UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf
Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Diseño de una aplicación web para el análisis de la evolución de carreras a pie

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Gonzalo San Bartolomé Marco

Tutor: Miguel Rebollo Pedruelo

Curso 2016-2017

Resumen

Para los corredores y los organizadores de las carreras a pie es muy importante los tiempos que se realizan, la afluencia de gente en determinadas zonas y la totalidad de las personas que participan en ellas para así poder prepararse mejor, tanto para correr la carrera como para organizarla correctamente. Las aplicaciones actuales no permiten un análisis de la carrera con todo el conjunto de los corredores, existe alguna que permite a los que hayan realizado el mismo recorrido compararse pero para ello es necesario que cada corredor esté registrado en la plataforma. Como solución se ha desarrollado una aplicación web que permite con solo la clasificación final ver el avance de los corredores, así como ciertas estadísticas importantes.

Palabras clave: corredores, carreras a pie, análisis, aplicación web, estadísticas.

Abstract

For runners and organizers of foot races is very important the times made, the influx of people in certain areas and all of the people involved in them to better prepare, both to run the race and to organize it correctly. Current applications do not allow analysis of the race with the whole set of riders, there is one that allows those who have done the same course to compare but for that it is necessary that each runner is registered on the platform. A solution has been developed a web application that allows, with only the final classification, to see the progress of the runners, as well as certain important statistics.

Keywords : runners, foot races, analysis , web application, statistics.

Tabla de contenidos

Índice de Ilustraciones

Índice de Tablas

Índice de Códigos

1. Introducción

1.1. Contexto

El atletismo es considerado el deporte más antiguo del mundo en el que se abarcan numerosas disciplinas, las carreras a pie entre ellas. En estas carreras se tiene en cuenta la velocidad de los corredores y los tiempos realizados. Estas medidas siempre se han tomado con instrumentos básicos, como los cronómetros.

A lo largo de los años esta disciplina ha ido expandiéndose por el mundo dando la oportunidad de realizar estas pruebas a cualquier persona y en cualquier espacio. El aumento de afluencia a estas carreras ha producido un incremento en la organización, tanto de los corredores como de los organizadores, y un interés en la mejora de los tiempos finales.

La evolución de la tecnología y de la ingeniería informática ha permitido desarrollar aplicaciones que proporcionan medidas prácticamente exactas de los tiempos realizados y otras estadísticas para así facilitar el trabajo de los corredores, aunque todavía no existe ninguna aplicación que permita ver el transcurso de una carrera con todos sus corredores.

En este proyecto se desarrolla una aplicación web en la que se muestra el avance de los corredores, mostrando donde se pueden acumular los corredores y estadísticas variadas. La aplicación es completamente gratis sin necesidad de registrarse y puede servir a corredores como a organizadores.

1.2. Objetivos

El objetivo principal del proyecto es desarrollar una aplicación web que permita ver el desarrollo de las carreras a pie, mostrando a todos los participantes, y dando la posibilidad de ver estadísticas generales. Esto facilita un análisis de los corredores que ayuda a los organizadores a preparar mejor las carreras. Este objetivo se desarrolla a través de otros objetivos más específicos:

- Analizar los sistemas similares más destacables que tengan funcionalidades similares al proyecto para definir las características generales de este.
- Permitir la subida del track de la carrera mediante un archivo.
- Permitir la subida de la clasificación.
- Visualizar la posición de los corredores durante todo el recorrido.

- Recalcar la posición de un corredor durante el transcurso de la carrera.
- Proporcionar estadísticas generales de las carreras.
- Proporcionar estadísticas individuales del corredor remarcado.

1.3. Descripción del documento

El documento se estructura en seis capítulos que detallan cada una de las fases del proyecto

Capítulo 1: Introducción

El primer capítulo se basa en la introducción al problema que se plantea, a la situación en la que se encuentra y a la solución que se prevé dar.

Capítulo 2: Estado del arte

La primera fase consiste en un análisis cuantitativo y cualitativo de los sistemas similares al proyecto, esto permite determinar qué puntos fuertes y puntos débiles tienen otro sistemas, y cuales queremos que tenga el sistema a desarrollar.

Capítulo 3: Especificación de requisitos

A continuación se determinan, gracias a las características del sistema que se han definido en el capítulo dos, las funcionalidades y los requisitos que el sistema deberá cumplir.

Capítulo 4: Diseño del sistema

Se muestra un diagrama conceptual que muestra de forma abstracta el sistema, seguido de una descripción del diseño basada en el modelo de programación por capas.

Capítulo 5: Implementación, implantación y evaluación del sistema

Tras los capítulos anteriores se muestra el resultado final del sistema, ya implementado, mediante capturas de pantalla y líneas de código relevantes.

Capítulo 6: Conclusiones

Para finalizar se exponen las conclusiones del proyecto. Se recalcan las dificultades halladas y las soluciones aplicadas durante el desarrollo, las aportaciones realizadas y las futuras mejoras que pueden enriquecer al sistema.

2. Estado del arte

2.1. Introducción

En el presente capítulo se abordará la discusión sobre algunos de los distintos modelos de aplicación web que pueden ser utilizados para la realización del seguimiento de carreras a pie.

Para empezar se expondrá el ámbito de realización del proyecto, una comparativa de las aplicaciones web ya existentes similares a la que se desea realizar, un estudio cuantitativo y cualitativo de estas aplicaciones, y por último una justificación del sistema desarrollado en comparación con los sistemas actuales.

2.2. Entorno de realización

Las carreras a pie no han estado muy relacionadas con la tecnología, no obstante existen desde hace algún tiempo distintos dispositivos y plataformas que permiten el seguimiento de las carreras y proporcionan información sobre estas a sus usuarios.

En este capítulo se realiza un análisis de este tipo de sistemas una vez probados y utilizados. Después de ser analizados se pueden determinar las características que ha de tener el sistema.

2.3. Sistemas similares

A continuación se muestran los distintos sistemas similares a nuestro proyecto que se han encontrado.

2.3.1. Runkeeper

Nombre	Runkeeper
Referencia	https://runkeeper.com
Desarrollador	FitnessKeeper, Inc.
Descripción	RunKeeper es una aplicación web y móvil de seguimiento para deportistas que recopila datos, mediante el GPS del smartphone, durante las actividades que realicemos. Esto nos proporciona un historial de actividades con estadísticas, dentro de estas estadísticas se incluyen el ritmo, la distancia y el tiempo.

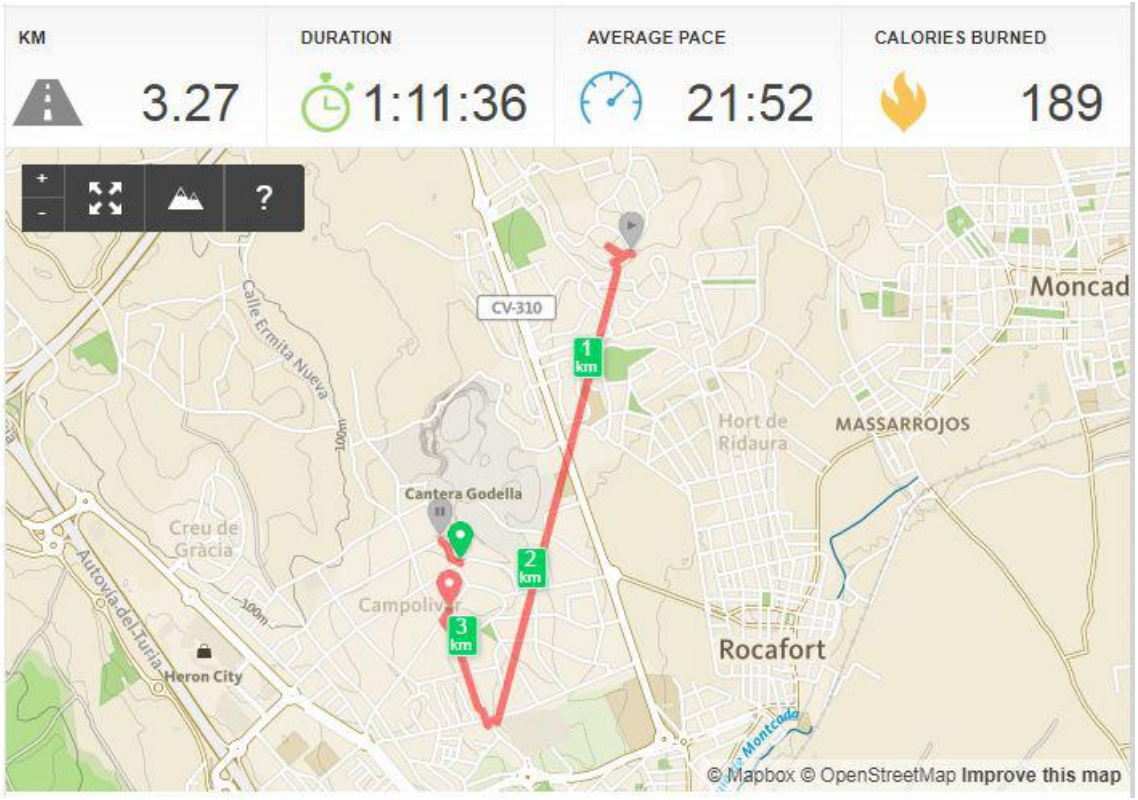

Ilustración 1: Interfaz de Runkeeper con recorrido del usuario mostrando la duración y kilómetros recorridos.

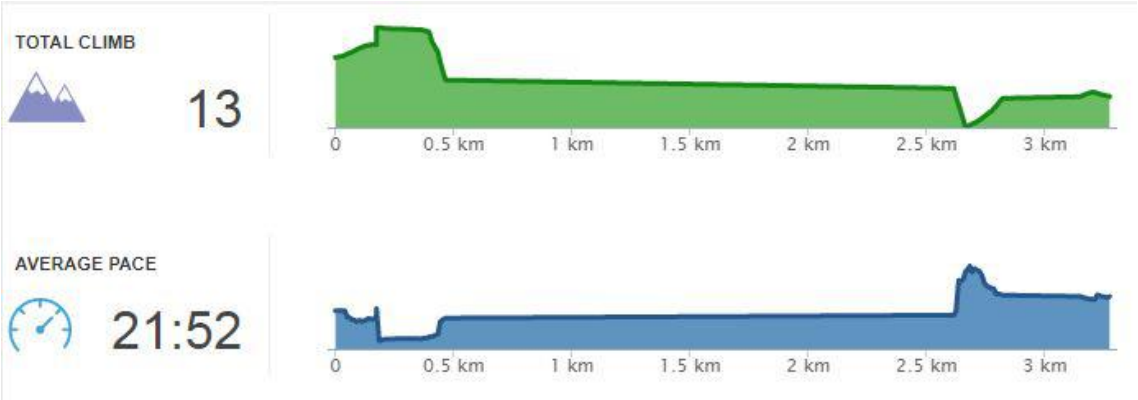

Ilustración 2: Interfaz de Runkeeper con gráficas de desnivel y de tiempos parciales.

2.3.2. Endomondo

Nombre	Endomondo
Referencia	https://www.endomondo.com
Desarrollador	Endomondo.com
Descripción	Endomondo es una aplicación web y móvil que permite el seguimiento en tiempo real de las actividades deportivas que se realizan. Nos proporciona estadísticas sobre la distancia recorrida, la velocidad, altitud y localización gracias al GPS y la utilización de los mapas de Google, a su vez nos muestra un historial de nuestro entrenamientos y nuestro progreso.

