ULPGC
Universidad de
Las Palmas de
Gran Canaria

Escuela de
Ingeniería Informática

Grado en Ingeniería Informática

Intensificación en Ingeniería del Software

Trabajo Fin de Título

RoutineXT: aplicación para realizar ejercicios de mantenimiento en casa

Autor: Néstor Bolaños Bolaños

Tutor: José Miguel Santos Espino

Junio 2021

Resumen

Hoy en día podemos encontrar muchas aplicaciones para móviles que permiten al usuario realizar ejercicios para estar en forma o fortalecer diferentes grupos musculares. No obstante, es difícil encontrar alguna que ofrezca una motivación seria para no interrumpir constantemente el esfuerzo que supone hacer deporte antes que una actividad menos productiva. Este trabajo permite que el usuario pueda realizar un seguimiento de su plan de entrenamiento personalizado mediante un sistema de puntuación, unos retos mensuales y demostrar a corto plazo los beneficios reales que obtiene al realizar ejercicios como si se tratara de un sistema de gamificación. El usuario hará uso de una aplicación web para realizar el seguimiento y configurar su plan de entrenamiento, junto con una aplicación móvil para realizar los ejercicios.

El proceso de desarrollo ha sido *SCRUM*, realizando iteraciones cada 2-4 semanas para ir desarrollando cada parte del proyecto. Estas iteraciones se han realizado mediante historias de usuario que han ido variando desde la pila de producto a la pila de "*Sprint*" para facilitar el desarrollo de las tareas.

Abstract

Nowadays we can find many mobile applications that allow the user to perform workouts to get fit or strengthen different muscle groups. However, it is difficult to find one that offers a serious motivation not to constantly interrupt the effort of doing sport rather than a less productive activity. This project allows the user to track their personalised training plan through a scoring system, monthly challenges and to demonstrate in the short term the real benefits they obtain by exercising as if it were a gamification system. The user will make use of a web application to track and configure their training plan, and a mobile application to perform the exercises.

The development process has been *SCRUM*, carrying out iterations every 2-4 weeks to develop each part of the project. These iterations have been done through user stories that have been changing from the product backlog to the sprint backlog to facilitate developmental tasks.

AGRADECIMIENTOS

Antes de comenzar, me gustaría realizar un pequeño agradecimiento al haber llegado al final de esta etapa universitaria, con la presentación de este Trabajo de Fin de Título.

Los primeros 2 años recuerdo que fueron muy difíciles para mí, porque tuve muchas dificultades con las asignaturas de programación. Me costó entender y aceptar mi incapacidad de aprobar las asignaturas de programación y, sin embargo, aprobar el resto sin tantos problemas. Quiero agradecer, en primer lugar, a mi familia por haberme apoyado e incentivado para resistir, y no abandonar. También, me gustaría agradecer a los compañeros y compañeras que he conocido que han logrado finalizar, y a otras personas que desafortunadamente no pudieron seguir. Ha sido una maravilla este período donde en muchas ocasiones nos hemos ayudado y lo hemos pasado muy bien en el entorno personal.

Finalmente, también me gustaría felicitar a mi tutor, José Miguel Santos Espino por la labor que ha realizado conmigo tanto a lo largo del desarrollo de este proyecto, como en el resto de la carrera.

¡Gracias a tod@s!

Néstor Bolaños Bolaños

Índice de Contenidos

Índice de Ilustraciones

Índice de Tablas

Capítulo 1. Introducción

En este capítulo vamos a hablar acerca de cómo surgió la idea de realizar este TFT, así como los objetivos, funcionalidades básicas que se han desarrollado, las competencias específicas cubiertas y lo que pretende aportar este TFT a los usuarios.

1.1 Antecedentes

En la actualidad existen muchas formas de realizar deporte. Hay personas que prefieren realizar ejercicio al aire libre, en grupo, o en un centro deportivo. Podemos encontrar personas que realizan ejercicio porque quieren estar en forma, porque quieren tonificar diferentes grupos musculares, o incluso porque su profesión lo requiere. No obstante, en el año 2020 hubo un confinamiento domiciliario que duró más de tres meses debido a la pandemia del COVID-19. Esto supuso que todas aquellas actividades que implicaran estar en el exterior o con más personas, se dejaran de realizar. Incluso, tras el confinamiento, no se podía hacer ejercicio con un gran número de personas porque los centros deportivos tenían un aforo limitado y existían restricciones de movilidad. Esto supuso que se incrementara considerablemente el número de horas de entretenimiento multimedia y se disminuyera mucho la necesidad de realizar deporte.

Afortunadamente, existen plataformas web y móviles para realizar deporte sin necesidad de salir de casa. En este trabajo, he desarrollado un prototipo de aplicación web denominado "*RoutineXT*" [1], que cumpla este objetivo, pero con un elemento diferenciador que aporte al usuario funcionalidades que le permitan estar motivado para realizar deporte en casa de forma constante.

El producto ideado en este TFT consiste en que un usuario registrado cree horarios personalizados que le permitan realizar ejercicios, de manera que pueda llevar a cabo un seguimiento de su progreso. Cada vez que el usuario realice una rutina de ejercicios de su horario, se le otorgarán una serie de puntos. Asimismo, si no cumple con su rutina, se le restarán puntos acumulados. De igual manera, la puntuación servirá para que el usuario conozca cómo está progresando o desarrollando su forma física.

1.2 Objetivos

Los objetivos del trabajo los podemos clasificar dependiendo de la funcionalidad que puede realizar cada tipo de usuario en la aplicación web.

Aplicación web

- *Usuario No Registrado*
  - Registrarse en el servicio.
  - Contactar con el administrador mediante un formulario.
  - Consultar las diferentes secciones de la página principal.

- *Cliente*
  - Crear y organizar rutinas de ejercicios que ha creado en su horario personalizado.

- Consultar los ejercicios por los que está formado sus rutinas de entrenamiento.
- Consultar los retos que ha realizado o desbloqueado.
- Consultar o editar los datos de su perfil.
- Consultar estadísticas de las rutinas y ejercicios realizados.
- Consultar consejos de entrenamiento.
- Imprimir horarios, ejercicios, y retos.
- Consultar un ranking de usuarios con sus puntuaciones.

- *Administrador*
  - Filtrar, eliminar, visualizar y editar las cuentas de los usuarios del sistema.
  - Responder a los correos recibidos a través del formulario de contacto.
  - Crear y organizar ejercicios de cada rutina.
  - Crear y gestionar los retos disponibles.

1.3 Competencias específicas

En este trabajo se han cubierto tres competencias específicas [2] que corresponden a la intensificación de *Ingeniería del Software:*

1.3.1 IS01

"Capacidad para desarrollar, mantener y evaluar servicios y sistemas software que satisfagan todos los requisitos del usuario y se comporten de forma fiable y eficiente, sean asequibles de desarrollar y mantener y cumplan normas de calidad, aplicando las teorías, principios, métodos y prácticas de la ingeniería del software".

Para llevar a cabo esta competencia se ha adaptado la metodología "*Scrum*" [3] a un proyecto individual. El desarrollo de las historias de usuario ha permitido cumplir con los requisitos del proyecto acorde a cada una de las vistas dependiendo del tipo de usuario. Por otra parte, debido a la gran flexibilidad que ofrece "*Angular*" [4] se ha logrado tener organizado todo el proyecto en componentes, de manera que cada uno está formado por sus propios ficheros "*HTML*" [5], "*CSS*" [6] y "*Typescript*" [7]. Además, una vez finalizada una historia de usuario específica, se ha llevado a cabo procesos de refactorización de código para facilitar su posterior mantenimiento o incremento de funcionalidades. También se ha hecho uso de nombres de variables y funciones descriptivos.

1.3.2 IS02

"Capacidad para valorar las necesidades del cliente y especificar los requisitos software para satisfacer estas necesidades, reconciliando objetivos en conflicto mediante la búsqueda de compromisos aceptables dentro de las limitaciones derivadas del coste, del tiempo, de la existencia de sistemas ya desarrollados y de las propias organizaciones".

En esta competencia, se ha hecho un análisis de las funcionalidades que ofrecen las aplicaciones móviles relacionadas con la realización de entrenamientos en casa. Posteriormente, se han desarrollado los requisitos especificados en las historias de usuario de manera que sean factibles de desarrollar en tiempo y coste. No obstante, debido al nivel de formación primerizo en las tecnologías involucradas en el proyecto, se ha hecho especial énfasis en la dificultad y tiempo de desarrollo.

1.3.3 IS04

"Capacidad de identificar y analizar problemas y diseñar, desarrollar, implementar, verificar y documentar soluciones software sobre la base de un conocimiento adecuado de las teorías, modelos y técnicas actuales".

Para poder identificar y analizar los problemas, se ha hecho uso de los prototipos realizados previamente a las tareas de implementación. Su utilidad se ha basado en anticipar aquellas historias de usuario que han supuesto una mayor cantidad de tiempo y esfuerzo. Esto ha permitido documentar soluciones alternativas para la implementación de aquellas historias de usuario que han tenido un tiempo de desarrollo mayor que el estimado.

1.4 Aportaciones

"RoutineXT" aporta a los usuarios una nueva forma de realizar ejercicios en casa donde trabajar diferentes grupos musculares es premiado mediante la adquisición de puntos que permiten llevar a cabo un seguimiento del trabajo realizado por el usuario a través de estadísticas, retos, puntuaciones y beneficios con la finalidad de hacer deporte de forma constante.

Por otra parte, "*RoutineXT*" no pretende ser una aplicación que intente competir con el resto de las aplicaciones disponibles en el mercado. Pretende ofrecer al usuario una manera alternativa y fácil de hacer ejercicio en casa mediante un sistema de gamificación diferente al funcionamiento tradicional que ofrecen las aplicaciones deportivas del mercado. Su finalidad es que el usuario sienta la suficiente motivación de manera que perciba el deporte como una necesidad que se puede realizar en poco tiempo, al ritmo deseado, y sin salir de casa. Por otra parte, "*RoutineXT*" pretende incluir un elemento diferenciador que permita al usuario establecer un horario semanal para lograr adquirir la disciplina de realizar deporte mediante un algoritmo que se encarga de controlar el sistema de puntuaciones.

Otro punto importante es que "*RoutineXT*" también incluye una herramienta para los administradores de manera que puedan realizar tareas de gestión sin necesidad de hacer uso de ningún aspecto del "*backend",* relacionado con la lógica o base de datos.

"*RoutineXT*" está orientado a personas entre 18 y 65 años. Se ha hecho especial énfasis en elaborar una interfaz de usuario moderna, intuitiva y que cumpla con las reglas de "*Shneiderman y Plaisant*" [8].

1.5 Reglas de diseño de "Shneiderman y Plaisant"

Cuando se realiza la maquetación de una página web, podemos encontrar una gran cantidad de componentes que nos facilitan el desarrollo de una interfaz de usuario ("modals", "accordions", "buttons", "cards" etc.), que complementan a los estilos y estructuras diferentes realizados con los lenguajes de programación "*HTML*" y "*CSS*" (biblioteca "*Bootstrap*" [9]). Sin embargo, el buen uso de estos elementos no garantiza que la interfaz de usuario esté bien diseñada. Puede haber interfaces de usuario muy bien trabajadas, pero poco interactivas, con una disposición de sus elementos mal organizados, comportamientos inesperados etc. Lo ideal es que una interfaz de usuario se desarrolle teniendo en cuenta una serie de pautas o consejos que garanticen que la interfaz de usuario esté bien diseñada.

A continuación, vamos a describir brevemente en qué consisten las pautas o reglas de diseño planteadas por *Ben Shneiderman* y *Catherine Plaisant:*

1. Consistencia: para que un usuario pueda completar tareas y objetivos con facilidad, es importante que las acciones y elementos de diseño sean similares entre sí. Por ejemplo, que los elementos de un menú se comporten de la misma forma al pasar el ratón por encima, o que tengan el mismo tamaño de letra, color etc.
2. Usabilidad Universal: la interfaz de usuario debe ser diseñada para cualquier tipo de usuario según su edad, si es principiante o experto, su cultura etc. Por ejemplo, el tamaño de letra o de una imagen debe ser suficientemente grande para no hacer un esfuerzo en poder leer el contenido de la página.
3. Realimentación informativa: el usuario debe saber en todo momento cómo interactuar con la aplicación web mediante mensajes de ayuda, iconos que sean claros, y lo que ocurre en cada etapa de un proceso. Por ejemplo, mostrar el porcentaje de subida de una imagen, acompañar los botones de una imagen que sirva para conocer la función rápidamente etc.
4. Flujo de Tareas: el usuario debe saber en todo momento cuándo se ha iniciado una tarea, y cuándo ha acabado. Por ejemplo, mostrar un mensaje al cerrar la sesión, saber si un mensaje ha sido enviado correctamente a través de un formulario etc.
5. Prevenir errores: el usuario debe saber si comete algún error mediante notificaciones claras y descriptivas, o bien diseñar la aplicación de forma que el usuario no pueda cometer ninguno. Por ejemplo, se puede evitar que un usuario cometa un error de manera que solo pueda seleccionar una cantidad de opciones determinada, o mostrar mensajes de aviso en un formulario.
6. Deshacer tareas: el usuario no debe sentir miedo si realiza una acción que no sabe si puede revertir. Por ejemplo, poder deshacer y volver a rellenar los campos de un formulario sin ningún inconveniente.
7. Control por parte del usuario: el usuario debe sentir que el sistema responde a toda acción que realice. Por ejemplo, actualizar la página cuando el usuario modifica algún dato, o que la respuesta que reciba del sistema sea la esperada.
8. Sin memoria a corto plazo: este principio está relacionada con la consistencia. En este caso, el usuario debe ser capaz de identificar botones si están situados en las mismas posiciones de cada página o, por ejemplo, deducir que un botón de color verde sirve para añadir, y uno de color rojo para eliminar etc.

Capítulo 2. Estudio del Mercado de Aplicaciones deportivas

En este capítulo vamos a realizar un breve análisis de cinco aplicaciones que se utilizan en la actualidad para hacer deporte en casa. Estas aplicaciones han sido buscadas en las tiendas oficiales de *"Android"* y *"IOS"* en la categoría de "Salud y forma física/bienestar". El criterio de selección de estas aplicaciones corresponde a que tienen una valoración igual o superior a 4.5 puntos (de un total de 5), han sido creadas por diferentes desarrolladores, y, además, he tenido la ocasión de probarlas personalmente.

Desglosaremos cada una de sus funcionalidades para tener una idea de lo que este TFT puede aportar al usuario final con un enfoque alternativo a estas aplicaciones.

2.1 Ejercicios en Casa

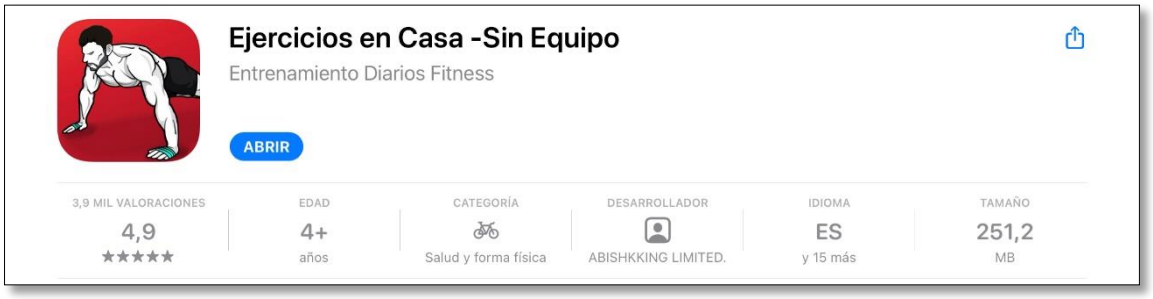


FIGURA 1: LOGOTIPO DE LA APLICACIÓN: "*EJERCICIOS EN CASA*" *[10]*

2.1.1 Características

- Rutinas de calentamientos y estiramientos.
- Registra el proceso de entrenamiento de forma automática.
- Permite personalizar recordatorios para realizar ejercicio.
- Proporciona guías de vídeo y animación.
- Aunque es gratuita, para desbloquear todas las opciones hay que realizar un pago anual.

2.1.2 ¿Cómo funciona?

Al iniciar por primera vez la aplicación tenemos que completar un pequeño formulario de opción múltiple donde debemos aportar información sobre nuestros objetivos. A continuación, la aplicación creará nuestro propio plan de entrenamiento dependiendo de información recopilada a través del formulario realizado. Podremos decidir si queremos realizar ejercicios acerca de nuestro plan de entrenamiento personalizado o bien ejercicios específicos de los siguientes grupos musculares: abdominales, pecho, brazos, piernas, hombros y espalda. Cuando iniciemos el entrenamiento se mostrará la duración, el día actual, el grupo muscular, el tema musical (de libre elección), el tipo entrenador/a (de libre elección), y los ejercicios a realizar junto con su número de repeticiones. En el entrenamiento, nos fijaremos cómo el entrenador realiza los ejercicios. También podemos cambiar de ejercicio en cualquier momento, consultar cómo realizarlo, cambiar el tema musical etc. En determinados momentos, una voz artificial nos informará del tiempo restante para cambiar de ejercicio, e incluso nos intentará animar a seguir realizando ejercicio como si se tratara de un entrenador real. Además, por cada ejercicio realizado

se mostrarán tiempos de descanso antes de pasar al siguiente ejercicio. Una vez finalizado el entrenamiento, se mostrará el tiempo total, las calorías quemadas y podremos programar una alarma para realizar el entrenamiento al día siguiente.

En conclusión, esta aplicación es muy completa, altamente descriptiva y sin una interfaz de usuario sobrecargada. No obstante, las imágenes que se muestran en la aplicación son "ideales", en el sentido, de que se muestran modelos jóvenes con un nivel de musculación avanzado. Esto no es correcto cuando un usuario quiere empezar por primera vez un plan básico o intermedio. Además, en cierta medida puede limitar a un usuario de cierta edad que quiera hacer ejercicio y perciba que los/las modelos jóvenes mostrados en la aplicación son los que pueden tener más facilidad haciendo ejercicio.

2.1.3 Imágenes

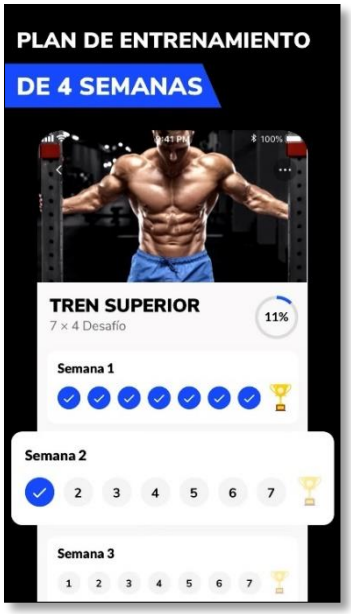


FIGURA 2: PLAN DE ENTRENAMIENTO [10]

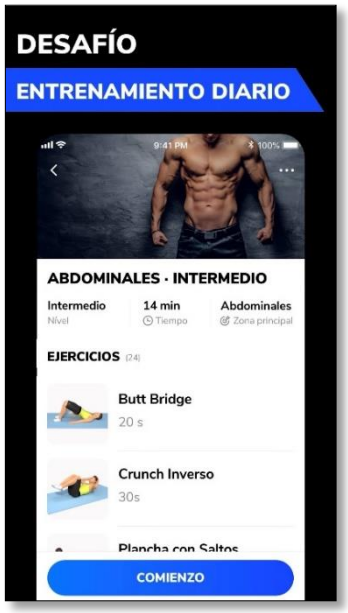


FIGURA 3: EJERCICIOS [10]

FIGURA 4: EJEMPLO DE ENTRENAMIENTO [10]

2.2 Freeletics

FIGURA 5: LOGOTIPO DE LA APLICACIÓN: *"FREELETICS" [11]*

2.2.1 Características

- Permite adaptar cada entrenamiento mediante un entrenador persona con Inteligencia Artificial.
- Permite elegir entre veinte tipos de entrenamientos.
- Puede incluir sesiones de audio para mantenerte motivado.
- Ofrece opciones de nutrición.
- Tiene una amplia variedad de opciones de suscripción, pero no hay ninguna gratuita.

2.2.2 ¿Cómo funciona?

Para utilizar "*Freeletics*" es necesario registrarse con una cuenta de correo y contraseña. Cuando iniciamos sesión por primera vez debemos responder unas preguntas para saber por dónde se debe orientar nuestro entrenamiento. "*Freeletics*" tiene un período de prueba de siete días. Tras adquirir una suscripción, "*Freeletics*" nos proporcionará un asistente con inteligencia artificial que tiene el rol de entrenador personal. A continuación, debemos elegir nuestros propios planes de entrenamiento que son aconsejados a partir de la información recopilada en la encuesta inicial. A medida que completemos los ejercicios, el entrenador ajustará los planes de entrenamiento dependiendo de nuestro rendimiento.

Por otra parte, "*Freeletics*" también proporciona un entrenador dedicado en exclusividad a la alimentación, bienestar mental, calentamientos, enfriamientos, y carreras de velocidad.

En conclusión, esta aplicación es bastante completa porque el usuario siente que tiene un entrenador personal exclusivo. Además, ofrece opciones de alimentación sana entre otras funciones. No obstante, vuelve a "abusar" un poco de las imágenes "ideales" comentadas en la aplicación anterior.

2.2.3 Imágenes

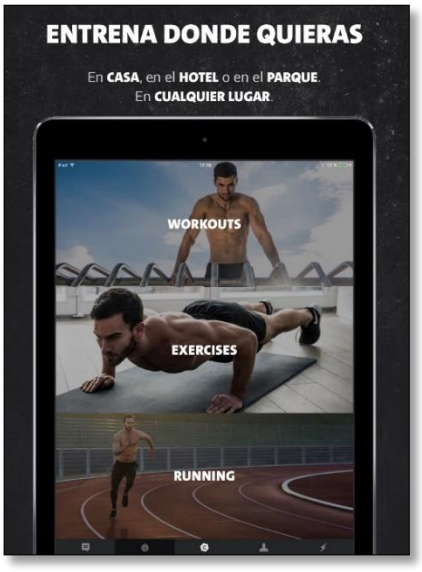


FIGURA 6: TIPOS DE ENTRENAMIENTO *[11]*

FIGURA 7: EJEMPLO DE RUTINA *[11]*

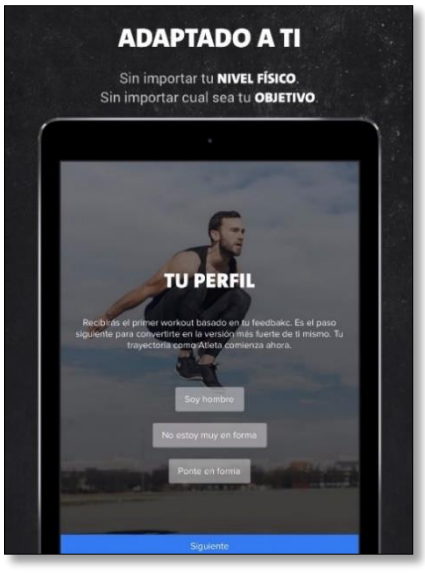


FIGURA 8: PERFIL DE USUARIO *[11]*

2.3 Entrenamientos Diarios Fitness

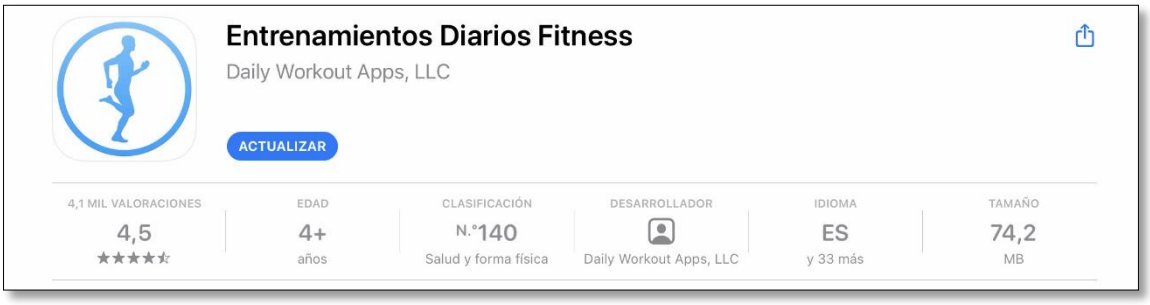


FIGURA 9: LOGOTIPO DE LA APLICACIÓN: *"ENTRENAMIENTOS DIARIOS FITNESS"* *[12]*

2.3.1 Características

- Desarrollado por un entrenador personal certificado.
- Oferta diferentes tipos de ejercicios para cada grupo muscular.
- No se requiere internet.
- Se puede elegir libremente la cantidad de tiempo para realizar ejercicios.
- La oferta gratuita es simple, pero se puede incrementar con una cuenta de pago.

2.3.2 ¿Cómo funciona?

Esta aplicación presenta una interfaz de usuario muy sencilla e intuitiva sin incluir elementos que puedan sobrecargar la misma. El funcionamiento siempre es de la misma forma. Al iniciar la aplicación debemos elegir uno de los siguientes grupos musculares que queramos trabajar: abdominales, brazos, glúteos, cardio, piernas o cuerpo completo. Una vez seleccionada una categoría, tenemos que elegir si queremos estar cinco, ocho o diez minutos haciendo ejercicio, y el tipo de entrenamiento. Cada entrenamiento solo difiere en los tipos de ejercicios que incluyen. Cuando comencemos, se muestra un entrenador o entrenadora realizando el ejercicio actual durante un tiempo determinado. Puede haber tipos de entrenamiento que pertenecen a la versión de pago.

En conclusión, esta aplicación es muy sencilla de utilizar, sin un exceso de publicidad invasiva, aunque con poca variedad de ejercicios en la versión gratuita. Las imágenes que promueve la aplicación son bastante más adecuadas que en las aplicaciones anteriores.

2.3.3 Imágenes

FIGURA 10: MENÚ PRINCIPAL *[12]*

FIGURA 11: EJEMPLO DE ENTRENAMIENTO *[12]*

FIGURA 12: PLAN DE ENTRENAMIENTO *[12]*

2.4 Piernas Fuertes 30 días

FIGURA 13: LOGOTIPO DE LA APLICACIÓN: *"PIERNAS FUERTES 30 DÍAS" [13]*

2.4.1 Características

- Permite tener recordatorios de entrenamiento para realizar ejercicios.
- Los entrenamientos están preparados para hacer ejercicio solo siete minutos al día.
- Ofrece opcionalmente un precalentamiento antes de comenzar el entrenamiento.
- Proporciona una puntuación al realizar cada rutina diaria de modo que para descargar nuevos entrenamientos es necesario alcanzar una puntuación mínima.
- La versión es completamente gratuita.

2.4.2 ¿Cómo funciona?

Esta aplicación está orientada exclusivamente a tonificar las piernas. Al iniciarla por primera vez, debemos completar los datos que nos aparecen para poder crear un plan de entrenamiento en base a la altura, peso, edad, y frecuencia con la que se hace deporte. La singularidad de esta aplicación es que a medida que se van completando ejercicios se otorgan una serie de puntos, de manera que para desbloquear ciertos tipos de planes de entrenamiento se debe obtener un mínimo de puntos en base a los ejercicios que se han realizado.

