t-Reto: Generador de retos deportivos con Strava

Alberto Corral López
Grado Ingeniería Informática
Desarrollo de aplicaciones para dispositivos móviles en HTML5 o Windows Phone
Nombre Consultor/a: Carlos Sánchez Rosa
Nombre Profesor/a responsable de la asignatura: Carles Garrigues Olivella

3 de enero del 2020

t-Reto: Generador de desafíos deportivos con Strava

FICHA DEL TRABAJO FINAL

Título del trabajo:	*t-Reto: Generador de desafíos deportivos con Strava*
Nombre del autor:	*Alberto S. Corral López*
Nombre del consultor/a:	*Carlos Sánchez Rosa*
Nombre del PRA:	*Carles Garrigues Olivella*
Fecha de entrega (mm/aaaa):	01/2020
Titulación:	*Grado Ingeniería Informática*
Área del Trabajo Final:	*Desarrollo de aplicaciones para dispositivos móviles en HTML5 o Windows Phone*
Idioma del trabajo:	*Castellano*
Palabras clave	*Ionic, Angular, Firestore, Gestión, Deporte*

Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de aplicación, metodología, resultados i conclusiones del trabajo.*

Este proyecto nace de una primera versión embrionaria realizada con Microsoft Excel en la que se creó una competición deportiva entre un grupo de amigos. En este caso, las actividades se debían registrar en un Excel que el administrador del grupo debía actualizar en el momento que un participante le enviara la actividad realizada.

Este flujo de trabajo era innecesario, máxime cuando todos los participantes registraban sus actividades en Strava. Strava se ha convertido en una de las mayores comunidades de deportistas aficionados, donde comparten todas sus actividades en el ámbito de cualquier deporte.

Estos deportistas aficionados pueden agruparse en clubs y proponer retos sencillos para evaluar su nivel de competencia, pero en la mayoría de los casos, estos retos son demasiado simples y provocan el abandono del desafío por parte de los participantes.

El objetivo de mi aplicación es poder generar desafíos parametrizables, de tal forma que el administrador del grupo pueda determinar una puntuación para cada parámetro del deporte practicado y en la que cada actividad registrada por el usuario recibirá una puntuación final según tipo de deporte, modalidad (indoor, outdoor), distancia, tiempo y desnivel.

Mediante estos parámetros, el administrador podrá decidir si una actividad en la que se recorre una distancia X tendrá mayor coeficiente en bicicleta de montaña, bicicleta de carretera o corriendo.

Pero la mayor ventaja es que los datos de las actividades los recogerá directamente de la API de Strava. De esta forma, no se dependerá de la dedicación en exclusiva del administrador.

Abstract (in English, 250 words or less):

This project was born from a first embryonic version made with Microsoft Excel, in which a sports competition was created among a group of friends. In this case, the activities should be updated in an Excel that the group administrator should update when the participant sent the activity.

This workflow was unnecessary, especially when all participants registered their activities in Strava. Strava has become one of the largest communities of amateur athletes, where they share all their sport activities.

These amateur athletes can be grouped into clubs and propose simple challenges to assess their competence level, but in most cases, these challenges are too simple and cause the participants drop the challenge.

The purpose of my application is to generate parameterizable challenges, so that the group administrator can determine a score for each parameter of the sport practiced, so each activity registered by the user will receive a final score according to the type of sport, modality (indoor, outdoor), distance, time and slope.

Through these parameters, the administrator can decide if an activity of 20km will have a higher coefficient on mountain biking, road bikes or running.

But the biggest advantage is that the activity data will be collected directly from the Strava API. In this way, you will not depend on the exclusive dedication of the administrator.

Índice

Lista de figuras

Capítulo 1. Introducción

1.1 Contexto y justificación del Trabajo

Entre los aficionados al deporte, cada vez existe un mayor interés por compartir sus actividades con el resto de la comunidad. Ya sea por mantener un registro individual o por satisfacción del ego personal, lo cierto es que son cada vez más los usuarios que registran todo lo que consideren actividad deportiva.

Las aplicaciones que registran actividades luchan por hacerse con estos datos y convertirse en comunidades deportivas de referencia. Estos valiosos datos permiten tener registrados los recorridos, estado físico, datos personales, y otros muchos datos de gran valor para su posterior comercialización. Pero para conseguirlo, antes deben eliminar la barrera tecnológica que se encuentran algunos usuarios poco versados en la tecnología.

Actualmente existen muchos dispositivos capaces de registrar actividades físicas. Marcas como Garmin, Apple, Suunto o Polar han creado multitud de dispositivos de fácil manejo para que los usuarios puedan registrar toda su actividad, ofreciendo consejos de rendimiento y mejora.

Marca	Web
Garmin	https://www.garmin.com/es-ES/
Apple	https://www.apple.com/es/watch/
Polar	https://www.polar.com/es
Suunto	https://www.suunto.com/es-es/

Figura 1 Aplicaciones especializadas en dispositivos deportivos.

Pero ¿qué ocurre cuando practican deporte con más de un dispositivo de diferente marca?. Sencillamente, se encuentran con la fragmentación en varias aplicaciones, por lo que tener una visión global de todo su plan de entrenamiento se antoja imposible.

1.1.1 Las aplicaciones deportivas

Desde el inicio de este fenómeno de deportistas aficionados, son varias las aplicaciones que se han disputado el mercado. Endomondo, Runtastic y Strava son algunas de las más conocidas, pero esta última es la que ha

conseguido mayor comunidad de seguidores, principalmente por su facilidad de integración con todo tipo de dispositivos deportivos.

Aplicación	Web
Endomondo	https://www.endomondo.com/
Runtastic	https://www.runtastic.com/es/
Strava	https://www.strava.com/dashboard

Figura 2 Aplicaciones especializadas en software deportivo

Resulta de vital importancia trabajar con una aplicación deportiva que permita tener todas las actividades centralizadas, pero para ello, las aplicaciones deben ser compatibles con gran multitud de dispositivos, y sobre todo, que sean fáciles de utilizar. A diferencia de otras aplicaciones deportivas, la estrategia de Strava se ha centrado en permitir al usuario registrar actividades desde su propia plataforma, pero también permitirle importar fácilmente todo lo que registrase en sus dispositivos.

Así, el pasar una actividad desde un reloj, un GPS o cualquier otro dispositivo era tan fácil como sincronizar las actividades del dispositivo y permitir que éste se conectase a la cuenta Strava. De esta forma, de manera totalmente transparente, lo que antes se debería hacer en un mínimo de dos pasos, ahora sólo requería uno:

CON STRAVA

Figura 3 Envío actividad en un paso

SIN STRAVA

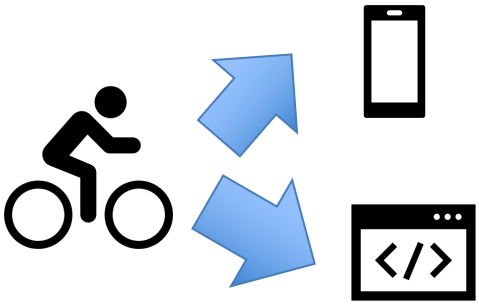


Figura 4. Envío actividad en dos pasos

Actualmente, sólo debemos registrar la actividad en el software del dispositivo, y automáticamente enviará los datos a la cuenta Strava (previa aceptación de permisos). Como hemos dicho antes, no todos los usuarios de estas aplicaciones están dispuestos a dedicar mucho tiempo al su estudio, por lo que es muy importante que el paso de estos datos sea ágil, intuitivo y esté centralizado.

1.1.2 La comunidad y las comparativas

Una vez registrados los datos, al aficionado a la práctica de los deportes le gusta compararlas con el resto de comunidad o con sus amigos de confianza.

Esa inquietud competitiva es la que ha convertido a Strava en un referente en este tipo de comunidades deportivas, donde sus desafíos o control de segmentos, se han convertido en todo un referente entre estas comunidades de usuarios. Conseguir el mejor tiempo en un tramo, o superar las distancias de tus contactos se ha convertido en una auténtica obsesión entre los usuarios de esta comunidad.

1.1.3 Desafíos muy descafeinados

No obstante, los desafíos de Strava son demasiado sencillos y no valoran el esfuerzo realizado en una actividad. Actualmente, estos desafíos se basan en distancia o tiempo recorrido, sin tener en cuenta el tipo de actividad realizada o el terreno recorrido.

Esta laxitud a la hora de valorar las actividades realizadas, ha provocado que el interés en estos desafíos fuera más bien bajo. Cuando compara actividades, la comunidad desea que lo haga en igualdad de condiciones, como mínimo por tipo de deporte.

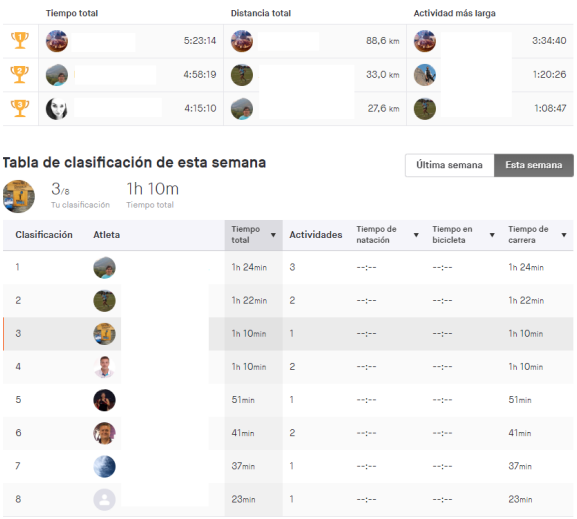


Figura 5 Desafío Strava

1.1.4 Desafíos personalizados

Tener la posibilidad de parametrizar los desafíos, según deporte y dificultad de la actividad, es el gran reto de esta aplicación híbrida.

Mediante un sistema de puntuación por actividades, el usuario podrá comparar sus entrenos con los del resto de la comunidad, pudiendo participar en diferentes competiciones y desafíos creados por un administrador del

grupo. De esta forma, se incentiva la competencia entre usuarios y se incentiva la práctica de deporte.

1.1.5 Strava y sus datos

Según los datos aportados por Strava del año 2018, cuenta con 36 millones de usuarios activos de 195 países. A su aplicación se suben 15 millones de actividades a la semana, lo que da una media de 20 actividades por segundo.

Estos datos aumentan año tras año, y a la espera del informe del 2019, Strava parece haberse convertido es la comunidad deportiva más utilizada por parte de los usuarios.

1.2 Objetivos del Trabajo

El objetivo del proyecto es crear una aplicación híbrida que pueda ser utilizada tanto en móviles como en dispositivos web, y cuyo principal campo sea el de poder crear retos deportivos, centrándose inicialmente en ciclismo de montaña, ciclismo de carretera, *running*. Además, contemplará la modalidades *indoor* de los deportes comentados anteriormente.
Para llevar a cabo proyecto, se ha puesto en práctica los conocimientos adquiridos durante el Grado de Ingeniería Informática, así como los adquiridos a partir del estudio de cursos y videotutoriales externos a la UOC, en especial los centrados en Ionic 4 y Angular 7 (o posterior).

Para conseguir el éxito del producto es necesario:

- Trabajar con el API aportado por Strava y poderla integrar y gestionar desde nuestra aplicación. (https://developers.strava.com/)
- Solventar las dificultades que hemos encontrado con los *token* temporales que ofrece Strava.
- Gestionar el registro de usuarios mediante Firebase de Google (https://firebase.google.com/?hl=es-419)
- Crear un entorno de navegación intuitivo y sencillo.

En lo que se refiere a requerimientos funcionales, la aplicación gestiona los siguientes puntos:

- Registro y *Login* de usuarios mediante Firebase.
- Crear retos personalizados.
    - Configurar usuarios del reto.
    - Configurar los deportes aceptados en el reto.
    - Configurar las normas aplicables al reto.
- Crear y gestionar las normas evaluables de cada deporte.
- Importar las actividades de los usuarios desde Strava o introducción manual.
- Generar clasificaciones por reto según las actividades de los usuarios.
- Calcular y cuantificar la calidad de las actividades según los parámetros del reto.

1.3 Enfoque y método seguido

Originalmente, la gestión de estos retos se realizaba con una hoja Excel, donde se registraban las actividades de cada uno de los usuarios, y se aplicaban los cálculos siguiendo las pautas indicadas en el propio Excel.

Los mayores problemas de la utilización del Excel eran:

- Documento poco accesible en línea.
- Actualización condicionada a un único usuario administrador.

La opción de adaptar esta primera versión a una aplicación móvil, solucionaba ambos problemas, además de añadir nuevas e interesantes funcionalidades, tal como se comentó en el punto 1.2.

Se opta por una aplicación híbrida, principalmente diseñada para móviles, porque el tipo de usuario para el que está pensada es un usuario que utiliza el terminal para poder registrar las actividades publicadas en Strava. A pesar de todo, no se descarta el uso de PWA (*Progressive Web Application*) ya que este tipo de aplicaciones no obligan tener la aplicación registrada en los principales mercados de aplicaciones: Google Play y Apple Store.

He decidido apostar por Ionic 4 porque es un framework UX que adapta el código programado a diferentes dispositivos, amén de disponer de una amplia comunidad de usuarios con los que compartir dudas y recursos.

A su vez, programar el producto con la base tecnológica de Angular 7+, se ha convertido en una motivación extra para aprender definitivamente este framework Javascript, tan extendido en el mundo laboral. Angular trabaja con TypeScript, lenguaje de programación libre desarrollado por Microsoft y que se considera una mejora de Javascript, añadiendo tipos estáticos y objetos estructurados en clases.

Tanto Angular como Ionic disponen de numerosos recursos en la red, vitales para poder afrontar las numerables dudas con las que me encontraré en el desarrollo del producto.

1.4 Planificación del Trabajo

Para poder llevar a cabo el proyecto, ha sido necesario un software y hardware específico.

HARDWARE

- Portátil Macbook pro 13” Intel Core I5 2,7 Ghz con 8 Gb de Ram y sistema operativo macOS Mojave 10.14.6
- Smartphone Android LG K4 2017 con sistema Android 6.0.1
- Iphone 7 con sistema IOS 12

SOFTWARE

- **Visual Studio Code 1.38.1** para la codificación: https://code.visualstudio.com/
- **Extensiones Visual Code:**
  - Angular 2 Typescript
  - Angular 8 Snippets
  - Angular Language Service
  - ColorHighlight
  - GitLens
  - HTML Snippets
  - Javascript ES6
  - TSLint
  - VSCode-icons

- **GitHub** como repositorio del proyecto. https://github.com/giger2000?tab=repositories
- **Ionic 4** https://ionicframework.com/
- **Angular 8** https://angular.io/
- **FireBase** para gestionar la persistencia de datos y Backend https://firebase.google.com/?hl=es-419
- **Microsoft 365** para la generación de la documentación https://www.office.com/?auth=2
- **Adobe XD** para realizar los prototipos y simulación de funcionamiento. https://www.adobe.com/es/products/xd.html
- **Draw.io** para la creación de diagramas UML. https://www.draw.io/

Las tareas en las que se estructura el proyecto están clasificadas en 4 entregas, ilustradas en la siguiente tabla:

Tarea	Descripción	Inicio	Final
PEC 1	• Contexto y justificación del Trabajo • Objetivos del Trabajo • Enfoque y método elegido • Planificación del Trabajo • Breve sumario de productos obtenidos • Breve descripción del resto de capítulos de la memoria • Aplicación "Hola Mundo"	18/9/2019	02/10/2019
PEC 2	• Usuarios y contexto de uso [Análisis] • Diseño conceptual [Diseño] • Prototipado [Diseño] • Evaluación [Evaluación] • Definición de los casos de uso • Diseño de la arquitectura	3/10/2019	30/10/2019
PEC 3	• Implementación	31/10/2019	11/12/2019
PEC 4	• Entrega	12/12/2019	3/1/2020
Defensa Virtual	• Defensa Virtual	13/1/2020	17/1/2020

Figura 6 Tabla de tareas del proyecto

En el siguiente diagrama de Gantt podemos ver la distribución de las tareas en el calendario del proyecto:

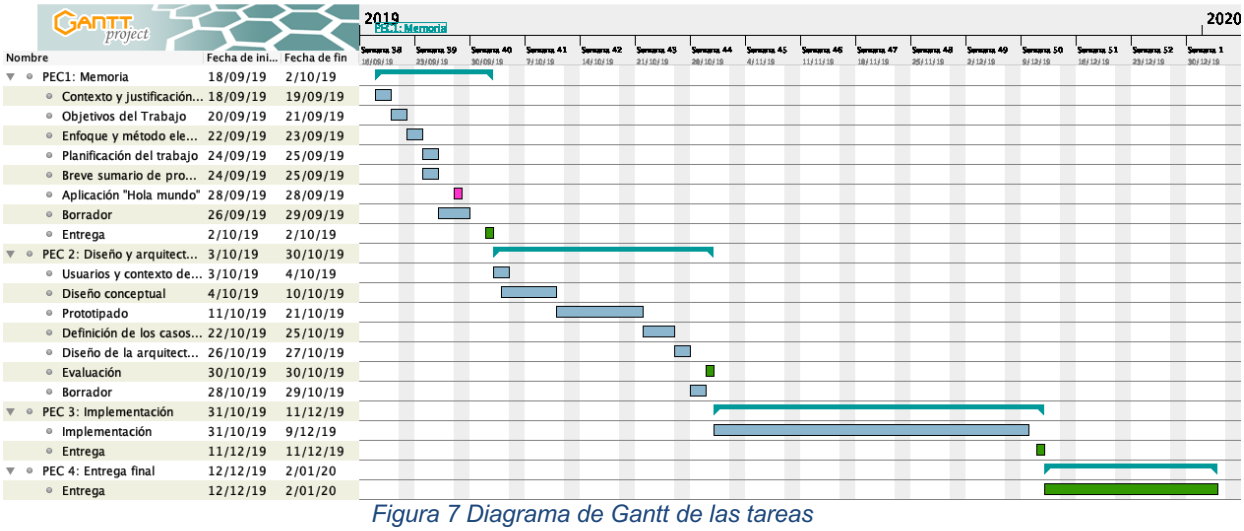


Figura 7 Diagrama de Gantt de las tareas

Teniendo en cuenta que para los días laborables se van a dedicar entre 2 y 3 horas, mientras que los fines de semana se dedicarán entre 4 y 6 horas. En un principio, los días laborales se pueden dedicar más horas en la parte de implementación, siempre dependiendo del calendario laboral.