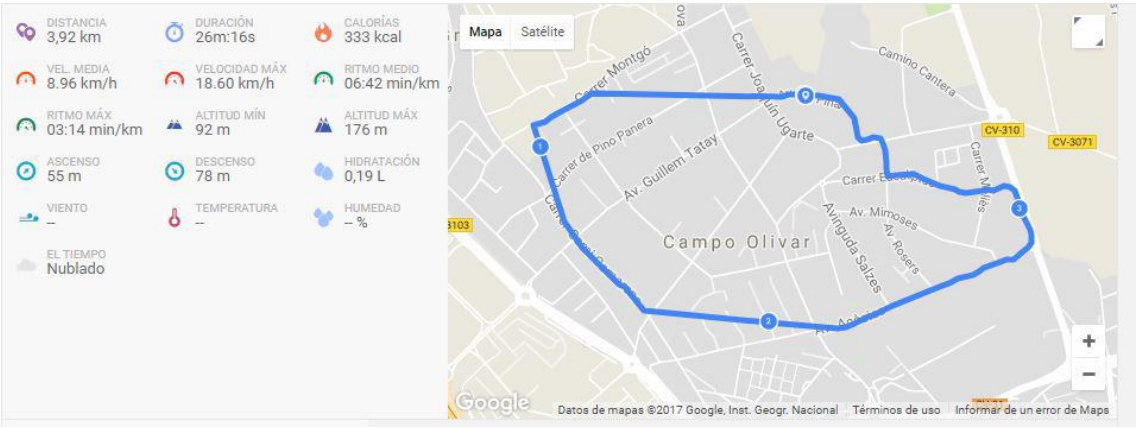

Ilustración 3: Interfaz de Endomondo que muestra el recorrido realizado con información variada como la velocidad media, la duración o la distancia recorrida.

Ilustración 4: Interfaz de Endomondo con gráfica de desnivel y de ritmo.

2.3.3. Runtastic

Nombre	Runtastic
Referencia	https://www.runtastic.com
Desarrollador	runtastic GmbH
Descripción	Runtastic es una aplicación móvil y web que te acompaña a la hora de hacer deporte, registra tus actividades fitness y deportivas guardando un historial de los entrenamientos. Las estadísticas que proporciona son la velocidad media, la distancia recorrida, las calorías quemadas, el ritmo y la duración.

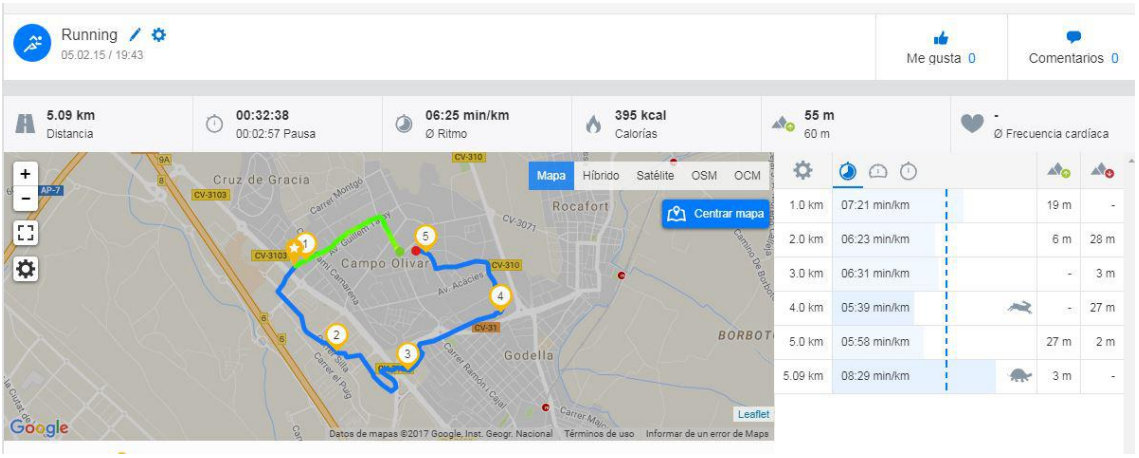

Ilustración 5: Interfaz de Runtastic con el recorrido realizado y la velocidad media por kilómetro.

2.3.4. Strava

Nombre	Strava
Referencia	https://www.strava.com/
Desarrollador	Strava Inc.
Descripción	Strava es una red social enfocada a deportistas, es a su vez una aplicación de seguimiento GPS, proporciona un análisis detallado de los datos que recoge, como la velocidad de la trayectoria realizada, las diferencias de altura y el consumo de energía, a través de sensores se pueden almacenar muchos más datos.

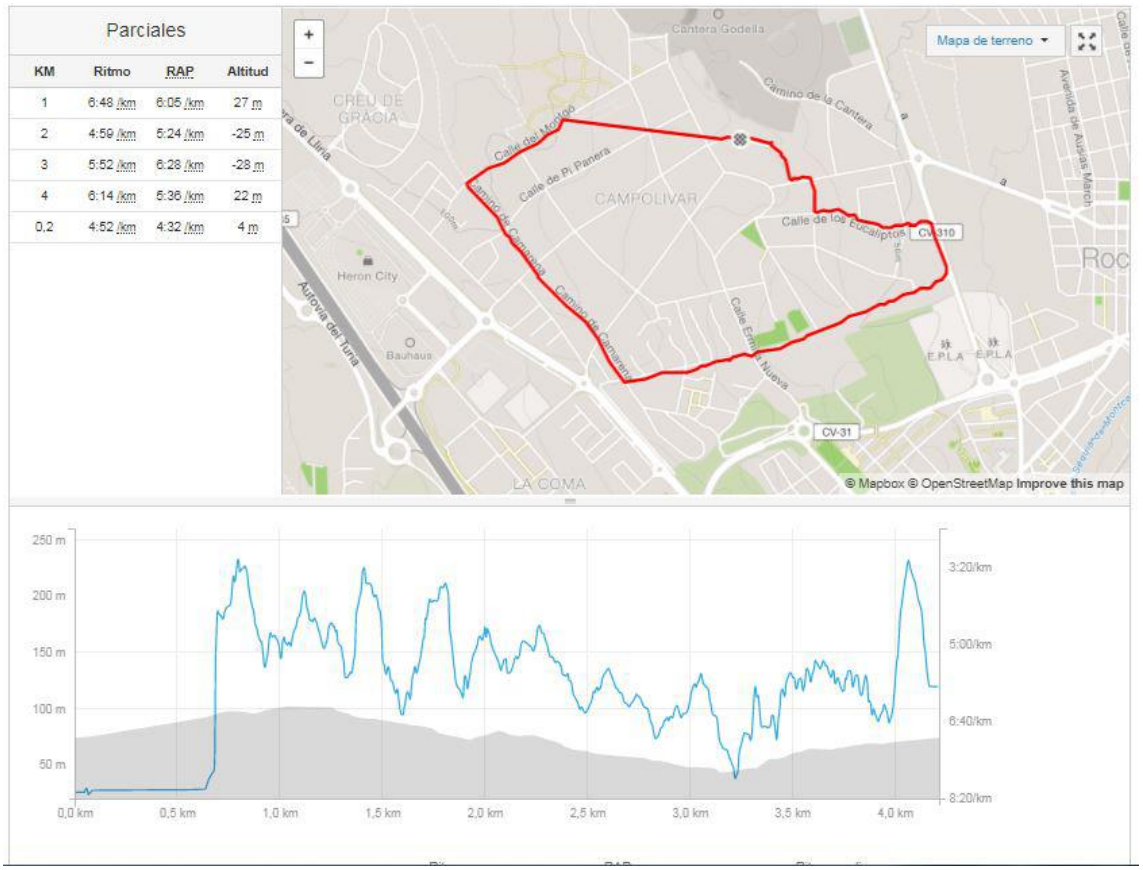

Ilustración 6: Interfaz de Strava con el recorrido realizado, la velocidad media por kilómetro y unas gráficas de desnivel y de ritmo.

2.3.5. Strava Flyby

Nombre	Strava Flyby
Referencia	https://www.strava.com/
Desarrollador	Strava Inc.
Descripción	Strava Flyby es una nueva herramienta de comparación que muestra la distancia y la elevación a lo largo de su actividad. unos círculos de color muestran las posiciones de otros usuarios que han recorrido el mismo camino que el usuario principal.

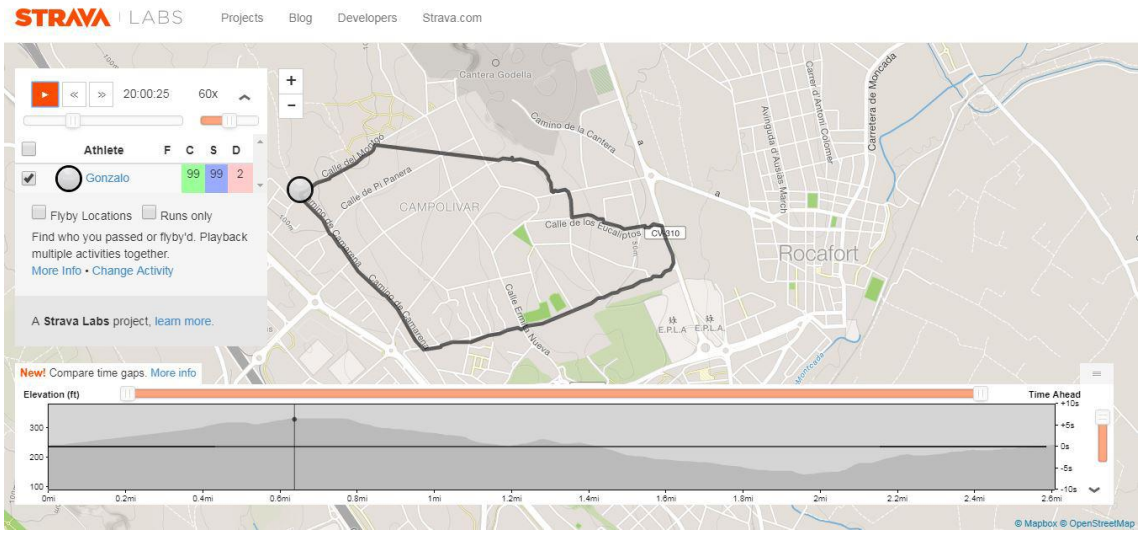

Ilustración 7: Interfaz de Strava FlyBy que muestra el recorrido realizado y permite ver el transcurso de la carrera.

2.4. Análisis

2.4.1. Objetivo (cuantitativo)

En la selección de los sistemas similares se han escogido aquellos que comparten ideas con el proyecto y con las carreras a pie. Todos ellos presentan propiedades relevantes, por esta razón se ha realizado un análisis objetivo de las siguientes características que comparten.

Características analizadas:

- Información de Km y tiempos en intervalos: especifica si la información obtenida de los tiempos y los kilómetros viene dada por intervalos.
- Resultados fiables: especifica si la proporción de la información es fiable o no.
- Importación o exportación de actividades: especifica si la plataforma permite la importación o exportación de las actividades de otro sistema.

- Aspecto social: especifica si el sistema tiene la opción de compartir las actividades en las principales redes sociales.
- Competividad entre sí mismo: especifica si el sistema permite a un usuario competir entre sí mismo.
- Competividad entre usuarios: especifica si el sistema permite a los usuarios competir entre ellos.
- Estadísticas de otros usuarios: especifica si el sistema proporciona estadísticas de otros usuarios.
- Estadísticas globales de una misma ruta: especifica si el sistema proporciona estadísticas de todos los corredores de una misma ruta.

Una vez se han revisado los sistemas es hora de analizarlos mediante una tabla organizada donde se puedan ver las diferencias y similitudes de los sistemas.

	Runkeeper	Endomondo	Runtastic	Strava	Strava Flyby
Info de Km y tiempos en intervalos	----------	X	X	X	----------
Resultados fiables	----------	X	X	X	X
Importación o exportación de actividades	----------	X	X	X	----------
Aspecto social	X	X	X	X	----------
Competividad entre usuarios	----------	----------	----------	X	----------
Competividad entre si mismo	X	----------	X	X	----------
Estadísticas de otros usuarios	----------	----------	X	X	----------
Estadísticas globales de una misma ruta	----------	----------	----------	-------	X

Tabla 1: Comparación de la característica cuantitativas de los sistemas similares.

Como se puede observar la mayoría de los sistemas proporcionan unos datos fiables, tienen aspecto social y la información de los kilómetros y los tiempos se muestran en intervalos. No obstante, la mayor parte de ellos no permite ver las estadísticas globales de una misma ruta.

