintos colores/niveles, en este caso por el “Login” y luego “Rutinas”.

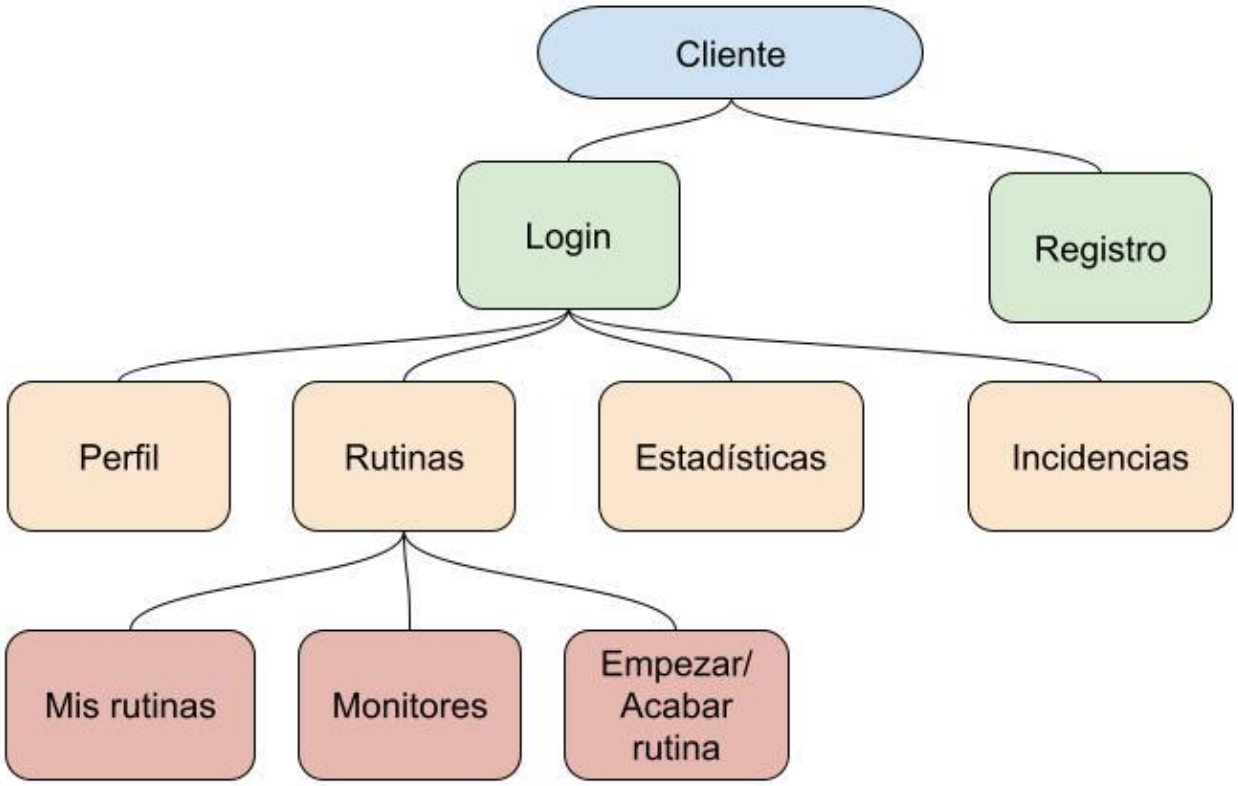


Figura 5: Mapa de navegación del cliente

En primer lugar, nos encontramos con el inicio de la aplicación web, el cual está compuesto por el “Login” y “Registro” que nos permitirá acceder a la aplicación. Si ya disponemos de usuario y contraseña usaremos el “Login” para acceder, en caso contrario podemos realizar el “Registro” en la aplicación para obtener los datos.

Una vez logueados en la aplicación, podremos ver nuestros datos en el apartado “Perfil”. Desde ahí podremos visualizar y modificar nuestros datos, además de poder cambiar la contraseña de acceso.

También podremos ver las rutinas que tenemos activas, para ello tenemos la pestaña “Rutinas” que tendrá tres apartados, el primero “Mis rutinas” nos servirá para poder ver nuestras rutinas activas, el segundo “Monitores” nos servirá para ver que monitor tenemos y poder cambiar de monitor en caso de que no nos gusten las rutinas que nos crea y el tercero “Empezar/Acabar rutina” que sirve para controlar el tiempo que has tardado en realizar la rutina. Además, disponemos de un apartado de “Estadísticas” donde se puede ver

el tiempo medio de entrenamiento o qué días son los que más vas al gimnasio. Finalmente, tenemos el apartado de “Incidencias” donde podemos reportar cualquier problema que tengamos con la aplicación.

En la figura 5 podemos observar las páginas por las que puede navegar el monitor de nuestra aplicación web. Los distintos colores representan los niveles de la aplicación, para llegar al nivel de más abajo, como por ejemplo a “Asignar rutina”, debemos pasar por los distintos colores/niveles, en este caso por el “Login” y luego por “Usuarios”.

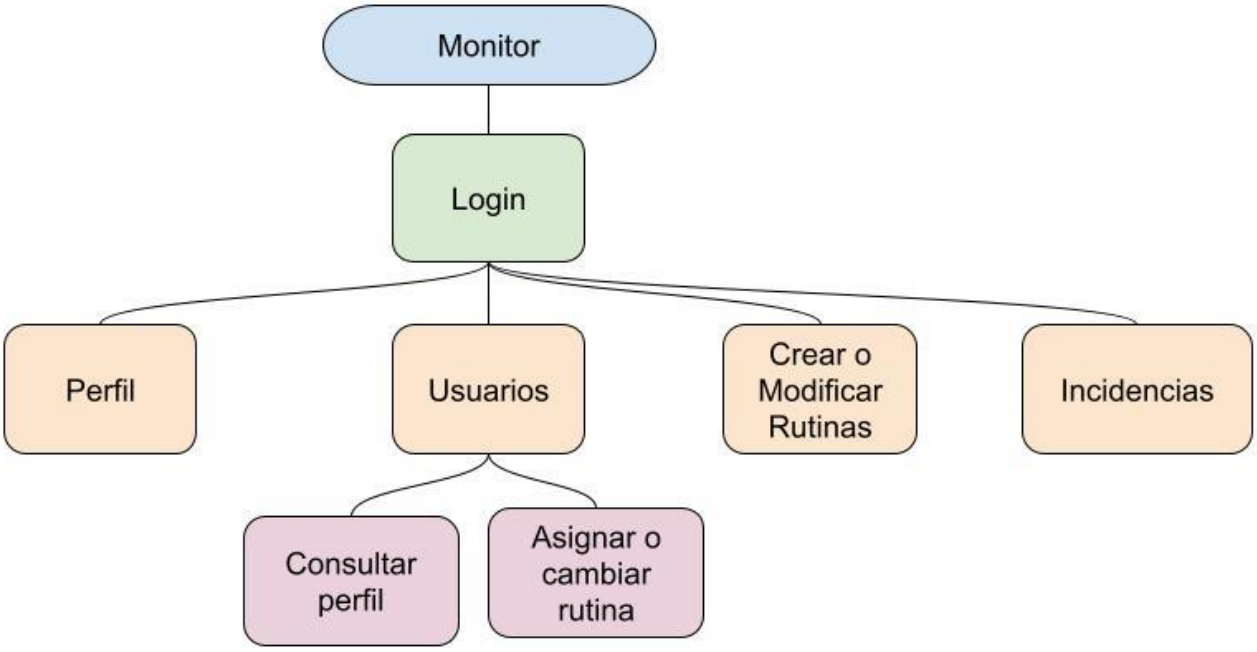


Figura 6: Mapa de navegación del monitor

En primer lugar, nos encontramos con el inicio de la aplicación web, el cual está compuesto por el “Login” que nos permitirá acceder a la aplicación. Una vez logueados en la aplicación podemos acceder a nuestro “Perfil”, donde podemos visualizar y modificar nuestros datos, además de cambiar la contraseña. También tenemos el apartado “Usuarios”, donde podemos ver los usuarios que tenemos asignados, para cada usuario podemos consultar sus datos públicos (sin contraseña) y asignar o cambiar la rutina. En el apartado “Crear o Modificar Rutinas” podemos crear rutinas nuevas o modificar las existentes, para luego asignarlas a los distintos usuarios. Finalmente, tenemos el apartado “Incidencias” para poder indicar los fallos que se encuentren en la aplicación.

3.3 Diseño de la base de datos

Para almacenar la información de la aplicación web utilizaremos una base de datos relacional *SQLite*. Pero no haremos consultas *SQL* directamente para modificar los datos, utilizaremos un *ORM* llamado Sequelize, que nos permitirá cambiar de base de datos sin tener que reescribir las consultas, además de acelerar el tiempo de desarrollo.

En la figura 6 podemos ver como ha quedado el modelo de datos de nuestra aplicación web y en la tabla 5 una descripción de cada entidad/tabla de la base de datos.

Tabla	Descripción
User	Almacena la información del usuario. La clave primaria es el id, la contraseña se guarda cifrada y el “idWorkout” hace referencia a la tabla Workout.
Monitor	Almacena la información del monitor. La clave primaria es el id y la contraseña se guarda cifrada.
HistoryUserMonitor	Almacena el historial de monitores que ha tenido un usuario, ya que puede tener como máximo 3. El campo “dateEnd” admite nulos, así si es nulo podemos saber que aún sigue con ese monitor.
Workout	Almacena la información de la rutina y el monitor que la ha creado.
Exercise	Almacena la información del ejercicio.
WorkoutExercises	Almacena todos los ejercicios que componen una rutina.
UserWorkoutHistory	Almacena las veces que un usuario empieza o termina una rutina. Nos sirve para sacar estadísticas sobre lo que tarda en realizar las rutinas. Está relacionado con la rutina por “idWorkout”.
Incidence	Almacena las incidencias que reportan los clientes y monitores. Tiene una relación opcional con “idUser”y otra relación opcional con “idMonitor”.

Tabla 13: Entidades/tablas de la base de datos

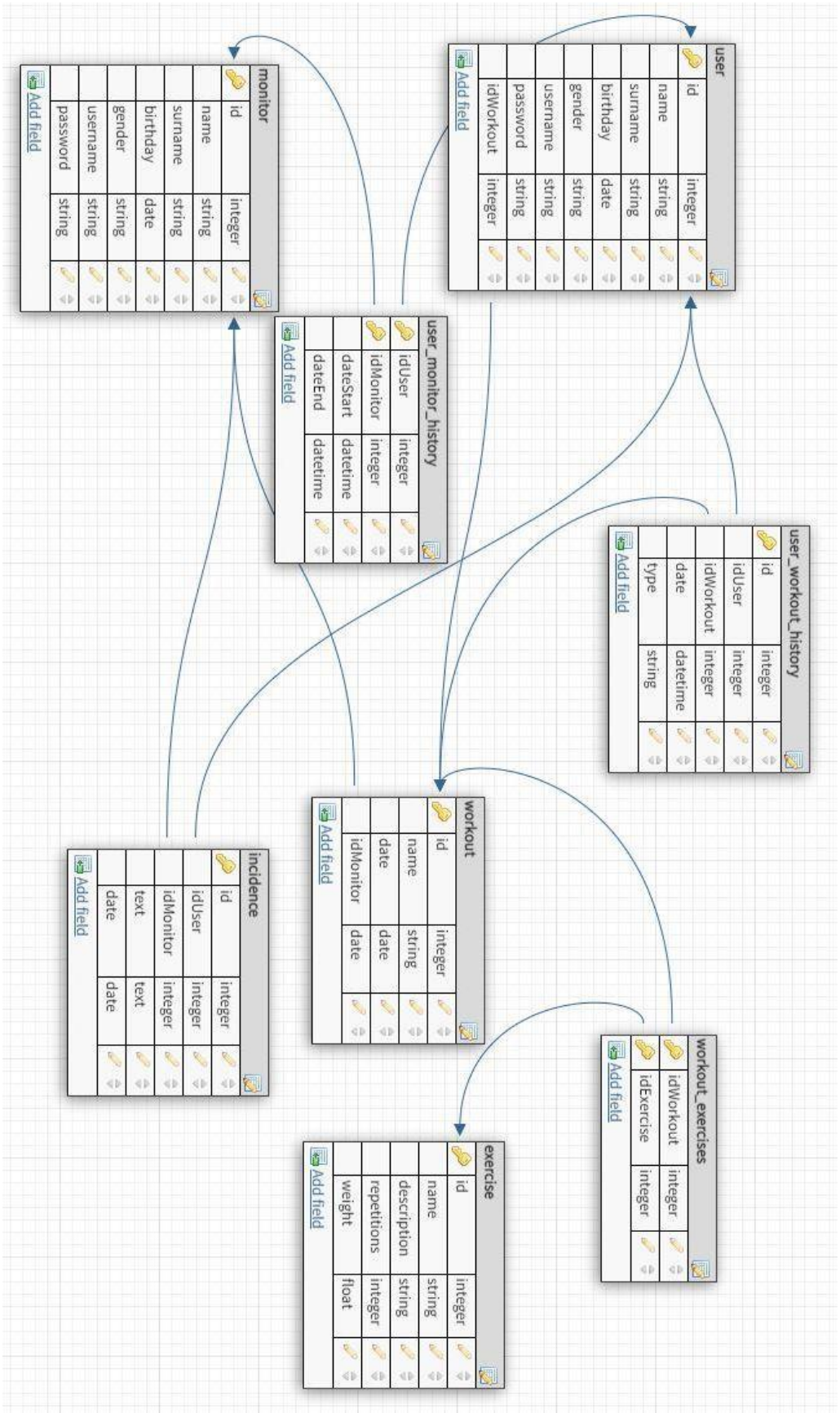


Figura 7: Mapa de navegación del monitor

3.4 Desarrollo Frontend

Para el desarrollo de la parte Front, utilizaremos Angular que es un framework para aplicaciones web desarrollado en TypeScript, por lo que podremos tener un tipado de datos y evitarnos errores a la hora de lanzar nuestra aplicación. Además, es de código abierto y mantenido por Google, por lo que es un framework con mucho soporte y comunidad detrás.

3.5 Desarrollo Backend

Para el desarrollo de la parte Back, utilizaremos Node.js con Express, que es un framework web transigente, escrito en Javascript y que necesita de Node.js para funcionar. Express nos proporciona los mecanismos necesarios para poder escribir manejadores de peticiones *HTTP* en diferentes rutas y la capacidad de añadir middlewares en dichas peticiones.