Teniendo en cuenta esta estimación, de las 300 horas de las que se disponen, tengo previsto dedicar 250 horas al proyecto, dejando 50 horas en caso de necesitar ajustes temporales.

1.5 Breve sumario de productos obtenidos

Una vez finalizado el proyecto, se realiza la entrega de la siguiente documentación:

- Aplicación T-Reto en formato *.apk y versión web.
- Memoria del trabajo de fin de grado
- Presentación Power-Point.
- Vídeo demostración del funcionamiento de la aplicación.

1.6 Breve descripción de los otros capítulos de la memoria

En los siguientes capítulos de la memoria nos centraremos en todos los pasos seguidos para conseguir el producto final.

En primera instancia hablaremos del *target* de usuario al que vas destinada la aplicación, indicando las ventajas que puede encontrar utilizándola.

Posteriormente realizaremos el estudio de diseño, ajustando la gama cromática a un máximo de 5 colores, y diseñando la tipología de pantallas que mostrará la aplicación. Para este caso, tendremos en cuenta las consideradas zonas cálidas de interacción entre usuario y móvil.

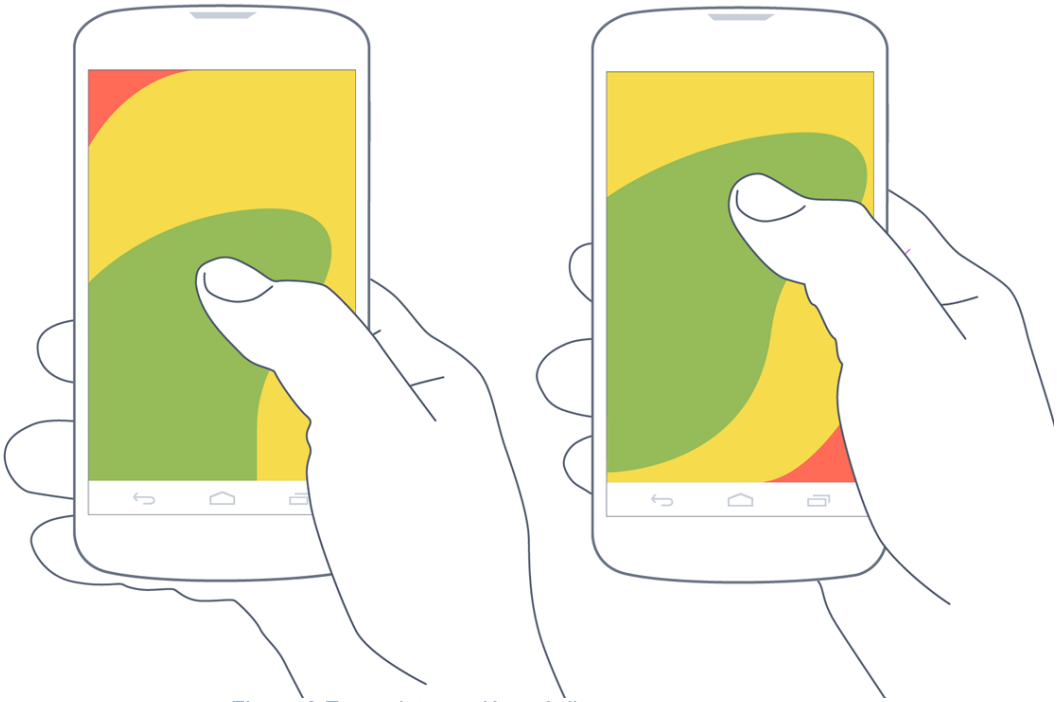


Figura 8 Zonas interacción móviles

El prototipado nos servirá para simular el funcionamiento de la navegación e interacción con el usuario. Lo haremos siguiendo la norma *mobile First*, donde se da prioridad a estos dispositivos para la disposición gráfica de los botones y funciones de interacción.

En el apartado de implementación se darán detalles de la tecnología utilizada, así como el uso de los frameworks y API'S de datos utilizadas, y se añadirán las partes de código más relevantes. Se especificará los mayores problemas encontrados, así como los recursos utilizados para solventarlos. Finalmente se detallarán las pruebas realizadas y los resultados obtenidos.

Por último, en la memoria del trabajo de fin de Grado, dedicaremos un capítulo a las conclusiones, así como un glosario de términos, bibliografía y anexo.

Capítulo 2. Diseño centrado en el Usuario (DCU)

En este apartado vamos a analizar a los posibles usuarios de la aplicación con el fin de detectar las funcionalidades mínimas que debe tener la aplicación, así como descartar otras no necesarias en esta primera versión del proyecto.

Trabajando en un diseño centrado en el usuario (DCU) podremos centrarnos en la utilidad y usabilidad de la aplicación, para conseguir un producto entendible y útil para todo tipo de usuarios, desde nivel inicial hasta avanzado. El objetivo de este proceso es estudiar el perfil de usuario final, lo que necesita y su contexto de uso.

El DCU es un proceso iterativo que consta de 3 etapas interconectadas: análisis, diseño y evaluación, y en los siguientes apartados describiremos los pasos seguidos en cada una de estas etapas, así como las metodologías escogidas para nuestra aplicación. Finalmente, expondremos las conclusiones.

2.1 Usuarios y contexto de uso

2.1.1 Encuesta

A partir de una encuesta de 31 preguntas, realizadas mediante un formulario de Google Docs a una muestra de 56 encuestados, he podido llegar a diferentes conclusiones con referencia al tipo de perfil que usaría la aplicación, así como de sus necesidades y objetivos.

Los datos de la encuesta se pueden consultar en el anexo correspondiente

El objetivo principal de la encuesta era determinar qué edades, deportes y dispositivos móviles tienen los usuarios potenciales, además de conocer sus usos habituales.

Como valores globales podemos ver que el 89,3% de los usuarios tienen edades comprendidas entre 31 y 55 años, siendo mayoría los de edades comprendidas entre 31 y 45 años (67,9%) , de los que el sexo masculino predomina en un 83,9%.

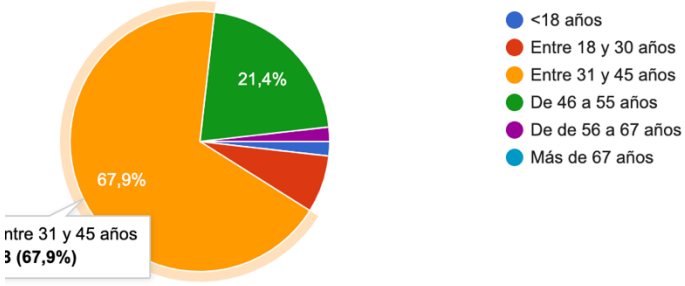


Figura 9 Edades de los encuestados

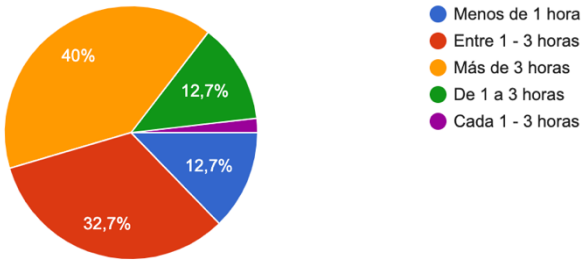


Figura 10 Uso de móvil

Respecto a conocer hasta qué punto están acostumbrados a utilizar el móvil, el 67,9% Android, prácticamente todos los usuarios utilizan el móvil entre 1 y más de 3 horas al día y principalmente para mensajería y redes sociales, por lo que son usuarios muy activos.

En lo referente a la práctica del deporte, el perfil mayoritario es ciclista, siendo mayoría los que practican la bicicleta de montaña (BTT). El segundo grupo mayoritario sería el de *runners*. De todos los encuestados, el 58,9% dicen practicar deporte entre 2-3 veces por semanas, por lo que serían muy activos subiendo actividades a la plataforma *Strava*.

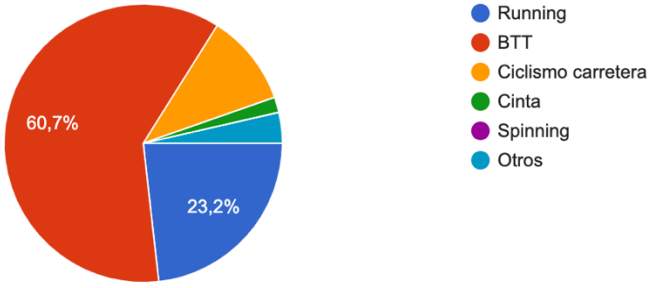


Figura 11 Deportes practicados

Por último, tenía interés en saber hasta qué punto era un perfil de usuario que estuviera acostumbrado a registrar sus actividades en la plataforma *Strava*, y visto los datos de la encuesta podemos ver que el 89,3% suelen registrar sus actividades en algún dispositivo, y de estos, el 78,6% son usuarios de *Strava*, un dato muy positivo.

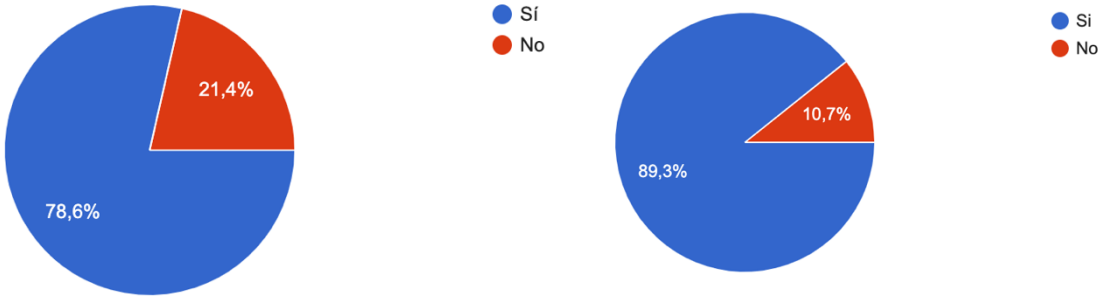


Figura 12 Uso de Strava

Al final de la encuesta se añadieron preguntas referentes a una modalidad de pago de la aplicación, y en este caso el dato es claro, la mayoría no pagaría por usar los retos.

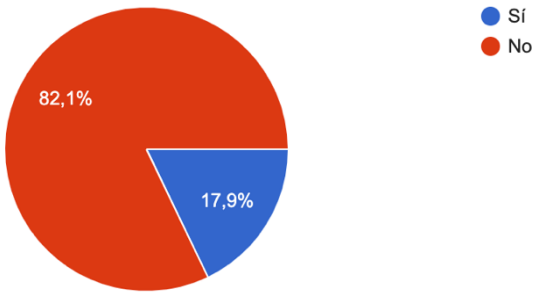


Figura 13 Pago por uso

2.1.2 Perfil de usuario

A pesar de que, originalmente, la aplicación se pensó para los usuarios aficionados que practican el ciclismo de montaña, una primera prueba basada en Excel hizo que ampliara el grupo de usuarios a todos aquellos que practican, de forma amateur, deportes en vinculados al ciclismo y *running*, ya que son los más populares entre los aficionados al deporte y además, son los que mejor se pueden cuantificar siguiendo el registro de actividades de la aplicación *Strava*.

A juzgar por los resultados de la encuesta, podríamos crear diferentes perfiles de usuario.

Perfil	Administrador
Características	Usuario responsable del correcto funcionamiento de la aplicación, con altos conocimientos técnicos y de desarrollo de la plataforma
Contexto de uso	Debe utilizar un ordenador con acceso a internet, para gestionar las bases de datos, así como disponer de las fuentes originales del programa.
Funcionalidades	Gestión base de datos de usuario. Programación y mantenimiento de la aplicación

Perfil	Líder
Características	Usuario con altos conocimientos en el uso de aplicaciones móviles y con suficientes capacidades técnicas como para poder configurarla correctamente. Es el que acostumbra a dinamizar el grupo y crea los retos para motivar al resto.
Contexto de uso	Debe ser el gestor del grupo de usuarios del desafío, por lo que deberá estar pendiente del correcto funcionamiento y configuración de la aplicación. La podrá usar desde el móvil o desde cualquier ordenador, siempre y cuando tenga conexión a Internet. Utilizará la aplicación como usuario normal, pero también para crear nuevos retos. Podrá aceptar las solicitudes de inscripción de los usuarios y deberá estar pendiente del correcto funcionamiento del reto creado.
Funcionalidades	Creación de reto, configurar usuarios del reto, configurar deportes aceptados, configurar normas del reto, aceptar solicitudes al reto.

Perfil	Anti-tecnología
Características	Usuario con conocimientos técnicos muy básicos o nulos. Sólo utiliza el móvil para mensajería y llamadas, y rara vez instala aplicaciones. Este usuario tendrá dificultades para darse de alta y vincular su perfil Strava a la aplicación.
Contexto de uso	Como usuario de la aplicación, deberá tener dispositivo móvil o sobremesa, y conexión a internet. Utilizan la aplicación sólo para consultar los retos a los que está registrado, así como las clasificaciones y actividades registradas tanto suyas como del resto de usuarios. Su ficha de usuario debe pautar perfectamente cómo vincular su identificativo Strava con la aplicación t-Reto.
Funcionalidades	Consultar actividades realizadas, consultar clasificación reto, solicitar inscripción a retos, modificar perfil.

Perfil	Hiperactivo
Características	Usuario con conocimientos técnicos medios. Está muy acostumbrado a disponer de varios dispositivos en los que registra todas sus actividades. Es un usuario muy activo y todo lo registra en Strava para posteriormente compararlo con sus contactos. Es muy competitivo y le gusta ser referencia.
Contexto de uso	Como usuario de la aplicación, deberá tener dispositivo móvil o sobremesa, y conexión a internet. Utilizan la aplicación para comparar sus actividades con el resto. Consulta constantemente la clasificación y la evolución de sus competidores, por lo que le resulta de vital importancia las notificaciones que reciba de la aplicación.
Funcionalidades	Consultar actividades realizadas, consultar clasificación reto, solicitar inscripción a retos, modificar perfil.

Perfil	Solitario
Características	Usuario con conocimientos técnicos medios. Es un perfil muy parecido al anterior pero basa su entrenamiento en las actividades outdoor y en solitario. Los fines de semana no le importa salir en grupo pero entre semana prefiere hacer entrenos cortos pero intensos. A este usuario le gusta descubrir nuevas rutas y compartirlas, por lo que agradecería un espacio para recomendar actividades
Contexto de uso	Como usuario de la aplicación, deberá tener dispositivo móvil o sobremesa, y conexión a internet. Utiliza la aplicación para registrar sus actividades y poder mostrar al resto que está activo, aunque muestre que no lo hace por competir.
Funcionalidades	Consultar actividades realizadas, consultar clasificación reto, solicitar inscripción a retos, modificar perfil, evaluar actividades.

