Universidad
Rey Juan Carlos

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

GRADO EN INGENIERÍA DEL SOFTWARE

Curso Académico 2022/2023

Trabajo Fin de Grado

DESARROLLO DE UNA APLICACIÓN WEB PARA LA PLANIFICACIÓN DE ENTRENAMIENTOS DESTINADOS A CORREDORES

Autor: Diego Jiménez Fernández-Pacheco

Tutor: Alberto Herrán González

Agradecimientos

Quería agradecerle este trabajo a mis padres, mi novia y mi tutor, los cuatro por apoyarme hasta terminarlo.

Resumen

Este proyecto está basado en el tipo de entrenamiento *FIRST*, que son las siglas del Furman Institute of Running and Scientific Training en Carolina del Sur. Este entrenamiento estaba inicialmente representado en un documento documento excel donde, introduciendo los datos actuales de ritmo y la organización de días, te permite generar un entrenamiento.

Los objetivos principales del proyecto son los siguientes:

- Crear una aplicación web que permita iniciar sesión e integrarse con la aplicación *Strava*.
- Permitir a los usuarios sincronizar sus datos desde *Strava* hacia la aplicación.
- Poder generar planes de entrenamiento semanales siguiendo la funcionalidad del documento excel.
- Los usuarios tendrán la capacidad de revisar los planes de entrenamiento junto con los resultados sincronizados y su rendimiento.

Para conseguir estos objetivos se ha comenzado analizando el funcionamiento del documento excel *FIRST*. Con este análisis se ha creado una base de datos y un modelo de base de datos para persistir la información.

Posteriormente se ha creado una aplicación Web utilizando HTML, Less y Javascript que replica la funcionalidad encontrada en el documento *FIRST*,

junto con funcionalidad adicional como la integración con *Strava* y el análisis de los datos.

Finalmente, se han añadido en *Strava* varias carreras simulando un entrenamiento y se han realizado extensivas pruebas para validar que el funcionamiento de la aplicación es el correcto y sigue las especificaciones recogidas en el análisis.

Palabras clave

- Strava
- Aplicación web
- Entrenamiento *FIRST*

Índice de contenidos

Índice de figuras

1 Introducción

En este capítulo se describe el proyecto y las razones que han motivado este proyecto para salir adelante. También se revisará las herramientas actuales que hay en el mercado y los objetivos que se han querido llegar a cumplir.

1.1. Motivación

Actualmente el running es uno de los deportes más realizados a nivel global, practicado por muchas personas y teniendo un nivel competitivo que abarca muchos niveles. Se pueden encontrar muchas maratones famosas en las que llegan a participar un gran número de competidores. Además, es el entrenamiento para mejorar la resistencia cardiorrespiratoria más realizado.

Este trabajo nace al conocer el entrenamiento *FIRST*. Este entrenamiento sirve para preparar carreras de 5 o 10 kilómetros, media maratón o incluso una maratón [1]. Este entrenamiento consiste en llevar una consistencia en unos entrenamientos semanales, que gradualmente van reduciendo los tiempos objetivos para conseguir llegar a un tiempo mejorado. Estos entrenamientos están divididos en series largas, series cortas y entrenamientos cruzados.

- **Series largas**: Son aquellas series de carrera a pie con distancias más largas, que aumentan la resistencia del atleta. Estas series tienen mayores distancias, pero se realizan con ritmos más bajos.

- **Series cortas**: Son series de carrera a pie con distancias más cortas, pero ritmos más altos. Estas series sirven para aumentar la resistencia del atleta al llevar velocidades más altas. Tanto las series largas como las series cortas van aumentando las distancias según van pasando las semanas.

- **Entrenamientos cruzados**: Otros tipos de entrenamientos que permiten trabajar a frecuencias cardiacas más bajas que en las series de carrera a pie. Algunos entrenamientos cruzados de ejemplo son la natación, el ciclismo o una sesión de gimnasio.

Este entrenamiento aprovecha los beneficios de los entrenamientos cruzados haciendo que el entrenamiento sea mucho más flexible y menos aburrido que realizar solo carreras a pie, produciendo un alto nivel de acondicionamiento general, una reducción del riesgo de lesiones y mejorando la agilidad y el equilibrio.

Para llevar a cabo este entrenamiento se ha basado el proyecto en una herramienta Excel que permite llevar un control de los entrenamientos, además de generar diferentes entrenamientos según lo vayamos necesitando. Esta herramienta tiene ciertas limitaciones a la hora de usarse, que provocan que se deje de utilizar y se opte por otras vías de entrenamiento. Las limitaciones son las siguientes:

- La **configuración inicial** de los entrenamientos es complicada, sobre todo si no se tiene conocimiento sobre las distintas siglas del entorno de los entrenamientos. La configuración incluye elegir el día de inicio, el orden de los entrenamientos y el tipo de la carrera objetivo para la que se va a realizar el entrenamiento. Además, se debe introducir el ritmo inicial con el que comienza el atleta.

- Otra limitación existente es debido a que la **funcionalidad está en un archivo Excel**. Esto provoca que se tengan que ejecutar macros en el

documento, disminuyendo la seguridad de cara a compartir el archivo a través de vías online.

- En el momento de actualizar los resultados se tiene que ir **añadiendo uno por uno** todos los tiempos que has hecho en cada serie. Para eso es necesario tener esos tiempos registrados en alguna aplicación y extraerlos al documento. Esto provoca que el documento solamente se use para la generación de los entrenamientos y posteriormente el abandono del mismo.

- La **utilización y rellenado** del excel es demasiado **manual**. Si el usuario tiene la necesidad de consultar el estado en el que lleva el entrenamiento necesita introducir todos los datos manualmente.

Teniendo en cuenta estas limitaciones se ha optado por realizar una mejora tecnológica basándola en la funcionalidad encontrada en la herramienta Excel, que permita una utilización más sencilla de la herramienta, junto con mejoras sustanciales que aumenten el grado de consistencia de utilización de la misma por los atletas.

1.2. Estudio del mercado

Actualmente en el mercado se pueden encontrar una gran variedad de herramientas que permiten crear, editar y llevar un registro de diferentes tipos de entrenamiento. Estas herramientas están más centradas en entrenamientos de gimnasio para fortalecer músculos, pero no se encuentra ninguna basada en el método *FIRST*.

Como ejemplo encontramos la aplicación de MadMuscles [2] (Ver figura 1.1, de pago, que permite crear planes de entrenamiento para gimnasio, junto con un control de la dieta y que ayuda a crear el hábito de entrenar todas las semanas.

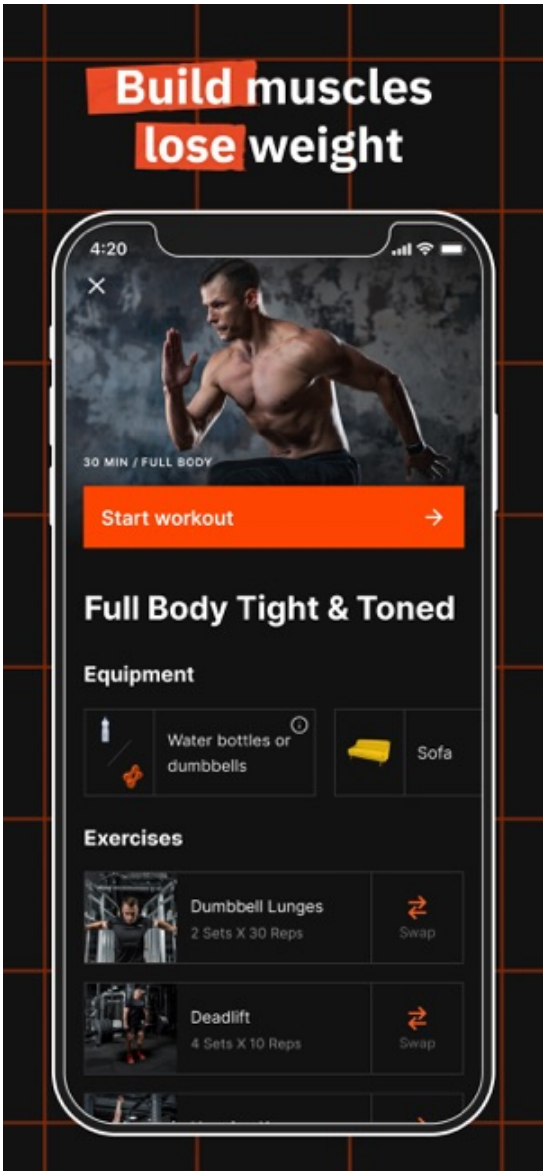


Figura 1.1: Aplicación MadMuscles

Sobre entrenamiento *FIRST* sólamente podemos encontrar páginas web como por ejemplo Foroatletismo [3] o Ciclonomadas [4] con planes de entrenamiento ya diseñados y que no son escalables dependiendo del ritmo del atleta.

Podemos encontrar aplicaciones que proveen entrenamientos de running de carácter general como por ejemplo Runnea [5]. Esta aplicación es de pago y proporciona planes de entrenamientos running adaptados a tí.

Además de estos entrenamientos está el documento Excel en el que se ha

basado la aplicación.

1.3. Objetivos

El objetivo general de este trabajo es llegar a conseguir una aplicación web que permita llevar un control de los diferentes entrenamientos que se creen. Este objetivo se puede dividir en varios objetivos más específicos que ayudan a entender la envergadura del proyecto:

- Crear una **aplicación web en .NET Core**. Esta aplicación tiene que poder desplegarse como servicio en la nube de Azure.
- Tiene que existir un **pipeline de trabajo en GitHub** que permita **publicar automáticamente** la aplicación cada vez que exista una versión estable y actualizada de la aplicación.
- La aplicación tiene que permitir la **entrada de todos los usuarios que estén registrados en *Strava***. Cada usuario tendrá acceso a sus propios entrenamientos y no podrán ver entrenamientos de otros usuarios.
- Los usuarios tienen que tener la capacidad de **generar planes de entrenamiento semanales**. Estos planes de entrenamiento tienen que informar al usuario de las diferentes actividades que tienen que llevar a cabo cada día de la semana, durante las semanas que dure el entrenamiento.
- Cuando un usuario actualice en *Strava* los tiempos realizados, la aplicación tiene que ser capaz de **descargar esos tiempos y actualizar** en la base de datos los tiempos, comparándolos con los tiempos objetivos informados al usuario. Los usuarios tienen que poder consultar esos tiempos en la aplicación directamente, además de tener información ampliada sobre el rendimiento ejercido.

Además de estos objetivos generales y específicos, también hay objetivos que se consiguen por el simple hecho de hacer el trabajo. Estos son:

- Aprender el lenguaje **C#** y el funcionamiento del API de *Strava* que nos permite hacer las integraciones necesarias.
- Perfeccionar el conocimiento en el uso y modelación de una base de datos **Microsoft SQL Server**.

1.4. Estructura de la memoria

El resto de la presente memoria se divide en siete capítulos:

En el **Capítulo 2** se hace un repaso sobre la generación de los diferentes entrenamientos y como se van a adaptar a nuestra aplicación.

En el **Capítulo 3** se comentan las tecnologías y metodologías utilizados para el desarrollo del proyecto.

En el **Capítulo 4** se desarrollan el análisis y los requisitos tomados, junto con diagramas UML y explicaciones del funcionamiento de la aplicación.

En el **Capítulo 5** se explica la arquitectura de la aplicación y varios ejemplos de flujos de la misma.

En el **Capítulo 6** se comenta todo el desarrollo realizado durante los distintos Sprints llevados a cabo.

En el **Capítulo 7** se desarrollan los resultados obtenidos en la aplicación y varios casos de uso.

Finalmente, en el **Capítulo 8** se plantean las conclusiones extraídas tras realizar este trabajo, los objetivos cumplidos y futuros trabajos de ampliación.

2

Generación de entrenamientos

En este capítulo se revisan los distintos planes de entrenamientos disponibles, el plan de entrenamiento *FIRST* y como aplicarlo. También se verá como funciona la plataforma *Strava* y como vamos a utilizarla en el proyecto.

2.1. Repaso de diferentes planes de entrenamiento existentes

Actualmente podemos encontrar varios planes de entrenamiento ya existentes que ayudan a desarrollar un hábito y mejorar los tiempos de las carreras del atleta. Algunos de estos planes de entrenamiento son los siguientes:

- En la página web **SportsSantander** [6] (Ver figura 2.1) se pueden encontrar varios planes de entrenamiento para preparar carreras de varias distancias. Estos planes de entrenamientos engloban carreras de 5, 10, 21 y 42 kilómetros, a partir de las cuales se dividen en los distintos entrenamientos dependiendo del ritmo que quieras llevar, como por ejemplo hacer 21 kilómetros en dos horas.

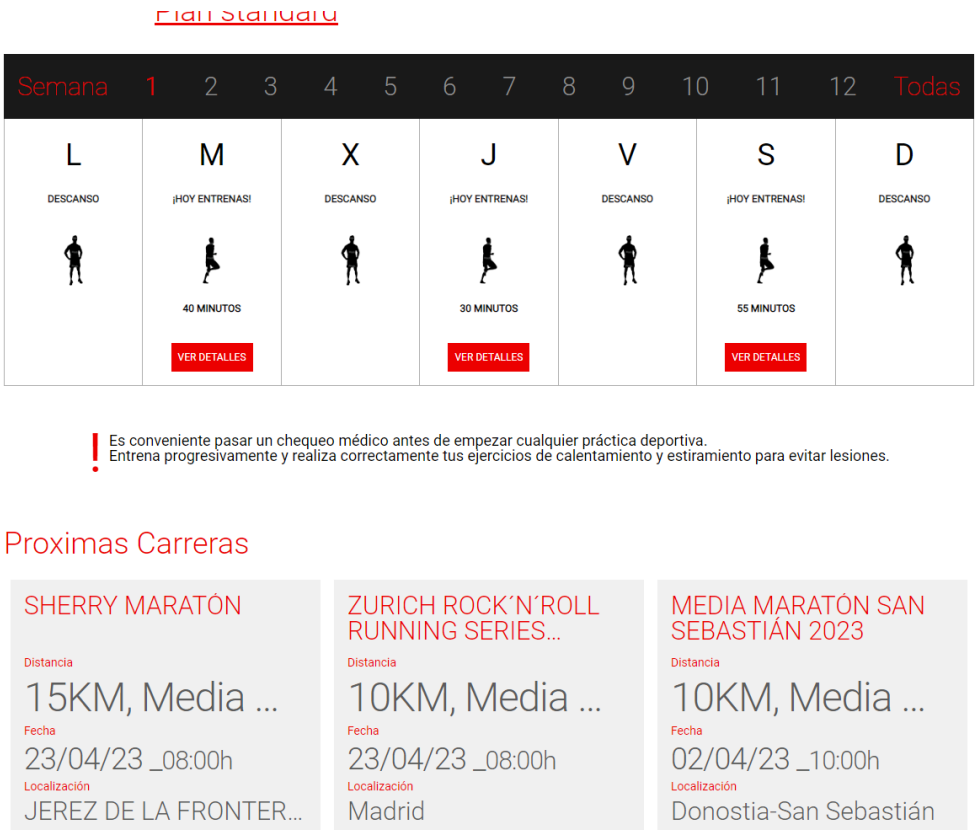


Figura 2.1: Plan de entrenamiento de SportsSantander

Estos entrenamientos se basan en 3 o 4 entrenamientos semanales durante 12 semanas, que preparan al atleta para la distancia y tiempo objetivos. Además, tiene entrenamientos cruzados durante los ejercicios como hacer flexiones o jumping jacks. Esta página web además te permite consultar las siguientes carreras que van a ocurrir en España en poco tiempo.

- En la página web MarathonRanking [7] se encuentran distintos planes de entrenamiento para preparar carreras de 5, 10, 21 y 42 kilómetros, y de igual manera que la página web anterior, estos entrenamientos se dividen dependiendo del tiempo final que quieras preparar en la distancia elegida.

  En cambio, estos entrenamientos varían en la cantidad de semanas dependiendo de la distancia a preparar. Por ejemplo la preparación una carrera de 10 kilómetros dura 8 semanas y en cambio la preparación una carrera de 42 kilómetros dura 16 semanas.