Además, otra particularidad es que el usuario puede realizar ejercicios de calentamiento y estiramiento nada más comenzar un entrenamiento. El funcionamiento es idéntico a las aplicaciones anteriores.

Esta aplicación es muy interesante porque proporciona elementos diferenciadores como el sistema de puntos, o una imagen resaltada de aquellos músculos que se están trabajando cuando se realiza un determinado tipo de ejercicio. No obstante, la aplicación no está totalmente traducida al español, porque hay ocasiones en las que la voz artificial habla en español y en inglés en un mismo ejercicio.

2.4.3 Imágenes [17]

FIGURA 14: PROGRESO *[13]*

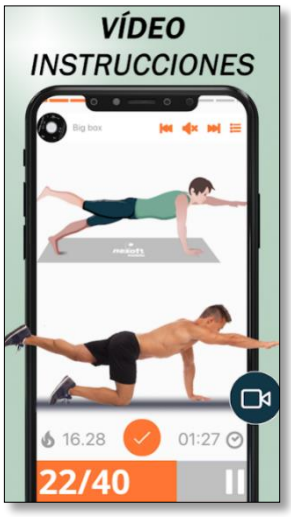


FIGURA 15: ENTRENAMIENTO *[13]*

FIGURA 16: PLANES DE ENTRENAMIENTO *[13]*

2.5 Reto de 30 días – Pectorales

FIGURA 17: LOGOTIPO DE LA APLICACIÓN: *"RETO DE 30 DÍAS" [14]*

2.5.1 Características

- Ofrece tres planes de entrenamiento para trabajar pectorales durante treinta días.
- Los ejercicios son muy variados durante cada día.
- Aumenta gradualmente la intensidad de los ejercicios y los entrenamientos de pecho.
- Las animaciones son de gran ayuda para hacer los ejercicios.
- Aunque la versión gratuita es muy completa, ofrece versiones de pago.

2.5.2 ¿Cómo funciona?

Esta aplicación a diferencia de las anteriores, solo se centra en realizar ejercicios para tonificar el pecho. Tiene tres niveles de dificultad compuestos que abarcan treinta días cada uno. Su funcionamiento es muy simple. Al iniciar la aplicación tan solo debemos elegir un nivel, y pulsar en el día actual. Los días en los que se han completado los ejercicios están marcados para indicar que se han realizado. Cuando vamos a iniciar el entrenamiento, se muestran previamente los ejercicios a realizar, y a continuación, hay que imitar a el ejercicio que está realizando un modelo artificial.

La aplicación tiene una forma muy original de poder acceder al nivel de entrenamiento 2 y 3. El usuario puede acceder pagando una suscripción mensual, o bien con la condición de realizar todos los ejercicios correspondientes al nivel anterior.

En conclusión, esta aplicación es muy sencilla de utilizar y tiene una curiosa forma de pasar de nivel, porque antes que pagar, es preferible forzarse a completar todos los ejercicios del nivel anterior porque, no tendría sentido pasar de nivel si no se han logrado completar todos los ejercicios del nivel anterior. Además, los modelos de entrenadores están hechos a ordenador y no se utiliza ninguna imagen de una persona real que tenga un nivel de musculación avanzado. De hecho, el modelo presentado en cada nivel es el mismo siempre, pero con un tono de color más intenso, para indicar que la actividad tiene un nivel de dificultad más alto.

2.5.3 Imágenes de la aplicación

FIGURA 18: PLAN DE ENTRENAMIENTO *[14]*

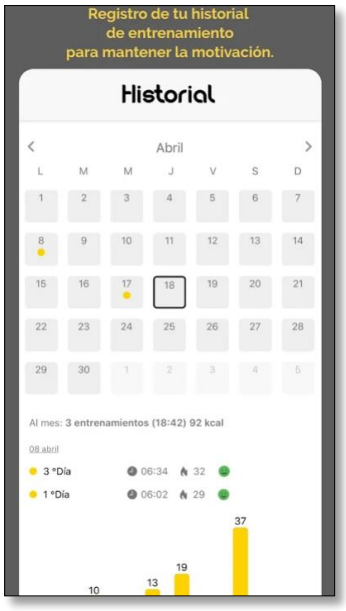


FIGURA 19: HISTORIAL *[14]*

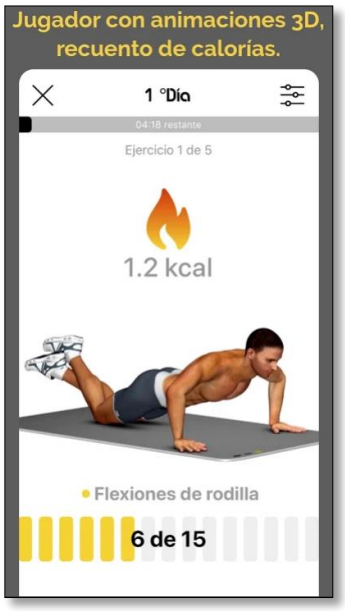


FIGURA 20: ENTRENAMIENTO *[14]*

2.6 Conclusiones

En conclusión, estas aplicaciones ofrecen diferentes formas de hacer ejercicio en casa. Tienen muchos elementos en común, como la posible creación de un plan de entrenamiento, realizar los ejercicios en un tiempo determinado, o incluso la elección de diferentes planes según el estado físico del usuario. Las características que me resultan más importantes para mi TFT corresponden a que un usuario puede ver un historial de los días que he realizado deporte, puede realizar una preparación que al mismo tiempo le sirva de descanso cuando va a realizar otro ejercicio, e incluso que se pueda permitir volver a realizar un ejercicio o pasar al siguiente sin necesidad de hacerlo.

No obstante, haré especial énfasis en buscar ejercicios, donde el/la modelo utilizado no tenga una forma física muy trabajada que pueda de alguna manera limitar al usuario que está haciendo ejercicio o hacerle creer que debe alcanzar un resultado parecido en su forma física.

Capítulo 3. Metodología, planificación y tecnologías

3.1 Metodología

Para el desarrollo de la aplicación de este TFT se ha empleado "*Scrum*", que es una metodología de desarrollo ágil que se basa en completar una serie de funcionalidades o necesidades del proyecto a partir de historias de usuario *[15]*.

Una historia de usuario es un guion que describe una funcionalidad del proyecto con información acerca de qué vamos a hacer y cuántos pasos (tareas) necesitamos completar para poder validar dicha funcionalidad.

"Scrum" ofrece una gran cantidad de ventajas que podemos apreciar a continuación:

1. Permite dividir la amplitud del proyecto en funcionalidades, conformando un contenedor formado por historias de usuarios asignadas con un nivel de prioridad. (pila de producto)
2. Cuando se comienza el proceso de desarrollo, se seleccionan un número de historias de usuario priorizadas con las que nos comprometemos a trabajar durante un período que suele durar entre dos y cuatro semanas de desarrollo (pila de *"Sprint"*). Este período es conocido como *"Sprint"*.
3. A medida que se van completando cada una de las historias de usuario, se obtienen pequeños incrementos siguiendo un proceso iterativo. Este incremento supone que la historia de usuario actual está terminada y probada.
4. Con este sistema, podemos apreciar diariamente el progreso en el desarrollo.
5. Debido a los períodos de tiempo de cada *"Sprint"*, nos aseguramos de realizar una entrega continua y en el tiempo establecido.
6. Debido a la flexibilidad que supone tener el proyecto o parte del proyecto dividido en historias de usuario, se pueden manejar con facilidad requisitos que cambien en el tiempo. Por ejemplo, una funcionalidad que deje de ser necesaria, una nueva que deba añadirse, o una que deje de tener prioridad.
7. El equipo de desarrollo, o en este caso, el desarrollador tiene el control de autoorganizarse en la realización de las tareas.
8. Al ser una metodología de desarrollo ágil, no es necesario una documentación exhaustiva que implique un gran proceso de diseño o análisis que no aporte un valor directo al producto.
9. Permite utilizar gráficas de control del esfuerzo. Por ejemplo, se puede predecir cuándo se completará todo el trabajo mediante un diagrama *"Burn-Down"*, o bien, cuánto trabajo se ha realizado respecto a la cantidad total de trabajo del proyecto mediante un diagrama *"Burn-Up"*.
10. En el caso de que ocurran impedimentos, se suele realizar un "*feedback*", para poder proporcionar una solución en el menor tiempo posible, así como valorar las últimas funcionalidades completadas favoreciendo la motivación del equipo o desarrollador.

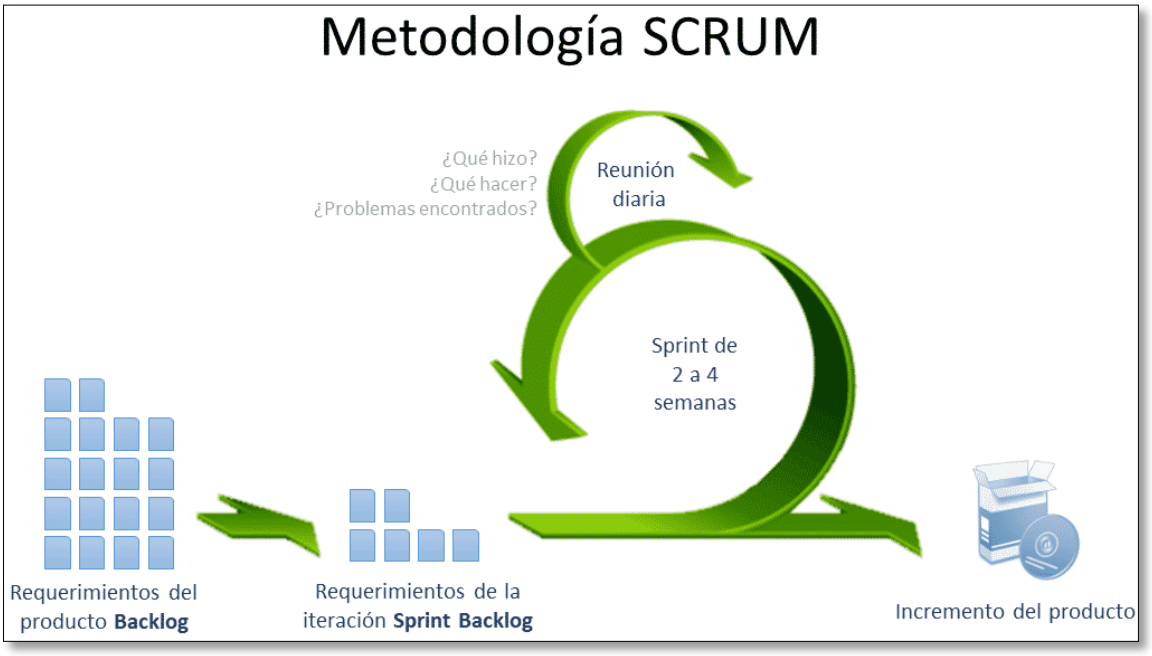


FIGURA 21: METODOLOGÍA *"SCRUM"* *[16]*

3.2 Planificación

"*Scrum*" es una metodología pensada para un equipo de desarrollo. En este TFT, se ha adaptado a un proyecto de forma que las decisiones serán tomadas por una sola persona, el desarrollador. Esto quiere decir, que habrá más flexibilidad en los plazos de duración de cada "*Sprint*", debido a que no habrá opiniones contrastadas para poder calcular los plazos de entrega. Los posibles problemas que puedan surgir en el desarrollo se realizarán en un tiempo añadido al "*Sprint*" actual para que puedan finalizarse las historias de usuario correctamente. Los "*Sprints*" tendrán una duración de dos semanas, de manera que si surgieran problemas, se añadirían dos semanas extras, respetando la duración real de un "*Sprint*" que suele ser entre dos y cuatro semanas.

Para llevar a cabo la planificación del proyecto, se ha divido el desarrollo en tres partes fundamentales correspondientes a diferentes vistas según el tipo de usuario:

- Desarrollo de la vista de usuario no registrado.
- Desarrollo de la vista de cliente.
- Desarrollo de la vista de administrador.

En cada una de estas tres partes se ha procedido de la siguiente forma:

1) **Elaboración del prototipo completo de la fase de desarrollo actual**. Se han desarrollado todas las vistas de cada una de las páginas que puede visitar un determinado tipo de usuario. Esto ha permitido, en primer lugar, obtener un diseño preliminar que ha servido de base para la posterior implementación, y, en segundo lugar, una búsqueda de funcionalidades para identificar las historias de usuario de la pila de producto.

2) **Elaboración de todas las historias de usuario de la fase de desarrollo actual**. Se ha elaborado, a partir de una presentación, historias de usuario que representen el título, la

descripción, la estimación, el tiempo real de desarrollo, la prioridad, el identificador, y los criterios de validación.

3) **Elaboración de la pila de producto**. Se ha elaborado, a partir de la presentación, la pila de producto completa en una hoja de cálculo, incluyendo el identificador, el nombre, la estimación, la duración real, la prioridad, y el número de *"Sprint"*.

4) **Elaboración de la pila de *"Sprint"*.** Se ha incluido, a partir de la hoja de cálculo anterior, aquellas historias de usuario que puedan realizarse en un período de dos semanas con una estimación de 50 horas aproximadamente, ordenadas por orden de prioridad. El orden de prioridad se ha simplificado utilizando la técnica "*MoSCoW*" [17], cuya utilidad reside en priorizar historias de usuario a partir de las siguientes reglas:

   - "MUST": Fundamental y necesaria.
   - "SHOULD": Importante y con valor, pero no es imprescindible.
   - "COULD": tiene cierto interés, podría añadirse si se dispone del tiempo necesario.
   - "WON'T": por lo pronto no se va a trabajar en esto. Si sobra tiempo, se añadirá cuando finalice el desarrollo.

5) **Elaboración del tablón "*Trello*"** [18]**.** Opcionalmente, se ha optado por crear cuatro columnas para poder apreciar gráficamente las historias de usuario de la pila de producto, de la pila de *"Sprint"*, aquellas que se encuentran en desarrollo, y aquellas que han sido completadas.

6) **Implementación de cada historia de usuario de la pila de *"Sprint"*:** durante el desarrollo se desarrollarán aquellas historias de usuario que se encuentren en la pila del "*Sprint*" actual, finalizando su desarrollo cuando se cumplan con sus criterios de validación.

7) **Actualización de la pila de *"Sprint"*, del tablón "*Trello*" y subida de los ficheros modificados a un repositorio de control de versiones.** Por cada historia de usuario se han subido todos aquellos ficheros modificados para realizar una copia de seguridad y para poder utilizar versiones previas del proyecto cuando sea necesario.

Por otra parte, se han realizado dos cursos de formación para poder tener una idea que sirva de base para realizar la implementación. Estos cursos corresponden a los siguientes:

- "Diseño Web Profesional El Curso Completo, Práctico y desde 0" [19].
- "La Web Empieza Aquí: TypeScript, Angular, Storage, Firebase" [20].

La elaboración de la memoria del TFT se ha realizado simultáneamente con la implementación de cada una de las fases que forman parte del desarrollo.

A continuación, se desglosa una tabla que muestra las fases de desarrollo, junto con las tareas y duración estimada:

Fases	Duración Estimada (horas)	Tareas
Estudio previo / Análisis	60	**Tarea 1.1:** Formación. Realización de tutoriales con las diferentes tecnologías que se utilizarán en el desarrollo: funcionamiento de "*Angular*", "*Firebase*" y repaso de tecnologías web.
		Tarea 1.2: Instalación y Preparación del Software Necesario para el Desarrollo del Proyecto: entorno de desarrollo, bibliotecas necesarias, configuración del servidor, creación de cuenta en "*Trello*".
Diseño / Desarrollo / Implementación	185	**Tarea 2.1:** Creación del prototipo. Crear un diseño que muestre una idea de la interfaz que se va a desarrollar incluyendo elementos interactivos que sirvan para tener toda la estructura organizada.
		Tarea 2.2: Planificación de "*Sprints*" *[21]*. Elaboración de la pila de producto, pila de "*Sprint*" e historias de usuario que se van a incluir en una fase concreta del desarrollo, y determinación del tiempo aproximado para el *Sprint* actual.
		Tarea 2.3: Fase 1. Desarrollo de la Vista de Usuario No Registrado. Primera parte donde se desarrollará la página donde el usuario puede consultar, iniciar sesión o registrarse.
		Tarea 2.4: Fase 2. Desarrollo de la Vista de Cliente. Segunda parte donde se desarrollará la página donde el usuario podrá crear su horario, ver su progreso o consultar estadísticas de puntuación.
		Tarea 2.5: Fase 3. Desarrollo de la Vista del Administrador. Tercera parte donde se desarrollará la página en la cual el administrador podrá consultar, eliminar, añadir, editar usuarios, retos o ejercicios.
		Tarea 2.6: Afinar Historias de Usuario Pendientes. Desarrollar ciertas historias de usuario que necesiten la finalización previa de determinadas historias de usuario para poder cumplir con su función. Además, completar otras historias de usuario de baja prioridad siempre que el tiempo lo permita.

Evaluación / Validación / Prueba	30	**Tarea 3.1:** Validación en Conjunto de Todo el Proyecto. Comenzar a verificar individualmente cada una de las partes desarrolladas del proyecto, incluyendo posibles errores que no impliquen la acción directa del usuario como el uso de sesiones o el manejo de la base de datos.
		Tarea 3.2: Medir la Experiencia de Usuario. Probar con diferentes usuarios el uso de la interfaz para medir la complejidad o dificultad de manejo, y obtener una retroalimentación para posibles mejoras que se tengan que llevar a cabo.
Documentación / Presentación	25	**Tarea 4.1:** Elaboración de la Memoria. Desarrollar la memoria del trabajo con todos los detalles necesarios incluyendo secciones de código.
		Tarea 4.2: Recopilación de Fuentes. Elaborar una bibliografía con toda la información de ayuda que ha sido utilizada en el desarrollo del proyecto.

TABLA 1: PLANIFICACIÓN

3.3 Tecnologías

Las herramientas y tecnologías que se han utilizado en este proyecto las podemos categorizar a partir de la siguiente tabla:

Lenguajes de Programación	*HTML, CSS, JavaScript* [22], *TypeScript*
Editor de Código	*Visual Studio Code* [23]
Base de Datos	*Firebase Firestore Database* [24]
Framework de Desarrollo	*Angular (Node.js)* [25]
Edición de Prototipos	*Adobe XD* [26]
Repositorio de Control de Versiones	*Github* [27]
Administración de Proyectos	*Trello, Google Docs, Google Sheets, Google Slides [28].*
Cursos de Formación	*Udemy* [29]
Bibliotecas	*Bootstrap, Ionicons [30] y Fontawesome* [31]

TABLA 2: TECNOLOGÍAS

Capítulo 4. Desarrollo de la Aplicación Web

En este capítulo, vamos a hablar acerca del diseño e implementación de cada una de las tres vistas que conforman cada una de las partes del proyecto: usuario no registrado, cliente y administrador. Estas vistas contienen hasta un máximo de dos *"Sprints"*, dependiendo del número de horas estimado para finalizar las historias de usuario involucradas, y partiendo de la base de que un *"Sprint"* puede tener una duración de entre dos y cuatro semanas.

Al comenzar cada parte del proyecto se ha realizado un prototipo completo de la vista que se va a desarrollar. Por otra parte, se han planificado las historias de usuario, así como la estimación y criterios de validación.

4.1 Tareas previas

Antes de comenzar con el desarrollo del proyecto, se han realizado un par de tareas fundamentales. Por una parte, se ha realizado un curso de formación enfocado en las tecnologías web y, en segundo lugar, la configuración y preparación del proyecto en el editor de código.

4.1.1 Curso de formación

El curso de formación realizado ha tenido una duración aproximada de 60 horas. Ha sido realizado en la plataforma *"Udemy"* para aprender a utilizar las siguientes tecnologías:

- *Visual Studio Code*.
- *Firebase Firestore Database*.
- *Firebase Storage* [32].
- *Angular*.
- *Adobe XD*.
- *TypeScript*.

En el curso de formación se ha comenzado por aprender "*Adobe XD*", seguido de ejemplos resueltos con "*Typescript*", y "*Angular*". Al finalizar se ha realizado un pequeño proyecto web, utilizando todas las herramientas que se han aprendido en el curso, incluyendo la plataforma "*Firebase*".

Este curso de formación ha sido vital para el desarrollo del proyecto, debido a que ha servido para tener una base mínima para comenzar con su implementación.

Adicionalmente, se ha realizado un repaso de los lenguajes de programación web, incluyendo "*HTML*", "*CSS*", "*Javascript*", y la biblioteca de "*Bootstrap*", para poder agilizar el desarrollo. Este repaso se ha llevado a cabo con otro curso de formación aparte. No obstante, debido a que este otro curso presentaba una gran cantidad de contenidos, solo se han realizado los ejemplos relacionados con los lenguajes de programación que hemos comentado.

4.1.2 Configuración y planificación del proyecto

Antes de comenzar con la implementación, se ha creado el proyecto "*RoutineXT-Web"* en "*Visual Studio Code"* incluyendo las bibliotecas y complementos necesarios para el desarrollo ("*Bootstrap"*, "*Fontawesome", "Iconify" y "*Jquery" [33]). Además, debido a la gran flexibilidad que proporciona "*Angular"*, se ha dividido el proyecto en carpetas correspondientes a las vistas o partes de la aplicación: Cada una de estas vistas se encuentran divididas en componentes o páginas web del proyecto. Estos componentes están formados cuatro ficheros específicos:

- Fichero "*HTML*".
- Fichero "*CSS*".
- Fichero de configuración.
- Fichero "*Typescript*".

Es realmente útil tener el proyecto organizado de esta forma, porque si en algún momento se quiere acceder a la hoja de estilos de una página en concreto, se podrá acceder al fichero con formato "*CSS*" que contiene dicha página o componente, sin necesidad de acceder un solo fichero "*CSS*" para todo el proyecto (en este caso sería muy complicado buscar una regla específica). Esto se aplica también a los ficheros "*HTML*", e incluso a los ficheros "Typescript" donde se maneja toda la lógica del componente.

Además, a partir del prototipo realizado, se han diseñado las historias de usuario para poder acotar la duración de los "*Sprints*" que serán necesarios para completar todas funcionalidades de la vista con la que se esté trabajando. Las historias de usuario han sido realizadas en "*Google Slides"*.

La pila de producto se ha realizado a partir de las historias de usuario en una hoja de cálculo en "*Google Sheets"*. Todas las historias de usuario se han ido añadiendo a la pila de producto a medida que se ha ido avanzando en el proyecto. Por otra parte, la pila de "*Sprint*" contiene un subconjunto de historias de usuario de la pila de producto que se van a trabajar en la iteración actual.

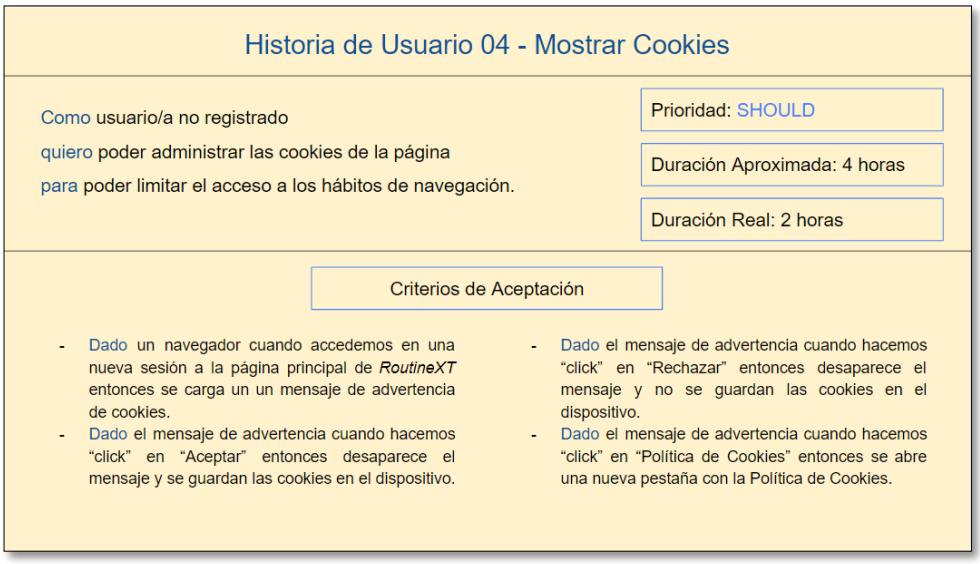
Opcionalmente, también he creado una gráfica donde se compara la estimación de tiempo de cada historia de usuario con su duración real para tener una idea visual de la diferencia que se ha dado, con la intención de mejorar las estimaciones de las siguientes historias de usuario.

Por otra parte, se ha utilizado la herramienta "*Trello",* donde se han creado cuatro columnas: pila de producto, la pila de "*Sprint*", historias de usuario en proceso e historias de usuario finalizadas. Esta herramienta es de gran utilidad porque permite entender el estado actual de desarrollo.

El siguiente paso ha sido crear un repositorio en "*Github"* para poder tener el proyecto almacenado y dividido en "*commits"*, que además sirven como copia de seguridad. He creado la rama "*master"*, y la rama "*develop"* donde se almacenarán todos los incrementos del proyecto. Debido a que el proyecto es individual, no he considerado necesario crear ramas adicionales.

Para finalizar, también he tenido en cuenta la planificación de las copias de seguridad de todo el proyecto. He realizado tres copias de seguridad cada vez que se ha completado un elevado número de funcionalidades cada aproximadamente dos semanas. Estas copias están guardadas en medios de almacenamiento diferentes: una copia local en el mismo equipo que se utiliza para el desarrollo, una copia en un medio de almacenamiento aparte, y una copia en la nube mediante *"Google Drive"*.

Cada copia de seguridad contiene: el proyecto, la memoria, los prototipos, la planificación e historias de usuario, y las imágenes. De igual forma, en el navegador, se han guardado las fuentes bibliográficas que han sido necesarias para el desarrollo del proyecto.



Historia de Usuario 04 - Mostrar Cookies

Como usuario/a no registrado
quiero poder administrar las cookies de la página
para poder limitar el acceso a los hábitos de navegación.

Prioridad: SHOULD
Duración Aproximada: 4 horas
Duración Real: 2 horas

Criterios de Aceptación

- Dado un navegador cuando accedemos en una nueva sesión a la página principal de *RoutineXT* entonces se carga un un mensaje de advertencia de cookies.
- Dado el mensaje de advertencia cuando hacemos "click" en "Aceptar" entonces desaparece el mensaje y se guardan las cookies en el dispositivo.
- Dado el mensaje de advertencia cuando hacemos "click" en "Rechazar" entonces desaparece el mensaje y no se guardan las cookies en el dispositivo.
- Dado el mensaje de advertencia cuando hacemos "click" en "Política de Cookies" entonces se abre una nueva pestaña con la Política de Cookies.