4. Plataforma de desarrollo

Para la implementación de este Trabajo Fin de Máster, vamos a utilizar los siguientes recursos tecnológicos de Software:

- **Visual Studio Code**. Como IDE de desarrollo de nuestra aplicación web.
- **Balsamiq**. Para la creación de wireframes.
- **SQLite**. Para el almacenamiento de datos de la aplicación web.
- **Angular**. Para el desarrollo de la parte frontal.
- **Node.js con Express**. Para el desarrollo de la parte back.
- **Microsoft Word**. Para la maquetación de la documentación.
- **Chrome**. Como navegador de testeo.

Para la parte Hardware se han utilizado los siguientes recursos:

- **Ordenador** i5 con 8GB de RAM.
- **Pantalla** de 24 pulgadas Lenovo.

Además, también se han utilizado distintas web-apps como recursos:

- **GitHub**. Para el control de versiones tanto de la parte frontal como de la parte back.
- **Google Drive**. Como respaldo de la documentación.

5. Prototipos

Antes del desarrollo de la aplicación, hemos desarrollado los prototipos Lo-Fi, también conocidos como wireframes, y los prototipos Hi-Fi, también conocidos como mockups. Gracias al diseño de estos prototipos conseguimos validar la aplicación y encontrar posibles problemas de usabilidad y accesibilidad, para así tener una mejor experiencia de usuario. Además, al disponer de tiempo limitado, nos vienen muy bien los prototipos para encontrar cualquier error y solucionarlo de forma más rápida, ya que si se encontrara con la aplicación desarrollada el cambio supondría un esfuerzo mucho mayor.

5.1 Lo-Fi

Los prototipos Lo-Fi son prototipos de baja fidelidad, donde se intenta realizar un primer boceto de la forma más simple posible. Estos pueden estar hechos en papel, pero para este proyecto los hemos realizado con una herramienta software. Con estos prototipos estamos validando la funcionalidad de la aplicación, analizando lo fácil o difícil que es para el usuario realizar tareas simples. A continuación, se muestran los distintos prototipos Lo-Fi que se han diseñado para la aplicación:

USUARIO

Primero, mostraremos los wireframes de la aplicación web del usuario, que es el que utilizará nuestra aplicación web cada día.

Pantalla de Login

La pantalla de login consiste en la solicitud de un usuario y contraseña, también se da la posibilidad de registrarse si no se dispone de cuenta. La estructura de esta pantalla no coincide con la del resto de la aplicación, ya que no tiene header y footer.

Figura 8: Wireframe login usuario

Pantalla de Registro

La pantalla de registro consiste en un formulario donde solicitamos todos los datos al usuario para que pueda empezar a utilizar la aplicación.

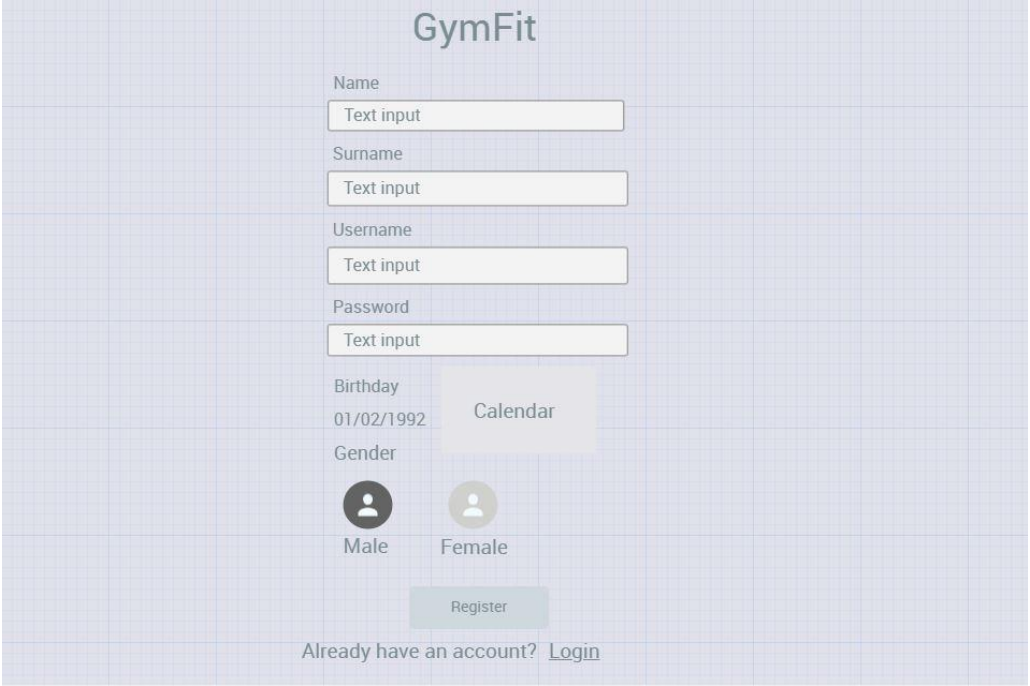


Figura 9: Wireframe registro usuario

Pantalla Estadísticas

Se puede acceder mediante la opción “Estadísticas” del header y como su nombre indica, nos mostrará las estadísticas disponibles para ese usuario.

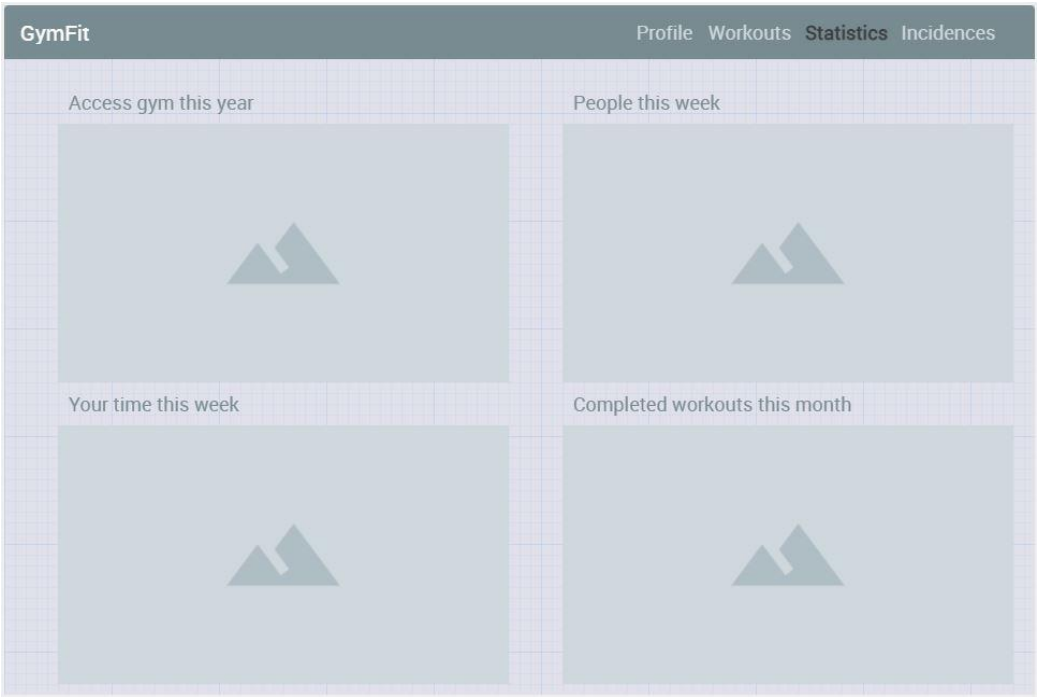


Figura 10: Wireframe estadísticas usuario

Pantalla Rutinas

La pantalla de rutinas nos muestra la rutina que tiene el usuario, la cual puede empezar o parar. Dicha rutina es la que le crea el monitor en base a su información, en caso de no gustarle la pueda cambiar un máximo de 3 veces en la subpestaña “monitors”.

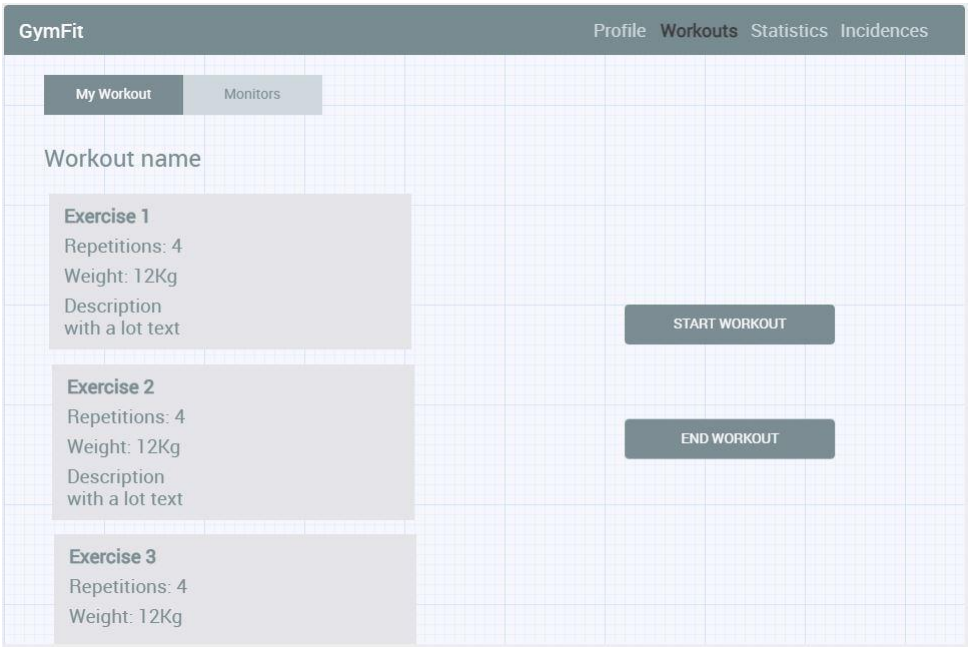


Figura 11: Wireframe rutina del usuario

Pantalla visualizar/cambiar monitor

Como su nombre indica, esta pantalla nos mostrará la información del monitor que tenemos seleccionado actualmente y nos mostrará un listado de monitores disponibles por si queremos cambiar de monitor. Solo se puede cambiar un máximo de tres veces de monitor, por lo que, si superamos el número de cambios, el back nos avisará con un código de error.

Figura 12: Wireframe visualizar/cambio monitor

Pantalla Perfil

La pantalla de perfil está compuesta por un formulario de edición, el cual estará rellenado con los datos que el usuario introdujo en el registro. El usuario puede modificar los datos necesarios y guardarlos.

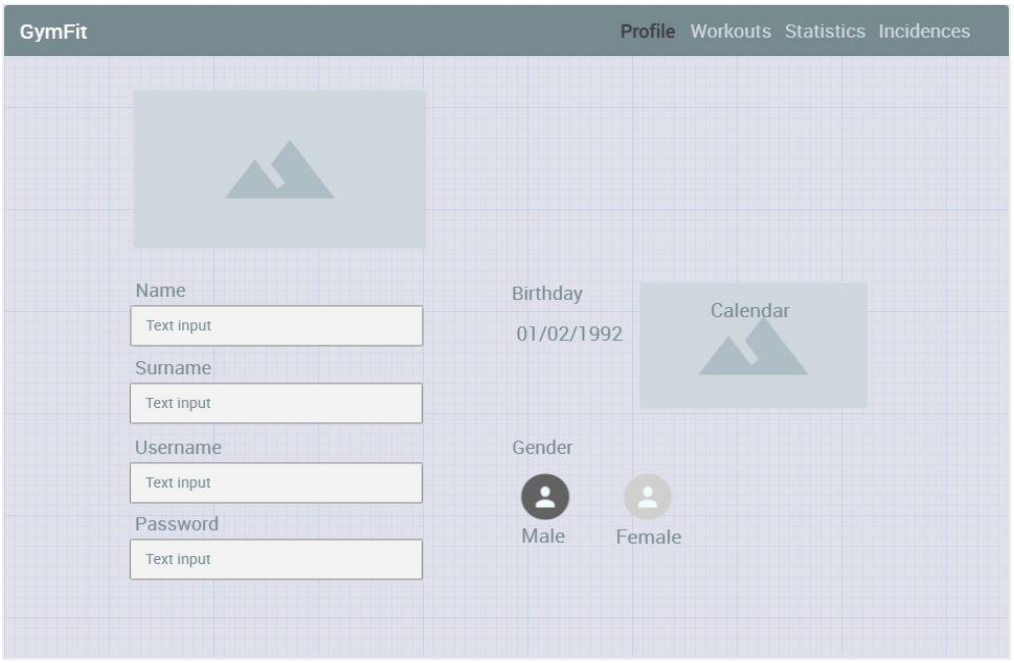


Figura 13: Wireframe perfil usuario

Pantalla Incidencias

Es la pantalla desde donde el usuario puede reportar los problemas que se encuentre tanto al utilizar la aplicación como en el propio gimnasio.

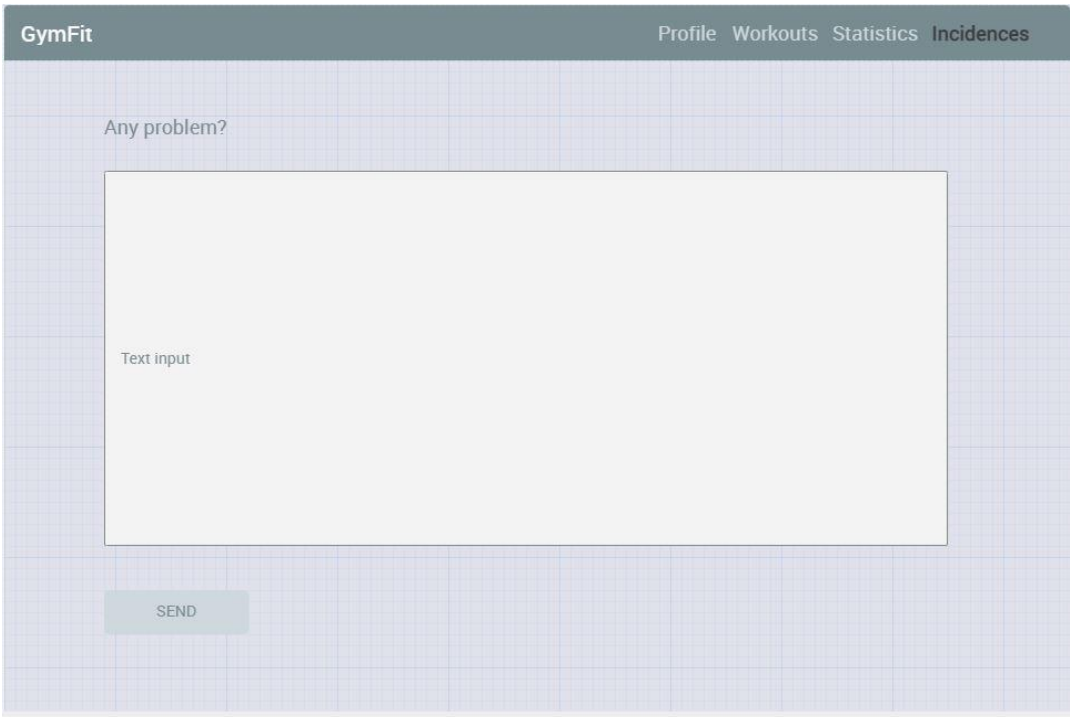


Figura 14: Wireframe enviar incidencia usuario

MONITOR

A continuación, se mostrarán los wireframes relacionados con la aplicación web del monitor, a diferencia de los usuarios, los monitores no tienen un registro, por lo que solo tendrán pantalla de “Login”. Hay algunas pantallas que no se mostrarán, como puede ser la de Incidencias, debido a que es idéntica a la que tienen los usuarios.

