**Universidad Internacional de La Rioja (UNIR)**

## Escuela de Ingeniería

**Máster universitario en Dirección e Ingeniería de Sitios Web**

**RAMA PROFESIONAL**

# Aplicación web de carreras deportivas en España

**Trabajo Fin de Máster**

**presentado por:** Ruiz Romero, Francisco

**Director/a:** Boubeta Puig, Juan

Ciudad: Madrid
Fecha: 18/09/2018

# Resumen

Hoy en día, Internet es una herramienta imprescindible para consultar información que desconocemos. En numerosas ocasiones, los deportistas se encuentran con el problema de querer apuntarse a un evento deportivo, pero no saber cuáles existen en una fecha y lugar concreto, ni buscando por Internet.

A raíz de esta necesidad, surge este Trabajo Fin de Máster con la finalidad de crear una aplicación web con un listado unificado de carreras deportivas en España donde los usuarios puedan encontrar su carrera deseada empleando diversos filtros (deporte, lugar o fecha). Además, se facilita la organización de carreras a través de un calendario, su integración con redes sociales, y el envío de alertas personalizadas a los aficionados interesados.

La aplicación se ha desarrollado tanto con tecnologías para la computación en el cliente (HTML, CSS3, JS, jQuery, Bootstrap y AJAX), como para la computación en el servidor (NodeJS). Los resultados demuestran que se trata de una aplicación novedosa totalmente funcional, *responsive,* internacionalizada, usable y accesible

# Abstract

Nowadays, the Internet has become an essential tool and key source of knowledge that may provide us information we are not familiar to. Sportsmenn have been encountered the problems of being willing to sign up for a sporting event and not even know how many will take place in a specific date and location.

As a result of this need, this Master's Thesis has been developed in order to create a web application containing an only and unified list of every Spanish race. Users will be able to find out their desired run by using various filters in it, such as a concrete sport, time and place. Furthermore, it facilitates the organization of races through a calendar, its integration with social networks and the sending of personalized alarms to interested fans.

The application has been developed with both technologies for computing in the client (HTML5, CSS3, jQuery, Bootstrap and AJAX), as well as for computing in the server side (NodeJS). The results show that is an innovative, fully functional, internationalized, responsive, usable and approachable application.

# Índice

# Índice de figuras

# Índice de tablas

# Capítulo 1: Introducción

En este capítulo se introduce brevemente la motivación y los antecedentes de este Trabajo Fin de Máster (TFM) y se detalla la estructura del documento.

## 1.1 Antecedentes

Actualmente en España, no existe ninguna página web o aplicación que contenga un listado con todas las carreras deportivas en España. Esto conlleva a que cualquier aficionado al deporte tenga que realizar, en numerosas ocasiones, una búsqueda muy larga para encontrar, por ejemplo, un triatlón en Alicante. Aunque existen algunas webs con el listado de una región o de un circuito de carreras, ¿no sería más adecuado tener una web que englobe toda esa información y que permita filtrar las carreras por distancia, tipo de deporte, localidad y comunidad autónoma, entre otros?

Por ejemplo, en la provincia de Ciudad Real, existe una web de carreras populares (www.carrerasciudadreal.es) que contiene información de todas las carreras de ese circuito, pero a su vez existen muchas otras que organizan los propios ayuntamientos o empresas.

Normalmente, el aficionado acaba conociendo todas las carreras que tienen lugar en su provincia. Sin embargo, cuando se desplaza a otra ciudad como, por ejemplo, Santander, no tiene a su alcance información sobre las carreras que se celebrarán en esa ciudad durante el fin de semana.

Para dar respuesta a esta necesidad, en este TFM se propone el desarrollo de una aplicación web que permita gestionar las carreras deportivas que se celebren en todo el territorio español. Entre las funcionalidades principales de la web, destacan la organización de los eventos de las distintas carreras a través de un calendario, su integración con redes sociales, y el envío de alertas personalizadas que permitan a cualquier aficionado estar informado en cada instante de tiempo de los cambios que se produzcan en las carreras organizadas.

Todas las funcionalidades se pondrán a disposición del usuario a través de la aplicación web, que deberá poseer una interfaz amigable, intuitiva, usable, *responsive*[1] e internacionalizada (castellano e inglés), para mejorar la experiencia de usuario. La aplicación web contará, tanto con tecnologías para la computación en el cliente (*HTML, CSS3, JS, jQuery, Bootstrap* y *AJAX*), como para la computación en el servidor (*NodeJS*). Por último, para la persistencia de los datos se utilizará *MongoDB*.

## 1.2 Estructura del documento

El presente TFM está compuesto por 7 capítulos y 5 anexos. A continuación, se describe el contenido de cada uno de ellos.

- **Introducción.** Capítulo que se encarga de explicar la motivación y el origen de este TFM, cómo se va a abordar el problema y se detalla la estructura de la memoria.
- **Contexto y estado del arte.** Capítulo donde se estudia a fondo el dominio de la aplicación y se realizan comparaciones con soluciones ya existentes para justificar la realización del trabajo.
- **Objetivos concretos y metodología de trabajo.** Capítulo que define cuáles son los objetivos que se desean alcanzar con el TFM, tanto el general como otros más específicos. También se explica cuál es la metodología de trabajo que se emplea para intentar abordarlo con éxito.
- **Desarrollo del proyecto.** Capítulo donde se explica las fases necesarias para conseguir construir la aplicación completa. También se indica todas las herramientas utilizadas para el desarrollo, tanto *hardware* como *software*.
- **Resultados.** Capítulo que recoge el análisis del resultado del TFM, comprobando si se han cumplido todos los objetivos que se pretendían alcanzar y muestra la aplicación web desarrollada.
- **Conclusiones y trabajo futuro.** Capítulo donde se hace una reflexión sobre el trabajo y se indican las conclusiones obtenidas con su realización. Además, se enumeran un conjunto de propuestas que podrían complementar a la aplicación y la dotarían de más funcionalidad.
- **Referencias bibliográficas.** Capítulo que muestra un listado con todas las referencias citadas durante la memoria del TFM y la bibliografía consultada.

[1] El diseño web *responsive* es la filosofía de diseño y desarrollo que pretende adaptar el aspecto de las páginas web al dispositivo desde el cual se esté visualizando.

Posteriormente, se incluyen diferentes anexos que ayudan a aclarar y ampliar información sobre ciertos temas tratados en los capítulos anteriores.

- **Anexo A: Motor de plantillas Jade.** Anexo que detalla cómo funciona el motor de plantillas *Jade* que permite crear el *frontend* de la aplicación.
- **Anexo B: Internalización i18next.** Anexo donde se explica el funcionamiento de la librería *i18next* que permite internacionalizar la aplicación.
- **Anexo C: API de Google Maps y Geocode.** Anexo que detalla en mayor profundidad cómo se usa la API de Google Maps y Geocode para mostrar las carreras en un mapa.
- **Anexo D: Importación masiva de datos.** Anexo donde se explica cómo se produce la importación múltiple de carreras a la aplicación.
- **Anexo E: Exportación del calendario**. Anexo que explica de qué manera se exporta el calendario para, posteriormente, poder importarlo en aplicaciones de tipo *Calendar*.

# Capítulo 2: Contexto y Estado del arte

Este capítulo tiene como objetivo presentar un estudio detallado sobre el problema que se pretende solucionar con este TFM, realizando comparaciones con soluciones ya existentes con el fin de justificar su realización.

Para dicho estudio, se ha realizado una búsqueda de aplicaciones que comparten ciertas características junto a una encuesta para conocer la opinión de los deportistas acerca de la aplicación propuesta (a grandes rasgos), si comparten la motivación de este trabajo, así como si conocen alguna aplicación o página similar.

## 2.1 Motivación

Como ya se ha comentado en el capítulo anterior, este trabajo surge de la necesidad de tener una aplicación web con un listado o calendario unificado de carreras deportivas en España, puesto que, aunque existen muchas otras aplicaciones que contienen información acerca de determinadas carreras, no existe ninguna que sea completa. Por este motivo, el usuario debe acceder a varias páginas para poder encontrar alguna carrera que le interese, y esa búsqueda múltiple no le asegura encontrar una prueba de las características deseadas.

## 2.2 Encuesta de opinión

Con el objetivo de justificar la motivación de este trabajo, se ha realizado una encuesta para conocer la opinión de los deportistas acerca del problema explicado. La encuesta contiene diversas preguntas acerca de cómo encuentran una carrera en la que posteriormente participan, si conocen alguna aplicación similar a la propuesta, ventajas y desventajas de la propuesta y opinión acerca de posibles funcionalidades extras del trabajo. La encuesta se ha distribuido tanto por grupos de *Whatsapp* sobre deporte, como en *Facebook*. El número de personas que han contestado a la encuesta es de 63. Las preguntas y resultados de la encuesta son los siguientes:

- **¿Cómo has encontrado las carreras en las que luego has participado?** En esta pregunta se proporcionan diversas opciones donde el encuestado puede marcar más de una de ellas e incluso indicar otra no detallada. Como se puede observar en la Figura 1, los deportistas suelen apuntarse a carreras por recomendación de un conocido (72,6%), por conocer la carrera a través de redes sociales o páginas deportivas (61,3%) o conocerla de otros años (53,2%).

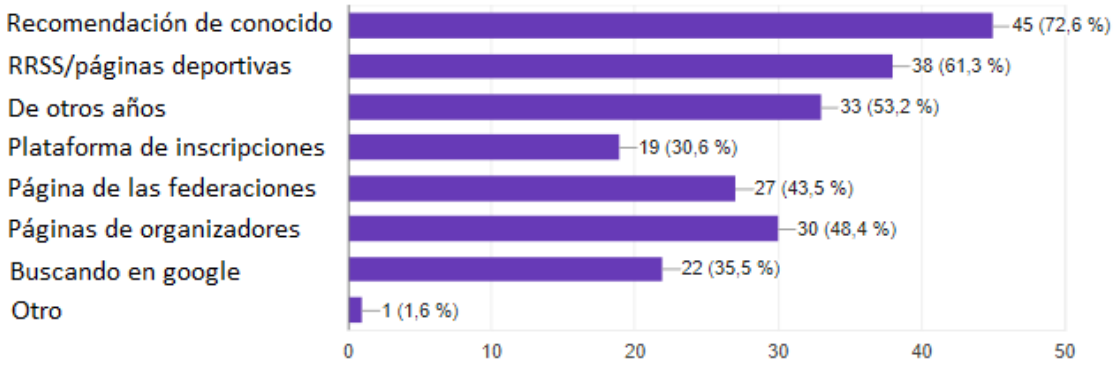


*Figura 1. Encuesta: resultado de la primera pregunta*

- **Si te fueras de vacaciones a un lugar y te gustaría comprobar si te coincide con algún evento deportivo para ver si te puedes apuntar, ¿Cómo lo buscarías?** Al igual que en la pregunta anterior, se proporcionan diversas opciones pudiendo seleccionar varias de ellas. En el resultado obtenido (véase Figura 2) destaca que el 82,5% de los encuestados buscarían en *Google* para encontrar la carrera, puesto que es la opción más sencilla y rápida.

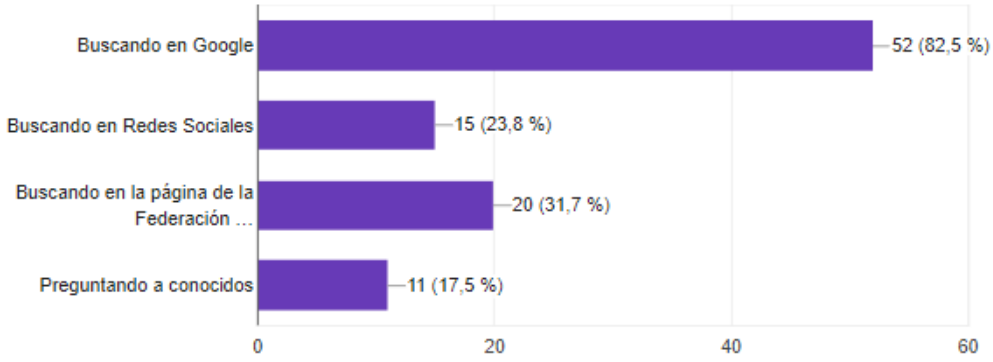


*Figura 2. Encuesta: resultado de la segunda pregunta*

- **Y si existiese una página web con un calendario unificado con todas las carreras deportivas del territorio español, con posibilidad de filtrar por tipo de deporte, distancia, fecha, lugar... ¿La usarías?** En esta pregunta se proporcionan solo tres respuestas: Si / No / Tal vez. El resultado de esta pregunta (véase Figura 3) confirma la necesidad de tener una aplicación con un calendario unificado, puesto que el 90,5% de los encuestados la utilizarían y el 9,5% restante quizás.

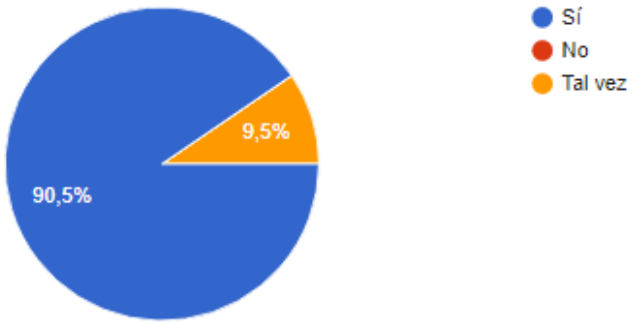


*Figura 3. Encuesta: resultado de la tercera pregunta*

- **En caso de que conozcas una web con dichas características, indícala aquí.** En esta pregunta de texto abierto, se han recogido cinco páginas distintas: atletasdelsol.com, carreraspopulares.com, correrenlarioja.com, carreraspormontana.com y runedia.mundodeportivo.com, siendo esta última la más mencionada.
- **¿Qué ventaja le ves a la web propuesta?** En esta pregunta se proporcionan dos opciones y la posibilidad de introducir texto libre. La principal ventaja que los encuestados ven a la aplicación (véase Figura 4) es el ahorro de tiempo al solo tener que mirar una página (86,7%), mientras que el 46,7% indican que, al ser un calendario completo, encontrarían seguro la carrera deseada. Un deportista ha contestado que otra ventaja sería prever fechas de eventos en base a años anteriores.

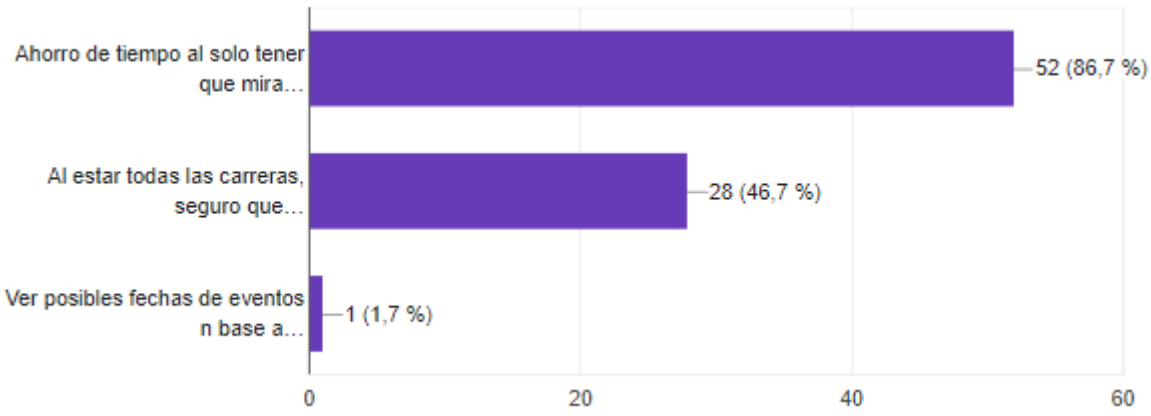


*Figura 4. Encuesta: resultado de la quinta pregunta*

- **¿Le ves alguna desventaja o "pero"?** En esta pregunta de texto libre, los encuestados se centran sobre todo en la dificultad de mantener un calendario completo actualizado.

- **¿Te gustaría que estuviera integrada con las redes sociales para poder indicar a qué carreras vas a asistir o invitar a tus amigos?** En esta pregunta se proporcionan solo tres respuestas: Si / No / Tal vez. Ante esta pregunta, aproximadamente dos tercios de los encuestados han respondido que sí les gustaría compartirlo por redes sociales, mientras que el 20,6% quizás y el 14,3% no les gustaría (véase Figura 5).

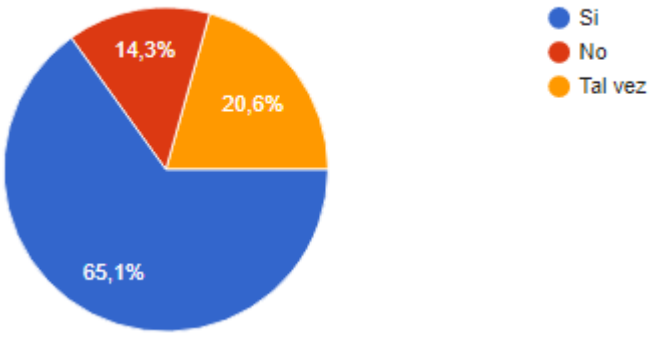


*Figura 5. Encuesta: resultado de la séptima pregunta*

- **¿Te gustaría recibir notificaciones de una carrera a la que te has apuntado si se actualizase su información, se cancelase o aplazase?** En esta pregunta se vuelven a proporcionan solo las tres respuestas anteriores. En los resultados de esta pregunta (véase Figura 6), se puede comprobar que la funcionalidad de notificar al usuario ante ciertos eventos que ocurran en una carrera donde se encuentran apuntados, es fundamental y necesario, puesto que el 93,7% de los encuestados han respondido afirmativamente.

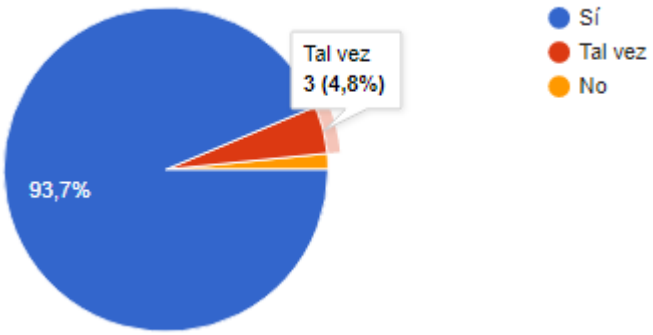


*Figura 6. Encuesta: resultado de la octava pregunta*

- **¿Te gustaría poder crear una alerta que te avisase de carreras que coincidan con alguna búsqueda realizada? (Según fecha, tipo, lugar...)** Al igual que en las dos preguntas anteriores, se proporcionan solo las respuestas de: Si / No / Tal vez. De la misma manera, las alertas personalizadas ante búsquedas es otra funcionalidad fundamental para la aplicación, habiendo contestado positivamente el 82,5% de los encuestados (véase Figura 7).

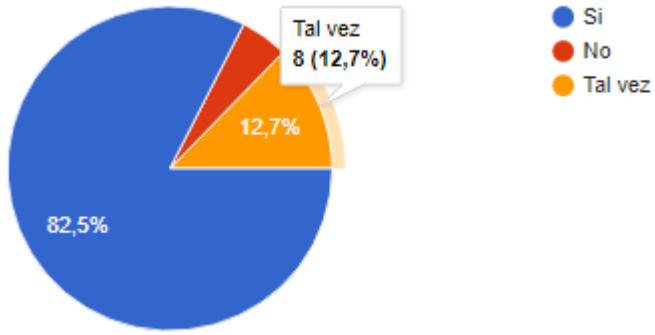


*Figura 7. Encuesta: resultado de la novena pregunta*

- **Recomendaciones y mejoras.** Para terminar, en esta pregunta de texto libre se da la posibilidad al encuestado de realizar recomendaciones y proponer mejoras para conseguir una aplicación mucho más completa. Las respuestas más significativas indican la posibilidad de introducir un método de valoración (similar a las estrellas de *Amazon)* o comentarios, alertas cuando las carreras abran inscripciones, filtros por precio de inscripción, realizarlo con una mentalidad global para muchos países o como interés económico para la plataforma, permitir a los organizadores patrocinar sus carreras para que sean más visibles en la plataforma.

## 2.3 Comparativa

A continuación, se realiza un estudio y una comparación entre las aplicaciones ya existentes y la propuesta en el trabajo, incluyendo algunas de las páginas recogidas en la cuarta pregunta de la encuesta. El conjunto de aplicaciones es el siguiente:

- *Sportmaniacs* [1]. Esta página es una plataforma de inscripciones que permite a los deportistas apuntarse a las carreras que ellos gestionan. Además, una vez concluidas, incluyen los resultados de las carreras. Su interfaz es muy amigable y *responsive* y añaden un filtro para buscar carreras por nombre o provincia. La aplicación se encuentra internacionalizada en 7 idiomas: castellano, catalán, inglés, francés, portugués, polaco y ruso.