FIGURA 22: EJEMPLO DE HISTORIA DE USUARIO REALIZADA CON *"GOOGLE SLIDES"*

	A	B	C	D	E	F	G
1	ID	Historia de Usuario	Nombre	Duración Aproximada (horas)	Duración Real (horas)	Prioridad	Nº. de Sprint
2	1	HU-01	Acceder a la Página Principal	7	9	MUST	1
3	2	HU-02	Visualizar el Vídeo de Demostración	3		WON'T	1
4	3	HU-03	Acceder a las Redes Sociales	3	3	COULD	1
5	4	HU-04	Mostrar Cookies	4	2	SHOULD	1
6	5	HU-05	Acceder a Inicio de Sesión	5	9	MUST	1
7	6	HU-06	Restablecimiento de la Contraseña	2	2	WON'T	1
8	7	HU-07	Google reCAPTCHA	3		WON'T	1
9	8	HU-08	Registro de Usuario	5	9,5	MUST	1
10	9	HU-09	Acceder a Preguntas Frecuentes	6	6	MUST	1
11	10	HU-10	Contactar con la Empresa	5	8	MUST	1
12	11	HU-11	Política de Cookies	2	4	SHOULD	1
13	12	HU-12	Política de Privacidad	2	0,5	SHOULD	1
14	13	HU-13	Aviso Legal	2	0,5	SHOULD	1
15	14	HU-14	Página No Encontrada	2	2	MUST	1

FIGURA 23: EJEMPLO DE PILA DE PRODUCTO REALIZADA CON *"GOOGLE SHEETS"*

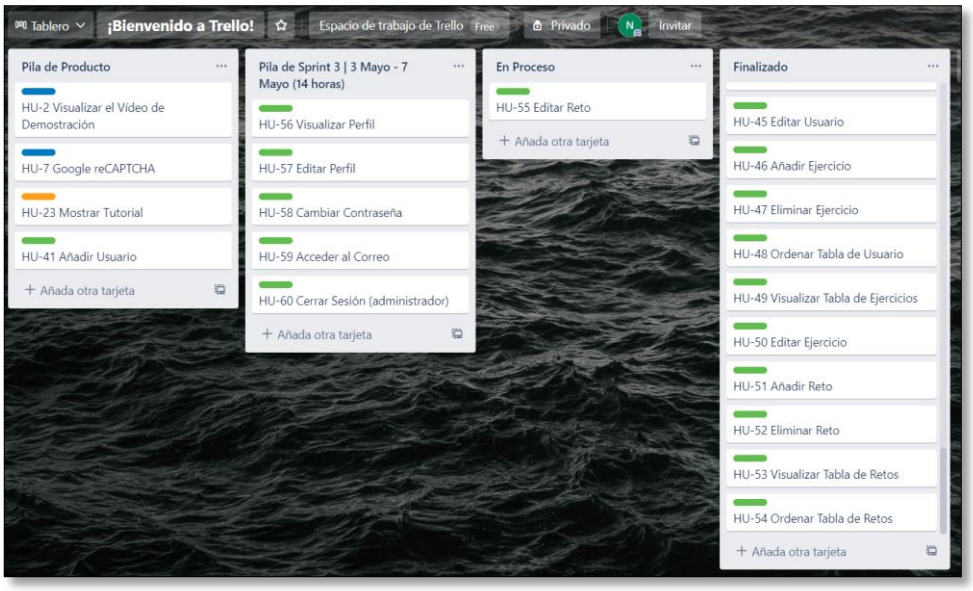


FIGURA 24: TABLÓN *"TRELLO"*

4.2 Historias de usuario

En este apartado vamos a presentar un catálogo que contiene todas las historias de usuario y "*Sprints*" que se han realizado a lo largo del desarrollo del proyecto, incluyendo aquellas historias de usuario que no han podido ser implementadas.

Historia de Usuario	Nombre	Duración Aproximada (horas)	Duración Real (horas)	Prioridad	N.º de Sprint
HU-01	Acceder a la Página Principal	7	9	MUST	1
HU-02	Visualizar el Vídeo de Demostración	4	4	WON'T	1
HU-03	Acceder a las Redes Sociales	3	3	COULD	1
HU-04	Mostrar Cookies	4	2	SHOULD	1
HU-05	Acceder a Inicio de Sesión	5	9	MUST	1
HU-06	Restablecimiento de la Contraseña	2	2	WON'T	1
HU-07	Google reCAPTCHA	3	(Sin implementar)	WON'T	1
HU-08	Registro de Usuario	5	9,5	MUST	1

Historia de Usuario	Nombre	Duración Aproximada (horas)	Duración Real (horas)	Prioridad	N.º de Sprint
HU-09	Acceder a Preguntas Frecuentes	6	6	MUST	1
HU-10	Contactar con la Empresa	5	8	MUST	1
HU-11	Política de Cookies	2	4	SHOULD	1
HU-12	Política de Privacidad	2	0,5	SHOULD	1
HU-13	Aviso Legal	2	0,5	SHOULD	1
HU-14	Página No Encontrada	2	2	MUST	1
HU-15	Mostrar Imagen de Carga al Iniciar Sesión	1	1	SHOULD	2
HU-16	Mostrar Fecha del Plan de Entrenamiento	5	7	MUST	2
HU-17	Visualizar Plan de Entrenamiento	6	25	MUST	2
HU-18	Añadir Rutina	6	8	MUST	2
HU-19	Eliminar Rutina	4	3	MUST	2
HU-20	Editar Rutina	4	2	MUST	2
HU-21	Cancelar Plan de Entrenamiento	3	1	MUST	2
HU-22	Imprimir Plan de Entrenamiento	4	1,5	COULD	2
HU-23	Mostrar Tutorial	4	5	WON'T	3
HU-24	Mostrar Ejercicios	6	11	MUST	2
HU-25	Imprimir Ejercicios	4	0,5	COULD	3
HU-26	Mostrar Retos	5	7	MUST	3
HU-27	Imprimir Retos	4	3	COULD	3
HU-28	Mostrar Puntuación Actual	1	0,5	MUST	3

Historia de Usuario	Nombre	Duración Aproximada (horas)	Duración Real (horas)	Prioridad	N.º de Sprint
HU-29	Mostrar Rutinas Completadas	0,5	0,5	MUST	3
HU-30	Mostrar Ejercicios Realizados	0,5	0,5	MUST	3
HU-31	Mostrar Retos Realizados	0,5	0,5	MUST	3
HU-32	Mostrar Máxima Puntuación	0,5	0,5	COULD	3
HU-33	Mostrar Estadísticas de la Semana	2	4	SHOULD	2
HU-34	Mostrar Ranking de Usuarios	6	2,5	MUST	2
HU-35	Visualizar Perfil	6	6	MUST	2
HU-36	Editar Perfil	6	3	MUST	2
HU-37	Cambiar Contraseña	0,5	0,5	WON'T	2
HU-38	Mostrar Consejos de Entrenamiento	6	6	MUST	3
HU-39	Cerrar Sesión	1	1	MUST	2
HU-40	Mostrar Imagen de Carga al Iniciar Sesión (administrador)	0,5	0,5	SHOULD	4
HU-41	Añadir Usuario	1,5	(Sin implementar)	MUST	4
HU-42	Eliminar Usuario	1,5	1,5	MUST	4
HU-43	Ordenar Tabla de Usuario	1,5	0,5	SHOULD	4
HU-44	Visualizar Tabla de Usuario	3	3	MUST	4
HU-45	Editar Usuario	1,5	2	MUST	4
HU-46	Añadir Ejercicio	1,5	2	MUST	4
HU-47	Eliminar Ejercicio	1,5	1,5	MUST	4

Historia de Usuario	Nombre	Duración Aproximada (horas)	Duración Real (horas)	Prioridad	N.º de Sprint
HU-48	Ordenar Tabla de Ejercicios	1,5	1	SHOULD	4
HU-49	Visualizar Tabla de Ejercicios	3	3	MUST	4
HU-50	Editar Ejercicio	1,5	1,5	MUST	4
HU-51	Añadir Reto	1,5	2	MUST	4
HU-52	Eliminar Reto	1,5	1,5	MUST	4
HU-53	Visualizar Tabla de Retos	3	3	MUST	4
HU-54	Ordenar Tabla de Retos	1,5	1,5	SHOULD	4
HU-55	Editar Reto	1,5	1,5	MUST	4
HU-56	Visualizar Perfil	0,5	1	MUST	4
HU-57	Editar Perfil	1	2,5	MUST	4
HU-58	Cambiar Contraseña	0,5	0,5	WON'T	4
HU-59	Acceder al Correo	0,5	0,5	COULD	4
HU-60	Cerrar Sesión (administrador)	0,5	0,5	MUST	4
HU-61	Confirmar Reto	2	2	MUST	5
HU-62	Visualizar Rutina	6	6	MUST	5
HU-63	Comenzar Rutina	8	9	MUST	5
HU-64	Cambiar de Ejercicio	12	12	MUST	5
HU-65	Finalizar Rutina	6	6	MUST	5
HU-66	Visualizar Información de la Rutina Realizada	4	4	MUST	5
		207 horas en total	**230 horas en total**		

TABLA 3: CATÁLOGO DE HISTORIAS DE USUARIO

Como podemos observar, para completar el proyecto se han realizado 66 historias de usuario de un total de 68. La historia de usuario número "07" no se ha podido implementar porque al no ser prioritaria se dejó para el final del desarrollo siempre que diera tiempo de ser implementada. Sin

embargo, la historia de usuario número 41, no pudo ser implementada a pesar de tener una alta prioridad (MUST), debido a que cuando se ejecuta la función predeterminada que permite añadir un nuevo usuario a la base de datos ("*createUserWithEmailAndPassword*") siempre cierra la sesión del usuario actual e inicia la del nuevo usuario que ha sido creado. Por lo tanto, esto supondría que cada vez que el administrador añade un nuevo usuario, automáticamente se cierra su sesión. Por esta razón, se ha decidido no implementar esta historia de usuario.

En la siguiente tabla podemos observar cómo han sido organizadas las historias de usuario en cada "*Sprint*":

Número de Sprint	Vista de usuario	Historias de Usuario	Historias de Usuario implementadas
Sprint 1	Usuario/a No Registrado/a	14	01-14
Sprint 2	Cliente/a	8	15-22
Sprint 3	Cliente/a	17	23-39
Sprint 4	Administrador/a	21	40-60
Sprint 5	Cliente/a	6	61-66

TABLA 4: ORGANIZACIÓN DE *"SPRINTS"*

Estos "*Sprints*" se han organizado de tal forma que abarquen aproximadamente dos semanas de desarrollo gracias a la duración estimada de sus historias de usuario. No obstante, hay ocasiones en que han llegado a durar más tiempo hasta un máximo de cuatro semanas. Gracias al prototipo realizado en cada "*Sprint*", se han determinado las funcionalidades que puede realizar un usuario en una vista determinada, clasificándolas en historias de usuario que forman parte de un determinado "*Sprint*".

En la elaboración de las historias de usuario se ha percibido que algunas abarcan funcionalidades muy similares. Esta similitud se ha reflejado mediante el uso de historias épicas que han permitido dividir estas historias con varias funcionalidades generales en historias de usuario específicas para que puedan ser implementadas (políticas de la web, gestión de usuarios, de ejercicios etc.). A continuación, podemos observar la tabla 5 que nos indica qué historias de usuario forman parte de las historias de usuario épicas:

Número de Sprint	Historias de Usuario Épicas	Nombre de Historia de Usuario Épica	Historias de Usuario que forman parte de una Épica
Sprint 1	2	1. Gestionar Autenticación. 2. Políticas de la Web.	1. 05, 06, 08 2. 11, 12, 13
Sprint 2 y 3	3	1. Gestionar Plan de Entrenamiento. 2. Imprimir Datos de Entrenamiento. 3. Mostrar Logros.	1. 16, 17, 18, 19, 20, 21. 2. 22, 25, 27. 3. 28, 29, 30, 31, 32.
Sprint 4	3	1. Gestión de Usuarios. 2. Gestión de Ejercicios. 3. Gestión de Retos.	1. 41, 42, 43, 44, 45. 2. 46, 47, 48, 49, 50. 3. 51, 52, 53, 54, 55.
Sprint 5	1	Gestionar Rutina Programada	63, 64, 65

TABLA 5: HISTORIAS DE USUARIO ÉPICAS

Antes de comenzar a explicar el desarrollo de la aplicación, vamos a describir qué decisiones han sido tomadas en cuenta para elaborar la interfaz de usuario tanto a nivel de prototipado como en la vista de usuario final que se ha implementado acorde a las reglas de "*Shneiderman y Plaisant*" que hemos comentado anteriormente.

4.3 Decisiones acerca de la interfaz de usuario

Cuando se realiza una aplicación web normalmente es necesario utilizar tres tipos de perfiles: la persona encargada de diseñar la apariencia de la aplicación, la persona encargada del desarrollo, y una persona encargada de realizar tareas de "marketing".

En este trabajo no se han realizado tareas de "marketing" porque es un aspecto que no se abarca en el ámbito de este TFT. Sin embargo, sí se ha hecho uso de las tareas de diseño y de desarrollo.

Como hemos comentado anteriormente, antes de comenzar cada "*Sprint*" se ha realizado una tarea que consiste en hacer un prototipo con "Adobe XD" de toda la vista que corresponde al "*Sprint*" en el que se va a trabajar. Este prototipo permite diseñar una interfaz de usuario preliminar que tiene dos funciones:

- Establecer la posible apariencia e interacción de la interfaz de usuario de la vista actual, de manera que servirá como un guion para que apenas se tengan que tomar decisiones en el desarrollo y se pierda tiempo.
- Identificar las historias de usuario con facilidad al tener una muestra de la interfaz de usuario de la aplicación.

Este prototipo es muy básico debido a que no aporta un valor directo al producto. Esto permite que se realice en poco tiempo y que el diseño de la interfaz de usuario final de la aplicación siempre sea superior respecto al prototipo inicial.

4.3.1 Diseño de la vista de usuario no registrado

En esta vista se han utilizado fundamentalmente dos tonalidades de colores: rojo y negro. La idea es que un usuario aprecie un diseño formal y sencillo que le transmita seriedad y seguridad. Esta vista no contiene una gran cantidad de elementos, para que el usuario solo se centre en los aspectos fundamentales que intenta demostrar la aplicación.

La estructura se basa en tres partes:

- Crear una cabecera formada por un color de fondo, el logotipo de "RoutineXT" y un botón que le permita iniciar sesión.
- La sección principal con un color de fondo negro donde se muestre el contenido de la página.
- Un pie de página que es exactamente igual en todas las páginas de la vista (opciones para acceder a las redes sociales, y al resto de páginas de la vista actual).

La idea es que el usuario tan solo visualice la información, y navegue por el resto de las páginas a través de los enlaces del pie de página.

4.3.2 Diseño de la vista de cliente

Esta vista es bastante más compleja que la anterior a nivel de maquetación. El objetivo principal no es transmitir un diseño serio y formal, sino mostrar una interfaz amigable con colores más vivos para fomentar la interacción del usuario.

La estructura principal se basa en presentar al usuario un "*dashboard*" [34], formado por un menú vertical con un máximo de seis enlaces. A continuación, en la barra superior podemos apreciar el logo de "RoutineXT", y una fila en blanco que siempre mostrará un título breve acerca de la funcionalidad que muestra una página concreta del "*dashboard*". En la parte derecha encontraremos la imagen de perfil del usuario (una por defecto), y una lista desplegable que le permitirá acceder a su perfil o cerrar su sesión.

El contenido del "*dashboard*" es lo único que cambiará en la interfaz de usuario, dependiendo de la página en la que se encuentre el usuario. Para diferenciarse de la barra superior, se ha añadido una tonalidad mínimamente gris para mostrar el contenido.

Dependiendo del contenido, la estructura será similar, de forma que en la parte superior se muestre un título que permita indicar alguna información relevante del contenido y en la derecha un botón que realice una determinada funcionalidad. Debajo se mostrarán elementos donde la elección de estilos nunca difiera en gran medida de los estilos por los que están formados los elementos fijos del "*dashboard*".

Se ha intentado utilizar la mayor cantidad de elementos posibles en cada página. Por ejemplo, en el plan de entrenamiento, se muestra un horario semanal construido con "*cards*" formados por un fondo blanco que fomenta que el usuario haga "*click*" en cada uno. Las rutinas añadidas difieren en el color de fondo dependiendo del tipo. Las "*cards*" que no se pueden modificar tienen una opacidad que informa al usuario no podrá interactuar con ellas.

En el caso de los ejercicios, siempre se muestra la misma estructura: un título que nos indica la rutina en color negro y, a continuación, un contenedor con un fondo blanco donde se muestra una foto de cada ejercicio con su nombre.

En el caso de los retos, encontramos una tabla con diferente disposición de colores que alertan al usuario sobre los retos que aún no puede realizar.

Los consejos se muestran mediante frases que podemos visualizar en un "*carousel*", como si fueran diapositivas que van cambiando con una animación.

Cada página de la vista contiene diferentes elementos para que el usuario pueda interactuar de la mejor forma posible. La facilidad de uso ha sido muy trabajada, así como que la interfaz sea amigable y cercana al usuario registrado.

4.3.3 Diseño de la vista de administrador

Este diseño es muy similar a la vista anterior, con la diferencia que se han escogido otro tipo de tonalidades y estilos para diferenciar el tipo de usuario que accede a esta vista. Estas tonalidades son más serias y muestran un diseño más continuista, debido a que las páginas están formadas fundamentalmente por tablas con datos.

Los elementos fijos del "*dashboard*" son exactamente iguales, se presenta un menú, el logotipo de "*RoutineXT*", un título que nos indica la página donde nos encontramos y una imagen de perfil del usuario administrador.

El contenido se basa en añadir botones con diferentes tonalidades dependiendo de su función:

- Tonalidad Roja: eliminar.
- Tonalidad Verde: añadir.
- Tonalidad Azul: editar.
- Tonalidad Naranja: filtrar u ordenar.

Los botones con tonalidad roja y azul están dispuestos en cada una de las filas de las tablas de datos de ciertas páginas a la que puede acceder el administrador.

A continuación, vamos a explicar el desarrollo a nivel de maquetación y lógica de cada una de las historias de usuario que se han implementado en cada página que forman parte de cada "*Sprint*". Antes de comenzar con la explicación de cada vista de usuario, se mostrará una tabla con información acerca de la misma, incluyendo las historias de usuario que se han implementado en cada una.

4.4 Vista de usuario no registrado

La vista de usuario no registrado correspondería al perfil de alguien que visita la página web y no se encuentra registrado en el sistema. Esta persona buscará información general acerca de la web de "*RoutineXT*"; preguntas frecuentes, políticas de *cookies*, redes sociales, o cómo registrarse en la página, entre otras funciones.

Páginas Web	Tipo de usuario	Sprint	Historias de Usuario
-Página Principal. -Página de Inicio de Sesión. -Página de Registro. -Página de Cambio de Contraseña. -FAQs. -Política de Cookies. -Aviso Legal. -Política de Privacidad. -Página No Encontrada (Error 404).	No Registrado	1	14 HU (2 épicas) (01-14)
Duración y fecha Estimada (inicial)	**Duración y fecha Real**		
2 semanas (22 Febrero - 5 Marzo) ~ 51 horas	3 semanas (22 Febrero - 12 Marzo) = 55,5 horas		

TABLA 6: PLANIFICACIÓN DE LA VISTA DE USUARIO NO REGISTRADO

En cada una de las páginas que forman parte de la aplicación, vamos a detallar los aspectos más relevantes del código tanto en la maquetación ("*front-end*"), como en la lógica ("*back-end*"). En la maquetación se explica el contenido de los ficheros "*HTML*" y "*CSS*", mientras que, en la lógica, se detalla el contenido del fichero "*Typescript*".

4.4.1 'Página Principal'

En la página principal se ha desarrollado la cabecera y el pie de página para utilizarlos de base en el resto de componentes que forman parte de la vista de usuario no registrado. La página principal corresponde al componente *"home"*.

Maquetación

- Sección "header": está formado por un elemento "*nav*", donde podemos encontrar un menú formado por el logotipo de "*RoutineXT*", y un botón para iniciar sesión en el sistema. Si el usuario tiene su sesión iniciada, podrá acceder directamente a la vista de usuario registrado.
- Sección "description": podemos encontrar una sección donde se encuentra un mensaje de bienvenida en un encabezado seguido de un vídeo, presentando brevemente el funcionamiento de la aplicación.
- Sección "articles": podemos encontrar cuatro "*articles*" dispuestos en un elemento "*row*" donde mostramos encabezados e imágenes para informar de las posibilidades que ofrece la aplicación.
- Sección "footer": en un elemto "*row*" se muestra: información para acceder a las redes sociales (Facebook, Instagram y Twitter), políticas de cookies, de privacidad, aviso legal, preguntas frecuentes y contacto.

- Sección de "cookies": mensaje donde se permite con dos botones aceptar o rechazar las cookies de la página web.

Lógica

Todos los ficheros con extensión "*.ts*" de cada componente verifican si el usuario está autenticado mediante la suscripción al servicio "*AuthService*". Este servicio ha sido creado en la carpeta "*services*" y contiene cuatro métodos relacionados con las sesiones:

- Método "login": permite iniciar la sesión con email y contraseña si el usuario existe.
- Método "register": permite crear un usuario con email y contraseña si el usuario no existe.
- Método "logout": permite cerrar la sesión de un usuario registrado.
- Método "resetPassword": permite modificar la contraseña de un usuario.

Este fichero siempre comprueba el ciclo de vida de la cookie de "*RoutineXT*" (en este caso, equivale a 72 horas). En el caso que no exista, estos métodos permitirán crearla o no dependiendo de la acción que tome el usuario:

- Método "rejectCookies": permite no almacenar la "*cookie*" en el navegador en el caso que el usuario pulse la opción "Rechazar".
- Método "acceptCookies": permite almacenar la "*cookie*" en el navegador durante 72 horas en el caso que el usuario pulse la opción "Aceptar".

```
37    ngOnInit(): void {
38      this.cookie_value = this.cookieSvc.get('RoutineXT_Cookie')
39      this.showCookieMessage = this.cookie_value != '' ? false : true;
40      console.log(this.showCookieMessage)
41    }
```

FIGURA 25: FRAGMENTO DE CÓDIGO PARA MOSTRAR LA *"COOKIE"*

4.4.2 'Aviso Legal', 'Política de Cookies' y 'Política de Privacidad'.

Estas páginas se han incluido al igual que el mensaje de aviso de cookies debido a que es un aspecto obligatorio hoy en día. He considerado que debe incluirse para poder hacer este proyecto lo más realista posible. El desarrollo de estas páginas es muy similar entre sí.

La generación del texto legal de cada página ha sido realizada con una herramienta online detallada en la bibliografía.

Maquetación

- Sección "header".
- Sección "title": esta sección muestra mediante un título el contenido que se puede visualizar en la página.

- Sección "article": el texto legal se ha escrito en un contenedor de artículos, distribuido en diferentes párrafos. Las partes que han sido destacadas contienen un elemento *"span"* para poder resaltar el contenido.
- Sección "footer".

Lógica

Este fichero no contiene ningún método. Tan solo se realiza una comprobación para determinar si se debe mostrar el botón para que el usuario inicie sesión dependiendo si está autenticado mediante el servicio *"AuthService"*.

FIGURA 26: MUESTRA DE LA PÁGINA PRINCIPAL

FIGURA 27: MUESTRA DE LA PÁGINA DE *"POLÍTICA DE COOKIES"*

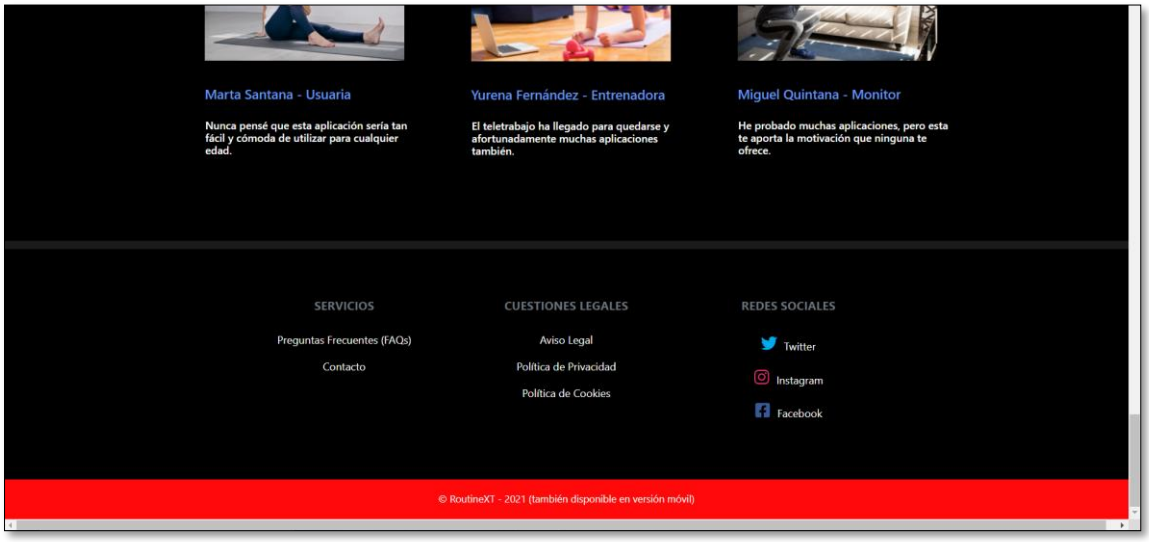


FIGURA 28: MUESTRA DEL *"FOOTER"*

4.4.3 'Preguntas Frecuentes' y 'Contacto'

La página de "Contacto" realmente es una sección que se encuentra a continuación de la página de "Preguntas Frecuentes". Las "FAQs" sirven para que el usuario pueda resolver sus dudas a partir de una serie de preguntas respondidas, mientras que la sección de "Contacto" consiste en un formulario para que el usuario pueda plantear dudas o sugerencias dirigidas al administrador/a de "*RoutineXT*".

Maquetación

- Sección "header".
- Sección "title": esta sección simplemente muestra mediante un título el contenido que se puede visualizar en la página.
- Sección "accordion": mediante un componente de *Bootstrap* denominado "*accordion*", se pueden enseñar las respuestas de las preguntas mostrando su contenido oculto al pulsar en cada.
- Sección "contact": en esta sección se utiliza un formulario que hace uso de los elementos "*formGroup*" y "*formControlName*" que sirven para validar los valores introducidos en el formulario desde el fichero "*Typescript*". Si el usuario introduce algún campo incorrecto respecto a su formato o necesario requerimiento, se mostrará un mensaje de error. Sólo cuando el formulario sea correcto se habilitará el botón "Enviar" a partir del contenido de la variable "*registerForm.valid*". El formulario está compuesto por un nombre (especificado como un elemento "*input*"), un email (especificado también como un elemento "*input*") y un mensaje (como un elemento "*textarea*").
- Sección "footer".

Lógica

En el constructor de la clase declaramos el servicio correspondiente a "*FormBuilder*", que permite crear formularios con capacidad de poder ser validados a partir de unas reglas. En el caso del formulario de contacto, se ha determinado que el nombre y el contenido del mensaje sean obligatorios, mientras que el email cumpla con el formato dado por una expresión regular. Una vez que el usuario envíe el contenido, se mostrará un mensaje breve de éxito, borrando los datos del formulario o mostrando un mensaje de error si se diera el caso. Este mensaje está implementado a partir del módulo "*sweetalert2*" *[35]*.