2.4.2. Subjetivo (cualitativo)

Después de haber analizados objetivamente los sistema similares es necesario analizar las características subjetivas que presentan cada uno de ellos. Para ello se ha realizado una comparación cualitativa de las propiedades más relevantes, se ha utilizado una escala del 1 al 5 para puntuar, donde cada valor representa lo siguiente:

- 1: Muy insatisfecho
- 2: Insatisfecho
- 3: Indiferente
- 4: Satisfecho
- 5: Muy satisfecho

Las características que se han analizado son:

- Facilidad de uso: Describe el nivel de dificultad que se encuentra un usuario a la hora de utilizar la aplicación.
- Diseño: Describe el nivel visual al que se enfrenta el usuario.
- Funcionalidad: Describe el número de funcionalidades que presenta la aplicación.
- Navegabilidad: Describe el nivel de dificultad que un usuario tiene al navegar por los diferentes menús.

Para poder observar las diferencias y similitudes, se analizara mediante una tabla organizada los distintos sistemas.

	Runkeeper	Endomondo	Runtastic	Strava	Strava Flyby
Facilidad de uso	4	5	4	4	4
Diseño	3	4	4	5	5
Funcionalidad	4	4	4	5	5
Navegabilidad	5	5	5	5	5

Tabla 2: Comparación de la característica cualitativas de los sistemas similares.

La mayoría de los sistemas presentan una funcionalidad, navegabilidad y usabilidad eficientes.

2.5. Síntesis (justificar el sistema desarrollado)

Una vez terminado el análisis de las distintas características que presentan los sistemas analizados, es el momento de elegir aquellas que formarán parte de nuestra aplicación. La siguiente tabla muestra las características escogidas, se ha utilizado una nomenclatura del tipo Fxx donde x son valores numericos que permiten la ordenación de las características.

Referencia	**Funcionalidad**
F01	Capacidad para mostrar datos fiables.
F02	Capacidad para importar actividades.
F03	Mostrar estadísticas generales.
F04	Mostrar estadísticas individuales.
F05	Mostrar el seguimiento de los corredores.
F06	Mostrar el seguimiento de un corredor.

Tabla 3: Características del sistema.

2.6. Tecnología a emplear

A continuación se va a tener en cuenta que entornos y módulos se van a utilizar para desarrollar la aplicación.

Para el desarrollo de la aplicación se ha decidido utilizar Node.js, un entorno en tiempo de ejecución para Javascript asíncrono que se ejecuta en el servidor. Se ha decidido utilizar Node.js ya que usa un modelo de operaciones de entrada/salida sin bloqueo y orientado a eventos.

También se utilizará el framework Express.js con el que se desarrollan la mayoría de las aplicaciones de Node.js y que nos proporciona funcionalidades como el enrutamiento.

Para la capa de persistencia utilizaremos MongoDB, un sistema de bases de datos no relacional que está orientado a documentos, es decir, guarda los datos en documentos. Para conectar MongoDB con Node.js se requerirá instalar mongoose.js que nos proporciona una solución basada en esquemas.

Además también se van a mostrar estadísticas de las carreras y para ello se piensa hacer uso la herramienta Google Charts.

2.7. Conclusiones

Al inicio del capítulo se ha estudiado el entorno sobre el que se iba a desarrollar el proyecto con el objetivo de determinar qué características iba a poseer nuestra aplicación.

Posteriormente se ha recopilado información de los distintos sistemas similares al que se quiere desarrollar para realizar un análisis objetivo y

subjetivo, de forma que se puedan obtener las funcionalidades deseadas. Aunque muchas de estas funcionalidades son importantes no satisfacen el objetivo principal del proyecto, que es la posibilidad de ver estadísticas generales en una carrera a pie.

Para el desarrollo se ha optado por la herramienta Node.js, junto a módulos externos, también se hará uso de MongoDB, un sistema de base de datos no relacional, y por último, la utilización del servicio Google Charts para la estadísticas a mostrar.

3. Especificación de requisitos

3.1. Introducción

El presente capítulo es una especificación de requisitos del software para una aplicación que permita el análisis y la visualización de los datos de las carreras a pie.

3.1.1. Propósito

El propósito de la aplicación web que se va a implementar es proporcionar una herramienta de análisis de carreras, tanto para los corredores como para los organizadores. De esta manera se podría ocupar ese espacio que las aplicaciones para correr todavía no han explotado.

3.1.2. Ámbito del sistema

El sistema a desarrollar es una aplicación web capaz de importar los tracks de las carreras y mostrar, a la vez, estadísticas sobre la actividad y el transcurso de esta.

La aplicación no se encargará de crear los tracks necesarios para la visualización de la carrera, como tampoco los datos de los corredores. El sistema será un intermediario entre los datos y la visualización de estos.

La principal función será la de visualizar el recorrido de la carrera y como cada uno de los corredores avanza en ella. Conjuntamente, una segunda función se encargará de recoger los datos y mostrar un análisis de ellos mediante estadísticas.

El sistema facilitará tanto a los corredores como a los organizadores de las carreras la recogida y el análisis de datos, mostrando estadísticas que se consideran útiles para la organización de este tipo de eventos.

3.1.3. Definiciones, acrónimos y abreviaturas

Definiciones:

- Organizador: Se trata de una persona que desea organizar una carrera a pie.
- Participante: Se trata de la persona que desea correr cierta carrera.
- Sistema: Se trata de la aplicación web que se va a desarrollar.
- Track: Un track es una sucesión de waypoints concatenados que definen un camino recorrido.
- Waypoint: Un waypoint es una posición, en un lugar determinado de este planeta, almacenada por un receptor GPS.

- Framework: Un framework es un esquema para el desarrollo y/o la implementación de una aplicación.

3.2. Descripción general

3.2.1. Perspectiva del producto

La aplicación será desarrollada para su utilización en entornos web, de manera independiente sin formar parte de otro proyecto.

3.2.2. Características de los usuarios

En el sistema solo habrá un tipo de usuario ya que no será necesario un registro previo para poder utilizar la aplicación, por lo tanto todos los usuarios podrán realizar las misma acciones. Serán capaces de subir los tracks a la aplicación, subir la clasificación y observa, tanto las estadísticas como el transcurso de la carrera.

3.2.3. Funciones del producto

El sistema proporciona las funciones que se presentan a continuación con un diagrama de casos de uso.

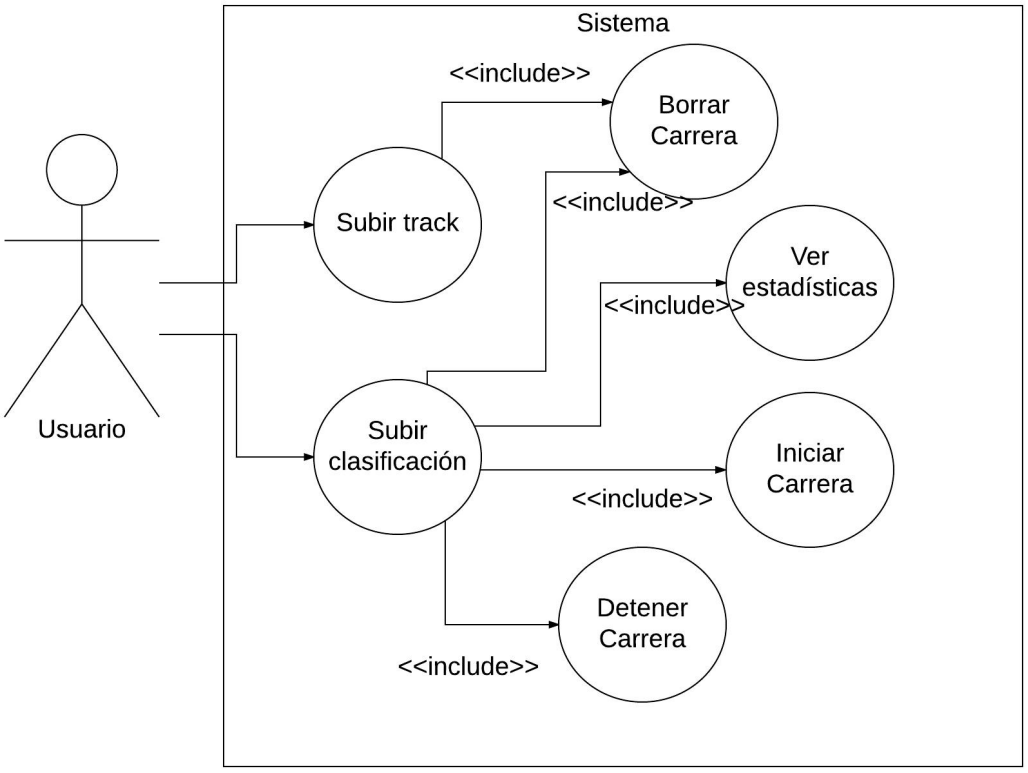

Ilustración 8: Diagrama de casos de uso.

Se ha realizado una tabla describiendo los casos de uso y dándoles una referencia, además se ha relacionado cada caso de uso con las características de la tabla 8.

Referencia	Descripción	Características
CU01	Subir track a la aplicación.	F01F02
CU02	Subir clasificación a la aplicación.	F01,F02
CU03	Borrar carrera.	
CU04	Ver estadísticas de la carrera.	F03,F04
CU05	Iniciar la carrera.	F05,F06
CU06	Detener la carrera.	F05,F06

Tabla 4: Casos de uso del sistema.

3.2.4. Restricciones

Las restricciones presentada por el sistema son pocas, será necesario un equipo capaz de ejecutar a la vez el servidor Node y el sistema de base de datos de MongoDB.

3.2.5. Suposiciones y dependencias

Se considera que los usuarios tienen acceso a internet mediante algún navegador web, y que a su vez poseen unos conocimiento mínimos de navegación web.

Para que la aplicación funcione correctamente se presupone que el track y la clasificación que el usuario suba a la aplicación no sean de carreras distintas, por lo tanto ha de tener en posesión un track de la carrera que desea visualizar y la clasificación de esta.

3.2.6. Requisitos futuros

Algunas posibles mejoras para el sistema que podrían implementarse en un futuro serían las siguientes:

- La implementación de un sistema de registro.
- La posibilidad de centrar el análisis de la carrera en un grupo de personas.

- Desarrollo de una aplicación móvil.

3.3. Requisitos específicos

En esta sección se procede a detallar los requisitos del sistema de manera que el diseño de adapte de la mejor manera a ellos.

El análisis obtenido de ser suficiente para desarrollar de forma correcta la aplicación, incluyendo las restricciones propias y la funcionalidad del sistema.

3.3.1. Interfaces externas

Referencia	R01
Nombre	Diseño adaptable
Tipo	Interfaz
Descripción	La interfaz de la aplicación debe ser adaptable para los distintos dispositivos donde se pueda utilizar.
Prioridad	Media
Rol	Usuario

Tabla 5: Características del requisito R01.

3.3.2. Funciones

Referencia	R02
Nombre	Subir track
Tipo	Funcional
Descripción	El usuario ha de ser capaz de poder subir un track a la aplicación y que el recorrido se muestre en el mapa.
Prioridad	Alta
Rol	Usuario

Tabla 6: Características del requisito R02.

Referencia	R03
Nombre	Subir clasificación
Tipo	Funcional
Descripción	El usuario ha de ser capaz de poder subir la clasificación del track a la aplicación.
Prioridad	Alta
Rol	Usuario

Tabla 7: Características del requisito R03.

Referencia	R04
Nombre	Ver el transcurso de la carrera.
Tipo	Funcional
Descripción	El usuario ha de ser capaz de poder visualizar el paso de los corredores por toda la carrera.
Prioridad	Alta
Rol	Usuario

Tabla 8: Características del requisito R04.

Referencia	R05
Nombre	Ver el transcurso de un corredor de la carrera.
Tipo	Funcional
Descripción	El usuario ha de ser capaz de poder visualizar el paso de los corredores y ver recalcado el paso de un corredor en especial.
Prioridad	Alta
Rol	Usuario

Tabla 9: Características del requisito R05.