Perfil	Dominguero
Características	Usuario con conocimientos técnicos medios. Este perfil acostumbra a salir sólo los fines de semana y su objetivo es reunirse con otros aficionados a su deporte para disfrutarlo. Es poco competitivo, pero le gusta mostrar sus actividades para sentirse parte del grupo.
Contexto de uso	Como usuario de la aplicación, deberá tener dispositivo móvil o sobremesa, y conexión a internet. Utiliza la aplicación para formar parte del grupo y valoraría la posibilidad de disponer de un espacio para comunicarse dejando mensajes al resto de usuarios.
Funcionalidades	Consultar actividades realizadas, consultar clasificación reto, solicitar inscripción a retos, modificar perfil.

2.1.3 Contexto de uso

La aplicación se usará en cualquier lugar donde el teléfono móvil tenga conexión a Internet ya sea por wifi o por tarifa de datos. También se podrá utilizar con ordenadores que tengan instalado un navegador web.

La aplicación está pensada para usarse una vez se hayan realizado las actividades deportivas y se hayan transmitido a la plataforma deportiva *Strava*. Este paso de datos, el usuario lo deberá realizar mediante un dispositivo que esté conectado a su cuenta *Strava*, o podrá realizarlo manualmente desde la propia plataforma *Strava*. En ningún caso se contempla que la aplicación pueda enviar una actividad ya que es una aplicación de consulta y evaluación de actividades.

Capítulo 3. Diseño conceptual

Una vez definidos los perfiles de usuario y los contextos de uso, es el momento de definir los escenarios de uso y los flujos de interacción, cara conceptualizar la estructura general de la aplicación.

3.1 Diseño conceptual

3.1.1 Escenarios de uso

Los escenarios de uso son descripciones narrativas sobre las diferentes situaciones antes las que se pueden encontrar los usuarios a la hora de utilizar la aplicación. A partir de estas descripciones, planteadas desde el punto de vista de un perfil específico, se detallará como deberá utilizar la aplicación para solventar los problemas planteados. A continuación se platearán los diferentes escenarios de usos aplicados al producto a desarrollar.

Escenario de uso 1: El usuario quiere crear un reto

Perfil de usuario: Líder

Contexto: Jordi ha hablado con su grupo de amigos aficionados al deporte, y ha decidido crear un reto para puntuar sus actividades.

Objetivos: Crear y configurar un reto con los diferentes participantes, deportes y normas de puntuación.

Descripción: Jordi se identifica en la aplicación y ésta le reconoce como usuario Gestor, lo que le permite la creación/configuración de los retos. Aparece la pantalla principal con una imagen de su perfil, sus datos de usuario y un botón '+' para crear un nuevo reto. También verá un menú 'hamburguesa' en la parte superior izquierda y 2 opciones en la barra inferior: Desafíos y Actividades.
En el momento que pulse el botón '+ Nuevo reto', accederá a una pantalla donde le pedirá el nombre del desafío, fecha de inicio, fecha final, y la posibilidad de añadir participantes, deportes y normas.

Funcionalidades: Creación de reto, configurar usuarios del reto, configurar deportes aceptados, configurar normas del reto, comprobar usuario Strava, aceptar solicitudes al reto, puntuar actividades, comentar actividades, consultar notificaciones.

Escenario de uso 2: Usuario quiere apuntarse a un reto

Perfil de usuario: Anti-tecnología

Contexto: A José Miguel le han dicho que Jordi ha creado un reto para competir con los amigos del grupo. Quiere utilizar su móvil para acceder a la aplicación y darse de alta, pero antes debe registrarse como usuario de la aplicación.

Objetivos: Participar en un reto creado por el Gestor

Descripción: Una vez registrado en la aplicación, e identificado correctamente, José Miguel verá una pantalla con los datos básicos de su perfil, donde le indicará si es usuario de Strava. En caso negativo, deberá registrarse como usuario de Strava y volver a la plataforma. También podrá ver un enlace a los retos existentes, en el que podrá consultar los retos y podrá inscribirse.

Funcionalidades: Modificar perfil, comprobar usuario Strava, aceptar solicitudes al reto, puntuar actividades, comentar actividades, consultar notificaciones.

Escenario de uso 3: Evaluar actividad **(futura implementación)**

Perfil de usuario: Solitario

Contexto: A Oscar le gusta entrenar haciendo pequeñas salidas y explorando nuevas zonas, por lo que considera que conviene tener una clasificación de las actividades que más le han gustado.

Objetivos: Puntuar y comentar las actividades realizadas.

Descripción: Una vez registrado en la aplicación, e identificado correctamente, Oscar entra en su perfil de actividades y se dispone a puntuar (1 a 5 estrellas) las que más le han gustado. Además, puede hacer un pequeño comentario al respecto.

Funcionalidades: Modificar perfil, comprobar usuario Strava, puntuar actividades, comentar actividades, aceptar solicitudes al reto, consultar notificaciones.

Escenario de uso 4: Consultar notificaciones pendientes **(futura implementación)**

Perfil de usuario: Hiperactivo

Contexto: Germán entrena cada día y combina los entrenos *indoor* con los *outdoor* de fin de semana. Es muy competitivo y le gusta estar al corriente de la evolución al reto ya que él participa para ganar al resto.

Objetivos: Estar al corriente de todas las novedades que surjan en el reto, sobretodo las actividades realizadas por otros participantes.

Descripción: Una vez registrado en la aplicación, e identificado correctamente, Germán verá un icono en su bandeja de entrada donde le indicará las notificaciones pendientes de leer. En esta primera versión, las notificaciones no serán utilizando el sistema *push*, sólo una vez identificados en la aplicación.

Funcionalidades: Modificar perfil, comprobar usuario Strava, puntuar actividades, comentar actividades, aceptar solicitudes al reto, consultar notificaciones.

3.1.2 Flujos de interacción

En este punto nos centraremos en los flujos de interacción de la estructura general alto nivel. Gracias al diagrama, podremos estudiar la navegación que podrá seguir el usuario interactuando con la aplicación.

Mediante estos flujos podremos identificar posibles problemas de navegación y beneficiar la usabilidad de la aplicación para mejorar la experiencia del usuario.

Pasamos a detallar el esquema desde un punto de vista global.

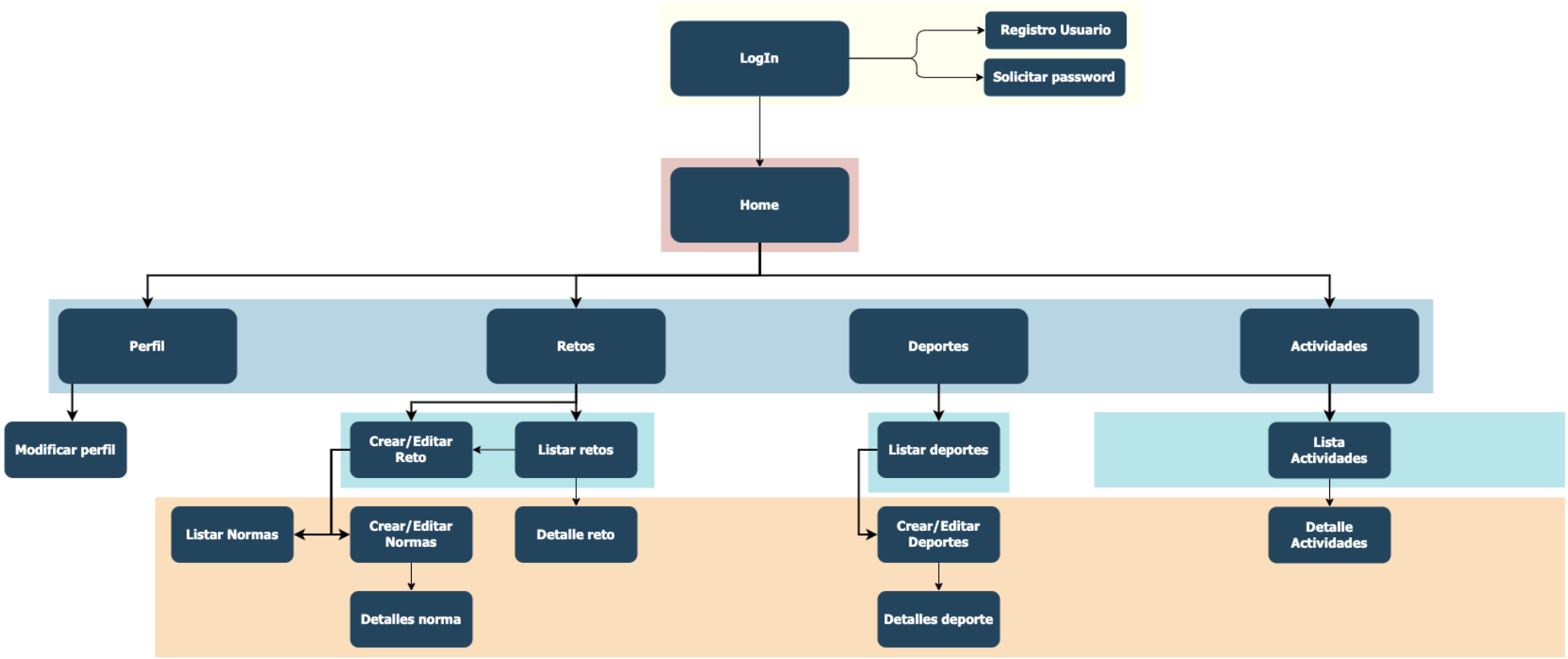
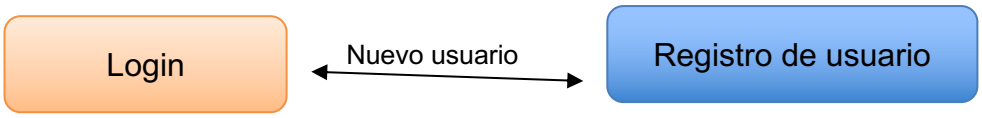


Figura 14 Mapa proyecto

Como se puede observar, existe una pantalla de identificación inicial, tras la que una vez verificada la identidad del usuario, se puede acceder a la pantalla inicial de la aplicación, donde se inicia el flujo de la aplicación pulsando sobre las opciones principales: Perfil, retos, deportes y actividades.

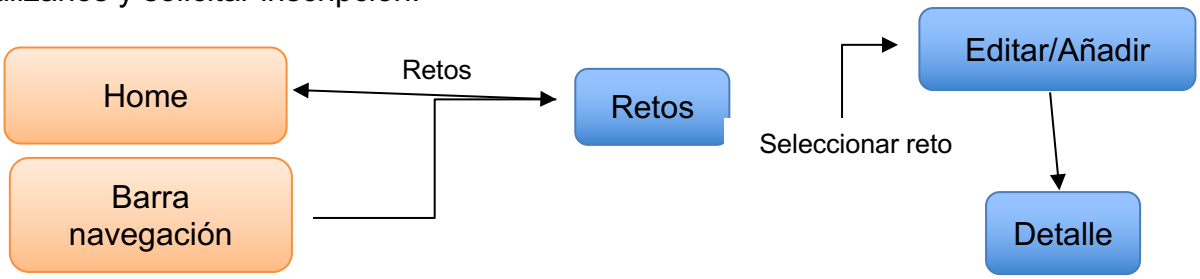
Login: Acceso a la aplicación

Esta primera pantalla tiene como objetivo que el usuario se identifique para acceder a la aplicación, y en caso de no ser usuario, que se pueda registrar en la plataforma.



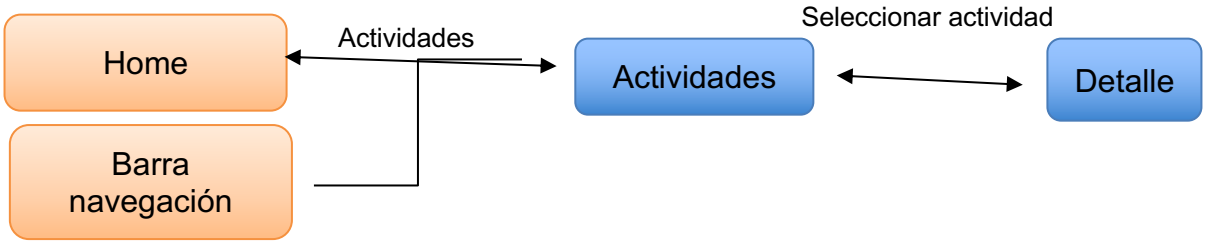
Retos: Gestión y consulta

Desde la pantalla principal y desde la barra de navegación lateral, se podrá acceder a la pantalla de retos. Un detalle importante a considerar, es que sólo el perfil de usuario **líder** podrá crear y gestionar retos. El resto de usuario sólo podrán visualizarlos y solicitar inscripción.



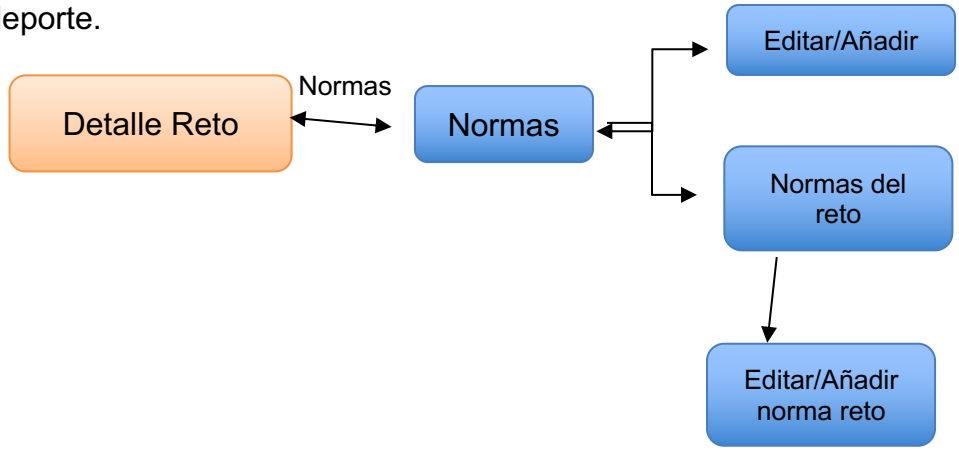
Actividades: Consulta

El listado de actividades se recuperará desde la aplicación Strava, por lo que esta sección sólo será de consulta. Al ser una de las opciones principales, también se accederá desde la barra de navegación.



Normas: Gestión y consulta

Cada reto tendrá la posibilidad de tener sus propias normas de puntuación, por lo que sólo desde un reto seleccionado se podrá acceder a la posibilidad de edición de normas. Existirá una norma mínima para cada reto. Desde el configurador de normas del reto, se podrá acceder al catálogo de normas ya creadas y configurar nuevas. Se debe tener en cuenta que las normas que se editen o creen en esta sección, afectarán únicamente al reto concreto, y que una norma va identificada a un deporte.



Capítulo 4. Prototipado

El prototipo de la aplicación permitirá mostrar la apariencia de la aplicación con una primera aproximación gráfica, pero enfocado en los componentes de usabilidad.

Gracias al prototipo se pueden detectar mejoras que ayuden a un mejor funcionamiento de la aplicación. A continuación se muestran los prototipos de la aplicación T-Reto.