Para que los mensajes puedan ser recibidos por el correo del administrador, se ha creado una cuenta de correo real denominada: "*routineXT_adm@outlook.com*". A continuación, se ha hecho uso de la herramienta online "*emailjs.com*" [36] que permite llevar a cabo el envío de mensajes sin necesidad de instalar un servidor de correo. En el fichero, tan solo hay que escribir una línea de código que permite, mediante un identificador único, enviar un correo a la dirección de correo electrónico configurada.

```
68    const templateParams = {
69      name: this.registerForm.get('name').value,
70      from_name: this.registerForm.get('email').value,
71      message: this.registerForm.get('message').value,
72      reply_to: this.registerForm.get('email').value
73    };
74
75    emailjs.send('service_1u7l7cw','template_rl1ubl5', templateParams
76      .then((result: EmailJSResponseStatus) => {
77        this.registerForm.reset()
78        Toast.fire({
79          icon: 'success',
80          title: 'Hemos recibido su mensaje correctamente'
81        })
82      }, (error) => {
83        Toast.fire({
84          icon: 'error',
85          title: '¡Oops, no hemos podido recibir su mensaje!'
86        })
87      });
```

FIGURA 29: FRAGMENTO DE CÓDIGO PARA ENVIAR CORREOS

4.4.4 'Inicio de Sesión'

Maquetación

- Sección "header".
- Sección "form": en esta sección podemos apreciar un formulario que también ha sido implementado haciendo uso de los componentes "*formGroup*" y "*formControlName*". En el formulario podemos observar dos elementos "*inputs*" con el email y la contraseña de usuario registrado. En el caso de que alguno de los campos sea erróneo, se borrarán los valores introducidos y se mostrará un mensaje de advertencia. Por otra parte, existe la opción de pinchar en un enlace mediante la propiedad "*routerLink*" para poder restablecer la contraseña en caso de haberla olvidado mediante la página de "Cambio de Contraseña". Además, también existe un botón para poder crear una cuenta en la página de "Registro".
  Para iniciar sesión será necesario introducir los datos del formulario correctamente para que pueda ser habilitado el botón "Entrar".

- Sección "footer".

```
<form class="form" [formGroup]="loginForm" (ngSubmit)="onLogin()">
<h1 class="form_title">Inicio de Sesión</h1>

<input type="email" placeholder="Email" formControlName="email">
<div *ngIf="loginForm.controls['email'].invalid && loginForm.controls['email'].dirty" class="alert alert-danger" role="alert">
  <span *ngIf="loginForm.controls['email'].errors.required">El correo es un campo obligatorio</span>
  <span *ngIf="loginForm.controls['email'].errors.pattern">El correo no es válido</span>
</div>
```

FIGURA 30: FRAGMENTO DE CÓDIGO PARA MOSTRAR EL FORMULARIO

Lógica

En el constructor de la clase vamos a utilizar tres servicios: *"AuthService"*, *"NgxSpinnerService"*, *"FormBuilder"*.

El formulario de inicio de sesión se valida a partir de las reglas introducidas gracias al servicio *"FormBuilder"* verificando el email a partir de una expresión regular y la contraseña con el servicio "AuthService".

El fichero está conformado por las siguientes funciones:

- Método "onLogin": esta función sirve para verificar si el email y la contraseña corresponden a los datos almacenados mediante el mecanismo de autenticación facilitado por *"Firebase"*. En el caso de que el usuario exista, se llama a la función *"showLoading"* que mostrará una breve animación para redirigir al usuario a su página de entrenamiento. Si el usuario que inicia sesión es el administrador, se llamará a la función *"showAdminLoading"*, que le redirigirá a la tabla de usuarios del sistema.
- Método "showLoading": esta función mostrará una breve animación para redirigir al usuario a su página de entrenamiento.
- Método "showAdminLoading": esta función mostrará una breve animación para redirigir al administrador a la tabla de usuarios del sistema.

Es importante destacar que la animación está implementada mediante el servicio *"NgxSpinnerService"* que proporciona un fragmento de código *"HTML"* permitiendo mostrar una animación determinada en función de varios parámetros como el tipo de animación, color de fondo, texto etc.

4.4.5 'Registro'

Maquetación

- Sección "header".
- Sección "form": esta página de creación de la cuenta es muy similar a la página de inicio de sesión, con la diferencia que el formulario estará compuesto por el nombre, apellidos, edad, correo, contraseña y repetición de la contraseña. Nuevamente, será controlado a partir de los valores introducidos mediante los elementos *"formGroup"* y *"formControlName"*. El botón "Registrarse" no se habilitará hasta que el formulario no sea

válido. Si existen errores de formato o de requerimiento de valores, se mostrará un mensaje de error justo debajo del campo del formulario correspondiente.

- Sección "footer".

Lógica

Este fichero contiene la lógica más compleja de toda la vista actual. En el constructor podemos encontrar tres servicios: "*AuthService*", "*FormBuilder*" y "*AngularFirestore*".

El formulario de registro controla cada campo a partir de este servicio de manera que se requiera obligatoriamente el nombre, el apellido, la edad, el email, y las contraseñas. Además, la edad deberá estar comprendida entre 18 y 65 años (a partir de las funciones "*Validators.min*" y "*Validators.max*"). El email volverá a verificarse cumpliendo una expresión regular, mientras que la contraseña deberá estar formada por un número de caracteres comprendidos entre 8 y 16 (a partir de las funciones "Validators.minLength", "Validators.maxLength"). Una vez que el usuario se registre correctamente, se le añadirán los retos almacenados en la base de datos automáticamente a partir del servicio "*AngularFirestore*".

A continuación, detallo las funciones utilizadas en la clase "*AccountComponent*":

- Método "private removeRepeatedChallenges": esta función recorre el "*array*" donde se almacenan los retos leídos de la base de datos de cada usuario, y guarda solo aquellos que no se repitan debido a que es posible que un usuario pueda tener el mismo tipo de reto que otro usuario diferente.

- Método "private passwordMatchValidator": esta función verifica que las contraseñas escritas coincidan al mismo tiempo que se están escribiendo.

- Método "onCheckboxChange": esta función verifica que el elemento "*checkbox*" esté pulsado para aceptar las condiciones antes de registrar un nuevo usuario.

- Método "onRegister": esta función hace uso del servicio "*AuthService*" para guardar un nuevo usuario en "*Firebase*". Un inconveniente que tiene esta función es que inicia la sesión del usuario registrado de forma obligatoria por seguridad. Si el usuario se registra correctamente, se llamará a la función "*logout*" del servicio "*AuthService*" para que tenga que iniciar sesión.

  Si el registro de usuario se realiza correctamente, se introducirán los datos de usuario en la base de datos, y se le añadirán los siguientes parámetros: foto de perfil por defecto, puntuación, ejercicios, rutinas, retos, máxima puntuación obtenida y un parámetro adicional que permite averiguar si el plan de entrenamiento está activo. Tras finalizar, se mostrará un mensaje de éxito utilizando nuevamente el módulo "*sweetalert2*". Si se produce un error, porque el usuario ya existe u otro motivo, se mostrará un mensaje de error.

```
ngOnInit(): void {
  this.registerForm = this.fb.group({
    name: ['', Validators.required],
    surname: ['', Validators.required],
    age: ['', Validators.compose([
      Validators.required,
      Validators.min(18),
      Validators.max(65)
    ])],
    email: ['', Validators.compose([
      Validators.required, Validators.pattern("^[a-zA-Z0-9._%+-]+@[a-z0-9.-]+\\.[a-z]{2,4}$")
    ])],
    password: ['', Validators.compose([
      Validators.required,
      Validators.minLength(8),
      Validators.maxLength(16)
    ])],
    confirmPassword: [''],
  },
  {
    validator: this.passwordMatchValidator
  })
```

FIGURA 31: FRAGMENTO DE CÓDIGO PARA VALIDAR EL FORMULARIO

4.4.6 'Cambio de Contraseña'

Maquetación

- Sección "header".
- Sección "form": en esta página se muestra un formulario compuesto por un solo campo que permite introducir el email para restablecer la contraseña. Si el email introducido tiene un formato incorrecto o el campo está vacío, se mostrará un mensaje de error, y el botón "Enviar Email" estará deshabilitado.
- Sección "footer".

Lógica

La clase "*ResetPasswordComponent*" utiliza el servicio "*FormBuilder*", para poder validar el correo introducido en el formulario a partir de una expresión regular.

El único método que contiene esta clase se denomina "*resetPassword*" y utiliza el servicio "*AuthService*" para que "*Firebase*" envíe un mensaje a la cuenta de correo del usuario para restablecer la contraseña. Si el correo se ha enviado correctamente, se mostrará un mensaje de éxito o de error en caso contrario.

4.4.7 'Página No Encontrada (Error 404)'

Esta página siempre se mostrará cuando se introduzca una dirección URL que no corresponda con ninguna registrada en la aplicación. Si se cumple este requisito, se mostrará en cualquiera de las vistas de cualquier tipo de usuario.

Maquetación

- Sección "header".

- Sección "article": esta página está formada por un elemento "*row*" formado por dos columnas. En la primera columna se encuentra una imagen con extensión "*.gif*", mientras que en la segunda se ha dispuesto un encabezado junto con otra imagen mostrando que la página no ha sido encontrada.
- Sección "footer".

```
<section class="article">
        <div class="row">
            <div class="col-xl-6">
                <img class="space_img" src="assets/Space_Error.gif">
            </div>
            <div class="col-xl-6">
                <div class="article1_text">
                    <h1 class="article_title">
                        ¡Oops! :(
                    </h1>
                    <h2 class="article_subtitle">
                        Página No Encontrada
                    </h2>
                    <img class="_404_img" src="assets/404_Error.gif">

                </div>
            </div>
        </div>
</section>
```

FIGURA 32: FRAGMENTO DE CÓDIGO PARA MOSTRAR LA PÁGINA DE ERROR

Lógica

La clase "*PaginaNoEncontradaComponent*" no dispone de ningún método. En el constructor se verifica si el usuario actual se encuentra con la sesión iniciada. En el caso de que la sesión esté iniciada se mostrará un botón alternativo a "Iniciar Sesión", mostrando "Hola" junto con el nombre el usuario. Este parámetro se ha obtenido leyendo la variable "*name*" de la colección de usuarios que forman parte de la base de datos.

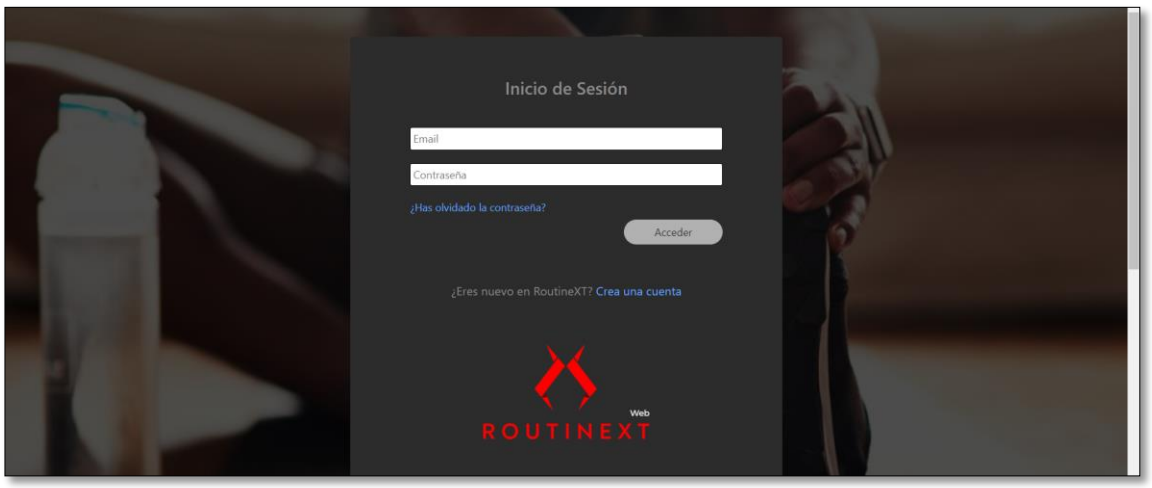


FIGURA 33: MUESTRA DE LA PÁGINA "*INICIO DE SESIÓN*"

FIGURA 34: MUESTRA DE LA PÁGINA *"FAQs_CONTACTO"*

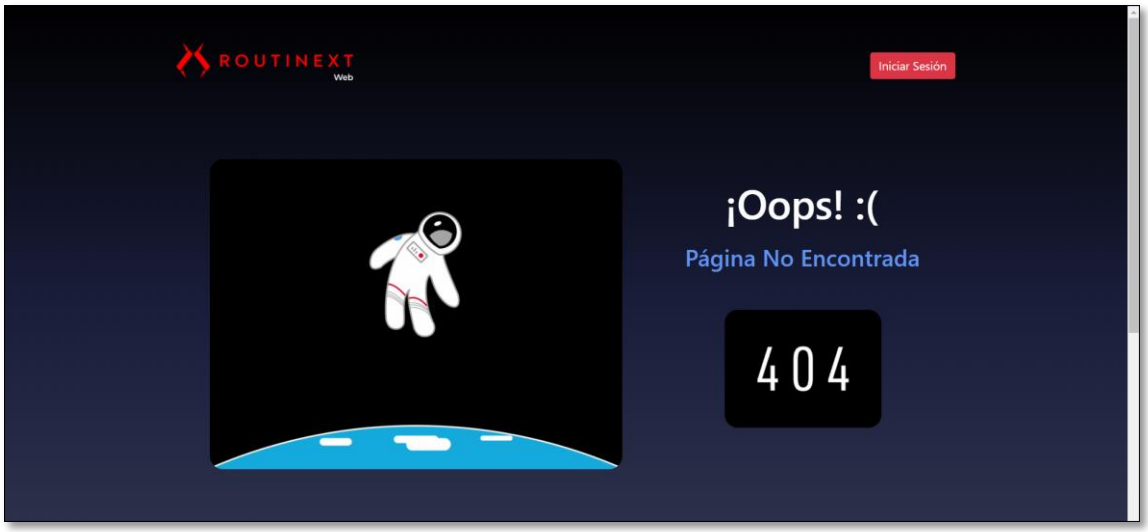


FIGURA 35: MUESTRA DE LA PÁGINA NO ENCONTRADA

4.5 Vista de cliente

La vista de cliente correspondería al perfil de un usuario que ha creado una cuenta en "*RoutineXT*". Esta persona podrá empezar a realizar ejercicios a partir de su plan de entrenamiento, adquirir nuevos retos o mejorar su puntuación. Además, podrá ver o editar su perfil, ver los ejercicios de la semana, consultar un ranking de usuarios o ver consejos para realizar un buen entrenamiento.

Páginas Web	**Tipo de usuario**	**Sprint**	**Historias de Usuario**
-Página de Carga. -Página de Plan de Entrenamiento. -Página de Ejercicios de la Semana. -Página de Listado de Retos. -Página de Sistema de Puntuación. -Página de Consejos. -Página de 'Mi Perfil'. -Página de 'Editar Perfil'.	Cliente	2,3	25 HU (3 épicas) (15-39)
Duración Estimada (inicial)	**Duración Real**		
Sprint 2: 2 semanas (22 Marzo - 2 Abril) ~ 51 horas **Sprint 3:** 2 semanas (19 Abril - 30 Abril) ~ 31,5 horas	**Sprint 2:** 4 semanas (22 Marzo - 16 Abril) = 68 horas **Sprint 3**: 1,5 semanas (19 Marzo - 29 Abril) = 27,5		

TABLA 7: PLANIFICACIÓN DE LA VISTA DE CLIENTE

Diagrama de clases [37]

Antes de comenzar a detallar cada uno de los componentes desarrollados, es conveniente conocer cómo se ha modelado la estructura del sistema, en base a sus clases, atributos, operaciones y relaciones entre objetos.

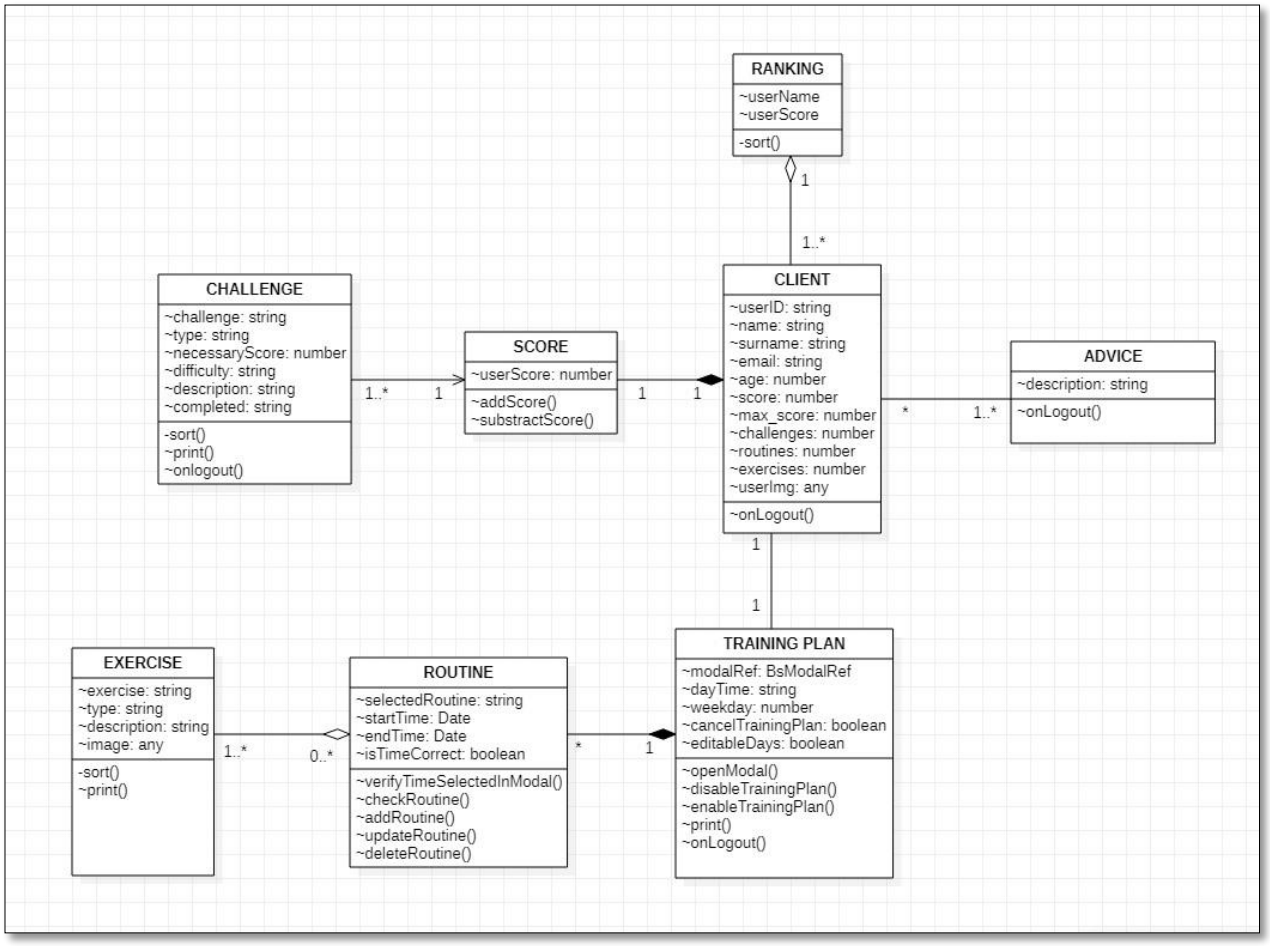


FIGURA 36: DIAGRAMA DE CLASES

En este diagrama podemos apreciar las siguientes características:

- Un ranking está formado por uno o más usuarios. No obstante, un cliente no depende del ranking.
- Un reto está relacionado directamente con la puntuación del usuario.
- Las clases "Client" y "Advice" comparten una asociación sin ningún tipo de dependencia.
- Un cliente tiene asociado un plan de entrenamiento específico.
- Un plan de entrenamiento está compuesto por rutinas que dependen de la existencia de este.
- Una rutina está formada por ejercicios. No obstante, los ejercicios no tienen que depender de una rutina para su existencia.

Diagrama entidad-relación [38]

En este apartado, también es necesario conocer cómo se ha estructurado la base de datos "*Firebase Firestore Database*". Las relaciones entre cada entidad no son complejas debido a que la base de datos está compuesta por cuatro tablas necesarias para implementar las historias de usuario correspondientes a la vista de cliente y administrador.

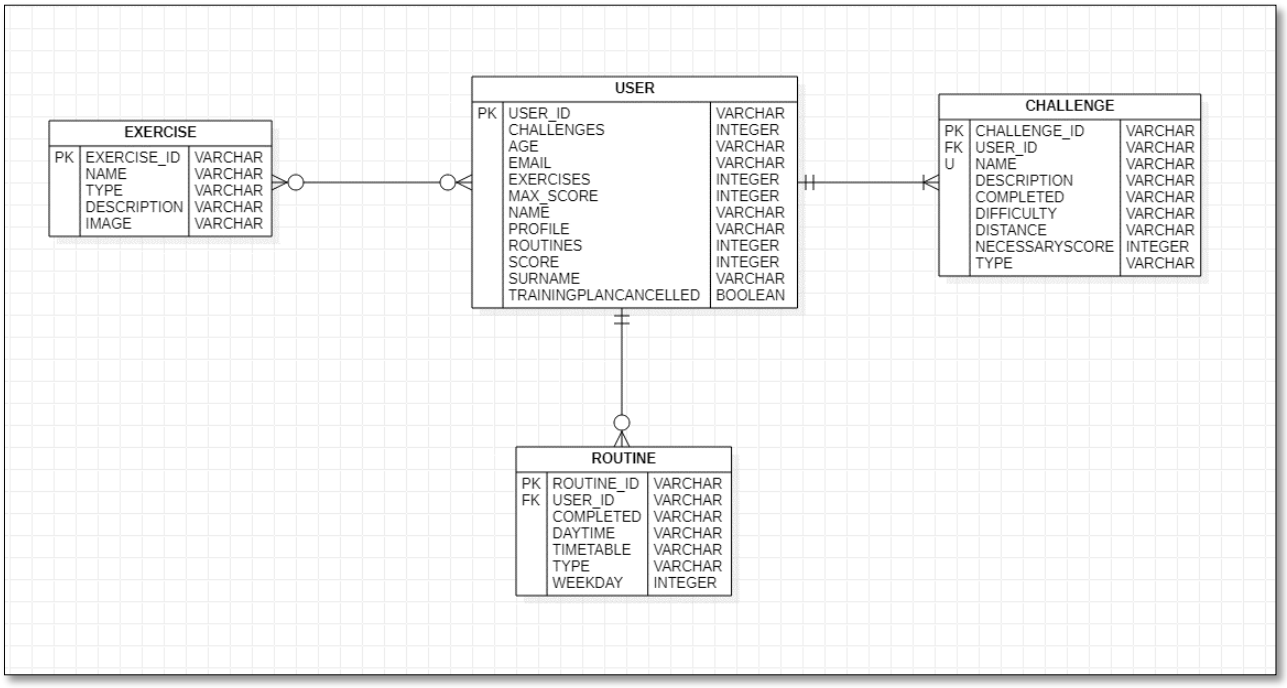


FIGURA 37: DIAGRAMA ENTIDAD-RELACIÓN

En este diagrama podemos apreciar las siguientes características:

- Un usuario como mínimo debe tener asociado un reto.
- Un reto está asociado a un usuario concreto (sus propiedades pueden repetirse entre más usuarios a excepción del identificador de usuario en la clave foránea, "FK").
- El nombre de un reto no puede repetirse. Es único.
- Un usuario puede tener asociado cero o más rutinas.
- Una rutina está asociada a un usuario concreto (sus propiedades pueden repetirse entre más usuarios a excepción del identificador de usuario en la clave foránea).
- Un usuario no necesariamente debe tener ejercicios.
- La tabla de ejercicios no depende de ninguna otra tabla.
- Cada tabla contiene un registro que actúa como identificador único ("PK").

4.5.1 'Plan de Entrenamiento'

Cuando un el usuario inicia sesión, se mostrará una pequeña animación antes de mostrar su plan de entrenamiento. Antes de explicar la utilidad del plan de entrenamiento es necesario entender tres conceptos:

- Plan de Entrenamiento: el plan de entrenamiento corresponde al horario semanal (de lunes a domingo) que está dividido en "Mañana", "Tarde" y "Noche". En cada parte del día se pueden establecer rutinas.

- Rutina: una rutina es un tiempo comprendido entre 15 y 60 minutos en el que se practica un determinado grupo muscular elegido por el usuario (piernas, brazos, flexibilidad, glúteos o abdominales). Una rutina se realiza en una parte del día concreta (mañana, tarde o noche).

- Ejercicios: conjunto de actividades por el que está formada una rutina. En la aplicación de "*RoutineXT*" todos los ejercicios tienen un minuto de duración.

Cada usuario tiene su propio plan de entrenamiento, de manera que puede personalizarlo de la siguiente forma:

- Añadir una rutina.
- Eliminar una rutina.
- Editar una rutina existente.
- Visualizar una rutina.
- Deshabilitar el plan de entrenamiento. *
- Habilitar el plan de entrenamiento.
- Imprimir el plan de entrenamiento.

* **Nota**: Deshabilitar el plan de entrenamiento significa que el usuario no podrá modificarlo. Esta opción se suele usar cuando el usuario quiere descansar la semana actual y que el sistema de puntuación no le reste puntos.

El plan de entrenamiento mostrará la semana actual, y el usuario no podrá modificar ninguna rutina si el plan de entrenamiento se encuentra deshabilitado. Además, tampoco podrá modificar los días de la semana anteriores al actual, es decir, si hoy es miércoles, el usuario no podrá modificar las rutinas ni del lunes ni del martes.

La vista de cliente tiene una apariencia similar a una plantilla de tipo "*dashboard*", donde existe un menú vertical situado a la izquierda para cambiar de página. En la parte superior, se encuentra un título descriptivo de la página, junto con la foto del usuario y, su nombre y apellidos situados a la derecha. Esta apariencia está presente en todas las páginas que forman parte de la vista de cliente y administrador.

Maquetación

- Sección "container-fluid": esta sección está formada por un elemento de tipo "row" formado por cuatro columnas. En la primera columna se situará el valor de la semana actual en un párrafo. En la segunda, encontraremos un botón que nos permitirá activar o desactivar el plan de entrenamiento actual dependiendo de su estado. La tercera columna se encuentra vacía para poder situar en la cuarta, el botón "Imprimir".

- Sección "card-group": en esta sección se ha utilizado un conjunto de elementos "card" para poder mostrar adecuadamente el plan de entrenamiento. En este grupo encontramos una fila con los días de la semana.

- Sección "card-group mornings": la siguiente fila de elementos "card" corresponden a las rutinas que el usuario puede gestionar en horario de mañana. Cada tarjeta se mostrará vacía si el usuario no ha introducido ninguna rutina o llena en caso contrario, a través de una ventana modal.

- Sección "card-group afternoons": la siguiente fila de elementos "card" corresponden a las rutinas que el usuario puede gestionar en horario de tarde. De igual manera, cada tarjeta se mostrará vacía o con la rutina correspondiente si el usuario ha pinchado en ella.