Referencia	R06
Nombre	Ver estadísticas generales de la carrera.
Tipo	Funcional
Descripción	El usuario ha de ser capaz de poder visualizar estadísticas generales de la carrera.
Prioridad	Alta
Rol	Usuario

Tabla 10: Características del requisito R06.

Referencia	R07
Nombre	Ver estadísticas de un corredor en particular.
Tipo	Funcional
Descripción	El usuario ha de ser capaz de poder visualizar estadísticas de un corredor concreto.
Prioridad	Alta
Rol	Usuario

Tabla 11: Características del requisito R07.

Referencia	R08
Nombre	Borrar carrera.
Tipo	Funcional
Descripción	El usuario ha de ser capaz de eliminar el contenido visualizado borrando la carrera.
Prioridad	Alta
Rol	Usuario

Tabla 12: Características del requisito R08.

3.3.3. Requisitos de rendimiento

No existen requisitos o restricciones del sistema aplicables a esta categoría.

3.3.4. Restricciones de diseño

No existen requisitos o restricciones del sistema aplicables a esta categoría.

3.3.5. Atributos del sistema

- Fiabilidad: La aplicación es sometida a una serie de pruebas para comprobar que todas y cada una de las funcionalidades desarrolladas se ejecutan de manera adecuada.
- Mantenibilidad: El sistema es objeto de mantenimiento.

3.4. Conclusiones

En este capítulo se ha realizado la especificación de requisitos del sistema, partiendo de las características mostradas en el capítulo dos y el diagrama de casos de uso se ha podido definir los requisitos específicos que ha de cumplir el sistema. La siguiente table recoge la relación entre los requisitos, los casos de uso y las características.

Requisitos	**Casos de uso**	**Característica**
R01	CU01, CU02, CU03, CU04, CU05, CU06	
R02	CU01	F02
R03	CU02	F01,F02
R04	CU05, CU06	F05,F06
R05	CU05, CU06	F05,F06
R06	CU04	F03,F04
R07	CU04	F03,F04
R08	CU03	

Tabla 13: Comparación de requisitos, casos de uso y características.

4. Diseño del sistema

4.1. Introducción

Una vez realizada la especificación de requisitos podemos continuar determinando el diseño del sistema.

Mediante una especificación conceptual se representará la estructura y el funcionamiento. A continuación una especificación formal se encargará de precisar las capas que lo componen: la capa de persistencia, la de la lógica y la de presentación.

4.2. Especificación conceptual

La ilustración 9 representa la estructura y el funcionamiento del sistema.

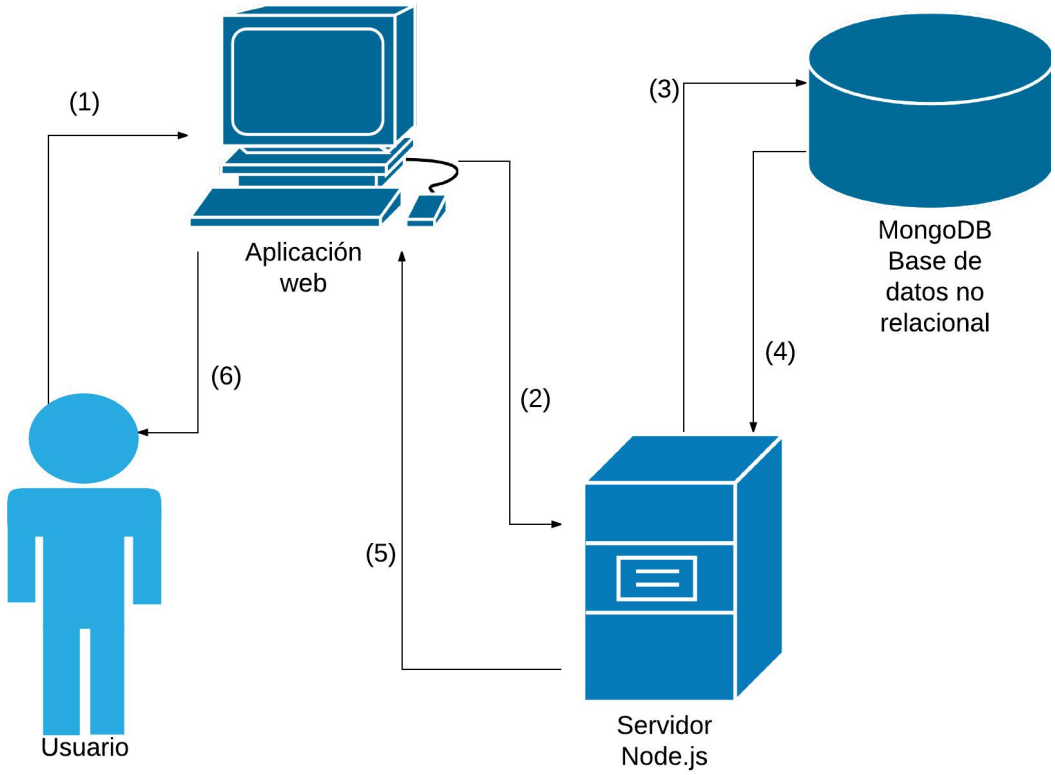

Ilustración 9: Estructura y funcionamiento del sistema.

Un usuario cualquiera interactúa desde un ordenador con la aplicación web (1). Desde ella es capaz de subir al servidor un track de una carrera, y la clasificación de esta. los datos son enviados al servidor (2) que se encarga a su vez de guardarlos en la base de datos de MongoDB (3). Una vez el usuario desea ver tanto las estadísticas como la visualización de la carrera (1), la aplicación solicita la representación del transcurso de la carrera (2), el servidor solicita a la base de datos la información necesaria (3). Seguidamente la base de datos le

proporciona la información y procesa los datos (4), realizando las operaciones convenientes, finalmente envía los datos a la aplicación (5) y por último la aplicación web muestra al usuario la visualización y estadísticas de la carrera (6).

4.3. Especificación formal

En este apartado comienza la formalización de las capas que se ha de implementar del sistema. Comenzando por la capa de presentación, seguido de la capa de persistencia y finalizando por la capa de la negocio.

4.3.1. Capa de presentación

En esta sección se muestra presentan una secuencia de bocetos representando la capa de presentación, la capa con la que el usuario interactúa. Cada dibujo viene acompañado de una explicación que permitirá entender mejor el funcionamiento del sistema.

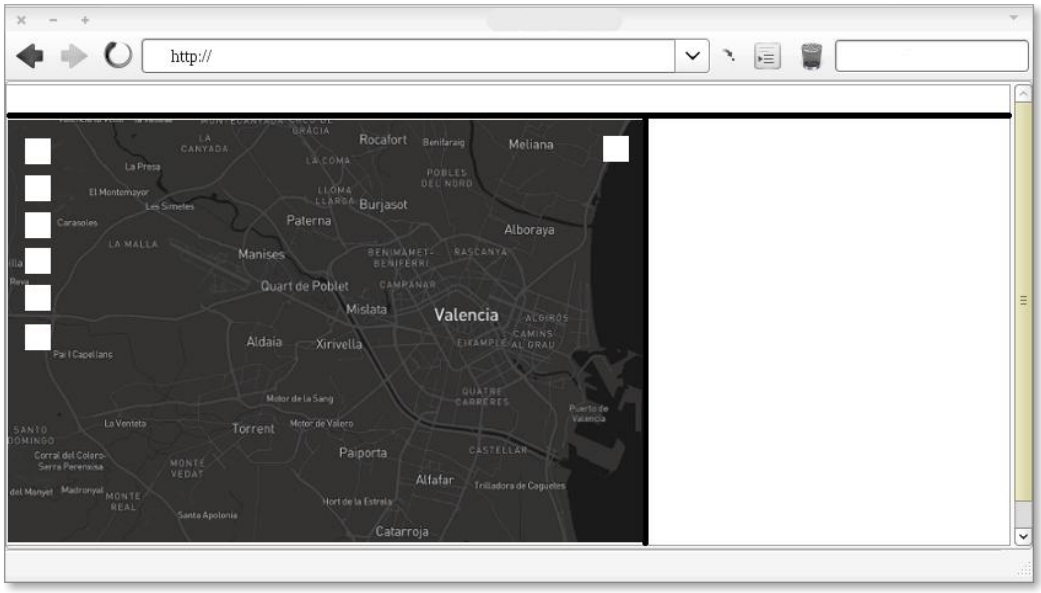

Ilustración 10: Dibujo de la ventana de inicio.

En la primera ventana que se encuentra el usuario, representada por la Ilustración 10, se puede distinguir un mapa donde se podrá observar el transcurso de la carrera y los distintos botones, representados por cuadrados blancos, que permitirán tanto subir un track o una clasificación como mostrar las estadísticas o el paso de la carrera y borrar el recorrido.

Ilustración 11: Dibujo de la ventana de subida de archivo.

Posteriormente a que el usuario pulse el botón de subir track o subir clasificación, se le abrirá una pantalla de búsqueda para encontrar el archivo y subirlo, así lo reproduce la ilustración 11.

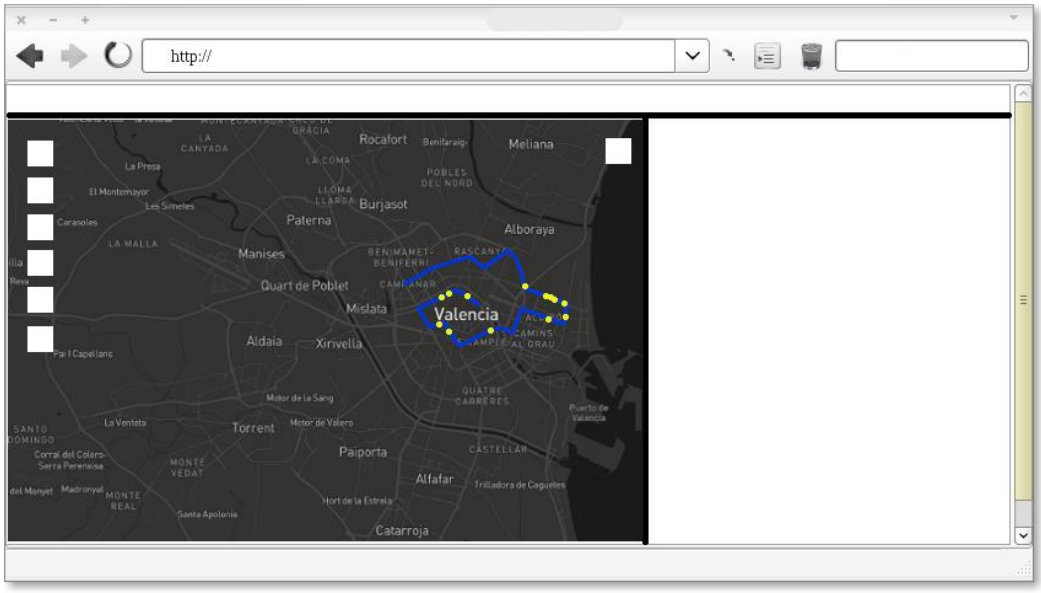

Ilustración 12: Dibujo de la ventana de muestra del recorrido y transcurso de la carrera.

Una vez estos archivos estén subidos, se trazará en el mapa el recorrido de la carrera en color azul. Si el usuario pulsa el botón de ver el paso de la carrera, todos los corredores encontrados en la clasificación serán representados por puntos amarillos dentro del recorrido, como se puede observar en la ilustración 12. Al elegir el botón de centrar en un usuario este aparecerá en color rojo.