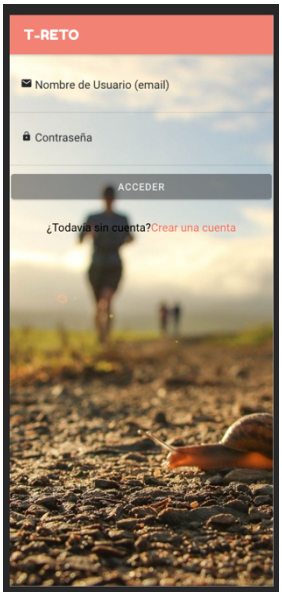


Figura 16 LogIn aplicación

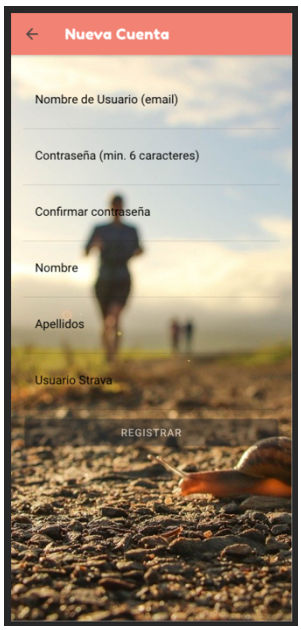


Figura 15 Pantalla Registro usuario

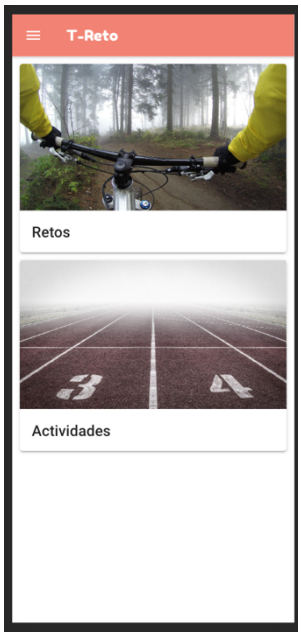


Figura 18 Pantalla Home

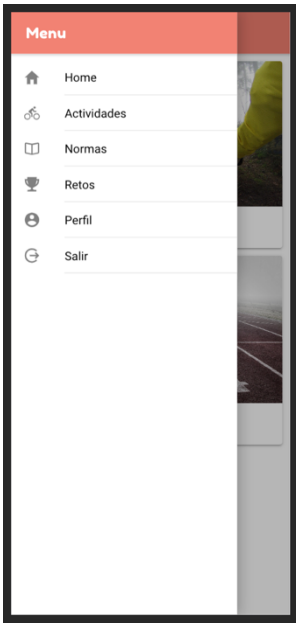


Figura 17 Menú lateral

Figura 19 Pantalla listado retos

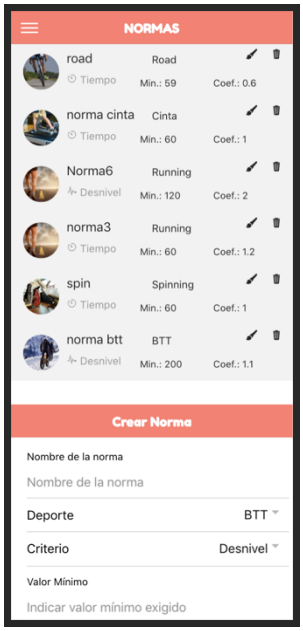


Figura 20 Pantalla crear norma

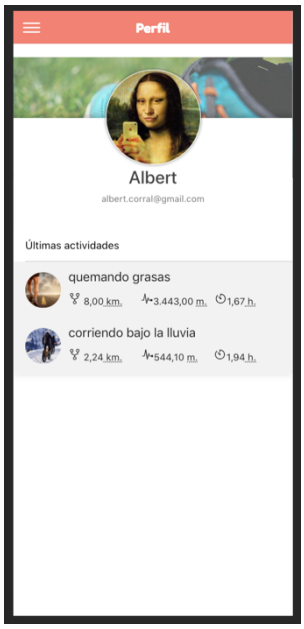


Figura 21 Panatalla actividades usuario

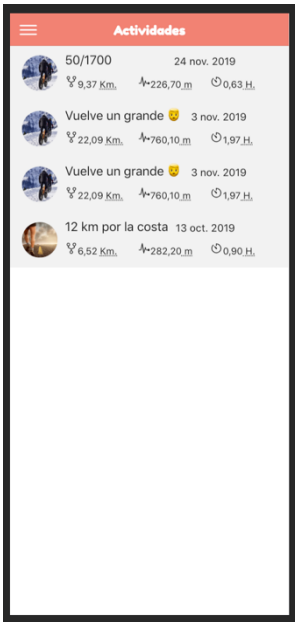


Figura 22 Pantalla actividades Reto

Capítulo 5. Evaluación

En este capítulo vamos a desarrollar el diseño de la arquitectura de la aplicación a partir de los casos de uso.

5.1. Definición de los casos de uso

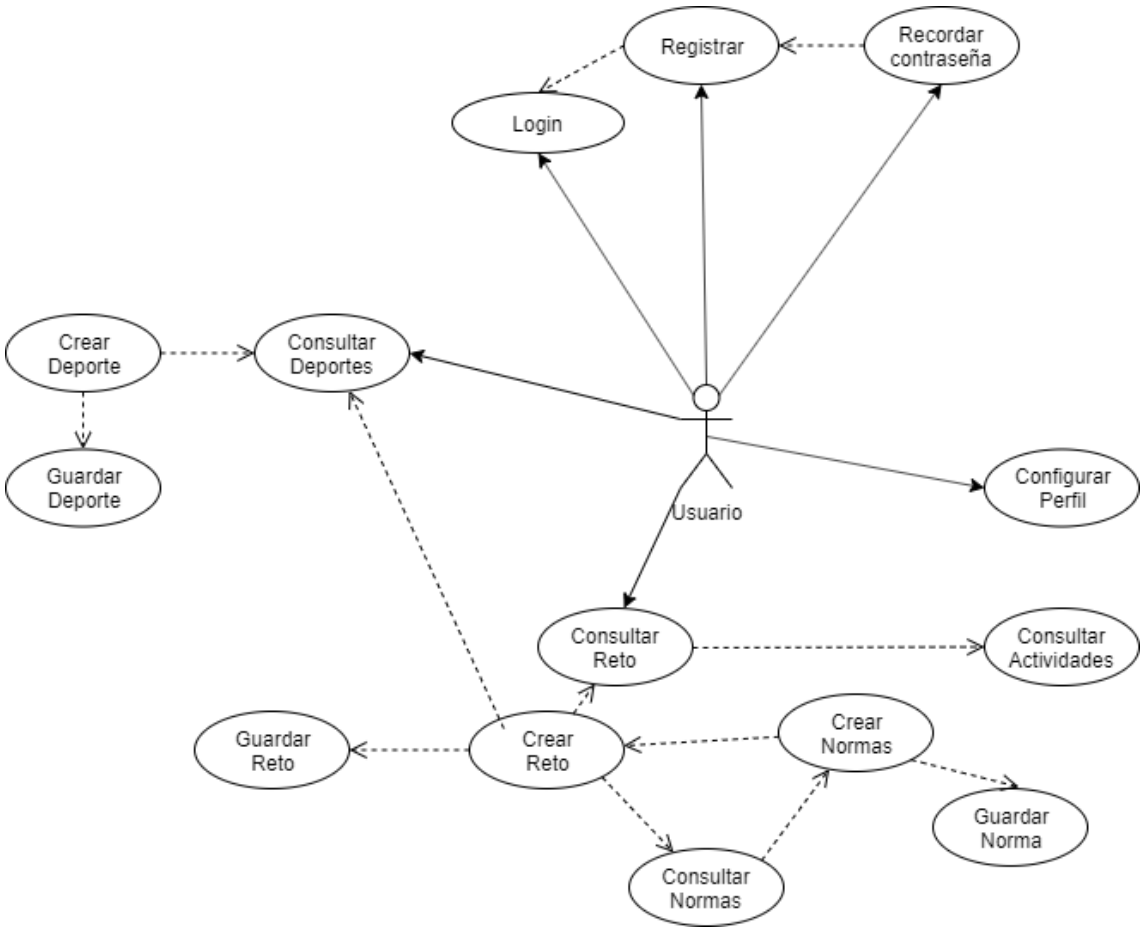


Figura 23 Diagrama casos de uso

Id	UC01
Nombre	Login
Actores	Usuario
Precondiciones	UC02. Estar registrado en el sistema
Flujo	Flujo principal: 1. El usuario introduce el usuario y contraseña. 2. El sistema comprueba los datos mínimos requeridos. 3. El sistema comprueba la validez de los datos en la base de datos. 4. El usuario accede a la aplicación. Flujo alternativo 1: 1. El usuario introduce el usuario y contraseña. 2. El sistema comprueba los datos mínimos requeridos. 3. El sistema comprueba la validez de los datos en la base de datos. 4. El usuario no aparece en la base de datos y muestra mensaje de error. Flujo alternativo 2: 1. El usuario no introduce el usuario y contraseña correctamente. 2. El sistema muestra un mensaje de error.
Postcondición	El usuario accede a la aplicación.
Notas	Ninguna

Id	UC02
Nombre	Registro
Actores	Usuario
Precondiciones	Ninguna
Flujo	Flujo principal: 1. El usuario accede al formulario de registro desde *login*. 2. El usuario rellena todos los campos solicitados. 3. El sistema comprueba que no exista un usuario con el mismo email en la base de datos. 4. El sistema registra al nuevo usuario. Flujo alternativo 1: 1. El usuario ya existe en el sistema. el usuario y contraseña. 2. El sistema muestra un mensaje de error. Flujo alternativo 2: 1. Las contraseñas introducidas no coinciden. 2. El sistema muestra un mensaje de error.
Postcondición	El usuario queda registrado en la base de datos de la aplicación.
Notas	Se podría pedir el usuario STRAVA en el momento del registro, así como comprobar la autenticidad con el sistema STRAVA, pero eso limitaría el acceso sólo a usuarios que se hayan registrado antes en STRAVA. Nos interesa que nuestros usuarios se registren en nuestro sistema y posteriormente en STRAVA para participar en los retos.

Id	UC03
Nombre	Recordar contraseña
Actores	Usuario
Precondiciones	Estar registrado en el sistema
Flujo	Flujo principal: 1. El usuario introduce su correo electrónico. 2. El sistema comprueba la existencia de un usuario con ese correo. 3. El sistema envía un mensaje al correo del usuario para que pueda reestablecer la contraseña. Flujo alternativo 1: 1. El usuario introduce un correo que no aparece registrado. 2. El sistema muestra un mensaje de error. Flujo alternativo 2: 1. El usuario no introduce un correo electrónico. 2. El sistema muestra un mensaje de error y no valida el envío.
Postcondición	El usuario recibe un correo con el acceso a la nueva contraseña.
Notas	

Id	UC04
Nombre	Consultar reto
Actores	Usuario
Precondiciones	Estar registrado en el sistema
Flujo	Flujo principal: 1. El usuario accede al listado de retos actuales. 2. El sistema muestra un listado con los retos registrados. 3. El usuario selecciona el reto a consultar. Flujo alternativo 1: 1. El sistema muestra mensaje de que no hay ningún reto creado. Flujo alternativo 2: 1. El usuario selecciona una fecha inicial superior a la fecha final. 2. El sistema muestra un mensaje de error.
Postcondición	No hay.
Notas	El usuario puede filtrar el listado de retos por fecha.

Id	UC05
Nombre	Crear reto
Actores	Usuario
Precondiciones	Estar registrado en el sistema. El usuario debe tener perfil de líder.
Flujo	Flujo principal: 1. El usuario selecciona la opción NUEVO RETO. 2. El sistema muestra un formulario con los campos obligatorios. 3. El sistema muestra un campo múltiple con las normas registradas en el sistema. 4. El sistema muestra el listado de deportes a los que pertenece el reto. Flujo alternativo 1: 1. El sistema muestra mensaje de que no hay normas creadas en el campo de normas. 2. El sistema muestra un botón de acceso a la pantalla de creación de normas. Flujo alternativo 2: 1. El sistema muestra mensaje de que no hay deportes asociados. 2. El sistema muestra un botón de acceso a la pantalla de consultar deportes.
Postcondición	El usuario crea un nuevo reto.
Notas	Para funcionalidades futuras, el usuario podrá invitar al reto al resto de usuarios. A su vez, todos los usuarios recibirán notificación de creación de nuevo reto. También, un usuario podrá solicitar apuntarse un reto y esperar aceptación.

Id	UC06
Nombre	Modificar reto
Actores	Usuario
Precondiciones	Estar registrado en el sistema. El usuario debe tener perfil de líder. El reto lo ha creado el mismo usuario que lo desea modificar.
Flujo	Flujo principal: 1. El usuario accede a la lista de retos creados por él. 2. El sistema muestra la lista de retos creados. 3. El usuario selecciona el reto a modificar. 4. El sistema muestra el detalle del reto seleccionado. 5. El usuario modifica los datos y guarda los cambios. 6. El sistema informa de la correcta modificación. Flujo alternativo 1: 1. El usuario selecciona una fecha de inicio o final inferior a la fecha actual. 2. El sistema muestra mensaje de error de fecha.
Postcondición	El reto aparecerá con los datos modificados.
Notas	

Id	UC07
Nombre	Eliminar reto
Actores	Usuario
Precondiciones	Estar registrado en el sistema. El usuario debe tener perfil de líder.
Flujo	Flujo principal: 1. El usuario accede a la lista de retos creados por él. 2. El sistema muestra la lista de retos creados. 3. El usuario selecciona el reto a eliminar. 4. El sistema pide confirmación al usuario. 5. El usuario confirma la eliminación del reto. 6. El sistema informa que el reto se ha borrado correctamente. Flujo alternativo 1: 1. El sistema tiene una fecha actual superior a la fecha de inicio e inferior a la fecha final del reto, por lo que se considera iniciado. 2. El sistema muestra un mensaje de alerta de que el reto está activo. 3. El usuario confirma la eliminación del reto. 4. El sistema informa que el reto se ha borrado correctamente
Postcondición	El reto no aparecerá después del borrado.
Notas	En futuras versiones, los usuarios participantes de un reto recibirán aviso del borrado.

Id	UC08
Nombre	Cambiar contraseña (futura implementación)
Actores	Usuario
Precondiciones	Estar registrado en el sistema.
Flujo	Flujo principal: 1. El usuario accede al apartado CAMBIAR CONTRASEÑA. 2. El sistema muestra un formulario. 3. El usuario rellena la nueva contraseña. 4. El sistema valida que los dos campos de contraseña coincidan. 5. El sistema pide confirmación al usuario. 6. El usuario confirma la nueva contraseña. 7. El sistema informa que la contraseña se ha cambiado. Flujo alternativo 1: 1. El usuario no rellena todos los campos. 2. El sistema no activa el botón de aceptación. Flujo alternativo 2: 1. El usuario introduce valores diferentes entre las dos contraseñas. 2. El sistema muestra mensaje de error. Flujo alternativo 3: 1. El usuario introduce una contraseña nueva igual que la actual. 2. El sistema muestra un mensaje de error.
Postcondición	El usuario tiene una nueva contraseña.
Notas	

Id	UC09
Nombre	Crear norma
Actores	Usuario
Precondiciones	Estar registrado en el sistema. El usuario debe tener perfil de líder.
Flujo	Flujo principal: 1. El usuario selecciona la opción NUEVA NORMA. 2. El sistema muestra un formulario con los campos obligatorios. 3. El sistema muestra un campo con los deportes disponibles en el sistema. 4. El usuario selecciona un deporte. 5. El sistema muestra tipología del deporte seleccionado: indoor/outdoor 6. El usuario selecciona tipo de deporte. 7. El sistema muestra un cálculo simulado con los parámetros utilizados por el usuario. Flujo alternativo 1: 1. El usuario selecciona deporte indoor. 2. El sistema muestra el campo tiempo y coeficiente de puntuación. 3. El sistema muestra el filtro mínimo para validar la actividad. Flujo alternativo 2: 1. El usuario selecciona deporte outdoor. 2. El sistema muestra un los campos distancia, desnivel y coeficiente de puntuación . 3. El sistema muestra el filtro mínimo para validar la actividad.
Postcondición	El usuario crea una norma para un reto.
Notas	Sólo el usuario creador del reto puede configurar las normas.