- *RockTheSport* [2]. Al igual que en la anterior, solo se incluyen las carreras que ellos mismos gestionan. Permiten filtrar también por deporte y por distancia. Incluyen la funcionalidad de añadir eventos a favoritos y añaden una sección para organizadores que les permite gestionar sus eventos siempre que hayan contratado su servicio. Al contrario que la anterior, no permite cambiar de idioma su interfaz.
- *Atletasdelsol* [3]. En este caso, este sitio web está dedicado al atletismo y las carreras de montaña y cuenta con diversos apartados como consejos, diarios deportivos, resultados ordenados año a año, organismos y organizaciones relacionados y un calendario con un gran número de carreras ordenadas por distancia. La página no permite cambiar el idioma y, por tanto, no se encuentra internacionalizada.
- *Carreraspopulares* [4]. Esta página incluye un calendario nacional de carreras populares y de *running* al que se pueden aplicar diversos filtros por comunidad autónoma y distancia. También permite realizar la inscripción a diversos eventos. Cuenta con una sección específica para que los organizadores puedan dar de alta sus carreras deportivas. Además, incluye diversas utilidades para los deportistas, como una calculadora de ritmos o un foro. Por último, la principal desventaja de esta página consiste en que solo es para carreras de *running*, no incluye otros deportes como el triatlón.
- *Correrenlarioja* [5]. Al acceder a esta página, se puede observar que se centra en el *running* y cuenta con diversos apartados como noticias, rutas o consejos de salud. Además, cuenta con un calendario de carreras que permite exportarlo a aplicaciones de tipo *Calendar*. La principal desventaja es que sólo es de la comunidad autónoma de La Rioja y el calendario no cuenta con diferentes filtros para encontrar la carrera deseada.
- *Runedia* [6]. Es, probablemente, la página más completa de todas las anteriores por su antigüedad, número de carreras y usuarios. Cuenta con un calendario deportivo que permite aplicar diversos filtros cómo distancia, deporte, país o provincia. También permite a los organizadores registrar sus carreras y, a los usuarios, gestionar su calendario. Por último, permite cambiar el idioma de la aplicación entre castellano, catalán, francés e inglés.

En la Tabla 1 se muestra visualmente una comparativa de todas las aplicaciones detalladas anteriormente con la propuesta en este TFM, según diversas características.

| | Sportmaniacs | RocktheSport | Atletasdelsol | Carreraspopulares | Correrenlarioja | Runedia | DiarySport |
|---|---|---|---|---|---|---|---|
| Calendario | ✗ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Aplicar filtros | ✓ | ✓ | ✗ | ✓ | ✗ | ✓ | ✓ |
| Realizar inscripción | ✓ | ✓ | ✗ | ✓ | ✗ | ✓ | ✗ |
| Guardar carreras | ✗ | ✓ | ✗ | ✓ | ✗ | ✓ | ✓ |
| Gestionar carreras | ✗ | ✓ | ✗ | ✓ | ✗ | ✓ | ✓ |
| Crear alarmas | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ |
| Resultados | ✓ | ✓ | ✓ | ✓ | ✗ | ✓ | ✗ |
| Exportar calendario | ✗ | ✗ | ✗ | ✗ | ✓ | ✗ | ✓ |
| Compartir en RRSS | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ |
| *Responsive* | ✓ | ✓ | ✗ | ✓ | ✓ | ✓ | ✓ |
| Internacionalizada | ✓ | ✓ | ✗ | ✗ | ✗ | ✓ | ✓ |

*Tabla 1. Comparativa de funcionalidades entre aplicaciones*

# Capítulo 3: Objetivos y Metodología

Una vez introducido este TFM, en este capítulo se detallan los objetivos, teóricos y prácticos, que se pretenden alcanzar con su realización. Además, se especifica cuál va a ser la metodología de trabajo a seguir para realizar el proyecto y las fases en las que se divide. Por último, se detallan todo el marco tecnológico empleado en la realización de este trabajo.

## 3.1 Objetivos

El objetivo principal de este TFM consiste en el desarrollo de una **aplicación web que permita gestionar un listado unificado y global de carreras deportivas en toda España**, para facilitar al deportista que encuentre su próximo reto deportivo.

Para alcanzar este objetivo general, es necesario abordar una serie de objetivos, teóricos y prácticos que se exponen en los siguientes apartados.

### 3.1.1 Objetivos teóricos

Los objetivos teóricos que se pretenden conseguir mediante la realización de este proyecto son los siguientes:

- Realizar un estudio del estado del arte y una comparativa de soluciones existentes a la propuesta.
- Estudiar el patrón de arquitectura *software* Modelo-Vista-Controlador (MVC) para posteriormente aplicarlo en el desarrollo.
- Estudiar las líneas guía de usabilidad y accesibilidad para plataformas web [7], [8]
- Realizar una familiarización con las diversas tecnologías utilizadas para este proyecto: *NodeJS, MongoDB* y *Bootstrap*.

### 3.1.2 Objetivos prácticos

De igual forma, los objetivos prácticos necesarios para el correcto desarrollo de este proyecto son los siguientes:

- Desarrollar una aplicación totalmente funcional y libre de fallos para la gestión de carreras deportivas.
- Incluir la gestión de cuentas de usuario (registro, inicio de sesión…)
- Permitir importar de forma masiva o manual los eventos deportivos.
- Integrar la herramienta con las redes sociales más populares (*Facebook*, *Twitter*…)
- Crear una funcionalidad de alertas personalizadas para el usuario.
- Implementar de forma usable y accesible la herramienta para mejorar la experiencia del usuario y facilitar el uso de esta.
- Internacionalizar la aplicación permitiendo así al usuario elegir el lenguaje (castellano o inglés).
- Dotar a la herramienta de capacidad para adaptarse a cualquier tipo de dispositivo, siendo así *responsive*.

## 3.2 Método de trabajo

En este apartado se explica y justifica la metodología elegida para realizar este trabajo. Se ha decidido utilizar **KANBAN** adaptado a las necesidades del proyecto**,** una metodología ágil basada en el desarrollo incremental, que permite visualizar el flujo de trabajo de manera sencilla con la finalidad de conseguir alcanzar los objetivos del proyecto con éxito.

### 3.2.1 Metodología KANBAN

La palabra *Kanban* es de origen japonés cuyo significado es “tarjetas visuales”. Esta metodología surgió a principios de 1940 gracias a la marca de automóviles japonesa Toyota, que creó un sistema simple de planificación cuyo objetivo era controlar y gestionar el trabajo e inventario de forma óptima en cualquier etapa del sistema de producción [9]. La primera vez que se utilizó esta metodología en un proyecto de ingeniería de software fue gracias a *David J.Anderson* en 2004 en la empresa Microsoft. El uso de la metodología fue un éxito rotundo que consiguió que la productividad del departamento se multiplicase por tres y los plazos de entrega disminuyesen en un 90% [10].

Existen multitud de formas de aplicar *Kanban* a un proyecto, pero existen un conjunto de etapas en común que siempre han de realizarse:

- **Definir el mapa del flujo de trabajo.** En esta fase se especifica la secuencia por la que pasa una necesidad desde que se detecta hasta que se da por finalizada.
- **Crear el tablero** con tantas columnas como estados de flujo se hayan definido anteriormente. Además, reflejará el flujo de trabajo real, no el deseado para un proceso.

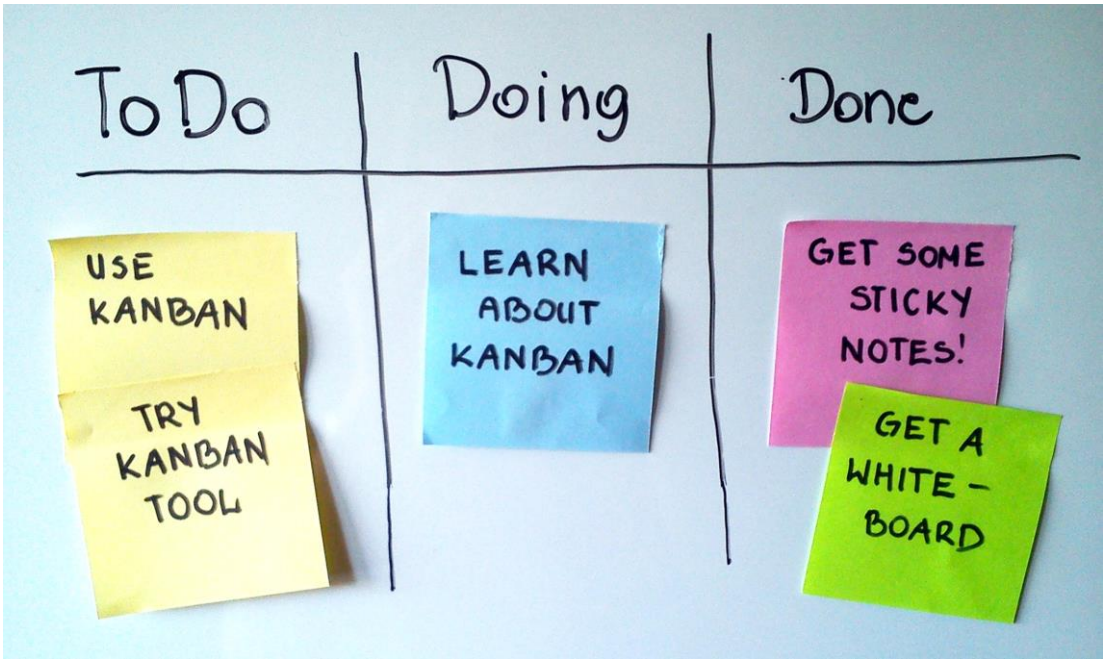


*Figura 8. Ejemplo de tablero básico Kanban*[2]

- **Definir el *WIP (Work In Progress)*** admitido por cada columna, es decir, el número máximo de tarjetas admitidas por columna.
- **Crear las tarjetas** e incluirlas dentro del tablero. Una tarjeta corresponderá con un trabajo que hay que realizar. Un ejemplo del contenido de tarjeta sería: identificador, título, descripción breve, fechas y persona asignada.
- **Medir el tiempo** empleado en cada tarjeta, desde que entra en el tablero hasta que se da por terminado. Esta fase se realiza con el objetivo de optimizar el proceso, eliminando todo aquello que no aporta valor y, por lo tanto, consiguiendo disminuir el tiempo del ciclo en futuras tareas.

Como se puede observar en las etapas comentadas, todo gira en torno al tablero y a las tarjetas. Gracias a ello, cualquier persona del equipo puede obtener una visión sobre cuál es el estado del proyecto, que tareas se están realizando, quién está realizando cada tarea… En consecuencia, permite optimizar todo el flujo de trabajo de manera sencilla.

[2] Imagen extraída de simpaticodesignstudio.com

Al igual que existen múltiples formas de aplicar la metodología, no existe un tablero único, si no que partiendo del tablero base, para cada proyecto se crea uno propio que se adapte al mismo. En la Figura 8 se muestra un ejemplo de tablero físico con tres columnas: tareas por hacer, tareas que se están llevando a cabo actualmente, tareas terminadas. Gracias a las nuevas tecnologías surgieron los tableros digitales basados en la nube, cuyo funcionamiento es similar al físico, pero se puede añadir más información, llevar un seguimiento más controlado, asignar tareas de forma más sencilla, etc.

A pesar de pertenecer al conjunto de metodologías ágiles, *Kanban* está basada en un conjunto de principios que la diferencian del resto [11]:

- **Calidad garantizada.** No se premia la rapidez, sino la calidad de las tareas realizadas. Todas las tareas deben salir bien a la primera con el objetivo de no tener que arreglar fallos después, puesto que en muchas ocasiones cuesta más arreglar el fallo que realizar la tarea correctamente a la primera.
- **Reducción del desperdicio.** Se basa en hacer bien las tareas realizando lo mínimo necesario, reduciendo todo aquello que sea secundario.
- **Mejora continua.** Su utilidad va más allá de ser un método de gestión, sino también un sistema para mejorar el desarrollo de proyectos *software*.
- **Flexibilidad.** El orden de tareas a realizar no está definido en un principio, si no que a cada tarea se le otorga una prioridad según las necesidades del proyecto pudiendo así dar respuesta a tareas imprevistas que sean urgentes.

### 3.2.2 Fases de trabajo

Para el correcto desarrollo de este proyecto, es necesario realizar previamente una planificación temporal, identificando las tareas necesarias y asignando tiempos estimados a cada una de ellas con el objetivo de optimizar al máximo el proyecto para realizarlo de forma productiva. Por lo tanto, el plan de trabajo propuesto es el siguiente:

- **Estructura de capítulos.** En esta tarea se elaborará el documento entregable donde se especifiquen todos los capítulos del TFM junto al contenido de cada uno de ellos.
- **Estado del arte.** En esta tarea se realizará la introducción al TFM, así como el contexto y estado del arte de este, realizando un estudio de otras soluciones parecidas para justificar la novedad o mejora de este proyecto.
- **Objetivos.** En esta tarea se detallará el objetivo principal que se desea conseguir con el desarrollo de este trabajo además de los objetivos secundarios que aportan funcionalidades extra o complementan al proyecto.

- **Primera fase.** En esta fase comenzará el desarrollo de la aplicación web, creando la funcionalidad básica y gestión de usuarios, aplicando *Bootstrap* e internacionalizando el sistema. Además, se creará la base de datos.
- **Segunda fase.** En esta fase se incluirán las carreras en la aplicación junto con la funcionalidad que permite importarlas masivamente.
- **Tercera fase.** En esta fase se crearán los distintos filtros que permitan encontrar un tipo de carrera. También se creará la funcionalidad de alertas personalizadas.
- **Cuarta fase.** En esta fase se incluirá el componente social de la aplicación, permitiendo integrarse con redes sociales.
- **Quinta fase.** En esta fase se realizarán pruebas en la aplicación, así como *tests* de usabilidad y accesibilidad. Por último, se realizarán mejoras y correcciones de los errores encontrados en estas pruebas.
- **Memoria.** En esta tarea se terminará el documento de la memoria del TFM, completando los apartados restantes por acabar.
- **Revisión y mejoras.** En esta tarea se revisará el TFM y se corregirán los errores encontrados y se llevarán a cabo las mejoras o cambios propuestos en la corrección del borrador final.

Por último, se ha creado un **Diagrama de Gantt** (véase Figura 9) con la planificación temporal asociada a las fases previamente explicadas. La primera fase comienza el 1 de junio y el proyecto termina el 26 de julio coincidiendo con el **depósito del TFM**

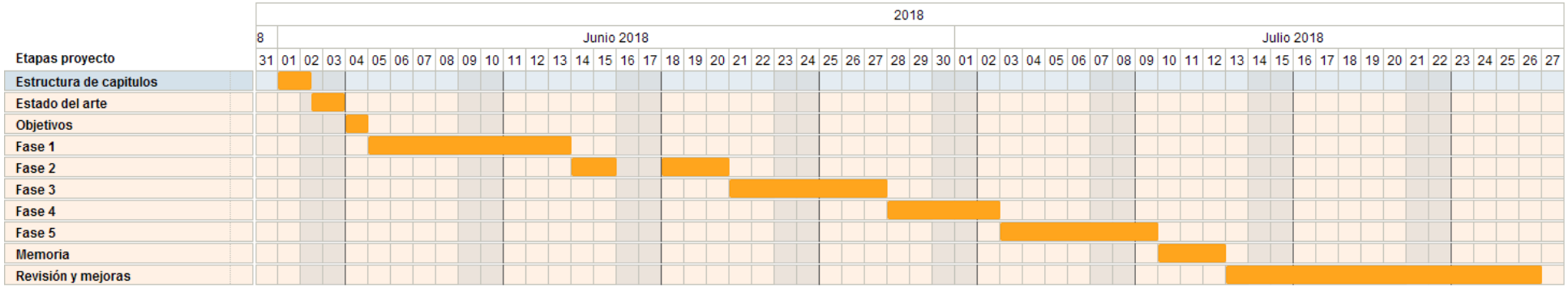


*Figura 9. Diagrama de Gantt.*

## 3.3 Marco tecnológico

En este apartado se describen brevemente las distintas herramientas y tecnologías, tanto *hardware* como *software*, que se han utilizado para la realización de este proyecto.

### *3.3.1* Herramientas *software*

**Sistemas operativos**

- **Windows 10 x64** [12]**.** *Windows 10* es una versión de Microsoft Windows diseñada para su uso en ordenadores y dispositivos portátiles como tabletas, móviles…

**Servidores**

- **mLab** [13]. *mLab* es un servicio *DBAS*, es decir un servicio de bases de datos *MongoDB* alojado en la nube. Permite crear bases de datos *MongoDB* en la nube y alojarlas en los servidores de *Amazon Web Service, Azure* o *Google*.
- **Evennode** [14]**.** *Evennode* es un hosting web flexible que permite desplegar aplicaciones desarrolladas en *Python* o *NodeJS*.

**Entorno de desarrollo**

- **WebStorm** [15]**.** *WebStorm* es un entorno de desarrollo integrado que permite crear aplicaciones web utilizando *JavaScript* o sus tecnologías relacionadas como *NodeJS*, *Angular* o *React*.

**Tecnologías y lenguajes de programación**

- **NodeJS** [16]**.** *NodeJS* es un entorno de ejecución para *JavaScript* construido con el motor de *JavaScript V8* de Chrome. Está orientado a desarrollo en el servidor
- **JavaScript** [17]**.** *JavaScript* es un lenguaje de programación interpretado que se utiliza, principalmente, en desarrollo web del lado del cliente.
- **jQuery** [18]**.** *jQuery* es la biblioteca más utilizada de JavaScript que simplifica la interacción con documentos HTML además de permitir manejar *eventos* y animaciones.
- **HTML + CSS.** *HTML* es un lenguaje de etiquetas para la elaboración de páginas web. *CSS* es un lenguaje de hojas de estilo utilizado para describir la presentación de un documento escrito en *HTML*.
- **Bootstrap** [19]. *Bootstrap* es un *framework HTML, CSS y JavaScript* para diseño de sitios y aplicaciones web *responsive.*

- **MongoDB** [20]**.** *MongoDB* es un sistema gestor de base de datos NoSQL orientado a documentos de código abierto.
- **i18next** [21]**.** *i18next* es una biblioteca de internalización para aplicaciones web independiente de la tecnología usada.
- **Jade** [22]**.** *Jade* es un motor de plantillas de alto rendimiento implementado con *JavaScript* para trabajar con *NodeJS*.

**Modelado**

- **Balsamic Mockups** [23]**.** *Balsamic Mockups* es una herramienta que permite el diseño de interfaces mediante la realización de bocetos.
- **yUML** [24]**.** *yUML* es una herramienta online de soporte al modelado, que permite realizar diagramas de clase, actividad y de casos de uso.

**Documentación**

- **Microsoft Office Word 365** [25]**.** *Microsoft Word* es un procesador de textos muy completo que se encuentra dentro del paquete Microsoft Office.
- **Sublime Text 3** [26]**.** *Sublime Text* es un editor de código y texto multiplataforma que soporta un gran número de lenguajes de programación.
- **Zotero** [27]**.** *Zotero* es un programa de software libre, abierto y gratuito para la gestión de referencias bibliográficas. Además, cuenta con una extensión para navegador web.

**Control de versiones**

- **Git** [28]**.** *Git* es un software de control de versiones pensado en la eficiencia y mantenibilidad de versiones en proyectos con un gran número de ficheros fuente.
- **Bitbucket** [29]**.** *Bitbucket* es un servicio de alojamiento basado en web para proyectos que utilicen el sistema de control de versiones *Mercurial* o *Git*.

### 3.3.2 Herramientas *hardware*

- **Ordenador personal**. Ordenador portátil ASUS GL552VW con el software anteriormente citado instalado. El ordenador cuenta con las siguientes características hardware:
  - Procesador Intel Core i7-6700HQ (4 núcleos 3.5GHz).
  - Tarjeta gráfica NVIDIA GeForce GTX 960M.
  - Memoria RAM de 8GB.
  - Disco duro SSD de 128GB.

- Disco duro sólido de 1TB.

# Capítulo 4: Desarrollo del proyecto

En este capítulo se desarrolla cada una de las fases indicadas en el *Fases de* trabajo con el objetivo de profundizar en más detalle cada una de ellas, explicando qué se realiza en cada fase, cómo se van a abordar y cuál va a ser el resultado al terminar cada una de ellas.

## 4.1 Primera fase: Aplicación web básica

En la primera fase del proyecto se desarrolla la funcionalidad básica de la aplicación web, correspondiente con la gestión de cuentas de usuario, junto con la creación de la interfaz web aplicando *Bootstrap*, la internalización y la creación y conexión con la base de datos de *MongoDB*.

Cualquier aplicación web permite al usuario crear una cuenta para, posteriormente, identificarse con ella y poder utilizar la aplicación de forma personalizada, es decir, con sus propios datos. Además, en caso de olvidar la contraseña, se debe permitir recuperarla para que la cuenta no quede inutilizable.

Como ya se indicó, uno de los objetivos teóricos consistía en estudiar el patrón de arquitectura MVC (véase Figura 10) con el fin de aplicarlo a este proyecto. Este patrón se basa en separar la lógica de la aplicación respecto a la interfaz de usuario, mediante el uso de tres componentes o capas [30]:

- **Modelo**. Corresponde con la representación de la información del sistema. Generalmente se conecta con la base de datos para acceder a la información.
- **Vista.** Es la representación visual de los datos. Recibe los datos enviados por el controlador para representarlos en un formato adecuado con el que el usuario interactúe.
- **Controlador.** Se encarga de recibir los eventos, de solicitar los datos al modelo y enviárselos a la vista

Adicionalmente, existe una “cuarta capa” directamente conectada con la del Modelo, que correspondería con la **Base de Datos** para permitir almacenar persistentemente la información de la aplicación.

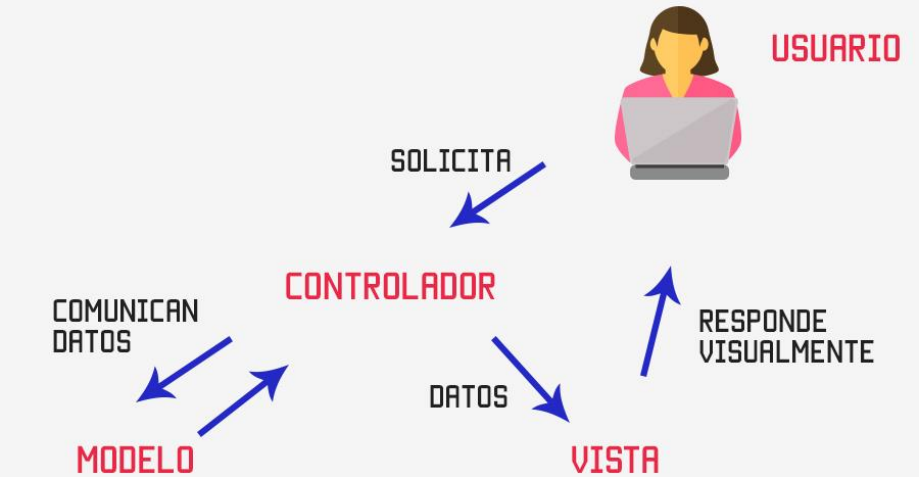


*Figura 10. Patrón Modelo-Vista-Controlador*[3]

En este proyecto, se ha utilizado una base de datos MongoDB donde toda la información se almacena en colecciones. Dentro de cada colección se encuentran los documentos (contienen los datos), que no definen un esquema fijo permitiendo así que puedan tener distintos atributos. En esta fase del proyecto, al solo almacenar la información de los usuarios, la base de datos contendría sólo una colección, *users*, aunque posteriormente, en función de las necesidades del proyecto y para poder cubrir más funcionalidades, el número de colecciones aumentaría.