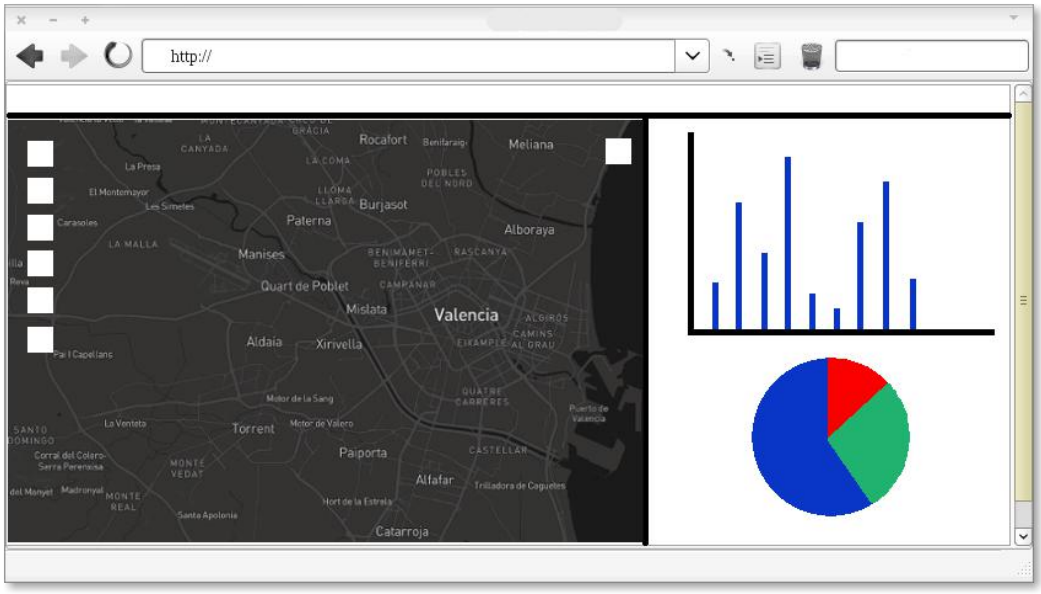

Ilustración 13: Dibujo de la ventana de muestra de estadísticas.

Si en cambio prefiere ver primero las estadísticas le será posible observarlas, siempre y cuando el track y la clasificación estén cargadas. En este caso las estadísticas tienen un lugar reservado dentro de la pantalla, en la parte derecha, todo ello representado por la ilustración 13.

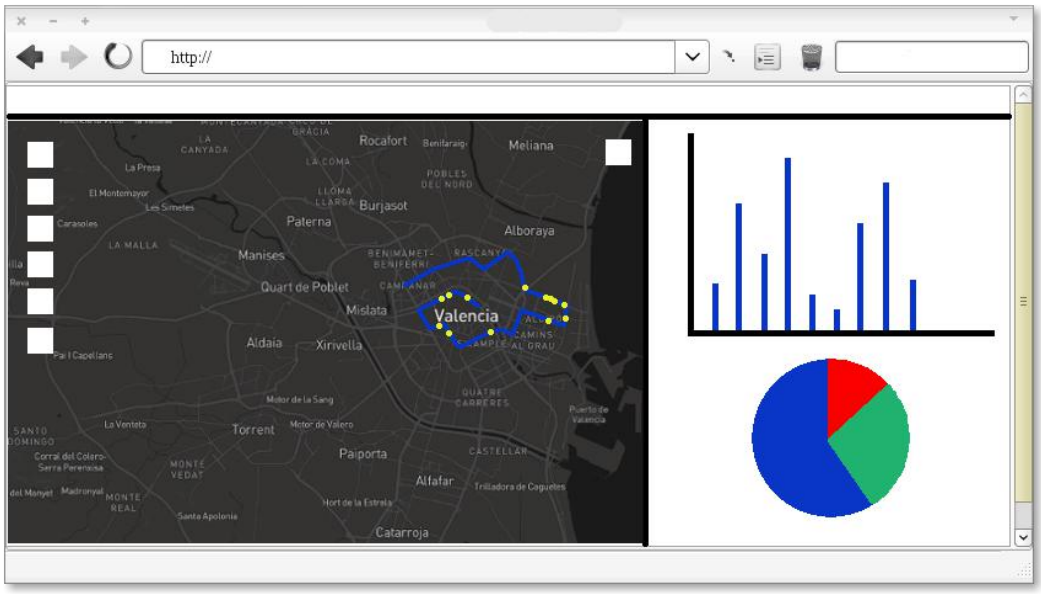

Ilustración 14: Dibujo de la ventana de muestra de estadísticas y muestra del transcurso de la carrera.

Finalmente el usuario puede observar las estadísticas de la carrera, o de un corredor en especial, y el transcurso de esta a la vez como se muestra en la ilustración 14.

4.3.2. Capa de persistencia

Para organizar la capa de persistencia se ha utilizado una base de datos NoSQL, que organiza los datos de forma distinta. En este caso se ha utilizado MongoDB que organiza los datos como objetos BSON, una representación binaria de los archivos JSON.

Para mejorar la compresión de la estructura se ha realizado un diagrama entidad-relación de cómo estarán relacionados los distintos objetos.

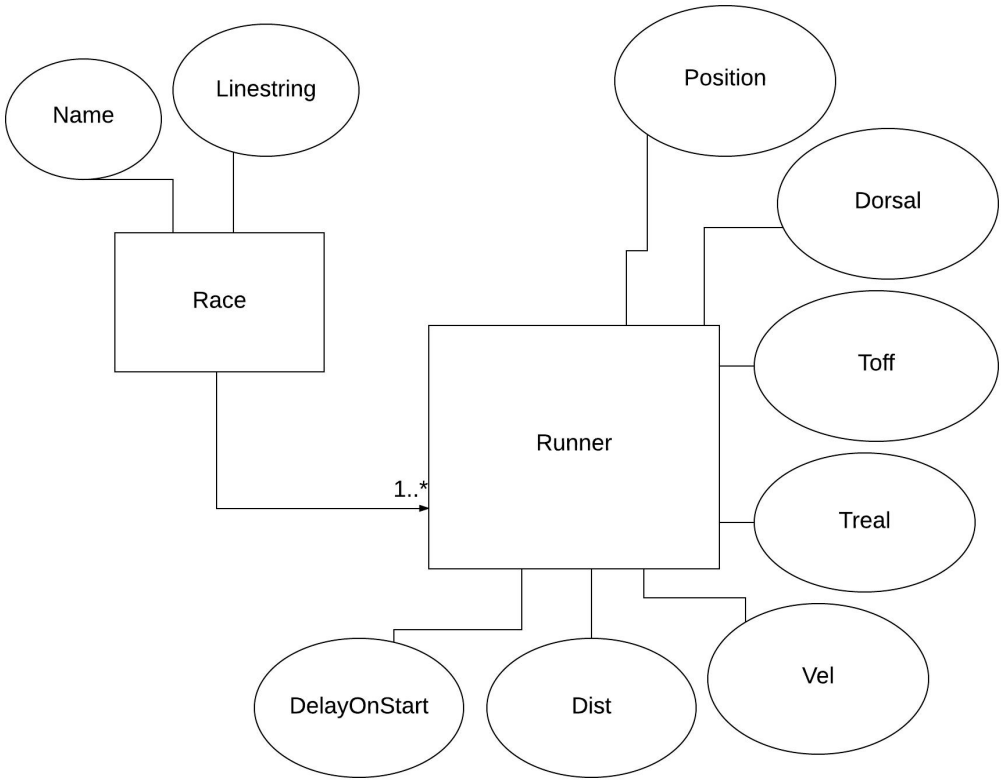

Ilustración 15: Diagrama de entidad-relación..

La información almacenada en la base de datos presentada en el diagrama se agrupa a continuación:

Race: Consiste en la carrera, sus atributos son los siguientes:

- Name: El nombre de la carrera.
- Linestring: Consiste en el recorrido de la carrera, este a su vez posee un tipo y un array de coordenadas.

Runner: Se trata del corredor de la carrera. Tiene los siguientes atributos:

- Position: La posición en la que se encuentra el corredor a la salida (es la misma para todos).
- Dorsal: El número del dorsal del corredor.
- Toff: El tiempo official del corredor en segundos.

- Treal: El tiempo real del corredor en segundos.
- Vel: La velocidad a la que ha corrido el corredor.
- DelayOnStart: La diferencia de tiempo entre el tiempo real y el oficial en segundos.
- dist: La distancia que lleva recorrida el corredor.

A su vez cada objeto que se guarda en MongoDB genera automáticamente un id propio a no ser que se le especifique a la hora insertarlo.

El diagrama mostrado se convierte en dos objetos JSON para introducirlos en la base de datos. A continuación se representan esos dos objetos.

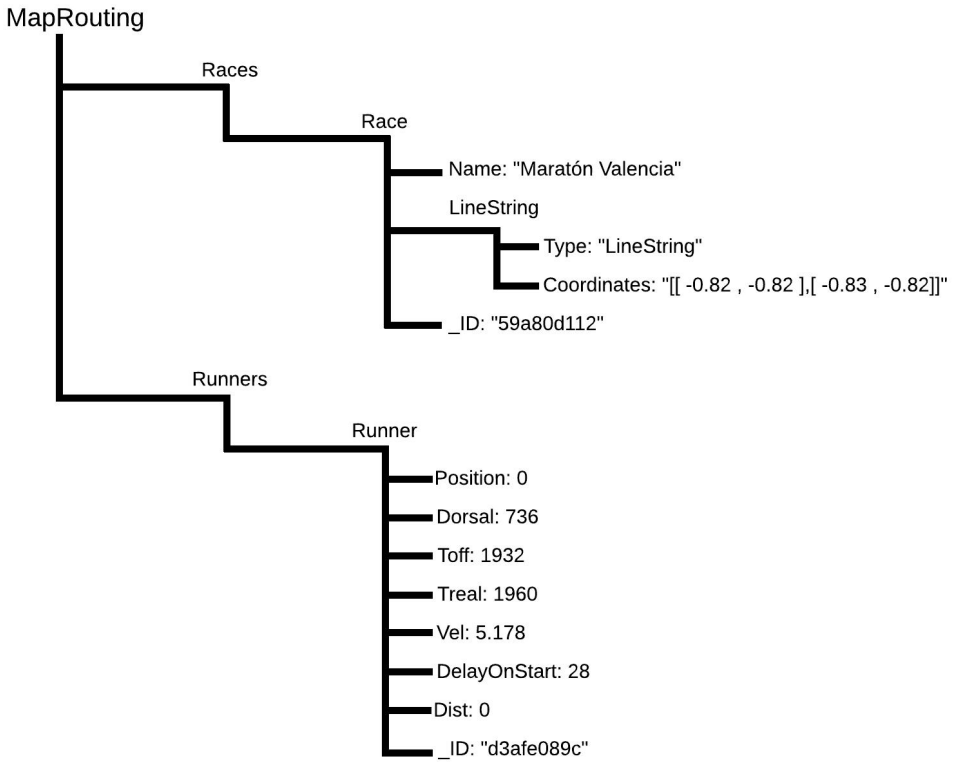

Ilustración 16: Diagrama de estructura de la base de datos.

Los nodos principales son races y runners, que coinciden con carreras y corredores. En las bases de datos NoSQL es necesario desnormalizar los datos y separarlos de manera que se pueda acceder a ellos eficazmente.

4.3.3. Capa de negocio

En el siguiente apartado se muestra el desarrollo de la lógica de la aplicación, los aspectos más relevantes se representan con diagramas de secuencia. Los diagramas que se han elaborado corresponden a los casos de uso vistos en el apartado 3.2.3.