Id	UC10
Nombre	Modificar norma
Actores	Usuario
Precondiciones	Estar registrado en el sistema. El usuario debe tener perfil de líder.
Flujo	Flujo principal: 1. El usuario accede a la lista de normas creadas. 2. El sistema muestra la lista de normas creadas. 3. El usuario selecciona la norma a modificar. 4. El sistema muestra el detalle de la norma seleccionada. 5. El usuario modifica los datos y guarda los cambios. 6. El sistema informa de la correcta modificación. Flujo alternativo 1: 1. El usuario selecciona una norma que pertenece a un reto activo. 2. El sistema muestra mensaje de error e invalida la modificación.
Postcondición	La norma aparecerá con las normas modificadas.
Notas	

Id	UC11
Nombre	Eliminar norma
Actores	Usuario
Precondiciones	Estar registrado en el sistema. El usuario debe tener perfil de líder.
Flujo	Flujo principal: 1. El usuario accede a la lista de normas creadas. 2. El sistema muestra la lista de normas creadas. 3. El usuario selecciona la norma a eliminar. 4. El sistema pide confirmación al usuario. 5. El usuario confirma la eliminación de la norma. 6. El sistema informa que la norma se ha borrado correctamente. Flujo alternativo 1: 1. El usuario selecciona una norma que pertenece a un reto activo. 2. El sistema muestra mensaje de error e invalida la eliminación.
Postcondición	La norma no aparecerá después del borrado.
Notas	

Id	UC12
Nombre	Crear deporte
Actores	Usuario
Precondiciones	Estar registrado en el sistema. El usuario debe tener perfil de líder.
Flujo	Flujo principal: 1. El usuario selecciona la opción NUEVO DEPORTE. 2. El sistema muestra un formulario con el campo de nombre de deporte y selecctor con 3 crierios: desnivel, tiempo y distancia, para deporte indoor/outdoor.. 3. El usuario introduce el nombre del deporte. 4. El usuario selecciona los criterios. Flujo alternativo 1: 1. El usuario introduce un nombre de deporte ya existente. 2. El sistema muestra un mensaje de error.
Postcondición	El usuario crea un deporte para participar en los retos..
Notas	

Id	UC13
Nombre	Modificar deporte
Actores	Usuario
Precondiciones	Estar registrado en el sistema. El usuario debe tener perfil de líder.
Flujo	Flujo principal: 1. El usuario accede a la lista de deportes creados. 2. El sistema muestra la lista de deportes creados. 3. El usuario selecciona el deporte a modificar. 4. El sistema muestra el detalle del deporte seleccionado. 5. El usuario modifica los datos y guarda los cambios. 6. El sistema informa de la correcta modificación. Flujo alternativo 1: 1. El usuario selecciona un deporte que pertenece a un reto activo. 2. El sistema muestra mensaje de error e invalida la modificación.
Postcondición	El deporte aparecerá con los deportes modificados.
Notas	

Id	UC14
Nombre	Eliminar deporte
Actores	Usuario
Precondiciones	Estar registrado en el sistema. El usuario debe tener perfil de líder.
Flujo	Flujo principal: 1. El usuario accede a la lista de deportes creados. 2. El sistema muestra la lista de deportes creados. 3. El usuario selecciona el deporte a eliminar. 4. El sistema pide confirmación al usuario. 5. El usuario confirma la eliminación del deporte. 6. El sistema informa que el deporte se ha borrado correctamente. Flujo alternativo 1: 1. El usuario selecciona a un deporte que pertenece a un reto activo. 2. El sistema muestra mensaje de error e invalida la eliminación.
Postcondición	El deporte no aparecerá después del borrado.
Notas	

Id	UC15
Nombre	Consultar Actividades
Actores	Usuario
Precondiciones	Estar registrado en el sistema
Flujo	Flujo principal: 1. El usuario accede al listado de sus actividades realizadas. 2. El sistema conecta con el API STRAVA 3. El sistema muestra un listado con las actividades registradas en Strava. Flujo alternativo 1: 1. El usuario no ha configurado su usuario Strava en el Perfil 2. El sistema muestra un mensaje de error y enlace al perfil del usuario. Flujo alternativo 2: 1. El usuario accede a las actividades desde un reto. 2. El sistema realiza el cálculo de los puntos de cada actividad. 3. El sistema muestra las actividades de todos los participantes al reto.
Postcondición	Se muestran las actividades del usuario. Se muestran las actividades de todos los participantes de un reto.
Notas	En un futuro, el usuario puede filtrar el listado de actividades por deporte, tipo de deporte y puntuación.

Id	UC16
Nombre	Consultar Perfil
Actores	Usuario
Precondiciones	Estar registrado en el sistema.
Flujo	Flujo principal: 1. El usuario accede al perfil. 2. El sistema muestra el perfil del usuario.
Postcondición	No hay
Notas	

Id	UC17
Nombre	Modificar Perfil
Actores	Usuario
Precondiciones	Estar registrado en el sistema.
Flujo	Flujo principal: 1. El usuario accede al perfil. 2. El sistema muestra el perfil del usuario. 3. El usuario modifica el perfil.
Postcondición	No hay
Notas	

Id	UC18
Nombre	Consultar Perfil Strava
Actores	Usuario
Precondiciones	Estar registrado en el sistema.
Flujo	Flujo principal: 1. El usuario accede al perfil. 2. El sistema muestra el perfil del usuario. 3. El usuario pulsa el botón de consulta de perfil STRAVA. 4. El sistema consulta API STRAVA 5. El sistema compara el identificativo de usuario Strava con el identificativo especificado en el perfil de usuario de la aplicación. Flujo alternativo 1: 1. El usuario no ha configurado su usuario Strava en el Perfil 2. El sistema muestra un mensaje de error y no consulta API Strava. Flujo alternativo 2: 1. El usuario Strava del Perfil no coincide con ningún usuario de STRAVA 2. El sistema muestra un mensaje de error e invita a consultar su aplicación Strava.
Postcondición	No hay
Notas	

5.2 Diseño de la arquitectura

La arquitectura seleccionada para desarrollar la aplicación está basada en el patrón Modelo Vista Controlador (MVC) junto con el patrón Observer, ya que la aplicación basa gran parte de su funcionamiento en una API externa gestionada por Strava, y datos consultados en tiempo real.

La idea es crear un interfaz que se encargue de centralizar todo el trabajo de recepción de datos, de modo que si cambia el API del proveedor sólo debamos centrarnos en dicho interfaz. Se puede entender como una "pasarela" a funcionalidades residentes en sistema externos al nuestro.

La problemática de trabajar con un API externo reside en que en cualquier momento te pueden cambiar la fuente y obligarte a cambiar tu código.

- El **modelo** es la capa donde se gestionan los datos.
- La **vista** es la capa que se encarga de mostrar dichos datos.
- El **controlador** es quien se encarga de pasarle a la vista los datos que recibe por el modelo

.

5.2.1 Diseño de la base de datos

Para el diseño de la base de datos se han utilizado los servicios de Google Firebase, que al tratarse de mi primera experiencia con este servicio, me ha llevado a tener muchos problemas.

La principal ventaja es que en su versión gratuita permite trabajar en un entorno Cloud, y el propio sistema se encarga del correcto funcionamiento de la base de datos. No obstante, el mayor problema fue el trabajar con bases de datos NO relacionales, ya que durante el Grado hemos trabajado siempre con bases de datos relacionales.

Capítulo 6. Implementación

6.1 Introducción

Para desarrollar la aplicación se ha optado por un desarrollo híbrido que la hace compatible con sistemas Android, iOS y PWA (Progressive Web Applications), con lo que se cubriría prácticamente el 100% del mercado.

Para conseguir dicho objetivo se ha utilizado el framework de desarrollo Ionic en su versión 5, lo que ha supuesto un gran reto puesto que sus componentes no siempre alcanzan los objetivos de una aplicación móvil convencional, por lo que se ha tenido que combinar con otros Frameworks, como Bootstrap.

6.2 Herramientas utilizadas

Para el desarrollo global de la aplicación se han utilizado las siguientes herramientas.

6.2.1. VS Code

Visual Studio Code, es un IDE de Microsoft, de licencia libre y que se centra n el desarrollo al que se le pueden integrar diferentes librerías para poderlo adaptar a todo tipo de lenguajes. En nuestro caso hemos optado por Angular 7, que es un framework Javascript basado en TypeScript, que mejora notablemente el tipado del código.

En nuestro caso, los complementos añadidos a VS Code son:

- Angular 2 TypeScript Emmet
- Angular 8 Snippets
- Angular Language Service
- BootStrap v4 Snippets
- Color Highlights
- Firebase
- Import Cost
- TSLint

La mayoría de estos complementos son ayudas para la correcta codificación.

6.2.2. Google Cloud Platform: FireBase

Mediante Firebase podemos disponer de una plataforma de desarrollo en la nube con diferentes servicios más propios de BackEnd. Para este proyecto utilizaremos la versión gratuita, que ya nos permite utilizar bases de datos no-SQL y servicio de autenticación.

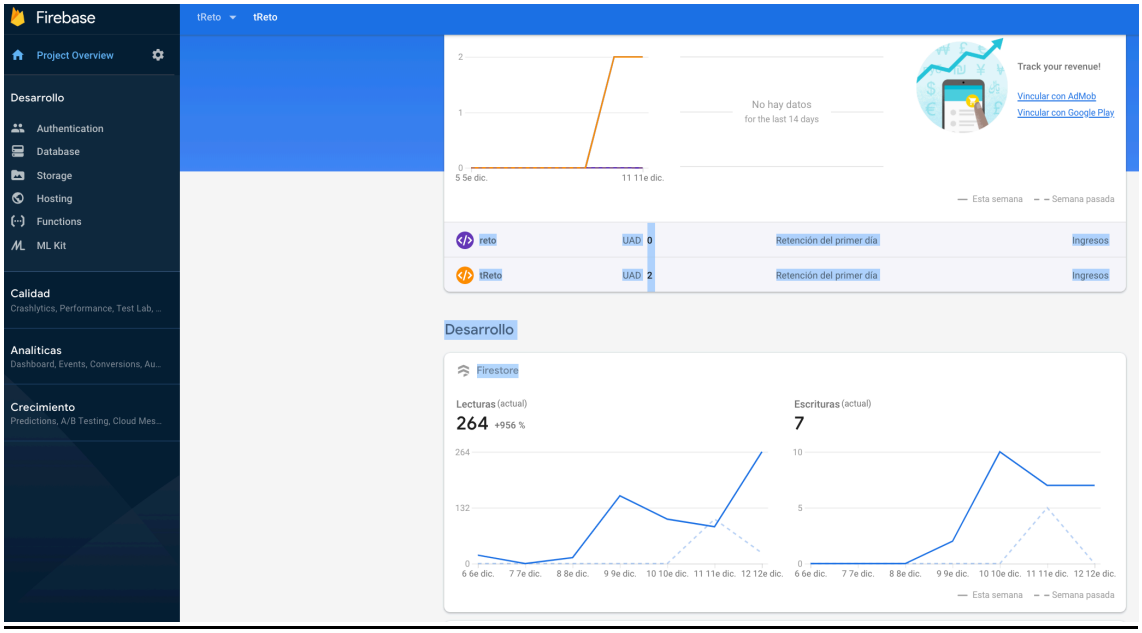


Figura 24 Consola Firebase

Autenticación

Es un servicio que nos simplifica el inicio de sesión y la gestión de la misma en nuestra aplicación. Si la usamos en aplicaciones web es extremadamente fácil de configurar, sobretodo si usamos el proveedor de Google, aun así, si usamos otros de los disponibles (Correo/Contraseña, Teléfono, Facebook, Twitter, GitHub, Anónimo) también es muy fácil, sólo es un paso más en el caso de las redes. En nuestro caso, optamos por el de Correo/Contraseña

Cloud FireStore

A pesar de hacer una primera aproximación a las bases de datos RealTime de Firebase, la aparición de CloudFirestore y, sobretodo, la gran cantidad de comandos para el desarrollo, me han hecho decantarme por esta última.

Cloud Firestore es una base de datos no-SQL flexible y escalable para la programación en servidores, dispositivos móviles y la Web desde Firebase y Google Cloud Platform. Permite mantener los datos sincronizados con la app y no tendré que preocuparme de la seguridad, latencia o mantenimiento.

Pasos a seguir

Lo primero que haremos es identificarnos con nuestro usuario de Gmail y acceder a la consola principal, donde crearemos nuestro proyecto, en este caso tReto.

Figura 25 Crear proyecto Firebase

Una vez hecho el proyecto, Firebase nos dará unos parámetros de acceso que son los que añadiremos a nuestro proyecto Angular (enviroment.ts).

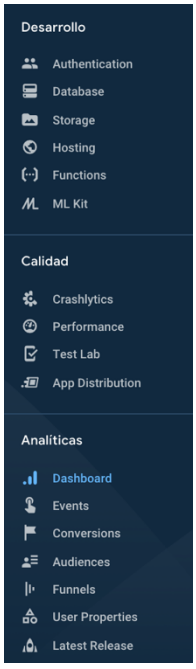


Figura 26 Menú opciones Firebasea

Como se puede ver en el menú de la izquierda, la plataforma Firebase ofrece multitud de servicios para poder realizar nuestra aplicación, así como entorno de testeo, métricas e incluso servicios de anuncios para poder rentabilizar nuestro esfuerzo.

Creación de la Base de datos

A pesar de hacer una primera aproximación mediante Realtime Database, por sus funcionalidades y mayor soporte finalmente migré a Cloud Firestore.

Este cambio durante el proceso de desarrollo me ha obligado a reestructurar el modelo de datos y el desarrollo para poder acceder a los datos gestionados por Cloud Firestore.

Mientras que es Realtime se trabaja con estructuras muy parecidas a los archivos JSON, en Cloud Firestore los datos se estructuran en Colecciones y Documentos.

A continuación muestro la estructura de la tabla de normas en ambos sistemas de gestión de base de datos:

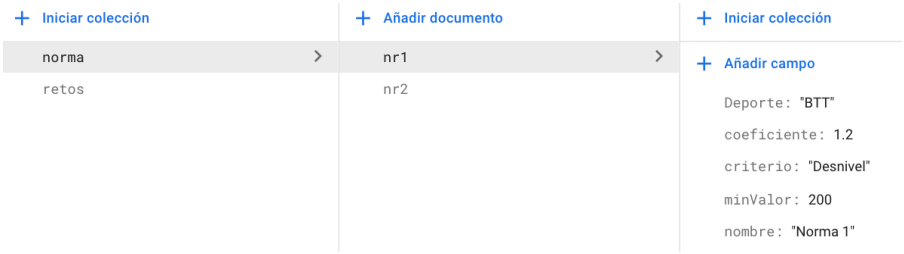


Figura 27 Estructura Cloud

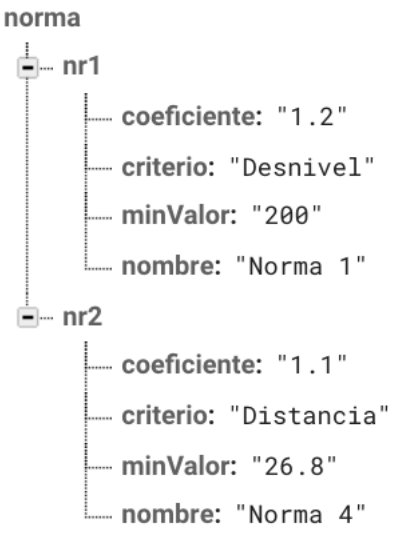


Figura 28 Estructura Realtime

6.2.3. GitHub

GitHub es un servicio web de control de versiones que nos permite tener controlado todos los cambios que hagamos en el código de nuestra aplicación.

Para integrarlo con nuestro proyecto, he creado el siguiente repositorio:

Repositorio GitHub

Desde el repositorio se puede descargar todo el proyecto, y posteriormente desde consola se puede instalar gracias al comando `npm install.`

Para mantener actualizado el repositorio, he optado por utilizar la aplicación de escritorio GitHub Desktop, ya que aporta un entorno visual mucho más amigable y claro que los comandos git por consola.

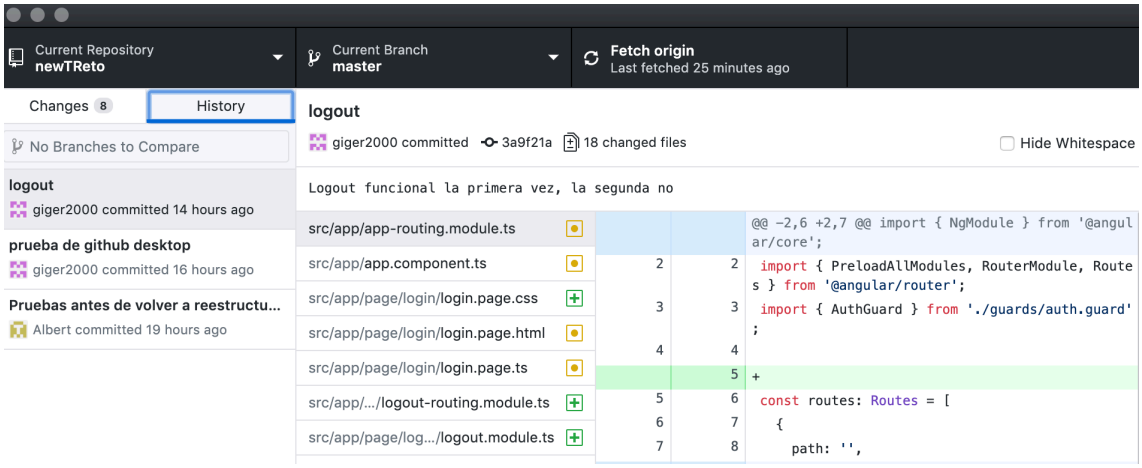


Figura 29 GitHub Desktop

6.2.4 NODE.JS

"Node.js es un entorno en tiempo de ejecución multiplataforma, de código abierto, para la capa del servidor (pero no limitándose a ello) basado en el lenguaje de programación ECMAScript, asíncrono, con I/O de datos en una arquitectura orientada a eventos y basado en el motor V8 de Google." Fuente: Wikipedia

Para intentar resumir lo comentado en el texto anterior, podemos decir que Node.js pues básicamente es un entorno de servidor que permite ejecutar JavaScript del lado del servidor.