Pantalla Login

La pantalla de login consiste en la solicitud de un usuario y contraseña. La estructura de esta pantalla no coincide con la del resto de la aplicación, ya que no tiene header y footer.

Figura 15: Wireframe login monitor

Pantalla Perfil

La pantalla de perfil está compuesta por un formulario de edición, el cual estará rellenado con los datos que el administrador introdujo al registrar el monitor. El monitor puede comprobar y modificar los datos necesarios, una vez estén correctos o revisados el monitor puede guardarlos.

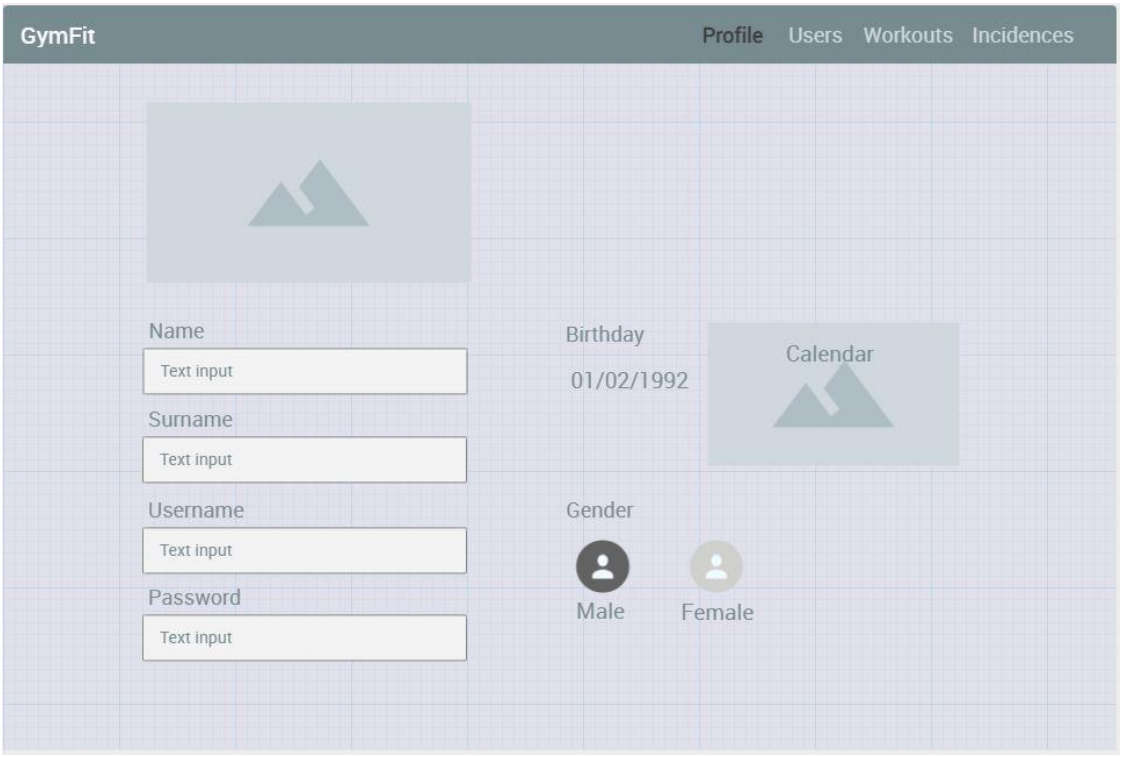


Figura 16: Wireframe perfil monitor

Pantalla Usuarios

La pantalla de usuarios muestra los usuarios que tiene asignados el monitor, desde la cual puede visualizar los datos de cada uno y modificar la rutina que tiene asignada.

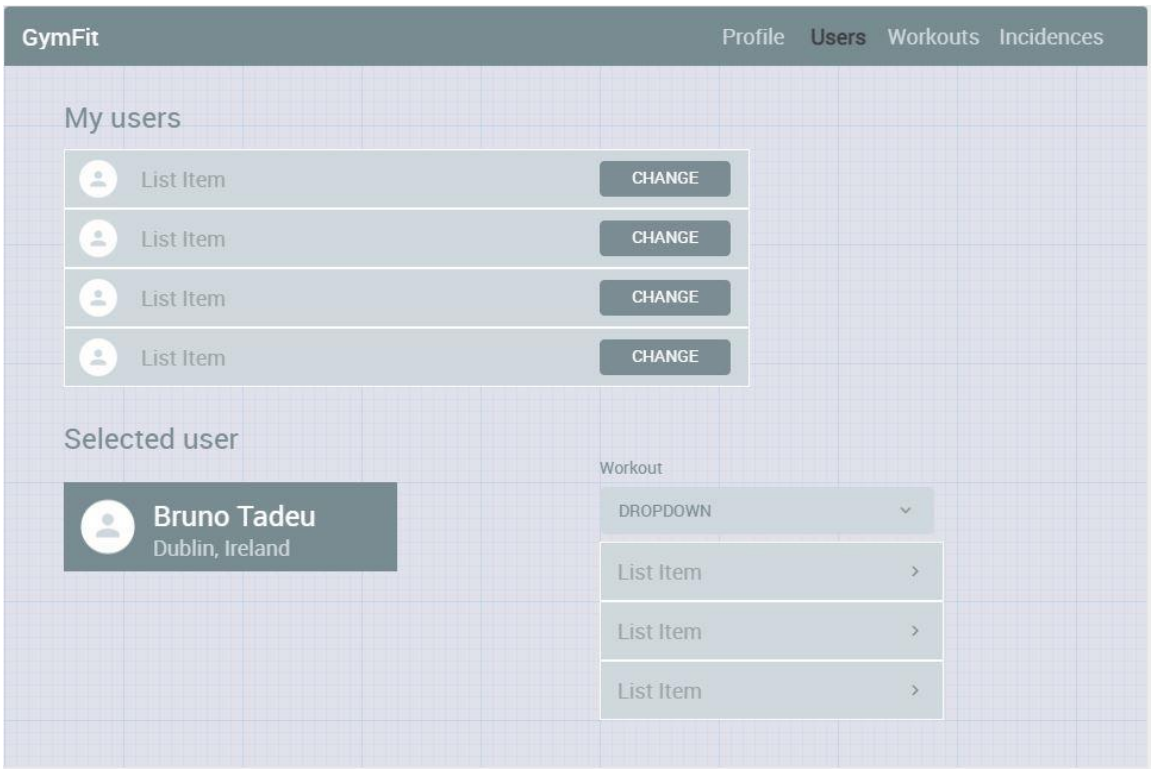


Figura 17: Wireframe usuarios asignados al monitor

Pantalla Rutinas

La pantalla rutinas muestra las rutinas que tiene creadas el monitor, las cuales puede editar en caso de que sea necesario.

Figura 18: Wireframe rutinas creadas por el monitor

Pantalla añadir/editar rutina

Como su nombre indica, es la pantalla que utilizaremos para crear o editar una rutina. En la cual podemos ir seleccionando los ejercicios que se pueden realizar en el gimnasio y que el administrador habrá introducido previamente.

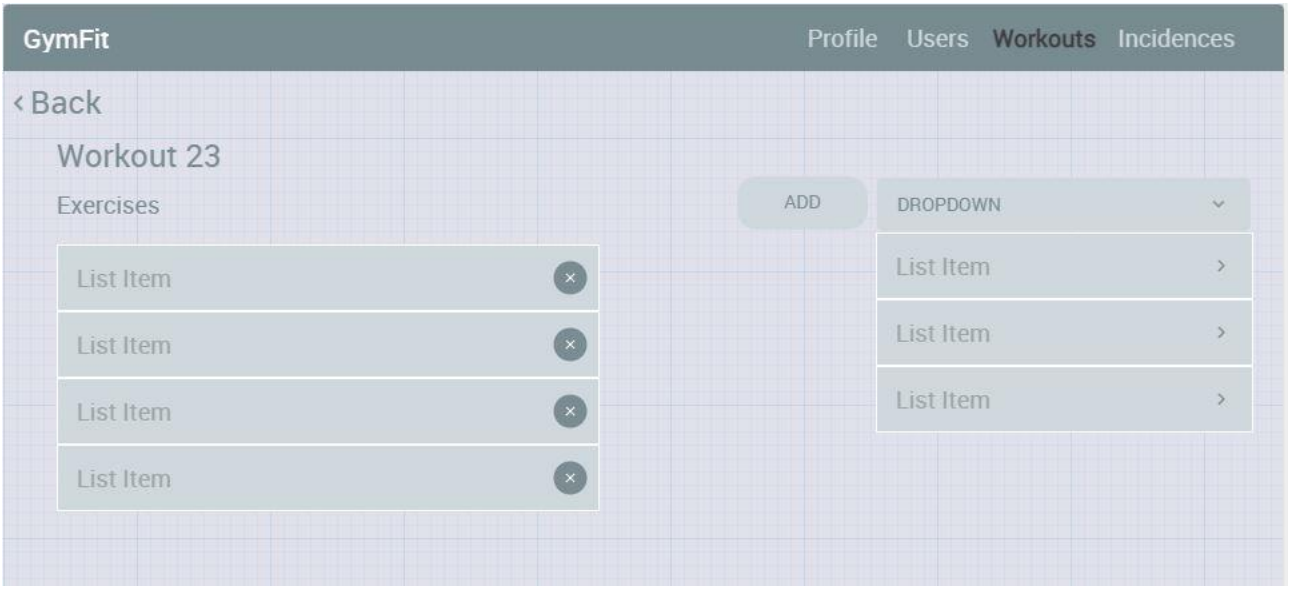


Figura 19: Wireframe añadir/editar rutina

5.2 Hi-Fi

Los prototipos Hi-Fi son prototipos de alta fidelidad, donde se intenta realizar un boceto lo más real de cómo será la aplicación. Al ser el boceto final debe contener los logos, la paleta de colores a utilizar en la aplicación y la tipografía, ya que así se puede comprobar que todo queda correctamente y no hay contrastes que puedan despistar al usuario. Si detectamos fallos o necesitamos mejorar el diseño debemos hacerlo, ya que a pesar de ser más costosos de solucionar que si los hubiéramos detectado en el boceto de baja fidelidad, será menos costoso que si los detectamos una vez esté programada la aplicación. A continuación, se muestran los distintos prototipos Hi-Fi que se han diseñado para la aplicación:

USUARIO

Primero mostraremos los mockups de la aplicación web del usuario, que es el que utilizará nuestra aplicación web cada día.

Pantalla de Login

Se ha mantenido la misma estructura que en el wireframe y se ha añadido un fondo diferente al resto de la aplicación, pero siguiendo con el estilo definido.

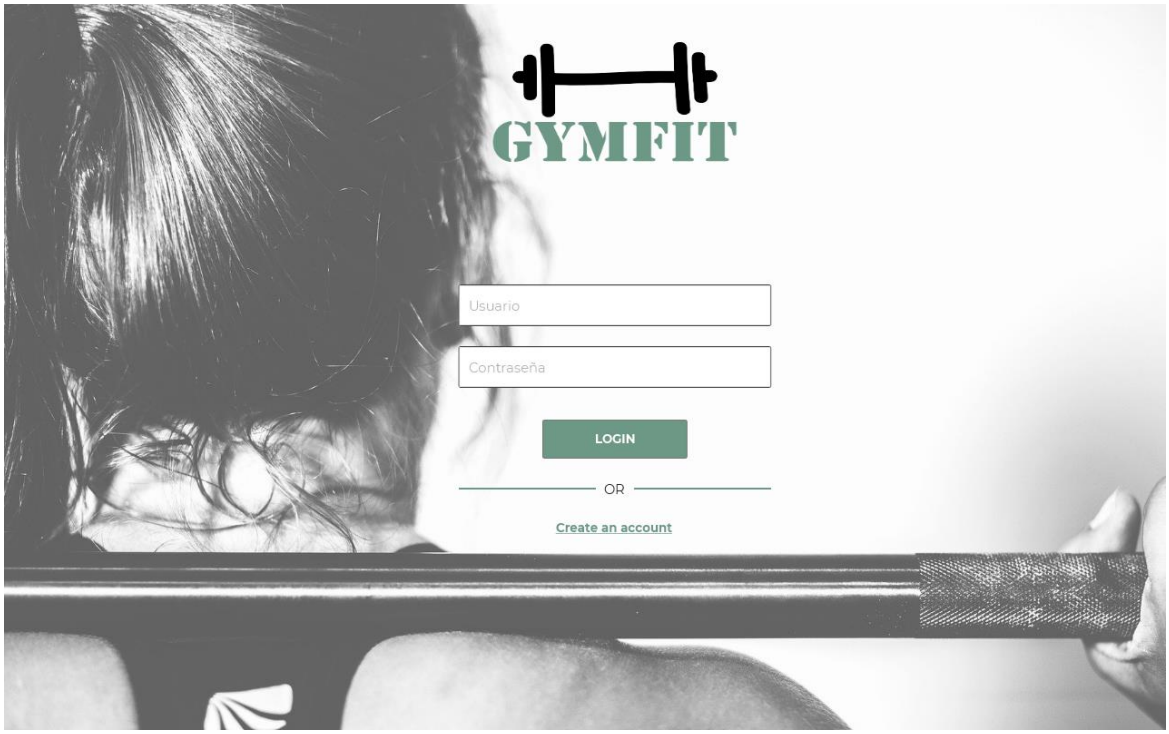


Figura 20: Mockup login usuario

Pantalla de Registro

Se ha mantenido la misma estructura que en el wireframe, se ha añadido el mismo fondo que en el “Login”. Se ha creado una tarjeta con transparencia para que resalten los campos a rellenar.