Aunque MongoDB permite que cada documento pueda tener diferentes atributos, incluso estando en la misma colección, para la colección de usuarios se define un esquema fijo (véase Figura 11). Este esquema define que cada documento de la colección de *users*, debe contener 5 atributos: *email*, *username*, *province*, *password*, *athlete* (*boolean* para indicar si es deportista u organizador) y otros dos opcionales: *resetPasswordToken* y *resetPasswordExpires* que se utilizarán para recuperar la contraseña del usuario.

[3] Imagen extraída de codigofacilito.com

```
var UserSchema = new mongoose.Schema({
    email: {
        type: String, unique: true, required: true, trim: true
    },
    username: {
        type: String, unique: true, required: true, trim: true
    },
    province:{
        type: String, required: true
    },
    password: {
        type: String, required: true
    },
    athlete:{
        type: Boolean, required: true
    },
    resetPasswordToken: String,
    resetPasswordExpires: Date
});
```

*Figura 11. Esquema colección users*

Para el diseño de la interfaz de usuario se han realizado una serie de bocetos que proporcionan una base sobre la que luego partir para conseguir la implementación.

En la Figura 12 se muestra el boceto de la interfaz de usuario para la página inicial correspondiente. En ella se pueden observar las próximas carreras que se van a celebrar, existe un menú desplegable para iniciar sesión y recuperar contraseña y un formulario de registro que aparecería de forma emergente en el centro de la página. Por último, en la parte inferior existe la opción de cambiar el idioma de la página web.

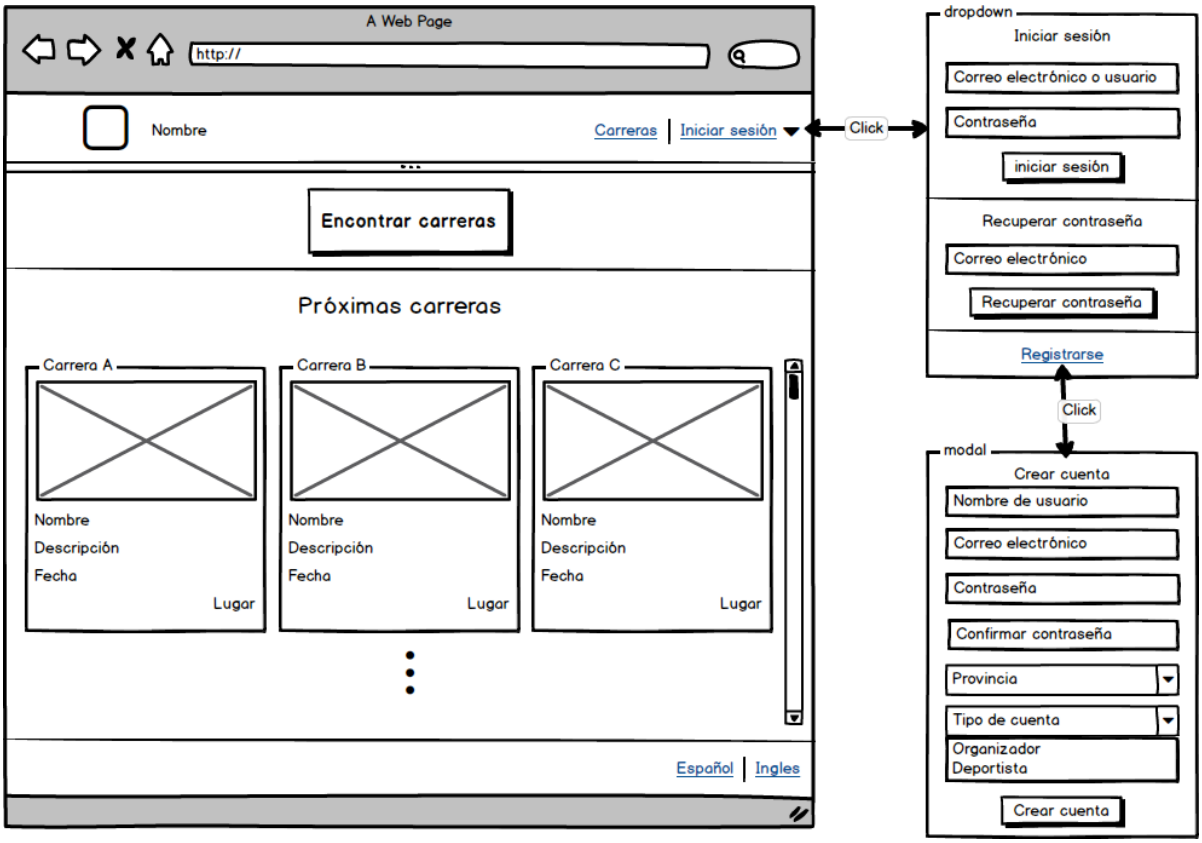


*Figura 12. Boceto página inicial*

En la Figura 13 se muestra el boceto de la página de recuperar contraseña. De la misma forma que en la anterior, contiene una barra de navegación superior (*navbar*) y el pie de página (*footer*) y en la parte central se muestra un formulario para cambiar la contraseña.

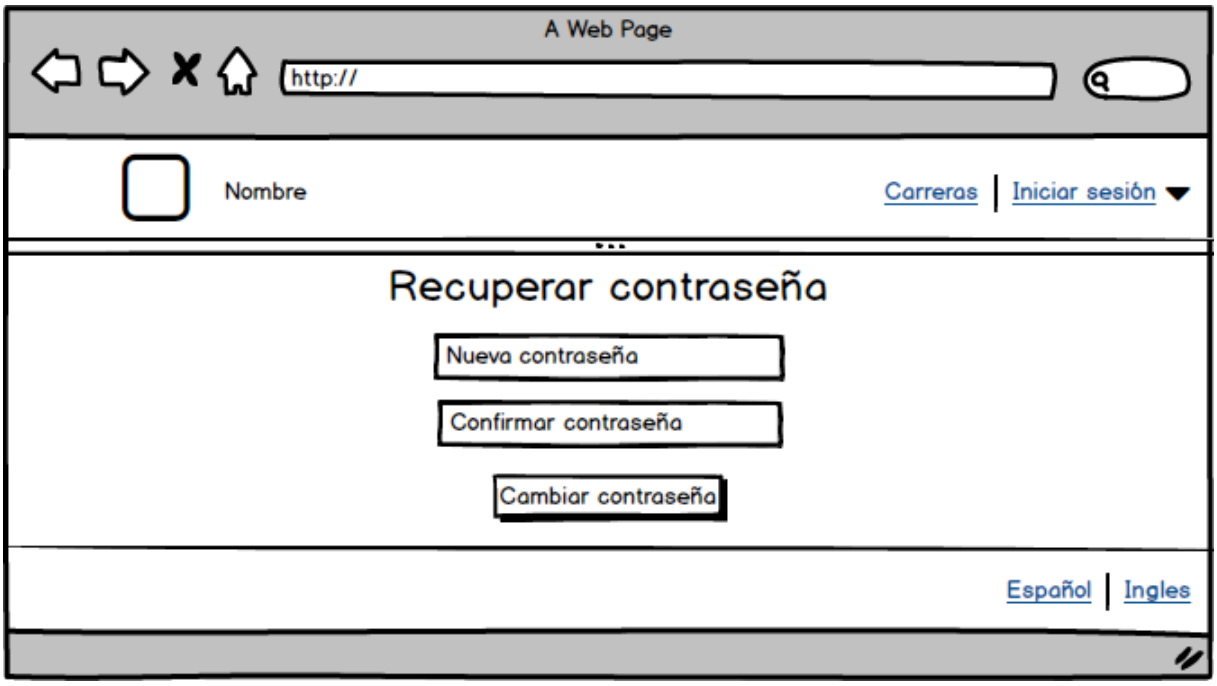


*Figura 13. Boceto página recuperar contraseña*

De esta forma, en la primera iteración se tendría el siguiente conjunto de casos de prueba (véase Figura 14).

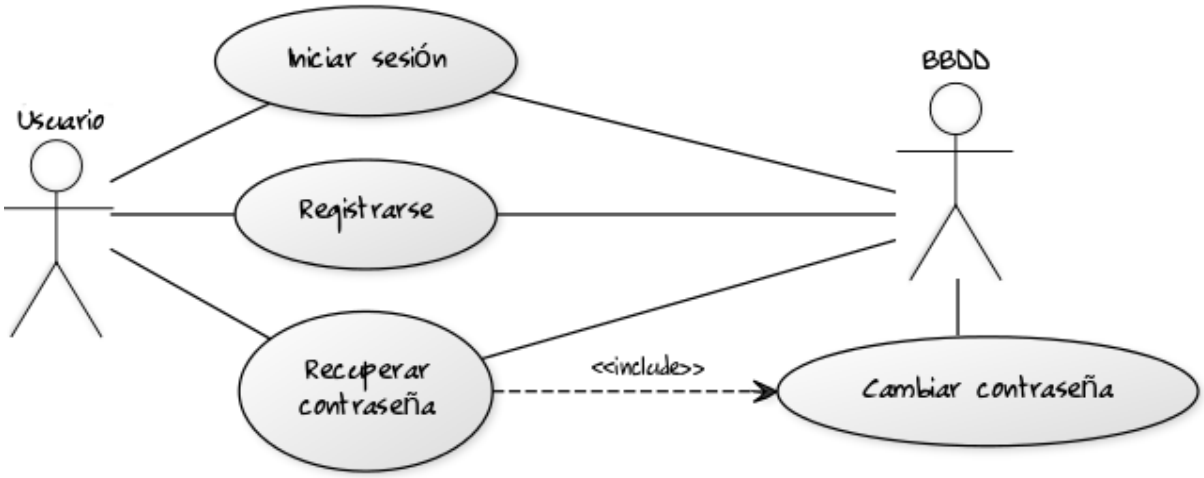


*Figura 14. Diagrama de casos de uso (primera fase)*

Una vez diseñada la primera iteración se procede a implementarla. Para implementar la interfaz de usuario se parte de una plantilla simple de *Bootstrap* que se modifica para adaptarla a los bocetos diseñados con anterioridad. Además, la interfaz pretende ser clara, usable y mostrar *feedback*, tanto positivo como negativo, para cualquiera de las acciones que el usuario realice en ella. La implementación de las funcionalidades de esta fase es la siguiente:

- **Registrarse.** Mediante una petición *AJAX*, se envían como parámetros los datos recogidos en el formulario de registro (se comprueban tanto en el cliente, para no mandar una petición innecesaria al servidor, como en el servidor, por si se ha alterado el contenido de la petición, que dichos datos sean válidos). El servidor recibe la petición, procesa los parámetros y crea un nuevo objeto usuario que insertará como documento en la base de datos, dentro de la colección de *users*. Para aumentar la seguridad, la contraseña no se almacena directamente en la base de datos, sino que se aplica una función *hash* sobre ellas para no comprometer la seguridad del usuario.
- **Iniciar sesión.** De la misma forma que la funcionalidad anterior, se envían los datos de inicio de sesión y en caso de ser satisfactorio, la interfaz cambia para mostrar los datos de usuario en el menú. En caso de ser erróneo, se informa al usuario para que corrija los datos introducidos.
- **Recuperar contraseña.** Esta funcionalidad se divide en dos fases, la primera consiste en la solicitud del cambio de contraseña y la segunda el propio cambio. Para la primera se envía al servidor el *email* de la cuenta que se desea recuperar la

contraseña, el servidor crea una *URL* con un *token* personalizado (que se insertará en el documento del usuario, junto a la fecha de expiración) y envía un correo electrónico a la dirección del usuario. En la segunda fase, el usuario accederá a la *URL* que cargará una página con un formulario donde debe introducir la nueva contraseña y confirmarla. Si el *token* es válido, no ha expirado y las contraseñas son adecuadas, se realizará el cambio de contraseña, notificando al usuario el resultado de la operación.

## 4.2 Segunda fase: Integración de carreras

En la segunda fase del proyecto se desarrolla toda la funcionalidad referente a las carreras, desde crearlas o importarlas masivamente, hasta mostrarlas al usuario en la aplicación, pasando por poder modificar su información.

Al igual que con los usuarios, para la colección carreras de carreras se ha definido un nuevo esquema fijo (véase Figura 15). Este esquema establece que cada documento *race* debe contener 6 atributos obligatorios: *name, sport, url, province, place, dateTime* y otro opcional *owner* que será una referencia al usuario que ha creado esa carrera en la aplicación y que la gestiona.

```
var RaceSchema = new mongoose.Schema({
    name: {
        type: String, unique:true, required: true, trim: true
    },
    sport: {
        type: String, required: true
    },
    url: {
        type: String, required: true
    },
    province:{
        type: String, required: true
    },
    place: {
        type: String, required: true, trim: true
    },
    dateTime:{
        type: Date, required: true
    },
    owner:{
        type: mongoose.Schema.ObjectId, ref:"user"
    }
});
```

*Figura 15. Esquema colección races*

En lugar de realizar directamente la implementación de la interfaz de usuario, se han realizado un conjunto de bocetos previos para establecer un diseño base sobre el que empezar a implementar.

En la Figura 16 se muestra el boceto de la página principal de carreras, donde se permite que el usuario utilice un filtro para encontrar su carrera deseada. El filtro permite buscar por nombre de carrera, tipo de deporte, provincia donde se disputa y fecha, pudiendo combinarlos para realizar una búsqueda más precisa. En la parte inferior se muestran todas las carreras que coincidan con el filtro de búsqueda definido. Aunque la funcionalidad de filtrar carreras no pertenece a la segunda fase del proyecto, sino a la tercera, se incluye ya en los bocetos debido a que comparten la misma pantalla, aunque no se implementa la funcionalidad.

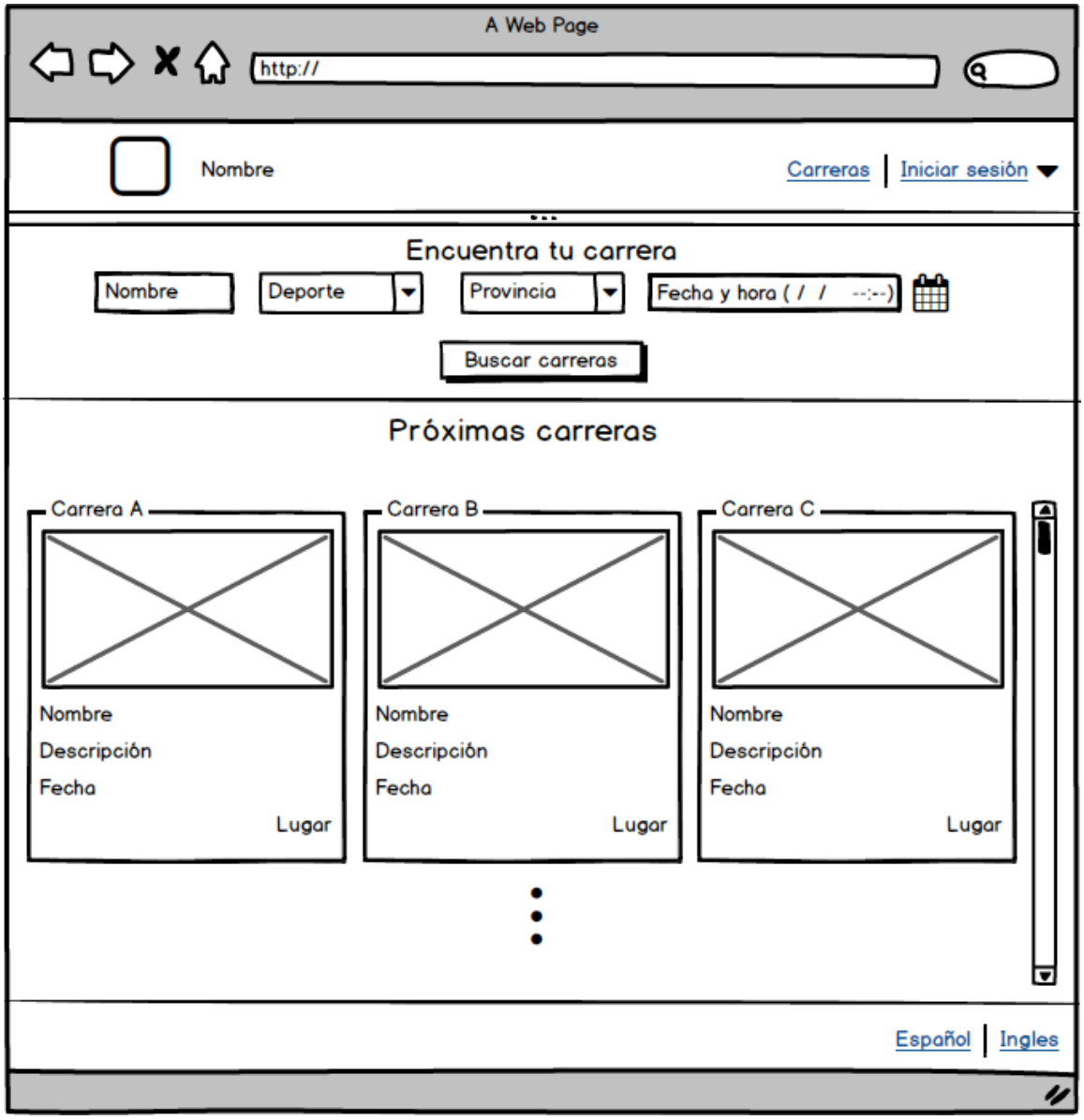


*Figura 16. Boceto página carreras*

En la Figura 17 se puede ver el boceto del formulario que permite crear una nueva carrera en la aplicación (esta misma interfaz, servirá para modificar una carrera ya existente, cambiando el texto del título y del botón). Este formulario recoge la información básica de cada carrera, especificada en el esquema de la Figura 15.

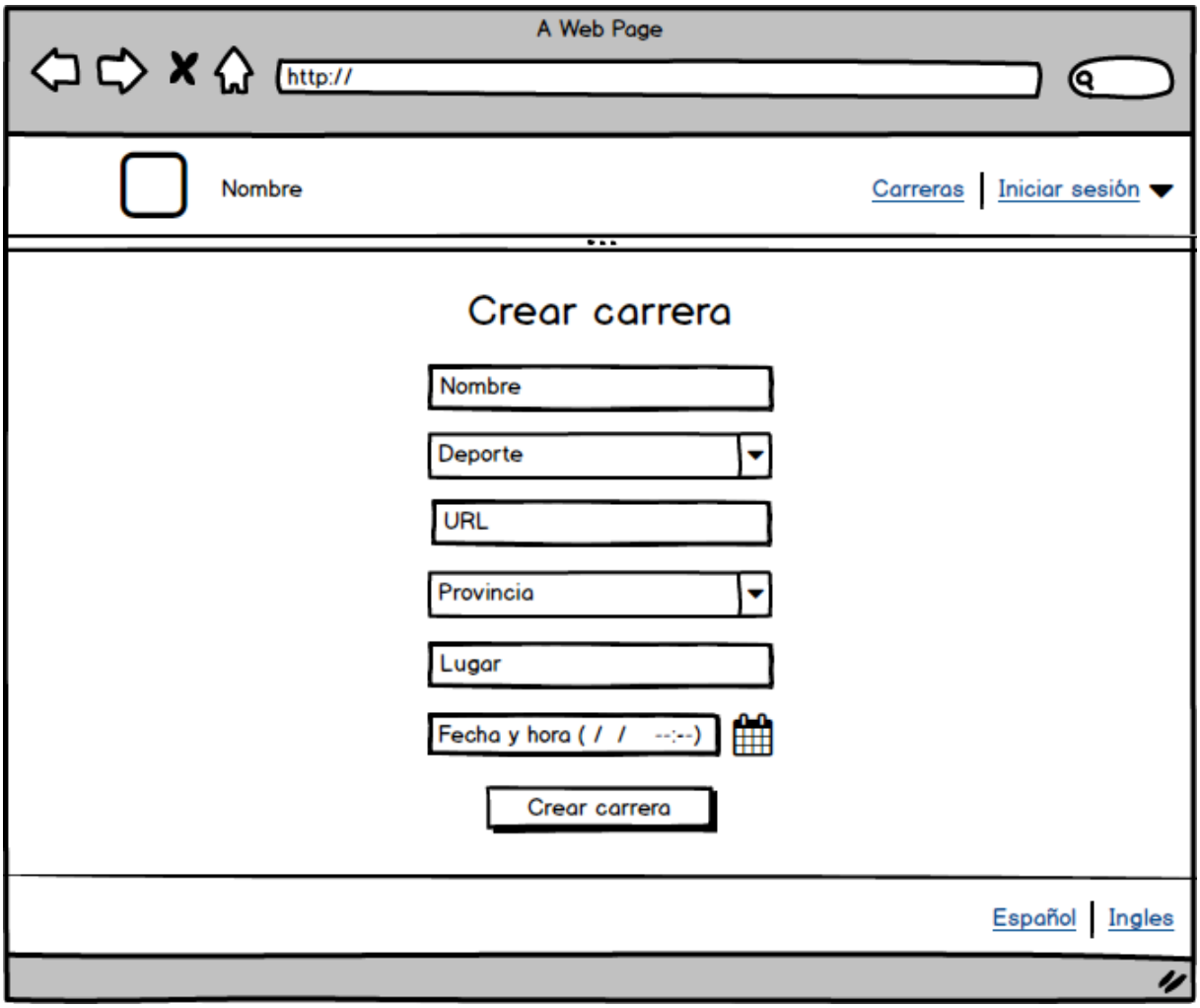


*Figura 17. Boceto página crear carrera*

El boceto de la página que permite importar carreras de forma masiva se corresponde con la Figura 18, que muestra una página básica que permite al usuario descargar la plantilla Excel para introducir los datos de su carrera para, posteriormente, subirla a través de la misma página. Para informar al usuario del progreso en la subida del archivo, existe una barra de progreso que indica en qué estado se encuentra.