Debemos asegurarnos de tenerlo instalado en nuestro ordenador para poder ejecutar la aplicación como si de un servidor se tratara.

Para comprobarlo, debemos poner en el terminal:

```
$ node -v
v13.2.0
```

Si no lo tenemos instalado, debemos instalarlo mediante su página web, que nos permitirá instalar tanto node.js como su gestor de paquetes NPM

Web oficial de Node.js

Figura 30 Zona de descarga de Node.js

6.2.5 NPM

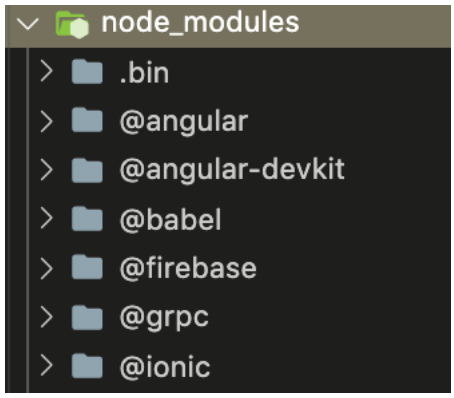


Figura 31 Estructura de módulos instalados

NPM es un gestor de paquetes (módulos) Open Source Javascript, orientado a Node.js. Fue creado en 2009 para compartir módulos entre los desarrolladores, y en mi proyecto lo he utilizado para instalar diferente librearías que posteriormente he ido añadiendo a mi proyecto: ionic, angular CLI, Firebase.

Funciona con un interfaz de línea de comando que nos permite como desarrolladores publicar, instalar, actualizar y eliminar módulos. Estos módulos, en el momento de publicar el proyecto, se añadirán automáticamente.

6.2.6 POSTMAN

Postman nos permite gestionar cómodamente las API's de diferentes proveedores, como por ejemplo la API de Strava que comentaremos posteriormente.

Este programa permite cargar cualquier API y poder ver los datos que nos retorna. Esto es muy útil para saber que nodos debemos consultar para conseguir los datos deseados. Por ejemplo, si trabajamos con una API gratuita que nos devuelve todos los países del mundo de habla hispana, podemos consultar la API y mostrarlos en nuestra aplicación. Esa API la podemos conseguir en la dirección https://restcountries.eu/.

Figura 32 Api de países

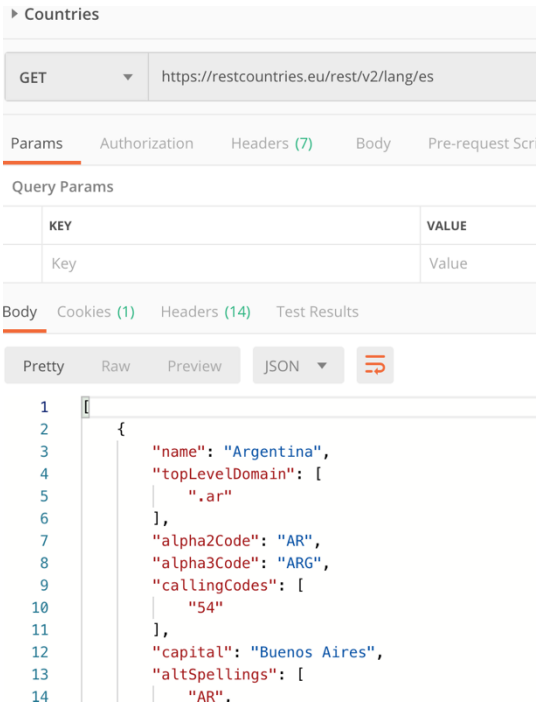


Figura 33 Entorno de Postman

La dirección que nos devuelve, la introducimos en el programa Postman y podremos ver los datos con los que trabajaremos.

Si nos fijamos, vemos toda la ruta de los nodos que podremos consultar, de tal forma que si cargamos los datos en un objeto llamado países, sólo que tendremos que referirnos a él como países.name, para recuperar su nombre.

En el caso de nuestra aplicación, la obtención de los datos es algo más compleja porque debemos consultar datos privados de Strava, y estos datos tienen una limitación temporal.

6.3 Frameworks y API's

Según Wikipedia, un framework, es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de software concretos, que puede servir de base para la organización y desarrollo de software.

En nuestro proyecto vamos a utilizar dos tipos de frameworks: framework CSS y framework Javascript. Ambos frameworks se pueden entender como una caja de herramientas que nos permitirá realizar funcionalidades complejas con sólo utilizar los términos que nos proporciona el framework.

6.3.1. Framework JS Angular (https://angular.io/

Angular es un framework de Javascript creado por Google. Su principal finalidad es facilitar el desarrollo de aplicaciones web SPA (Single Page Application) gracias a las numerosas herramientas que nos proporciona.

Otra de las finalidades por las que fue creado es conseguir una clara separación entre el proceso de producción Front-end (presentación de datos) y back-end (gestión de datos) aunque a medida que evoluciona esta frontera está cada vez más difusa.

6.3.2. Framework Ionic (https://ionicframework.com/)

El principal objetivo de Ionic es conseguir crear aplicaciones híbridas, es decir, que como desarrollador no tengas que preocuparte por los comandos o funcionalidades de entornos como Android, iOs o Web.

Utilizando los componentes que nos ofrece el framework Ionic, simplemente tendremos que usarlos y el framework se encargará de añadir a nuestro proyecto aquello que sea necesario. Las ventajas de trabajar con Ionic son:

- No necesitamos aprender distintos lenguajes de programación.
- No necesitamos conocer diferentes IDE's
- No necesitamos desarrollar bajo diferentes sistemas operativos, aunque sí necesitamos tener instalados los diferentes SDK's para poder empaquetar el proyecto.

Como Ionic está basado en Javascript, su integración a Angular es perfecta, es más, originalmente se planteó para este Framework y poco a poco ha ido incorporando a otros como React.

6.3.3. Bootstrap(https://getbootstrap.com/)

Bootstrap es un Framework CSS, que mediante una larga colección de clases CSS nos permite dar un aspecto profesional a nuestras aplicaciones.

La razón de uso de Bootstrap es mejorar algunos elementos que con Ionic se quedan algo cortos, como por ejemplo la gestión de tablas.

6.3.4. API Strava (https://www.strava.com/api/v3/athlete)

Mediante el API Strava podemos recuperar los datos de las actividades realizadas por nuestros usuarios. El API de Strava trabaja con Token, por o que deberé gestionar la renovación automática de dichos Token sin que sea un proceso demasiado complicado para el usuario.

En un principio, el usuario entra en nuestra aplicación e indica su número de usuario Strava, para que haga una solicitud de conexión a su API. Una vez conectado, la aplicación guarda en LocalStorage el Token para que pueda consultar los datos tantas veces como desee hasta su caduciadad.

Para las pruebas he utilizado Postman, que era el encargado de solicitar el token. Una vez lo tenía, debía ponerlo manualmente en mi código para que la aplicación recuperara los datos en formato JSON.

6.3.5. Angularfire

Para que Ionic se comunique con Firebase, se requiere instalar un paquete mediante npm. La instrucción es:

```
npm install Firebase angularfire --save
```

6.4 Desarrollo

Utilizamos el VSCode para crear la carpeta de trabajo tReto, y apartir de aquí creamos, desde terminal, el proyecto de ionic tReto, para ello clicamos en la consola de VisualCode la instalación del entorno de desarrollo de ionic:

```
npm install -g ionic
```

Posteriormente podemos crear el proyecto con los comando de Ionic:

Una vez hecho eso, tendremos todos los archivos necesarios para que nuestro proyecto funcione bajo el Framework Angular JS.

```
ionic start tReto
```

El CLI de Ionic nos preguntará si queremos partir de una plantilla en blanco, un entorno con TABS o un menú lateral. Inicialmente escogemos menú lateral.

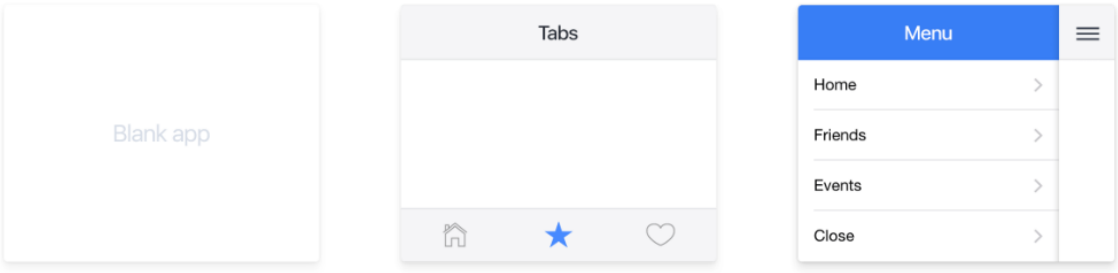


Figura 34 Tipos de proyectos Ionic

Posteriormente, debemos instalar el módulo de angular en nuestro proyecto Ionic:

```
npm install @ionic/angular@latest --save
```

Ionic genera automáticamente toda la estructura de carpetas de nuestro proyecto. Para nuestra aplicación deberemos centrarnos en las carpetas:

- App: Todas las funcionalidades de nuestra aplicación.
- Assets: Todo el material gráfico o extras que añadiremos a nuestra aplicación (imágenes, archivos css,…)
- Enviroment: Gestionamos los entornos de desarrollo o producción.
- Index.html: Primera página de la aplicación. Punto de partida.

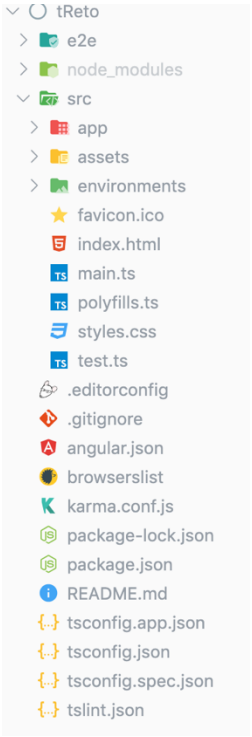


Figura 35 Estrutura proyecto

6.5 Estructura carpeta app

6.5.1 Creación de componentes

Los componentes son módulos reutilizables entre diferentes aplicaciones. Nuestra aplicación estará formada por un componente LOGIN y diversas páginas que serán las opciones de la aplicación.

Para crear los componentes, la instrucción es:

`ionic g c components/login` -> g de generate, y c de component

En nuestro caso, crearemos una carpeta para los componentes, y otra para las página.

El componente Login se encargará de gestionar el alta y registro de los usuarios, bloqueando el acceso a todo aquel no se haya registrado.
Mediante el servicio de Autenticación de Firebase podremos registrar los datos de los usuarios.

Posteriormente, en el archivo app-routing.module.ts, se crean las rutas a los diferentes componentes de la aplicación. También se especifica que si buscamos cualquier ruta que no coincida con un componente, vaya por defecto al home:

```
{ path : '**', pathMatch: 'full', redirectTo: 'home'},
{ path : '', pathMatch: 'full', redirectTo: 'home'}
```

Una vez hecho esto, en el `app.module.ts` es donde se añade el módulo de rutas a nuestra aplicación.

```
import { RouterModule} from '@angular/router';


import { AppComponent } from './app.component';
import { LogInComponent } from './components/home/login.component';
import { ROUTES } from './app.routes';
```

y se importa el componente:

```
  imports: [
    BrowserModule,
    RouterModule.forRoot (ROUTES, {useHash: true})
  ],
```

6.5.2 Creación de páginas

```
ionic g page pages/<nombre>
```

Crearemos las siguientes páginas:

- Home: Página inicial de la aplicación. Básicamente mostrará un enlace a los retos y otro a las actividades, que son las secciones más importantes.
- Logout: Página que nos desconectará de la aplicación y borrará el token de conexión. Una vez desconectados no podremos entrar a las secciones gracias a los GUARD de Angular.

- Normas: Página para gestionar las normas ya creades y para editarlas. Sólo las podrá gestionar el usuario Administrador. Los creadores de retos sólo podrán gestionar las normas de los retos que hayan creado.
- Profile: Página con los datos del usuario.
- Registro: Página para que un nuevo usuario se pueda registrar.
- Actividad: Página donde se muestran las actividades entre un rango de fechas. Debe conectarse a la API de Strava.
- Reto: Página de detalles de un reto seleccionado desde la pantalla de Retos. Desde esta pantalla se pueden modificar los retos ya creados.
- Retos: Listado de todos los retos. Desde aquí se podrá ver el detalle y borrar el reto.

6.5.3 Creación de servicios

Los servicios son los encargados de gestionar los datos de la aplicación para posteriormente usarlos en los diferentes componentes o páginas.

Un servicio se debe encargar de recuperar datos y de disponer de las funciones para manipularlos, por ejemplo, el servicio de retos tendrá funciones para recuperar los retos, para recuperar un solo reto, para modificar un reto,…

Nuestra aplicación dispone de un servicio para cada sección de la aplicación:

- Actividades.service.ts: Recupera y gestiona los datos que vienen de l API de Strava.
- Auth.service.ts: Gestiona los datos de login y registro de usuario.Utiliza el servicio de autenticación de Firestore.
- Normas.service: Se encarga de gestionar las normas de nuestra aplicación.
- Retos.service: Como el anterior de normas, pero en este caso todo los referente a los retos.

6.5.4 Las rutas

Todo proyecto Angular debe tener una correcta gestión de las rutas de navegación entre todos los componentes, para eso, Angular utiliza un módulo especial de *routing*.

Lo primero que se debe hacer es importar el módulo de *routing* al gestor de módulos del proyecto (`app.module.ts`):

```
import { RouterModule, Routes } from '@angular/router';
```

Debemos crear el archivo que gestionará todas las rutas, y le llamaremos `app-routing.module.ts`. En este archivo gestionaremos todas las rutas del proyecto, tanto las navegables como las que deben aparecer en caso de error.

```
import { Routes } from '@angular/router';
import { HomeComponent } from './components/home/home.component';
import { NormasComponent } from './components/normas/normas.component';
import { RetosComponent } from './components/retos/retos.component';
import { UsuariosComponent } from
'./components/usuarios/usuarios.component';
import { DeportesComponent } from
'./components/deportes/deportes.component';
import { SearchComponent } from './components/search/search.component';
export const ROUTES: Routes = [
  { path : 'home', component: HomeComponent},
  { path : 'normas', component: NormasComponent},
  { path : 'retos', component: RetosComponent},
  { path : 'usuarios', component: UsuariosComponent},
  { path : 'deportes', component: DeportesComponent},
  { path : 'search', component: SearchComponent},
  { path : '**', pathMatch: 'full', redirectTo: 'home'},
```

```
  { path : '', pathMatch: 'full', redirectTo: 'home'}
];
```

Una vez definido, lo podemos importar todo a nuestro `app.module.ts`:

```
import { ROUTES } from './app.routes';


@NgModule({
  declarations: [
   …
  ],
  imports: [
    BrowserModule,
    RouterModule.forRoot (ROUTES, {useHash: true})
  ],
```

Y ahora ya podemos editar el `app.component.html` para que el router funcione perfectamente.

```
<app-navbar></app-navbar>

<div class="container m-3">
  <router-outlet></router-outlet>
</div>
```

6.5.5 Creación de Guards para proteger rutas

Los guard son mecanismos de Angular que impiden que un usuario pueda acceder a una sección de la aplicación sin antes identificarse. Esto está más pensado para las aplicaciones web que para las de móvil, puesto que los móviles no disponen de barra de navegación.

Mediante terminal, se crea el Guard como un servicio especial:

```
ionic g guard guards/auth
```

Gracias al Guard, bloqueamos todas las secciones de la página para que no pueda acceder nadie que no esté registrado.

6.6 Tokens y API's externas

6.6.1 HTTP para gestionar datos de API

La gestión de Tokens es básica para el correcto funcionamiento de la aplicación, principalmente para recuperar las actividades de los usuarios en la plataforma Strava.