2.2. Entrenamiento *FIRST*

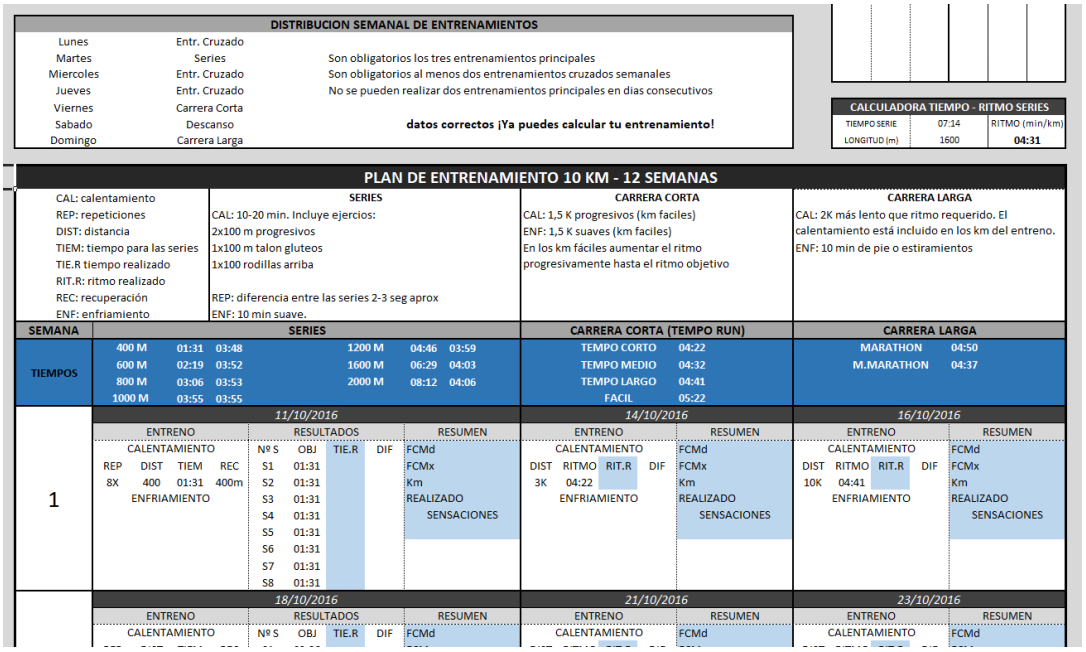
El entrenamiento *FIRST* es un entrenamiento definido en el libro "*Runners World, Run Less, Run Faster: Become a Faster, Stronger Runner with the Revolutionary First Training Program*"del *Furman Institute of Running & Scientific Training* (*FIRST*). En este libro se describen diferentes planes de entrenamiento de 8 a 16 semanas pensado en atletas que quieran aumentar sus tiempos pero que tengan poco tiempo para dedicarlo a entrenar [3].

Durante cada semana se tienen que hacer tres entrenamientos principales, que deben estar separados entre sí al menos por un día. Los tres entrenamientos principales son:

- **Series**: Es un entrenamiento donde se realizan diferentes series de entre 400 y 1500 metros, donde el ritmo es un poco más bajo que el ritmo máximo del atleta para la carrera a entrenar.
- **Carrera corta**: Son entrenamientos donde el atleta realiza carreras de entre 5 y 8 kilómetros, pero aumentando el ritmo. Permiten entrenar la resistencia a la velocidad, adaptando al atleta para que en las carreras largas pueda aumentar el ritmo y aguantarlo sin problemas.
- **Carrera larga**: Son entrenamientos donde se realizan carreras de más duración a un ritmo más relajado. Estas carreras permiten ir adaptándose al atleta a la distancia y ritmo objetivos, aumentando la resistencia. Las distancias que se suelen realizar en estas carreras son de 15 a 30 kilómetros.

Todos estos entrenamientos tienen un tiempo de calentamiento y un tiempo de enfriamiento antes y después de llevar a cabo los entrenamientos principales.

Estos datos se pueden observar en el documento excel (ver figura 2.2). En este documento se pueden observar varios entrenamientos distintos basados en la carrera que se quiera preparar. Tiene tiempos preparados sobre las distintas series, carreras cortas y carreras largas dependiendo del tiempo inicial que inserte el usuario.



DISTRIBUCION SEMANAL DE ENTRENAMIENTOS

Lunes	Entr. Cruzado	
Martes	Series	Son obligatorios los tres entrenamientos principales
Miercoles	Entr. Cruzado	Son obligatorios al menos dos entrenamientos cruzados semanales
Jueves	Entr. Cruzado	No se pueden realizar dos entrenamientos principales en dias consecutivos
Viernes	Carrera Corta	
Sabado	Descanso	**datos correctos ¡Ya puedes calcular tu entrenamiento!**
Domingo	Carrera Larga	

CALCULADORA TIEMPO - RITMO SERIES

TIEMPO SERIE	07:14	RITMO (min/km)
LONGITUD (m)	1600	**04:31**

PLAN DE ENTRENAMIENTO 10 KM - 12 SEMANAS

CAL: calentamiento	**SERIES**	**CARRERA CORTA**	**CARRERA LARGA**
REP: repeticiones	CAL: 10-20 min. Incluye ejercios:	CAL: 1,5 K progresivos (km faciles)	CAL: 2K más lento que ritmo requerido. El
DIST: distancia	2x100 m progresivos	ENF: 1,5 K suaves (km faciles)	calentamiento está incluido en los km del entreno.
TIEM: tiempo para las series	1x100 m talon gluteos	En los km fáciles aumentar el ritmo	ENF: 10 min de pie o estiramientos
TIE.R tiempo realizado	1x100 rodillas arriba	progresivamente hasta el ritmo objetivo	
RIT.R: ritmo realizado			
REC: recuperación	REP: diferencia entre las series 2-3 seg aprox		
ENF: enfriamiento	ENF: 10 min suave.		

SEMANA	SERIES	CARRERA CORTA (TEMPO RUN)	CARRERA LARGA
TIEMPOS	400 M 01:31 03:48 · 1200 M 04:46 03:59 · 600 M 02:19 03:52 · 1600 M 06:29 04:03 · 800 M 03:06 03:53 · 2000 M 08:12 04:06 · 1000 M 03:55 03:55	TEMPO CORTO 04:22 · TEMPO MEDIO 04:32 · TEMPO LARGO 04:41 · FACIL 05:22	MARATHON 04:50 · M.MARATHON 04:37
1	11/10/2016	14/10/2016	16/10/2016
	ENTRENO: CALENTAMIENTO · REP 8X · DIST 400 · TIEM 01:31 · REC 400m · ENFRIAMIENTO	ENTRENO: CALENTAMIENTO · DIST 3K · RITMO 04:22 · RIT.R · DIF · ENFRIAMIENTO	ENTRENO: CALENTAMIENTO · DIST 10K · RITMO 04:41 · RIT.R · DIF · ENFRIAMIENTO
	RESULTADOS: Nº S · OBJ · TIE.R · DIF — S1 01:31 · S2 01:31 · S3 01:31 · S4 01:31 · S5 01:31 · S6 01:31 · S7 01:31 · S8 01:31	RESUMEN: FCMd · FCMx · Km · REALIZADO · SENSACIONES	RESUMEN: FCMd · FCMx · Km · REALIZADO · SENSACIONES
	18/10/2016	21/10/2016	23/10/2016
	ENTRENO: CALENTAMIENTO	ENTRENO: CALENTAMIENTO	ENTRENO: CALENTAMIENTO
	RESULTADOS: Nº S · OBJ · TIE.R · DIF	RESUMEN: FCMd	RESUMEN: FCMd

Figura 2.2: Excel con entrenamientos *FIRST*

El entrenamiento *FIRST* aprovecha los entrenamientos cruzados para hacer la preparación de las carreras más dinámicas, además de aumentar la resistencia del atleta. Estos entrenamientos pueden ser:

- **Entrenamientos aeróbicos**: Son actividades como por ejemplo montar en bicicleta, nadar o saltar a la comba. Son ejercicios que no requieren de mucho esfuerzo y que ayudan a eliminar agujetas, además de entrenar otros grupos musculares, previniendo los desequilibrios musculares creados por la carrera.

- **Acondicionamiento físico**: Son aquellos ejercicios realizados en un gimnasio para aumentar la musculación de grupos musculares que no se entrenan durante las carreras. Además, ayudan a fortalecer el tronco para reducir y absorber los impactos durante las carreras.

- **Actividades dirigidas**: Son actividades en grupo que trabajarán los puntos débiles del atleta. El punto negativo de estos entrenamientos es que suelen ser entrenamientos intensos, por lo que lo mejor es realizarlos fuera de la temporada. Algún ejemplo de este tipo de entrenamiento son

Body-pump, pilates o Kick Boxing.

- **Actividades acuáticas**: Estas actividades ayudan a rehabilitar zonas de las piernas, ya que al ser actividades realizadas en el agua las rodillas no sufren casi ningún impacto. Algún ejemplo de este tipo de entrenamiento son Aquagym, Hydrogym o Aquaruning

2.3. Plataforma *Strava*

Strava [8] es una red social de fitness para deportistas que se centra en el seguimiento de actividades físicas y deportivas. Es una plataforma que permite a los usuarios registrar sus actividades de running, ciclismo, natación y otros deportes, utilizando un reloj inteligente o una aplicación móvil para realizar el seguimiento de la ruta, la velocidad, la distancia y otros datos relacionados con el entrenamiento.

Figura 2.3: Logo de *Strava*

Además de permitir a los usuarios registrar y analizar sus propias actividades, *Strava* también ofrece una función de seguimiento en tiempo real, que permite a los amigos y seguidores ver las actividades de los demás y animarlos durante el entrenamiento. También hay una función de segmentos, que son secciones específicas de una ruta que los usuarios pueden competir para establecer la mejor marca personal o compararse con otros usuarios.

Strava también ofrece una plataforma para que los usuarios se unan a clubes, se comuniquen con otros deportistas y encuentren eventos y desafíos relacionados con el deporte.

Strava posee una gran cantidad de información que se puede extraer a través de sus APIs. Estas APIs tienen documentación muy detallada en su pagina web [9], donde se pueden encontrar todas las llamadas con las que se pueden obtener

los datos desde el proyecto que se va a desarrollar.

Por lo tanto, durante el proyecto se utilizará *Strava* como aplicación principal para varias de las funciones que proveerá la aplicación resultante. *Strava* permite registrar los tiempos de los parciales y se utilizarán estos para obtener los que se han realizado durante los entrenamientos exactos, pudiendo compararlos con los registros creados que existen en la base de datos.

También se aprovechará la funcionalidad que posee *Strava* para poder realizar una integración con su sistema de usuarios a través del protocolo OAuth 2.0, obteniendo de esta manera un registro de usuarios externo a nuestra aplicación y con toda la funcionalidad completamente probada. Esto permite que los esfuerzos de programación se centren en la funcionalidad realmente importante de la aplicación.

3

Herramientas y metodología

En este capítulo se describirán las distintas herramientas que se han utilizado para la organización, el diseño y el desarrollo de la aplicación, junto con la metodología que se ha seguido para llevarla a cabo.

3.1. Herramientas

Durante el proyecto se han utilizado varias herramientas útiles para el desarrollo de cada una de las fases.

La principal herramienta que se ha utilizado para la organización del proyecto ha sido Trello. Es una herramienta de gestión de equipos que permite la creación y organización de las distintas tareas necesarias para llevar a cabo el proyecto.

Figura 3.1: Logo de Trello

Se ha gestionado un tablero de Scrum con las columnas Backlog, To Do, Doing, Done, permitiendo la gestión durante cada Sprint de las tareas que tenían que llevarse a cabo y las tareas pendientes. Durante las reuniones para definir los requisitos se ha utilizado Trello para recoger los requisitos iniciales necesarios para el análisis.

Durante la fase de análisis del proyecto se han utilizado las siguientes herramientas:

- **Diagrams.net**: Ésta página web permite realizar diagramas de todo tipo, muy útil para cualquier esquema que se quiera realizar y con una gran facilidad de uso [10] (Ver figura 3.2). En el caso del proyecto actual se ha utilizado para realizar los esquemas UML que se pueden observar más adelante.

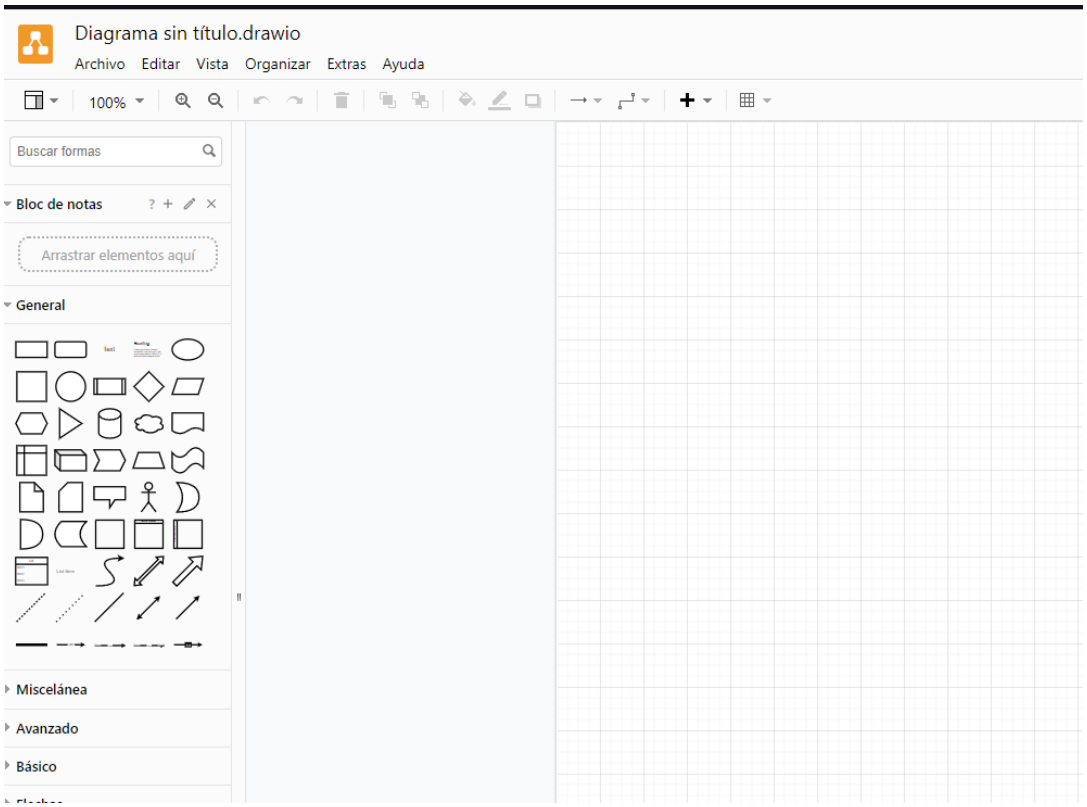


Figura 3.2: Pantalla principal de diagrams

- **Marvelapp**: Es una página web que se utiliza para realizar prototipos sin funcionalidad pero navegables de páginas web [11] (Ver figura 3.3). En el

caso del proyecto se ha utilizado para generar un prototipo inicial a partir del que comenzar el desarrollo. La utilidad proporciona la capacidad de modificar rápidamente las pantallas para poder mostrarlas a un cliente y facilitar la aceptación del prototipo.

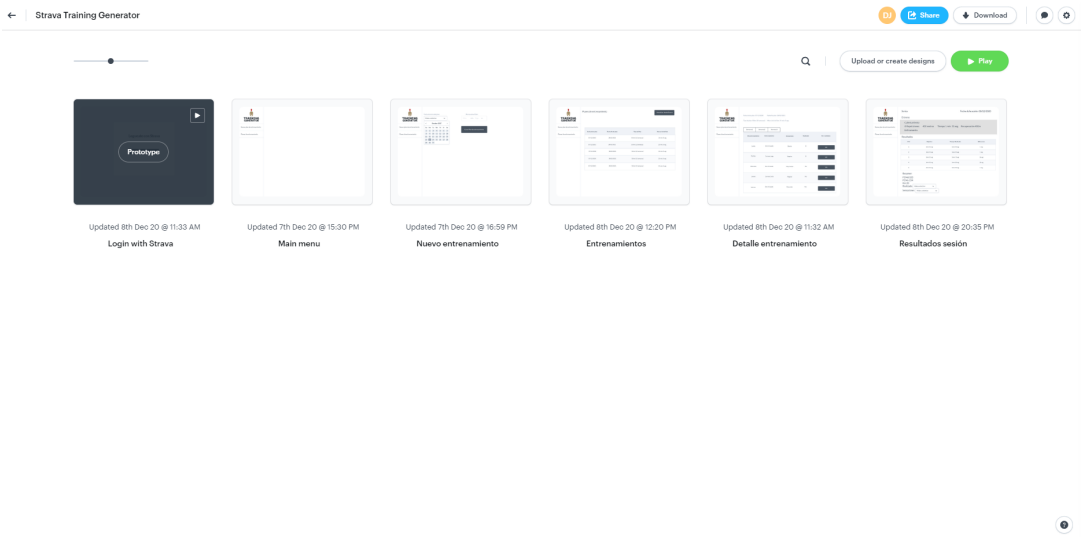


Figura 3.3: Pantalla principal de marvelapp

Para la fase de desarrollo se han utilizado varias herramientas y librerías que han facilitado el desarrollo de la aplicación:

- **Visual Studio 2022**: Es la herramienta principal para realizar desarrollos en tecnología .Net. Se ha utilizado la última versión de Visual Studio ya que tiene más facilidades a la hora de desarrollar, como por ejemplo el uso de una IA para el autocompletado de las líneas de código a partir del código que ya se ha escrito.

- **Bootstrap**: Para la maqueta de la aplicación se ha utilizado la librería de Bootstrap. Esta librería permite la utilización de componentes preconstruidos, como botones, menús desplegables o formularios, que facilitan la maquetación de las distintas pantallas de la aplicación.