*Figura 18. Boceto página importar carreras*

En la Figura 19 se muestra la página de perfil de un usuario, que cuenta con un formulario para que éste pueda modificar sus datos una vez creada la cuenta o eliminarla. Esta funcionalidad correspondería con la primera fase al tratarse de funcionalidad del usuario, pero se ha aplazado al no ser prioritario para el desarrollo del proyecto. Esta es una de las muchas ventajas que se obtienen utilizando la metodología *Kanban*. La parte derecha varía según el tipo de usuario que acceda: si el usuario es atleta, se muestran las carreras a las que dicho usuario se ha apuntado, pudiendo desapuntarse también (esta funcionalidad se implementará posteriormente), junto con la opción de exportar su calendario. Si el usuario es organizador, se muestran las carreras que él mismo ha creado, pudiendo modificar o eliminar cada una de ellas. Además, aparecen dos botones que le permiten crear o importar carreras, redirigiéndole a la página correspondiente (véase Figura 17 o Figura 18 respectivamente)

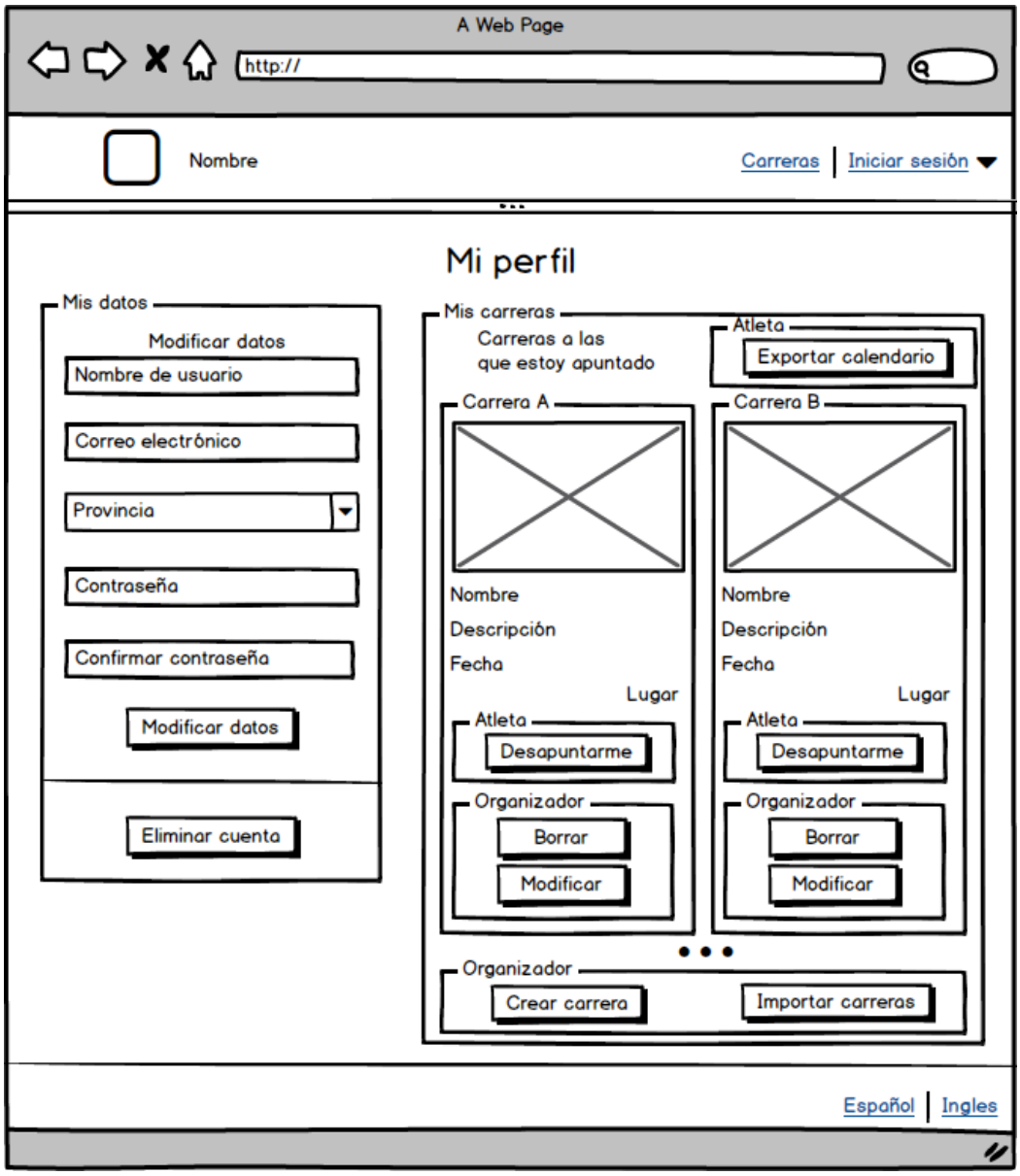


*Figura 19. Boceto página perfil de usuario*

Una vez diseñados todos los bocetos de esta fase, el diagrama de casos de uso que recoge las funcionalidades que se van a desarrollar es el siguiente (véase Figura 20).

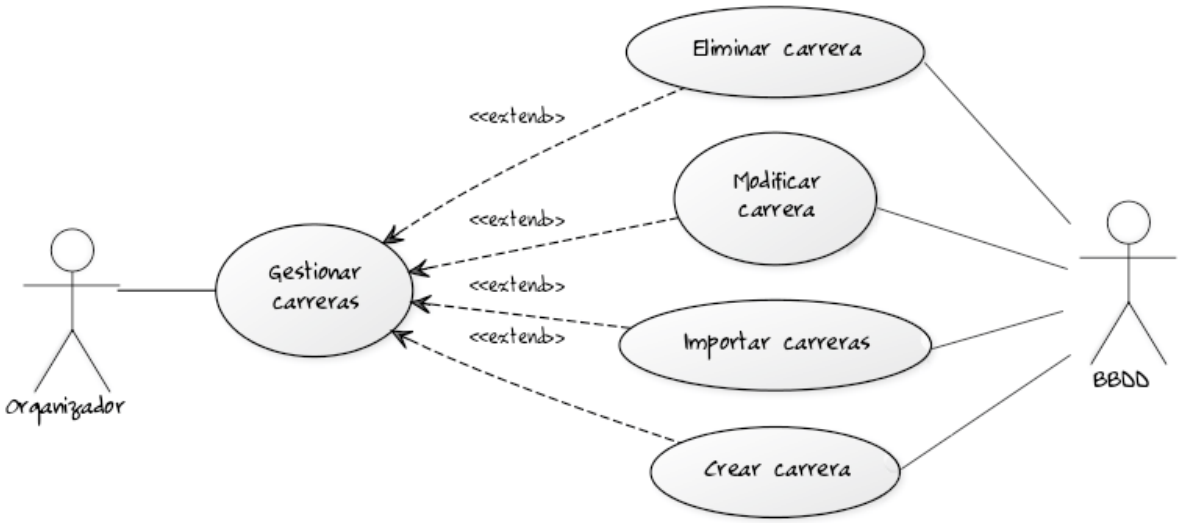


*Figura 20. Diagrama de casos de uso (segunda fase)*

En este diagrama se muestra un caso de uso general (Gestionar carreras) del que extienden las funcionalidades más específicas: crear, modificar, eliminar e importar carreras.

- **Crear carrera.** Cuando el usuario accede mediante una petición HTTP a la página */races/create*, el servidor comprueba que dicho usuario esté autenticado y sea organizador, puesto que los atletas no pueden crear carreras. En caso afirmativo manda en la respuesta el formulario para crear la carrera (véase Figura 17) y en caso contrario, muestra un mensaje de error. Una vez el organizador haya rellenado los campos del formulario, se envían al servidor mediante una petición *AJAX*, donde son validados y si no existe ningún error, se inserta la carrera en la base de datos y se informa al usuario que la operación ha sido satisfactoria.
- **Importar carreras.** Al igual que en la funcionalidad anterior, se comprueba que el usuario que accede a la página de importar carreras sea organizador. En ella, el organizador puede subir una plantilla de Excel con sus propias carreras, que previamente ha descargado de la misma página y rellenado con los datos de sus carreras, el servidor procesa dicho archivo y va insertando de una en una las carreras. Si se produce un error, el proceso parará e informará al usuario de dónde ha encontrado un fallo en los datos, para que este lo pueda corregir. Esta funcionalidad se encuentra explicada con más detalle en el Anexo D: Importación masiva de datos.

- **Modificar carrera.** De la misma forma, se comprueba que el usuario que quiere modificar una carrera sea organizador y además sea el creador de esta. En dicho caso, el servidor enviará una página con un formulario con los datos de la carrera y, una vez el usuario los modifique y pulse el botón de guardar, se enviarán los datos al servidor, dónde se procesarán y actualizarán en la base de datos.
- **Eliminar carrera.** Asimismo, en esta funcionalidad se comprobará que el usuario que solicite eliminar una carrera sea quien la haya creado. En ese caso, se eliminará la carrera de la base de datos y se notificará al usuario del resultado de la operación. Además, antes de realizarla, se mostrará un mensaje de confirmación para que el usuario confirme que quiere eliminarla, dado que se trata de una acción irreversible.

## 4.3 Tercera fase: Filtro de carreras y alarmas

La tercera etapa del proyecto aborda el resto de las funcionalidades respecto a las carreras, como permitir al usuario filtrarlas según diferentes aspectos y crear alarmas para que el sistema les envíe una notificación cuando se registre una carrera que coincida con su búsqueda.

De la misma forma que en las dos etapas previas, la funcionalidad de las alarmas necesita guardar cierta información de forma persistente, añadiéndola a la base de datos. Para ello se ha creado un nuevo esquema fijo llamado *Alarm* (véase Figura 21). Este esquema define que los documentos *alarm* deben contener un *String* correspondiente a la búsqueda deseada y una lista con los *emails* de los usuarios que están interesados en ella.

```
var AlarmSchema = new mongoose.Schema({
    search: {
        type: String, unique:true, required: true, trim: true
    },
    users:[
        { type : String}
    ]
});
```

*Figura 21. Esquema colección alarms*

Para esta fase no es necesario realizar un gran número de bocetos como en la fase anterior debido a que se implementan ciertas funcionalidades que el usuario utilizará con las interfaces definidas anteriormente.

En la Figura 22 se puede observar una variante de la página de carreras (véase Figura 16). Al realizar una búsqueda que no devuelva ningún resultado, se mostrará en la pantalla junto con la opción de crear una alarma con el filtro empleado.

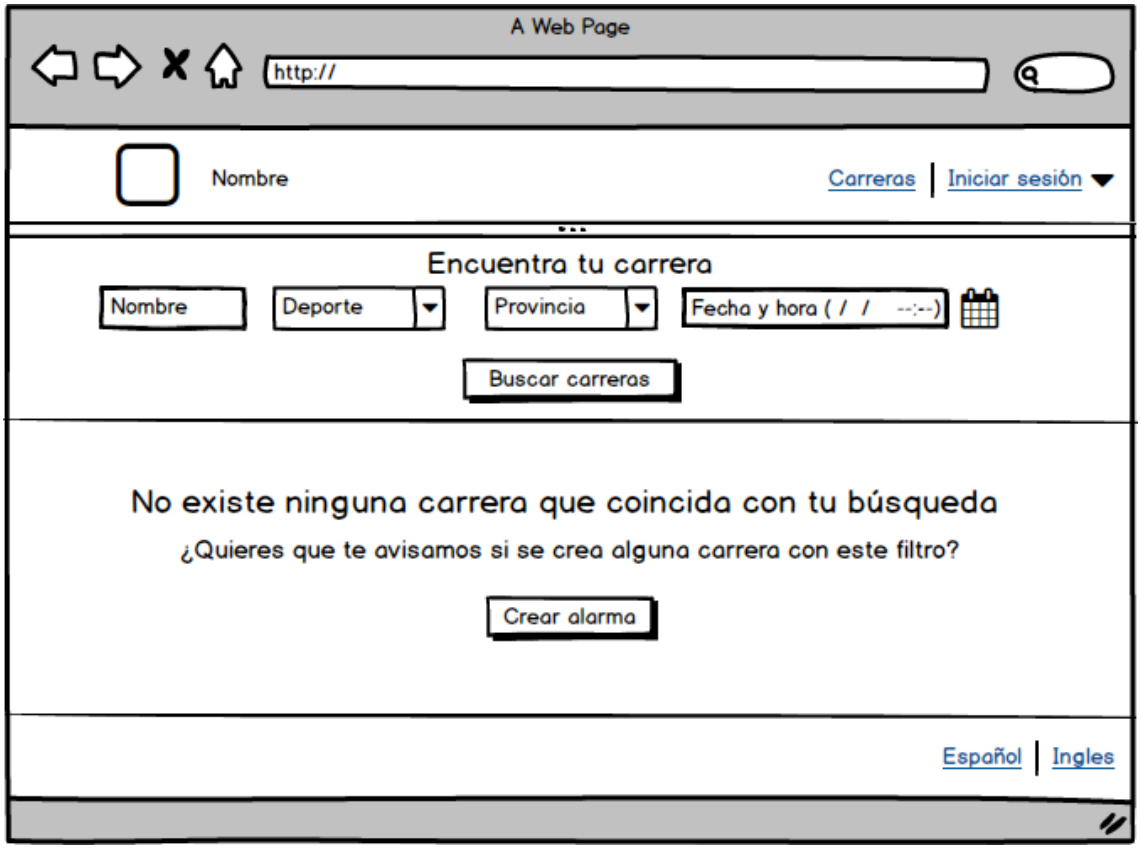


*Figura 22. Boceto página carreras (crear alarma)*

En la Figura 23 se muestra una página muy similar a la de carreras (véase Figura 16), pero eliminando la opción que permite filtrarlas. En esta página se observarán todas las carreras que pertenezcan a un mismo organizador junto con un título que indique quien es el organizador que gestiona esas carreras.

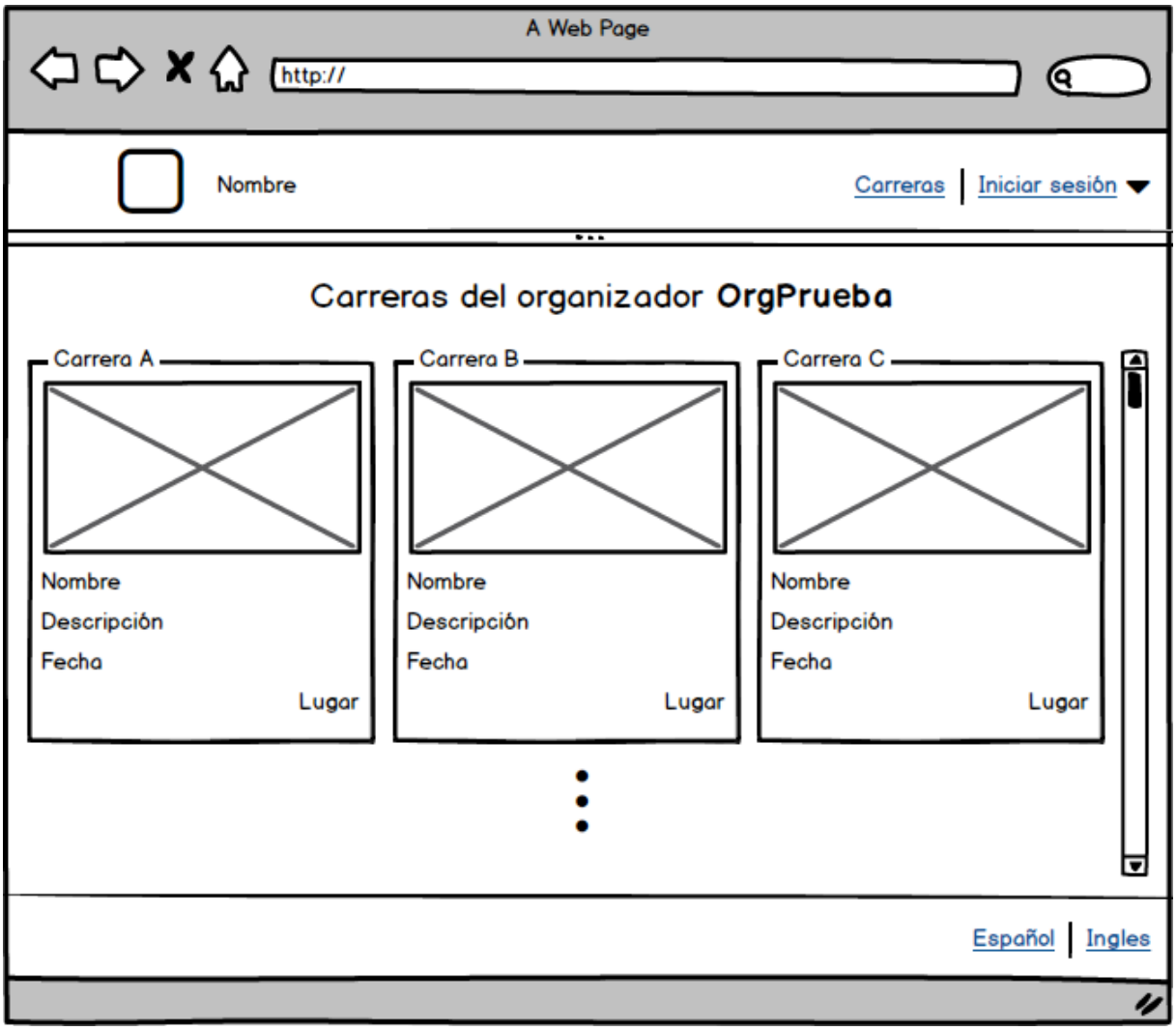


*Figura 23. Boceto página carreras de un organizador*

En la Figura 24 se detalla cómo es la página de una carrera específica, donde se puede ver en más detalle toda su información. En la parte izquierda se muestra la imagen de la carrera junto un botón para que el usuario se apunte a ella. En la parte derecha, se muestran dos secciones: información de la carrera y del organizador, con el que se podrá contactar para preguntar cualquier duda.