Con el módulo HTTPClient, angular permite capturar datos de API's externas, con lo que después podremos añadirlas a nuestra aplicación y gestionarlas como deseemos.

6.6.2 Servicios y Tokens

En Angular, los servicios son los módulos encargados de gestionar los datos para que sean accesibles por todos los componentes. En nuestra aplicación, los servicios deben recibir los datos mediante un token temporal que administra Strava. Este token viene identificado por un client_ID y un client_secret, mediante los cuales Strava genera un token codificado para que nuestra aplicación pueda recibir los datos deseados.

Cuando deseemos utilizar el servicio en un componente, debemos importarlo y añadirlo en el constructor como una instancia, pero antes debemos proveer al servicio de datos.

6.6.3 Datos del usuario Strava

En Strava, los usuarios reciben el nombre de *Athletes* y para recuperar todos sus datos nos podemos valer de la dirección de la API https://www.strava.com/api/v3/athlete.

Mediante postman, generamos una consulta aportando el *Authorization code* que tengo en mi perfil de desarrollador de STRAVA. Posteriormente, este proceso lo deberá automatizar la aplicación.

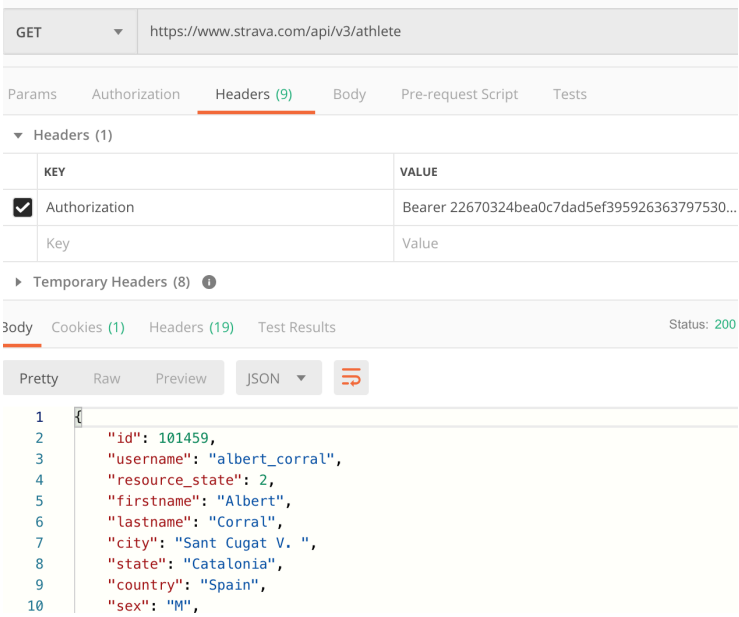


Figura 36 Datos capturados de Api Strava

Pero para conseguir el código token, Strava nos pide una consulta POST a https://www.strava.com/oauth/token en la que tendremos que indicarle:

- Client_id: Aportado por el panel de control de Strava
- Client_secret: Aportado por el panel de control de Strava
- Code: Se debe obtener siguiento las instrucciones de la página oficial de STRAVA (instrucciones)
- grant_type: authorization_code

Una vez comprobado que se reciben los datos, nos disponemos a crear el servicio `Strava.service.ts` que hemos generado con `ionic c s services/strava`:

```
export class StravaService {

  constructor(private http: HttpClient) {

    console.log('Servicio STRAVA listo');
  }

  getAthlete(){

    const headers = new HttpHeaders ({
      'Authorization': 'Bearer 22670324bea0c7dad5ef3959263637975302e032'
    })
      this.http.get('https://www.strava.com/api/v3/athlete',{headers})
          .subscribe ( data => {
            console.log (data);
          })
```

6.7 Firebase y Angular

6.7.1 Realtime Database vs. Cloud Firestore

Inicialmente me decanté por trabajar con esta tecnología proporcionada por Firebase. A pesar de ser realmente potente, la gestión del modelado de datos se hace complicado y poco receptivo a futuros cambios, es por eso me vi obligado a cambiar a Cloud Firestore, con soporte actualizado y con un modelado de datos más natural.

6.7.2 Creación de la estructura de datos

Siguiendo las indicaciones de la página:

https://github.com/angular/angularfire/blob/master/docs/install-and-setup.md

Instalamos el módulo de firebase en angular y posteriormente seguiremos los pasos para añadir los datos de nuestra base de datos Firebase en la estructura del proyecto angular.

Una vez creado el proyecto en firebase (tReto) podemos trabajar con diferentes tecnologías preparadas. En un primer caso escogemos la base de datos, y la tecnología seleccionada es Cloud Firestore.

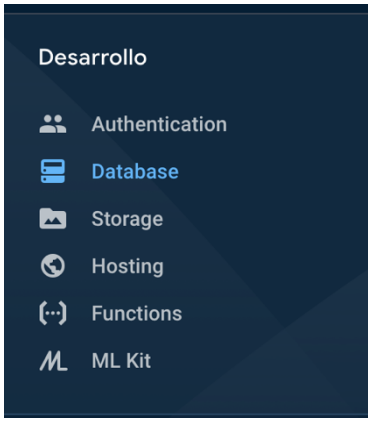


Figura 37 Opciones Firebase

Al trabajar con no-SQL base de datos, estamos cambiando el concepto de las base de datos relacionales. En un principio, la idea es guardar documentos con una estructura de datos determinada donde los datos se estructuran en colecciones y documentos pertenecientes a colecciones.

En la siguiente captura podemos ver un ejemplo de la estructura de los retos:

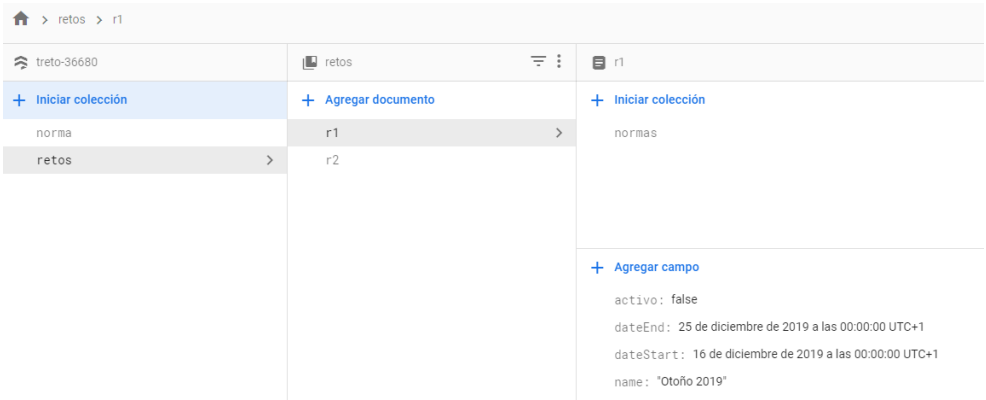


Figura 38 Estructura con Cloud Firestore

Esta misma estructura se puede crear con Realtime Database, pero resulta bastante más complicado navegar por los datos.

La estructura de Realtime es mucho más parecida a la estructura JSON, pero conseguir la relación de datos como una base de datos relacional conlleva hacer muchos más nodos de los necesarios y se convierte en una estructura difícil de manejar.

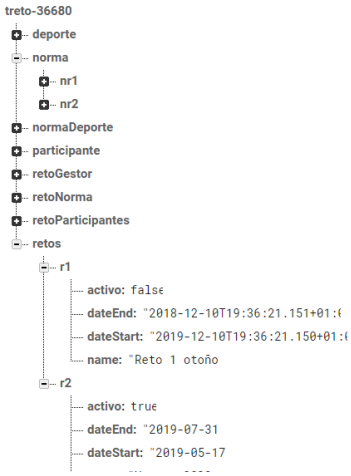


Figura 39 Estructura Realtime Database

6.7.3 Modelo de datos

El proyecto inicial estaba planteado para una base de datos relacional, y con una estructura de clases como la que vemos a continuación:

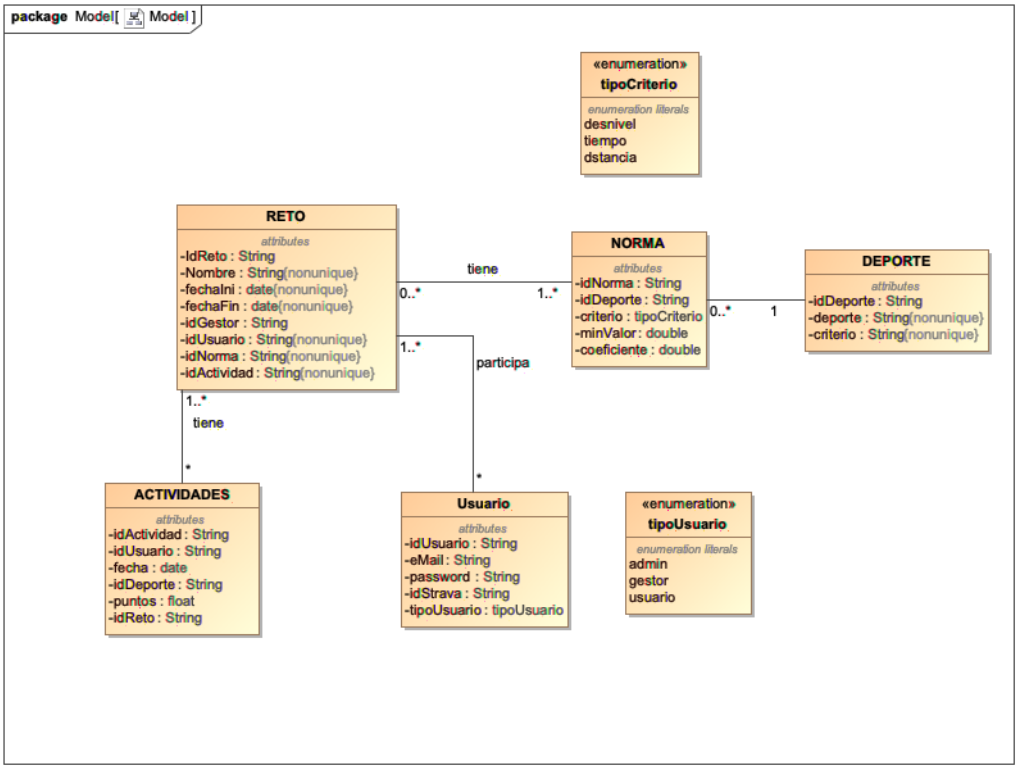


Figura 40 Modelo de datos

Tener deportes en una clase externa, no tiene mucho sentido cuando la aplicación se centra sólo en 5 deportes: Btt, Road, Running, Cinta, Spinning, por lo que dejo para posteriores mejoras la posibilidad de separarlo en una clase independiente.

Capítulo 7. Testing

7.1 Entorno de pruebas

Para probar la aplicación, Ionic da la opción de trabajar con la app Ionic DevApp, por o que sólo nos pide que la descarguemos en el terminal y que nos registremos como usuarios Ionic.

Mediante la instalación de la aplicación Ionic Devapp en el terminal móvil, se puede ver en tiempo real cómo queda la aplicación que estamos desarrollando. Para eso, antes se debe hacer un registro en:

Ionic Devapp

Una vez registrado e instalada la aplicación en el móvil, ambos se comunican mediante la red internar.

Otra opción es ejecutar la aplicación en el ordenador mediante el comando:

```
Ionic serve -lab
```

Si no tenemos instalado el Ionic lab, nos dará la opción de hacerlo.

Gracias al ionic lab podemos ver nuestra aplicación simultáneamente en un terminal Android y otra iOS, ambos simulados.

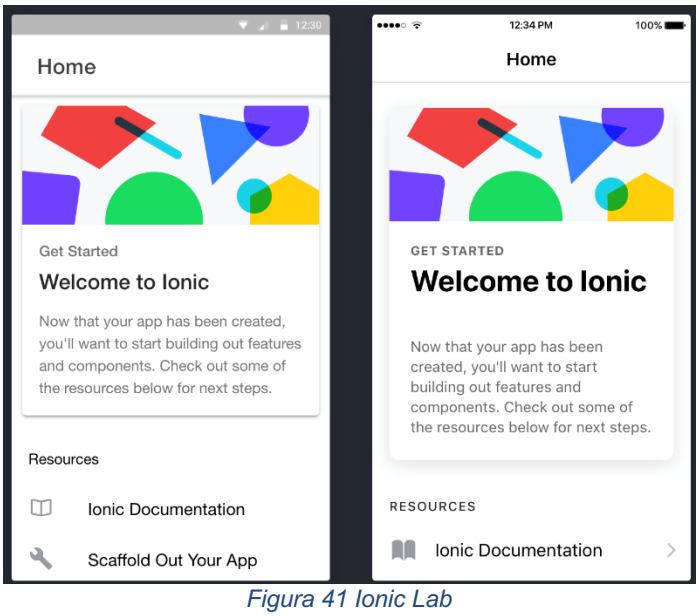


Figura 41 Ionic Lab

Este entorno de pruebas es muy útil para poder comparar el aspecto visual de nuestros componentes en los diferentes sistemas.

7.2 Pruebas reales

Cuando la aplicación esté finalizada, se instalará en los terminales de diferentes potenciales usuarios de la aplicación, los mismos usuarios a los que se le pasó la encuesta para que valoren el resultado.

7.3 Pruebas login

Se ha probado que el acceso a la aplicación funcionara correctamente, teniendo en cuenta que si el usuario no está registrado, no pueda acceder a la página principal (Home).

Capítulo 8. Conclusiones

8.1 Primeros pasos

Al tratarse del primer proyecto que he desarrollado con Angular, y también con Firebase, conseguir combinar ambos me ha resultado muy lento y complicado. Han sido numerosas las veces que he tenido que volver sobre el camino ya realizado para reestructurar el planteamiento inicial, especialmente en el apartado de gestión de datos con Firebase.

Durante el trabajo de final de grado he disfrutado del aprendizaje constante de 3 entornos que nunca había tocado con anterioridad: Angular, Ionic y Firebase.

A pesar de encarar el proyecto con mucha ilusión, el desconocer dichos entornos me ha consumido mucho tiempo de consulta de videotutoriales, manuales y diversos recursos que me impendían evolucionar a un ritmo constante en el proyecto. Además, este desconocimiento me ha provocado que tuviera que reestructurar en varias ocasiones el enfoque del desarrollo.

En un principio planteé la aplicación con una estructura gráfica basada en el framework de Materialize, con el que Angular trabaja perfectamente, además de los componentes aportados por Ionic. La idea era que donde los componentes de Ionic no fueran suficientemente atractivos, los de Materialize tomaran el mando. La realidad es que Materialize es un framework demasiado inmaduro en comparación con otros como Bootstrap, por lo que finalmente elegí este último para trabajar algunos componentes.

8.2 Angular y su curva de aprendizaje

Otro de los problemas que tuve fue la dificultad de aprendizaje experimentado con Angular. Existen muchos enfoques diferentes para hacer una misma cosa, y dar con el adecuado me consumió mucho tiempo, por ejemplo, el proceso de *Login* y Registro.

En varios manuales y videotutoriales tratan el tema, pero cada uno lo hace de forma direfentes: con token, con Firebase Database, Firebase Storage, LocalStorage,…

Decirdirme por uno sólo fue viable a partir de la experiencia de tratar varios de ellos. Finalmente me decidí por un sistema que registrara los usuarios en

Firebase Storage y que comprobara si los usuarios ya estaban registrados en la base de datos.

Como el caso del acceso y registro de usuarios, me he encontrado con otros muchos en los que continuamente debía consultar 3 formas diferentes de trabajar y decidirme por la más idónea para el tiempo del que disponía.

8.3 Firebase Realtime o Cloud FireStore

Convertir esta estructura de datos en la estructura de una base de datos relacional es sencillo, pero me encontré con el problema de que Firebase NO trabaja con bases de datos relacionales y tuve que elegir entre los dos tipos de base de datos NO-SQL que utiliza, el Realtime Databa o el Cloud Firestore.