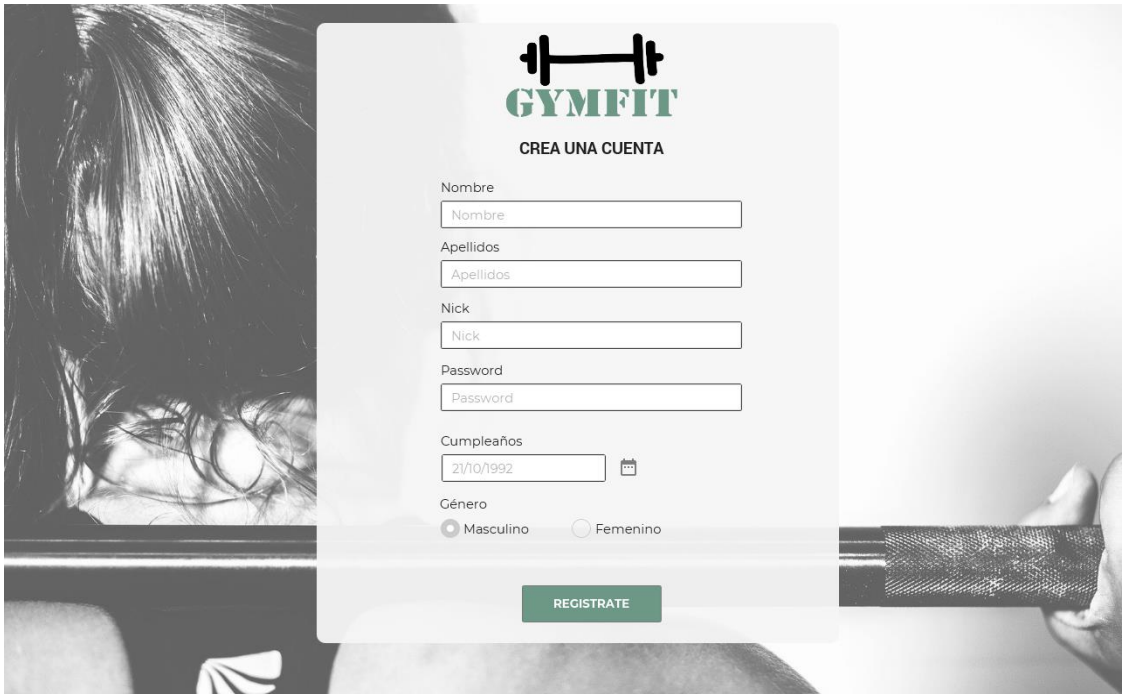


Figura 21: Mockup registro usuario

Pantalla Estadísticas

Se ha mantenido la misma estructura que en el wireframe, se ha añadido un fondo que sigue el estilo de la aplicación.

Figura 22: Mockup estadísticas usuario

Pantalla Rutinas

Se ha mantenido la misma estructura que en el wireframe y se ha añadido un scroll para poder visualizar todos los ejercicios de la rutina.

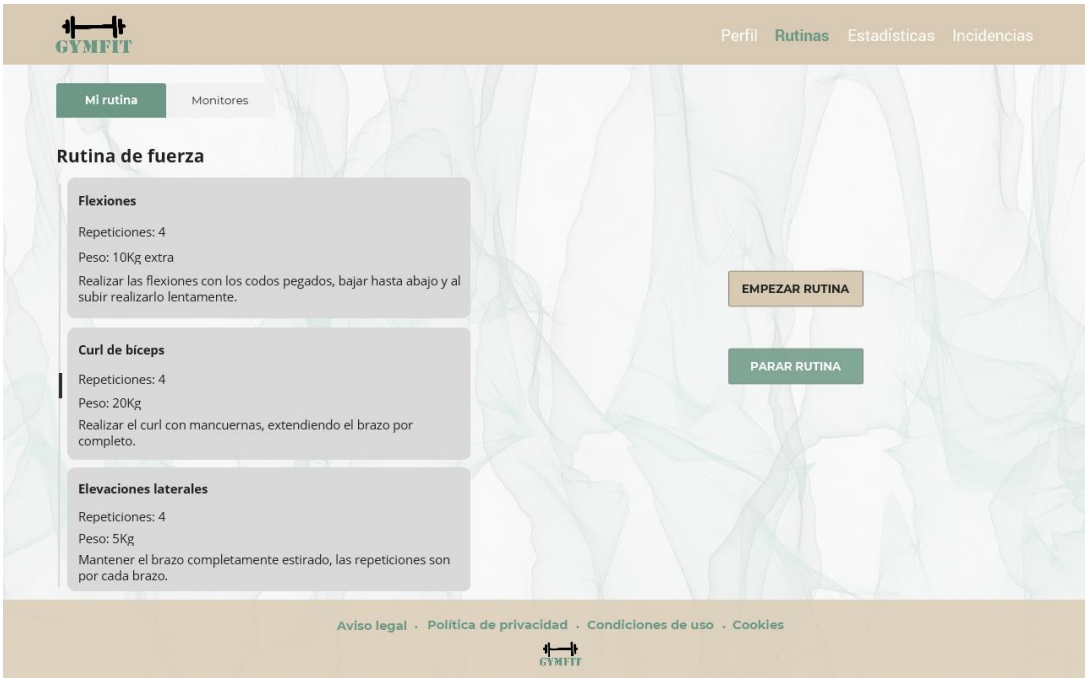


Figura 23: Mockup rutina del usuario

Pantalla Perfil

Se ha cambiado ligeramente la estructura del wireframe, se ha centrado la foto en el medio de la pantalla y añadido el botón de guardar.

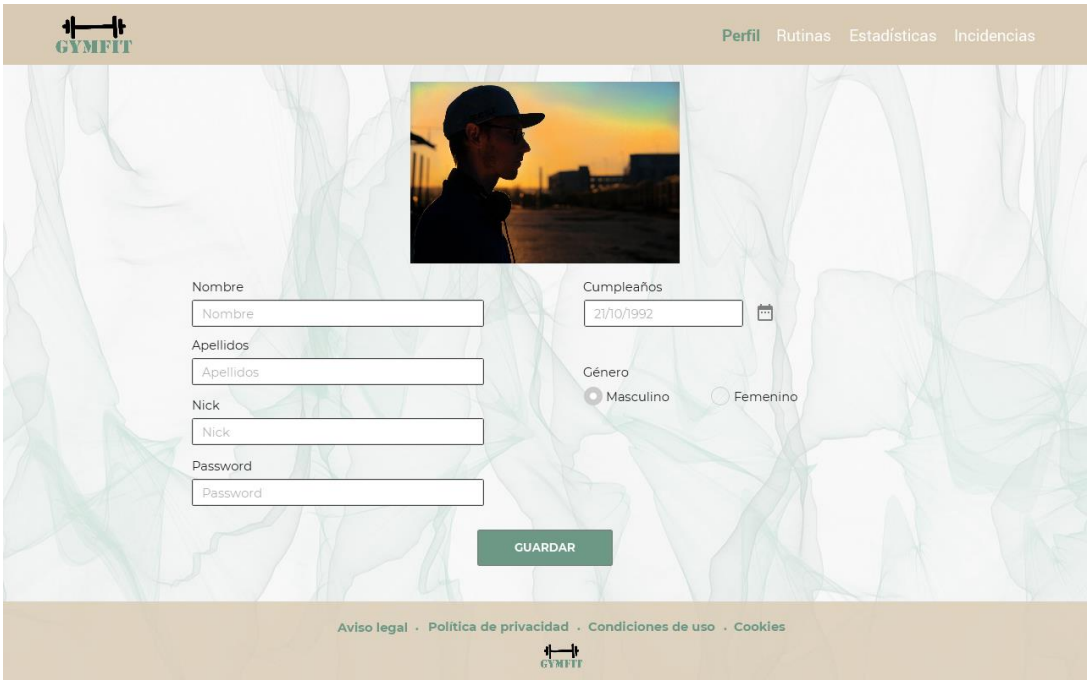


Figura 24: Mockup perfil usuario

Pantalla Incidencias

Se ha mantenido la misma estructura que en el wireframe pero con ciertas mejoras. Se ha añadido un fondo representativo que mantiene el diseño de la aplicación, además se ha creado una tarjeta con transparencia para resaltar el texto y se ha centrado para una mayor visibilidad.

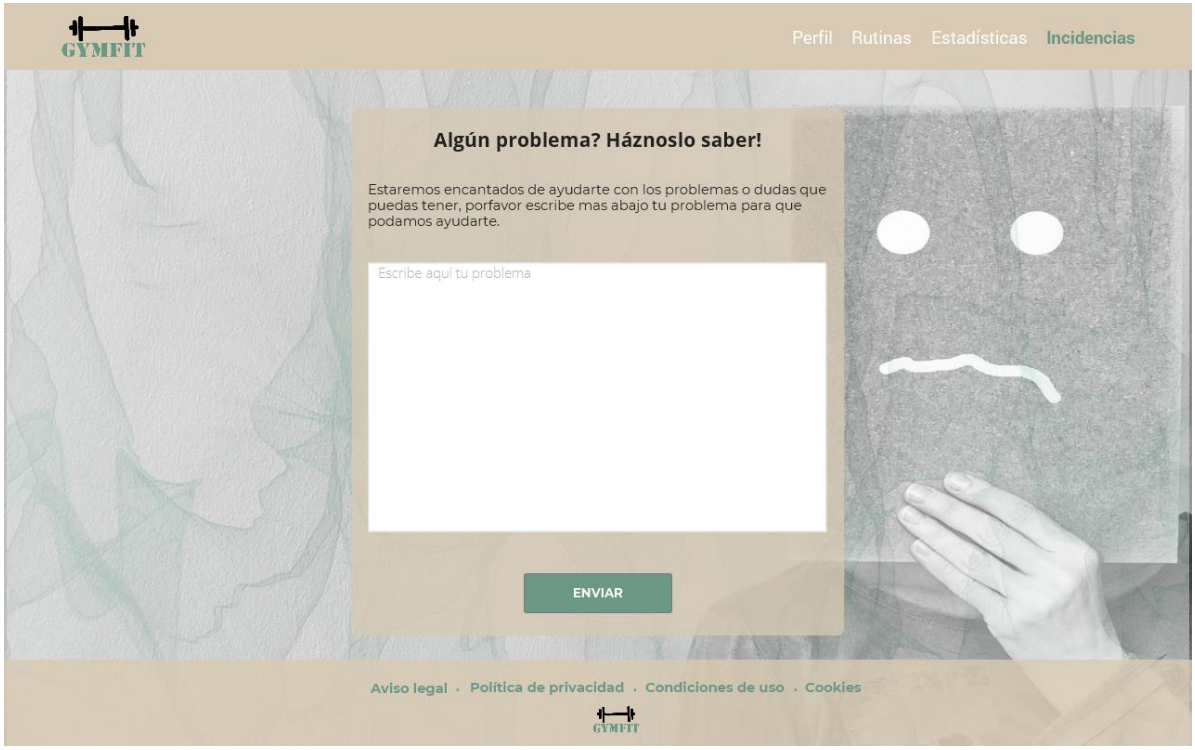


Figura 25: Mockup enviar incidencia usuario

MONITOR

A continuación, se mostrarán los mockups relacionados con la aplicación web del monitor. Hay algunas pantallas que no se mostrarán, como puede ser la de Incidencias, debido a que es idéntica a la que tienen los usuarios.

Pantalla Login

Se ha mantenido la misma estructura que en el wireframe y se ha añadido un fondo diferente al resto de la aplicación, pero siguiendo con el estilo definido. Comparte el mismo fondo que el usuario, pero no tiene la opción de registrarse.

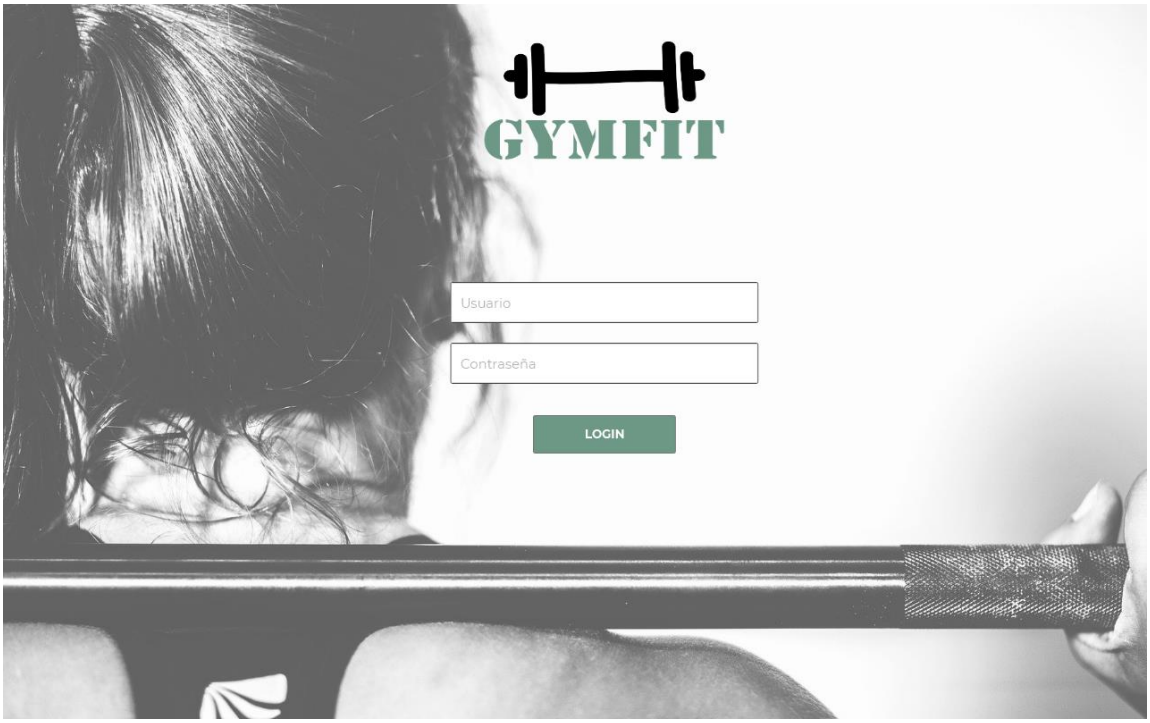


Figura 26: Mockup login monitor

Pantalla Perfil

Se ha cambiado ligeramente la estructura del wireframe, se ha centrado la foto en el medio de la pantalla y añadido el botón de guardar.