- Sección "card-group nights": la siguiente fila de elementos "card" corresponden a las rutinas que el usuario puede gestionar en horario de noche. De igual manera, cada tarjeta se mostrará vacía o con la rutina correspondiente si el usuario ha pinchado en ella.

- Sección "modal": este elemento muestra una pequeña ventana con unas opciones que el usuario debe rellenar para gestionar una rutina. Las partes corresponden a las siguientes:

    - "modal-header": contiene el día de la semana actual junto con el horario que podrá elegir el usuario dependiendo en el tipo de tarjeta en el que haya pinchado. Por ejemplo, "Lunes 00:00-12:59". En la parte derecha podemos encontrar un botón para cerrar la ventana "modal".

    - "modal-body": podemos encontrar tres filas formadas por dos columnas, en las que el usuario puede escoger el tipo de rutina que desea realizar mediante un elemento "select", la hora de inicio y de finalización de la rutina. Las horas han sido implementadas instalando el módulo "*timepicker*".

    - "modal-footer": contiene un botón que permite añadir la rutina si ha sido creada por primera vez o editarla si ya estaba creada. Existe otro botón que permite eliminar la rutina si estaba creada. Para poder añadir o editar una rutina, se debe elegir una hora de inicio y de finalización que esté comprendida en el horario que corresponde a la tarjeta donde se ha hecho *"click"* y que, además, esté comprendido entre 15 y 60 minutos.

Lógica

Este fichero contiene la lógica más compleja que forma parte de la vista de cliente. Para evitar realizar múltiples operaciones de lectura en la base de datos, se ha decido guardar el contenido del plan de entrenamiento en un array bidimensional, que tiene la ventaja de poder acceder a un dato determinado con tan solo conocer la posición.

En la clase "TrainingPlanComponent" podemos apreciar que lo primero que se realiza en el método *"ngOnInit"* es utilizar un elemento "*switch"* para obtener el primer y último día de la semana actual en base al día actual con ayudas de los métodos del módulo importado: "*formatDate"*.
A continuación, se lee todo el contenido almacenado en la base de datos del usuario actual, guardando los datos en el array bidimensional que hemos comentado. En el fichero "*HTML"* se leerá el contenido de este array mostrando el estado actual de todo el plan de entrenamiento.

A continuación, vamos a explicar cada uno de los métodos de la clase:

- Método "verifyTimeSelectedInModal": este método se encarga de verificar que la hora de inicio y de finalización de una rutina esté comprendida entre 15 y 60 minutos. Además, también verifica que las horas introducidas se incluyan en el rango de tiempo correspondiente a la hora del día actual. Si se cumplen estas condiciones, se guardará un valor "booleano" en una variable que será leído como una condición para habilitar el botón "Añadir" del modal.

- Método "openModal": este método recibe una referencia del modal que ha sido desarrollado en el fichero "HTML", el día de la semana y el momento del día correspondiente. Su única función es mostrar el modal dependiendo de sus parámetros.
- Método "checkRoutine": este método tiene la función de verificar la rutina que se ha incluido en la tarjeta. En el caso de que la tarjeta esté vacía, se llamará al método "*addRoutine*" para añadir una nueva rutina. Sin embargo, si la rutina ya había sido añadida, se llamará al método "*editRoutine*" para poder editarla.
- Método "addRoutine": este método recibe los valores introducidos en el modal (tipo de ejercicio, hora y minutos de inicio, y hora y minutos de finalización). Su función será acceder a la base de datos y añadir estos valores para que puedan ser leídos cuando el usuario vuelva a acceder a esta página mediante la función "*add*". De igual forma, se añadirá al array bidimensional para mostrar la rutina añadida en el plan de entrenamiento.
- Método "updateRoutine": este método es exactamente igual que el anterior, con la diferencia de que la llamada a la base de datos se realiza con la función "*update*".
- Método "deleteRoutine": esta función también es muy similar a las anteriores. Dependiendo de la tarjeta en la que se ha hecho *"click"* se eliminará la rutina actual de la base de datos y el array bidimensional quedará vacío eliminando los datos de dicha rutina.
- Método "disableTrainingPlan": para que las tarjetas de los días anteriores al actual estén deshabilitadas se utiliza un array "booleano" donde se va anotando los días que no se deben poder modificar. Este método deshabilita todos los días modificando todos los valores de este array. Además, establece a "false" la variable "*trainingPlanCancelled*" que forma parte del usuario en la base de datos.
- Método "enableTrainingPlan": este otro método funciona exactamente igual que el anterior, con la diferencia de que habilita todos los días de la semana actual sin incluir los días anteriores al actual. También, establece a "true" la variable *"trainingPlanCancelled"* que forma parte del usuario en la base de datos.
- Método "print": esta función simplemente realiza una llamada a "*window.print*" cuando el usuario hace *"click"* en el botón "Imprimir".
- Método "onLogout": esta función es llamada cuando el usuario hace *"click"* en la ventana desplegable que aparece cuando hace *"click"* en su nombre y apellidos. Para ello, se llama a la función "*logout",* mediante el servicio "*AuthService",* donde se muestra la página de inicio de sesión con mensaje de confirmación del cierre de sesión gracias al módulo "*sweetalert2*".

Impresión de los ejercicios

Para poder imprimir el horario se ha añadida una serie de reglas en las hojas de estilo del componente actual para que el horario se pueda imprimir en horizontal, manteniendo todos los colores originales y aprovechando el mayor espacio posible.

Esto se ha podido realizar gracias a la notación "*@media print*" que permite añadir propiedades "*CSS*" para que solo afecte al formato de impresión.

```
@media print {

    div.d-flex {
        writing-mode: tb-rl;
        height: 100%;
        margin: 0 auto;
    }

    header, nav, .menu, img, button, #sidebar-container {
        display: none;
    }

    #content {
        height: 180%;
        background-color: white !important;
    }

    #content > div {
        margin-bottom: 1%;
    }
```

FIGURA 38: FRAGMENTO DE CÓDIGO PARA LA IMPRESIÓN DE LOS EJERCICIOS

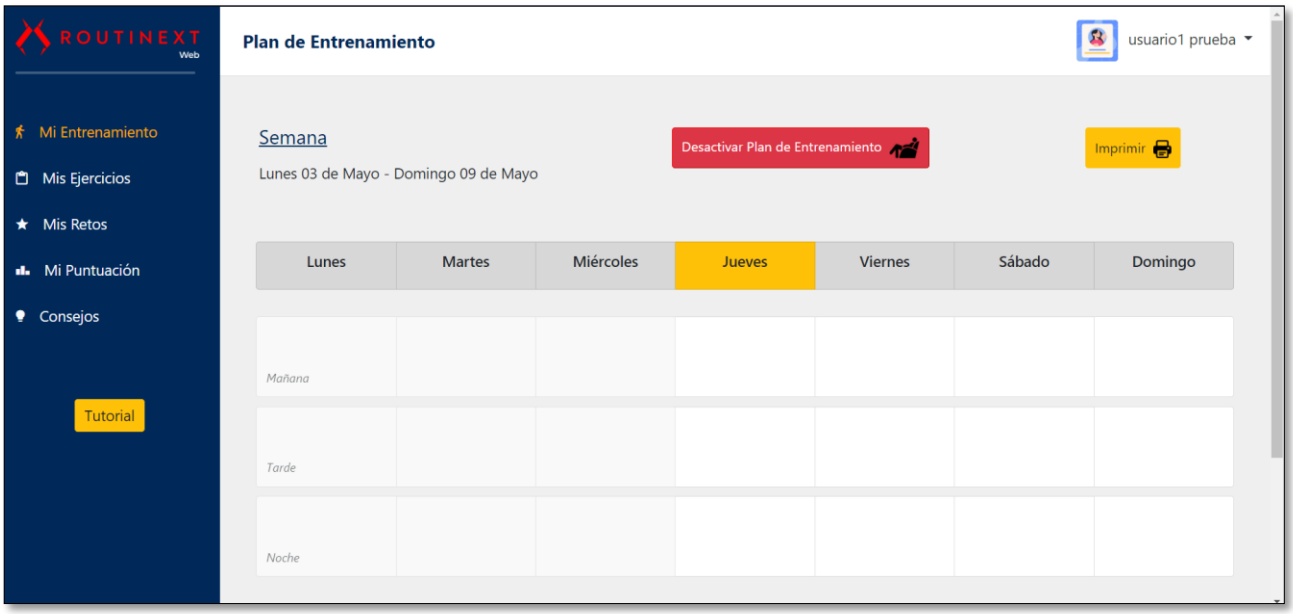


FIGURA 39: MUESTRA DE LA PÁGINA DE ENTRENAMIENTO

FIGURA 40: MUESTRA DEL *"MODAL"*

4.5.2 'Ejercicios de la Semana'

En esta página el usuario podrá visualizar cada uno de los ejercicios que forman parte de la rutina actual. Se mostrará en orden los ejercicios de cada rutina por día de la semana. Se podrá ver el título, y la foto de cada ejercicio en forma de columnas y filas. Esta lista de ejercicios también podrá ser impresa si el usuario lo desea.

Maquetación

- Sección "container-fluid": esta sección muestra los ejercicios con su título y foto distribuidos tres columnas por cada fila mediante los elementos de "*row*" y "*col-4*". Para ello, se recorre mediante un bucle el array bidimensional de ejercicios a través de la directiva "**ngFor*", de manera que se muestren mediante "*cards*" como se ha realizado con el plan de entrenamiento. Los ejercicios están ordenados dependiendo del día de la semana que incluya el array, de manera que se mostrarán en primer lugar todos los ejercicios que formen parte de una rutina creada a principio de semana.
- Sección "emptyPlanning": esta sección mostrará a partir de un párrafo, un mensaje que ocupe gran parte de la pantalla del usuario que no se encuentran ejercicios disponibles para mostrar debido a que el plan de entrenamiento está vacío.

Lógica

Este fichero también contiene una lógica bastante compleja, pero no tanto como en el caso del plan de entrenamiento.

En el método "*ngOnInit*" creamos un array bidimensional que permita almacenar cada uno de los ejercicios que forman parte de las rutinas creadas por el usuario. Para ello, guardamos desde la base de datos todos los ejercicios disponibles. A continuación, comenzamos a leer cada rutina, guardando sus valores en diferentes variables que mostraremos al usuario mediante la interpolación antes de enseñar la lista de ejercicios. También comprobaremos que el plan de entrenamiento no esté vacío guardando en una variable "booleana" su estado para que se muestre o no la sección de "EmptyPlanning" que comentamos anteriormente.

Los métodos por los que está formado la clase "*ExerciseComponent*", no serán detallados al ser exactamente idénticos a los que hemos comentado en el plan de entrenamiento:

- Método "print".
- Método "onLogout".

FIGURA 41: MUESTRA DE LA PÁGINA *"EJERCICIOS DE LA SEMANA"*

4.5.3 'Listado de Retos'

Esta página muestra al usuario una tabla con los retos que forman parte de "*RoutineXT*". Además, puede consultar la puntuación actual y el número de retos que ha completado. Esta tabla también puede ser impresa si el usuario lo desea. No obstante, los retos no podrán estar visibles debido a que se necesita alcanzar una puntuación mínima para desbloquearlos. Aquellos retos de la tabla donde el usuario no cumpla este requisito se mostrarán con un signo de interrogación en la celda correspondiente, a excepción del nombre del reto.

Cada reto completado aporta al usuario la siguiente puntuación:

- Dificultad Fácil: +25 puntos.
- Dificultad Media: +45 puntos.
- Dificultad Difícil: +75 puntos.

Maquetación

- Sección "row": en esta sección se mostrará una fila con cuatro columnas, mostrando en distintos párrafos la puntuación actual del usuario, los retos que ha completado, una pequeña leyenda con los puntos que aporta cada reto y el botón "Imprimir".
- Sección "table": la tabla solo se muestra si el array bidimensional que contiene los retos tiene una longitud superior a cero. Para mostrar cada uno de ellos se utilizará la directiva "**ngFor*", donde se mostrará por pantalla el reto dependiendo de si se cumplen las condiciones dadas por las directivas "**ngIf*" que permitirán que se muestre un campo concreto del reto o un signo de interrogación "?" si no se cumplen. La tabla está distribuida de la siguiente forma:
    - Reto.
    - Tipo de Reto.
    - Puntuación Necesaria.

- Dificultad.
- Descripción.
- Completado.

El campo de la tabla "Completado", mostrará una imagen que muestre que un reto ha sido finalizado junto con la fecha correspondiente. En el caso que no sea haya completado se mostrará un guion "-".

Lógica

Cada reto al igual que una rutina, tiene asociado un usuario específico, debido a que más de un usuario puede tener acceso a los mismos tipos de retos. El método *"ngOnInit"* lee en la base de datos los retos del usuario autenticado, agregando a un array bidimensional todo su contenido para que pueda ser mostrado en la tabla. No obstante, antes de mostrar la tabla, es necesario ordenar el array de manera que los retos aparezcan de menor a mayor puntuación. A continuación, mostramos los métodos que se han utilizado en la clase "*ChallengeComponent*":

- Método "private sort": este método ordena las filas del array bidimensional dependiendo de la puntuación que almacene cada uno. El mecanismo de ordenación utilizado es el "ordenamiento por burbuja", donde comparamos cada elemento con el siguiente verificando si la puntuación de la posición actual es mayor que la del siguiente.
- Método "print".
- Método "onLogout".

```
81  private sort() {
82    let aux: number;
83    for (let k = 1; k < this.challenges.length; k++) {
84      for (let i = 0; i < (this.challenges.length - k); i++) {
85        if (this.challenges[i][3] > this.challenges[i+1][3]) {
86          aux = this.challenges[i];
87          this.challenges[i] = this.challenges[i+1];
88          this.challenges[i+1] = aux;
89        }
90      }
91    }
92  }
```

FIGURA 42: FRAGMENTO DE CÓDIGO DONDE SE MUESTRA EL MÉTODO DE *"ORDENAMIENTO POR BURBUJA"* [39]

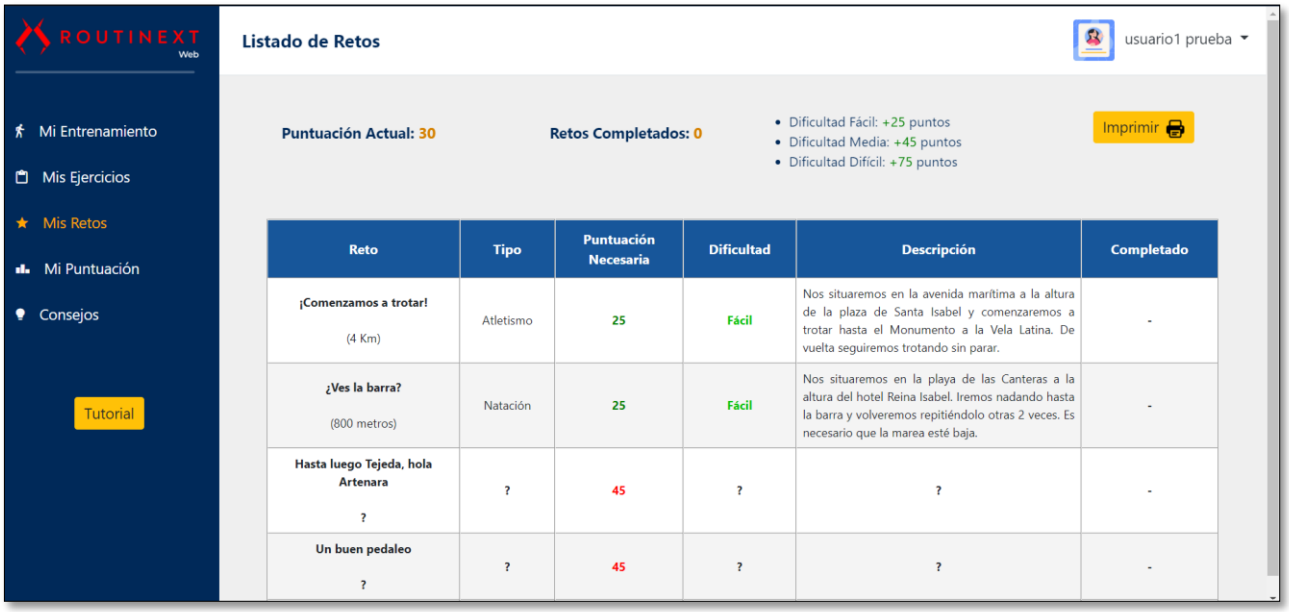


FIGURA 43: MUESTRA DE LA PÁGINA *"LISTADO DE RETOS"*

4.5.4 'Sistema de Puntuación'

Esta página permite mostrar al usuario estadísticas acerca de su entrenamiento de la semana. Los datos que se muestran corresponden a la puntuación actual, los puntos obtenidos en la semana actual, el número de rutinas realizadas a lo largo de la semana, las rutinas que deben realizarse o que han sido realizadas especificando la puntuación obtenida en cada una de ellas, y finalmente, un ranking en el que se muestra la posición que ocupa el usuario respecto a todos los clientes de "*RoutineXT*".

Maquetación

- Sección "col-6 routines": esta sección está formada por una única columna. Se mostrará en una lista desordenada ("*ul*") las estadísticas que hemos comentado anteriormente resaltando en color rojo la obtención de una puntuación negativa y en verde la obtención de una puntuación positiva.
  Para mostrar las rutinas de la semana se ha utilizado la directiva "**ngFor*", que permite recorrer el array compuesto por todas las rutinas que pertenecen al usuario actual. Estas rutinas se muestran ordenadas por día de la semana. Si una rutina ha servido para obtener puntos se mostrará en color verde si se cumple la condición específica dada por la directiva "**ngIf*". En caso contrario, será mostrada en color rojo.

- Sección "col-6 ranking": para mostrar el *ranking* se ha diseñado una tabla de tres columnas añadiendo la propiedad "*overflow: scroll*" en las hojas de estilo para que la tabla tenga una barra de navegación y no ocupe una gran extensión de la página.
  La tabla contendrá el puesto de cada usuario, el nombre del usuario y su puntuación. El icono que representa el puesto del usuario actual será similar a una corona para que pueda distinguirse del resto de usuarios. Esto ha podido ser posible gracias a un emoticono insertado por parte de la biblioteca de "*Iconify*".

Lógica

En el método "ngOnInit" se recorre la colección de rutinas que se encuentra en la base de datos del proyecto de manera que se almacene su contenido en un array bidimensional. En el caso del día de la semana, al estar almacenado como un número, se ha hecho uso de un módulo aparte que se encuentra en la carpeta "*modules*". Dentro contiene el fichero "*maps.ts*", que permite realizar conversiones directas de parámetros a valores de la siguiente forma:

```
44    indexWeekdayToWeekday = new Map([
45        [0, "Lunes"],
46        [1, "Martes"],
47        [2, "Miércoles"],
48        [3, "Jueves"],
49        [4, "Viernes"],
50        [5, "Sábado"],
51        [6, "Domingo"],
52    ])
```

FIGURA 44: FRAGMENTO DE CÓDIGO DONDE SE UTILIZA LA INTERFAZ *"MAP"*

Tras haber guardado los resultados en el array, los ordenaremos por puntuación de menor a mayor. A continuación, procedemos a detallar los métodos que forman parte de la clase "*ScoreComponent*":

- Método "private sort": este método permite ordenar las filas de un array bidimensional de menor a mayor según la puntuación de cada usuario.
- Método "onLogout".

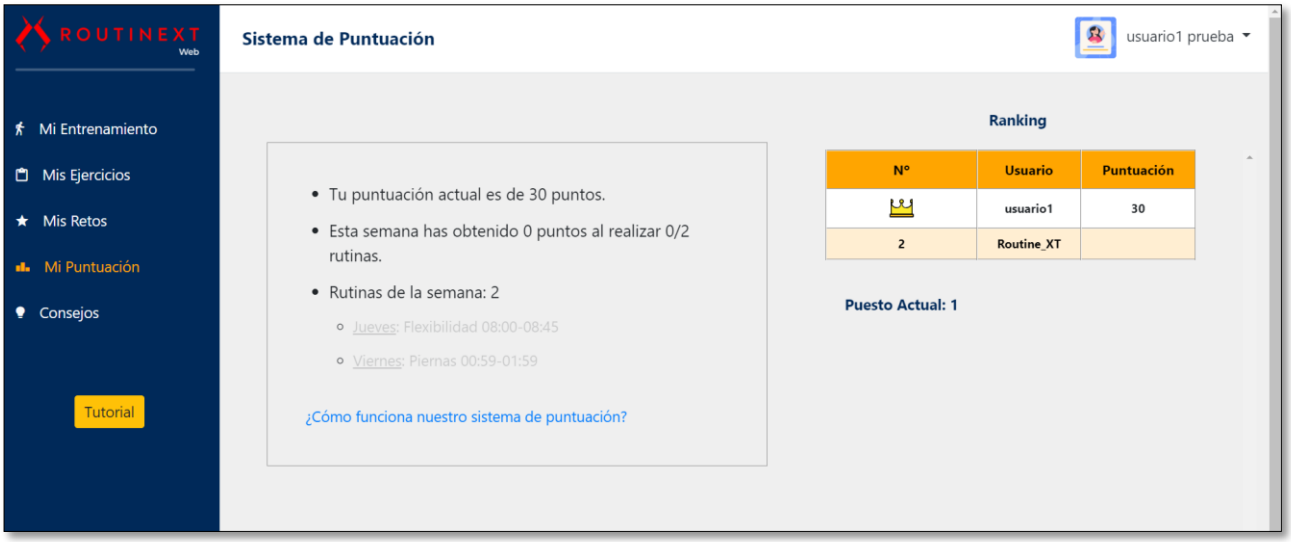


FIGURA 45: MUESTRA DE LA PÁGINA *"SISTEMA DE PUNTUACIÓN"*

4.5.5 Consejos

Esta página permite proporcionar al usuario una serie de consejos y buenas prácticas. Se ha intentado que la semántica de los consejos no sean meramente frases motivadoras, sino argumentos que tengan utilidad para realizar un buen entrenamiento. Las buenas prácticas son frases que se muestren como una lista de dos columnas, mientras que los consejos se muestran como diapositivas automáticas haciendo uso del elemento *"carousel"*.

Maquetación

- Sección "carousel": para poder mostrar los consejos se ha utilizado el elemento "*carousel-inner*" que permite añadir consejos distribuidos en diferentes párrafos. Para que las diapositivas que forman parte del carrusel cambien automáticamente, se ha tenido que añadir la subclase "*active*" a continuación de la clase "*carousel-item*" en el primer elemento del carrusel.
  Los botones de "siguiente" (">") y "anterior" ("<") son controlados mediante la clase "*carousel-control-prev*".

- Sección "row": esta sección contiene una fila formada por dos columnas con tres elementos en cada una donde se distribuirán las frases relacionadas con las buenas prácticas haciendo uso de una lista desordenada ("*ul*").

```
<div class="container-fluid">
    <div class="row mx-auto">
        <p class="title">Consejos</p>

        <div id="demo" class="carousel slide mx-auto" data-ride="carousel">

            <div class="carousel-inner">
                <div class="carousel-item active">
                    <p>A continuación, te presentamos algunos consejos útiles para tu entrenamiento:</p>
                </div>

                <div class="carousel-item">
                    <p>Si puedes, intenta variar haciendo ejercicio fuera de casa o con más gente.</p>
                </div>
```

FIGURA 46: FRAGMENTO DE CÓDIGO DONDE SE MUESTRAN LOS CONSEJOS

Lógica

Cuando se comenzó a implementar la clase *"AdviceComponent"* se crearon dos colecciones relacionadas con consejos y buenas prácticas. No obstante, esta idea fue desechada más tarde debido a que no se consideró que fuesen elementos que podrían modificarse o variar con el tiempo por lo que se prescindió de estas colecciones.

El método *"ngOnInit"*, se encuentra vacío porque es un método que siempre debe tener la clase. El único método que contiene la clase sería:

- Método "onLogout".

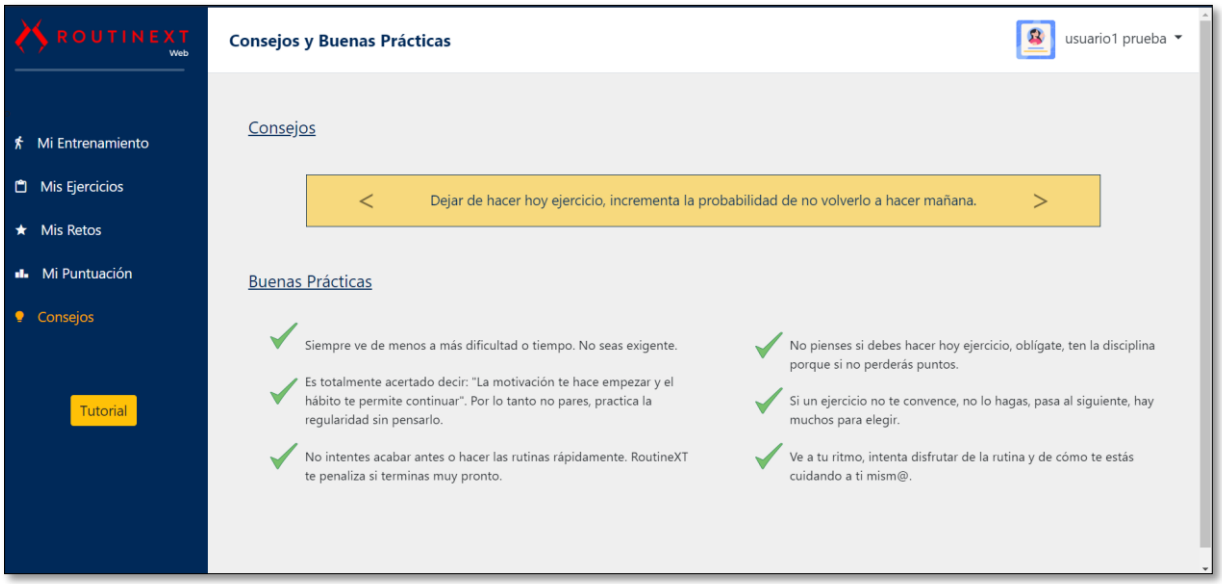


FIGURA 47: MUESTRA DE LA PÁGINA "*CONSEJOS*"

4.5.6 'Mi Perfil'

La mayor parte de las plantillas "*dashboard*" siempre deben tener una sección dedicada al perfil del usuario. He decidido crear esta sección para que el usuario pueda visualizar los siguientes datos:

- Foto de Perfil.
- Nombre.
- Apellidos.
- Correo Personal.
- Puesto en el "*ranking*".
- Puntuación.
- Rutinas Completadas.
- Ejercicios Realizados.
- Retos Completados.
- Puntuación Máxima Obtenida.

Además, el usuario podrá cambiar su contraseña y editar sus datos siempre que lo desee.

Maquetación

- Sección "col-4": en esta sección se utiliza un elemento "*card*" para poder mostrar la foto del perfil de usuario. Esto se realizará con la propiedad "*binding*" que permite mostrar una imagen a través de su dirección contenida en una variable. Justo debajo de la imagen, mostraremos la puntuación máxima obtenida por el usuario mediante un párrafo ("*p*").
- Sección "col-2": en esta sección podemos encontrar varios párrafos donde mostramos el nombre, apellidos, edad y correo del usuario.