- **Less**: Less es un lenguaje de hojas de estilo que puede compilarse en CSS. Es más robusto y fácil de utilizar que CSS y permite la utilización de variables y funciones para realizar cálculos.

- **HTML y Javascript**: Se han utilizado las dos principales herramientas para la creación de la página web. HTML permite la estructuración de la información que posteriormente será representada por el navegador web apoyandose en el código CSS compilado de los archivos Less. Javascript permite darle la funcionalidad necesaria para que la página web funcione correctamente.

- **EnterpriseLibrary**: Para la conexión con la base de datos se ha utilizado EnterpriseLibrary, que es una librería que, a parte de sus muchas otras funcionalidades, facilita las conexiones con la base de datos, además de añadir capas de seguridad contra ataques a ésta como por ejemplo SQL Injection.

- **JsGrid**: JsGrid es un plugin de JQuery que permite la creación de tablas dinámicas a partir de código Javascript de una manera muy sencilla [12]. Se ha utilizado para la creación de las distintas rejillas dinámicas que son requeridas para mostrar la información necesaria al usuario.

Durante la fase de despliegue se ha utilizado la herramienta de Github Actions [13]. Esta herramienta perteneciente a GitHub permite la ejecución de distintas pipelines tras realizar una subida al repositorio al que están relacionadas. Las pipelines se han utilizado para llevar a cabo el continuous deployment, que permite realizar publicaciones y entregas automáticas desde el propio repositorio de github directamente hacia azure.

Para el despliegue de la aplicación hemos utilizado el portal de Azure. Azure es una plataforma de computación en la nube desarrollada por Microsoft, que permite a los usuarios ejecutar y administrar aplicaciones y servicios en la nube, utilizando los servidores y el almacenamiento de Microsoft. Esto permite la capacidad de escalado automático de las diferentes estancias de las aplicaciones dependiendo del tamaño de cada aplicación.

Para desplegar la aplicación desarrollada se ha utilizado un App Service, que es un servicio proporcionado por Azure que permite la ejecución de una página web. También se ha desplegado una base de datos SQL Server en Azure,

consiguiendo con esto la capacidad de funcionamiento completo desde la nube del proyecto.

3.2. Metodología

Para la realización de este proyecto se ha seguido la metodología **Scrum** [14]. Scrum es una metodología Ágil que se basa en un enfoque iterativo e incremental, en el que el trabajo se divide en sprints de 1 a 4 semanas de duración.

Antes de comenzar cada sprint se lleva a cabo una reunión de Sprint donde se definen las tareas que se quieren llevar a cabo durante ese Sprint. Las tareas definidas tienen como objetivo proporcionar la mayor cantidad de funcionalidad y proporcionar un producto potencialmente utilizable al final de cada Sprint.

En el caso de este proyecto, se han definido 5 Sprints de 2 semanas de duración. Cada sprint se ha utilizado para aplicar el siguiente paso necesario para la culminación del proyecto, dividiendo los sprints de la siguiente manera.

El **primer sprint** se basó en definir los requisitos funcionales y diseñar un prototipo para la aplicación, generando de esta manera un listado de requisitos y una base para las pantallas de la aplicación y la funcionalidad que se va a realizar. Realizar el prototipo en este Sprint permite que el cliente tenga una idea general de como será la aplicación, aumentando de esta manera la utilidad del análisis descrito.

Durante el **segundo Sprint** se realizó la maquetación de la aplicación y la conexión de estas pantallas. Con esto el cliente tiene la capacidad de ver la aplicación en la página web directamente desde el navegador con el aspecto final que tendrá la aplicación y permite dar feedback sobre funcionalidades que puede que no tuviese claras desde un principio.

Tras esto, en el **tercer Sprint** se llevó a cabo el modelado de la base de datos. Este Sprint no proporciona ninguna funcionalidad visible para el cliente, pero permite que el futuro desarrollo de la aplicación sea mucho más efectivo. Aún así, se puede utilizar el modelado de la base de datos para mostrarle al

cliente como van a guardarse los datos y que éste de feedback sobre posibles campos faltantes en el modelo.

Teniendo la base de datos, en el **cuarto Sprint** se llevó a cabo la programación principal de la aplicación. Durante este Sprint se conectaron las distintas pantallas que estaban ya maquetadas con la base de datos modelada en el anterior Sprint. Esto proporciona que el resultado de este Sprint sea una aplicación que el cliente puede probar casi completamente, y proveer de feedback para realizar correcciones o mejoras.

Como **Sprint final** se ejecutó la integración con *Strava* y las distintas correcciones provenientes del feedback del sprint anterior. Teniendo ya todo el código desarrollado de la aplicación, se realizaron las diferentes conexiones con *Strava*, completando la funcionalidad restante. Además, se aplicaron las distintas correcciones y mejoras comentadas en el cuarto Sprint.

Finalizando el quinto Sprint se realizó una reunión con el cliente confirmando de esta manera que la aplicación estaba terminada y confirmando que la funcionalidad es la correcta.

4
Análisis

El sistema que se va a desarrollar es una aplicación web que permita la gestión de entrenamientos basados en el entrenamiento *FIRST*.

Los requisitos funcionales que se han obtenido de la descripción del problema son los siguientes:

4.1. Autenticación de usuarios

La aplicación requiere que los usuarios estén autenticados para la utilización de la aplicación. Para realizar la autenticación se utilizará la plataforma de *Strava*.

La autenticación con *Strava* se realizará siguiendo el protocolo OAuth 2.0. Cuando el usuario vuelve a la aplicación ya estaría identificado, por lo que los entrenamientos se podrán asignar a cada usuario individualmente.

El usuario identificado aparecerá en el menú lateral izquierdo, en la parte inferior.

4.2. Creación de entrenamientos

Estando los usuarios autenticados tendrán una opción para crear los distintos entrenamientos. Esta opción aparecerá en el menú lateral izquierdo de la aplicación.

En esta página el usuario podrá rellenar los datos necesarios para la creación de un entrenamiento, los cuales son:

- **Fecha de inicio del plan**: Esta fecha es obligatorio que sea un lunes, ya que todo el entrenamiento parte de la organización semanal de los entrenamientos.
- **Marca actual de 5 kilómetros**: Es necesario conocer el ritmo actual del atleta en los 5 kilómetros, para que todo el entrenamiento sea adecuado al tiempo real que realiza y no haya sobresfuerzos que puedan acabar siendo perjudiciales.
- **Plan de entrenamiento**: La aplicación va a proveer de varios planes de entrenamiento diferentes dependiendo de la distancia que se quiera entrenar. Las opciones serán entrenamientos de:
  - Entrenamiento para 5 kilómetros en 12 semanas.
  - Entrenamiento para 10 kilómetros en 12 semanas.
  - Entrenamiento para media maratón en 16 semanas.
  - Entrenamiento para una maratón completa de principiantes en 16 semanas.
  - Entrenamiento para una maratón completa en 16 semanas.
- **Fecha de nacimiento**: La aplicación utilizará la fecha de nacimiento para elegir adecuadamente el ritmo objetivo del atleta y así prevenir sobresfuerzos.

Tras seleccionar los campos anteriores, el usuario tendrá que elegir la distribución semanal de los entrenamientos. Esta distribución tiene que seguir el

siguiente conjunto de reglas:

- Son obligatorios los tres entrenamientos principales: Series, Carrera corta y carrera larga.
- Hay que añadir al menos dos entrenamientos cruzados semanales.
- Los tres entrenamientos principales tienen que estar separados por al menos un día entre medias.

Cuando el usuario acepte el formulario de creación de un entrenamiento se generará un nuevo entrenamiento y podrá entrar a ver los detalles.

4.3. Gestión de entrenamientos

En el menú lateral izquierdo habrá una opción para consultar todos los entrenamientos del usuario autenticado. Este menú llevará a una pantalla donde aparecerá una rejilla con los entrenamientos.

La rejilla mostrará información sobre cada uno de los diferentes entrenamientos y tendrá la capacidad de ir al detalle de cada entrenamiento realizando doble click en cada una de las filas.

La rejilla tendrá los siguientes campos:

- Fecha de inicio del plan: Será la fecha que el usuario haya seleccionado en la creación del plan de entrenamiento.
- Fecha de fin del plan: Será la fecha en la que acabe el plan de entrenamiento. Esta información permite al usuario tener un conocimiento a primera vista de cuánto le queda para terminar el entrenamiento y si este encaja con las fechas de futuras carreras.
- Tipo de plan: Será el tipo de plan que el usuario haya seleccionado en la creación del plan de entrenamiento.

- Marca inicial de 5 kilómetros: Será el ritmo que el usuario haya introducido en la creación del plan de entrenamiento. Permite al usuario ver la diferencia entre los distintos planes que haya creado y su evolución en el tiempo.

4.4. Visualización de plan de entrenamiento por semanas

Cuando el usuario haga doble click en una fila en la rejilla de entrenamientos entrará en el detalle de un entrenamiento. Esta vista tendrá toda la información relativa al entrenamiento seleccionado, que será:

- Fecha de inicio del plan: Será la fecha que haya seleccionado el usuario en la creación del plan de entrenamiento.
- Fecha de fin del plan: Será la fecha en la que acabe el plan de entrenamiento.
- Tipo de plan: Será el tipo de plan que el usuario haya introducido en la creación del plan de entrenamiento.
- Marca inicial de 5 kilómetros. Es el ritmo que el usuario haya introducido en la creación del plan de entrenamiento.

Además de toda esta información, existirán las distintas semanas del entrenamiento en forma de pestañas. Debajo de estas pestañas habrá una rejilla con la distribución de la semana de los distintos entrenamientos. Cuando se seleccione un entrenamiento en las pestañas se recargará la rejilla con los nuevos datos de la semana seleccionada y mostrará el día de comienzo de la semana.

La rejilla tendrá los siguientes datos:

- **Día de entrenamiento**: Será el día de la semana.
- **Entrenamiento**: Es el tipo de entrenamiento para ese día. Podrá ser Series, Carrera Corta, Carrera Larga, Entrenamiento cruzado o descanso.

- **Fecha de entrenamiento**: Será el día de mes exacto en el que cae el entrenamiento.
- **Realizado**: Indicará si el entrenamiento está terminado o programado.
- **Sensaciones**: Cuando se rellenen los datos automáticamente se añadirá un comentario sobre las sensaciones a partir de la frecuencia cardiaca.
- **Ver Resultados**: Aparecerá un botón para poder acceder al detalle del entrenamiento si el perteneciente a ese día es principal. En caso de ser un entrenamiento cruzado o un día de descanso aparecerá un botón para poder marcar el día como Completado.

4.5. Visualización de entrenamiento de un día

Al realizar click en el botón "Ver Resultados"de los entrenamientos principales definidos en la rejilla de entrenamiento semanal descrita en el anterior requisito el usuario se dirigirá a la pantalla de descripción del entrenamiento.

En la parte superior de esta pantalla se visualizará el tipo de entrenamiento, la frecuencia máxima obtenida cuando se recogen los datos, los kilómetros totales a realizar o realizados y el día del entrenamiento.

4.6. Datos del entrenamiento

En la pantalla anteriormente descrita podremos encontrar una rejilla con la información del entrenamiento a realizar. Saldrán para cada entrenamiento las distintas series a realizar, junto con los calentamientos, descansos y enfriamientos. En esta rejilla hay varios campos que describen el orden de los parciales en los que hay que realizar cada serie, junto con los resultados en el momento que se sincroniza la información con *Strava*. Alguno de estos campos cambiarán dependiendo de si el entrenamiento ha sido recogido desde *Strava* o no.

Los campos de la rejilla son:

- **Parcial**: Indica que tipo de parcial es la fila indicada.
- **Distancia**: Indica la distancia a realizar o la distancia realizada, dependiendo de si se han recogido los datos desde *Strava* o no. Indicado en metros.
- **Ritmo Objetivo**: Este campo indica el ritmo objetivo para cada uno de los parciales a realizar. Indica el ritmo medio en minutos y segundos en recorrer 5 kilómetros.
- **Ritmo realizado**: Este campo contiene el ritmo realizado por el atleta. En un principio el campo estará vacío, pero en el momento que se recojan los datos se mostrará. Indica el ritmo medio en minutos y segundos en recorrer 5 kilómetros.
- **Diferencia**: Indicará la diferencia en segundos entre el ritmo objetivo y el ritmo realizado. Este campo solamente aparecerá cuando estén los datos recogidos desde *Strava*. El campo tendrá el color del texto modificado en verde o rojo dependiendo de si es positivo o negativo.
- **Desnivel**: Indicará el desnivel del terreno recogido en *Strava*.
- **Frecuencia media**: Indicará la frecuencia media cardiaca recogida en *Strava*.
- **Cadencia media**: Indicará la cadencia media de pasos recogida en *Strava*.

4.7. Recuperación y comparación de datos con *Strava*

En la pantalla con el listado de entrenamientos habrá un botón que permitirá actualizar los datos utilizando la información extraída desde *Strava*. Este botón iniciará el proceso de importación de datos al sistema.

Esta funcionalidad ejecutará el siguiente proceso para el usuario conectado:

1. Existirá en base de datos un registro de la última sincronización del usuario.

2. A partir de la fecha de la última sincronización se recogerán todas las actividades del usuario conectado. Las actividades son los conjuntos de parciales que ha guardado el atleta en *Strava* a través de su dispositivo inteligente. Se recogerán un límite de 20 actividades en cada sincronización, para reducir el peso de cada sincronización.

3. A partir de las actividades se recogen las vueltas de cada actividad. Estas vueltas se asemejan a los parciales que existen en base de datos.

4. Con estas vueltas se crea un listado de posibles actualizaciones a realizar en la base de datos, y se envían a ésta para iniciar el proceso de coincidencia de los datos.

5. En la base de datos se unen los datos utilizando el día en el que se ha realizado el entrenamiento y el número de la serie recogido por *Strava* como split.

4.8. Análisis de resultados

Tras realizar la coincidencia de datos habrá que generar las comparaciones de los datos con los objetivos. Estas comparaciones se realizarán en el momento de presentar los datos, sin necesidad de almacenar las diferencias ya que la cantidad máxima de cálculos a realizar para cada pantalla a mostrar es baja.

Para la generación de sensaciones se generará en base de datos una función que permitirá el procesado de las sensaciones a partir de las frecuencias media y máxima en cada entrenamiento y la edad proporcionada por el usuario.

Las posibles sensaciones a mostrar son las siguientes:

- Muy mala
- Mala

- Regular
- Buena
- Muy buena

La función que seguirá el cálculo de las sensaciones es la siguiente:

$$threshold = min\left(max\left(\frac{5 \cdot (220 - yearsOld - maxFc)}{50}, 1\right), 5\right) \tag{4.1}$$

A 220 se le resta la edad del atleta y la frecuencia máxima, se multiplica por 5 y se divide entre 50. Esta fórmula suele dar un valor entre 1 y 5, aunque por si acaso se añade que el mínimo sea 1 y el máximo 5 aplicando las funciones matemáticas *max* y *min* (Ver ecuación 4.1).

5 Diseño

Para llevar a cabo el análisis descrito se adoptará una solución basada en una aplicación web monolítica. Esta aplicación estará desarrollada en .Net Core y estará conectada con una base de datos SQL Server.

5.1. Conectividad

La aplicación estará conectada con *Strava* mediante dos tipos de conexiones distintas:

- **Protocolo OAuth 2.0**: Para la gestión de los usuarios la aplicación se conectará con *Strava* siguiendo el protocolo OAuth 2.0, que permitirá la identificación de los usuarios de una manera segura. Este protocolo proporcionará un token que se utilizará para el acceso a través de API.

- **API**: Para la sincronización de los datos se accederá desde la aplicación a *Strava* a través de su API Rest. Esta conexión permitirá recoger los datos necesarios para la actualización de los valores en base de datos.

Estas conexiones seguirán el diagrama de conexión de la figura 5.1.

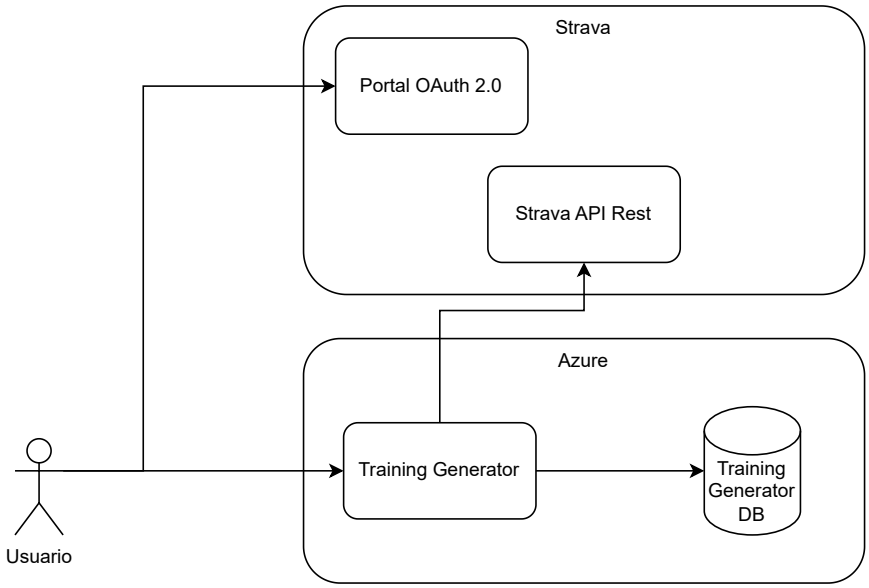


Figura 5.1: Diagrama de conexión

5.2. Flujo de la aplicación

Durante el análisis se definieron los dos principales flujos de trabajo en la aplicación.