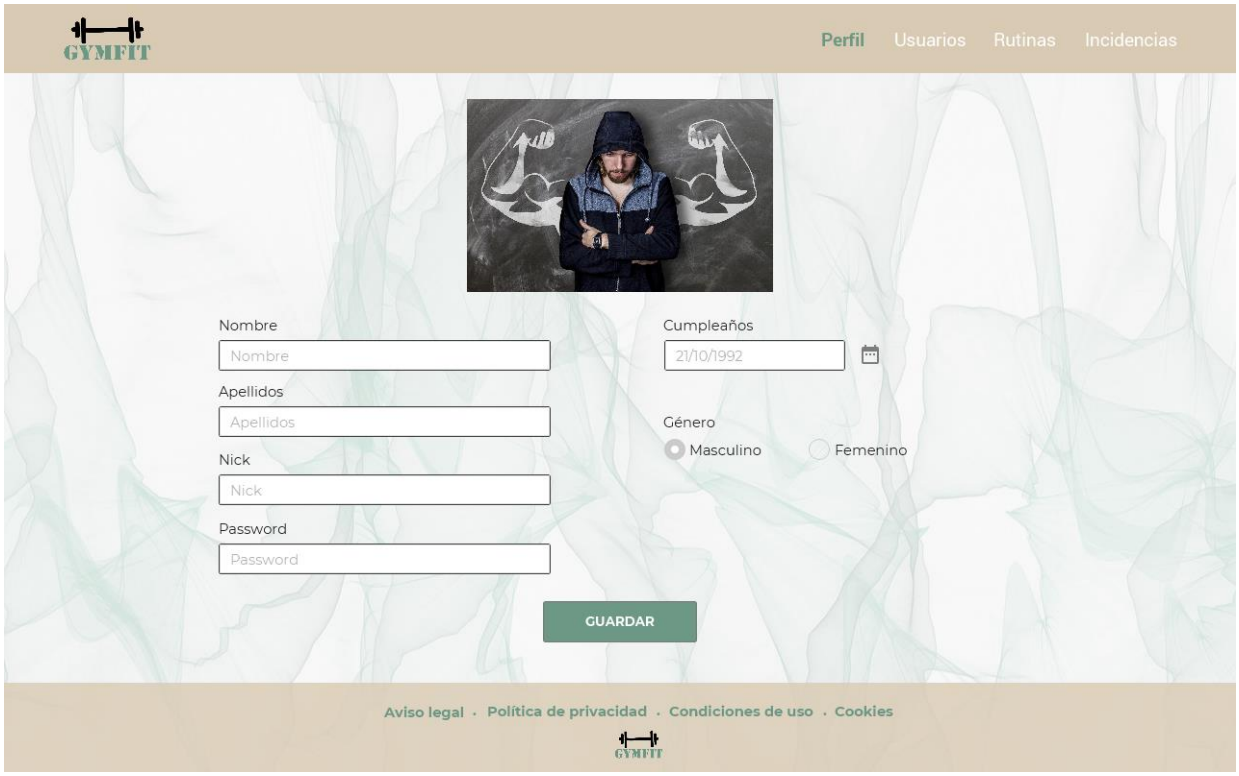


Figura 27: Mockup perfil monitor

Pantalla Usuarios

Respecto al wireframe se ha cambiado la ubicación del usuario seleccionado, pasando a ocupar la parte derecha de la aplicación y dejando más espacio para mostrar a todos los usuarios.

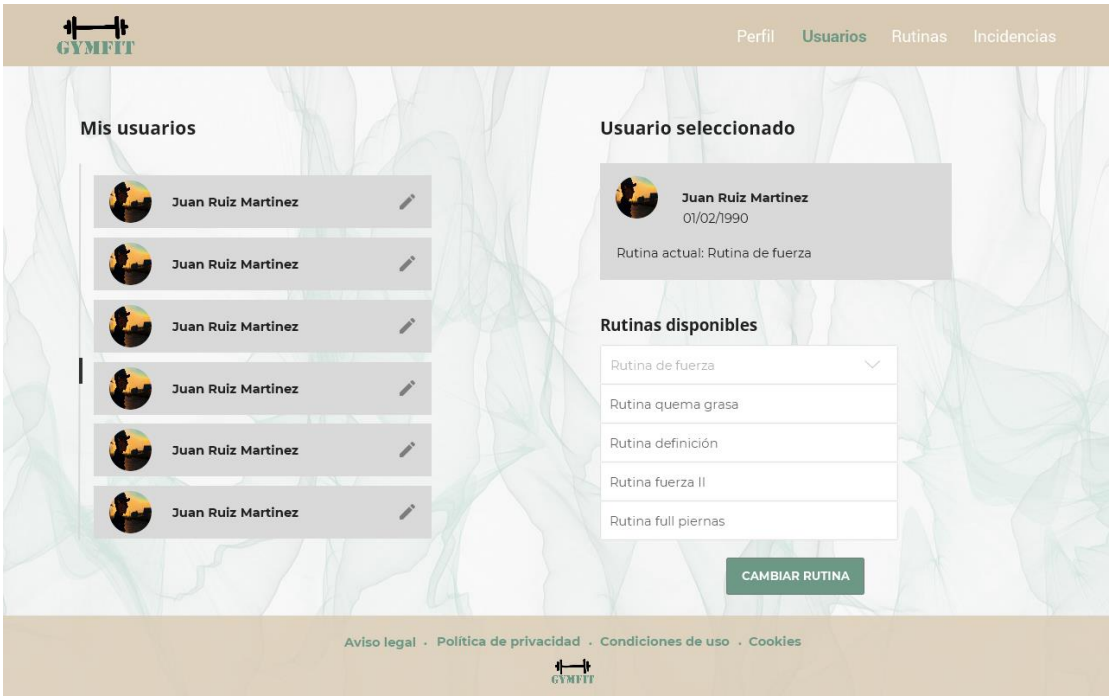


Figura 28: Mockup usuarios asignados al monitor

Pantalla Rutinas

Se ha mantenido la misma estructura que en el wireframe, añadiendo un scroll para poder visualizar todas las rutinas.

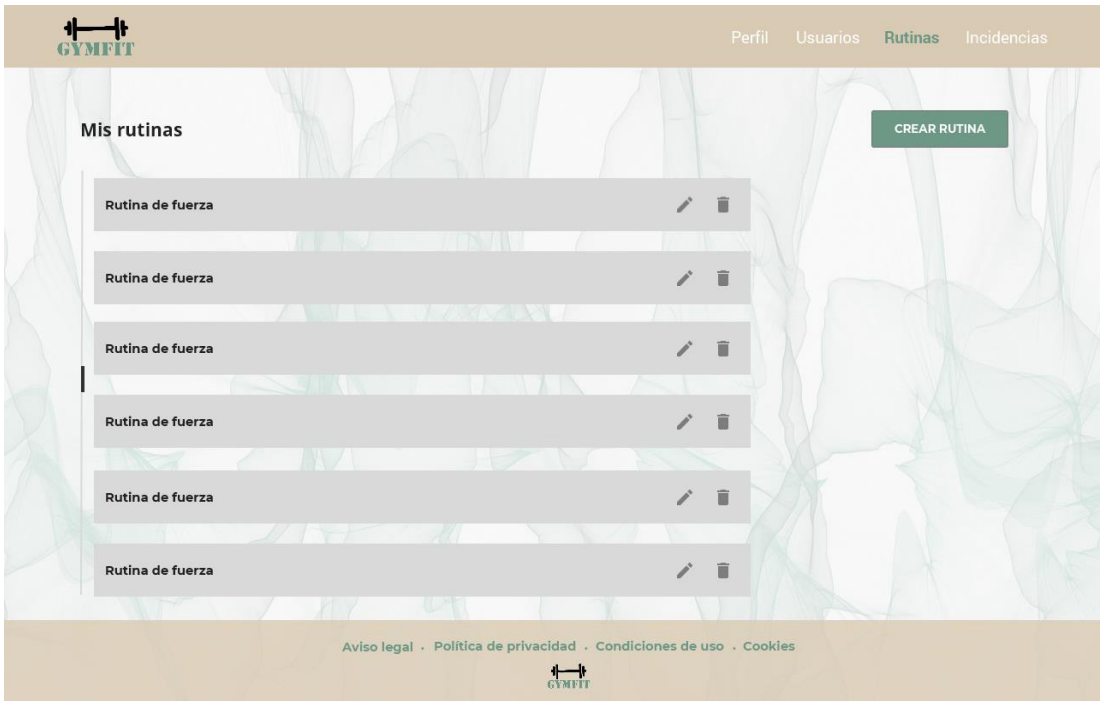


Figura 29: Mockup rutinas creadas por el monitor

Pantalla añadir/editar rutina

Como su nombre indica, es la pantalla que utilizaremos para crear o editar una rutina. En la cual podemos ir seleccionando los ejercicios que se pueden realizar en el gimnasio y que el administrador habrá introducido previamente. Se ha mantenido la misma estructura que en el wireframe.

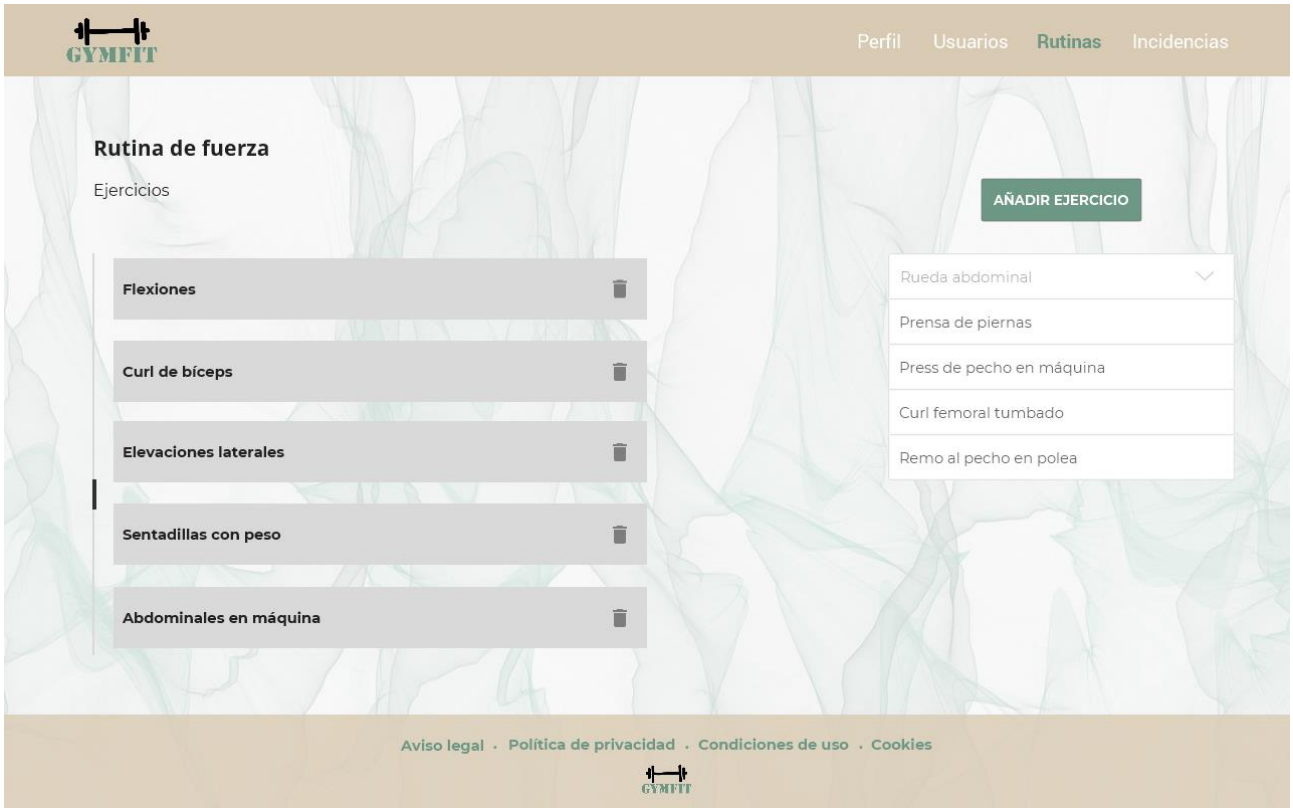


Figura 30: Mockup añadir/editar rutina

6. Usabilidad/UX

Durante el desarrollo de la aplicación web se ha tenido en cuenta la usabilidad de la misma, debido a que multitud de usuarios van a interactuar con nuestra aplicación, debemos intentar que esta sea lo más fácil y cómoda posible. Obtener una buena usabilidad hará que los usuarios sean más fieles, reducirá costes y aumentará la eficiencia. Por ello, durante todo el desarrollo nos hemos basado en los 10 principios de diseño web y usabilidad que redactó Jakob Nielsen:

1. **Visibilidad del estado del sistema.** La aplicación web debe mostrar en todo momento qué está pasando y en qué punto de la navegación se encuentra. Esto se puede ver en nuestra aplicación, en el header, donde se muestra con un color diferente en la sección que nos encontramos en ese momento.
2. **Adecuación entre el sistema y el mundo real.** La aplicación web o sistema debe hablar con el mismo lenguaje que los usuarios. Un ejemplo de esto en nuestra aplicación es cuando mostramos un error al hacer login, en vez de mostrar el error de código, mostramos un mensaje que el usuario entiende.
3. **Libertad y control por el usuario.** Los usuarios pueden volver fácilmente a un estado anterior. Por lo que es conveniente mostrar las opciones de “deshacer” y “rehacer” cuando realizan una acción. En nuestra aplicación, se puede ver este principio cuando mostramos el listado de rutinas del monitor, donde el monitor tiene la posibilidad de volver a una página anterior de rutinas.
4. **Consistencia y estándares.** Es conveniente seguir y repetir algunos patrones que ya están más que probados y asentados, para no confundir a los usuarios. En nuestra aplicación web se puede ver en el menú de navegación, que aparece en la parte superior al igual que en la mayoría de las webs, así los usuarios no necesitan invertir tiempo en buscar donde está.
5. **Prevención de errores.** Es mucho mejor prevenir los errores para que el usuario no los cometa en vez de generar mensajes de explicación cuando el usuario ya ha cometido el error. Podemos verlo en nuestra aplicación en el formulario de alta, donde te va mostrando si los datos introducidos son válidos o por el contrario tienen algún error.
6. **Reconocer mejor que recordar.** Se quiere que el usuario memorice lo menos posible, por lo que hay que mostrar los diferentes objetos, acciones y opciones para que el usuario pueda lograr su objetivo. Un ejemplo sería cuando se te muestra un historial de los monitores que has visitado recientemente.
7. **Flexibilidad y eficiencia de uso.** Se debe adaptar la web a todo tipo de usuarios, desde los más expertos hasta los que entran por primera vez. Es ideal crear atajos de teclado o aceleradores para mejorar el rendimiento y la usabilidad a los usuarios más experimentados, pero estas acciones no dificultan la navegación a los usuarios con menos experiencia. Un ejemplo sería en el formulario de registro, que al usar la tabulación vaya cambiando de input.
8. **Estética y diseño minimalista.** Decorar en exceso una aplicación solo puede llevar a confusión, por lo que hay que intentar simplificar lo máximo posible y eliminar todos los elementos innecesarios

que generan ruido, puedan distraer al usuario y no aporten información. Con esto conseguiremos diseños limpios y mejoraremos la carga de nuestra aplicación. En nuestra aplicación se ha tenido en cuenta los tamaños utilizados, una paleta de colores adecuada para la aplicación y se ha jugado con los espacios para crear un diseño limpio y elegante.