- Sección "col-3": se ha hecho uso de una lista de definición ("*dd*") para poder mostrar el puesto en el "*ranking*" de usuarios, la puntuación, las rutinas completadas, los ejercicios realizados y retos completados haciendo uso de los elementos "*dt*" y "*dd*" que forman parte de la lista de definición.

```
<div class="col-3">
    <dl>
        <div class="box ">
            <dt>Ranking</dt>
                <dd>{{userRanking}}º Puesto</dd>
        </div>
        <div class="box ">
            <dt>Puntuación</dt>
                <dd>{{score}}</dd>
        </div>
        <div class="box ">
            <dt>Rutinas Completadas</dt>
                <dd>{{routines}}</dd>
```

FIGURA 48: FRAGMENTO DE CÓDIGO DE LA LISTA DE DEFINICIÓN

Lógica

En la clase "*ProfileComponent*", se obtienen todas las propiedades del usuario del base de datos, almacenándolos en un array bidimensional para que puedan ser mostrados en la vista del usuario. Para mostrar el puesto en el "*ranking*" ha sido necesario ordenar el array por puntuación con el mecanismo "ordenamiento por burbuja". Los métodos por los que está formada la clase corresponden a los siguientes:

- Método "private sort".
- Método "onLogout".

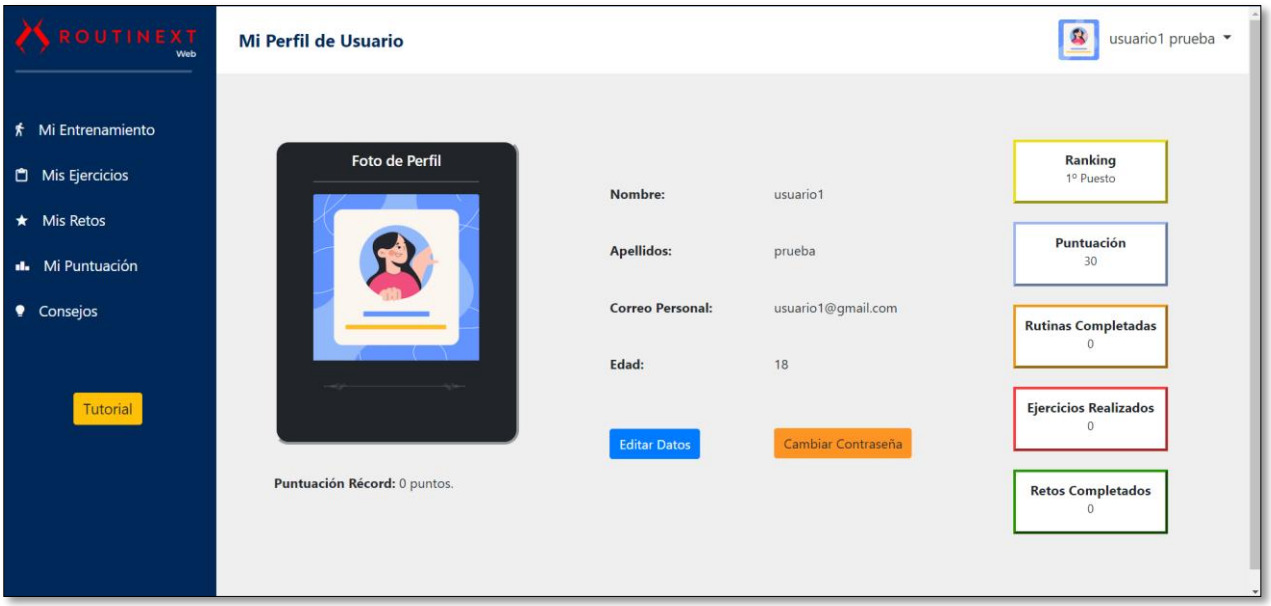


FIGURA 49: MUESTRA DE LA PÁGINA *"MI PERFIL"*

4.5.7 'Editar Perfil'

Esta página permite al usuario editar sus datos personales, cuando el usuario pincha en el botón "Editar Datos" que se encuentra en la página de "Mi Perfil". Los datos que puede modificar corresponden a los siguientes:

- Nombre.
- Apellido.
- Edad.
- Correo Personal.
- Foto de Perfil.

Maquetación

- Sección "col-4": esta sección es idéntica a la que aparece en la página "Mi Perfil" con la diferencia que se utiliza el elemento *"progressbar"* para poder mostrar una barra de progreso respecto a la subida de una nueva foto de usuario. Cuando la foto se haya subido, la página se refrescará automáticamente mostrando "Mi Perfil".
- Sección "col-2": esta sección sí difiere de la que aparece en la página de "Mi Perfil", debido a que se hace uso de los elementos *"formGroup"* y *"formControlName"* debido a que los datos del usuario se muestran como un formulario que puede modificarse y requiere la validación de los datos que se introducen en él. Si algún dato no es correcto, se mostrará debajo del campo un mensaje de error para que el usuario pueda corregirlo.
- Sección "col-3": esta sección es exactamente idéntica a la que aparece en la página "Mi Perfil" mostrando las estadísticas del usuario.

Lógica

La clase "*EditProfileComponent*" realiza una lectura de los datos de usuario que se encuentran en la base de datos guardando los datos del usuario en un array bidimensional. Además, el método *"ngOnInit"* permite crear un formulario de validación para poder verificar que los datos que ha introducido el usuario son correctos. A continuación, vamos a detallar las diferentes funciones que se han implementado en esta clase:

- Método "private sort": permite ordenar el array correspondiente a los datos relacionados con el ranking de usuarios para mostrar el puesto que ocupa el usuario autenticado.
- Método "updateImg": este método se utiliza cuando el usuario sube una nueva imagen para actualizar su foto de perfil. Para ello, es necesario acceder a la carpeta que se encuentra en la base de datos (en "Firebase Storage"), para poder guardar la imagen. Si la imagen ha sido subida correctamente, se guardará también la URL que ha sido generada en una de las propiedades de la colección del usuario autenticado. Además, se mostrará una barra de progreso para poder actualizar la página cuando termine la imagen de subirse.
- Método "onUpdate": esta función solo sirve para poder actualizar los datos del usuario accediendo a la colección "usuarios" para actualizar el nombre, el apellido, la edad y el correo mediante la función "*update*" que proporciona "*Firebase*".

- Método “onLogout”.

```
updateImg(evento) {
  if(evento.target.files.length > 0) {
    let archivo = evento.target.files[0];
    let nombre = archivo.name.toString();
    let ruta = 'profile_Images/' + nombre;
    const ref = this.storage.ref(ruta)
    const tarea = ref.put(archivo)
    tarea.then((objeto) => {
      console.log('Imagen subida');
      ref.getDownloadURL().subscribe((url) => {
        this.db.collection('users').doc(this.userID).update({
          profile: url
        })
      })
    })
    tarea.percentageChanges().subscribe((porcentaje) => {
      // console.log(porcentaje)
      this.uploadPercentage = parseInt(porcentaje.toString())
      if(this.uploadPercentage == 100) {
        setTimeout(() => {
          this.router.navigate(['/Perfil'])
        }, 2000)
        // setInterval(() => window.location.reload(), 1000);
      }
    })
  }
}
```

FIGURA 50: FRAGMENTO DE CÓDIGO PARA ACTUALIZAR LA FOTO DE PERFIL

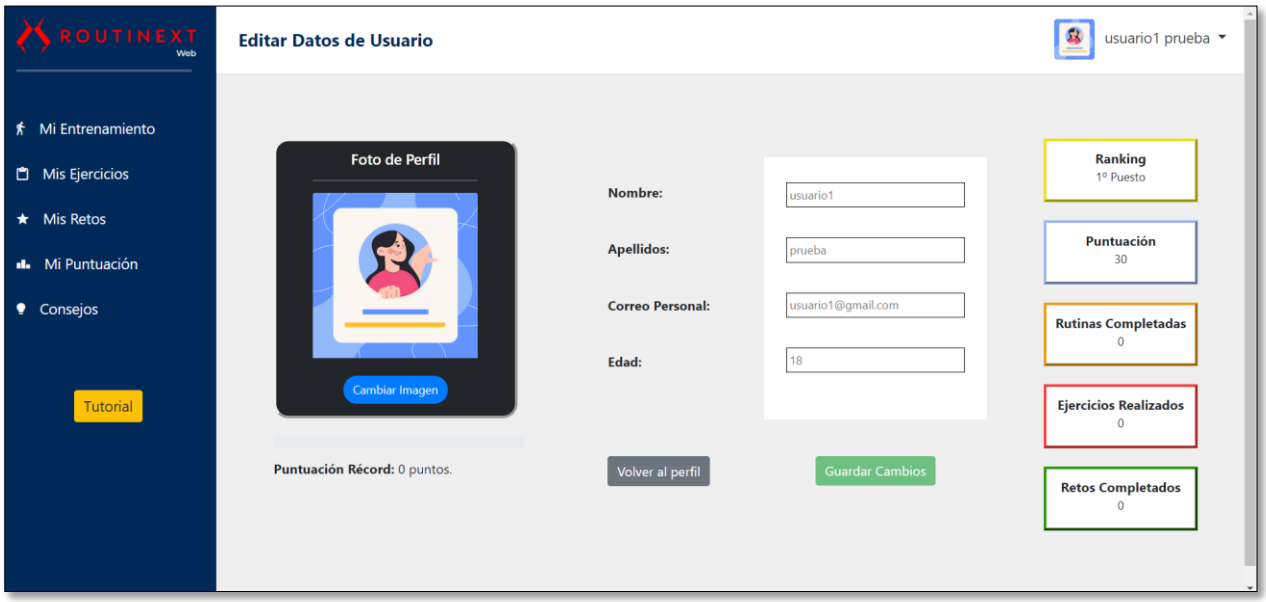


FIGURA 51: MUESTRA DE LA PÁGINA *“EDITAR PERFIL”*

4.6 Vista de administrador

"RoutineXT" contiene un único administrador/a que utiliza la cuenta de correo: routineXT_adm@outlook.com. El administrador puede realizar cuatro funciones básicas relacionadas con la gestión del sistema:

- Visualizar, editar, eliminar y filtrar un usuario registrado con el rol de cliente del sistema.
- Visualizar, editar, eliminar, añadir y filtrar los ejercicios almacenados en la base de datos.
- Visualizar, editar, eliminar, añadir y filtrar los retos almacenados en la base de datos.
- Visualizar, enviar y responder a los correos recibidos en su bandeja de entrada a través del formulario de contacto de un usuario no registrado.
- Visualizar o editar los datos de su perfil.
- Cambiar su contraseña de acceso.

Páginas Web	**Tipo de usuario**	**Sprint**	**Historias de Usuario**
-Página de Carga -Página de Usuarios del Sistema -Página de Tabla de Ejercicios -Página de Tabla de Retos -Página de 'Mi Perfil' -Página de 'Editar Perfil'	Administrador	4	21 HU (3 épicas) (40-60)
Duración Estimada (inicial)		**Duración Real**	
Sprint 4: 1 semana (30 Abril - 7 Mayo) ~ 18 horas		**Sprint 4:** 1 semana (30 Abril - 4 Mayo) = 30,5 horas	

TABLA 8: PLANIFICACIÓN DE LA VISTA DE ADMINISTRADOR

4.6.1 'Usuarios del Sistema'

En esta página el administrador puede visualizar una tabla donde aparecen los siguientes datos de cada usuario:

- Nombre.
- Apellidos.
- Edad.
- Correo.
- Puntuación.

Adicionalmente, el administrador podrá editar o eliminar un usuario pulsando en los botones que se encuentran en una columna adicional. En esta vista, también se utiliza una plantilla "dashboard",

muy similar a la que aparece en la vista de un usuario registrado, pero con una disposición de colores de fondo diferentes.

Maquetación

- Sección "row": esta sección contiene una fila con dos botones que sirven para ordenar la tabla de usuario por puntuación (de menor a mayor) o por edad (de menor a mayor). Cuando se hace *"click"* en uno de los botones se llama a la función encargada del ordenamiento de las filas de la tabla.
- Sección "table": esta sección está formada por una tabla que contiene los datos de cada usuario. Cada dato es accedido mediante la directiva "*ngFor" que permite recorrer el array bidimensional y mostrarlos en la tabla a través de la interpolación "{{array[0]}}".
- Sección "ng-template": cuando el administrador o administradora quiere editar los datos de un usuario, se le desplegará una ventana modal al pinchar en "Editar". Esta ventana está formada por diferentes filas compuestas por elementos "*inputs*" que permiten modificar los datos del usuario. El botón "Editar" será habilitado cuando los datos sean correctos como si se tratara de un formulario.

Lógica

En la clase "*UserAdmComponent*", utilizamos el método *"ngOnInit"* para poder leer los datos de cada usuario que se encuentra en la base de datos a excepción del administrador/a. Estos datos serán almacenados en un array bidimensional para mostrar su contenido en la tabla. A continuación, vamos a detallar cada uno de los métodos que forman parte de esta clase:

- Método "openModal": este método permite establecer en la ventana "*modal*" los valores que corresponden al usuario que se quiere editar, y además sirve para mostrar el "*modal*".
- Método "checkModalValues": este método comprueba los valores editados o introducidos en cada elemento "*input*" del modal en tiempo real gracias a la propiedad "*two way binding*".
- Método "orderByAge": este método ordena las filas de la tabla mediante el ordenamiento por burbuja a partir de la edad de cada usuario.
- Método "orderByScore": este método ordena las filas de la tabla mediante el ordenamiento por burbuja a partir de la edad de cada usuario.
- Método "updateUser": esta función accede a la base de datos a partir del identificador del usuario proporcionado por parámetro, y actualiza el contenido de la base de datos. Además, también actualiza los datos del array bidimensional del usuario con su identificador único. Cuando el usuario se edita correctamente, se muestre un breve mensaje de que la operación se ha realizado con éxito o con error si se diera el caso.
- Método "deleteUser": esta función accede a la base de datos a partir del identificador del usuario (proporcionado con un parámetro), y elimina el usuario actual. Además, también vacía los datos del array bidimensional del usuario mediante su identificador único. Esta función también eliminará todos los retos asociados al usuario eliminado.
- Método "onLogout".

```
updateUser(id: string) {
  this.modalService.hide();
  this.db.collection('users').doc(id).update({
    name: this.nameModal,
    surname: this.surnameModal,
    age: this.ageModal,
    email: this.emailModal
  }).then((registered) => {
    this.toast.fire({
      icon: 'success',
      title: 'Usuario editado correctamente'
    })
  }).catch((error) => {
    this.toast.fire({
      icon: 'error',
      title: 'El usuario no ha podido ser editado'
    })
  })

  for (let i = 0; i < this.users.length; i++) {
    if(this.users[i][0] == id) {
      this.users[i][1] = this.nameModal;
      this.users[i][2] = this.surnameModal;
      this.users[i][3] = this.ageModal;
      this.users[i][4] = this.emailModal;
    }
  }

}
```

FIGURA 52: FRAGMENTO DE CÓDIGO PARA ACTUALIZAR LOS DATOS DEL USUARIO

FIGURA 53: MUESTRA DE LA PÁGINA *"USUARIOS DEL SISTEMA"*

4.6.2 'Tabla de Ejercicios'

En esta página el administrador puede visualizar una tabla donde aparecen los datos principales de cada ejercicio:

- Nombre del ejercicio.
- Categoría (piernas, brazos, abdominales, glúteos, flexibilidad).
- Descripción.

Adicionalmente, el administrador podrá editar o eliminar un ejercicio pulsando en los botones que se encuentran en una columna adicional. Por otra parte, el administrador también podrá ordenar los ejercicios por categoría y añadir uno nuevo.

Maquetación

- Sección "row": esta sección contiene una fila con dos botones que sirven para ordenar la tabla de ejercicios por categoría o añadir un nuevo ejercicio a la base de datos. Cuando se hace *"click"* en el botón "Filtrar por categoría" se llama a la función encargada del ordenamiento de las filas de la tabla. Por otra parte, si se pulsa el botón "+ Añadir Ejercicio" se despliega una ventana modal para rellenar los datos del nuevo ejercicio.
- Sección "table": esta sección está formada por una tabla que contiene los datos de cada ejercicio. Cada dato es accedido mediante la directiva "**ngFor*" que permite recorrer el array bidimensional y mostrarlos en la tabla a través de la interpolación "{{*array*[0]}}".
- Sección "ng-template editar": cuando el administrador o administradora quiere editar los datos de un ejercicio, se le desplegará una ventana modal al pinchar en "Editar". Esta ventana está formada por diferentes filas compuestas por elementos *"inputs"* que permiten modificar los datos del ejercicio. El botón "Editar" será habilitado cuando los datos sean correctos como si se tratara de un formulario.
- Sección "ng-template añadir": cuando el administrador o administradora quiere añadir los datos de un ejercicio, se le desplegará una ventana modal al pinchar en "+ Añadir Ejercicio". Esta ventana está formada por diferentes filas compuestas por elementos *"inputs"* que permiten rellenar los datos del ejercicio. El botón "Añadir" será habilitado cuando los datos sean correctos como si se tratara de un formulario.

Lógica

En la clase "*ExerciseAdmComponent*", utilizamos el método *"ngOnInit"* para poder leer los datos de cada ejercicio que se encuentra en la base de datos. Estos datos serán almacenados en un array bidimensional para mostrar su contenido en la tabla. A continuación, vamos a detallar cada uno de los métodos que forman parte de esta clase:

- Método "openModal_Edit": este método permite establecer en el "*modal*" los valores que corresponden al ejercicio que se quiere editar, y además sirve para mostrar el "*modal*".
- Método "openModal_Add": este otro método tiene una función muy similar al anterior con la diferencia que, al mostrar el "*modal*", vacía el valor que contienen las variables que se muestran en el "*input*".
- Método "checkModalValues": este método comprueba los valores que se han introducido en los "*inputs*" del "*modal*" independientemente de si se ha querido añadir o editar un ejercicio. También se utiliza la propiedad "two way binding".
- Método "orderByCategory": este método ordena las filas de la tabla a partir de la categoría. No obstante, la lógica es mucho más compleja debido a que se recorre el array bidimensional que contiene los ejercicios, y se almacena en diferentes "*arrays*" cada ejercicio según la categoría. Este ordenamiento se lleva a cabo mediante un elemento "*switch*". Una vez que se ha realizado esta operación, estos ejercicios son añadidos al

"*array*" de ejercicios en el siguiente orden: flexibilidad, abdominales, glúteos, piernas y brazos.

- Método "addExercise": esta función accede a la base de datos, y añade los valores introducidos creando una nueva instancia en la colección de ejercicios. Además, también añade los datos al array bidimensional de ejercicios. Cuando el ejercicio es añadido correctamente, se muestra un breve mensaje de que la operación se ha realizado con éxito o con error si se diera el caso. En el caso de la imagen, se añade una por defecto al no tener esta página diseñada para añadir o editar fotografías.
- Método "updateExercise": esta función es muy similar a la anterior con la única diferencia que se accede a la base de datos a partir del identificador del ejercicio proporcionado por parámetro para editar su contenido. Además, también actualiza los datos al "*array*" bidimensional de ejercicios. Cuando el ejercicio se edita correctamente, se mostrará un breve mensaje acerca de que la operación se ha realizado con éxito o con error si se diera el caso.
- Método "deleteUser": esta función accede a la base de datos a partir del identificador del usuario (proporcionado por parámetro), y elimina el ejercicio actual. Además, también vacía los datos del array bidimensional del ejercicio gracias a su identificador único.
- Método onLogout().

FIGURA 54: MUESTRA DE LA PÁGINA "*TABLA DE EJERCICIOS*"

```
orderByCategory() {
  let flexArray = []
  let abdArray = []
  let glutArray = []
  let pierArray = []
  let brazArray = []
  let flex = 0, abd = 0, glut = 0, pier = 0, braz = 0;

  for (let i = 0; i < this.exercises.length; i++) {
    switch(this.exercises[i][2]) {
      case 'Flexibilidad':
        flexArray[flex++] = this.exercises[i].slice()
        break
      case 'Abdominales':
        abdArray[abd++] = this.exercises[i].slice()
        break
      case 'Glúteos':
        glutArray[glut++] = this.exercises[i].slice()
        break
      case 'Piernas':
        pierArray[pier++] = this.exercises[i].slice()
        break
      case 'Brazos':
        brazArray[braz++] = this.exercises[i].slice()
        break
    }
  }

  let j = 0;
  for (let i = 0; i < flexArray.length; i++) {
    this.exercises[j++] = flexArray[i]
  }
  for (let i = 0; i < abdArray.length; i++) {
    this.exercises[j++] = abdArray[i]
  }
```

FIGURA 55: FRAGMENTO DE CÓDIGO PARA REALIZAR EL ORDENAMIENTO POR CATEGORÍA

4.6.3 'Tabla de Retos'

En esta página el administrador puede visualizar una tabla donde aparecen los datos principales de cada reto:

- Nombre.
- Categoría (senderismo, natación, atletismo, ciclismo…).
- Puntuación mínima necesaria.
- Dificultad.
- Descripción.
- Distancia.

Adicionalmente, el administrador podrá editar o eliminar un reto pulsando en los botones que se encuentran en una columna adicional. Por otra parte, el administrador también podrá ordenar los retos por categoría o puntuación y añadir uno nuevo.

Maquetación

- Sección "row": esta sección contiene una fila con tres botones que sirven para ordenar la tabla de retos por categoría o puntuación, y añadir un nuevo reto a la base de datos. Cuando se hace *"click"* en el botón "Filtrar por categoría" o "Filtrar por puntuación" se

llama a la función encargada del ordenamiento de las filas de la tabla. Por otra parte, si se pulsa el botón "+ Añadir Reto" se desplegará una ventana "*modal*" para rellenar los datos del nuevo reto.

- Sección "table": esta sección está formada por una tabla que contiene los datos de cada reto. Cada dato es accedido mediante la directiva "**ngFor*" que permite recorrer el array bidimensional y mostrarlos en la tabla a través de la interpolación "{{*array*[0]}}".
- Sección "ng-template editar": cuando el administrador o administradora quiere editar los datos de un reto, se le desplegará una ventana modal al pinchar en "Editar". Esta ventana está formada por diferentes filas compuestas por "*inputs*" que permiten modificar los datos de cada reto. El botón "Editar" será habilitado cuando los datos sean correctos como si se tratara de un formulario.
- Sección "ng-template añadir": cuando el administrador o administradora quiere añadir un nuevo reto a la base de datos, se le desplegará una ventana "*modal*" al pinchar en "+ Añadir Reto". Esta ventana está formada por diferentes filas compuestas por "*inputs*" que permiten rellenar los datos de un reto. El botón "Añadir" será habilitado cuando los datos sean correctos como si se tratara de un formulario.

Lógica

En la clase "*ChallengeAdmComponent*", utilizamos el método *"ngOnInit"* para poder leer los datos de cada reto que se encuentra en la base de datos. Estos datos serán almacenados en un "*array*" bidimensional para mostrar su contenido en la tabla. A continuación, vamos a detallar cada uno de los métodos que forman parte de esta clase:

- Método "openModal_Edit": este método permite establecer en el "*modal*" los valores que corresponden al reto que se quiere editar, y además sirve para mostrar el "*modal*".
- Método "openModal_Add": este otro método tiene una función muy similar al anterior con la diferencia que, al mostrar el "*modal*", se vacía el valor que contienen las variables que se muestran en el cada "*input*".
- Método "removeRepeatedChallenges": debido a que en la base de datos la colección de retos está formada por documentos con un identificador de usuario, es necesario eliminar los retos que se repitan, mediante un bucle "*for*".
- Método "checkModalValues": este método comprueba los valores que se han introducido en los "*inputs*" del modal independientemente de si se ha añadido o editado un reto. También, se utiliza la propiedad "two way binding".
- Método "orderByCategory": este método ordena las filas de la tabla a partir de la categoría. No obstante, la lógica está basada en el mismo método de la página anterior debido a que se recorre el array bidimensional que contiene los retos, y almacena en diferentes *"arrays"* cada uno, según la categoría. Este ordenamiento se lleva a cabo mediante un *"switch"*. Una vez que se ha realizado esta operación, se añade al "*array*" de ejercicios en el siguiente orden: flexibilidad, senderismo, natación, atletismo y ciclismo.
- Método "orderByScore": este método ordena las filas de la tabla mediante el ordenamiento por burbuja a partir de la puntuación mínima necesaria de cada reto.

- Método "addChallenge": con esta función se accede a la base de datos, y se añaden los valores introducidos en los "*inputs*" del modal en un nuevo documento de la colección de retos. Además, también se añaden los datos al "*array*" bidimensional de retos. Cuando el reto es añadido correctamente, se mostrará un breve mensaje de que la operación se ha realizado con éxito o con error si se diera el caso. Todos los cambios realizados supondrán añadir nuevos documentos del reto añadido del resto de usuarios.
- Método "updateChallenge": esta función es muy similar a la anterior con la única diferencia que se accede a la base de datos a partir del identificador del reto proporcionado por parámetro, para editar su contenido. Además, también actualiza los datos al "*array*" bidimensional de retos. Cuando el reto es editado correctamente, se muestra un breve mensaje de que la operación se ha realizado con éxito o con error si se diera el caso. Todos los cambios realizados supondrán una modificación de los documentos del reto editado del resto de usuarios.
- Método "deleteChallenge": esta función accede a la base de datos a partir del identificador del reto proporcionado por el reto pasado por parámetro, y elimina el reto actual. Además, también vacía los datos del "*array*" bidimensional del reto eliminado gracias a su identificador único. Todos los cambios realizados supondrán una eliminación de los documentos del reto eliminado del resto de usuarios.
- Método "onLogout".