El primer flujo está basado en la primera interacción de un usuario con la aplicación para generar un entrenamiento nuevo.

El segundo flujo está basado en las futuras interacciones del usuario con la aplicación para la actualización de los datos y la extracción de los mismos de *Strava*.

5.2.1. Creación de entrenamiento

Este flujo de la aplicación se puede encontrar reflejado en la figura 5.2.

Para comenzar este flujo el usuario accederá a la aplicación. En la primera pantalla observará un botón que permitirá identificarse en la aplicación utilizando *Strava*. Esto redirigirá al usuario a *Strava* para que registre un nuevo usuario

o se identifique con uno ya creado.

Tras esto, lo primero que tendrá que realizar el usuario es la creación de un entrenamiento. Para esto, desde la pantalla principal con el listado de entrenamientos hará click en el botón *Nuevo plan de entrenamientos* situado en el menú lateral izquierdo de la aplicación.

En esta pantalla rellenará los datos siguiendo las restricciones de creación del formulario. Cuando tenga completamente rellenos los campos presionará el botón *Crear plan de entrenamiento*. Esta acción generará el plan de entrenamiento en base de datos y redirigirá al usuario a la pantalla con el listado de los planes.

En este momento el usuario tendrá la capacidad de entrar en el detalle del plan de entrenamiento y visualizar los siguientes entrenamientos a realizar con la descripción de las distancias y los ritmos objetivos a realizar, junto con los calentamientos, descansos y enfriamientos.

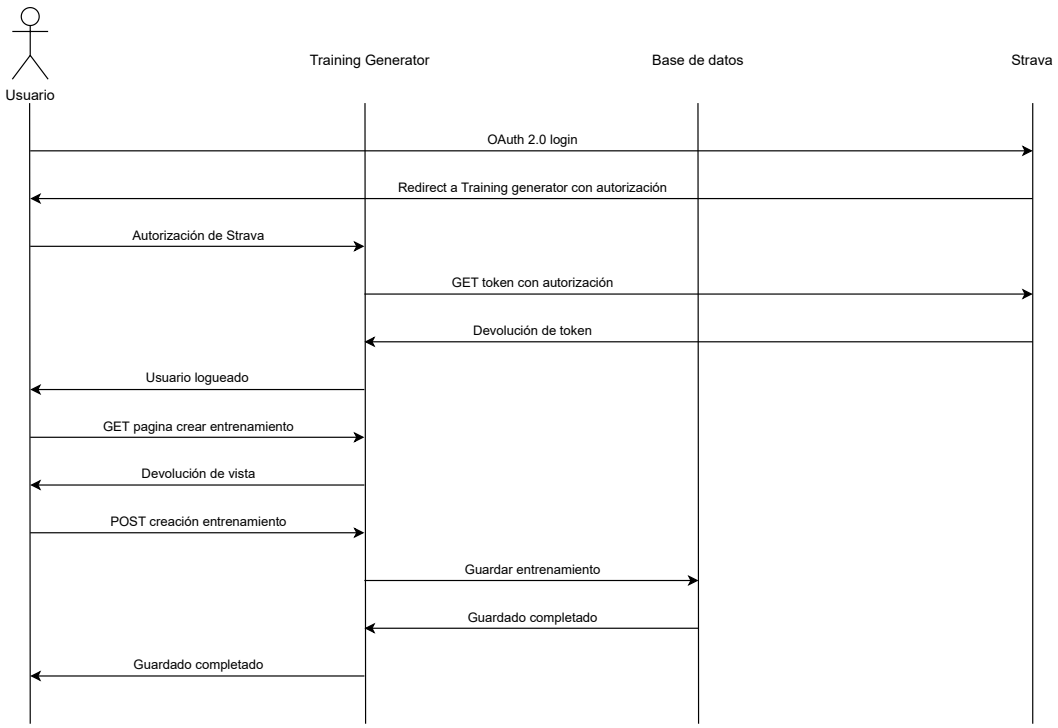


Figura 5.2: Flujo de creación de entrenamientos

5.2.2. Actualización y comparación de los datos

Este flujo de la aplicación se puede encontrar reflejado en la figura 5.3.

Se parte de la situación en la que un usuario ya ha creado un plan de entrenamiento y ha realizado los entrenamientos utilizando su dispositivo inteligente. Mediante este dispositivo inteligente ha cargado las actividades realizadas en *Strava*, y ha vuelto a ingresar en la aplicación *Training generator*.

En este momento el usuario se encontrará en la pantalla de listado de entrenamientos. Aquí hará click en el botón *Actualizar desde Strava* que ejecutará la sincronización de las actividades que ha realizado el usuario y cargado anteriormente en *Strava*.

Tras esta acción y la confirmación de que la sincronización ha sido efectuada correctamente el usuario accederá al detalle del entrenamiento que ha realizado durante la semana. En el detalle del entrenamiento podrá observar que las tres actividades principales se han marcado como completadas.

El usuario manualmente marcará como completadas el resto de actividades semanales que hay que marcar manualmente, siendo estas los entrenamientos cruzados y los descansos.

Tras esto el usuario hará click en un botón *Ver Resultados* de uno de los entrenamientos principales para acceder al detalle del entrenamiento.

En esta pantalla podrá observar los parciales que ha realizado y cargado en *Strava* actualizados en la rejilla. Podrá observar su rendimiento a partir del campo *Diferencia* situado en la tabla, que comparará su ritmo obtenido con el ritmo objetivo para el parcial. Además, podrá ver las sensaciones obtenidas en el entrenamiento y datos relativos a los entrenamientos realizados.

Con esta información el atleta podrá identificar si está siguiendo el plan de entrenamiento correctamente y además consultar los siguientes pasos a realizar la semana siguiente.

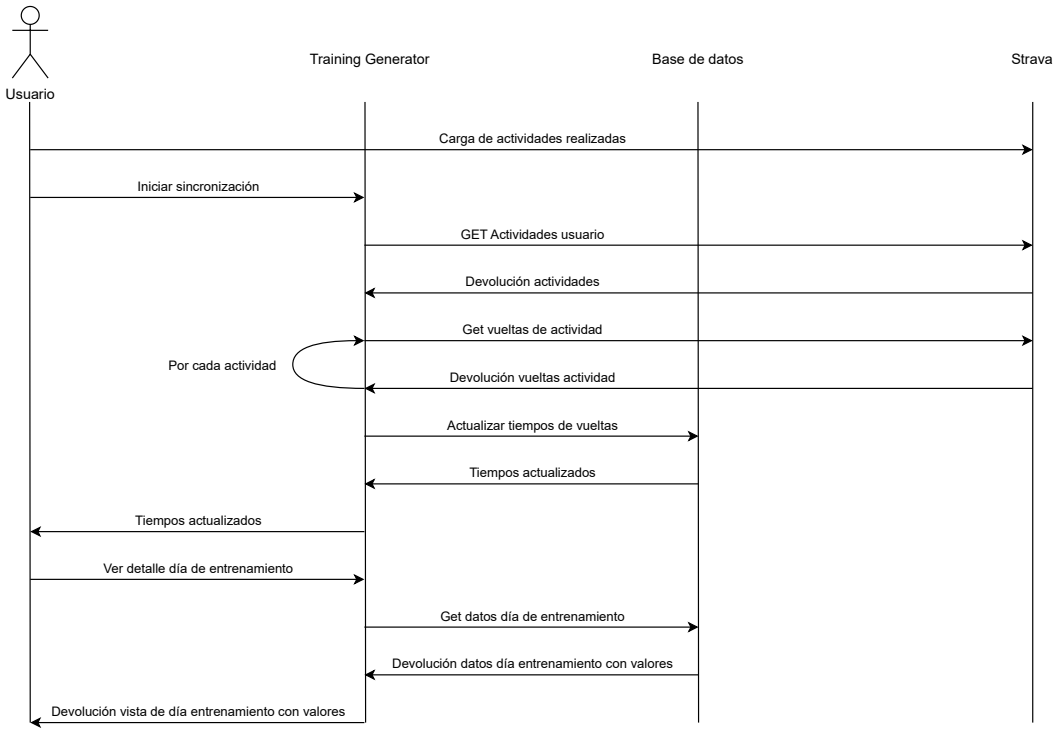


Figura 5.3: Flujo de actualización y comparación de los datos

5.3. Diseño de la aplicación .NET

El diseño del software de la aplicación .Net se basará en el patrón Modelo Vista Controlador, a partir de ahora patrón MVC; junto con una arquitectura interna de tres capas. Las capas definidas son las siguientes:

- **Capa de presentación**: Esta capa es el punto de entrada a la aplicación. Aquí se pueden encontrar los tres tipos de objetos necesarios para seguir el patrón MVC: Controladores, modelos y vistas. Los controladores y los modelos estarán programados en c# y las vistas en Razor, las cuales permitirán ejecutar código c# para terminar de definir el HTML necesario para presentar la aplicación. Esta capa también contendrá el código Javascript necesario para completar la funcionalidad de la aplicación. Junto con estos archivos se pueden encontrar todos los recursos estáticos necesarios para la correcta funcionalidad de la aplicación.
- **Capa de lógica de negocio**: Esta capa estará contenida en una librería

de clases nombrada BLL (Business Logic Layer) y contiene los archivos y objetos necesarios para contener toda la lógica de negocio de la aplicación. Esta capa será accesible por la capa de presentación.

- **Capa de acceso a datos**: Esta capa estará contenida en una librería de clases nombrada DAL (Database Access Layer) y contendrá las clases necesarias para realizar las conexiones con la base de datos y la gestión de los datos. Será accesible desde la capa de lógica de negocio.

En la figura 5.4 se muestra de forma visual la arquitectura de la aplicación.

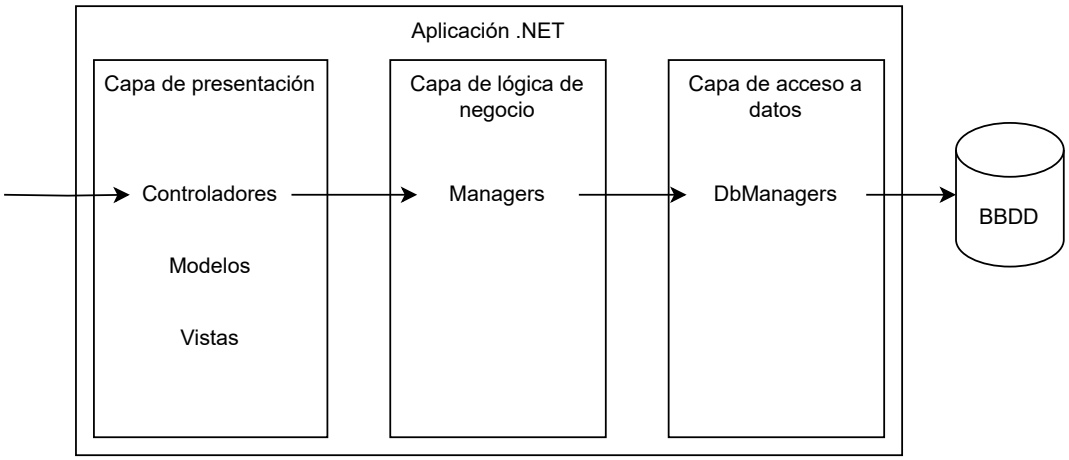


Figura 5.4: Arquitectura de la aplicación

Además de estas tres capas, la aplicación tendrá una librería de clases que estará dedicada a la conexión con *Strava* y la recogida de los datos. Esta última librería será accesible desde la capa de lógica de negocio y actuará de una forma parecida a la capa de acceso a datos, proporcionando los datos que provienen desde *Strava* a partir del token de acceso del usuario.

6
Desarrollo

El desarrollo de la aplicación se ha realizado dividido en varios Sprints. Cada sprint ha proporcionado al cliente un producto utilizable y revisable para que se pudiese ir adaptando en cada Sprint según las necesidades de los clientes.

Se han realizado 5 sprints diferentes y cada uno se ha centrado en un aspecto distinto de la aplicación. Además, en cada sprint se han realizado correcciones y mejoras a partir de feedback de los anteriores.

- Sprint 1: Prototipado de la aplicación.
- Sprint 2: Maquetación de la aplicación.
- Sprint 3: Definición de la base de datos.
- Sprint 4: Programación de la aplicación.
- Sprint 5: Integración con *Strava*.

6.1. Prototipado de la aplicación

En el primer Sprint se llevó a cabo el prototipado de la aplicación *Training Generator*, es decir, se definió la estructura que llevarían cada una de las pantallas, las conexiones entre ellas y el aspecto final que se ha definido para la aplicación.

Para realizar este prototipado se ha utilizado la herramienta *marvelapp* [11]. Esta herramienta permite colocar y diseñar de una manera sencilla y visual las distintas pantallas de la aplicación.

La primera pantalla es la pantalla de Login (ver figura 6.1). Esta pantalla será sencilla, con un botón que permitirá redirigir al usuario a *Strava* para loguearse y así acceder a la aplicación. Tras volver de *Strava*, el usuario se redirigirá a la pantalla principal de la aplicación.

Figura 6.1: Pantalla prototipo Login

En la pantalla principal de la aplicación se pueden observar dos opciones en el menú lateral. Estas opciones son *Nuevo plan de Entrenamiento* y *Planes de entrenamiento*. Además, este menú tendrá el nombre del usuario logueado y una opción para que el usuario cierre sesión. El menú lateral existirá en toda la aplicación exceptuando la pantalla de Inicio de sesión descrita anteriormente.

La primera opción de menú llevará a la pantalla de *Nuevo plan de entrenamiento* (ver figura 6.2). Esta pantalla mostrará los datos necesarios para crear un entrenamiento. En el prototipado se definieron dos campos: Fecha de inicio del plan y Marca actual de los 5km. Estos campos se fueron expandiendo durante los sprints según se ha ido estudiando el fichero excel y recibiendo feedback. Junto a esos campos encontramos el botón para crear el plan de entrenamiento.

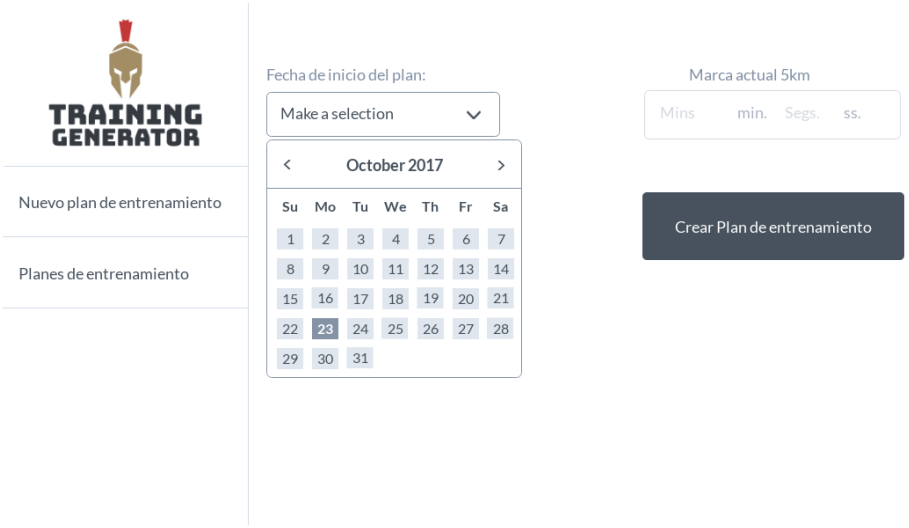


Figura 6.2: Pantalla prototipo creación plan de entrenamiento

Cuando se realiza click en la segunda opción del menú lateral, se accederá a la pantalla de *Planes de entrenamiento* (ver figura 6.3). Esta pantalla mostrará un listado con los planes de entrenamientos creados por el usuario. La tabla lleva la fechas de inicio y fin de plan, el tipo del plan seleccionado y la marca de los 5km.

En la parte superior derecha de la pantalla se encuentra el botón para actualizar datos desde *Strava*, que permitirá al usuario sincronizar al usuario los planes de entrenamiento creados con los entrenamientos guardados en *Strava*.

Desde esta pantalla se puede acceder al detalle de cada plan de entrenamiento realizando doble click sobre la fila deseada.

Figura 6.3: Pantalla prototipo listado planes de entrenamiento

La pantalla de detalle de un plan de entrenamiento (ver figura 6.4) contendrá la información semanal del plan y su estado actual, además de la información general sobre el plan de entrenamiento.

En la parte superior se encuentran los mismos datos que se encontraban en la tabla de planes de entrenamiento: Fechas de inicio y fin del plan, el tipo del plan seleccionado y la marca de los 5km.