9. **Ayudar a los usuarios a reconocer, diagnosticar y solucionar los errores.** Hay que mostrar mensajes de error amigables para el usuario, evitando mostrar errores que llegan directamente. Un ejemplo puede ser mostrar directamente un error 500, sería mejor mostrar el error con un lenguaje más sencillo y entendible para que cualquier usuario pueda entenderlo y saber cuál es la solución.
10. **Ayuda y documentación.** Se debe disponer de una ayuda que el usuario pueda identificar de forma fácil para que pueda resolver sus dudas, además esta ayuda no tendrá que ser demasiado extensa. Una opción sería tener una sección de FAQs o preguntas frecuentes en el footer de la aplicación para que los usuarios puedan ver las respuestas.

Al seguir los principios mencionados anteriormente, la aplicación es muy parecida a las que hay en la actualidad, por lo que los usuarios nuevos no tendrán ningún impedimento para poder utilizar la aplicación de forma cómoda, rápida y sin necesidad de memorizar.

También se ha tenido en cuenta durante el desarrollo de la aplicación los siguientes patrones de diseño:

- **Selección de calendario.** Utilizado en el registro y la edición del perfil, cuando el usuario o monitor desea buscar o enviar información en función de una fecha.
- **Pestañas de navegación.** Utilizado en las rutinas del usuario para poder cambiar entre su rutina y los monitores.
- **Enlace de inicio.** En todas las páginas tenemos un enlace que nos lleva a la página de inicio, en nuestro caso es el logo de la aplicación.
- **Tarjetas.** En varios lugares de la aplicación utilizamos las tarjetas para mostrar la información, como puede ser cuando un monitor selecciona a un usuario.
- **Registro de cuenta.** Para conocer los usuarios activos de nuestra aplicación.
- **Scroll continuo.** Se ha utilizado en distintos lugares de la aplicación, como puede ser cuando el usuario está viendo los ejercicios que tiene su rutina.
- **Modales.** Para confirmar algo antes de borrarlo.
- **Colores de fila alternativos.** Cuando mostramos al monitor sus rutinas, se alterna el color de las filas para que pueda visualizar correctamente cada una de ellas.

7. Seguridad

Durante todo el diseño y desarrollo de la aplicación web, se ha tenido en cuenta la seguridad de la misma y las posibles vulnerabilidades que esta pudiera tener.

En la parte backend, se ha incorporado la autenticación mediante *JWT* (*JSON Web Tokens*), el cual es un estándar abierto de uso libre y disponible para varios lenguajes de programación. Lo utilizamos cuando tenemos que validar o enviar datos sensibles, como pueden ser las contraseñas. En vez de guardar o enviar las contraseñas en texto plano, las ciframos con *JWT* y así son incapaces de leer. Cuando recibimos una petición de login, ciframos la contraseña y verificamos que coincide con la que tiene el usuario guardada en la base de datos, en caso de no coincidir se devuelve un error genérico que no especifica el campo concreto que ha sido incorrecto.

Al lanzar la aplicación back, se necesita pasar por variables de entorno ciertas contraseñas, ya que así no están puestas en el código y si alguien accediera al código no podría acceder a los datos que tenemos almacenados. Las contraseñas que pasamos por variable de entorno son la de cifrado para *JWT* y las de acceso a la base de datos. Además, se utiliza una librería para la interacción con la base de datos, dicha librería es el *ORM* Sequelize, el cual proporciona una seguridad extra a la hora de interactuar con la base de datos, como puede ser el escapado de caracteres para no tener inyecciones *SQL*.

Como mejora en el futuro, se quiere dockerizar el backend para que el código no quede expuesto en ningún servidor web. Para ello se utilizará *Docker*, que es una herramienta para crear contenedores ligeros y portables para las aplicaciones que pueden ejecutarse en cualquier máquina que tenga *Docker* instalado, independientemente del sistema operativo. Gracias a dockerizar la aplicación, no tendremos expuesto el código ni las librerías necesarias para el funcionamiento del mismo, por lo que será más complicado que un virus infecte nuestra aplicación.

8. Requisitos de instalación

SERVIDOR

Para poder instalar los requisitos necesarios para que funcione correctamente el servidor se necesitan unos conocimientos básicos de servidores y aplicaciones web.

Los requisitos hardware necesarios para poder instalar la aplicación del servidor son los siguientes:

- **Ordenador**. Un ordenador con Linux, Windows o Mac.
- **Hosting**. En caso de correr la aplicación en la nube, será necesario un hosting.

Los requisitos software necesarios para poder instalar la aplicación del servidor son los siguientes:

- **Node.js**. Tener instalado Node.js, a partir de la versión v14.17.6
- **Npm**. Tener instalado npm con la versión 6.14.15 o superior.
- **Angular**. Tener instalado Angular CLI con la versión 12.2.6

CLIENTE

Para poder instalar los requisitos necesarios para que funcione correctamente el cliente, no se necesita una experiencia previa, cualquier usuario promedio puede realizar la instalación sin problemas.

Los requisitos hardware necesarios para poder instalar la aplicación del cliente son los siguientes:

- **Ordenador**. Un ordenador con cualquier sistema operativo.
- **Tablet**. Una tablet con cualquier sistema operativo.
- **Móvil**. Un móvil con cualquier sistema operativo.

Los requisitos software necesarios para poder instalar la aplicación del cliente son los siguientes:

- **Navegador**. Un navegador con conexión a internet o conectado en la misma red local que esté corriendo el servidor. Para las pruebas se ha utilizado el navegador Chrome.

9. Instrucciones de instalación

Para poder instalar la aplicación, debemos tener instalado Node.js con la versión 14.17.6 y su gestor de paquetes npm, además de Angular cli con la versión 12.2.6.
Una vez instalado el software necesario para poder ejecutar la aplicación, podemos proceder a instalar la parte backend con el comando “npm ci”, esto nos instalará todas las dependencias que necesita el proyecto para su correcto funcionamiento. Una vez instaladas las dependencias, podemos ejecutarlo con el comando “npm run start” el cual se pondrá a escuchar las peticiones en el puerto 3100 y guardará los datos en una base de datos *SQLITE* llamada “gymfit.development.sqlite” en el propio directorio de la aplicación. Para la parte frontend tendremos que usar el comando “npm i” para instalar todas las dependencias de esta parte de la aplicación y luego podremos poner en marcha la aplicación con el comando “ng serve --open”, con esto se nos abrirá una pestaña en el navegador para poder empezar a utilizar la aplicación.

10. Instrucciones de uso

Una vez se nos abre la pestaña en el navegador, ya podemos empezar a utilizar la aplicación como usuarios, más adelante explicaremos como acceder si somos monitores.

1. **Acceso a la aplicación**. Lo primero que se nos presenta es la pantalla de login, en la cual podemos rellenar el formulario de acceso y pulsar en el botón de “login”. Si los datos son correctos se nos redirigirá automáticamente al Perfil del usuario.

2. **Registro como usuario para acceder a la aplicación**. Si no disponemos de una cuenta, en la primera pantalla que nos aparece, se nos muestra un link para crear una cuenta. Si vamos al link, nos aparecerá un formulario con todos los datos necesarios para registrarnos en la aplicación. Una vez rellenados todos los datos de forma correcta pulsaremos sobre el botón de “registro” y nos redirigirá automáticamente al Perfil del usuario. Si algún dato no es correcto, se nos marcará en rojo con un texto explicando el error.

3. **Perfil del usuario**. Es la primera pantalla que ve el usuario una vez ha hecho login o se ha registrado, en esta pantalla podrá ver su información y editarla en caso que sea necesaria. Para poder editar la información solo tiene que modificar la que necesite y pulsar sobre el botón de guardar. Si la información es correcta se le actualizará, en caso de que algún campo no sea correcto se le mostrará en rojo y con un texto explicativo.

4. **Rutina del usuario**. Para poder acceder a la pantalla de rutinas, el usuario puede seleccionar la opción desde la barra de navegación superior. Una vez en la pantalla el usuario puede visualizar la rutina que tiene activada, pudiendo ver todos los ejercicios que componen esa rutina. Además, puede empezar a realizar la rutina pinchando en el botón de “empezar rutina” y cuando termine de entrenar puede indicarlo pinchando en el botón que se habilita una vez empezada la rutina “parar rutina”.

5. **Ver las estadísticas**. Para poder acceder a la pantalla de estadísticas, el usuario puede seleccionar la opción desde la barra de navegación superior. Una vez en la pantalla, el usuario puede ver las distintas estadísticas que el sistema le ofrece.

6. **Registrar una Incidencia**. Para poder registrar una incidencia, el usuario puede seleccionar la opción desde la barra de navegación superior. Una vez en la pantalla solo tiene que rellenar el formulario explicando lo que ha ocurrido y pulsar sobre el botón de “enviar”. Si la incidencia se registra correctamente el formulario se limpiará y se le avisará al usuario, en caso contrario se mostrar el error.

Si somos monitores y queremos acceder a la aplicación, tendremos que situarnos en la URL “/monitor/login” ya que si intentamos hacer login en la pestaña que nos abre la aplicación no nos dejará.

7. **Acceso a la aplicación**. Una vez situados en la ruta correcta, se nos muestra un formulario de acceso, rellenamos el formulario con los datos y pulsamos sobre el botón de “login”. Si los datos son correctos se nos redirigirá automáticamente al Perfil del monitor. Si hay algún dato incorrecto, se nos mostrará un mensaje de error con un texto explicativo.

8. **Perfil del monitor**. Una vez logueados se nos muestra la pantalla de perfil, en la que podemos ver la información del monitor y editar la misma. Para modificar la información solo se tiene que modificar los datos que se quieran cambiar y pulsar sobre el botón de “guardar”. Si la información es correcta se le actualizará, en caso de que algún campo no sea correcto se le mostrará en rojo y con un texto explicativo.

9. **Visualizar mis usuarios**. Para poder ver los usuarios que tiene el monitor asignados, el monitor puede seleccionar la opción desde la barra de navegación superior. Una vez en la pantalla verá una lista con todos los usuarios que tiene asignados ese monitor y se le seleccionará el primer usuario. En la lista puede seleccionar el usuario con el que quiere trabajar y en la parte derecha de la aplicación se le cargará la información de ese usuario. Una vez tiene el usuario seleccionado, se le puede cambiar la rutina que tiene asignada con el desplegable de rutinas disponibles por ese monitor. Para cambiar la rutina solo tendría que seleccionar la rutina que le quiere asignar al usuario y pulsar sobre el botón de “cambiar rutina”. Si todo ha ido correcto se le notificará al monitor y el usuario verá su nueva rutina, en caso de que haya algún error se le notificará con un mensaje de texto.

10. **Listado de rutinas**. Para poder ver las rutinas que ha configurado el monitor, puede seleccionar la opción desde la barra de navegación superior. Una vez esté en la pantalla verá un listado de las rutinas que ha ido creando, las rutinas las puede editar o eliminar con los botones que aparecen al lado de ellas. Además, el monitor puede crear una rutina nueva pulsando sobre el botón de crear rutina.

11. **Crear o editar una rutina**. Para poder crear o editar una rutina el monitor ha tenido que pulsar sobre el botón de “crear rutina” desde la pantalla “rutinas”. Una vez ahí, se le mostrará un listado de los ejercicios que componen esa rutina y el nombre de la misma, a la derecha de la pantalla se mostrar un desplegable con los ejercicios disponibles y un botón para añadir el ejercicio. Si el monitor quiere eliminar algún ejercicio que ha añadido a la rutina, lo puede hacer pulsando sobre la papelera que aparece al lado del ejercicio, si se ha eliminado correctamente el listado se refrescará y el ejercicio habrá desaparecido.

12. **Registrar una Incidencia**. Para poder registrar una incidencia, el monitor puede seleccionar la opción desde la barra de navegación superior. Una vez en la pantalla solo tiene que rellenar el formulario explicando lo que ha ocurrido y pulsar sobre el botón de “enviar”. Si la incidencia se registra correctamente el formulario se limpiará y se le avisará al monitor, en caso contrario se mostrar el error.

11. Proyección a futuro

Información, predicciones y sugerencias acerca de ampliaciones a futuro del trabajo, y/o lista de mejoras a realizar en hipotéticas futuras versiones del servicio/aplicación.
El proyecto tiene pendiente la realización del módulo de administración, para que el dueño del gimnasio pueda dar de altar los monitores, añadir/modificar/eliminar ejercicios y controlar la actividad del gimnasio con varias estadísticas. Por el momento estas acciones se pueden realizar directamente sobre la base de datos o con peticiones al backend.

Antes de empezar con el módulo del administrador, será necesario completar las tareas que han quedado pendientes por falta de tiempo de desarrollo, además de añadir una serie de mejoras al proyecto como las que se detallan a continuación:

- **Añadir certificado SSL**. Se incluirá un certificado SSL para mejorar la seguridad y posicionamiento de la aplicación.

- **Reforzar la seguridad en la parte backend**. Se van a implementar una serie de mejoras en la parte backend para reforzar la seguridad, ya que hasta el momento realizar algunas acciones sencillas pero necesarias.

- **Realizar una documentación de los endpoints**. Se realizará una documentación de los endpoints necesarios por si alguna otra aplicación se desea integrar con la nuestra.

- **Optimizar la aplicación**. Se realizará una serie de acciones para optimizar la aplicación, para que funcione en la mayoría de los navegadores, que la carga sea rápida en caso de no disponer de buena conexión a internet y aplicar lazy loading para no cargar cosas que no se están consultando.

- **Traducciones**. Se traducirá toda la aplicación al inglés y se añadirá en la cabecera las banderas de España y Reino Unido para que al pinchar sobre ellas la aplicación se traduzca automáticamente.

12. Análisis de mercado

Una vez se tuvo la idea de desarrollar una aplicación web para la gestión de gimnasios, se ha hecho un análisis de mercado, para ver las aplicaciones que existen en el mercado actual. Gracias a esto, podemos saber las necesidades que cubren y las posibles necesidades de los clientes que no están cubriendo para tener una visión global de las soluciones que hay en el mercado.