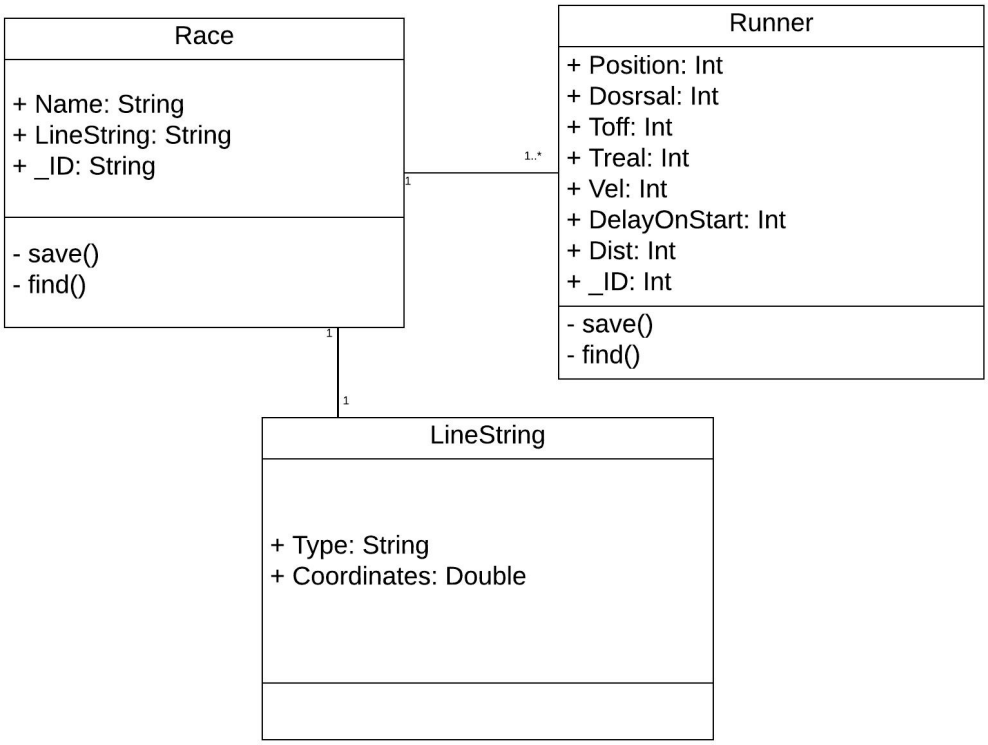

Ilustración 17: Diagrama de clases correspondiente a las clases que que intervienen en la capa de lógica y persistencia.

En el anterior diagrama se muestra la representación de las clases implicadas en la capa de lógica y persistencia. En el se puede observar la relación que existe entre la carrera, los corredores y el LineString, es decir, los puntos de coordenadas.

El diagrama muestra como una carrera puede estar relacionada con uno o varios corredores, en cambio un LineString, o recorrido, solo puede estar relacionado con una carrera. Un corredor estará siempre relacionado con una carrera ya que al depender de los dorsales no puede existir dos corredores con el mismo dorsal en distintas carreras.

La mayoría de los casos de uso dependen de otro anterior. Los casos de uso CU02 (Subir la clasificación) y CU03 (Borrar los datos de la carrera) depende del caso de uso CU01 (Subir el track de la carrera), y los casos de uso CU04 (Ver las estadísticas), CU05 (Iniciar el recorrido de la carrera) y CU06

(Detener el transcurso de la carrera) dependen del caso de uso CU02. Por esta razón se ha decidido realizar dos diagramas de secuencia, uno con los tres primeros casos de uso y otro con los otros tres, asumiendo que en el segundo diagrama se han realizado los casos de uso necesarios, es decir los casos de uso CU01 y CU02.

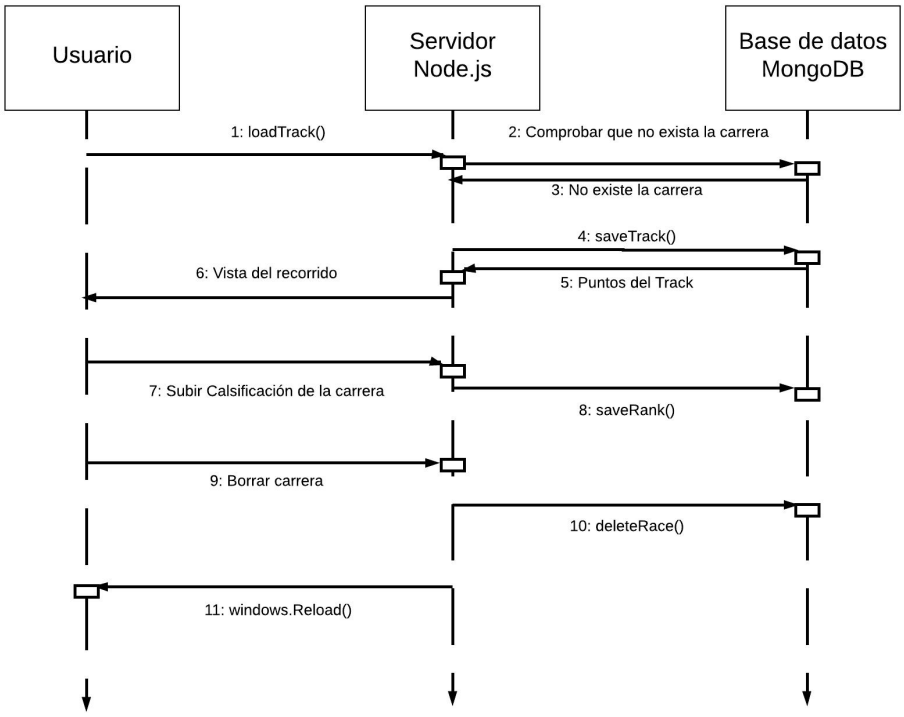

Ilustración 18: Diagrama de secuencia de los tres primeros casos de uso.

El diagrama anterior muestra la secuencia de los casos de uso CU01, CU02 y CU03. En el primer caso de uso la función loadTrack() permite subir el track de la carrera, antes de guardarlo en la base de datos el servidor comprueba que no exista ya en ella. Si no existe entonces la introduce, en caso contrario recupera el track que había almacenado y proporciona la vista del recorrido.

El caso de uso CU02 nos permite subir la clasificación de la carrera, el servidor recibe los datos de los corredores y los guarda en la base de datos con la función save().

Por último, el caso de uso CU03 lo que realiza es un borrado de los corredores y de la carrera y reinicia la ventana del navegador para que no se muestra ningún recorrido.

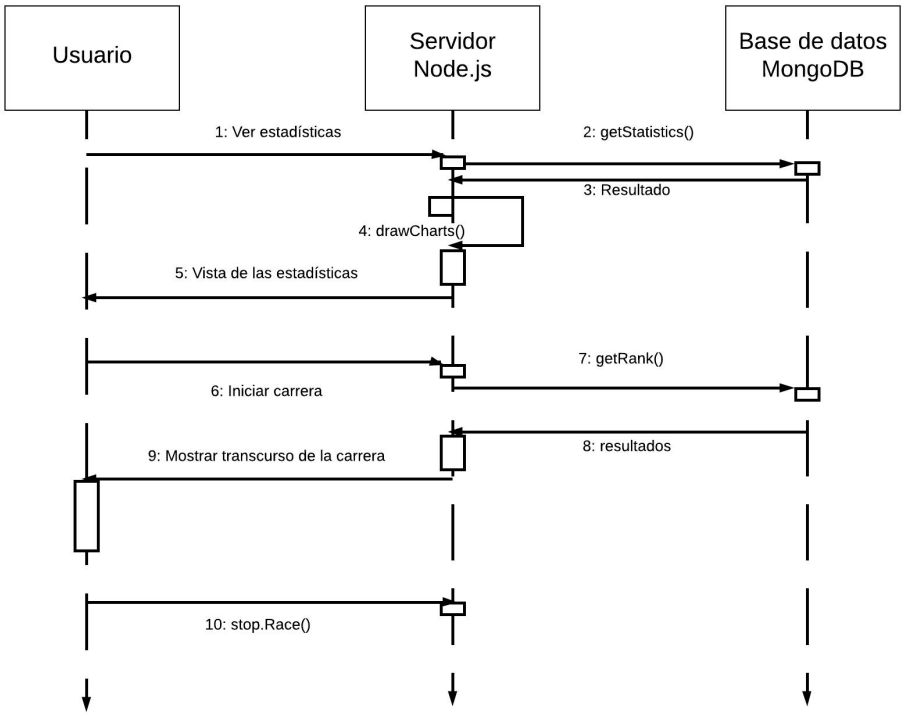

Ilustración 19: Diagrama de secuencia de los tres últimos casos de uso.

La ilustración 18 muestra el diagrama de secuencia de los casos de uso CU04, CU05 y CU06 que solo pueden producirse si se ha cargado el track y la clasificación. El caso de uso CU04 utiliza la función loadStats() para mostrar las estadísticas de la carrera, para ello el servidor solicita la información necesaria a la base de datos, realiza las operaciones necesarias y muestra los gráficos al usuario.

El segundo caso de uso que se representa es el CU05 que permite observar el transcurso de la carrera. La función simulate() se encarga de simular la carrera, el servidor pide la información de la clasificación a la base de datos y muestra al usuario el paso de los corredores por la carrera.

Finalmente, el caso de uso CU06 realiza un reinicio de la carrera a la espera de que el usuario vuelva a pulsar el botón de play, en este caso de uso no interviene ni el servidor ni la base de datos..

4.4. Conclusiones

En este capítulo se ha determinado el diseño de la aplicación. Primero se ha realizado un esquema conceptual para entender cómo iba a funcionar el sistema. Más tarde se ha desarrollado el diseño por capas ahondando en cada una de ellas.

En la capa de presentación se han mostrado los bocetos de cómo la aplicación se verá estéticamente. En la capa de persistencia se ha incluido un esquema entidad-relación y se ha modificado en relación a como la base de datos NoSQL guarda la información.

Para finalizar, en la capa de negocio se han mostrado los diagramas de secuencia que agrupan todos los casos de uso planteados. Una vez finalizado el diseño del sistema es el momento de comenzar la implementación de la aplicación.

5. Implementación, implantación y evaluación del sistema

5.1. Introducción

Después de haber decidido el diseño del sistema se procede a la presentación de su implementación.

En el apartado se describe con más detalle las tecnologías y herramientas utilizadas para el desarrollo, incluyendo capturas de pantalla de la aplicación y líneas de código relevantes.

5.2. Implementación

En esta sección se presentará la implementación de cada una de las capas vistas en el capítulo anterior. La aplicación se ha desarrollado con la idea de permitir a cualquier usuario hacer uso de ella desde un simple navegador web.

Para el desarrollo de la aplicación web se ha empleado el editor de texto Atom. Se ha desarrollado en un entorno Node.js con la ayuda del framework Express.js que nos proporciona una estructura web rápida flexible y minimalista para este tipo de aplicaciones. Para la capa de persistencia se hará uso del sistema de base de datos NoSQL MongoDB y para la visualización de las estadísticas se utilizará la herramienta de Google Charts.

El sistema se ha desarrolla utilizando el patrón de arquitectura software MVC (Modelo – Vista – Controlador), por lo que las diferentes acciones o funciones están separadas las unas de la otras. El Modelo se encarga de gestionar el acceso a la información con la cual el sistema opera. El Controlador envía peticiones al Modelo para la manipulación de la información, y finalmente, la Vista se ocupa de presentar la información al usuario en un formato adecuado para que interactúe.

A continuación se muestran unas capturas de pantalla que demuestran que se ha implementado el sistema en base al diseño presentado en el capítulo cuatro. Aparte de estas interfaces del sistema, existe código relevante dentro de la aplicación que conviene destacar y que a continuación de cada imagen se muestra y se explica.

Vista inicial de la aplicación.