En la parte inferior se pueden encontrar varias pestañas con las diferentes semanas del plan de entrenamiento y una tabla con los detalles de la pestaña seleccionada. Las pestañas se compondrán de 12 o 16 semanas. Cada vez que se haga click en una pestaña se recargará la rejilla inferior con los datos sobre la semana seleccionada.

La rejilla llevará información de los diferentes entrenamientos durante los días de la semana seleccionada. Tendrá las siguientes columnas:

- **Día de entrenamiento**: Día de la semana con nombre.

- **Entrenamiento**: El tipo de entrenamiento de ese día, puede ser Carrera corta, carrera larga, series, entrenamiento cruzado o descanso.
- **Sensaciones**: Las sensaciones que introduzca el usuario aparecerán aquí.
- **Realizado**: Indicará si el entrenamiento ha sido realizado.
- **Ver Resultados**: En esta columna existirá un botón cuando el entrenamiento sea uno de los principales y llevará a la pantalla de detalle del día de entrenamiento.

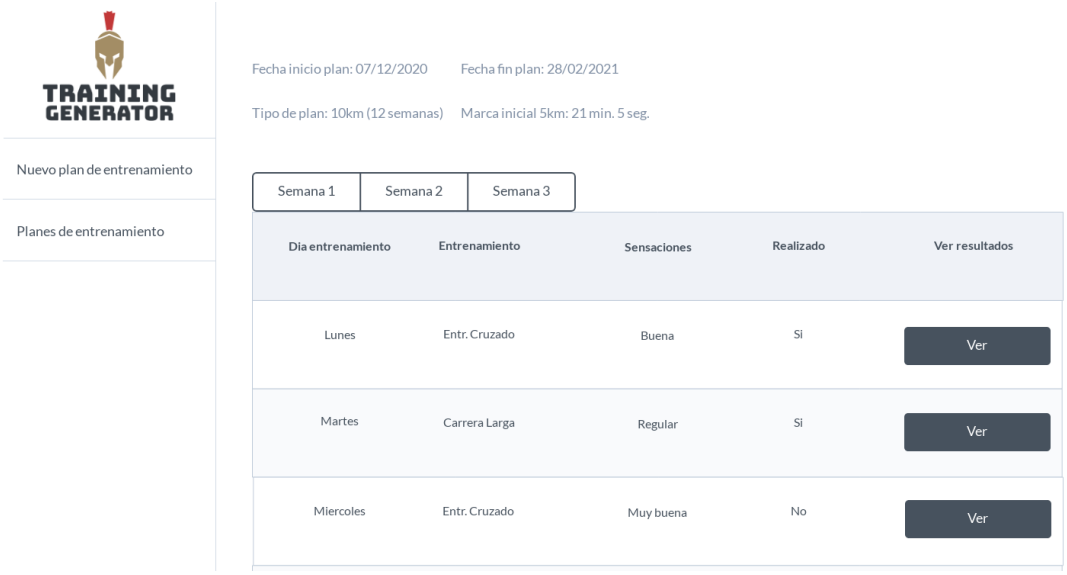


Figura 6.4: Pantalla prototipo detalle de plan de entrenamiento

La última pantalla definida es la de detalle del día de entrenamiento (ver figura 6.5). Esta pantalla contendrá más o menos información dependiendo de si el día del entrenamiento ha sido sincronizado con *Strava* o no. En la parte superior tendrá el tipo de entrenamiento y la fecha del mismo, además de una descripción del entrenamiento a realizar.

Tras eso aparecerá una rejilla donde se informarán los resultados obtenidos tras la sincronización con *Strava*. La información a mostrar será el número de la serie, el tiempo objetivo, el tiempo realizado y la diferencia entre los dos tiempos.

En la parte inferior de la pantalla se mostrará un resumen de los datos recogidos de *Strava*. Estos datos son la frecuencia media, la frecuencia máxima, los kilómetros realizados y dos combos que permitirán seleccionar si se ha realizado o no, y las sensaciones obtenidas tras el entrenamiento.

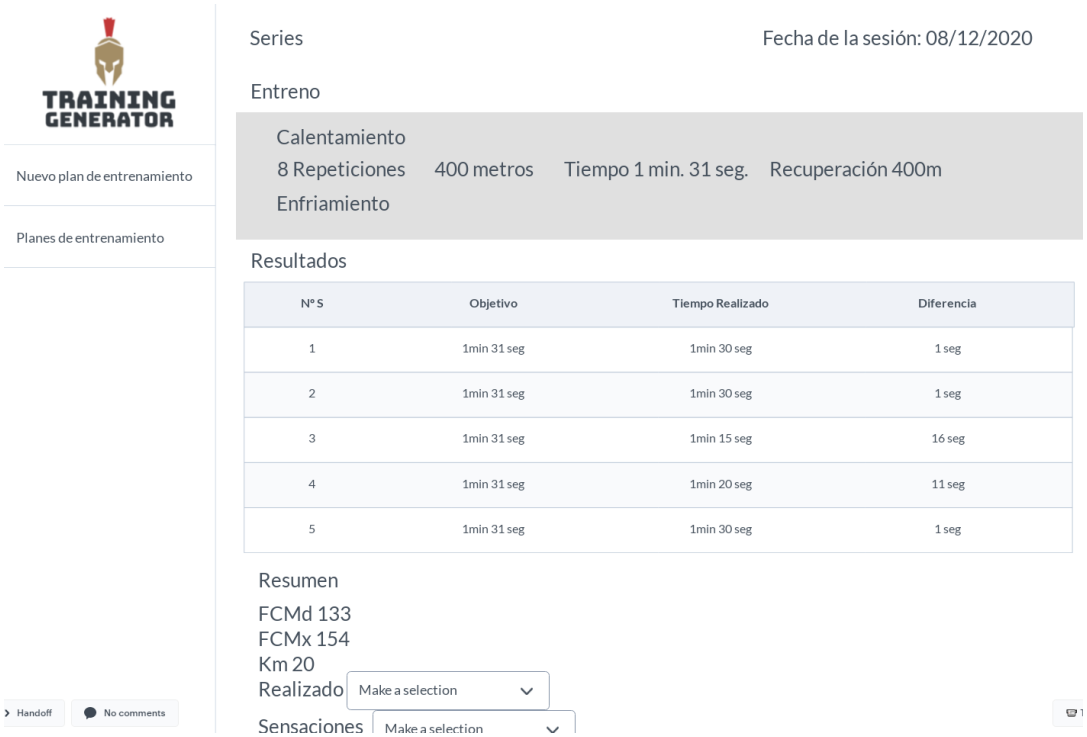


Figura 6.5: Pantalla prototipo detalle de día de entrenamiento

Además de todo esto, se ha decidido el logo que se utilizaría para identificar a la aplicación, y que se encontrará durante todas las páginas de la misma. Este logo se ha generado utilizando la pagina web logo generator [15] y se ha decidido que el logo sea un casco de espartano.

6.2. Maquetación de la aplicación

Durante el segundo Sprint se llevó a cabo la creación de la aplicación en .NET Core y la maquetación de las pantallas utilizando HTML, Less y Bootstrap.

Para la creación de la aplicación se utilizó una plantilla de Visual Studio, la cual prepara la arquitectura básica de carpetas y de librerías de clases.

Tras esto se pasó a crear los controladores y las acciones principales de las pantallas definidas en el prototipo. Se han definido tres controladores principales a partir de los cuales se generarán las distintas acciones:

- **LoginController**: En este controlador estarán todos los métodos relacionados con el login de la aplicación y la redirección a *Strava*.

- **TrainingsController**: En este controlador se encontrarán las acciones relativas al listado de planes de entrenamiento y los detalles de un plan de entrenamiento y un día de entrenamiento.

- **CreateTrainingController**: Desde este controlador se llegará a los métodos relativos a la creación de un plan de entrenamiento.

Con estos tres controladores se ha procedido a crear los métodos relativos a cada vista, y la devolución de las vistas en los mismos para poder realizar la maquetación.

Las vistas que se han maquetado utilizando HTML y Less, permitiendo de esta manera adaptar los diseños al prototipo y hacerlos mas llamativos para el usuario. Se pueden identificar tres pantallas principales a maquetar, ya que el resto de pantallas son sencillas:

6.2.1. Crear plan de entrenamiento

La pantalla para crear un plan de entrenamiento ha llevado varios combos que se han maquetado utilizando selects de HTML y añadiéndoles etiquetas encima con Bootstrap. Esta pantalla ha ido modificándose poco a poco según iba avanzando el desarrollo de la aplicación. Al finalizar el segundo sprint ha sido una pantalla sencilla con varios campos colocados siguiendo un formulario sencillo de dos columnas (ver figura 6.6).

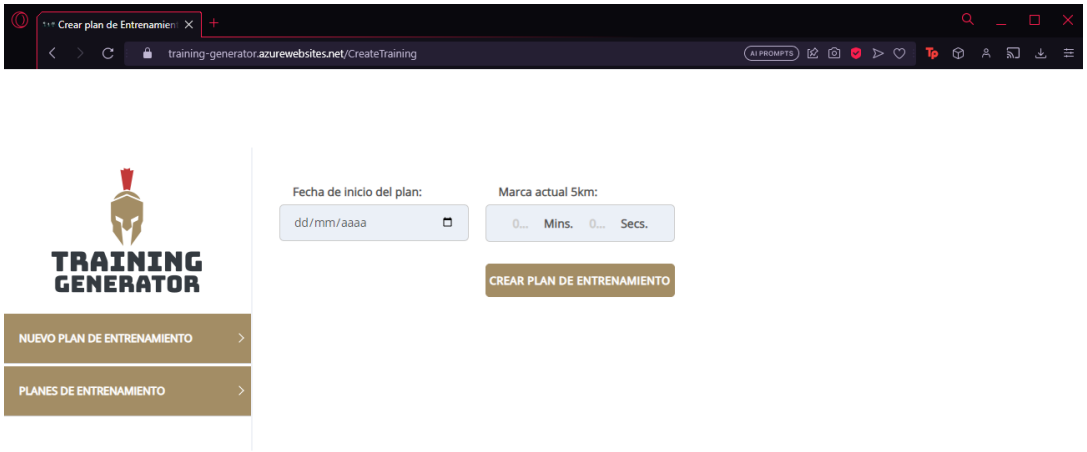


Figura 6.6: Pantalla creación plan de entrenamiento

Durante los diferentes Sprints la pantalla ha sufrido modificaciones para adaptarse a las peticiones del cliente. Estas modificaciones han conllevado el añadir varios campos más y modificar la estructura principal de la pantalla.

6.2.2. Detalle de plan de entrenamiento

Para el detalle de un plan de entrenamiento ha habido que maquetar varios elementos diferentes.

Primero se han maquetado los campos superiores que han sido sencillos, ya que Bootstrap te proporciona campos sencillos y fácilmente utilizables. Tras esto se ha tenido que maquetar las diferentes pestañas de las distintas semanas que existirán en el plan de entrenamiento. Para estas pestañas se ha tenido que tener en cuenta que puede haber 12 o 16 semanas, por lo que se ha probado con ambas y se ha preparado para que se viese correctamente en ambos casos.

Por ultimo se ha maquetado la rejilla. Para maquetar la rejilla se ha utilizado el componente JsGrid [12] que permite la utilización de rejillas de una manera muy sencilla y directamente desde Javascript, lo que reduce la carga de trabajo en servidor a la hora de procesar las vistas. Se han creado todas las columnas necesarias y se ha preparado para que siguiesen los colores definidos para la

aplicación. Mostrar imagen de detalle de plan de entrenamiento.

6.2.3. Detalle de día de entrenamiento

El detalle del día de entrenamiento ha sido parecido al detalle de un plan de entrenamiento con algunas modificaciones (ver figura 6.7).

Se ha creado una zona resaltada donde dentro se han añadido los campos de información sobre los distintos ejercicios a realizar. Para esto se ha definido un elemento div principal en HTML que se ha preparado para contener datos dinámicos y crecer sin dejar de verse correctamente.

Dentro de este elemento se ha preparado para que se puedan añadir programáticamente varias lineas para representar los ejercicios a realizar.

Por último se ha definido una rejilla con la información que se recogerá en futuros Sprints para mostrar esos datos recogidos. Esta primera maquetación de la pantalla se puede observar en la imagen.

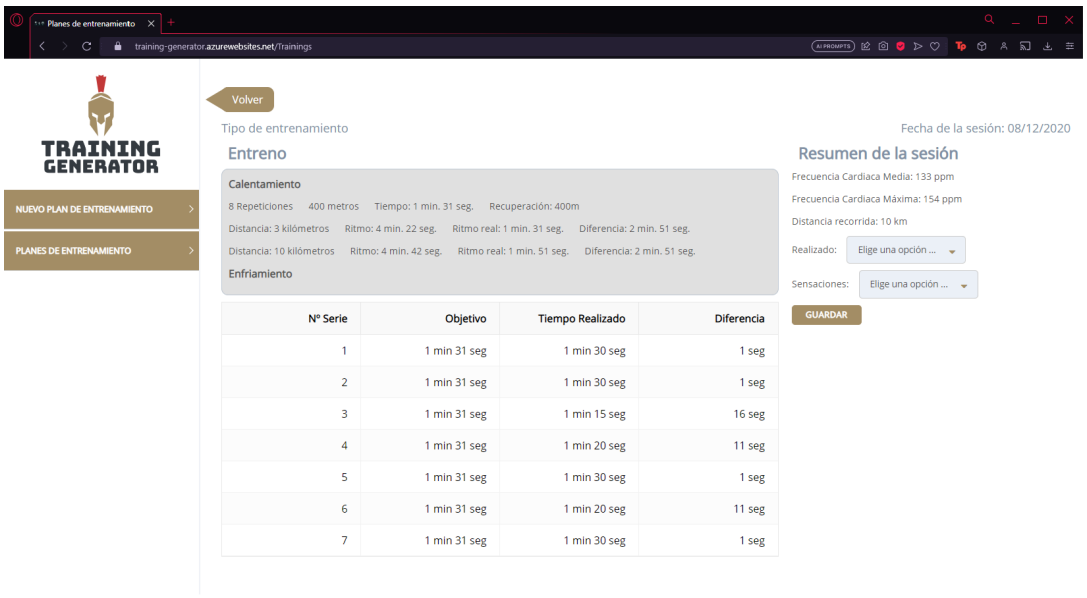


Figura 6.7: Pantalla detalle de día de entrenamiento

Esta pantalla más adelante se modificaría para adaptarse a nuevas peticiones por parte del cliente. Toda la información superior se añadiría a la tabla,

haciendo que esta ocupe toda la pantalla. Además se modificaría la tabla para qué algunas filas estén mas oscuras y algunos campos cambien de color, todo por programación en Javascript.

6.3. Definición de la base de datos

El objetivo principal del Sprint 3 ha sido preparar la base de datos para que almacene los datos necesarios para que la aplicación funcione, siguiendo el análisis detallado del comienzo del proyecto.

Para modelar la base de datos se ha utilizado la herramienta Microsoft SQL Server Management Studio [16], que permite realizar modelados de base de datos de una manera más visual, y dejar definidos esquemas que posteriormente se pueden utilizar como documentación.

La nomenclatura de todas las tablas y procedimientos de la base de datos sigue la misma regla. Todos los elementos empezarán por *STRA_*, indicando que pertenecen al esquema principal. Tras esto la regla cambia dependiendo del tipo de elemento que sea:

- Para las tablas directamente se pondrá el nombre de la tabla, exceptuando las tablas catálogo que llevarán un *C_* intermedio, indicando que son de tipo catálogo y que no se deberían incluir ningún valor por código.
- Para los procedimientos almacenados se pondrá el nombre de la tabla principal con la que van a trabajar, y se añadirá un sufijo indicando el tipo de acción que se realiza en él. Por ejemplo *DAY_TRAINING_SelWithResults*.
- Para las funciones se añadirá entre el nombre de la función y el prefijo *STRA_* un *fn_*, indicando que es una función.

Las tablas catálogo son un tipo de tabla especial que estarán rellenas desde un principio con valores fijos que no se modificarán, y sirven para que los valores de ciertos campos de otras tablas no puedan tomar otros valores que no estén en estas tablas.

Todas estas tablas están definidas de la misma manera, tienen un código Integer, que identifica la clave primaria de la tabla y con la que se relaciona con otras tablas; y un valor NVarchar, que suele ser la descripción del valor.

Al separar la clave del valor descriptivo nos permite tener estos valores relacionados con varias tablas y poder modificar la descripción de todas las tablas relacionadas con solo cambiar un valor.

En la base de datos se pueden identificar dos bloques de tablas principalmente. El primer bloque contendría todas aquellas tablas donde están almacenados los distintos planes de entrenamiento para la creación de los mismos.

Estas tablas son *STRA_C_RUN*, *STRA_C_SERIES_INFO* y *STRA_TIMES* (ver figura 6.8). La primera tabla contiene las distintas carreras cortas y largas que se tienen que realizar cada día agrupadas por código de grupo. La segunda tabla contiene las distintas series que se tienen que realizar cada día de entrenamiento agrupadas por código de grupo. Durante la creación de los planes de entrenamiento se obtienen las distintas carreras y series a partir de estas tablas, y se guardan dentro de la tabla *STRA_DAY_TRAINING* que se verá más adelante.