Lo primero que realizamos fue un análisis de Debilidades, Amenazas, Fortalezas y Oportunidades (*DAFO*), el cual nos permite establecer las estrategias para que nuestra aplicación sea viable. El análisis se divide en dos partes, la primera es un análisis interno con las fortalezas y debilidades y la segunda es un análisis externo con las amenazas y oportunidades de las empresas del mundo exterior.

	INTERNO	EXTERNO
NEGATIVO	Necesidad de contratar un servidor web. Fecha límite de desarrollo. Escasa inversión.	Mercado limitado. Gran competencia en el mercado. Soporte 24/7.
POSITIVO	Aplicación gratuita. Compatible con todos los navegadores. Accesible desde el teléfono móvil. Proyecto de código libre. Sistema simple.	Reducción en los costes. Sistemas demasiados complejos con múltiples opciones.

Tabla 14: Análisis DAFO

Las estrategias que se han obtenido después de realizar el análisis *DAFO* han sido las siguientes:

- Creación de la aplicación responsive para llegar a un mercado mayor.
- Crear una aplicación simple, que pueda manejar cualquier tipo de usuario.
- Implantación de metodologías ágiles para una correcta planificación.

Una vez realizado el análisis *DAFO*, vamos a estudiar a la competencia, para ello hemos realizado una búsqueda con las siguientes palabras clave “aplicación web gimnasios”, “software gestión gimnasios” y “aplicación web para gestionar gimnasios”. Nos aparecen múltiples empresas, por lo que vamos a analizar alguna de ellas.

ISMYGYM

Es una de las que aparece al principio cuando buscamos “aplicación web gimnasios”. Nos ofrece una prueba de 30 días gratuita, una vez superada la prueba sus tarifas van desde los 36€ hasta los 99€ mensuales. Nos ofrece una aplicación para clientes y otra para los administradores, ambas muy completas. Entre sus funcionalidades ofrece reservas online, pago online, rutinas deportivas, control de peso y nutrición, incidencias y fidelización, todo ello desde su aplicación web o desde una aplicación móvil para Android e IOS.

Es nuestra principal competidora ya que ofrece las principales características que ofrece GymFit para los clientes, a pesar de que no ofrece la aplicación para monitores, pero sí para administradores.

Figura 31: Página principal de IsMyGym (https://www.ismygym.com/).

T2GO

Es una herramienta para que los monitores puedan gestionar los entrenamientos de sus clientes. Permite editar los entrenos de los clientes, gestionar grupos y planes de entrenamiento, disponer de entrenos favoritos e integración con Strava, todo ello desde una aplicación web o aplicación móvil para Android e IOS. Los precios van desde los 10€ hasta los 70€ mensuales, con una opción gratis para 2 clientes.

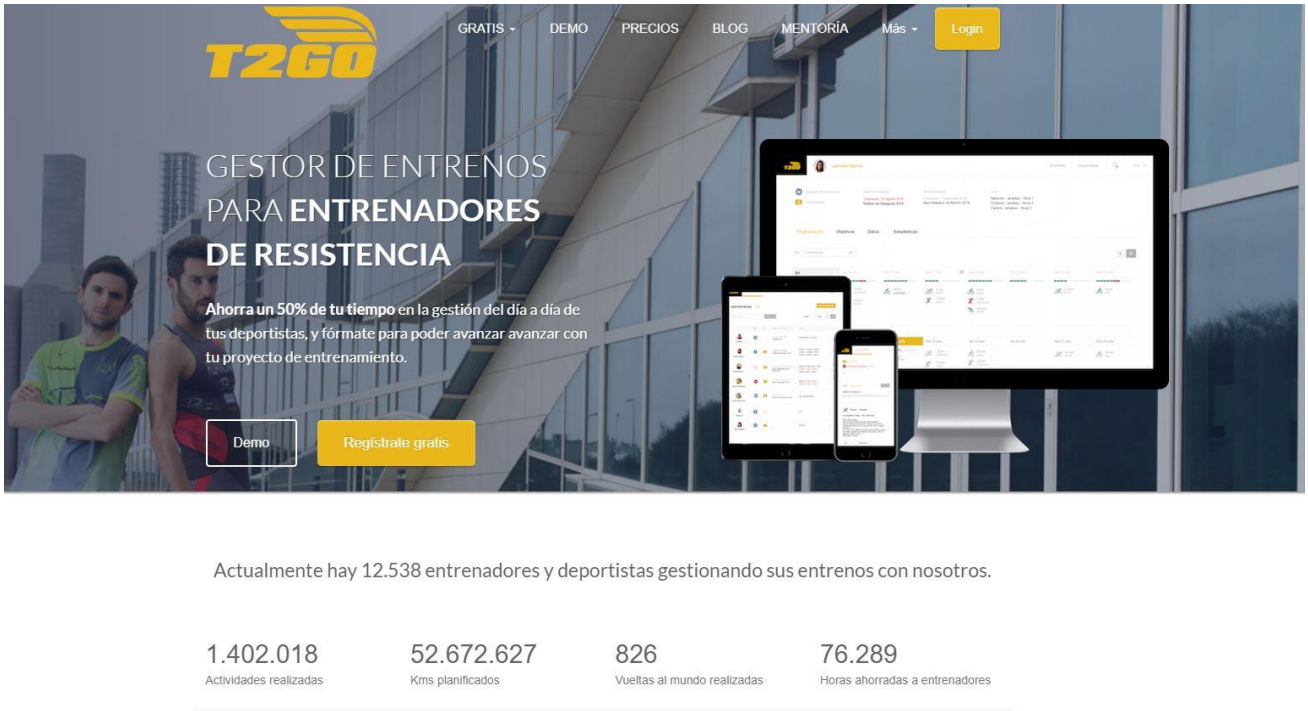


Figura 32: Página principal de T2GO (https://train2go.com/).

Trainingym

Es un software para la gestión de negocios fitness que se centra en mejorar objetivos. Permite entrenamientos multiplataforma, reserva de actividades grupales, gestión de marketing y comunicación, gestión de clientes y gamificación. Los precios van desde 179,9€ hasta los 199,9€ mensuales, con una demo gratuita para que pruebes sus funcionalidades.

Figura 33: Página principal de trainingym (https://trainingym.com/soluciones-para-gimnasios).

Gestigym

Es un software de gestión deportiva para gimnasios con posibilidad de instalación y configuración de control de accesos. Ofrece una aplicación móvil para que los clientes puedan reservar las clases y consultar sus rutinas de entrenamiento. El software permite la gestión de clientes, reservas y pagos, control de los empleados y almacén, informes de pagos y rutinas de entrenamiento. Los precios van desde 999€ hasta los 3.999€ en un pago único.

Figura 34: Página principal de gestigym (https://gestigym.com/).

Una vez analizadas las distintas competencias del proyecto, vemos que muchas de ellas cubren las mismas necesidades que GymFit, pero son opciones de pago con un coste mensual por empleados o bien un pago único inicial muy grande. La ventaja de GymFit, es que, al ser de código abierto, la comunidad o clientes pueden desarrollar funcionalidades que necesiten, por lo que la aplicación web puede ir escalando a medida que las empresas lo necesiten.

13. Conclusiones

El objetivo principal de este proyecto era el de crear una aplicación web para poder gestionar gimnasios, que se pudiera adaptar a cualquier dispositivo y fuera sencilla de utilizar por los usuarios. Una vez terminada la planificación, diseño y desarrollo de la aplicación web podemos afirmar que se ha conseguido el objetivo propuesto. Tenemos una aplicación que permite a los usuarios gestionar sus datos, consultar las rutinas o crear/editar/borrar las mismas, ver las estadísticas y poder enviar incidencias. Además, la aplicación se adapta a cualquier dispositivo y es muy sencilla de utilizar, por lo que no será un impedimento como sí lo son las de la competencia.

El poder realizar este proyecto me ha servido para poder aprender una tecnología que era totalmente desconocida para mí, como es el caso de Angular, que tiene un futuro muy prometedor de cara al mercado laboral. Además, me permite tener un portfolio con una de las tecnologías más demandadas en estos momentos. También he podido profundizar mis conocimientos a la hora de planificar y organizar un proyecto, ya que he aprendido cosas nuevas tanto en la asignatura del máster como mientras realizaba la planificación del proyecto. Finalmente, he mejorado mis habilidades en HTML y CSS, cosa que hasta hace un par de años pensaba que sería imposible, por lo que salgo muy satisfecho con los conocimientos que he obtenido realizando este máster y proyecto.

Como trabajo futuro queda implementar el apartado de mejoras que se nombra en esta memoria, pero se puede lanzar una primera versión totalmente funcional para ver su funcionamiento con los usuarios reales y que puedan testear la aplicación, para en el caso de ser necesario, añadir mejoras y poder implementarlas en un futuro.

Finalmente, quiero añadir que este trabajo me ha ayudado a sintetizar e integrar las competencias y enseñanzas aprendidas durante todo el máster, por lo que estoy muy satisfecho de los resultados obtenidos y de las habilidades adquiridas durante todo este tiempo.

Anexo 1. Entregables del proyecto

Listado de archivos que se entregan y su descripción.

PEC_Final_prj_Ignacio_Vicent_Salvador.zip

- **Carpeta backend**. Contiene el código del proyecto para la parte backend.
- **Carpeta frontend**. Contiene le código del proyecto para la parte frontend.
- **Carpeta bbdd**. Contiene una base de datos con datos iniciales. Hay que poner esta bbdd dentro de la carpeta backend si se quiere utilizar.

PEC_Final_mem_Ignacio_Vicent_Salvador.zip

- **TFM_mem_Ignacio_Vicent_Salvador.pdf**. Contiene la memoria completa del trabajo final de máster.
- **Informe_Autoevaluacion_TFM_Ignacio_Vicent_Salvador.** Contiene el autoinforme.
- **Anexos.** Carpeta con los distintos anexos del proyecto.
    - **LO-FI**. Todas las imágenes LO-FI que se muestran en la memoria del proyecto.
    - **HI-FI**. Todas las imágenes HI-FI que se muestran en la memoria del proyecto.
    - **Imágenes**. El resto de imágenes que se muestran en la memoria del proyecto.

PEC_Final_URL_Aplicacion_Web_Ignacio_Vicent_Salvador.pdf

Se trata del fichero donde están las URL para poder probar la aplicación

PEC_Final_vid_Ignacio_Vicent_Salvador.mp4

Se trata del video de presentación de la aplicación web en .mp4

PEC_Final_prs_Ignacio_Vicent_Salvador.zip

- **PEC_Final_prs_Ignacio_Vicent_Salvador.pdf.** Presentación del proyecto final en formato PDF.
- **PEC_Final_prs_Ignacio_Vicent_Salvador.pptx.** Presentación del proyecto final en formato Powerpoint.

Anexo 2. Código fuente (extractos)

Backend

Para la parte backend se va a mostrar un ejemplo de CRUD (créate, read, update and delete), para ello se mostrará el fichero de rutas, el de controlador y el de repositorio.

src/routes/user.js

```
import { celebrate, Joi } from "celebrate";
import express from "express";
import { User } from "../controllers";

const router = express.Router();

router.get(
    "/:id",
    celebrate(
      {
        params: {
          id: Joi.string().required(),
        },
      },
      { abortEarly: false }
    ),

    (req, res) => {
        User.getUser(req, res);
    }
  );


router.post(
  "/startWorkout",
  celebrate(
    {
      body: Joi.object().keys({
        userId: Joi.string().required(),
        workoutId: Joi.number().required(),
      }),
    },
    { abortEarly: false }
  ),
  (req, res) => {
    User.startUserWorkout(req, res);
  }
);
```

```
router.put(
  "/:id",
  celebrate(
    {
      body: Joi.object().keys({
        name: Joi.string().required(),
        surname: Joi.string().allow(null),
        birthday: Joi.date().optional().allow(null),
        gender: Joi.string().allow(null),
        email: Joi.string().required(),
        password: Joi.string().required(),
      }),
    },
    { abortEarly: false }
  ),
  (req, res) => {
    User.updateUser(req, res);
  }
);


export default router;
```

src/controllers/user.js

```
import moment from "moment";
import { sendErrorResponse } from "../core";
import { Users } from "../repository";


export async function getUser(req, res) {
    try {
        const user = await Users.findUserById(req.params.id);
        if(!user) {
            res.status(404).send({ message: "User not found" });
            return;
        }
        res.status(200).send(user);
    } catch (error) {
        console.info("GET user ERROR", new Date(), " with error ", error);
        sendErrorResponse(res, error);
    }
}

export async function updateUser(req, res) {
    try {
        const { id } = req.params;
```

```
        const { name, surname, email, password, birthday, gender } = req.body;


        await Users.updateUser({
            id,
            name,
            surname,
            email,
            password,
            birthday,
            gender
        });

        res.status(201).send();
    } catch (error) {
        console.info("UPDATE USER ERROR", moment(), " with error ", error);
        sendErrorResponse(res, error);
    }
}

export async function startUserWorkout(req, res) {
    try {
        const {userId, workoutId } = req.body;

        await Users.startWorkout({userId, workoutId });

        res.status(201).send();
    } catch (error) {
        console.info("UPDATE START_USER_WORKOUT ERROR", moment(), " with error ",
error);
        sendErrorResponse(res, error);
    }
}
```

src/repository/user.js

```
import db from "../models";

const findUserById = async (id) =>
  await db.user.findOne({
    where: { id, isDeleted: 0 },
});

const getUserData = async ({ id }) =>
  await db.user.findOne({
    where: { id: id, isDeleted: 0 },
    attributes: ["id", "name", "surname", "email", "password", "nick"],
  });
```

```
const findUserLogin = async (email, password) =>
  await db.user.findOne({
    where: { email: email, password, isDeleted: 0 },
  });

const saveUser = async ({
  name, surname, birthday, gender, email, password
}) =>
  await db.user.create({
    name, surname, birthday, gender, username: email, email, password
  });

  const updateUser = async ({
    id, name, surname, email, password, birthday, gender
  }) =>
    await db.user.update(
      {
        name, surname, email, password, birthday, gender
      },
      {
        where: {
          id
        }
      }
    );


const startWorkout = async ({
  userId,
  workoutId
}) =>
  await db.user_workout_history.create({
    idUser: userId,
    idWorkout: workoutId,
    type: "START",
    date: new Date()
  });

export { findUserById, getUserData, findUserLogin, saveUser, updateUser,
startWorkout };
```

Frontend

Para la parte fronted se va a mostrar un ejemplo de servicio, para mostrar cómo se comunica la parte frontend con la parte backend y también un controlador.