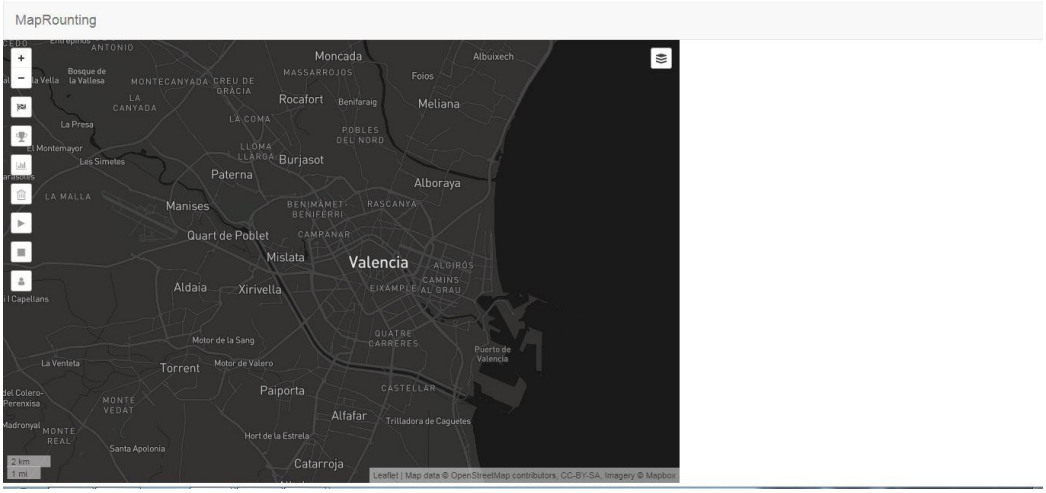

Ilustración 20: Captura de pantalla del sistema - Ventana de inicio.

```
var MAPBOX_TILES = 'https://api.tiles.mapbox.com/v4/{id}/{z}/{x}/{y}.png?access_t
var MAPBOX_ATTRIBUTION = 'Map data &copy; <a href="http://openstreetmap.org">Open
'<a href="http://creativecommons.org/licenses/by-sa/2.0/">CC-BY-SA</a>, ' +
'Imagery © <a href="http://mapbox.com">Mapbox</a>'
var mapboxSatellite = L.tileLayer(MAPBOX_TILES, {
  maxNativeZoom: 22,
  maxZoom: 22,
  attribution: MAPBOX_ATTRIBUTION,
  id: 'mapbox.satellite',
  token: 'pk.eyJ1IjoibXJlYm9sbG8iLCJhIjoiY2ozb2J5MHcyMDAxMzMycG5jdWRrb3B2YyJ9.s_G
})
var default1 = L.tileLayer('https://api.mapbox.com/styles/v1/mapbox/dark-v9/tiles
  attribution: 'Map data &copy; <a href="http://openstreetmap.org">OpenStreetMap<
  maxZoom: 18,
  id: 'gonzasanba',
  accessToken: 'pk.eyJ1IjoiZ29uemFzYW5iYSIsImEiOiJjaXZjb3BmN3cwMDZmMnlwNGFpNzJjcX
})
var map = L.map('map', {
//    center: [39.467706, -0.377766],
//    zoom: 13,
    layers: [mapboxSatellite, default1]
});
map.locate({setView: true, maxZoom: 18});
var baseMaps = {
    "Satelite": mapboxSatellite,
  "Inicio": default1
};
var LayersControl = L.control.layers(baseMaps).addTo(map);
L.control.scale().addTo(map);
```

Código 1: Función para cargar el mapa.

El código 1 muestra la inicialización del mapa, para poder disponer de un mapa era necesario un id y un accessToken. En este caso se crearon dos capas, es decir, dos mapas urbanos con características distintas. Esas dos capas se introdujeron en la variable *baseMaps* y la función *L.control.layers().addTo(map);* es la encargada de agregar esos dos mapas a la capa principal. Para cambiar de mapa solo es necesario acercar el cursor al botón de la parte superior derecha, que se muestra en la ilustración 19, y elegir cual de los dos se prefiere.

Subida de archivo por medio de HTML5 y JQuery.

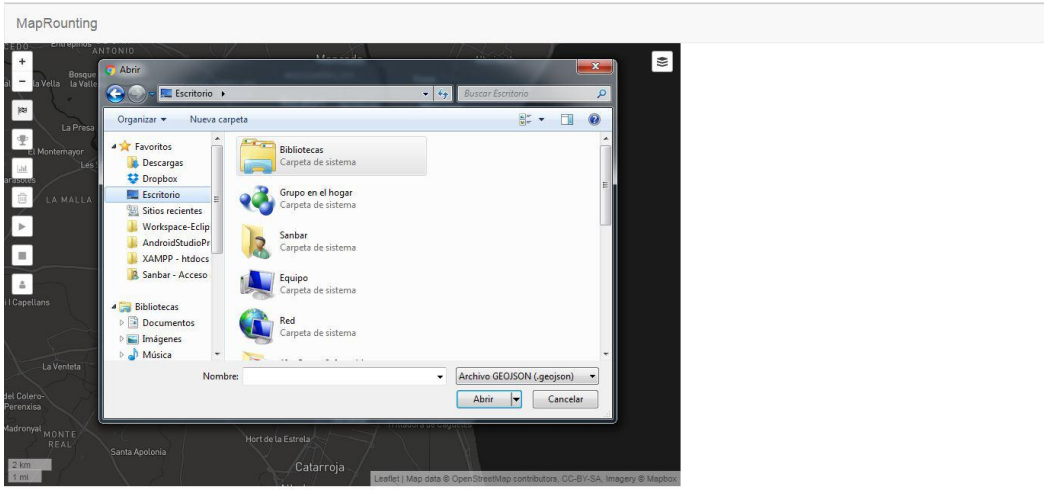

Ilustración 21: Captura de pantalla del sistema - Ventana de subir archivo.

```
//Cargar el recorrido
var clickOnLoadTrack = function () {
  var openFileDialog = document.getElementById("openFileDialog")
  openFileDialog.setAttribute('accept', '.geojson')
  openFileDialog.click()
  document.getElementById("openFileDialog").onchange = function () {
    var reader = new FileReader()
    reader.readAsText(this.files[0])
    reader.onload= function(){
      var jsonData = reader.result
      var jsonObject = JSON.parse(jsonData)
```

Código 2: Función para cargar el track.

```
//CARGA LA CARRERA
var clickOnLoadClassification = function () {
    var openFileDialog = document.getElementById("openFileDialog")
    openFileDialog.setAttribute('accept', '.tsv, .csv, .txt')
    openFileDialog.click()
    document.getElementById("openFileDialog").onchange = function () {
      var reader = new window.FileReader()
      reader.readAsText(this.files[0])
      reader.onload = function () {

        var tsvData = reader.result
        var tsvObject = d3.tsv.parse(tsvData)
```

Código 3: Función para cargar la clasificación.

Estas dos imágenes muestran el código necesario para subir archivos por medio de HMTL5 y JQuery. Primero se selecciona el input desde donde se va a subir el archivo, en nuestro caso el elemento con id "*openFileDialog*", después se precisa que tipos de archivos están permitido mediante el método *setAttribute*. Cada vez que se pulse el botón, la variable *reader* guardará el archivo subido, el método readAsText() se encargará de leer el contenido del texto, y una vez se haya leído (función *onload()*) se guardará el resultado de la lectura y se transformará el resultado a un objeto (función JSON.parse()). Para los dos casos, subir el track y subir la clasificación, los métodos son los mismo, cambiando algunas opciones por la diferencia de tipo de archivo.

Almacenamiento del Track de la carrera y de la clasificación de esta.

```
var Track = require('../models/geojson');

exports.trackload = function(req, res) {
  Track.find({name:req.body.info.name}, function(err, docs) {
    if (err){
      //No existe la carrera
      var race = new Track({
          name: req.body.info.name,
          linestring: {
              type: req.body.track.type,
              coordinates: req.body.track.coordinates
          }
      });
      race.save(function (err) {
        if (err) return handleError(err);
        else console.log("Guardado");
      });
      trackToPoints = req.body.track;
      res.send(trackToPoints);
    } else {
      //Existe la carrera
      trackToPoints = docs[0].linestring;
      res.send(trackToPoints);
    }
  });
}
```

Código 4: Función para guardar la carrera.

```
  //save runners
  var run = new Runner({
    position: runner.pos,
    dorsal: runner.dorsal,
    toff: runner.toff,
    treal: runner.treal,
    vel: runner.vel,
    delayOnStart: runner.delayOnStart,
    dist: runner.dist
  })
  run.save(function (err) {
    if (err) return handleError(err);
    else console.log("Guardado");
  });
})
```

Código 5: Función para guardar los corredores.

En los códigos anteriores se muestran los dos métodos save() que se encargan de guardar en la base de datos la carrera y los corredores. Previamente en el archivo principal de la aplicación se ha conectado con la base de datos por medio del método connect() que nos proporciona mongoose.

Estos dos métodos son llamados una vez se cargan los respectivos ficheros en el servidor, en el caso del track se realiza primero una búsqueda para comprobar si la carrera ya está subida mediante la función *Track.find()*, en caso de estar ya almacenada se recogen los puntos que forman la carrera y se

devuelven para su representación, en caso contrario se crea un objeto track y se guarda con la función *race.save()*. El caso del corredor es similar, por cada corredor encontrado en el archivo se crea un objeto runner y se almacena mediante la función *run.save()*.

Transcurso de la carrera.

Ilustración 22: Captura de pantalla del sistema - Ventana de transcurso de la carrera.

```
var Runner = require('../models/runner');
exports.simulateRace = function(req, res){
  runnerNumber = req.body.runner
  points = req.body.points
  var data=[];
  var aux=[];
  Runner.find({}, function(err, participants) {
    if (err) throw err;
    else{
      participants.forEach(function(o){
        aux.push((o.dorsal).toString());
      })
      var runnerPos = aux.indexOf(runnerNumber)
      console.log("posicion del corredor es: "+runnerPos)
      //calcular retardo
      var offset = Math.floor(totalTime / (numSteps - 1))
      //Llena un array con los puntos de coordenadas de cada corredor a la salida.
      var startCoord = Array(participants.length).fill(points[0])
      //se crea un layer con todos los corredores como puntos.
      data.push(startCoord)
      for (var t = offset; t <= totalTime; t += offset) {
        //calcula la distancia que llevan recorrida los corredores
        var distance = CalculateRunnerLocation(participants, offset)
        //calcula el número de step en el que está cada corredor.
        var posInCumDist = CalculateRunnerPositions(participants)
        //calcula el punto geográfico donde se encuentra cada corredor
        var runnerCoords = indexToCoord(posInCumDist, distance)
        data.push(runnerCoords)
      }
      //data.push(runnerPos);
      res.send({race: data, runner: runnerPos});
    }
  });
}
```

Código 6: Función para mostrar el transcurso de la carrera.

La función encargada de realizar el transcurso de la carrera se muestra en el código 6. Primero se recogen los datos que nos envían al servidor, los puntos de la carrera y en caso de que haya el dorsal del corredor que se desea remarcar.

A continuación se solicitan todos los corredores de la carrera, se recogen todos los dorsales de los corredores en orden de llegada y se comprueba la posición del corredor seleccionado, si el resultado es -1 significa que no se había seleccionado ningún corredor o que ese dorsal no existe dentro de la carrera.

Se calcula el tiempo que pasa entre cada capa y se rellena un array con la posición de todos los corredores en la salida, en cuyo caso será siempre el punto de inicio, este array representará la primera capa. Seguidamente para cada capa se calcula primero la distancia en la que se debe encontrar el corredor, con el simple calculo de la distancia recorrida, más el tiempo desde el inicio por la velocidad.

Después la función *CalculateRunnerPosition()* calcula, en base a la distancia que cada corredor lleva recorrida, la posición en la que se encuentra el corredor dentro del array de los puntos que forman el track.

A su vez la función *indexToCoord()*, con la posición de cada corredor dentro del array del track, calcula el punto geográfico en el que se encuentra el corredor. Una primera versión de esta función posicionaba a los corredores en los puntos según la distancia que habían recorrido. La segunda versión utiliza cálculos geométricos para posicionar los corredores en los puntos geográficos correctos.

Primero se calcula la pendiente entre los dos puntos, el punto donde se encuentra el corredor y el siguiente. Después se calcula el arcotangente de la pendiente, es decir el angulo de inclinación. Por último se pasa a la función *destinationPoint()* el angulo de inclinación, la distancia recorrida y el punto en el que se encuentra.

```
var destinationPoint = function (brng, dist, point) {
  const DIAMETER_OF_EARTH = 6378.137
  dist = dist / DIAMETER_OF_EARTH
  brng = toRad(brng)
  var lat1 = toRad(point.lat)
  var lon1 = toRad(point.lng)
  var lat2 = Math.asin(
    Math.sin(lat1) * Math.cos(dist) +
    Math.cos(lat1) * Math.sin(dist) * Math.cos(brng))
  var lon2 = lon1 + Math.atan2(
    Math.sin(brng) * Math.sin(dist) * Math.cos(lat1),
    Math.cos(dist) - Math.sin(lat1) * Math.sin(lat2))
  if (isNaN(lat2) || isNaN(lon2)) {
    return null
  }
  point.lat = toDeg(lat2);
  point.lng = toDeg(lon2);
  return point;
} // destinationPoint()
```

Código 7: Función para calcular el punto geográfico del corredor.