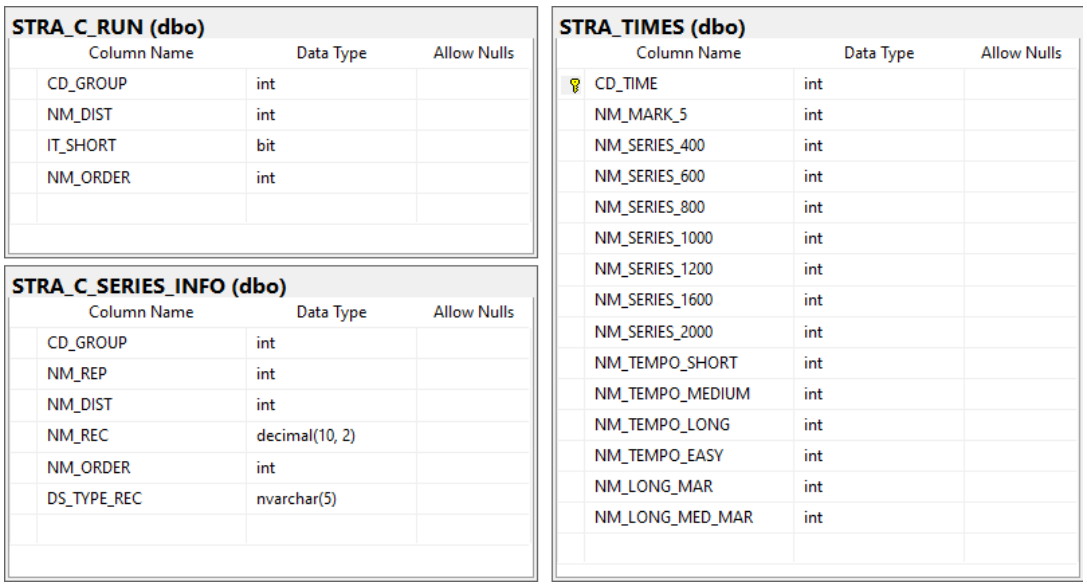


Figura 6.8: Tablas con los planes de entrenamiento definidos

En la tercera tabla *STRA_TIMES* se pueden encontrar los tiempos objetivo

para cada tipo de carrera dependiendo del tiempo inicial que incluya el usuario en la creación del plan de entrenamiento. Esta tabla permite delegar la decisión de los tiempos y el poder modificarlos en cualquier momento de una manera rápida y efectiva.

El segundo bloque principal contiene todas las tablas donde se guardarían los planes de entrenamiento con todos los días de entrenamiento de cada usuario. Estas tablas son *STRA_TRAINING*, *STRA_DAY_TRAINING* y *STRA_RESULTS_DAY* (ver figura 6.9).

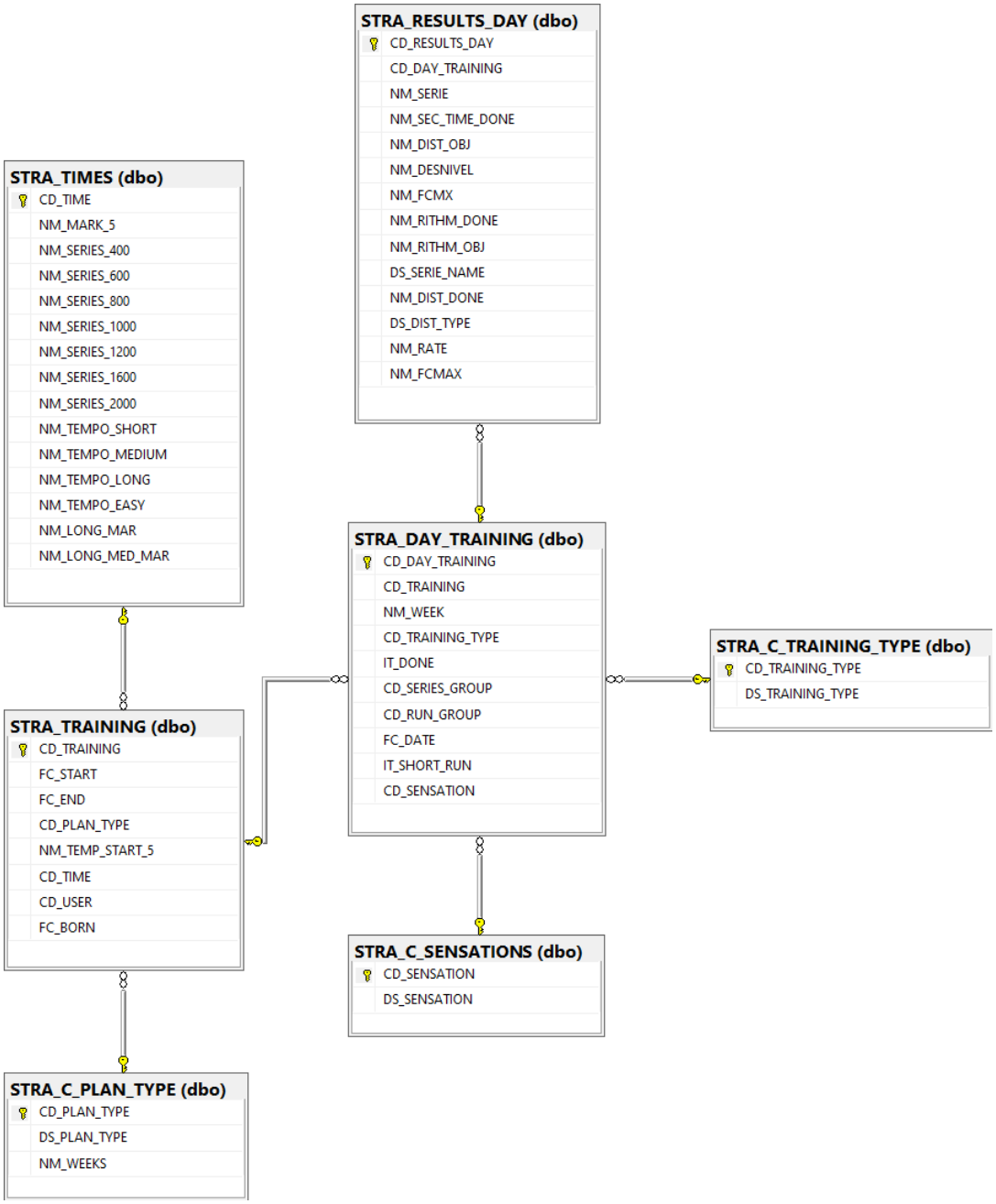


Figura 6.9: Tablas con los planes de entrenamiento de los usuarios

En la tabla *STRA_TRAINING* se guardan los planes de entrenamiento junto con los datos introducidos por el usuario y una relación con el tipo de entrenamiento seleccionado y otra relación con la tabla de tiempos a partir del valor introducido por el usuario.

En la tabla *STRA_DAY_TRAINING* se almacenan la organización de los

días y la relación con las tablas catálogo de carreras y series a partir del código del grupo. Esta tabla lleva los datos necesarios para que durante la sincronización con *Strava* el sistema sea capaz de unir los datos.

Por último, en la tabla *STRA_RESULTS_DAY* se almacenan todos los datos obtenidos para cada serie de cada día del plan de entrenamiento. A partir de esta tabla se muestran los datos exactos y las comparaciones con los tiempos objetivos.

6.4. Programación de la aplicación

Durante el Sprint 4 se ha llevado a cabo la programación de la aplicación, junto con algunas modificaciones a la maquetación debido al feedback obtenido del cliente y necesidades que se iban obteniendo según se iba programando la aplicación.

Para crear las conexiones con la base de datos se ha utilizado la librería *EnterpriseLibrary* [17]. Esta librería encapsula el acceso a base de datos y la ejecución de procedimientos almacenados haciendo estas conexiones mucho más seguras que si se realizasen consultas directamente contra la base de datos, y facilitando el código para el desarrollador.

Una de los problemas que evita esta librería es el *SQLInjection*, problema bastante extendido y grave entre el desarrollo de aplicaciones Web. Este problema radica en la ejecución de código SQL a través de la introducción de caracteres no esperados en los campos de la aplicación, como por ejemplo en un campo nombre de usuario o un parámetro id en la query de acceso al detalle de un plan de entrenamiento.

Esta librería transforma todos esos caracteres especiales para que la base de datos los entienda como caracteres normales dentro de un texto, evitando de esta manera este problema de seguridad.

Para las conexiones a la base de datos se han creado varios Managers de base de datos en la capa *DatabaseAccessLayer*, a través de los cuales se accede

a la base de datos. El código siempre sigue una estructura parecida, preparando primero el procedimiento almacenado que se va a ejecutar, recogiendo los parámetros de entrada y lanzando la consulta a base de datos (ver ejemplo en figura 6.10). Tras esto se recogen los resultados y se rellenan objetos similares a los que están en base de datos y que se devuelven desde el procedimiento y se devuelven.

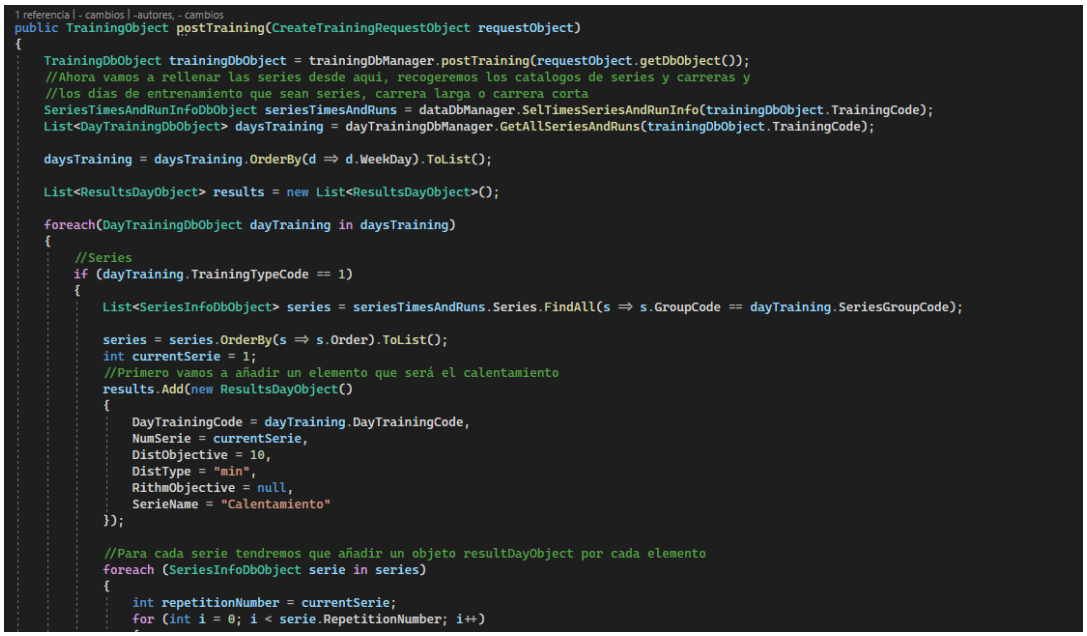
```
1 referencia | 0 cambios | 0 autores, 0 cambios
public List<DayTrainingDbObject> GetByTrainingWeek(int trainingId, int week, long userId)
{
    try
    {
        Database db = GetDatabase();
        using (DbCommand dbCommand = db.GetStoredProcCommand("STRA_DAY_TRAINING_SelByTrainingAndWeek"))
        {
            db.AddInParameter(dbCommand, "@CD_TRAINING", DbType.Int32, trainingId);
            db.AddInParameter(dbCommand, "@NM_WEEK", DbType.Int32, week);
            db.AddInParameter(dbCommand, "@CD_USER", DbType.Int64, userId);

            using (DataSet ds = db.ExecuteDataSet(dbCommand))
            {
                List<DayTrainingDbObject> result = new List<DayTrainingDbObject>();

                if (ds != null && ds.Tables.Count > 0 && ds.Tables[0].Rows.Count > 0)
                {
                    foreach (DataRow row in ds.Tables[0].Rows)
                    {
                        result.Add(new DayTrainingDbObject(row));
                    }
                }

                return result;
            }
        }
    }
    catch (Exception e)
    {
        throw new DatabaseException(e);
    }
}
```

Figura 6.10: Código de ejecución de procedimiento almacenado

Estos objetos obtenidos desde base de datos pasan a través de la capa *BusinessLogicLayer*, la cual es la encargada de toda la lógica de negocio de la aplicación, como por ejemplo la creación de un entrenamiento. Esta capa tiene la capacidad de acceder las veces necesarias a la capa *DatabaseAccessLayer* para obtener o guardar los datos con los que trabaja.

Durante la creación de un entrenamiento, accede primero a base de datos para crear un entrenamiento, que inicializará el entrenamiento en la tabla principal. A partir de esta acción recoge las series y las carreras largas y cortas para ese tipo de entrenamiento, y procesa para cada día de entrenamiento las distintas series, calentamientos y enfriamientos que tocan, y las vuelve a guardar en

base de datos de una forma adecuada (ver figura 6.11).



```
1 referencia | - cambios | -autores, - cambios
public TrainingObject postTraining(CreateTrainingRequestObject requestObject)
{
    TrainingDbObject trainingDbObject = trainingDbManager.postTraining(requestObject.getDbObject());
    //Ahora vamos a rellenar las series desde aqui, recogeremos los catalogos de series y carreras y
    //los dias de entrenamiento que sean series, carrera larga o carrera corta
    SeriesTimesAndRunInfoDbObject seriesTimesAndRuns = dataDbManager.SelTimesSeriesAndRunInfo(trainingDbObject.TrainingCode);
    List<DayTrainingDbObject> daysTraining = dayTrainingDbManager.GetAllSeriesAndRuns(trainingDbObject.TrainingCode);

    daysTraining = daysTraining.OrderBy(d => d.WeekDay).ToList();

    List<ResultsDayObject> results = new List<ResultsDayObject>();

    foreach(DayTrainingDbObject dayTraining in daysTraining)
    {
        //Series
        if (dayTraining.TrainingTypeCode == 1)
        {
            List<SeriesInfoDbObject> series = seriesTimesAndRuns.Series.FindAll(s => s.GroupCode == dayTraining.SeriesGroupCode);

            series = series.OrderBy(s => s.Order).ToList();
            int currentSerie = 1;
            //Primero vamos a añadir un elemento que será el calentamiento
            results.Add(new ResultsDayObject()
            {
                DayTrainingCode = dayTraining.DayTrainingCode,
                NumSerie = currentSerie,
                DistObjective = 10,
                DistType = "min",
                RithmObjective = null,
                SerieName = "Calentamiento"
            });

            //Para cada serie tendremos que añadir un objeto resultDayObject por cada elemento
            foreach (SeriesInfoDbObject serie in series)
            {
                int repetitionNumber = currentSerie;
                for (int i = 0; i < serie.RepetitionNumber; i++)
```

Figura 6.11: Código de creación de un plan de entrenamiento

Por último se ha conectado las vistas con los controladores, y estos mismos con los Managers de la capa *BusinessLogicLayer*. Dentro de los controladores se han recogido los distintos datos necesarios para cada vista, se ha generado un modelo que contiene todos estos datos y se han pasado a las vistas para que estas rellenen los datos necesarios.

La funcionalidad principal radica en las distintas rejillas que existen en la aplicación, que están programadas con JsGrid. Para definir las columnas de cada rejilla se define desde Javascript cuales van a ser y de que propiedad del objeto relacionado van a coger el valor. A las rejillas definidas con JsGrid se les pueden añadir paginación y ordenación con tan solo poner una variable a verdadero al definirlas.

Hay dos formas distintas en las que se cargan los datos en las rejillas de la aplicación.

La primera forma está en casi todas las rejillas de la aplicación. Cuando se devuelve la vista se devuelve en la misma una variable para Javascript con los datos relativos a la rejilla serializados en formato json, de tal manera que

Javascript los interpreta y los puede incluir en la rejilla. Esta manera es la más sencilla, ya que en el controlador se recogen los datos y se le pasan a la vista a través del modelo para que ésta los represente en la variable Javascript.