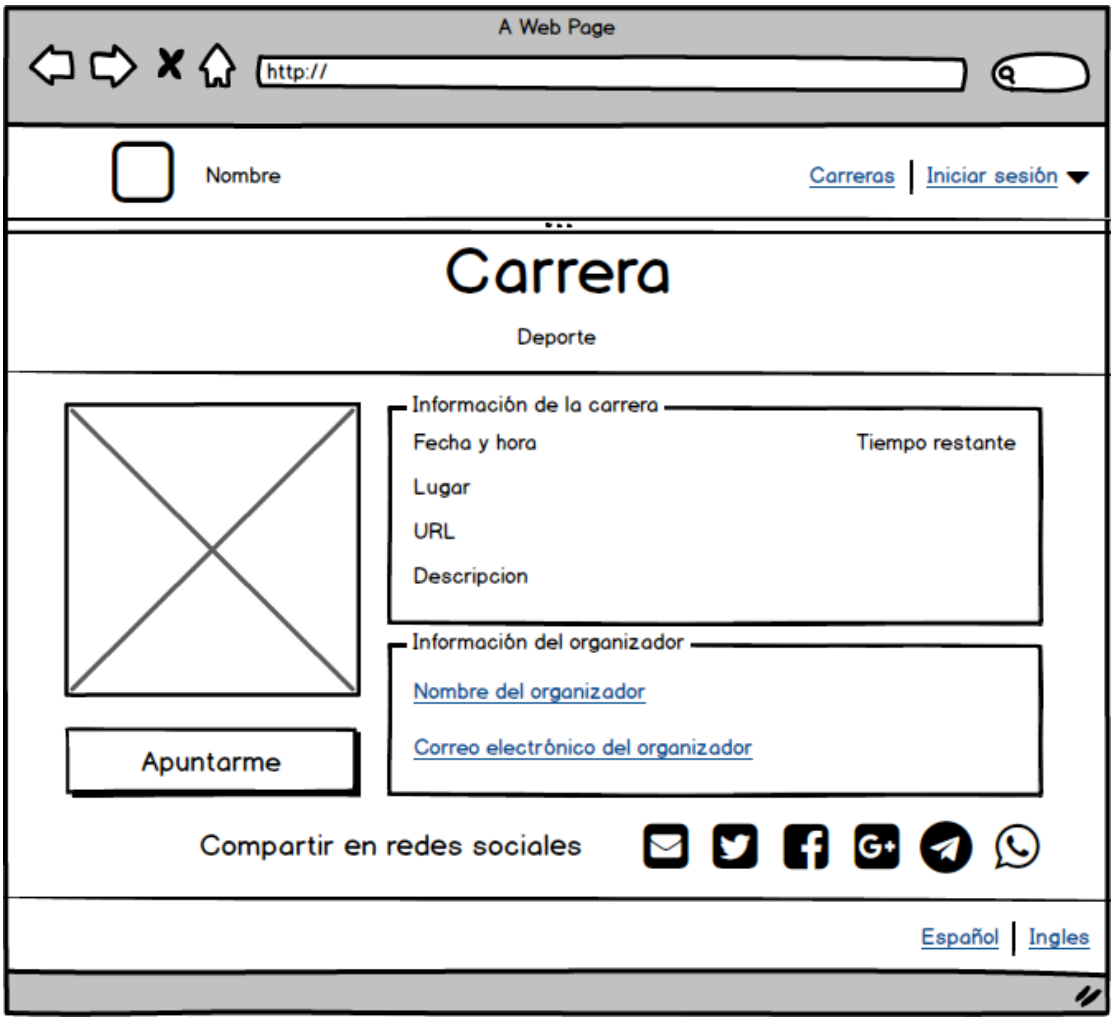


*Figura 24. Boceto página carrera*

Una vez explicados todos los bocetos correspondientes a las nuevas interfaces de usuario, en la Figura 25 se muestra el diagrama de casos de uso para esta fase. En él, se pueden observar cuatro casos de uso que relacionan a diversos agentes entre si (Usuario, Servidor y Base de Datos) y uno, enviar aviso, que forma parte de la funcionalidad de comprobar alarmas. A continuación, se detalla cómo funcionan cada una de ellas:

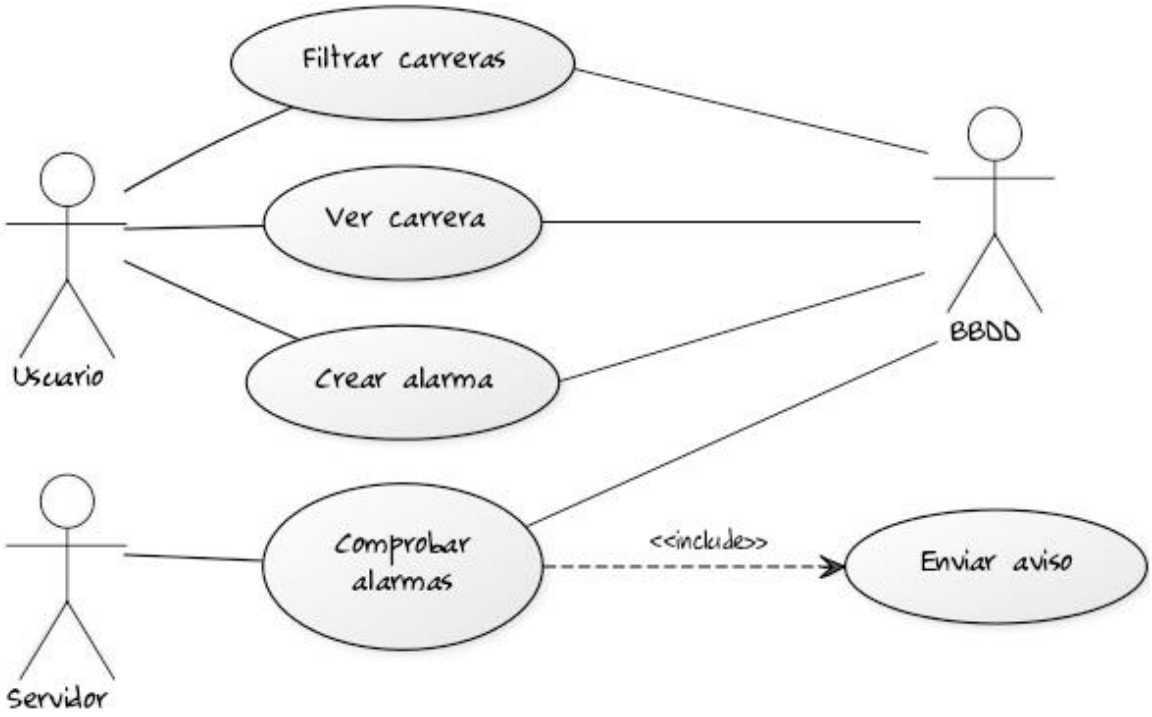


*Figura 25. Diagrama de casos de uso (tercera fase)*

- **Filtrar carreras.** El usuario tiene diversas formas de filtrar las carreras, según los diferentes campos de las carreras (nombre, deporte, lugar y fecha) o por organizador. En el primer caso, el usuario especifica su filtro de búsqueda y lo envía al servidor, que realizará una búsqueda en la base de datos para ver si coincide alguna carrera. En caso afirmativo, devuelve y muestra al usuario las carreras que ha devuelto la búsqueda. En caso contrario, informa al usuario que no ha existe ninguna carrera que cumpla con esas condiciones y le ofrece la funcionalidad de crear una alarma. El filtro por organizador devuelve todas las carreras que hayan sido creadas por el mismo usuario, siendo muy útil esta búsqueda para ver diversas carreras del mismo circuito.
- **Crear alarma.** Cuando el usuario realice una búsqueda específica y no encuentre ninguna carrera, el sistema dará la opción al usuario de crear una alarma que le avise cuando se registre una carrera que cumpla con esas características. El usuario, al pinchar en el botón de crear alarma, envía mediante una petición *AJAX* los parámetros de la búsqueda que ha utilizado. El sistema los recibe y procesa, accediendo a la base de datos para insertar un nuevo documento con la búsqueda realizada y el *email* del usuario. Si esa búsqueda la ha realizado con anterioridad otro usuario, no se crea un nuevo documento, sino que se inserta el email del usuario en la lista habilitada para ello.

- **Comprobar alarma.** Esta función se ejecuta de forma automática en el servidor cada vez que se añaden nuevas carreras al sistema. Al añadir una nueva carrera, se recuperan todas las alarmas que se hayan registrado en la base de datos, se comprueba si existe alguna carrera que coincida con alguna y, en caso afirmativo, se envía un *email* a los usuarios correspondientes (eliminando además la alarma de la base de datos al haber enviado ya el aviso) informando de qué carreras corresponden con la alarma creada previamente.
- **Ver carrera.** Cuando el usuario necesita ver toda la información de una carrera en más detalle, puede acceder a ella mediante esta funcionalidad. Se envía una petición al servidor indicando el nombre de la carrera que se desea obtener más información, el servidor realiza una búsqueda en la base de datos y la muestra al usuario.

## 4.4 Cuarta fase: Motor social e integración con RRSS

En la cuarta fase, se desarrolla el motor social de la aplicación, que incluye la gestión de inscripciones a carreras por parte del usuario atleta (para establecer su propio calendario y exportarlo) y la posibilidad de compartir en diversas redes sociales como Facebook o Twitter.

En esta fase, al desarrollar la funcionalidad que permite a los usuarios inscribirse a las carreras, es necesario realizar una leve modificación del esquema *User*, añadiendo un nuevo campo llamado *races* que será una lista con todos los identificadores de las carreras a las que esté apuntado. Por lo tanto, al esquema habría que añadirle lo siguiente:

```
races:[
    {type: mongoose.Schema.ObjectId, ref:"race"}
]
```

*Figura 26. Modificación esquema users*

Para esta iteración no es necesario realizar ningún boceto de interfaz de usuario debido a que todas las funcionalidades que se contemplan en ella se utilizan a través de interfaces ya desarrolladas.

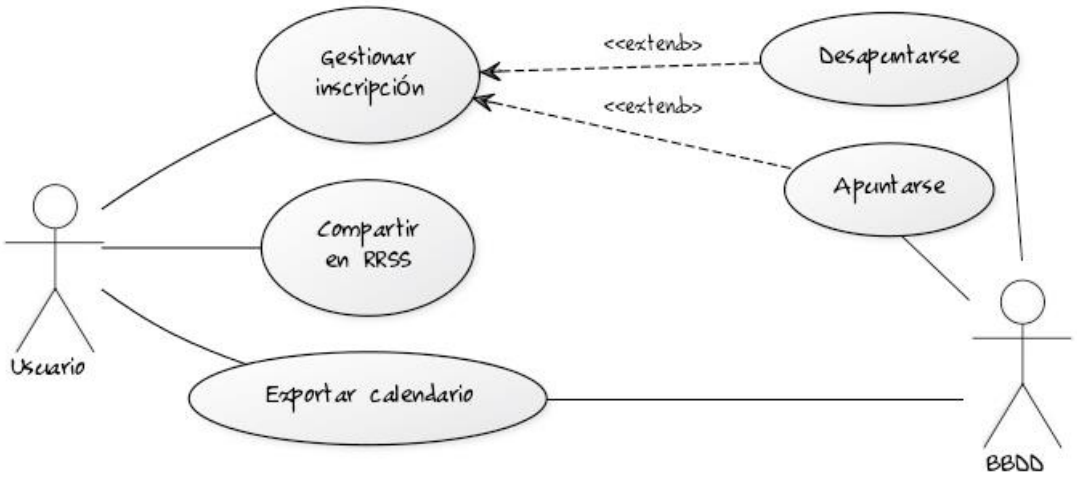


*Figura 27. Diagrama de casos de uso (cuarta fase)*

El diagrama de casos de uso (véase Figura 27) muestra un caso de uso general (Gestionar inscripción) que se compone de dos más específicos (Apuntarse y desapuntarse) y otros dos que completan el resto de las funcionalidades:

- **Apuntarse a una carrera.** Cuando un usuario esté interesado en una carrera, tiene la posibilidad de inscribirse en ella. Para ello, puede activar el botón habilitado, que envía una petición indicando la carrera elegida. El servidor procesa la información y comprueba si el usuario ha iniciado sesión, en caso afirmativo, añade en el vector de carreras del usuario, el identificador de la carrera elegida.
- **Desapuntarse de una carrera.** Esta funcionalidad es muy similar a la anterior, exceptuando que, al recibir la petición el servidor, comprueba que el usuario con la sesión iniciada esté inscrito en la carrera y en ese caso, la elimina de la lista de carreras de ese usuario.
- **Exportar calendario.** El usuario, con esta funcionalidad, puede exportar su calendario de carreras a cualquier aplicación de tipo *Calendar* como *Google Calendar* u *Outlook Calendar*. Para ello, al solicitarlo mediante una petición *AJAX*, el servidor recupera el listado de carreras qué el usuario está apuntado, genera un archivo *.ics* (que contiene eventos con toda la información de cada una de las carreras) y lo envía al cliente para que pueda descargarlo e importarlo en su aplicación *Calendar* preferida. Esta opción sólo está disponible si el usuario está inscrito a alguna carrera. Esta funcionalidad se encuentra explicada con más detalle en el Anexo E: Exportación del calendario
- **Compartir en redes sociales.** Si el usuario desea compartir en redes sociales alguna carrera a la que se ha apuntado, sólo necesita activar el botón de la red social

correspondiente. Esta funcionalidad no requiere de interacción con el servidor, toda su funcionalidad se ejecuta en la parte del cliente.

## 4.5 Quinta fase: Tareas aplazadas y nuevas funcionalidades

Esta quinta fase, al inicio del proyecto, se planificó para realizar diversas pruebas de usabilidad y accesibilidad con el objetivo de corregir los posibles fallos que pudiese tener la aplicación. Por necesidades del proyecto, en la quinta fase de este se llevan a cabo ciertas funcionalidades que no habían sido desarrolladas al no ser prioritarias (actualizar y borrar una cuenta, añadir una foto a la carrera) junto a nuevas ideas que han surgido a lo largo de su realización para mejorar el proyecto (mapa de carreras y carreras multi-distancia).

En primer lugar, para poder añadir una foto de portada a una carrera, es necesario modificar el esquema de la base de datos, añadiendo un nuevo campo que guarde la ruta a la fotografía. Así mismo, para la realización del mapa de carreras, se añade otro campo llamado *coordinates* que almacena la longitud y latitud de la carrera. También, para permitir que una carrera pueda tener varias distancias, es necesario modificar el campo *sport* y establecerlo como un *array* de cadenas. En la Figura 28 se pueden observar estos cambios:

```
imagePath:{
    type:String
},
coordinates:{
    type:mongoose.Schema.Types.Mixed
},
sport: {
    type: [String], required: true
}
```

*Figura 28. Modificación esquema races para implementar mejoras*

Para esta etapa, sólo es necesario desarrollar una nueva interfaz, que corresponde con la funcionalidad de ver el mapa de carreras. La Figura 29 muestra su boceto, dónde se puede observar una interfaz muy similar a la página de mostrar carreras con la única diferencia, que, en lugar de presentarlas en recuadros con toda su información, se muestran en el mapa según su localización y al seleccionarlas se despliega su información con el enlace para ir a su página. Esta nueva interfaz permite encontrar carreras de una manera más visual.

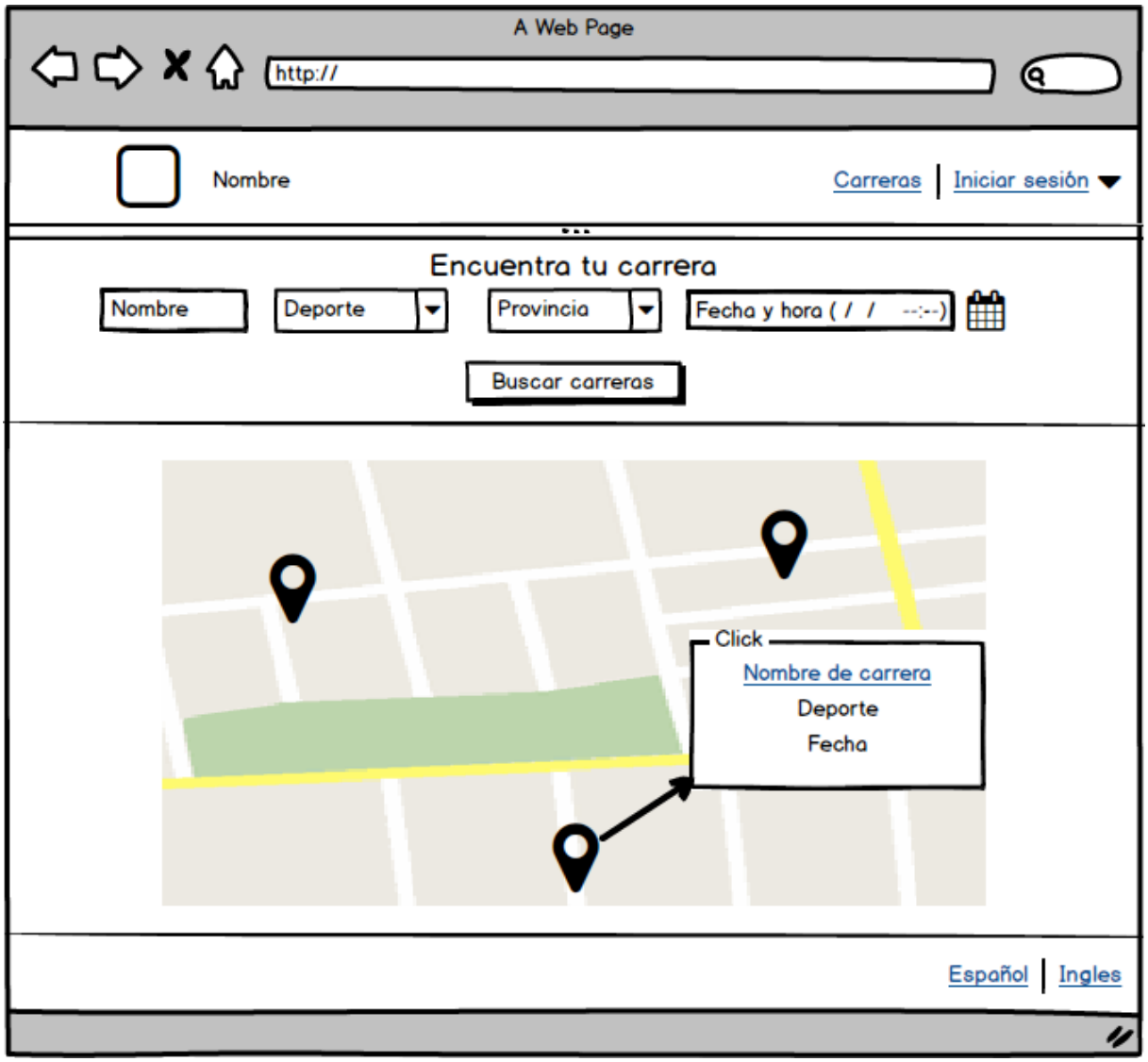


*Figura 29. Boceto página mapa de carreras*

De esta forma, el conjunto de funcionalidades que se desarrollan en esta iteración, junto con su explicación, es el siguiente:

- **Actualizar cuenta**. Si el usuario desea cambiar alguno de sus datos, debe acceder a la página de su perfil (siempre que se encuentre con la sesión iniciada), donde se muestra un formulario para modificar sus datos (véase Figura 19). Al activar el botón de guardar datos, se enviará una petición AJAX al servidor con los datos, éste la procesará y guardará en la base de datos los nuevos datos. En caso de haber cambiado la contraseña, al igual que al crear cuenta o recuperar contraseña, se almacena cifrada para no poner en riesgo su seguridad.
- **Eliminar cuenta**. Cuando el usuario solicita esta acción, se muestra un dialogo de confirmación por si ha pinchado el botón por error. En caso de desear eliminar

la cuenta, se realiza una petición al servidor, que comprueba quién es el usuario con la sesión activa y elimina su perfil de la base de datos.

- **Añadir foto a carrera**. Esta funcionalidad se puede dar en dos acciones distintas: al crear una nueva carrera o al modificar una carrera existente, aunque el funcionamiento es el mismo. En el formulario habilitado para ello (véase Figura 17, añadiendo un nuevo campo para seleccionar archivo), si se selecciona un archivo, al enviar la petición el servidor comprueba que se trate de un fichero de imagen y, en caso afirmativo, lo copia en su disco (en el directorio público del servidor para poderla mostrar a los usuarios) y guarda la ruta al archivo en el documento de esa carrera. Para no alterar el rendimiento de la página y su comportamiento, el tamaño del archivo es limitado.
- **Mapa de carreras**. Según el filtro utilizado por el usuario, el servidor devuelve la información de las carreras que coincidan, pero, en lugar de mostrarlas en formato lista, gracias a la API de Google Maps (véase Anexo C: API de Google Maps y Geocode), añade un marcador en la localización de cada carrera. A raíz de esta funcionalidad, se necesitan las coordenadas de cada carrera, puesto que, para añadir un marcador en el mapa, no se puede indicando una dirección postal, si no que necesita las coordenadas cartesianas. Para ello se puede utilizar la API de Geocoder (véase también Anexo C: API de Google Maps y Geocode), también de Google, que, indicando una dirección postal, devuelve la latitud y longitud. Con el objetivo de optimizar este proceso y no tener que realizar una petición a la API por cada carrera que haya que añadir al mapa, se decide realizar una sola petición (al crear o modificar la carrera) y guardar esas coordenadas en la base de datos.
- **Carreras multi-distancia**. Existen ciertas carreras que no solo tienen una distancia o modalidad, sino que se da la posibilidad al usuario de participar en varias. Por ejemplo, un organizador, el mismo día que se disputa el maratón, decide disputar también una media maratón y una carrera de 10Km. En este caso, para no tener tres carreras prácticamente con el mismo nombre y del mismo organizador, se permite la opción de indicar varios deportes en una misma carrera. Para ello, en el formulario de crear carrera o modificarla (véase Figura 17), se permite añadir varias opciones de la lista desplegable en lugar de seleccionar una sola.

## 4.6 Sexta fase: Pruebas de usabilidad y accesibilidad

En la última fase del proyecto se realizan diversas pruebas en la aplicación con el objetivo de encontrar y corregir los fallos que pueda haber en ella. Estas pruebas consisten en preparar un cuestionario para que los usuarios den su opinión acerca de una versión funcional de la aplicación y el empleo de diversas herramientas software que validen la usabilidad y accesibilidad del sitio web.

### 4.6.1 Cuestionario de usabilidad

Existen diversas técnicas para conocer si una página o aplicación web es usable, cómo la **Escala de Usabilidad del Sistema** (*SUS*) [31], [32], inventada por John Brooke en 1986 con el objetivo de evaluar la usabilidad de cualquier tipo de sistema de una forma rápida y barata.

Este sistema consiste en crear un cuestionario de 10 preguntas con cinco posibles respuestas: desde totalmente en desacuerdo hasta totalmente de acuerdo. A cada respuesta se le asigna una puntuación de 1 a 5, donde 1 significa totalmente en desacuerdo y 5 significa totalmente de acuerdo. El conjunto de preguntas elegido corresponde con la plantilla original (véase Figura 30) traducidas al castellano, porque si se realizase alguna modificación, habría que tener en cuenta ciertas consideraciones que afectan a los algoritmos que calculan el resultado.

1. I think that I would like to use this system frequently.
2. I found the system unnecessarily complex.
3. I thought the system was easy to use.
4. I think that I would need the support of a technical person to be able to use the system.
5. I found the varios functions in this system were well integrated.
6. I thought there was too much inconsistency in this system.
7. I would imagine that most people would learn to use this system very quickly.
8. I found the system very cumbersome to use.
9. I felt very confident using the system.
10. I needed to learn a lot of things before I could get going with this system.

*Figura 30. Plantilla de preguntas para el cuestionario de usabilidad (técnica SUS). Fuente [32]*

Para evaluar el cuestionario y obtener los resultados, según el sistema es necesario realizar un promedio en cada una de las preguntas con las siguientes consideraciones: las preguntas impares toman el valor asignado por el usuario y se le restará uno. En las pares, el valor corresponderá de la resta de 5 menos el valor asignado por el usuario. Posteriormente se suman todos los valores y se multiplican por 2,5 para obtener una valoración sobre 100. La media de las evaluaciones corresponde con 68 puntos,

significando que la aplicación es usable, pero se puede mejorar. Si se obtuviese una puntuación de 51 o inferior, la usabilidad de la aplicación pasaría a ser máxima prioridad en cualquier proyecto. A partir de 80,3 puntos significaría que a los usuarios les gusta mucho el sitio y que, posiblemente, se lo recomendarán a sus amigos.