Esta función *destinationPoint()* calcula la latitud y longitud del puntos en el que se encuentra el corredor mediante cálculos geográficos gracias a a distancia y el angulo.

De todas las funciones que *simulate()* utiliza, *indexToCoord()* es la más importante ya que es la que define el punto geográfico en el que se encuentra el corredor. Esta función crea un array con todos los puntos de los corredores en cada capa a lo largo de la carrera, luego ese array bidimiensional será enviado a la vista que se encarga de montar las capas, añadirlas y eliminarlas de la interfaz.

Dibujar gráficos generales de la carrera.

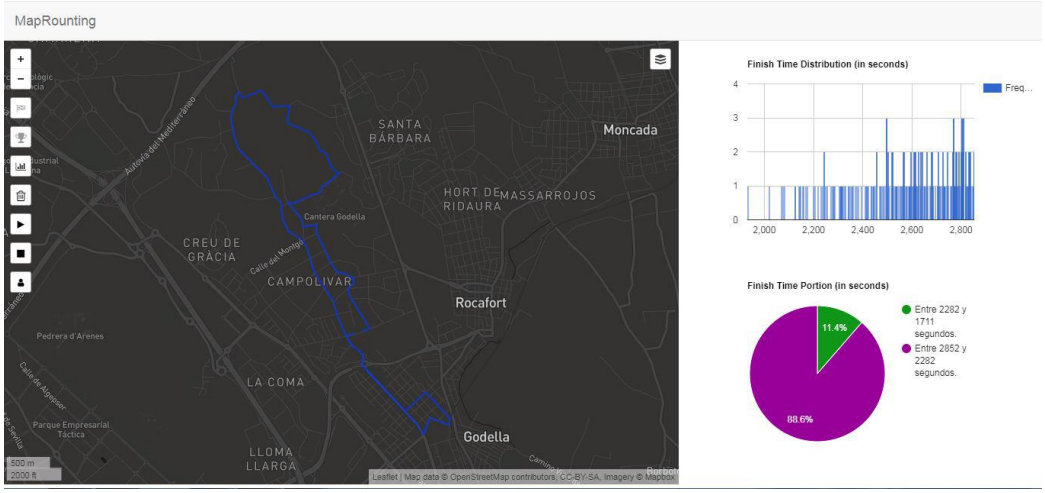

Ilustración 23: Captura de pantalla del sistema - Ventana de muestra de estadísticas.

```
google.charts.load('current', {packages: ['corechart', 'bar']});
google.charts.setOnLoadCallback(clickOnLoadstatistics);
var clickOnLoadstatistics = function () {
  $.ajax({
    type: 'POST',
    contentType: 'application/json',
    //dataType: "json",
    data: JSON.stringify({dorsal: runnerNumber}),
    url: 'http://localhost:3000/loadstats',
    success: function(data) {
      console.log(data.dorsal);
      if(data.dorsal < 0){
        estadisticasGenerales(data.times);
      }
      else {
        estadisticasIndividuales(data.dorsal, data.times);
      }
    }
  });
}
```

Código 8: Función para mostrar gráficos sobre las carreras.

En el caso de las estadísticas se pueden producir dos tipos de casos, que se desee ver estadísticas generales o individuales, todo dependerá de si se ha introducido un dorsal perteneciente a un corredor de la carrera.

```
var estadisticasGenerales = function(times){
  var data1 = new google.visualization.DataTable();
  data1.addColumn('number', 'Finish time');
  data1.addColumn('number', 'Frequency');
  var  count = [];
  times.forEach(function(i) {
    count[i] = (count[i]||0)+1;
  });
  times.forEach(function (object){
    data1.addRow([object, count[object]] );
  })
  var options1 = {
    'title':'Finish Time Distribution (in seconds)',
    'height':290
  };

  var chart1 = new google.visualization.ColumnChart(document.getElementById("sub_chart_div1"));
  chart1.draw(data1, options1);
```

Código 9: Función para mostrar gráficos generales sobre las carreras.

En caso de que no se haya introducido algún dorsal la función *estadisticasGenerales()* se encargará de recoger todos los datos de los corredores y realizar una serie de gráficos mediante el método *draw()* que nos facilita la herramienta Google Charts. Lo que se realiza es un gráfico de distribución de los tiempos y se asigna al elemento con id *sub_chart_div1*.

```
var estadisticasIndividuales = function(number, times){
  var data1 = new google.visualization.DataTable();
  data1.addColumn('string', 'Position');
  data1.addColumn('number', 'Finish time');
  var last = times.length;
  var pos = number+1;
  var bests = times.slice(0,3);
  bests.forEach(function(object){
    data1.addRow([(bests.indexOf(object)+1).toString()+"º" , object]);
  })
  data1.addRow(["Runner selected, position: "+pos, times[number]]);
  data1.addRow([(last).toString()+"º" , times.last()]);



  var options1 = {
    'title':'Finish Time Distribution (in seconds)',
    'height':290
  };
  var chart1 = new google.visualization.ColumnChart(document.getElementById("sub_chart_div1"));
  chart1.draw(data1, options1);
}
```

Código 10: Función para mostrar gráficos individuales sobre las carreras.

Si se ha introducido un dorsal perteneciente a algún corredor de la carrera, se mostrará un gráfico con la comparación de los tiempos entre el primero, segundo, tercero y último de la carrera. La función *estadisticasIndividuales()* recibe tanto los tiempos como el número de la posición del corredor, se extrae los tres primeros tiempos y el último tiempo del array, y mediante el método *draw()* crea el gráfico y lo representa en el div de id *sub_chart_div1*, como en la anterior ocasión.

Visualización de los gráficos y transcurso de la carrera.

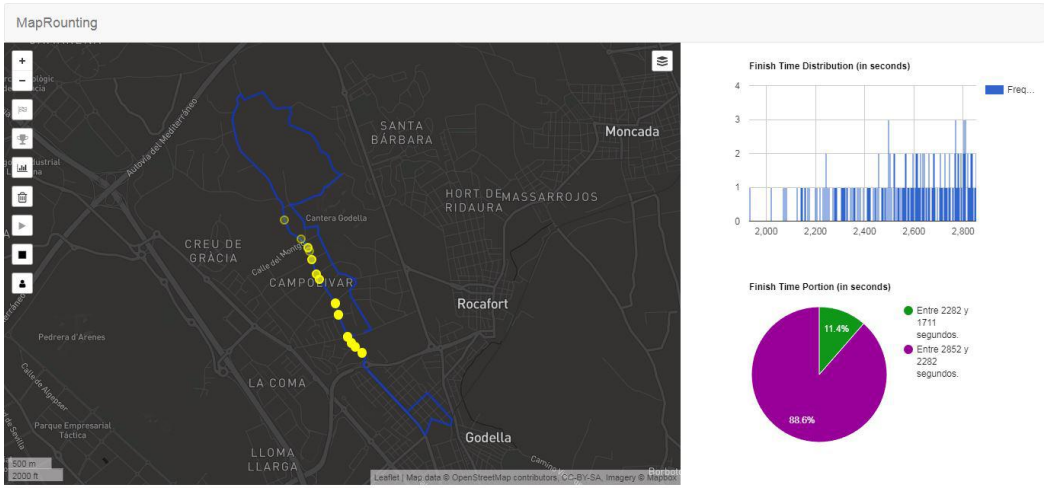

Ilustración 24: Captura de pantalla del sistema - Ventana de muestra de estadísticas y transcurso de la carrera.

Por último, recalcar que cabe la posibilidad de observar tanto las estadísticas como el transcurso de la carrera a la vez, pero que si en algún momento se inserta un dorsal perteneciente a algún corredor, las estadísticas y la visualización de la carrera corresponderán a la del corredor seleccionado.

5.3. Implantación

Al ser una aplicación web el usuario no está obligado a realizar ninguna instalación, la aplicación estará alojada en un servidor, por lo que lo único que necesitará será un navegador web.

5.4. Conclusiones

Para llevar a cabo la implementación del sistema se han utilizado varias herramientas, elegir las correctas ha sido muy importante ya que otras podrían haber presentado más problemas y haber dificultado la realización del proyecto.

Durante todo este capítulo se ha abordado con detalle la implementación del sistema, concluyendo con que se han cumplido con el diseño establecido en el capítulo cuatro y las funcionalidades establecidas en el capítulo tres.

6. Conclusiones

6.1. Trabajo realizado

El desarrollo del proyecto se ha planteado poco a poco, comenzando con un análisis, para continuar con la especificación de requisitos y como resultado se ha establecido un diseño que ha permitido realizar la implementación del sistema.

Comenzar con un análisis era necesario para conocer el entorno sobre el que se iba a basar nuestras aplicación. Se efectuó una búsqueda de sistema similares, se compararon entre sí para ver qué funcionalidades parecían interesantes y se eligieron las características básicas que iba a poseer el sistema.

Con todo esto se decidió concretar los requisitos específicos de la aplicación, con la ayuda de diagramas de casos de uso, que debíamos implementar.

Más tarde fue el turno del diseño, donde por medio de una especificación conceptual se determinó como iba a ser el funcionamiento del sistema. Después una especificación formal se encargaría de describir el funcionamiento de todas las capas de la aplicación.

Finalmente con el análisis, la especificación y el diseño ya decidido, llegó la hora de la implementación del sistema. Se especificaron unos objetivos al inicio del documento y se han cumplido con éxito.

6.2. Dificultades y soluciones

Aunque ya se había visto algo de Node.js durante la carrera, el conocimiento desde el que partía era bastante básico, pero con la ayuda de la documentación de Node.js y el framework Express.js me fue sencillo aprender más sobre el funcionamiento de este tipo de aplicaciones.

A la hora de buscar información de sistemas similares al que se deseaba desarrollar también encontré alguna dificultad, pero unas cuantas búsquedas minuciosas me permitieron encontrar suficiente información.

6.3. Aportaciones

La principal aportación es la aplicación web desarrollada que cubre el objetivo principal del proyecto. Esta aplicación permite el análisis de las carreras a pie, lo que se puede entender como una herramienta de ayuda tanto como para corredores como para organizadores de carreras. Supone un progreso en lo que a sistemas de análisis de carreras se refiere, ya que la mayoría de los sistemas descritos en el capítulo dos no poseían la característica principal de la aplicación.

6.4. Trabajo futuro

El sistema puede mejorar con ciertas funcionalidades que se deberían implementar en un futuro.

En primer lugar, la necesidad de una autenticación antes de comenzar a realizar alguna acción posible, para ello un framework que nos facilita el trabajo sería Passport y que nos permitiría entrar con la cuenta de twitter, facebook o el correo.

En segundo lugar, y centrándonos más en las funcionalidades de la aplicación, un servicio a implementar sería la posibilidad de ver el paso por el recorrido de un grupo específico de personas y sus estadísticas.

Por último la implementación de una aplicación móvil también sería posible, aunque la funcionalidades de esta no serían las mismas que las de la aplicación web.

7. Bibliografía

Google Developers. (2017). Charts | Google Developers. [online] Disponible en: https://developers.google.com/chart/ [Acceso 30 julio 2017].

Automattic. (2016). Mongoose. [online] Disponible en: http://mongoosejs.com/ [Acceso 20 julio 2017]

Fundación Node.js. (2017). Express.js. [online] Disponible en: http://expressjs.com/es/ [Acceso 5 julio 2017]

MongoDB, Inc. (2016). MongoDB [online] Disponible en: https://www.mongodb.com/ [Acceso 14 julio 2017]

Node.js Developers. (2017). Node.js. [online] Disponible en: https://nodejs.org/en/ [Acceso 28 junio 2017]