La segunda forma es más complicada, ya que conlleva realizar llamadas asíncronas ajax que recogerán los datos del controlador, haciendo que este actúe como un API Rest. En el momento de definir la rejilla se define una variable controller con un parámetro loadData, el cual es una función que será la que se ejecutará en el momento de cargar los datos y utilizará los que se obtengan en esta misma llamada. En esta llamada se definirá una promesa de Javascript y se lanzará una llamada Ajax para recoger los datos. Tras obtener los datos estos se procesan para, por ejemplo, mostrar correctamente la fecha; y se devuelven utilizando la promesa definida en un principio, por lo que JsGrid interpreta los datos y los muestra. Se puede observar un resumen del código descrito en la figura 6.12.

```
sorting: true,
paging: true,
autoload: true,

//rowClass: editRowClass,

controller: {
    loadData: function () {
        var d = $.Deferred();

        $.ajax({
            type: "GET",
            url: urlGetDayTrainings.replace("-1", currentWeek),
            dataType: "json"
        }).done(function (response) {
            console.log(response);
            if (response && !response.errorMsg) {
                for (let i = 0; i < response.length; i++) {
                    let itemResponse = response[i];
                    let EndDate = new Date(itemResponse.Date)
                    itemResponse.DateStr = `${EndDate.getDate()}/${EndDate.getMonth() + 1}/${EndDate.getFullYear()}`;
                    response[i] = itemResponse;

                    //Cogemos el dia mas bajo, que será el primero de la semana
                    let lowestDateObj = Math.min.apply(null, response.map(a => new Date(a.Date)));
                    let lowestDate = new Date(lowestDateObj);
                    $("#dateWeekStart").html(`${lowestDate.getDate()}/${lowestDate.getMonth() + 1}/${lowestDate.getFullYear()}`);
                }
                d.resolve(response);
            } else
                d.reject(response.errorMsg);
        }).fail(function () {
            d.reject();
        });

        return d.promise();
    }
},
```

Figura 6.12: Carga asíncrona de una rejilla de JsGrid

6.5. Integración con *Strava*

Para el quinto y último Sprint se ha realizado la integración con el sistema *Strava*. Para realizar las distintas integraciones se requieren unos pasos iniciales en *Strava*.

Lo primero de todo hay que crearse una cuenta que funcionará como cuenta de desarrollo en *Strava*. Dentro de esta cuenta hay que entrar en la sección de APIs de *Strava* y darle a crear una. Se rellenan todos los datos que se piden y se añade un icono de aplicación. Esto es requerido ya que en la integración con *Strava* a través de OAuth 2.0 se informará al usuario de la aplicación que está intentando acceder a sus datos y el objetivo de la misma.

6.5.1. Login OAuth 2.0

Tras la creación de la cuenta y de la api en *Strava* se ha comenzado con la integración del login a través del protocolo Oauth 2.0.

En la clase *Startup* de la aplicación se ha utilizado la autenticación integrada de .NET definiendo los distintos parámetros necesarios para la redirección a *Strava* y definiendo un nuevo handler *StravaOAuthHandler* para que se utilice cuando vuelva de la web de *Strava* (ver figura 6.13).

Esta clase *StravaOAuthHandler* (ver figura 6.14) se encargará de federar al usuario, obtener su token de acceso y guardar en una cookie y en la sesión del usuario los datos obtenidos para mantener su login. Para que .NET detecte que el usuario está identificado hay que crear una identidad con sus parámetros, y utilizar la función SignInAsync del contexto principal de la aplicación para que esta guarde al usuario. Esta clase también recargará los datos en la sesión si encuentra una Cookie con un access token de usuario, preguntando a *Strava* a quien pertenece.

En caso de que el token de acceso obtenido de la Cookie no sea válido o haya caducado, será esta clase la que se encargará de limpiar de la sesión y las Cookies todos los datos del usuario y redirigirle al login para que vuelva a

```
services.AddAuthentication(options =>
{
    //Indicamos que vamos a utilizar cookies para autenticar y logar
    options.DefaultAuthenticateScheme = CookieAuthenticationDefaults.AuthenticationScheme;
    options.DefaultSignInScheme = CookieAuthenticationDefaults.AuthenticationScheme;

    //Clave de configuración de flujo OAUTH
    options.DefaultChallengeScheme = "StravaSettings";
})
    .AddCookie()
    .AddOAuth<StravaOAuthOptions, StravaOAuthHandler>("StravaSettings", options =>
    {

        options.StravaSettings = stravaSettings;
        options.ClientId = stravaSettings.client_id;
        options.ClientSecret = stravaSettings.client_secret;

        //Ruta a la que llamara el proveedor de identidad tras autenticar al usuario
        options.CallbackPath = new PathString(stravaSettings.redirect_uri);

        options.AuthorizationEndpoint = $"{stravaSettings.authorize_url}";
        options.TokenEndpoint = $"{stravaSettings.strava_url}";

        options.Scope.Add(stravaSettings.scopes);

        //Endpoint del proveedor de identidad del que obtener información del usuario autenticado
        //options.UserInformationEndpoint = $"{sipSettings.URL}{sipSettings.UserInformationEndpoint}";
        options.ClaimActions.Clear();
        /*
            //Match de los atributos del objeto de usuario para la respuesta del UserInformationEndpoint
            foreach (var elem in sipSettings.ClaimActions)
            {
                options.ClaimActions.MapJsonKey(elem.Key, elem.Value);
            }
        */
        options.SaveTokens = true;
        options.Events = new OAuthEvents
```

Figura 6.13: Activación de autenticación en la clase Startup

acceder.

```
bool ajaxRequest = Request.Headers.ContainsKey("X-Requested-With") && Request.Headers["X-Requested-With"] == "XMLHttpRequest";
string authorization_code = Request.Path.Value.ToLower().StartsWith("/login/authorized") && Request.Query.ContainsKey("code") ? Request.Query["code"]
if (authorization_code != null)
{
    //Con la autorización, federamos al usuario
    StravaAccessToken response = new OAuthRestManager(Options.StravaSettings.strava_url).GetToken(authorization_code, Options.StravaSettings.client_
    if (response != null)
    {
        CookieOptions cookieOptions = new CookieOptions();
        //Asignamos un tiempo de expiración a la cookie
        cookieOptions.Expires = DateTimeOffset.Now.AddSeconds(response.expires_in);
        //Le decimos que la cookie es esencial
        cookieOptions.IsEssential = true;
        cookieOptions.HttpOnly = true;
        Response.Cookies.Append("t", response.access_token, cookieOptions);
        return await Task.FromResult(false); //Comprobación hecha, no hace falta seguir
    }
}
else if (Request.Path.Value.ToLower().StartsWith("/login") || Request.Path.Value.ToLower().StartsWith("/errores") || Request.Path.Value.Equals("/")
{
    return await Task.FromResult(false);
}
else if (Request.Cookies.ContainsKey("t"))
{
    //Recuperar cookie y recuperar identidad de usuario
    authorization_code = Request.Cookies["t"];

    //Options.UserInformationEndpoint Se debería utilizar esto, pero como ya existe en el SDK una llamada ... pues la usamos.
    Athlete response = Request.HttpContext.Session.HasValue(SessionKeys.UserKey) ? Request.HttpContext.Session.Get<Athlete>(SessionKeys.UserKey) : ne
    if (response != null)
    {
        //Asignar al Identity
        ClaimsIdentity identity = new System.Security.Claims.ClaimsIdentity(new List<Claim>()
        {
            new Claim("name", response.firstname),
            new Claim("idSTRAVA", response.id.ToString())
        }, "Custom");
        ClaimsPrincipal principal = new ClaimsPrincipal(identity);
        await Context.SignInAsync(CookieAuthenticationDefaults.AuthenticationScheme, principal, new AuthenticationProperties()
        {
            AllowRefresh = true,
```

Figura 6.14: Clase StravaOAuthHandler, con verificación de token y comprobación de cookie

6.5.2. Recogida y unión de datos de *Strava*

La segunda integración con *Strava* que se ha realizado ha sido realizada en la clase *SyncManager* de la capa *BusinessLogicLayer* (ver figura 6.15). Esta integración se encarga de recoger las últimas actividades del usuario identificado de *Strava* y unirlas con los datos que existen previamente en la base de datos. Para comenzar el proceso se recoge desde base de datos la fecha de la última sincronización contra *Strava* de este usuario. Esto permitirá que no se vuelvan a recoger actividades pasadas a la última que se ha recogido. Tras esto se hace una llamada al API Rest de *Strava* para recoger un máximo de 20 actividades del usuario a partir de la última fecha de sincronización. Con estas actividades se procede a realizar una llamada para cada actividad que recogerá las vueltas de cada actividad. Cada vuelta corresponde con una serie, enfriamiento o calentamiento de los datos que se tienen en la base de datos de la aplicación.

```
//Vamos a leer solamente 20 actividades cada vez que se sincronice
List<StravaActivity> activities = activitiesManager.GetUserActivities(token, before, after, 1, 20);

if(activities.Count > 0)
{
    List<LapResultDbObject> lapsToSave = new List<LapResultDbObject>();
    //TODO: Recoger un listado de sensaciones a partir de las frecuencia cardiaca máxima y media para cada día de entrenamiento y guardarlo en bbdd
    DateTimeOffset? lastDate = null;
    foreach (StravaActivity activity in activities)
    {
        //Para cada actividad sacaremos las laps
        List<ActivityLaps> laps = activitiesManager.GetActivityLaps(token, activity.id);

        //Ahora vamos a guardar las laps en la lista que irá a base de datos para actualizar valores
        foreach (ActivityLaps lap in laps)
        {
            lapsToSave.Add(new LapResultDbObject()
            {
                NumSerie = lap.split,
                Date = lap.start_date,
                TimeDone = lap.moving_time,
                Desnivel = (int)lap.total_elevation_gain,
                AverageFrecuency = (int)lap.average_heartrate,
                MaxFrecuency = (int)lap.max_heartrate,
                RithmDone = (int)(lap.moving_time * 1000 / lap.distance),
                DistanceDone = (int)lap.distance,
                RateDone = lap.average_cadence
```

Figura 6.15: Código de sincronización con *Strava* en SyncManager

Tras procesar todas las vueltas de todas las actividades y guardarlas en un listado de vueltas se mandan a base de datos para hacer las comparaciones y los guardados de los datos en la tabla de *STRA_RESULTS_DAY*.

El guardado de las vueltas se realiza en el procedimiento almacenado *STRA_pr_RESULTS_DAY_InsLaps*. Este procedimiento actualiza la tabla indicada anteriormente a partir de la fecha de realización de la actividad y el número de la serie, que corresponde al número de la vuelta registrada en *Strava*.

Con estos datos almacena el tiempo realizado, el desnivel, la frecuencia máxima, el ritmo y distancias realizadas y las pulsaciones obtenidas en el entrenamiento y guardados en *Strava*. Tras esto actualiza todos los días que se han actualizado en la tabla *STRA_DAY_TRAINING* marcándolos como completados y ejecutando la función de cálculo de sensaciones (ver figura 6.16).

```
UPDATE RD
    SET NM_SEC_TIME_DONE = R.NM_SEC_TIME_DONE
        , NM_DESNIVEL = R.NM_DESNIVEL
        , NM_FCMX = R.NM_FCMX
        , NM_RITHM_DONE = R.NM_RITHM_DONE
        , NM_DIST_DONE = R.NM_DIST_DONE
        , NM_RATE = R.NM_RATE
    OUTPUT
        INSERTED.CD_RESULTS_DAY
        , INSERTED.CD_DAY_TRAINING
        , INSERTED.NM_SERIE
        , INSERTED.NM_SEC_TIME_DONE
        , INSERTED.NM_DIST_OBJ
        , INSERTED.NM_DESNIVEL
        , INSERTED.NM_FCMX
        , INSERTED.NM_FCMAX
        , INSERTED.NM_RITHM_DONE
        , INSERTED.NM_RITHM_OBJ
        , INSERTED.DS_SERIE_NAME
        , INSERTED.NM_DIST_DONE
        , INSERTED.DS_DIST_TYPE
        , INSERTED.NM_RATE
    FROM STRA_RESULTS_DAY RD
        INNER JOIN STRA_DAY_TRAINING D
            ON RD.CD_DAY_TRAINING = D.CD_DAY_TRAINING
        INNER JOIN @RESULTS R
            ON R.FC_DATE = D.FC_DATE
            AND RD.NM_SERIE = R.NM_SERIE

UPDATE D
    SET IT_DONE = 1
    , CD_SENSATION = dbo.STRA_fn_CalcSensation(D.CD_DAY_TRAINING)
FROM STRA_DAY_TRAINING D
    INNER JOIN @RESULTS R
        ON R.FC_DATE = D.FC_DATE
```

Figura 6.16: Actualización de valores recogidos de *Strava* en base de datos

Tras esto añade o actualiza en la tabla de sincronizaciones la última sincronización del usuario a fecha de la última actividad recogida.

7 Resultados

Para probar toda la aplicación el usuario se sitúa en 2016 y quiere comenzar a preparar una carrera de 10 kilómetros en 12 semanas. Para esto entra en la página web de training generator y se dirige a la creación de un plan de entrenamiento.

El usuario decide comenzar el día 19 de diciembre de 2016, Lunes. Añade cual es su marca actual de 5 kilómetros, que es 23 minutos y 50 segundos, y elige el plan de entrenamiento de 10 kilómetros en 12 semanas. También ha añadido su fecha de nacimiento en el campo dedicado para ello.

Tras esto el usuario decide cual va a ser su distribución semanal de entrenamientos. Cumpliendo con las restricciones indicadas el usuario termina de rellenar la página, siguiendo la figura 7.1.

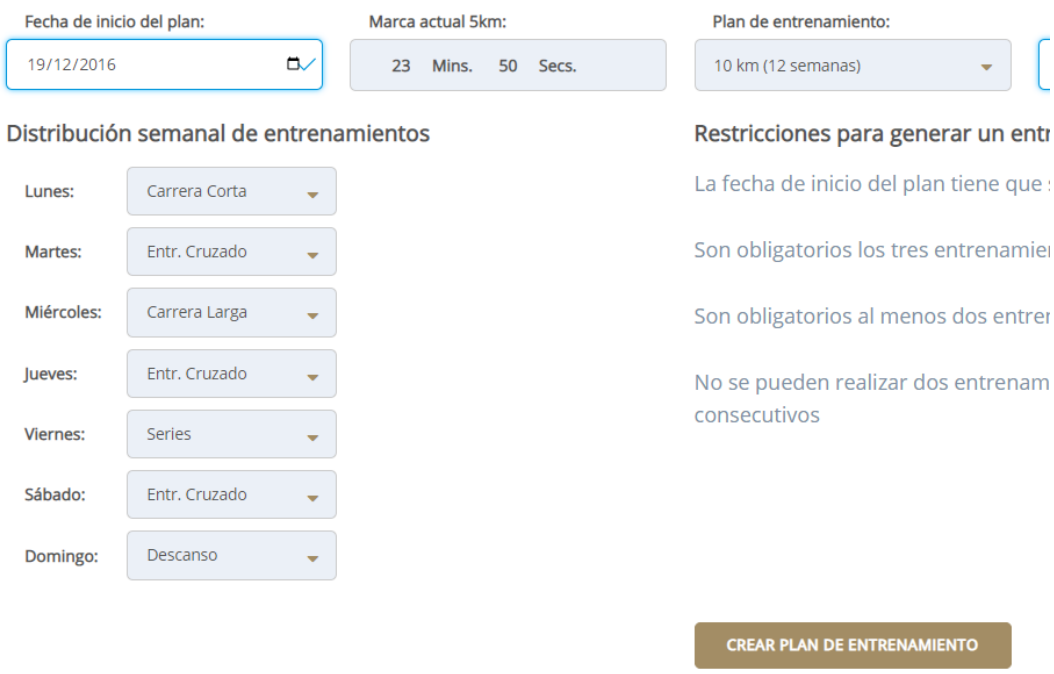
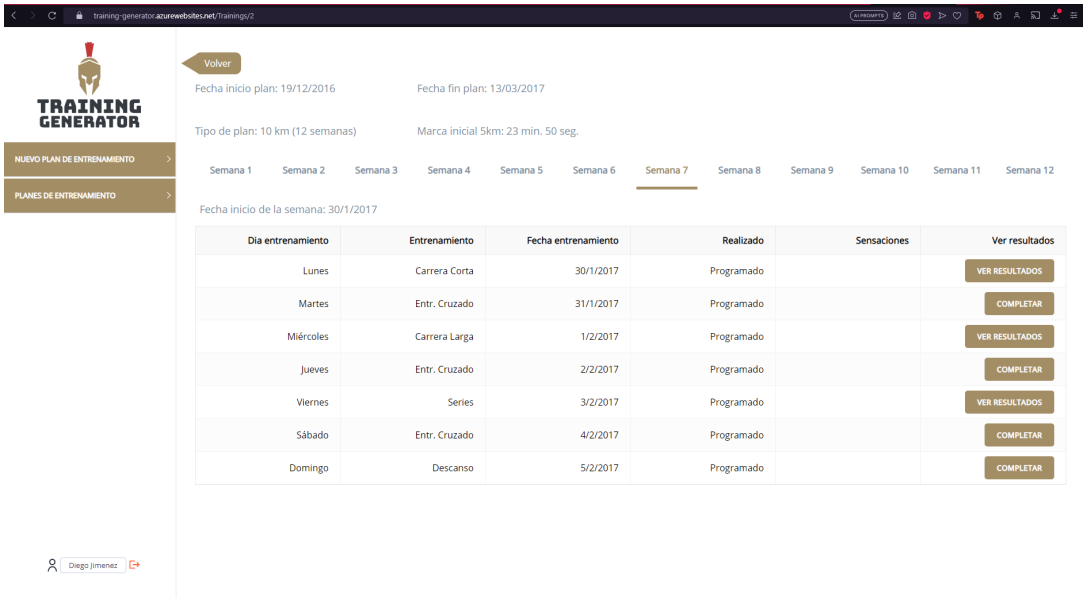


Figura 7.1: Creación de un plan de entrenamiento

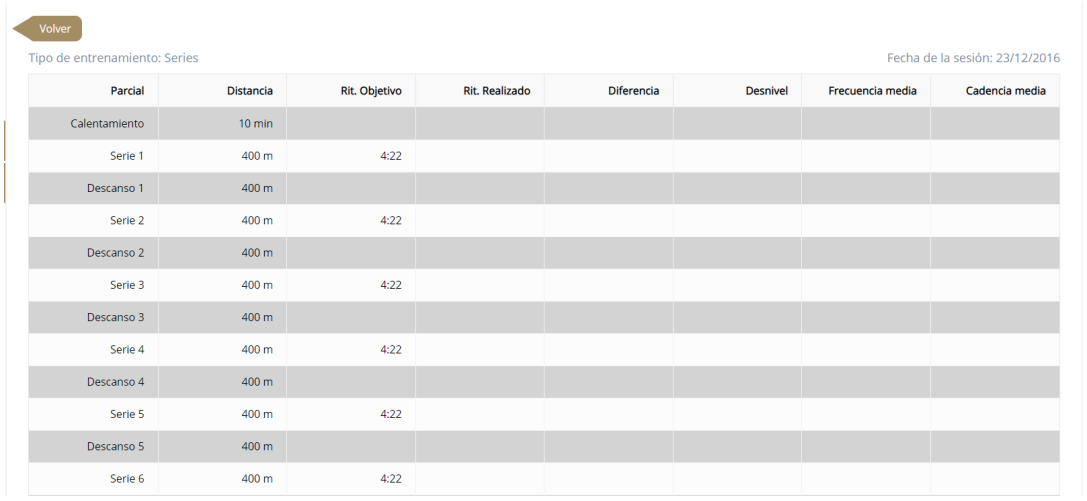
Al guardar estos datos la pantalla redirige al usuario al detalle del entrenamiento, donde se puede observar la distribución semanal seleccionada además de las fechas indicadas para su entrenamiento (Ver figura 7.2).