En un principio me decanté por el Realtime Database, porque convertir una estructura relacional en una estructura de árbol JSON pareció relativamente sencillo, como se ve en este ejemplo:

```
"retos":{
  "r1": {
    "name":"Otoño 2019",
    "dateStart":"2019-05-17",
    "dateEnd":"2019-12-31",
    "activo":false
  },
  "r2": {
    "name":"Verano 2019",
    "dateStart":"2019-05-17",
    "dateEnd":"2019-07-31",
    "activo":true
  }
},
"retoParticipantes":{···
},
"retoGestor":{···
},
"retoNorma":{
  "r1": {
    "nr1":"Norma 1",
    "nr2":"Norma 2"
  },
  "r2": {
    "nr1":"Norma 1"
  }

}
```

Figura 43 Ejemplo reto-norma JSON

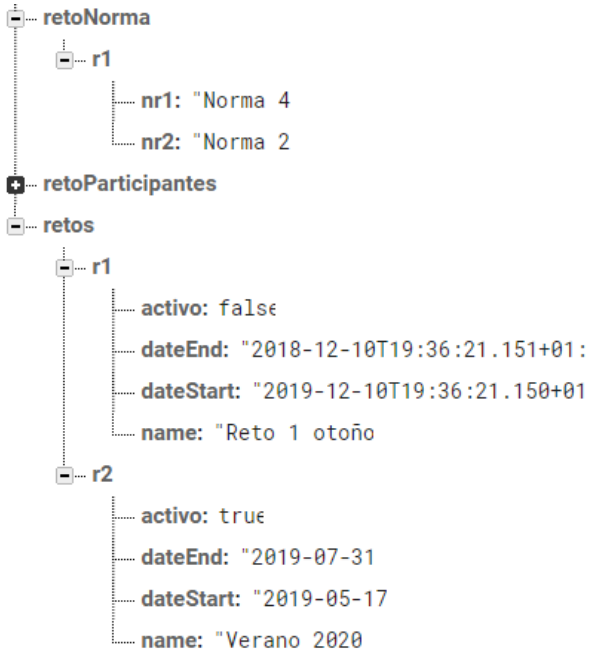


Figura 42 Estructura Reto-norma Realtime

El mayor problema con las bases de datos Realtime se me presentó cuando quise integrarlo en mi proyecto ya que existen librerías propias de Firebase,

pero bastante limitadas y hacer un recorrido por los datos para extraer los datos anidados era demasiado complejo.

En el siguiente ejemplo vemos los pasos a seguir para conectar los con los datos:

ESTRUCTURA BBDD REALTIME

```
orderByKey -> Primary Key
orderByChild -> Where
orderByValue ->
```

Estructura de conexión

```
cont db = firebase.database().ref();
const cursosRef = db.child ('cursos');
cursosRef.funcionOrden().funcionConsulta();
```

Funciones de consulta:

```
RANGOS:
startAt('valor')
endAt('valor')

WHERE
equalTo('child_key')

CANTIDAD
limitToFirst(10)
limitToLast(10)
```

Convertir consulta SQL a Realtime query

```
SELECT *
FROM Retos
Where Nombre == "Otoño 2019"
LIMIT 1;

cont db = firebase.database().ref();
const retosRef = db.child ('retos);
const query = retos
      .orderByChild('name)
      .equelTo('Otoño 2019')
      .limitToFirst(1);
```

El caso contrario me encontré con Cloud FireSotre, muchos más recursos y soporte, y consultas a los datos siguiendo una sintaxis algo más natural. Al poder estructurar los datos en Colecciones y Documentos de colecciones, el acceso a los datos y su estructuración fue bastante más sencilla que la anterior, pero como en casos anteriores mi falta de experiencia me hizo perder un tiempo valioso probando el Realtime Database.

8.4 API STRAVA con Token temporal

Ante la falta de tiempo y los problemas comentados anteriormente, me he visto obligado a trabajar la sección de actividades con un json local con la misma estructura que recibiré de Strava API.

Así, para la sección de Actividades los datos que se verán son los del json ubicados en la carpeta asstes/data/activities.json.

En la versión final se trabajar el acceso al API y la gestión del token temporal.Los pasos que requiere el API de Strava son:

- Identificación de usuario Strava
- Solicitud de token temporal

8.5 Reflexión crítica

A pesar de ser un proyecto que me interesa mucho y de haber dedicado muchas más horas que las que le correspondería a una asignatura de 12 créditos la sensación de ahogo temporal ha sido un lastre. El no poder dedicar todo el tiempo necesarios a adquirir los conocimientos necesarios ha sido frustrante, y el resultado final no es el deseado por mi parte.

A pesar de todo voy a seguir mejorando el producto para conseguir el objetivo deseado y seguir aprendiendo dentro de este cmapo.

8.6 Mejoras futuras

Login/Registro

Me gustaría utilizar un sistema basado en Tokens Firebase para que controle la durabilidad de los acceso y que pasado un tiempo, el acceso sea invalidado.
También se deberiá mejorar el sistema de *guards* para que no pudiera entrar por URL directamente en las secciones, aunque es una mejora sólo para las versiones web puesto que en móvil no es posible.

Funciones Firebase

Mediante las funciones Firebase se puede hacer un sistema de tokens en servidor para poder trabajar con los datos del API de Strava. Es una opción muy interesante para poder hacer que el usuario se identifique sólo una vez en la aplicación y conseguir sus datos Strava para poder consultar sus actividades tantas veces como se desee.

API Strava conexión directa

En futuras versiones me gustaría añadir un sistema por el cual el usuario pudiera conectarse directamente a la aplicación STRAVA sin tener que saber su propio identificativo. Des esta forma poder consultar las actividades que vaya registrando.

Gestión de Retos

Por la problemática temporal comentada anteriormente, me ha sido imposible crear el gestor de Retos que debería contemplar el CRUD. Cada reto debe poder controlas los usuarios participantes, las normas y las actividades. Además, debería valorar las actividades según esas normas y crear una clasificación.

Lo que hay ahora es un visualizador de retos que coge los datos de un documento estático.

Gestión de perfiles de usuarios

Debido a la falta de tiempo, todas las opciones están visibles para todos los usuarios registrados en la aplicación, pero en un futuro quiero crear una política de usuarios, donde sólo los usuarios administradores pueden crear y gestionar tanto los retos como las normas.

Capítulo 9. Glosario

Materialize (https://materializecss.com/)

Framework CSS creado por Google y basado en Material Design de Google.

Bootstrap (https://getbootstrap.com/)

Framework CSS de licencia libre y con varios años en proceso de desarrollo, lo que le da un alto nivel de sus componentes.

Framework

Un framework, entorno de trabajo o marco de trabajo. SE pueden entender como una caja de herramientas con todo lo necesario para utilizarlo en nuestras aplicaciones. Sirven para solventar problemáticas comunes ya resueltas.

API

(Application Programming Interface) Conjunto de subrutinas, funciones y procedimientos que ofrece una biblioteca.

IDE

(Integrated Development Environment) Entorno de Desarrollo Integrado. Aplicación informática que proporciona servicios integrales para facilitar al desarrollador el desarrollo de software.

CRUD
(Create, Read, Update and Delete) Crear, Leer, Actualizar y Borrar. Se refiere a las funciones básicas de manipulación en bases de datos.

Capítulo 10. Bibliografía

EscuelaIT: Curso Ionic 4 (20h)

- **Autor**: Nicolás Molina
- **Web**: https://escuela.it/cursos/curso-ionic-4

Angular desde 0 a avanzado (240h):

- **Autor**: Fernando Herrera
- **Web**: https://www.udemy.com/course/angular-2-fernando-herrera

Vídeo de sistema Login/Registro Ionic Firebase:

- **Autor**: CodeDamn
- **Web**: https://github.com/codedamn/social-media-app-ionic4

Librería referencia Angular:

- **Autor**: Angular Framework
- **Web**:https://angular.io/

Librería referencia Ionic:

- **Autor**: Ionic Framework
- **Web**: https://ionicframework.com/

Librería referencia Firebase:

- **Autor**: Google
- **Web**: https://firebase.google.com/docs?hl=es

Capítulo 11. Anexos

11.1 Instalación

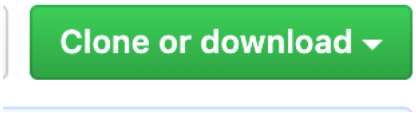
Para la instalación y puesta en funcionamiento se puede optar por utilizar los archivos adjuntados con la entrega o descargar del repositorio GitHub con el proyecto.

11.1.1 Desde repositorio

1. Acceder al repositorio mediante la siguiente dirección:

   https://github.com/giger2000/tfg-tReto

2. Seleccionar la opción Clone or download y copiar la url que nos indica



3. Una vez copiada, podemos acceder a nuestra carpeta de proyectos, y abrir el terminal, donde deberemos poner el siguiente comando:

```
git clone <url copiada>
```

```
$ git clone https://github.com/giger2000/tfg-tReto.git
Clonando en 'tfg-tReto'...
remote: Enumerating objects: 483, done.
remote: Counting objects: 100% (483/483), done.
remote: Compressing objects: 100% (358/358), done.
remote: Total 483 (delta 250), reused 346 (delta 114), pack-reused 0
Recibiendo objetos: 100% (483/483), 3.43 MiB | 369.00 KiB/s, listo.
Resolviendo deltas: 100% (250/250), listo.
```

4. Una vez clonado ya podemos ver la carpeta del proyecto con todos los archivos necesarios:

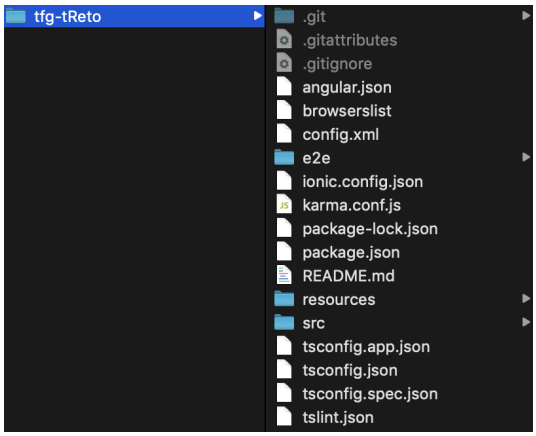


Figura 44 Estructura archivos proyecto

5. Ahora deberemos generar todas las dependencias, por lo que accederemos a la carpeta del proyecto (tfg-tReto) y en el terminal pondremos el siguiente comando:

   ```
   npm install
   ```

11.1.2 Desde archivos adjuntos al proyecto

1. Simplemente se debe ejecutar el comando del paso 5 anterior en la carpeta donde hayamos descargado nuestro proyecto.

11.2 Encuesta

Web: [Encuesta perfil de usuario de Google Docs]

Preguntas de la encuesta

1. Edad
   a. <18 años (1,8%)
   b. Entre 18 y 30 años (7,1%)
   c. Entre 31 y 45 años (67,9%)
   d. De 46 a 55 años (21,4%)
   e. De 56 a 67 años (1,8%)
   f. Más de 67 años (0%)
2. Sexo
   a. Masculino (83,9%)
   b. Femenino (16,1%)
3. Estado civil
   a. Solter@ (10,7%)
   b. Casad@ (64,3%)
   c. Separad@ (1,8%)
   d. Divorciad@ (7,1%)
   e. Viud@ (1,8%)

    f. Vivo en pareja (14,3%)
4. ¿Tiene hijos?
    a. Ninguno (23,2%)
    b. 1 (30,4%)
    c. 2 (44,6%)
    d. 3 (1,8%)
    e. Más de 3 (0%)
5. ¿Dispone de teléfono móvil?
    a. Sí (100%)
    b. No (0%)
6. ¿Cuánto tiempo le dedica a su móvil al día?
    a. Menos de 1 hora (12,7%)
    b. Entre 1-3 horas (47,3%)
    c. Más de 3 horas (40%)
7. ¿Cuál es el uso principal que le da a su móvil? (Respuesta múltiple)
    a. Llamadas (69,6%)
    b. Mensajería instantánea (92,9%)
    c. Redes Sociales (42,9%)
    d. Reproductor música (26,8%)
    e. Ver películas/Series (10,7%)
    f. Leer (8,9%)
    g. Navegar (50%)
    h. Otras aplicaciones (23,2%)
8. ¿En qué momento usa su móvil? (Respuesta múltiple)
    a. Mientras estudio (10,7%)
    b. Comiendo (41,1%)
    c. En el trabajo (58,9%)
    d. Practicando deporte (28,6%)
    e. Otras opciones (58,9%)
9. ¿Con qué frecuencia consulta su móvil durante el día?
    a. Menos de 10 minutos (8,9%)
    b. Entre 10 minutos y 1 hora (51,8%)
    c. Entre 1 y 3 horas (30,4%)
    d. Más de 3 horas (8,9%)
10. ¿Sabría qué sistema operativo tiene su móvil?
    a. Android (67,9%)
    b. iOS (26,8%)
    c. Windows Phone (1,8%)
    d. Lo desconozco (3,6%)
11. ¿Le gusta practicar deporte?
    a. Sí (96,4%)
    b. No (3,6%)
12. ¿Con qué frecuencia lo practica durante la semana?
    a. Nunca (1,8%)
    b. 1 vez a la semana (10,7%)
    c. 2-3 veces (58,9%)
    d. 4 o más veces (28,6%)
13. ¿Se considera deportista de fin de semana?
    a. Sí (39,3%)
    b. No (60,7%)

14. ¿Utiliza el móvil cuando lo practica?
    a. Sí (66,1%)
    b. No (33,9%)
15. En caso afirmativo, ¿para qué lo utiliza? (Respuesta múltiple)
    a. Música (37%)
    b. Radio (5,6%)
    c. Registrar entrenos (53,7%)
    d. Seguir rutas (18,5%)
    e. Otros usos (33,3%)
16. ¿Registra sus actividades en algún dispositivo?
    a. Sí (89,3%)
    b. No (10,7%)
17. ¿Qué tipo de dispositivo? (Respuesta múltiple)
    a. Reloj deportivo (55,6%)
    b. GPS deportivo (50%%)
    c. Móvil (31,5%)
    d. Otro (3,7%)
18. En caso de responder reloj deportivo, ¿sabría la marca?
    a. Suunto (6,3%)
    b. Apple Watch (6,3%)
    c. Garmin (46,9%)
    d. Polar (21,9%)
    e. Otro (18,8%)
19. En caso de responder GPS deportivo, ¿sabría la marca?
    a. Mio (0%)
    b. Garmin (80%)
    c. Polar (14,3%)
    d. Otro (5,7%)
20. Cuando entrena, ¿qué le gusta registrar? (Respuesta múltiple)
    a. Tiempo (75,5%)
    b. Distancia (100%)
    c. Desnivel (77,4%)
    d. Otro (18,9%)
21. Utiliza la plataforma Strava
    a. Sí (78,6%)
    b. No (21,4%)
22. ¿Le resulta sencillo enviar sus actividades a Strava?
    a. Sí (87,3%)
    b. No (12,7%)
23. ¿Ha enlazado alguno de sus dispositivos de deporte a su cuenta de Strava?
    a. Ninguno (20%)
    b. 1 (47,3%)
    c. 2 (23,6%)
    d. Más de 2 (9,1%)
24. ¿Qué actividades practica más?
    a. Indoor (sala) (8,9%)
    b. Outdoor (exterior) (91,1%)
    c.
25. ¿Qué proporción de *indoor/outdoor* suele practicar

a. 100% indoor/ 0% outdoor (3,6%)
b. 75% indoor/ 25% outdoor (14,3%)
c. 50% indoor/ 50% outdoor (10,7%)
d. 25% indoor/ 75% outdoor (28,6%)
e. 0% indoor/ 100% outdoor (42,9%)

26. ¿Cuál considera que es su actividad principal?
    a. Running (23,2%)
    b. BTT (60,7%)
    c. Ciclismo carretera (10,7%)
    d. Cinta (1,8%)
    e. Spinning (0%)
    f. Otro (3,6%)
27. ¿Y la segunda?
    a. Running (25%)
    b. BTT (26,9%)
    c. Ciclismo carretera (23,1%)
    d. Cinta (0%)
    e. Spinning (7,7%)
    f. Otro (17,3%)
28. ¿Le gustaría competir con sus contactos en lo referente a actividad física?
    a. Sí (44,6%)
    b. No (55,4%)
29. En caso de medirse con sus amigos, lo haría por:
    a. Demostrar ser el mejor (3,6%)
    b. Diversión (40%)
    c. Generar dinámica de grupo (23,6%)
    d. Entrenar con más frecuencia (29,1%)
    e. Otros (0%)
    f. No competiría (3,6%)
30. ¿Pagaría algún concepto por participar en retos?
    a. Sí (17,9%)
    b. No (82,1%)
31. En caso afirmativo, ¿qué modalidad cree que sería de pago?
    a. Pagar por NO publicidad (31,8%)
    b. Pagar por participar (22,7%)
    c. Donación (45,5 %)