A continuación, una vez explicado cómo funciona este sistema de evaluación, se presentan los resultados obtenidos por el cuestionario con un tamaño de muestra de 9 personas. Los resultados se han ordenado por preguntas, para calcular la puntuación de cada una de ella y, posteriormente, la general. Desde la Figura 31 hasta la Figura 40 se muestran los resultados de cada una de las preguntas.

El resultado final, una vez realizado los cálculos necesarios explicados anteriormente, es de 83,89 puntos, por lo que se puede afirmar que la aplicación es altamente usable para los usuarios.

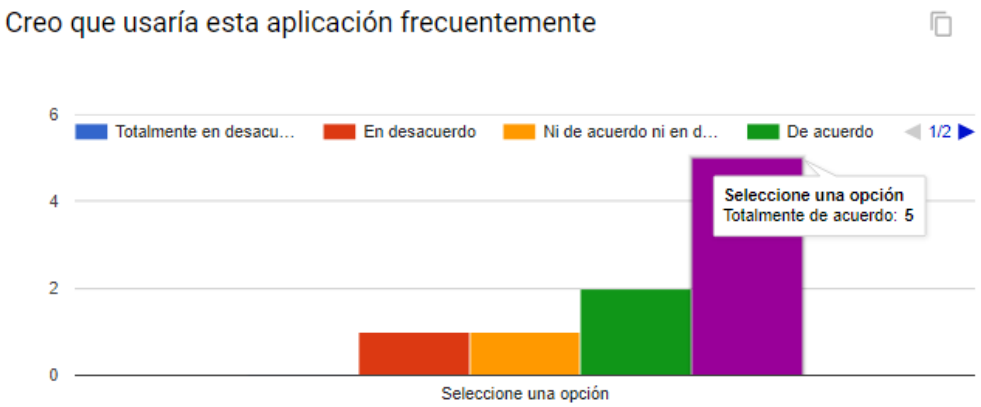


*Figura 31. Resultado 1ª pregunta encuesta de usabilidad.*

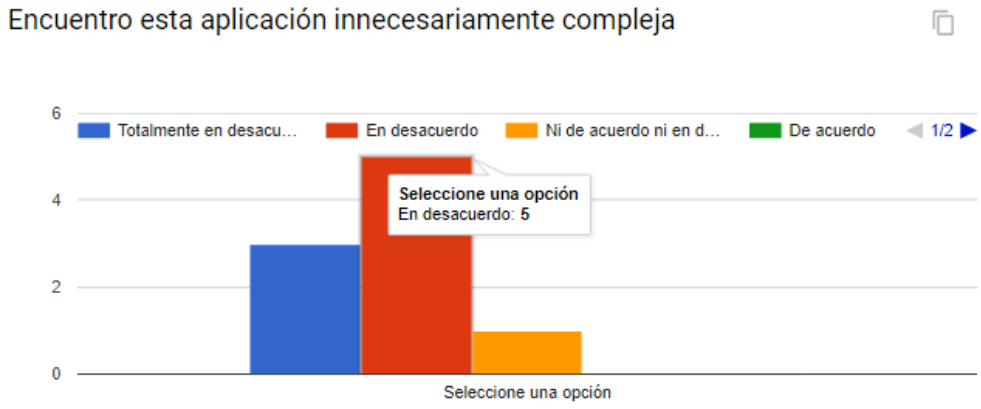


*Figura 32. Resultado 2ª pregunta encuesta de usabilidad.*

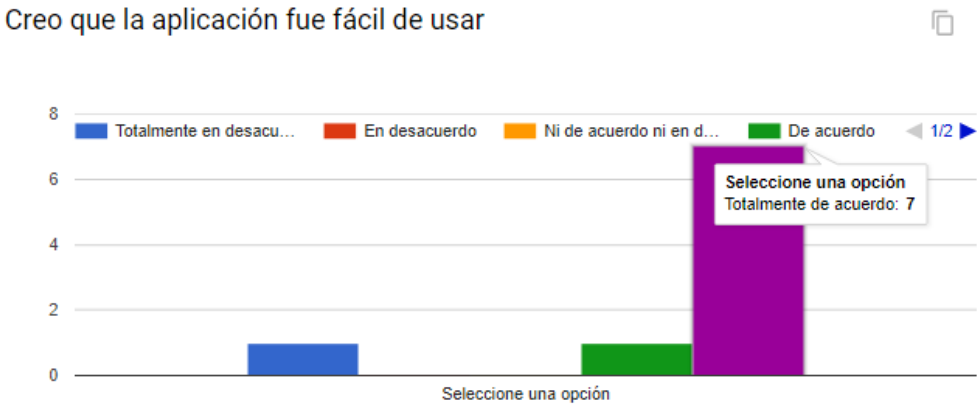


*Figura 33. Resultado 3ª pregunta encuesta de usabilidad.*


*Figura 34. Resultado 4ª pregunta encuesta de usabilidad.*

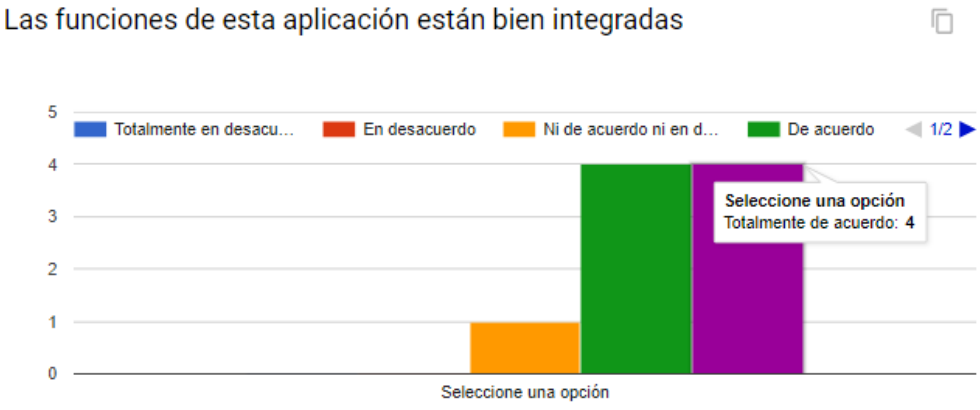


*Figura 35. Resultado 5ª pregunta encuesta de usabilidad.*


*Figura 36. Resultado 6ª pregunta encuesta de usabilidad.*

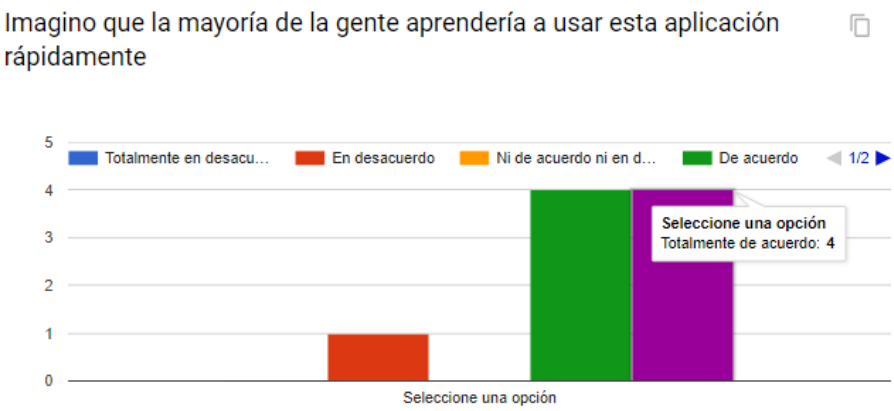


*Figura 37. Resultado 7ª pregunta encuesta de usabilidad.*

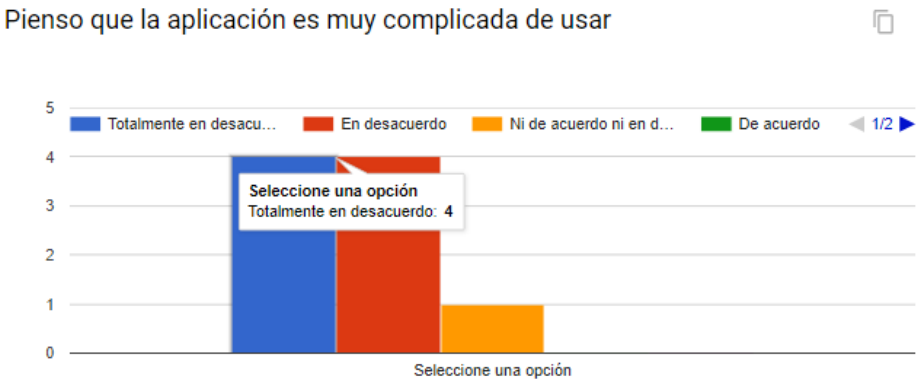


*Figura 38. Resultado 8ª pregunta encuesta de usabilidad.*

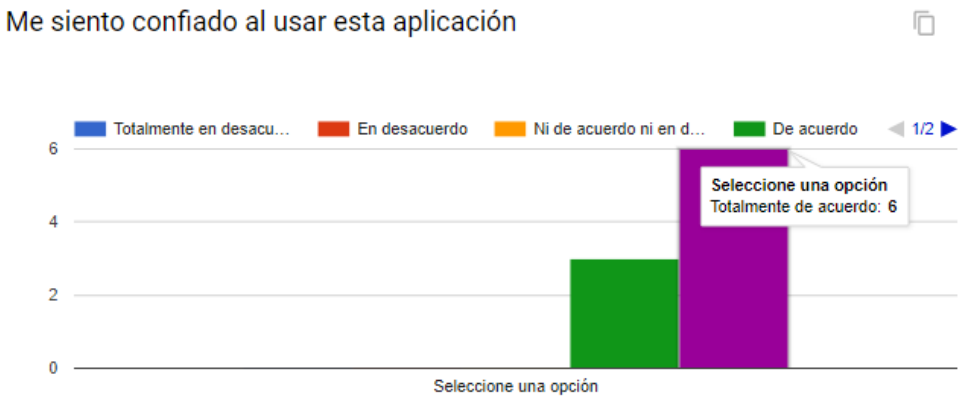


*Figura 39. Resultado 9ª pregunta encuesta de usabilidad.*

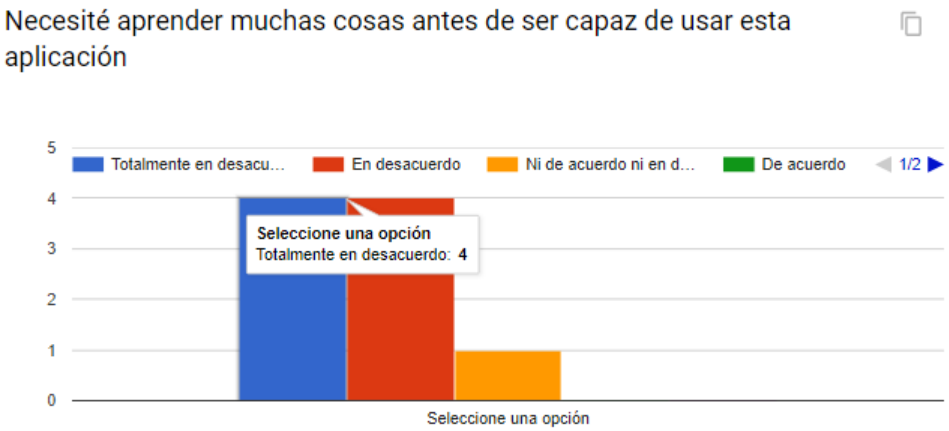


*Figura 40. Resultado 10ª pregunta encuesta de usabilidad.*

### 4.6.2 Evaluación de accesibilidad

Para realizar una evaluación de accesibilidad existen multitud de herramientas software *online* que realizan una búsqueda en profundidad de errores según ciertos principios. En este proyecto se van a utilizar tres de las más conocidas, *Taw (Test de Accesibilidad Web)* [33], *WAVE* (*Web Accesibility Evaluation Tool*) [34] y *AChecker* [35], basadas en los principios *WCAG 2.0* [36].

*WCAG 2.0* es un conjunto de pautas y recomendaciones para desarrollar contenido web de una forma más accesible. Siguiendo estas pautas, se podrá conseguir un contenido accesible para personas con diversas discapacidades, como problemas de visión o limitación de movimiento, puedan utilizar la página. Además, al disponer de una página web accesible, mejora también la usabilidad para usuarios generales. *WCAG 2.0* está basado en cuatro principios: perceptible, operable, comprensible, robusto, que a su vez contienen diversas pautas y recomendaciones para conseguir cumplir ese principio.

Al utilizar la herramienta TAW sobre la aplicación, se genera un informe (véase Figura 41) que indica los problemas encontrados en ella junto a advertencias y posibles errores cuya comprobación no puede realizarse con la herramienta, siendo necesario comprobación completamente manual.

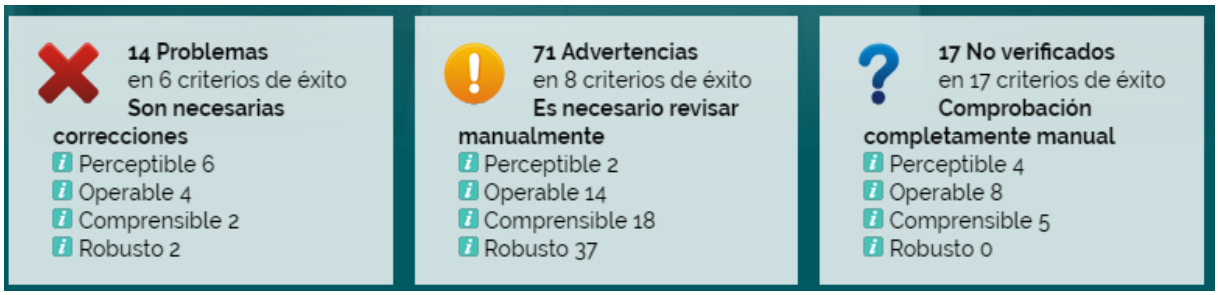


*Figura 41. Informe inicial de la herramienta TAW*

- **Pauta 1.3. Adaptable.** En la página existen elementos de entrada de texto que no tienen asociados ninguna etiqueta que los describa. La herramienta *Achecker* indica exactamente el lugar del problema (véase Figura 42). En este caso, la entrada de la contraseña no tiene asociada ninguna etiqueta que la describa.

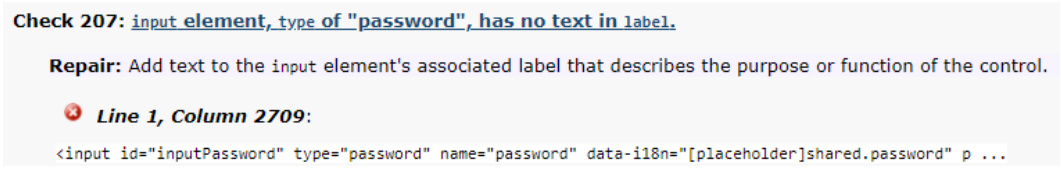


*Figura 42. Error de accesibilidad indicado por Achecker*

- **Pauta 1.4. Distinguible.** La herramienta *Achecker* muestra que existen elementos <*b*> e <*i*> en la aplicación. Estos elementos deben ser sustituidos por elementos <*em*> o <*strong*> para separar el contenido del formato visual. Además, ambas herramientas indican que pueden existir errores de contraste, pero es necesaria una comprobación manual. Gracias a la herramienta *WAVE*, tal como se puede ver en la Figura 43, se puede comprobar que existen tres errores de contraste en el pie de página (a simple vista, un usuario normal puede distinguir sin problema el texto del fondo, pero para un usuario con una discapacidad visual puede no diferenciarlos). Esta herramienta incluye una utilidad para poder corregirlos, indicando el color (en código hexadecimal) del fondo y del texto, informa si la combinación de colores es apropiada o no.


*Figura 43. Errores de contraste indicados por la herramienta WAVE*

- **Pauta 2.1. Accesible por teclado**. Las herramientas de evaluación no permiten comprobar esta pauta, por lo que se requiere comprobarla manualmente. Para ello, al intentar navegar por la página web a través del teclado (mediante el uso del tabulador y diversos atajos de teclado) se puede comprobar que se puede navegar por toda la página web. De esta forma, se puede afirmar que personas con discapacidades que les impiden utilizar el ratón (discapacidad visual o física con movilidad reducida), pueden utilizar la aplicación a través de otro periférico como el teclado.
- **Pauta 2.4. Navegable.** Esta pauta trata de proporcionar ayuda a los usuarios para que naveguen, encuentren contenido y determinen dónde están en cada momento. La herramienta *WAVE* informa que existen enlaces sin contenido y enlaces redundantes. En la Figura 44 se puede observar que los recuadros con la información de la carrera contienen dos errores, existe un enlace para la imagen

y otro para el texto (enlaces redundantes), estando el primero sin ningún contenido que pueda explicar qué es y a donde lleva.


*Figura 44. Errores en los enlaces identificados por la herramienta WAVE*

- **Pauta 3.3. Entrada de datos asistida.** Como se comentó en la pauta 1.3, las entradas de texto necesitan una etiqueta que las describa. Al no tenerla, pueden darse ocasiones que generen errores y el usuario no entienda que ha pasado. En este caso la entrada que está ocasionando el problema es la misma que en la Figura 42.
- **Pauta 4.1. Compatible.** Para verificar esta pauta, es necesario utilizar un validador HTML como el validador de *W3C* [37], que nos informa que nuestro código contiene 8 errores (véase Figura 45). El documento HTML debe estar bien formado para ser compatible con ayudas técnicas como los narradores web. El primero de ellos corresponde con el atributo *width* que no permite incluir valores con caracteres. Desde el segundo hasta el séptimo, nos informa que ningún enlace puede contener espacios. El último nos indica que no existe ningún elemento que corresponda a ese ID.

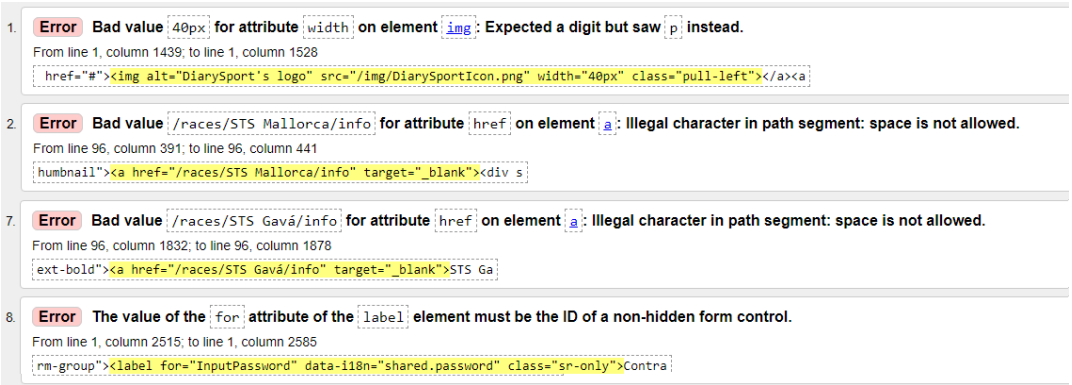


*Figura 45. Errores en la validación del documento HTML*

Una vez analizados y corregidos todos los errores identificados por las herramientas, el resultado es muy positivo. La Figura 46 muestra el resultado de realizar la validación del código HTML. La Figura 47 contiene el informe final de la herramienta *TAW*, mientras que la

Figura 48 corresponde con la herramienta *Achecker*, demostrando que todos los errores se han corregido con éxito.

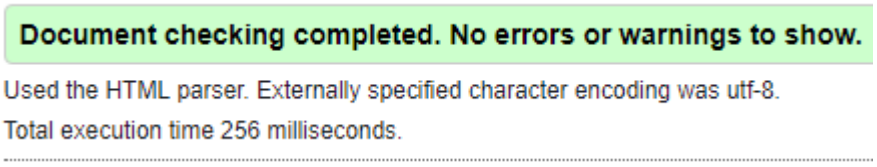


*Figura 46. Validación HTML de la aplicación*

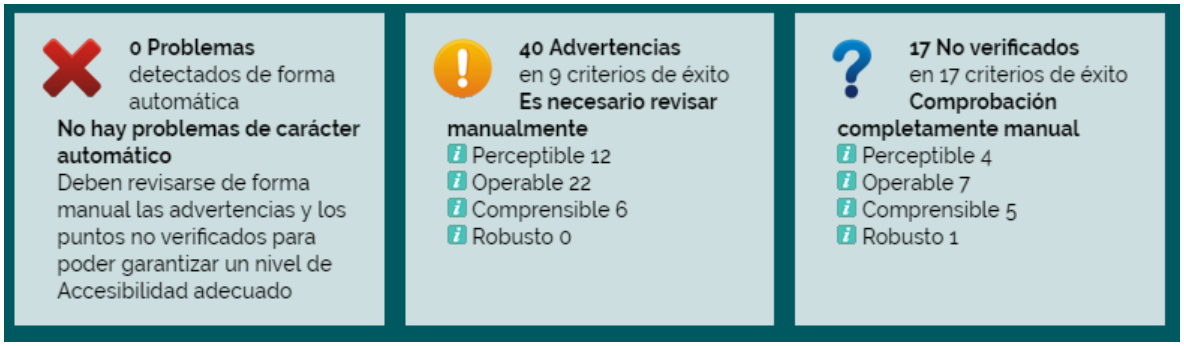


*Figura 47. Informe final de la herramienta TAW*

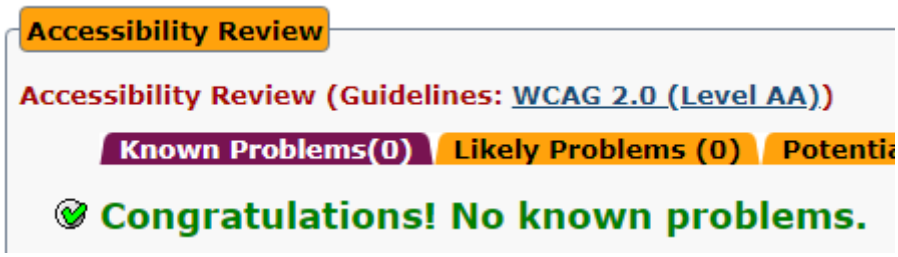


*Figura 48. Informe final de la herramienta Achecker*

# Capítulo 5: Resultados

En este apartado se exponen los resultados obtenidos con la realización de este trabajo, una vez terminadas todas las fases anteriormente descritas.