Dia entrenamiento	Entrenamiento	Fecha entrenamiento	Realizado	Sensaciones	Ver resultados
Lunes	Carrera Corta	30/1/2017	Programado		VER RESULTADOS
Martes	Entr. Cruzado	31/1/2017	Programado		COMPLETAR
Miércoles	Carrera Larga	1/2/2017	Programado		VER RESULTADOS
Jueves	Entr. Cruzado	2/2/2017	Programado		COMPLETAR
Viernes	Series	3/2/2017	Programado		VER RESULTADOS
Sábado	Entr. Cruzado	4/2/2017	Programado		COMPLETAR
Domingo	Descanso	5/2/2017	Programado		COMPLETAR

Figura 7.2: Plan semanal de entrenamiento

El usuario en este momento quiere saber los entrenamientos que tiene que realizar durante la semana. Consulta los entrenamientos de series, carrera corta y carrera larga, y se centra en el de series que es el viernes, entrando en el detalle del entrenamiento. Se puede observar en la figura 7.3



Parcial	Distancia	Rit. Objetivo	Rit. Realizado	Diferencia	Desnivel	Frecuencia media	Cadencia media
Calentamiento	10 min						
Serie 1	400 m	4:22					
Descanso 1	400 m						
Serie 2	400 m	4:22					
Descanso 2	400 m						
Serie 3	400 m	4:22					
Descanso 3	400 m						
Serie 4	400 m	4:22					
Descanso 4	400 m						
Serie 5	400 m	4:22					
Descanso 5	400 m						
Serie 6	400 m	4:22					

Figura 7.3: Entrenamiento del Viernes

Tras esto llegan los días de los entrenamientos y el usuario los realiza correctamente, registrando en su dispositivo inteligente los resultados obtenidos

durante los entrenamientos. Estos resultados el usuario los sube a *Strava* y los consulta desde la web (Ver figura 7.4).

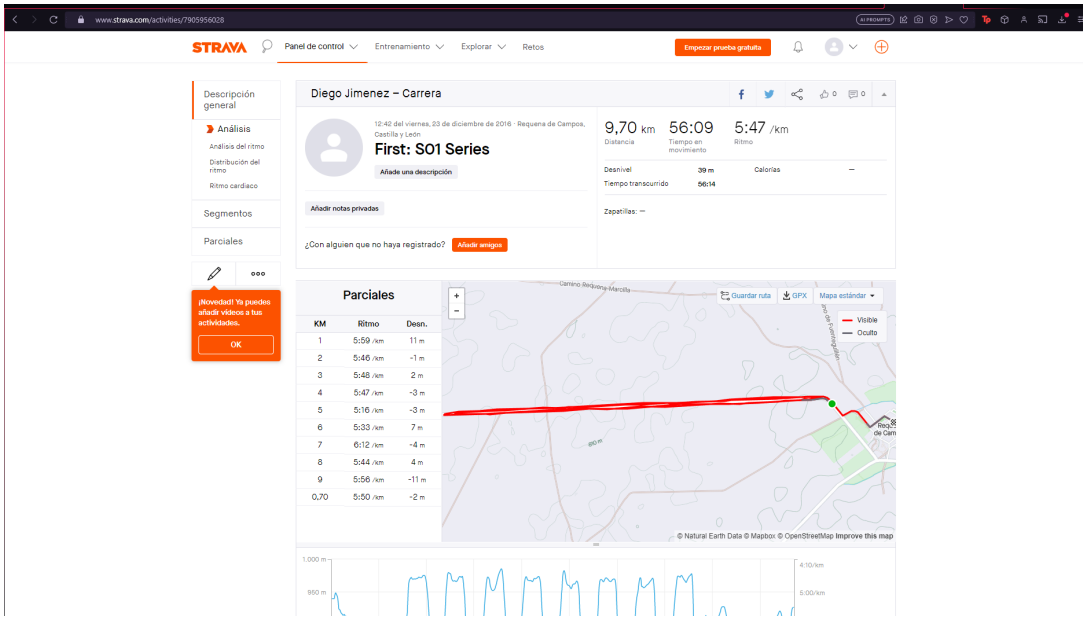


Figura 7.4: Pantalla de *Strava* con datos sobre las series realizadas

Al tener las actividades subidas a *Strava*, el usuario decide actualizarlas en la aplicación de Training Generator. Para realizar esto hace click en el botón de *Actualizar desde Strava* de la pantalla principal de la aplicación y obtiene un mensaje informándole que los datos se han actualizado correctamente (Ver figura 7.5). Esta función mezcla las actividades que el usuario ha subido a *Strava* junto con los datos de la aplicación Training Generator.

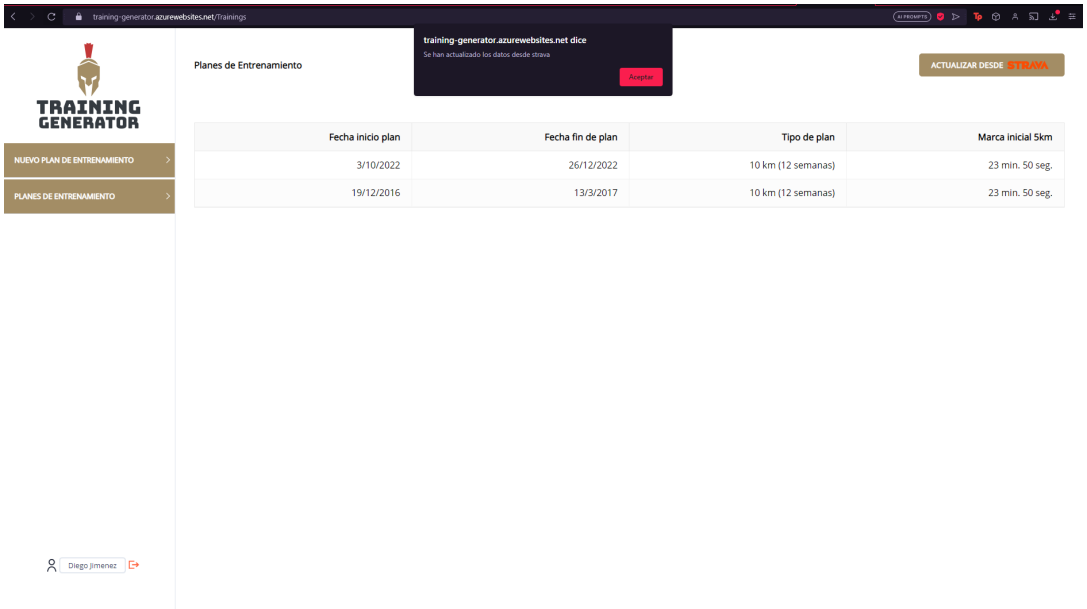


Figura 7.5: Datos actualizados desde *Strava*

Posteriormente, el usuario accede al plan de entrenamiento, ya que quiere ver la comparación de los resultados obtenidos con los objetivos planteados por la aplicación. El usuario entra en el detalle del plan de entrenamiento y observa que aparecen los días de la semana actual sincronizados (Ver figura 7.6).

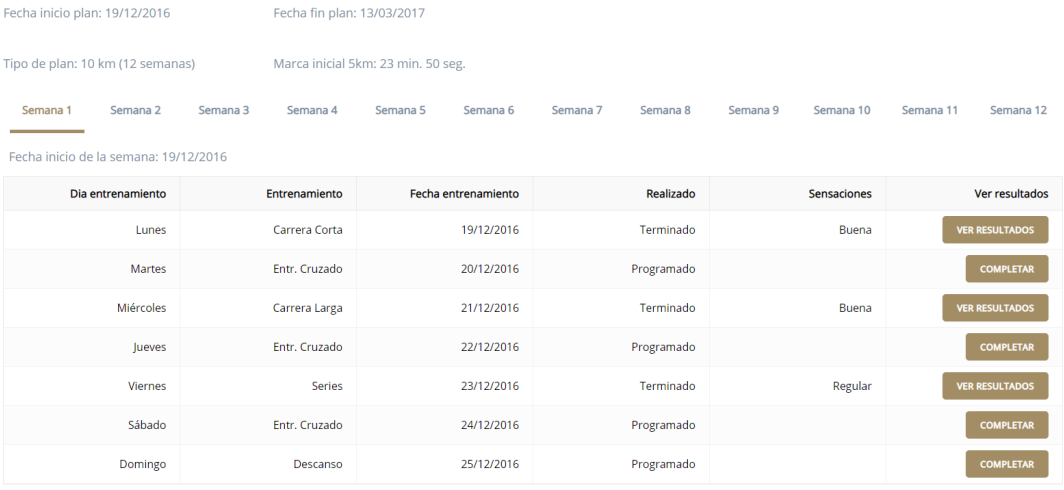


Figura 7.6: Días de la primera semana de entrenamientos ya sincronizados

El usuario entonces vuelve a entrar en el detalle del entrenamiento de Series, para poder comparar los resultados obtenidos. Se puede observar que los datos obtenidos aparecen registrados en cada fila, junto con los ritmos, distancias

y frecuencias cardiacas. Además, el usuario en este momento observa que el ritmo que ha realizado está ajustado con los ritmos objetivos propuestos por la aplicación, ya que la diferencia de los ritmos propios con los objetivos no es muy alta (Ver figura 7.7).

Parcial	Distancia	Rit. Objetivo	Rit. Realizado	Diferencia	Desnivel	Frecuencia media
Calentamiento	1613 m		6:11		13	153
Serie 1	400 m	4:22	4:25	3	0	180
Descanso 1	400 m		6:15		0	166
Serie 2	400 m	4:22	4:20	-2	2	176
Descanso 2	400 m		6:27		0	166
Serie 3	400 m	4:22	4:17	-5	0	177
Descanso 3	400 m		6:15		0	169
Serie 4	400 m	4:22	4:20	-2	0	176
Descanso 4	400 m		6:00		0	168
Serie 5	400 m	4:22	4:25	3	4	180
Descanso 5	400 m		6:37		2	167
Serie 6	400 m	4:22	4:20	-2	0	178
Descanso 6	400 m		6:40		0	165

Figura 7.7: Entrenamiento de series completado

Tras esto el usuario accede a consultar el entrenamiento de Carrera Corta. Entrando en el detalle del entrenamiento del lunes puede observar que para la carrera corta no ha cumplido el objetivo, quedándose por debajo del ritmo objetivo en 6 minutos (Ver figura 7.8).

Figura 7.8: Entrenamiento de Carrera corta completado

Por último, el usuario quiere comprobar sus resultados para la Carrera Larga de esa semana, ya que es lo último que le queda por comprobar. Accede al detalle

y comprueba que el ritmo realizado está muy cerca del ritmo objetivo (Ver figura 7.9), por lo que se siente satisfecho con los resultados obtenidos.

Figura 7.9: Entrenamiento de Carrera Larga completado

Junto a todo esto, el usuario hace click en los botones de Completar de la figura 7.6 para marcar esos días como completados en vez de programados, pudiendo llevar de esta manera un registro de los días según va avanzando en el plan de entrenamiento.

8

Conclusiones y trabajo futuro

En este capítulo se describirán las conclusiones sobre el trabajo realizado, y posibles mejoras a aplicar en la aplicación desarrollada.

Como conclusión, con esta aplicación desarrollada podemos ahorrar mucho tiempo y esfuerzo en crear un plan de entrenamiento basado en *FIRST*.

De los objetivos principales se ha conseguido crear una aplicación web que permite crear planes de entrenamiento basados en el entrenamiento *FIRST* y sincronizar los mismos con las actividades añadidas en *Strava*. Los usuarios son capaces de saber qué tienen que realizar cada día de entrenamiento de una manera sencilla y visual.

También los usuarios pueden visualizar correctamente los resultados obtenidos y ver fácilmente la diferencia de los mismos con los objetivos propuestos por el entrenamiento.

Además, la aplicación puede desplegarse automáticamente a través de un pipeline de trabajo en GitHub en un servicio en la nube de Azure.

Para un trabajo futuro se podría añadir que permita reordenar los días cada

semana, ya que es posible que algún usuario cambie días de entrenamiento o alguna semana modifique su entrenamiento para adaptarla más a su tiempo libre.

También se podrían añadir nuevos tipos de entrenamiento, que den más variedad al usuario a la hora de tener distintos entrenamientos. Para los días de entrenamiento cruzado se podría proponer al usuario opciones y que las marque, haciendo de esta manera que varíe también en estos tipos de entrenamientos.

Además, se quiere modificar la descripción de la actividad de *Strava* con los resultados obtenidos, de tal manera que el usuario pueda visualizar tanto en *Strava* como en la aplicación los resultados obtenidos con la diferencia con los tiempos objetivos.

Bibliografía

[1] Entrenamiento first en Carreras populares.com. URL: https://www.carreraspopulares.com/noticia/plan-first-corre-menos-para-correr-mas (vid. pág. 1).

[2] Página de entrenamientos. URL: https://madmuscles.com (vid. pág. 3).

[3] Plan de entrenamiento First en foro atletismo. URL: https://www.foroatletismo.com/planes-de-entrenamiento/plan-de-entrenamiento-first-correr-rapido-corriendo-menos/ (vid. págs. 4, 9).

[4] Método first en ciclonomadas. URL: https://ciclonomadas.com/metodo-first/ (vid. pág. 4).

[5] Planes de entrenamientos premium en runnea. URL: https://shop.runnea.academy/collections/suscripcion-premium (vid. pág. 4).

[6] Entrenamientos para carreras en sportssantander. URL: https://www.sportssantander.es/Running/entrenamientos/maraton-acabar-la-carrera (vid. pág. 7).

[7] Planes de entrenamiento para carreras en marathonranking. URL: https://www.marathonranking.com/noticias-sobre-maraton/planes-de-entrenamiento-para-5k-10k-21k-y-42k/ (vid. pág. 8).

[8] Página web de Strava. URL: https://www.strava.com/get-started (vid. pág. 11).

[9] Documentación de las apis de Strava. URL: https://developers.strava.com/docs/reference/ (vid. pág. 11).

[10] Página web para realizar diagramas. URL: https://www.diagrams.net/ (vid. pág. 14).

[11] Página web para realizar prototipos. URL: https://marvelapp.com/ (vid. págs. 14, 34).

[12] Herramienta de javascript para hacer rejillas. URL: http://js-grid.com/ (vid. págs. 16, 40).

[13] Github actions. URL: https://github.com/features/actions (vid. pág. 16).

[14] Scrum - Wikipedia. URL: https://es.wikipedia.org/wiki/Scrum_(desarrollo_de_software) (vid. pág. 17).

[15] Página de generación de logos. URL: https://www.freelogodesign.org/ (vid. pág. 38).

[16] SQL Server management Studio. URL: https://learn.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-ver16 (vid. pág. 42).

[17] Enterprise Library. URL: https://en.wikipedia.org/wiki/Microsoft_Enterprise_Library (vid. pág. 46).