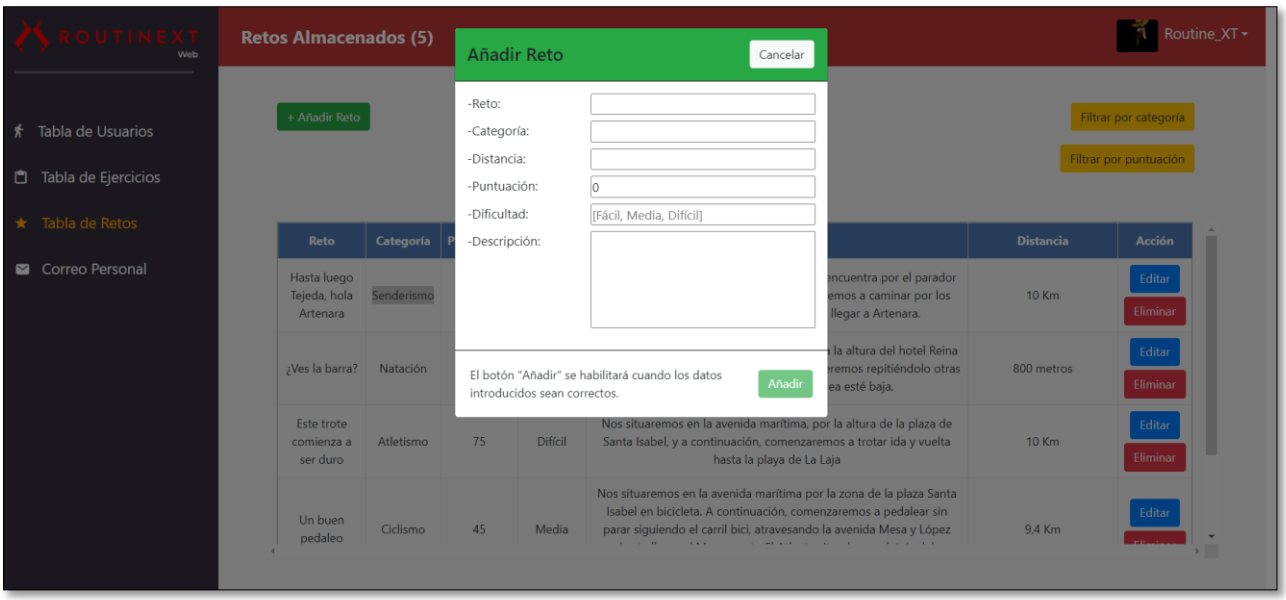


FIGURA 56: MUESTRA DE LA PÁGINA *"TABLA DE RETOS"*

4.6.4 'Mi Perfil'

Esta página es una versión adaptada al administrador/a de la página de "Mi Perfil" de la vista de cliente. El administrador podrá visualizar los siguientes datos de su perfil:

- Nombre.
- Correo Personal.
- Foto de Perfil.

Adicionalmente, el administrador/a podrá cambiar también su contraseña. Debemos recordar que el administrador no puede visualizar retos, puntuaciones o rutinas debido a que no comparte las propiedades que puede tener un cliente de la aplicación.

Maquetación

- Sección "col-4": en esta sección se utiliza un elemento "*card*" para poder mostrar la foto del perfil del administrador/a. Esto se realizará con la propiedad "*binding*" que permite mostrar una imagen a través de su dirección contenida en una variable.
- Sección "col-2": en esta sección podemos encontrar un par de párrafos donde mostramos el nombre y el correo del administrador.

Lógica

En la clase *ProfileAdmComponent*, se obtienen todas las propiedades del administrador de la base de datos, almacenándolos en un "*array*" bidimensional para que puedan ser mostrados en la vista del usuario. La clase está formada solo contiene el siguiente método:

- Método "onLogout".

Si un usuario con el rol de cliente intenta acceder a la vista de un administrador, será redirigido a una página de su vista. De forma alternativa, si un administrador intenta acceder a la vista de un cliente, también será redirigido a una página de su vista.

4.6.5 'Editar Perfil'

En esta página el administrador/a podrá editar los datos que puede visualizar en la página "Mi Perfil":

- Nombre.
- Correo Personal.
- Foto de Perfil.

De igual forma como ocurre con un cliente, cuando el administrador edite algún dato, la página se refrescará automáticamente para visualizar los cambios que han sido realizados en sus datos.

Maquetación

- Sección "col-4": esta sección es idéntica a la que aparece en la página "Mi Perfil" con la diferencia que se utiliza el elemento "*progressbar*" para poder mostrar una barra de progreso en relación con la subida de una nueva foto de usuario. Cuando la foto se actualice, la página se refrescará automáticamente mostrando el perfil de usuario.
- Sección "col-2": esta sección sí difiere de la página de "Mi Perfil", debido a que se hace uso de los elementos "*formGroup*" y "*formControlName*" debido a que los datos del usuario se muestran en un formulario que puede modificarse y requiere la validación de

los datos que se introducen en él. Si algún dato no es correcto, se mostrará debajo del campo del formulario un mensaje de error para que el usuario pueda corregirlo.

- Sección "col-3": esta sección es exactamente idéntica a la que aparece en la página "Mi Perfil" mostrando las estadísticas del usuario.

Lógica

La clase "*EditProfileAdmComponent*" realiza una lectura de los datos de usuario que se encuentran en la base de datos guardando los datos del usuario en un "*array*" bidimensional. Además, el método *"ngOnInit"* permite crear un formulario de validación para poder verificar que los datos que ha introducido el usuario son correctos. A continuación, vamos a detallar las diferentes funciones que se han implementado en esta clase:

- Método "updateImg": este método se utiliza cuando el usuario sube una nueva imagen para actualizar su foto de perfil. Para ello, es necesario acceder a la carpeta que se encuentra en la base de datos (en "*Firebase Storage*"), para poder guardar la imagen. Si la imagen ha sido subida correctamente, se guardará también la URL que ha sido generada en una de las propiedades de la colección del usuario autenticado. Además, se mostrará una barra de progreso para poder actualizar la página finalice la subida de la imagen.
- Método "onUpdate": esta función solo sirve para poder actualizar los datos del usuario accediendo a la colección de usuarios para actualizar el nombre, el apellido, la edad y el correo mediante la función "*update*" que proporciona "*Firebase*".
- Método "onLogout".

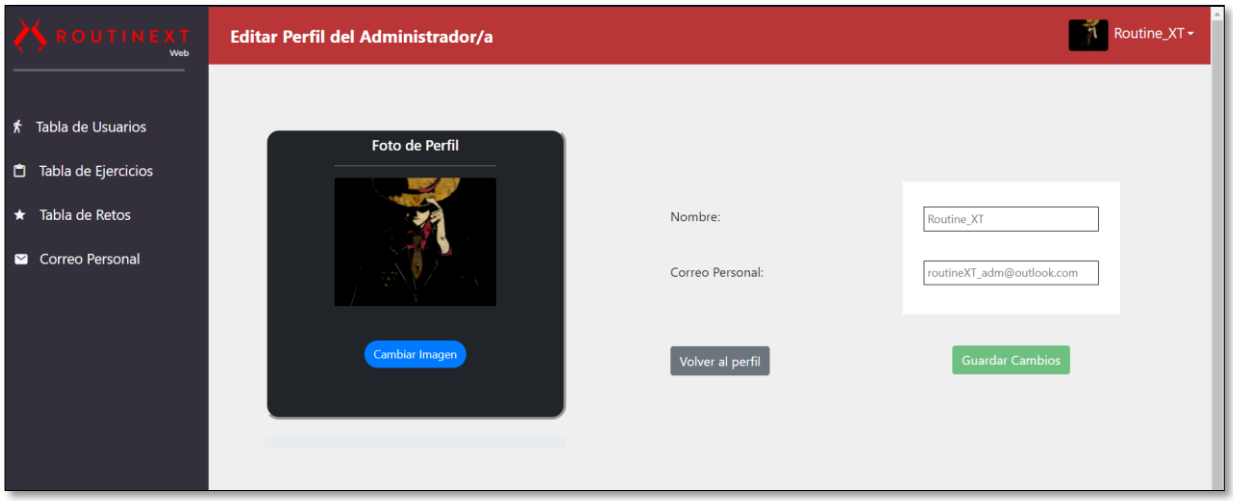


FIGURA 57: MUESTRA DE LA PÁGINA *"EDITAR PERFIL"*

Capítulo 5. Adaptación de la Aplicación Móvil a Web

Al inicio del proyecto la idea era desarrollar una aplicación web y móvil complementarias para este TFT. Sin embargo, debido a diversos problemas de tiempo, no se ha podido desarrollar la aplicación móvil. En este capítulo, vamos a explicar los factores que han impedido desarrollar la aplicación móvil que había sido planificada desde el inicio del proyecto, y cuáles han sido las soluciones que se han llevado a cabo para que la aplicación web sea lo más completa posible.

Además, también explicaremos los componentes que han sido añadidos a la vista de cliente a nivel de maquetación y lógica, de la misma forma que han sido explicados en el capítulo previo.

5.1 Contexto

En el documento *"TFT-01"*, se explica que la intención es realizar una aplicación móvil para el sistema operativo "*Android*" [40], en el lenguaje de programación "*Kotlin*" [41]. Una de las cuestiones por las que se decidió incluir esta aplicación fue para el alumno pudiera adquirir sus primeros conocimientos acerca de cómo realizar aplicaciones para móviles. Además, se había buscado y adquirido un curso de formación introductorio para desarrollar aplicaciones en "*Android*" haciendo uso de *"Kotlin" y "Firebase"*: *"The Complete Android 10 & Kotlin Development Masterclass"* [42]. También, se llegó a crear un prototipo funcional en *"Adobe XD"* para poder visualizar el posible aspecto de la interfaz de usuario y las funcionalidades que se iban a incluir en la aplicación móvil. Estas funcionalidades se identificaron planificando ocho historias de usuario con una duración estimada de 48 horas en total.

Previamente a la planificación, se pensó en la posibilidad de que pudiera ocurrir un posible contratiempo en el desarrollo de ambas aplicaciones. No obstante, debido a que a lo largo del cuatrimestre éste era el único proyecto que se estaba realizando (única asignatura), no se consideró que pudiera haber problemas de tiempo.

El problema surgió cuando no se fue cumpliendo la estimación de tiempo planificada en cada parte del proyecto. El proyecto comenzó su realización la última semana del mes de enero con el curso de formación.

En la tabla 9 podemos visualizar las duraciones de cada parte:

Etapa del Proyecto	Duración Estimada	Duración Real	Diferencia
Curso de Formación Web.	2 semanas	3 semanas	+ 1 semana
Prototipado y planificación de las historias de usuario de la vista de usuario no registrado.	1 semana	1 semana	-
Sprint 1 (usuario no registrado)	2 semanas	3 semanas	+ 1 semana
Prototipado y planificación de las historias de usuario de la vista de cliente	1 semana	1 semana	-
Sprint 2 (cliente)	2 semanas	4 semanas	+ 2 semanas
Sprint 3 (cliente)	2 semanas	2 semanas	-
Prototipado y planificación de las historias de usuario de la vista de administrador	1 semana	1 semana	-
Sprint 4 (administrador)	1 semana	1 semana	-
Prototipado y planificación de las Historias de Usuario de la vista de cliente en la aplicación móvil.	1 semana	1 semana	-
Prototipado y planificación de las Historias de Usuario de la adaptación de la aplicación móvil a web	1 semana	1 semana	-
Sprint 5 (adaptación)	2 semanas	2 semanas	-

TABLA 9: DURACIÓN DEL PROYECTO

En esta tabla se puede apreciar cómo a medida que se ha ido desarrollando el proyecto, ha habido etapas que no se han podido realizar en el tiempo estimado suponiendo un retraso de cuatro semanas que han impedido el desarrollo de la aplicación móvil. A lo largo del desarrollo del proyecto, se ha intentado simular un entorno de trabajo real, trabajando alrededor de ocho horas diarias. No obstante, en la realidad no ha sido así, hay días en los que se han trabajado más de 10 horas, otros días en los que se han trabajado cinco horas, y otros días en los que se ha descansado. Una cifra realista ha sido de unas seis horas diarias, cinco días a la semana (30 horas semanales aproximadamente).

La primera semana de mayo, justo al poco tiempo de finalizar la vista de administrador, se acordó que la fecha para entregar la documentación del TFT sería entre el 21 de mayo y el 11 de junio. Justo esa semana, se habían realizado aproximadamente 360 horas de TFT, incluyendo todas sus partes (implementación, planificación, memoria, prototipado etc.). En la realización del prototipado y la planificación de las historias de usuario de la aplicación móvil, se tuvo que sumar la duración estimada de 48 horas a un tiempo de al menos 40 horas para realizar el curso de formación que comentamos anteriormente. Por lo tanto, no hubiera sido posible de ninguna manera finalizar el desarrollo de la aplicación móvil a tiempo.

5.2 Solución

La primera semana de mayo, comenté con mi tutor el problema que había surgido respecto al alcance del proyecto. La solución que acordamos fue justificar tanto en la defensa como en la memoria, el cambio que había tenido que realizar al no poder incluir la aplicación móvil. Debido a que el documento "*TFT-01*" no se puede modificar, hice un cambio de título mediante el documento *"TFT-03"*.

No obstante, yo personalmente quise añadir las funcionalidades de la aplicación móvil al entorno web porque, desde mi punto de vista, la aplicación web no estaba suficientemente completa sin demostrar ser un producto mínimo viable. Esto fue debido a que el sistema de puntuación de la aplicación web no había sido incluido porque era un aspecto que se tenía pensado implementar en la aplicación móvil. Por otra parte, todas las funcionalidades de la aplicación móvil se podían realizar en la aplicación web, debido a que muchas de ellas ya estaban implementadas sin necesidad de realizar un curso de formación extra para poder llevarlo a cabo. Esto se pudo implementar en tan solo dos semanas cumpliendo con los plazos. Si se hubiera hecho la aplicación móvil, se hubieran necesitado más de cuatro semanas para poder desarrollarla con el curso de formación.

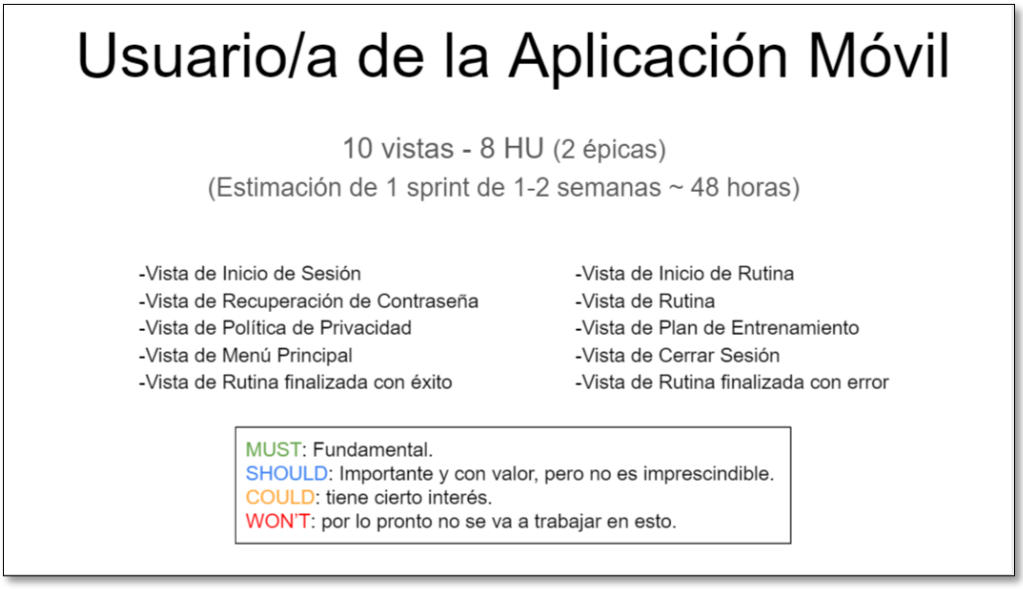


FIGURA 58: MUESTRA DE LA PRIMERA PÁGINA DE PLANIFICACIÓN DE LA APLICACIÓN MÓVIL

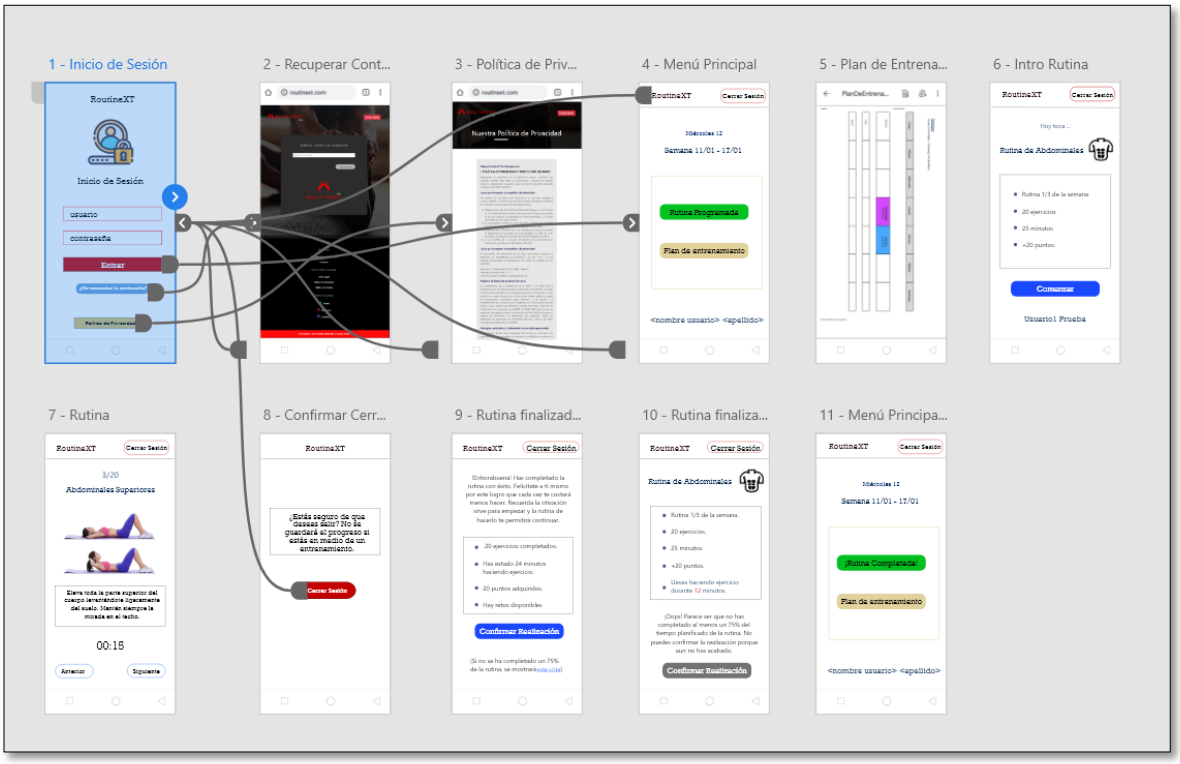


FIGURA 59: MUESTRA DEL PROTOTIPO DE LA APLICACIÓN MÓVIL

Las funcionalidades de la aplicación móvil se han implementado añadiendo una nueva sección en la vista de cliente, que permite realizar una rutina con sus ejercicios actualizando el sistema de puntuación. A continuación, vamos a explicar cómo se ha desarrollado esta nueva sección.

5.3 Vista de cliente (realización de una rutina)

En esta sección el cliente puede realizar las siguientes funciones:

- **Visualizar a una rutina programada:** cuando el usuario acceda a la página "Comenzar_Rutina", podrá iniciar una rutina programada en el plan de entrenamiento siempre que la hora actual esté dentro del rango de tiempo de realización de la rutina.
  Si el usuario accede a esta página fuera del rango de tiempo de realización de una rutina, se le mostrará un mensaje de advertencia sin posibilidad de acceder a la misma.
- **Realizar una rutina:** si el usuario decide comenzar la rutina, se mostrará una nueva página con una apariencia diferente mostrando los ejercicios con sus descripciones de la rutina. También, se mostrará el tiempo empleado en su realización.
- **Cambiar de ejercicio:** el usuario podrá en cualquier momento cambiar un ejercicio para realizar uno posterior o anterior al actual.
- **Pausar una rutina:** el usuario puede en cualquier momento pausar una rutina para que el tiempo de realización sea pausado.

- **Finalizar una rutina sin finalizarla:** el usuario puede en cualquier momento salir de una rutina sin terminarla renunciando a la actualización de sus puntos.
- **Completar una rutina:** el usuario puede actualizar sus puntos cuando finaliza una rutina. Para poder terminar una rutina, el sistema de puntuación debe asegurar que el usuario ha completado al menos un 75% de tiempo programado de realización de una rutina.

Páginas Web	Tipo de usuario	Sprint	Historias de Usuario
-Página de Inicio de Rutina. -Página de Rutina.	Cliente	5	6 HU (1 épica) (61-66)
Duración Estimada (inicial)		**Duración Real**	
Sprint 5: 2 semana (10 Mayo - 21 Mayo) ~ 38 horas		**Sprint 4:** 2 semanas (10 Mayo - 23 Mayo) = 40 horas	

TABLA 10: PLANIFICACIÓN DE LA VISTA DE CLIENTE (REALIZACIÓN DE UNA RUTINA)

5.3.1 'Inicio de Rutina'

Para acceder a esta página se ha añadido una nuevo enlace en el panel del "*dashboard*", donde el usuario podrá hacer *"click"* en la pestaña "Comenzar Rutina".

Maquetación

- Sección "content": esta única sección está dividida en dos partes dependiendo si se debe mostrar la descripción de una rutina programada o no, en el caso que la hora actual no coincida con el tiempo de realización de la rutina.

  - Mostrar Rutina: esta parte estará dividida en dos filas. Primeramente, se mostrará un título que muestre la rutina programada mediante un párrafo. A continuación, se mostrará en la segunda fila un cuadro donde se puede visualizar información de la misma mediante una lista desordenada (puntuación, tiempo de realización, número de ejercicios, tiempo de descanso, número de rutina de la semana etc.).
  - Mostrar Mensaje de Advertencia: si no se muestra la rutina porque no está planificada para la hora actual, se mostrará un párrafo seguido de una imagen para que el usuario sepa que no pude comenzar una rutina.

Lógica

En la clase "*StartRoutineComponent*", utilizamos el método *"ngOnInit"* para poder obtener todas las rutinas del usuario autenticado. A continuación, las ordenamos en un array bidimensional para poder descartar aquellas cuya fecha y hora sean posteriores al día y hora actual. Una vez realizado esto, almacenamos en un array de una sola dimensión los datos de la rutina comprendida en la hora actual. Ahora, vamos a detallar cada uno de los métodos que forman parte de esta clase:

- Método "convertMinutesToHours": este método recibe una unidad de tiempo determinada en minutos y transforma su contenido en horas y minutos respetando un formato adecuado.
- Método "private sort": este método tiene una lógica bastante compleja porque mediante una estructura *"switch"* recorre el array que contiene todas las rutinas del usuario y transforma la *"string"* que contiene la hora de la rutina, en minutos para poder compararla con el resto de rutinas del día actual. Este proceso se repite para cada día de la semana, de forma que una vez se ha realizado esta transformación se hace una llamada al método "orderBytime" para poder ordenar cada rutina comparando el tiempo de comienzo y finalización de cada una. Finalmente, se añade por orden del día de la semana las rutinas ordenadas al array bidimensional inicial que contenía todas las rutinas del usuario tal cuál estaban almacenadas en la base de datos.
- Método "orderBytime": este método recibe un array que contiene todas las rutinas de un determinado día de la semana. A continuación, se realiza una ordenación de este priorizando aquellas rutinas cuyo tiempo de realización sea previo a otra rutina.
- Método "onLogout".

```
107   private convertMinutesToHours(time: number) {
108     let hours = Math.floor(time/60)
109     let minutes = time % 60
110     let h   = ('0' + hours).slice(-2)
111     let m   = ('0' + minutes).slice(-2)
112     return h+":"+m
113   }
```

FIGURA 60: FRAGMENTO DE CÓDIGO PARA OBTENER EL TIEMPO EN HORAS Y MINUTOS

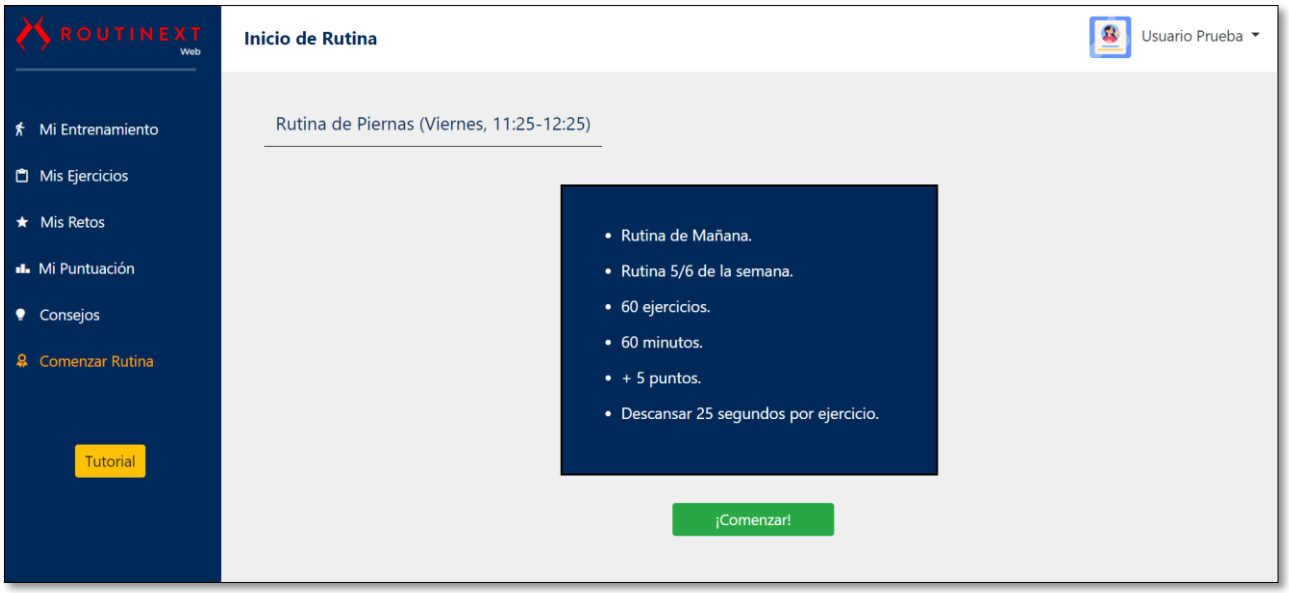


FIGURA 61: MUESTRA DE LA PÁGINA "*COMENZAR_RUTINA*" (CON RUTINA DISPONIBLE)

FIGURA 62: MUESTRA DE LA PÁGINA "*COMENZAR_RUTINA*" (SIN RUTINA DISPONIBLE)

5.3.2 'Rutina'

Para acceder a esta página se el usuario debe hacer "*click*" en el botón "Comenzar Rutina", que se muestra en la página "Comenzar_Rutina".

Esta página ha sido dividida por primera vez en dos componentes: "*routine*" y "*routine-content*". El propósito ha sido simplificar la lógica debido a que cada ejercicio se muestra mediante un elemento que permite realizar una paginación, de manera que "*routine*" contenga aquellos elementos estáticos de la página como la cabecera y el fondo, mientras que "*routine-content*" contenga cada uno de los ejercicios que se muestran.

Componente "routine"

Maquetación

- Sección "container-fluid": esta única sección tan solo está formada por un párrafo que muestra la rutina que se está realizando y un botón que permite finalizar la rutina.
- Sección "content": esta única sección tan solo se encarga de oscurecer el fondo donde se mostrarán cada uno de los ejercicios. Gracias a la directiva "*<app-routine-content>*", se puede mostrar el componente "*routine-content*".