## 5.1 Arquitectura de la aplicación

Cómo se explicó en el Capítulo 3, el objetivo principal planteado para este TFM era el "desarrollo de una aplicación web que permita gestionar un listado unificado y global de carreras deportivas en toda España", con el fin de ayudar al deportista a encontrar su próximo reto. En la Figura 49 se muestra la arquitectura de la aplicación desarrollada. En este esquema se pueden observar las tres capas del modelo MVC con el conjunto de tecnologías que se utilizan en cada una.

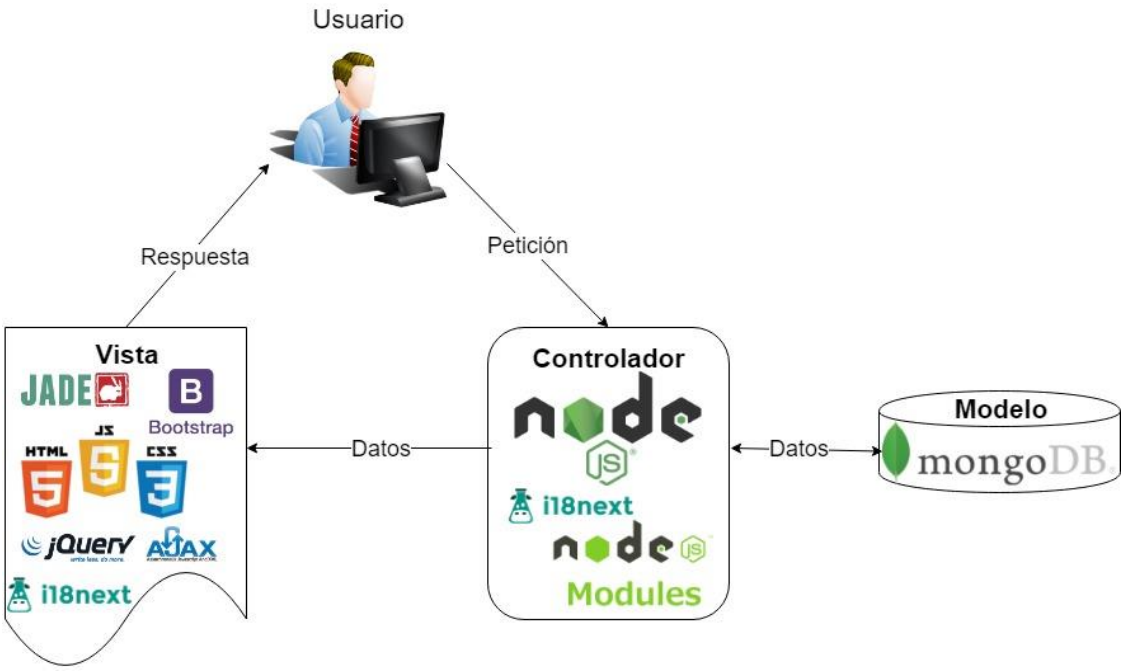


*Figura 49. Arquitectura de la aplicación*

En la parte superior del diagrama se encuentra el usuario, que puede acceder a la aplicación mediante cualquier dispositivo que tenga un navegador y conexión a internet. Este usuario interactúa con la página web (en el lado del cliente) y envía peticiones al controlador, que las recibe y procesa y, en caso de necesitar datos almacenados previamente en la base de datos, accede a ella y los transfiere a la vista, que renderiza la nueva página con los datos recibidos desde el controlador. Esta vista se muestra al usuario para que interactúe con ella y vuelva a realizar todo este proceso en caso de desearlo.

## 5.2 Aplicación desarrollada

En primer lugar, en relación con la primera fase del proyecto, la aplicación permite al usuario crear una cuenta personal mediante un correo electrónico y contraseña para poder **gestionar su calendario** de carreras. En la Figura 50 se observa la página principal de la aplicación, que muestra las tres próximas carreras que se encuentran registradas en ella. Además, en la parte superior derecha se encuentra la opción de iniciar sesión. Por último, en la parte inferior derecha, se incluyen dos enlaces para cambiar el idioma entre castellano e inglés, con el fin de permitir al mayor número de usuarios el uso de la aplicación y uno central que lleva al formulario de contacto.

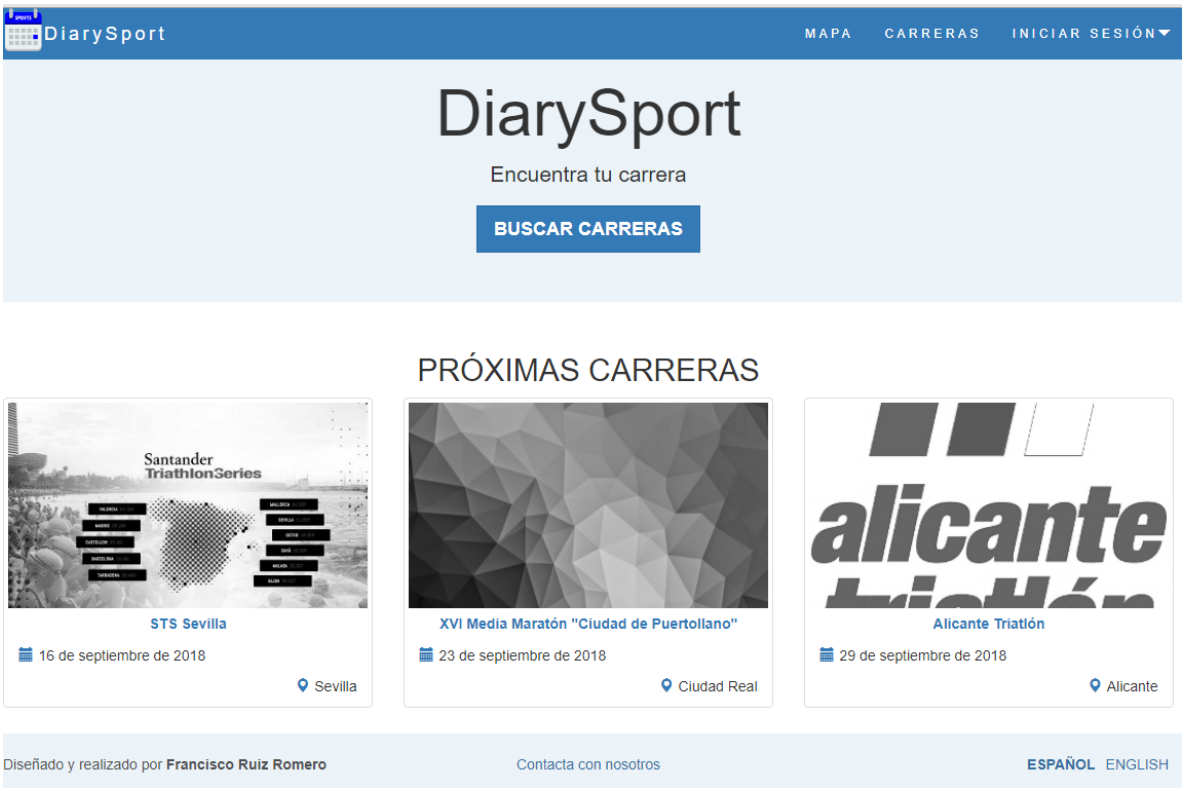


*Figura 50. Página principal de la aplicación.*

En la Figura 51 se muestra la página inicial con los formularios de crear cuenta, inicio de sesión y recuperar contraseña desplegados. Para hacer uso de la aplicación no es necesario registrarse, pero para ciertas funciones, como apuntarse a carreras o crear alarmas, es imprescindible haber iniciado sesión en la cuenta creada con anterioridad.


*Figura 51. Formulario de registro, inicio de sesión y recuperación de contraseña*

Si un usuario ha olvidado su contraseña para iniciar sesión, tiene la opción de recibir un correo electrónico para recuperarla, indicando su correo electrónico en el formulario habilitado para ello (véase Figura 51). El correo electrónico que el usuario recibe es similar al de la Figura 52, dependiendo del idioma que tenga elegido.

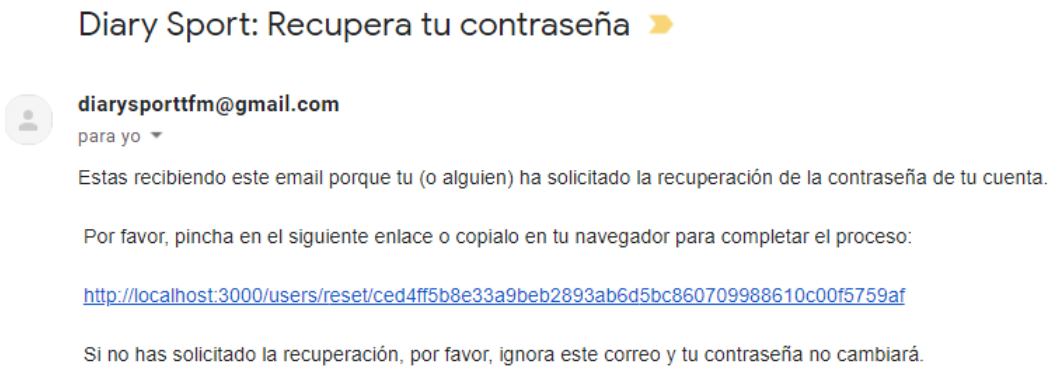


*Figura 52. Correo electrónico para recuperar la contraseña.*

El enlace del correo redirige a la página con el formulario de restablecer la contraseña (véase Figura 53), que contiene dos entradas de texto para definir la nueva contraseña.

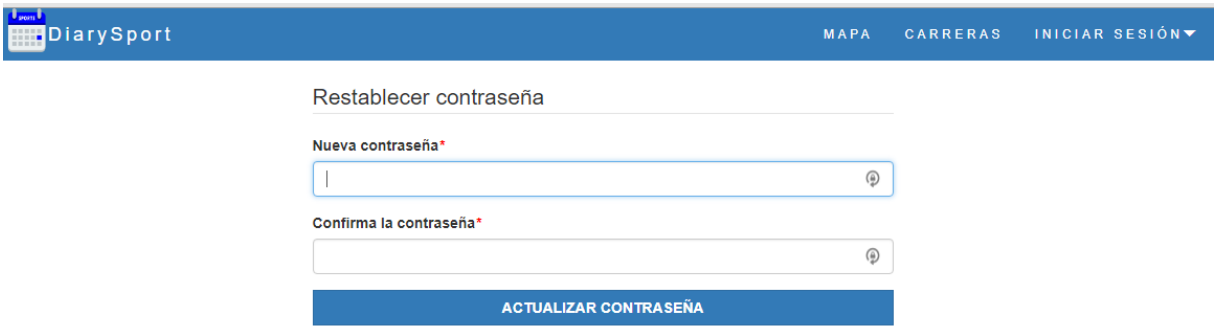


*Figura 53. Formulario de restablecer la contraseña*

En la parte superior de la página de carreras (véase Figura 54), se presenta un formulario que permite filtrar las carreras según los criterios ya explicados. En la parte central se muestran las carreras que coinciden con ese filtro. Si ninguna carrera coincide con el filtro definido, se muestra al usuario la opción de crear una alarma que le notifique cuando se registre una que si coincida a través de un correo electrónico (véase Figura 55).

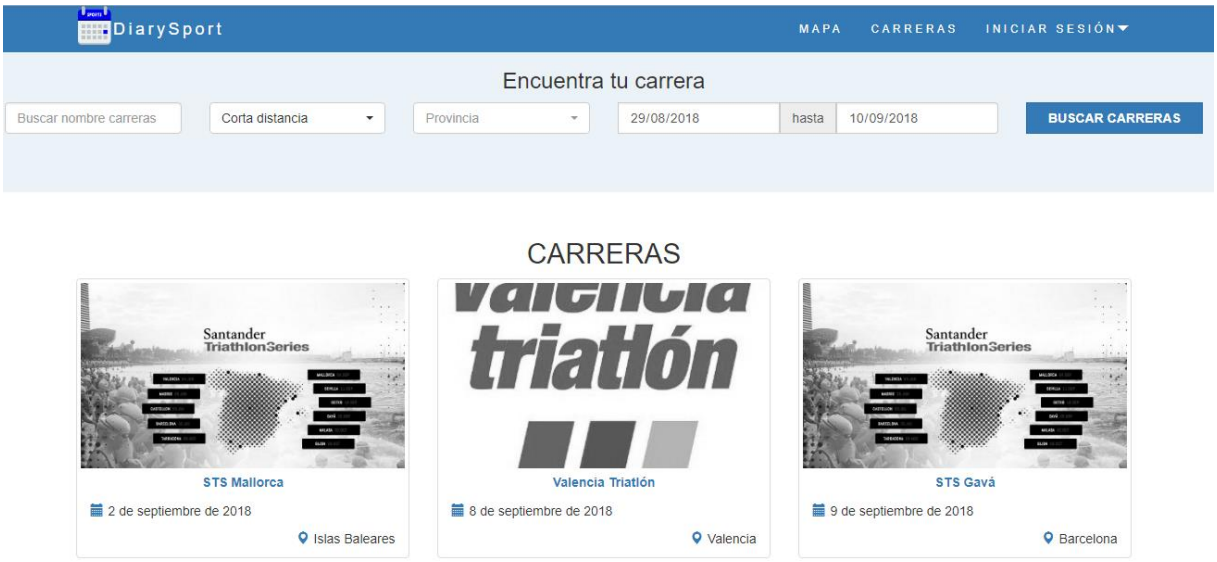


*Figura 54. Página del listado de carreras*

Si el usuario no se encontrase con la sesión iniciada, no aparecería el botón de crear alarma y, en su lugar, aparecería un mensaje que le informaría que debe iniciar sesión para crear una alarma.


*Figura 55. Opción de crear una alarma en la página de carreras*

La Figura 56 muestra la página de una carrera, una vez ha sido seleccionada. Esta página contiene toda la información relativa a esa carrera, la posibilidad de compartirla en redes sociales y, si el usuario ha iniciado sesión, se le permite apuntarse a ella. Si el usuario se encontrase apuntado, el botón cambiaría y sería para desapuntarte.

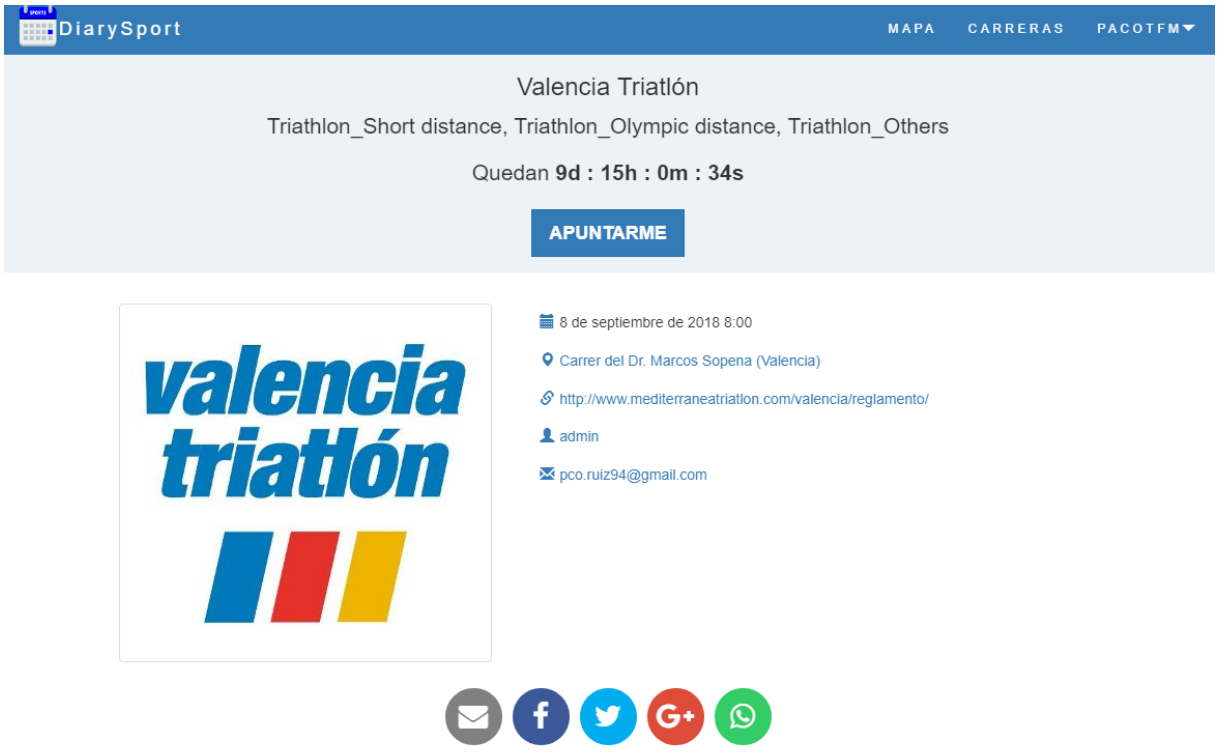


*Figura 56. Página con la información de una carrera*

Una página muy similar a la de carreras es la de carreras de un organizador, que muestra todas las carreras que pertenecen a un mismo organizador (véase Figura 57).

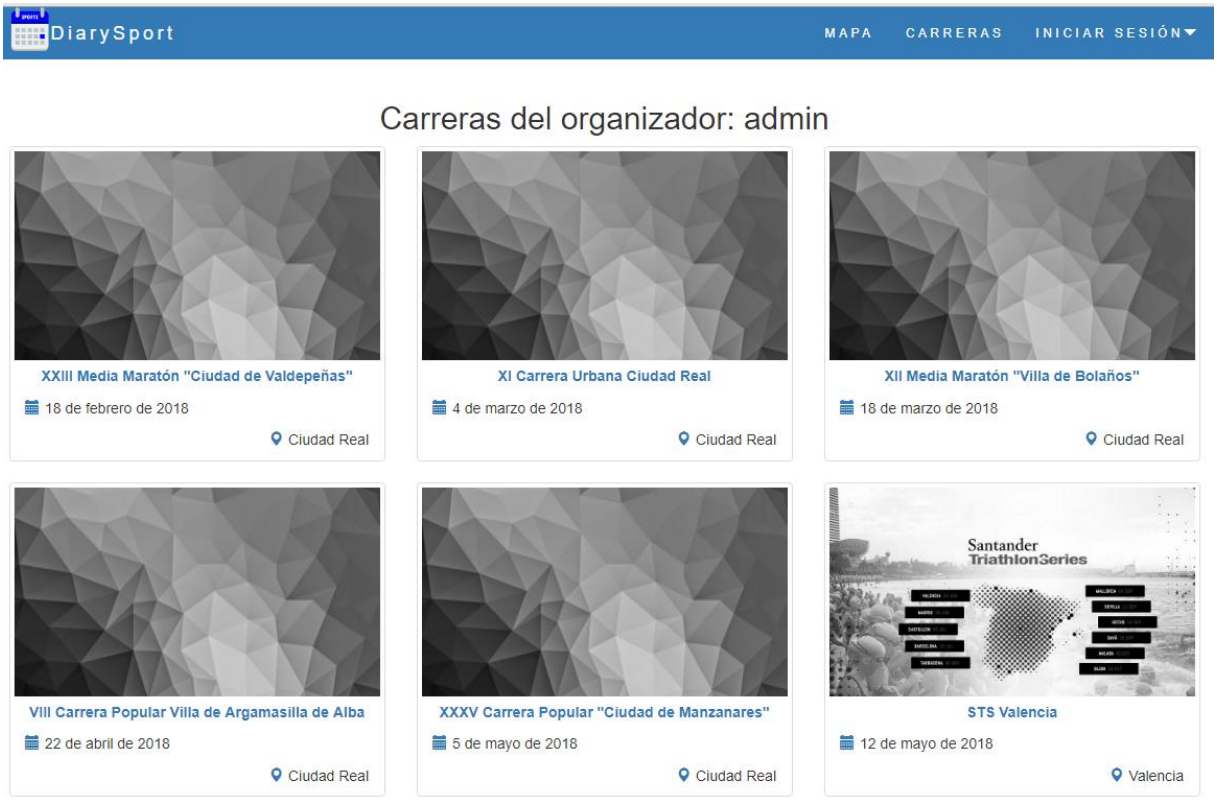


*Figura 57. Página de carreras de un organizador*

En la Figura 58 se presenta la página de perfil de un usuario, a la que se accede desplegando el menú superior y seleccionando la opción *perfil*. En esta página se incluye un formulario para editar los datos de perfil y un botón para eliminar la cuenta del usuario. En la parte derecha se listan todas las carreras a las que el usuario se ha apuntado con la opción de exportar su calendario a una aplicación de tipo *Google Calendar*. Además, en cada una de las carreras, se permite al usuario que se desapunte, al igual que en la página de una carrera individual.