src/app/services/user.service.ts

```
import { HttpClient } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { Observable } from 'rxjs';
import { UserDTO } from '../models/user.dto';
import { WorkoutDTO } from '../models/workout.dto';

@Injectable({
  providedIn: 'root',
})
export class UserService {
  private urlGymfit: string;
  private controller: string;

  constructor(private http: HttpClient) {
    this.controller = 'user';
    this.urlGymfit = 'http://localhost:3100/' + this.controller;
  }

  register(user: UserDTO): Observable<UserDTO> {
    return this.http.post<UserDTO>(this.urlGymfit, user);
  }

  updateUser(userId: string, user: UserDTO): Observable<UserDTO> {
    return this.http.put<UserDTO>(this.urlGymfit + '/' + userId, user);
  }

  getUSerById(userId: string): Observable<UserDTO> {
    return this.http.get<UserDTO>(this.urlGymfit + '/' + userId);
  }

  startWorkout(userId: string, workoutId: string): Observable<void> {
    return this.http.post<any>(this.urlGymfit + '/startWorkout', {
      userId,
      workoutId,
    });
  }
  stopWorkout(userId: string, workoutId: string): Observable<void> {
    return this.http.post<any>(this.urlGymfit + '/stopWorkout', {
      userId,
      workoutId,
    });
  }
}
```

src/app/components/user/login/login.component.ts

```
import { Component, OnInit } from '@angular/core';
import {
  FormBuilder,
  FormControl,
  FormGroup,
  Validators,
} from '@angular/forms';
import { Router } from '@angular/router';
import { AuthDTO } from 'src/app/models/auth.dto';
import { AuthService } from 'src/app/services/auth.service';
import { LocalStorageService } from 'src/app/services/local-storage.service';
@Component({
  selector: 'app-login',
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.scss'],
})
export class LoginComponent implements OnInit {
  public user: AuthDTO;
  public status: string;
  email: FormControl;
  password: FormControl;
  loginForm: FormGroup;

  constructor(
    private formBuilder: FormBuilder,
    private authService: AuthService,
    private _router: Router,
    private localStorageService: LocalStorageService
  ) {
    this.user = new AuthDTO('', '', '', 'juan@gmail.com', 'testtest');
    this.status = '';

    this.email = new FormControl(this.user.email, [
      Validators.required,
      Validators.pattern('[a-z0-9._%+-]+@[a-z0-9.-]+.[a-z]{2,4}$'),
    ]);

    this.password = new FormControl(this.user.password, [
      Validators.required,
      Validators.minLength(8),
      Validators.maxLength(16),
    ]);

    this.loginForm = this.formBuilder.group({
      email: this.email,
      password: this.password,
```

```
    });
  }

  ngOnInit(): void {}

  async login(): Promise<void> {
    let responseOK: boolean = false;
    let errorResponse: any;

    this.user.email = this.email.value;
    this.user.password = this.password.value;

    this.authService.login(this.user).subscribe(
      (user) => {
        responseOK = true;
        this.user.access_token = user.access_token;
        this.localStorageService.set('access_token', user.access_token);
        this.localStorageService.set('user_id', user.user_id);
        this.localStorageService.set('mode', 'user');

        this._router.navigate(['profile']);
      },
      (error) => {
        console.log('Error es ', error);
        responseOK = false;
        errorResponse = error.error;
      }
    );
  }
}
```

Anexo 3. Librerías/Código externo utilizado

A continuación, se expone información detallada acerca de qué librerías, código, archivos, y cualquier otra herramienta tecnológica desarrollada por terceros utilizada en el trabajo, y qué partes de los mismos han sido usadas y cómo.

Bootstrap

Se ha utilizado bootstrap como base para algunos elementos de la aplicación, también para la estructura del miso, facilitándonos el diseño y la parte responsive de la aplicación. El diseño se ha realizado desde cero y mucho código de bootstrap utilizado ha sido modificado mediante CSS.

JWT

Se ha utilizado para generar los tokens que usan los usuarios y monitores a lo largo de la aplicación. JWT nos ofrece la creación y comprobación de tokens de forma segura, por lo que nadie que no tenga un token válido podrá utilizar la aplicación.

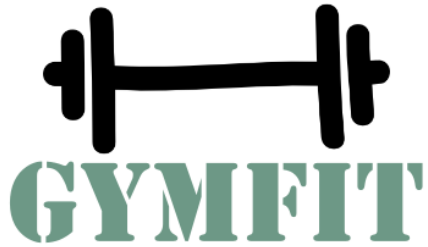
Sequelize

Se ha utilizado el ORM de Sequelize para la base de datos. Todas las consultas a la misma se realizan con el ORM, pero se ha separado de la capa del controlador, por lo que si queremos cambiar de ORM el cambio sería muy rápido. Además, Sequelize nos permite cambiar de gestor de base de datos sin necesidad de rehacer todas las queries.

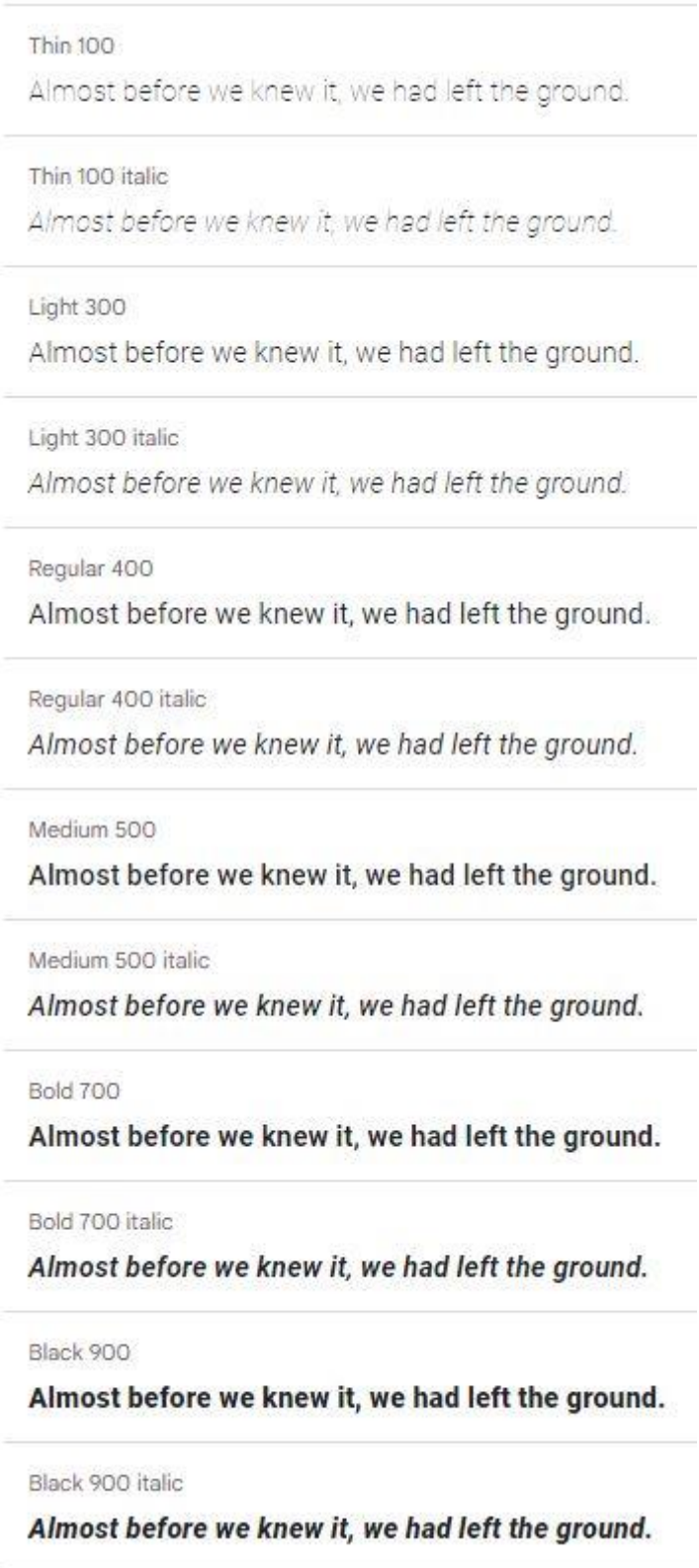
Anexo 4. Libro de estilo

A continuación, se muestra la línea gráfica del trabajo realizado.

- Logotipo.
    - Fuente: Stencil
    - Tamaño: 72



- Paleta de colores.
    - **#6D9886** color principal
    - **#D9CAB3** color secundario
    - **#212121** color del texto
    - **#F6F6F6** color secundario de texto
- Tipografía.
    - Para los títulos o zonas de menús Roboto



- Para el texto Montserrat

Thin 100

Almost before we knew it, we had left the ground.

Thin 100 italic

Almost before we knew it, we had left the ground.

Extra-light 200

Almost before we knew it, we had left the ground.

Extra-light 200 italic

Almost before we knew it, we had left the ground.

Light 300

Almost before we knew it, we had left the ground.

Light 300 italic

Almost before we knew it, we had left the ground.

Regular 400

Almost before we knew it, we had left the ground.

Regular 400 italic

Almost before we knew it, we had left the ground.

Medium 500

Almost before we knew it, we had left the ground.

Medium 500 italic

Almost before we knew it, we had left the ground.

Semi-bold 600

Almost before we knew it, we had left the ground.

Semi-bold 600 italic

Almost before we knew it, we had left the ground.

Bold 700

Almost before we knew it, we had left the ground.

Bold 700 italic

Almost before we knew it, we had left the ground.

Extra-bold 800

Almost before we knew it, we had left the ground.

Extra-bold 800 italic

Almost before we knew it, we had left the ground.

Black 900

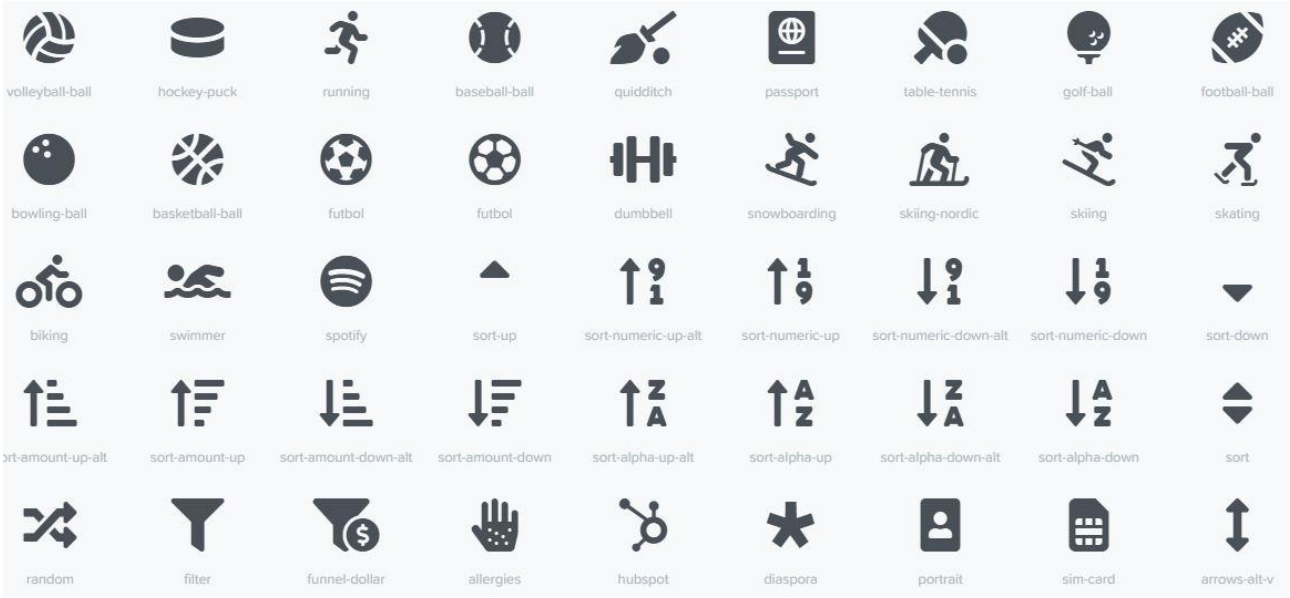
Almost before we knew it, we had left the ground.

Black 900 italic

Almost before we knew it, we had left the ground.

- Iconos

  Para los iconos utilizados en la aplicación web, se ha utilizado la página Fontawesome con licencia Creative Commons. La cual nos ofrece una serie de iconos vectoriales y estilos css. Un ejemplo de los iconos que nos ofrece son los siguientes:



- Botones
  - Botón

    BUTTON

  - Botón con hover

    BUTTON

Anexo 5. Bibliografía

Angular (2021). Documentación oficial de Angular: https://angular.io/docs

Expressjs (2021). Documentación oficial de Express: https://expressjs.com/es/

Jakob Nielsen (2020). Ten Usability Heuristics for User Interface Design: https://www.nngroup.com/articles/ten-usability-heuristics/

Schwaber, K.; Sutherland, J. (2016). The Scrum Guide: http://www.scrumguides.org/docs/scrumguide/v2016/2016-Scrum-Guide-US.pdf.

Scrumg.org (2017). ¿Qué es Scrum?: https://www.scrum.org/resources/blog/que-es-scrum

Node.js (2021). Documentación oficial de Node.js: https://nodejs.org/docs/latest-v14.x/api/

Sequelize (2021). Documentación oficial de sequelize: https://sequelize.org/

UI-Patterns (2021). Calendar Picker: http://ui-patterns.com/patterns/CalendarPicker

UI-Patterns (2021). Navigation Tabs: http://ui-patterns.com/patterns/NavigationTabs

UI-Patterns (2021). Home Link: http://ui-patterns.com/patterns/HomeLink

UI-Patterns (2021). Cards: http://ui-patterns.com/patterns/cards

UI-Patterns (2021). Account Registration: http://ui-patterns.com/patterns/AccountRegistration

UI-Patterns (2021). Continuous Scrolling: http://ui-patterns.com/patterns/ContinuousScrolling

UI-Patterns (2021). Modal windows: http://ui-patterns.com/patterns/modal-windows

UI-Patterns (2021). Alternating Row Colors: http://ui-patterns.com/patterns/AlternatingRowColors

UOC (Sin fecha). Web adaptativa: http://multimedia.uoc.edu/blogs/dii/es/tendencies/web-adaptatiu/

UOC (Sin fecha). Diseñar para varias plataformas: http://multimedia.uoc.edu/blogs/dii/es/disseny/dissenyar-per-a-diverses-plataformes/

UOC (Sin fecha). Pautas de diseño de páginas web: http://multimedia.uoc.edu/blogs/dii/es/disseny/pautes-de-disseny/pautes-de-disseny-de-pagines-web/

UOC (Sin fecha). Prototipado: http://multimedia.uoc.edu/blogs/dii/es/prototipatge/