Lógica

En la clase "*RoutineComponent",* utilizamos el método *"ngOnInit"* para poder obtener todas las rutinas del usuario autenticado. A continuación, las ordenamos en un array bidimensional para poder descartar aquellas cuya fecha y hora sea posterior al día y hora actual. Una vez realizado esto, almacenamos en un array de una sola dimensión los datos de la rutina comprendida en la hora actual. Este método es exactamente al comentado en la página anterior, con la única diferencia que si se detecta que el usuario inicia una rutina que ha finalizado (si el usuario permaneciera en la página de "Comenzar_Rutina mucho tiempo"), se le vuelve a redirigir a la página anterior al no poder iniciar una rutina cuyo tiempo de realización ha acabado. A continuación, vamos a detallar cada uno de los métodos que forman parte de esta clase:

- Método "convertMinutesToHours": este método es exactamente idéntico al comentado en la sección anterior.
- Método "private sort": este método es exactamente idéntico al comentado en la sección anterior.
- Método "orderBytime": este método es exactamente idéntico al comentado en la sección anterior.
- Método "showConfirmMessage": este método se ejecuta cuando el usuario quiere finalizar la rutina sin acabarla mostrando un mensaje mediante el módulo "*sweetalert2*" para incentivar que continúe realizándola. En el caso de que finalice la rutina, se mostrará la página "Comenzar_Rutina".
- Método "onLogout".

```
showConfirmMessage() {
  Swal.fire({
    title: '¡No te rindas!',
    text: "Si finalizas el entrenamiento no se guardará tu progreso. ¿Deseas continuar?",
    icon: 'error',
    showCancelButton: true,
    cancelButtonColor: '#3085d6',
    cancelButtonText: 'Cancelar',
    confirmButtonColor: '#d33',
    confirmButtonText: 'Salir sin guardar',
    backdrop: `
        rgba(0, 0, 0, 0.4)
        url("assets/registered/angry_img2.gif")
      `
  }).then((result) => {
    if(result.isConfirmed) {
      this.router.navigate(['/Comenzar_Rutina'])
    }
  })
}
```

FIGURA 63: FRAGMENTO DE CÓDIGO PARA MOSTRAR EL MENSAJE QUE PERMITE FINALIZAR UNA RUTINA

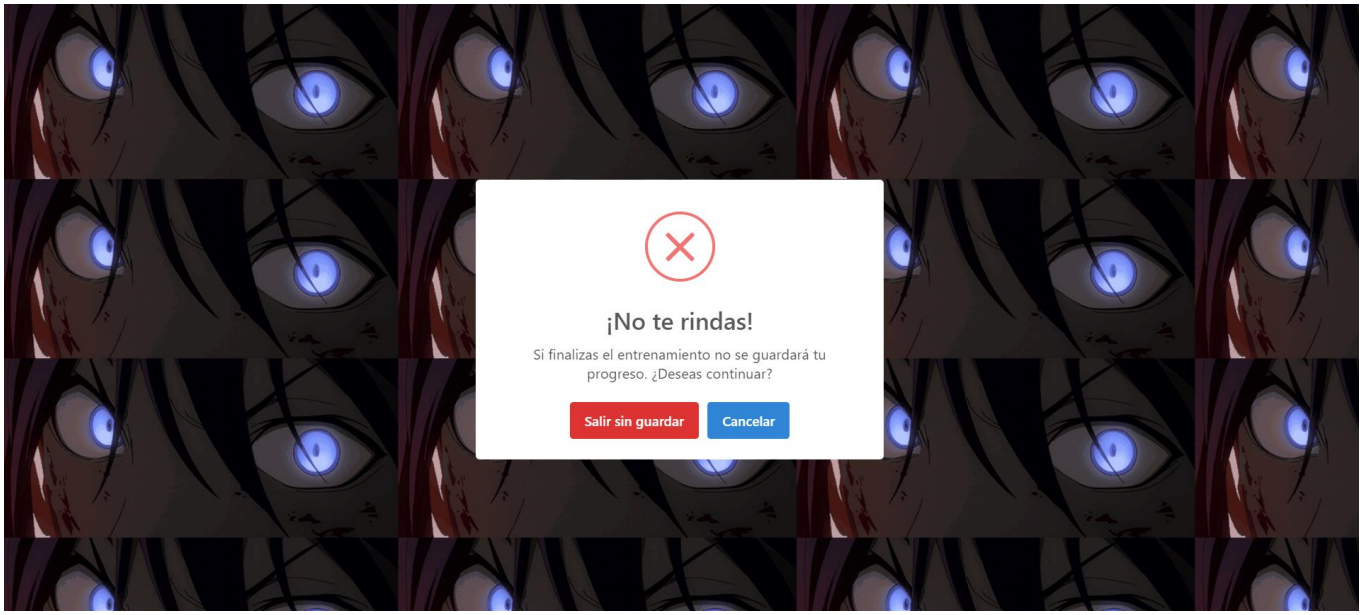


FIGURA 64: MUESTRA DEL MENSAJE PARA FINALIZAR EL ENTRENAMIENTO

Componente "routine-content"

Maquetación

- Sección "row mt-5 mx-4": en esta sección se muestra en una fila con una columna dos botones que permiten mostrar información acerca de cómo poner el modo de pantalla completa que ofrece el navegador para lograr una mayor interactividad. También, se

muestra el tiempo empleado en realizar la rutina. Ambos botones contienen un elemento "*tooltip*", que permite mostrar el texto "Ocultar" al pasar el ratón justo por encima del botón sin pulsar en él. Estos botones se minimizan al pinchar en ellos por si el usuario considera que no deben mostrarse durante la realización de la rutina.

- Sección "row bg-white mt-4 mx-auto": en esta sección se recorre cada ejercicio que forma parte de la rutina actual mediante la directiva "**ngFor*" para mostrar el título, imagen y descripción mediante interpolación. Al mismo tiempo se realizará la paginación, donde cada ejercicio corresponde a una página.
  Esta sección se encuentra dividida en tres columnas, donde mostramos mediante párrafos el tiempo de realización del ejercicio (un minuto), junto con su tiempo de preparación (25 segundos), el nombre del ejercicio y el número de ejercicio que se está realizando.

- Sección "row bg-white mx-auto": en esta sección simplemente se muestran en dos columnas, la imagen del ejercicio junto con su descripción.
  Justo debajo, podemos encontrar otra fila con dos columnas donde podemos pausar o reanudar el entrenamiento mediante un botón o controlar el ejercicio que estamos realizando a través de la paginación.

- Sección "row bg-white mx-auto adicional": esta sección solo se muestra cuando el usuario se encuentra realizando el último ejercicio y aún no ha completado el 75% del tiempo de realización de la rutina. Se muestra el botón "Continuar" deshabilitado y un mensaje de advertencia contenido en un párrafo junto con el tiempo en minutos que se necesitan para alcanzar el 75%.

Lógica

En la clase "*RoutineContentComponent",* utilizamos el método *"ngOnInit"* de forma similar a como lo hemos utilizado en las secciones previas debido a que se necesitamos conocer la rutina que se ha planificado para la hora actual. Es necesario que cuando se esté realizando no finalice abruptamente si se requiere más tiempo para completarla. Además, en este método se almacena en un array bidimensional los ejercicios que conforman la rutina actual, y se inicia el temporizador que muestra el tiempo de realización de la rutina y del ejercicio. A continuación, vamos a detallar cada uno de los métodos que forman parte de esta clase:

- Método "hideButton": este método trata de ocultar o mostrar el botón que contiene el tiempo empleado en realizar la rutina mediante una expresión ternaria.

- Método "hideRow": este método realiza lo mismo que el método anterior, pero con el botón que muestra cómo establecer la página en pantalla completa en el navegador.

- Método "startTimer": este método hace uso de la función interna "*setInterval*" para poder mostrar un temporizador acerca del ejercicio que se está realizando, incluyendo el tiempo de preparación y de realización.

- Método *"endTimer"*: este método es llamado cuando el usuario cambia de ejercicio para poder cancelar los temporizadores que contabilizan cada ejercicio que se está realizando mediante la función interna "*clearInterval*". Además, se comprobará si el usuario ha realizado el 75% del tiempo de la rutina.

- Método "startChrono": este método se ejecuta al comenzar la rutina de manera que con la función "*setInterval*", simula un cronómetro.

- Método "stopTimers": la implementación de este método es muy similar a "*hideButton*" debido a que pausa el temporizador y cronómetro de la aplicación dependiendo de un valor booleano.

- Método "endRoutine": este método se ejecuta cuando el usuario finaliza con éxito una rutina actualizando el contenido de la base de datos. La tabla "Rutinas" se actualizará de forma que se almacene que la rutina ha sido realizada, y la tabla "Usuarios" actualizará la puntuación del usuario.
  Para finalizar la rutina la aplicación redirigirá al usuario a la página "Sistema de Puntuación", donde se mostrará un mensaje de felicitación con información de los logros obtenidos mediante el módulo "sweetalert2".

- Método "convertMinutesToHours": este método es exactamente idéntico al utilizado en las secciones anteriores.

- Método "private sort": este método es exactamente idéntico al utilizado en las secciones anteriores.

- Método "orderBytime": este método es exactamente idéntico al utilizado en las secciones anteriores.

```
181   startChrono() {
182     let intervalTime, minutes = 0, seconds = 0
183     intervalTime = setInterval(() => {
184       if(!this.stopTimer) {
185         if(seconds < 60) {
186           seconds++;
187           if(seconds == 60) {
188             seconds = 0;
189             minutes++
190             this.mm = minutes < 10 ? "0" + String(minutes) : String(minutes)
191           }
192           this.ss = seconds < 10 ? "0" + String(seconds) : String(seconds)
193         }
194       }
195     }, 1000)
196   }
```

FIGURA 65: FRAGMENTO DE CÓDIGO PARA COMENZAR A CONTAR LA DURACIÓN DE LA RUTINA

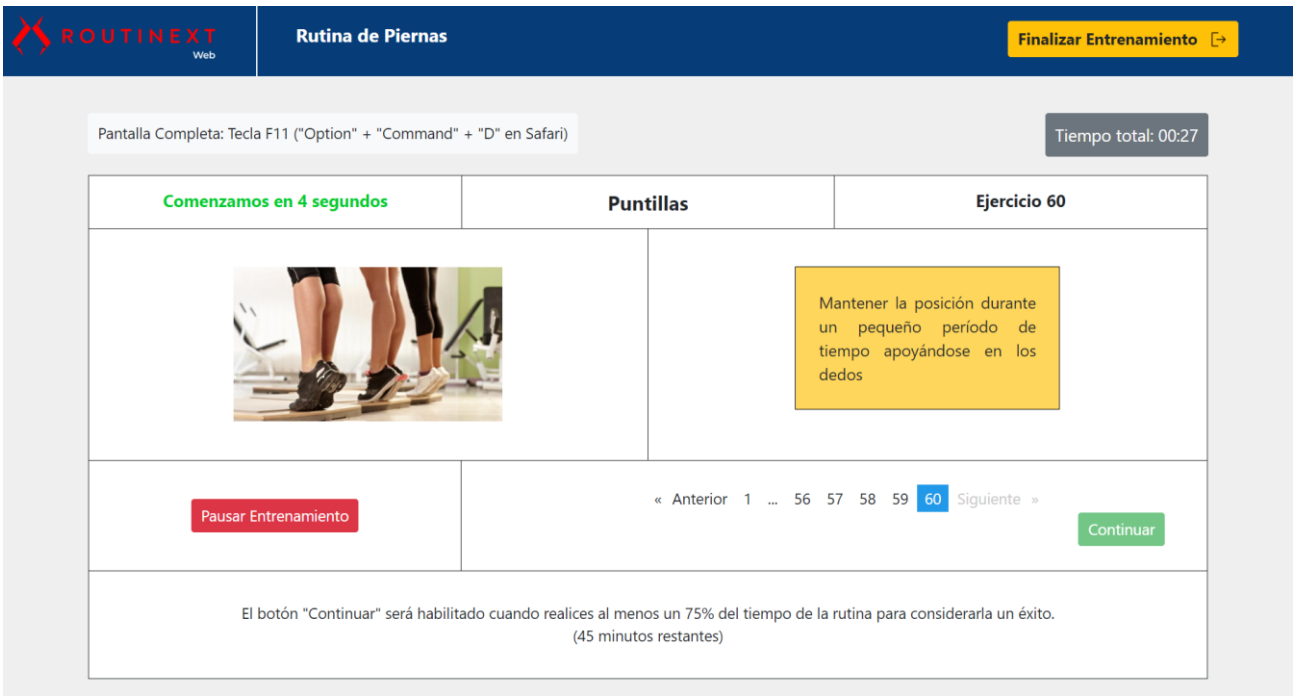


FIGURA 66: MUESTRA DE LA PÁGINA *"RUTINA"*

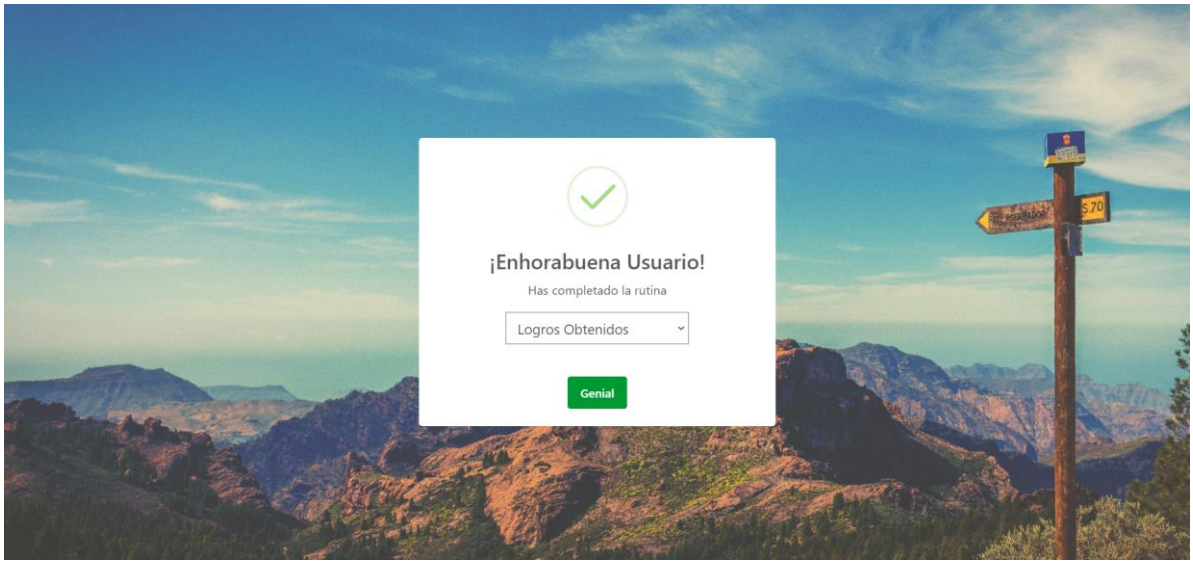


FIGURA 67: MUESTRA DEL MENSAJE DE FINALIZACIÓN DE UNA RUTINA

Capítulo 6. Conclusiones y Trabajos Futuros

6.1 Conclusiones

El desarrollo de este proyecto se ha podido realizar de forma satisfactoria, a pesar de los impedimentos de tiempo que han surgido. Inicialmente de un total de 68 historias de usuario, se han podido completar 66. Ha sido esencial haber realizado un curso de formación previo porque ha permitido adquirir los conocimientos mínimos necesarios para poder realizar este trabajo.

Me gustaría resaltar la necesidad de haber realizado un prototipo funcional de cada parte de la aplicación porque ha ayudado mucho a establecer el alcance de este trabajo pudiendo deducir las funcionalidades que se han realizado mediante historias de usuario. El uso de "*Scrum*" ha sido muy importante porque ha permitido establecer fechas para lograr incrementos del producto en los que se ha estado trabajando. Las historias de usuario han contribuido también a completar los incrementos debido a que se han seleccionado en cada "*Sprint*" un determinado número de funcionalidades para implementar en el tiempo de duración de un "*Sprint*" concreto.

El producto final corresponde a una aplicación que permite a un usuario realizar la función principal (realizar ejercicio), con una serie de añadidos (sistema de puntuación, retos, ranking) que intentan incentivar al usuario un uso constante. Además, que la aplicación sea exclusivamente páginas web tiene sentido, pues en el estudio de mercado que se hizo antes de comenzar su desarrollo, se demostró cómo existen aplicaciones de éxito que no están desarrolladas exclusivamente para dispositivos móviles.

Respecto al desarrollo, he podido obtener conocimientos acerca de tecnologías web que no había adquirido, logrando más experiencia en los lenguajes de programación con los que tenía un contacto previo: "*HTML*", "*CSS*" y "*Javascript*". Me ha sorprendido gratamente que la estructura de "Angular" permita dividir una aplicación en componentes, de manera que cada uno esté conformado por sus propios ficheros. Esto me ha ayudado mucho a separar completamente el código que forma parte de la maquetación de la lógica de desarrollo.

Por otra parte, el uso de "*Firebase*", me ha permitido conocer cómo se estructura esta base de datos logrando ser capaz de escribir, leer datos, crear registros o colecciones para poder realizar las diferentes partes de la aplicación. Además, lo mejor que he visto, ha sido la facilidad con la que se puede conocer si en la base de datos se ha modificado, añadido o eliminado algún campo o valor debido a que muestra en un color determinado la acción que se ha hecho sobre un determinado registro, documento o colección.

También he logrado introducirme a la biblioteca de "*Bootstrap*", que es una tecnología que jamás había utilizado anteriormente, pero que me ha permitido agilizar en gran medida todas las cuestiones relativas al estilo y apariencia de la aplicación. También, he logrado saber utilizar e integrar el uso de módulos como "*sweetalert2*" y otras bibliotecas como "*Ionicons*" y "*Fontawesome*", además de herramientas de planificación como "*Trello*", o "*Google Sheets*". El uso de "*Adobe XD*" ha sido fantástico porque la propia herramienta facilita la elaboración de prototipos debido a su gran facilidad de uso.

Pienso que la aplicación ha logrado todos los objetivos iniciales. Mis objetivos a nivel personal los he cumplido:

- Obtener conocimientos de nuevas tecnologías web.
- Realizar una aplicación que visualmente destaque manteniendo una apariencia uniforme.
- Realizar una aplicación con una interfaz de usuario que intente ser moderna y clara ante cualquier persona.
- Lograr realizar al menos un 90% de todas las funcionalidades que quería incluir en mi aplicación para realizar ejercicios en casa añadiendo características similares a las de un videojuego para lograr un interesante sistema de gamificación. (recompensas, retos con diferentes dificultades, pérdida de puntos si no se logran los resultados esperados, planificar una tarea y que al refrescar se muestre etc.)
- Intentar que la aplicación web sea lo más interactiva posible desde el punto de vista del usuario. (apariencia visual con diferentes colores, botones, modales etc.)
- Ser capaz de interactuar con la base de datos a través de la interfaz de usuario.
- Lograr realizar algunas tareas de administración (editar un usuario, añadir un ejercicio, etc.).

En conclusión, estoy muy contento por haber sido capaz de haber logrado este resultado de forma totalmente autodidacta. No obstante, me gustaría también comentar posibles mejoras a lo largo del desarrollo:

1) **Exceso de funcionalidades e historias de usuario**: cambiar contraseñas, buscar herramientas para recibir correos a través de un formulario, crear páginas de políticas de cookies, privacidad, aviso legal, consejos de entrenamiento, ciertas funciones del administrador, tareas de impresión y el esfuerzo de lograr una interfaz de usuario trabajada.
   Es algo difícil determinar a priori de qué funcionalidades se pueden prescindir porque pueden impedir realizar las tareas en el tiempo estimado o no ser tan importantes.

2) **Posiblemente muy ambicioso para tener un tiempo limitado:** a pesar del gran número de funcionalidades y esfuerzo en lograr una interfaz muy trabajada, querer también realizar una aplicación móvil es muy ambicioso. Esta cuestión es también compleja, porque es difícil limitar el alcance de un trabajo al no tener experiencia previa.

3) **Poca experiencia con las tecnologías de este trabajo**: esto hace referencia a que los objetivos y funcionalidades probablemente sí se podían haber realizado en el tiempo establecido (unas 300 horas de trabajo) siempre que se hubiera trabajado en estas tecnologías previamente con suficiente fluidez, para adquirir los conocimientos necesarios que permitieran prescindir de los cursos de formación.

Por otra parte, me gustaría comentar que este TFT ha sobrepasado las 400 horas de trabajo en su totalidad, y la aplicación está compuesta 10.000 líneas de código aproximadamente. No obstante, aunque a priori pueda ser mucho tiempo y trabajo, pienso que ha merecido la pena porque sin entrar en temas de eficiencia u optimización de código, he logrado hacer una aplicación por mí mismo sin ningún tipo de ayuda, y me siento muy orgulloso en ese sentido.

6.2 Trabajos futuros

Para finalizar, me gustaría comentar posibles mejoras que se podrían realizar en la aplicación:

- Adaptar la aplicación web a dispositivos móviles: en un principio debido a que se iba a realizar una aplicación móvil, no se pensó en la idea de que la web fuera adaptable a dispositivos móviles. No obstante, si la web fuera adaptable, un usuario podría hacer uso de ella en una pantalla de menor tamaño.
- Implementar la historia de usuario número 07 ("*Google recaptcha*" [43]) para detectar el tráfico procedente de programas automatizados.
- Optimización y eficiencia de código: se podrían disminuir el número de llamadas a la base de datos, y modificar ciertas implementaciones con un código más limpio o adecuado.

No añadiría más funcionalidades, porque la aplicación está bastante completa en este aspecto, y tampoco considero oportuno sobrecargar la interfaz de usuario con más elementos que puedan dificultar el uso que le puedan dar personas con cierta edad.

Bibliografía

[1] "*RoutineXT": nombre ficticio del logotipo creado en la plataforma "Renderforest" (página 9).*
https://www.renderforest.com/es/logo-maker

[2] *Competencias específicas IS01, IS02 e IS04 (página 10).*
https://www.eii.ulpgc.es/tb_university_ex/?q=objetivos-y-competencias-del-gii#Segundo_Enlace

[3] *Metodología "Scrum" (página 10).*
https://www.wearemarketing.com/es/blog/metodologia-scrum-que-es-y-como-funciona.html

[4] "*Framework" Angular (página 10).*
https://angular.io/

[5] *Lenguaje de programación "HTML" (página 10).*
https://developer.mozilla.org/es/docs/Web/HTML

[6] *Lenguaje de programación "CSS" (página 10).*
https://developer.mozilla.org/es/docs/Web/CSS

[7] *Lenguaje de programación "Typescript" (página 10).*
https://www.typescriptlang.org/

[8] *Reglas de "Shneiderman y Plaisant" (página 11).*
https://www.kicorangel.com/diseno-de-interfaces/
https://webdesign.tutsplus.com/es/articles/8-golden-rules-for-better-interface-design--cms-30886

[9] *Biblioteca "Bootstrap" (página 11).*
https://getbootstrap.com/

[10] *Aplicación con imágenes: "EJERCICIOS EN CASA" (página 13 y 14).*
https://apps.apple.com/es/app/ejercicios-en-casa-sin-equipo/id1313192037

[11] *Aplicación con imágenes: "FREELETICS" (página 14 y 15).*
https://apps.apple.com/es/app/freeletics-hiit-fitness-coach/id654810212#?platform=iphone

[12] *Aplicación con imágenes: "ENTRENAMIENTOS DIARIOS FITNESS" (página 16 y 17).*
https://apps.apple.com/es/app/entrenamientos-diarios-fitness/id469068059

[13] *Aplicación con imágenes: "PIERNAS FUERTES 30 DÍAS" (página 17 y 18).*
https://play.google.com/store/apps/details?id=legworkout.formen.legsworkoutstraining&hl=es&gl=US

[14] *Aplicación con imágenes: "RETO DE 30 DÍAS - Pectorales" (página 18 y 19).*
https://apps.apple.com/cr/app/reto-de-30-d%C3%ADas-pectorales/id1474062602

[15] *Historias de usuario (página 21).*
https://www.atlassian.com/es/agile/project-management/user-stories

[16] *Imagen de la metodología "Scrum" (página 21).*
https://www.diegocalvo.es/wp-content/uploads/2018/04/Metodolog%C3%ADa-SCRUM.png

[17] *Técnica "MoSCoW" (página 23).*
https://adrianalonso.es/project-management/priorizacion-requisitos-software-con-moscow/#:~:text=La%20t%C3%A9cnica%20de%20priorizaci%C3%B3n%20de,like%20but%20won't%20get

[18] *Tablón "Trello" (página 23).*
https://trello.com/b/GPmV9mw8/bienvenido-a-trello

[19] *Curso de formación de diseño web (página 23).*
https://www.udemy.com/course/diseno-web/

[20] *Curso de formación de "Angular", "Firebase" y "Typescript" (página 23).*
https://www.udemy.com/course/la-web-empieza-aqui-typescript-angular-storage-firebase/

[21] *Definición de "Sprint" o iteración (página 25).*
https://www.ycoinbound.com/blog/qu%C3%A9-es-un-sprint-de-scrum

[22] *Lenguaje de Programación "Javascript" (página 25).*
https://developer.mozilla.org/es/docs/Web/JavaScript

[23] *Editor "Visual Studio Code" (página 25).*
https://code.visualstudio.com/

[24] *Base de datos "Firebase Firestore Database" (página 25).*
https://firebase.google.com/docs/firestore

[25] *Entorno "Node.js" (página 25).*
https://nodejs.org/es/

[26] *Herramienta de Prototipado "Adobe XD" (página 25).*
https://www.adobe.com/es/products/xd.html

[27] *Repositorio "Github" (página 25).*
https://github.com/Nestructor/RoutineXT_Web

[28] *Herramientas de ofimática: "Google sheets", "Google slides", "Google docs" (página 25).*
https://www.google.es/intl/es/docs/about/

[29] *Plataforma "Udemy" (página 25).*
https://www.udemy.com/

[30] *Biblioteca "Ionicons" (página 25).*
https://ionic.io/ionicons

[31] *Biblioteca "Fontawesome" (página 25).*
https://fontawesome.com/v4.7/

[32] *Almacenamiento "Firebase Storage" (página 26).*
https://firebase.google.com/docs/storage

[33] *Biblioteca "*Jquery*" (página 27).*
https://jquery.com/

[34] *Estructura "Dashboard" (página 35).*
https://startbootstrap.com/

[35] *Módulo "Sweetalert2" (página 41).*
https://sweetalert2.github.io/

[36] *Herramienta "Emailjs" (página 41).*
https://www.emailjs.com/

[37] *Diagrama de clases (página 47).*
https://www.lucidchart.com/pages/es/tutorial-de-diagrama-de-clases-uml

Software:
https://staruml.io/

[38] *Diagrama entidad-relación (página 48).*
https://www.lucidchart.com/pages/es/que-es-un-diagrama-entidad-relacion

Software:
https://staruml.io/

[39] *Algoritmo "Ordenamiento por burbuja" (página 56).*
https://www.ecured.cu/Ordenamiento_de_burbuja#:~:text=Algoritmo%20de%20ordenamiento.,est%C3%A1n%20en%20el%20orden%20equivocado.&text=Tambi%C3%A9n%20es%20conocido%20como%20el%20m%C3%A9todo%20del%20intercambio%20directo.

[40] *Sistema Operativo "Android" (página 74).*
https://www.android.com/intl/es_es/

[41] *Lenguaje de programación "Kotlin" (página 74).*
https://kotlinlang.org/

[42] *Curso de formación en "Android" (página 74).*
https://www.udemy.com/course/android-kotlin-developer/

[43] *Servicio "Google recaptcha" (página 88).*
https://www.google.com/recaptcha/about/