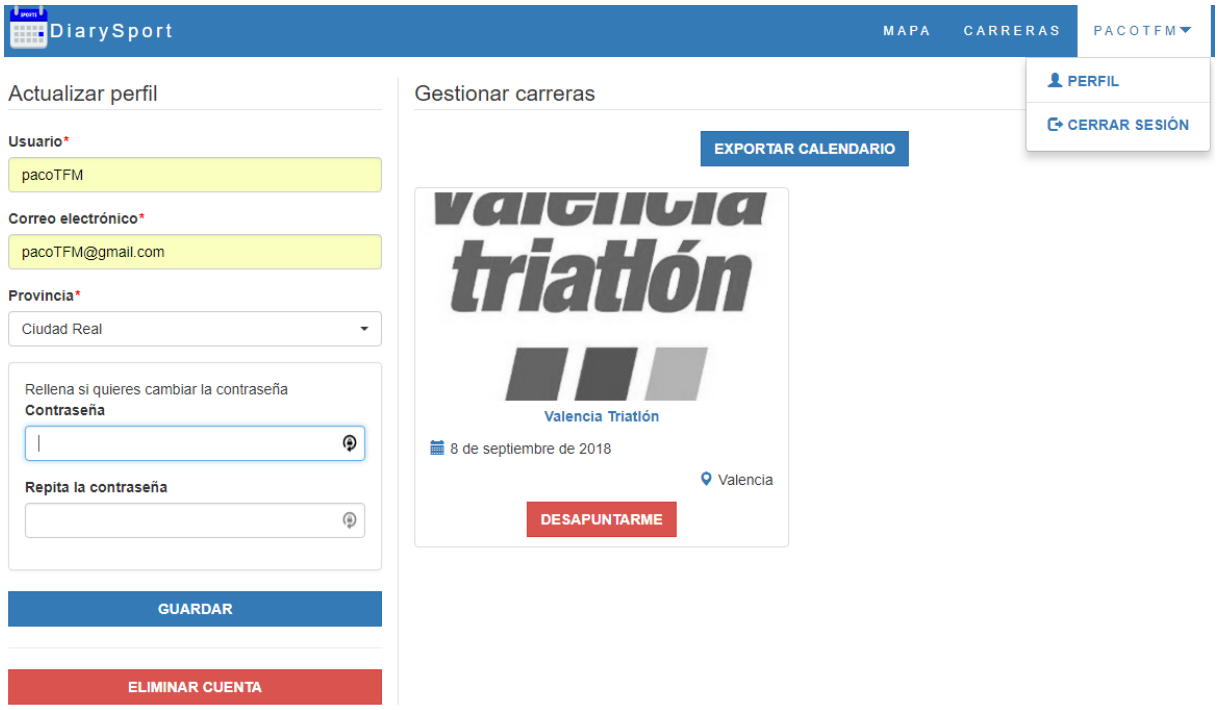


*Figura 58. Página de perfil de un usuario deportista*

Si el usuario con la sesión iniciada fuese un organizador, la página de perfil sería diferente, como en la Figura 59. En esta imagen varía el apartado de carreras, que contiene todas las carreras que el usuario ha creado, pudiendo modificarlas o borrarlas. En la parte superior de gestionar carreras, aparecen dos botones que redirigen a las páginas de crear e importar carreras.

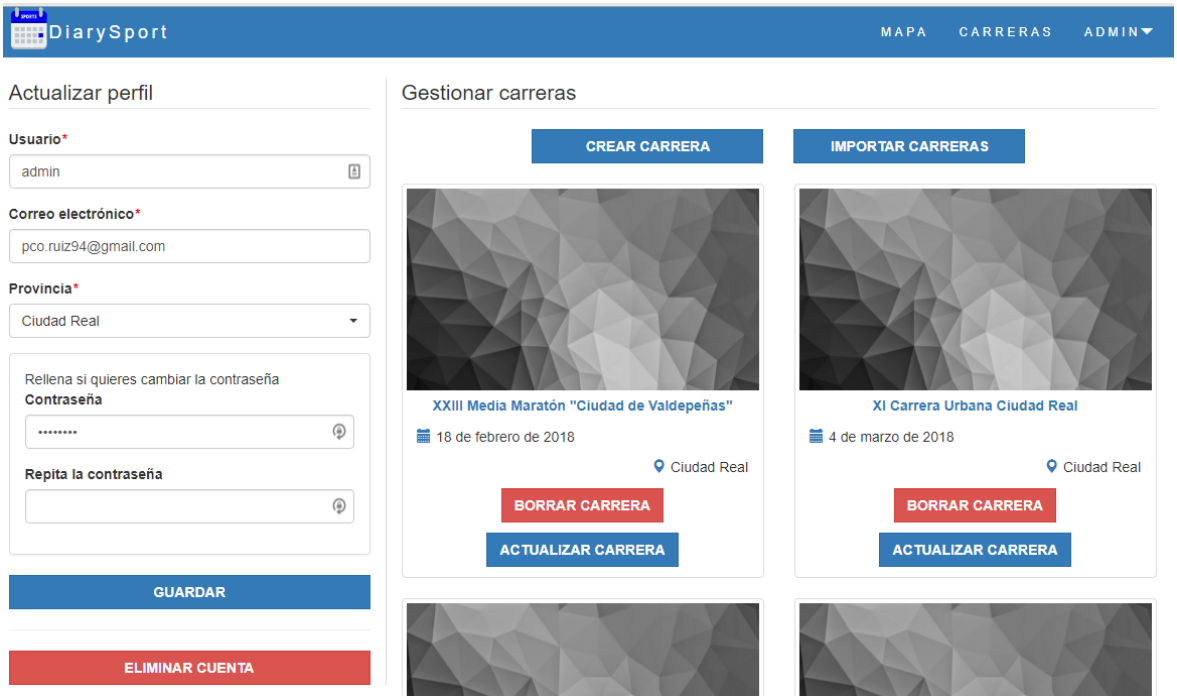


*Figura 59. Página de perfil de un usuario organizador*

La página de crear una carrera (véase Figura 60) incluye un formulario con todos los datos necesarios para crearla, incluyendo dos desplegables, uno para seleccionar el deporte y otro para la provincia. Además, se añade un botón para seleccionar una foto y establecerla como imagen de la carrera. La página para modificar una carrera es muy similar exceptuando el título del formulario, que sería “Actualizar carrera” y todos los campos estarían rellenos con los datos de la carrera.

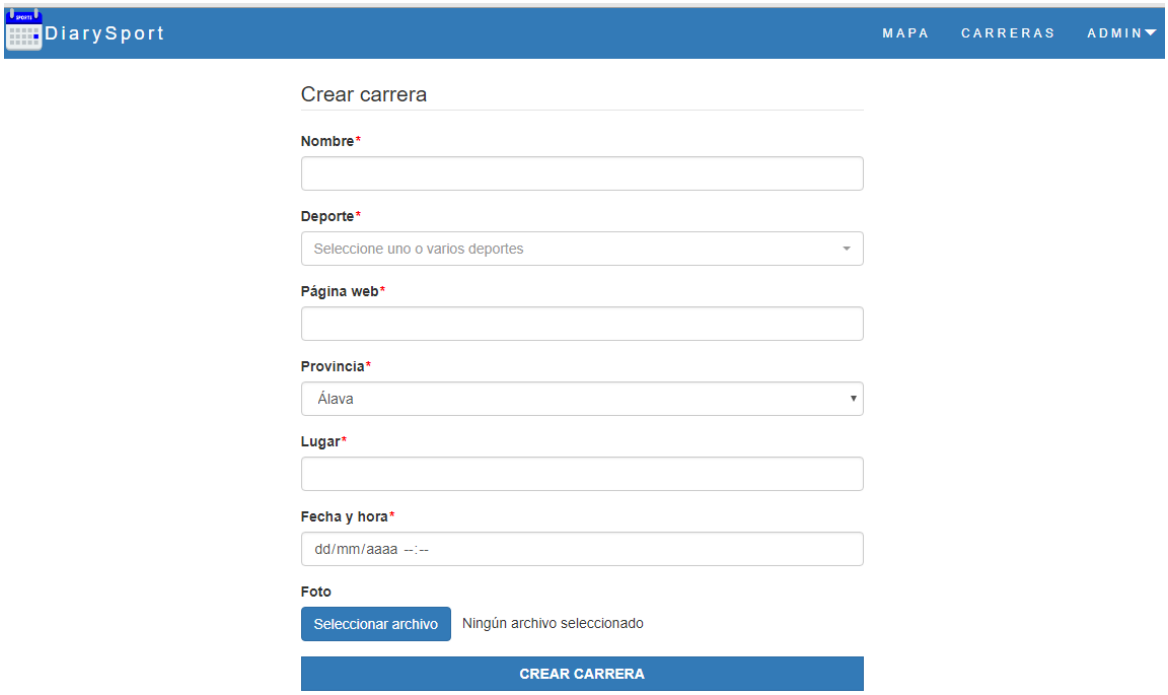


*Figura 60. Página para crear una carrera*

Por otro lado, la página de importar carreras (véase Figura 61) permite descargar el archivo Excel con la plantilla para importarlas. Una vez rellena con todas las carreras que se deseen añadir, el usuario la mandaría al servidor a través del formulario de la página. El botón para importar carreras sólo se activa al seleccionar un archivo válido.

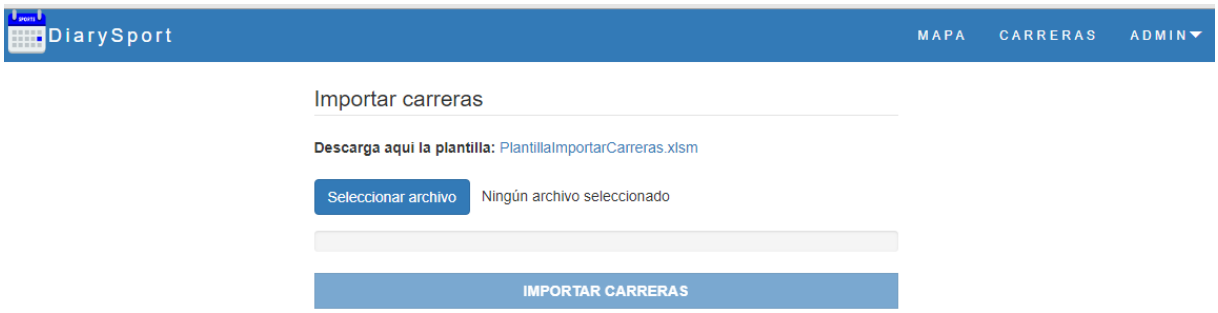


*Figura 61. Página para importar carreras*

En la Figura 62 se observa la página que permite, de forma visual, filtrar carreras. En esta página se incluye, el mismo formulario para filtrar carreras y, en la parte inferior, se añade un mapa que incluye un marcador por cada carrera que coincide con la búsqueda. Además, estos marcadores incluyen información sobre la carrera, que, al ser seleccionados, despliegan un recuadro para mostrarla.

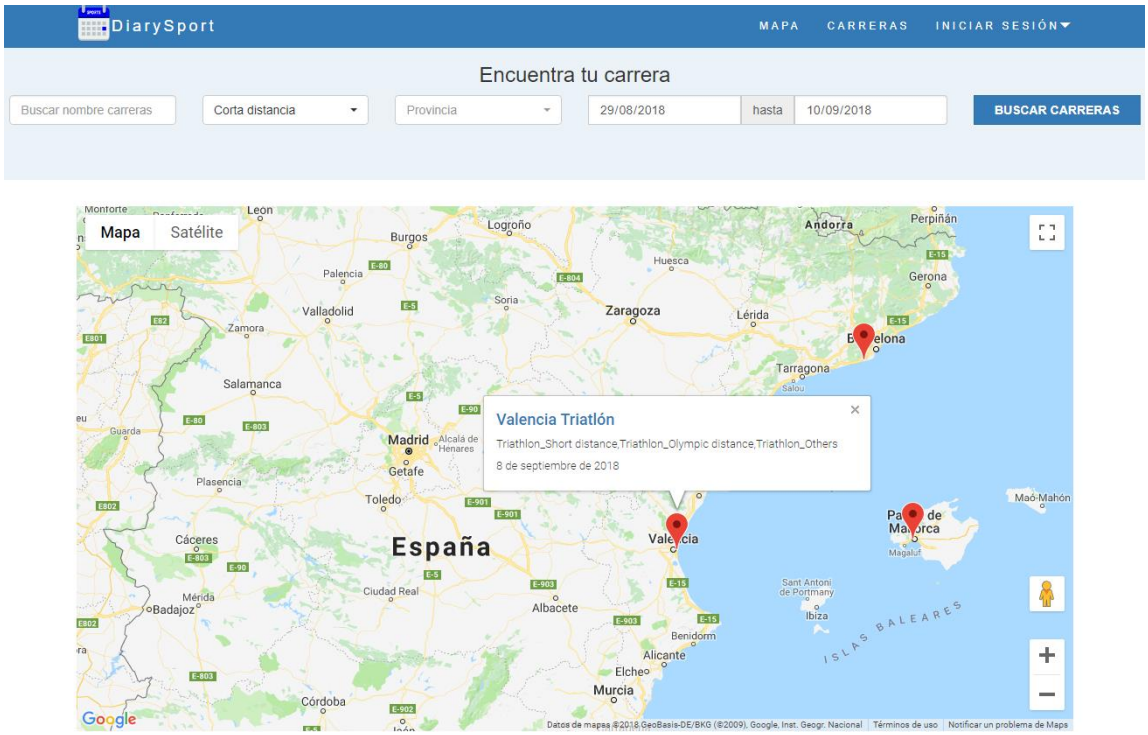


*Figura 62. Página con el mapa para mostrar las carreras*

Además, la página contiene un formulario para ponerse en contacto con el administrador de esta (véase Figura 63). En este formulario se puede introducir el nombre, correo electrónico y mensaje que el usuario desee enviar.

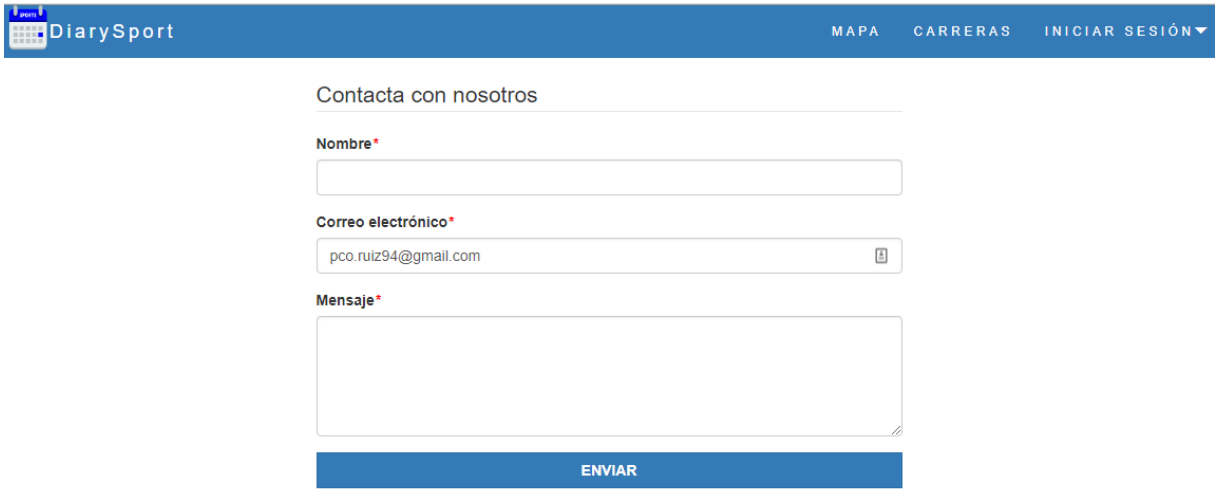


*Figura 63. Página de contacta con nosotros*

Para facilitar la tarea al usuario y mejorar su experiencia de uso, la aplicación posee diversos mecanismos de ayuda, como mensajes de confirmación al eliminar (véase Figura 64), *feedback* al usuario tras realizar una acción (véase Figura 65 y Figura 66) o mensajes de error al intentar realizar una opción no permitida (véase Figura 67).

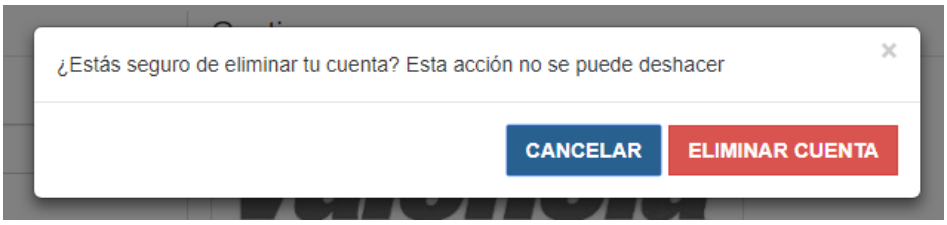


*Figura 64. Mensaje de confirmación al solicitar la eliminación de la cuenta*


*Figura 65. Feedback proporcionado al usuario en el formulario de inicio de sesión.*

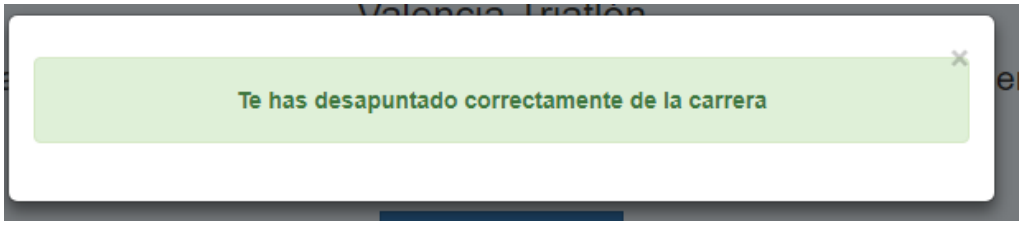


*Figura 66. Feedback positivo proporcionado al usuario al solicitar desapuntarse de una carrera*

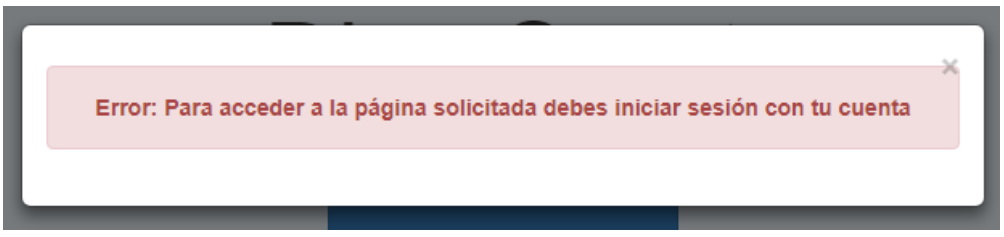


*Figura 67. Mensaje de error al acceder a una zona no autorizada*

Con el objetivo de desarrollar una herramienta usable desde cualquier dispositivo, se la ha dotado de la capacidad de adaptarse a la pantalla de cada uno (siendo *responsive*). La Figura 68 muestra la página de perfil de usuario, pero adaptada a un teléfono móvil.

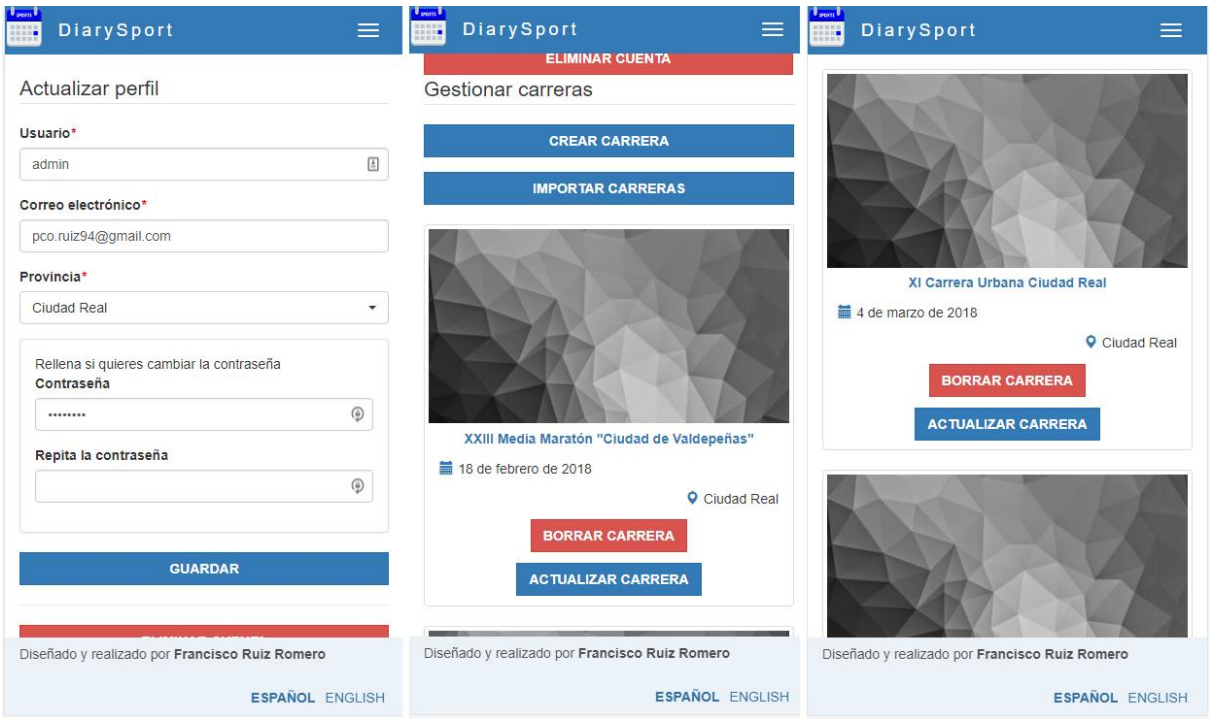


*Figura 68. Página de perfil de usuario adaptada a un dispositivo móvil*

Por último, cabe destacar que, en algunas funcionalidades, desde los dispositivos móviles, se ocultan parcialmente. Este es el caso del inicio de sesión, puesto que la cabecera de la aplicación se oculta dando lugar a un menú desplegable que se activa al pulsar el botón de la esquina superior derecha. En la Figura 69 se muestran los pasos a seguir para, en la página principal de la herramienta, acceder a la funcionalidad de inicio de sesión.

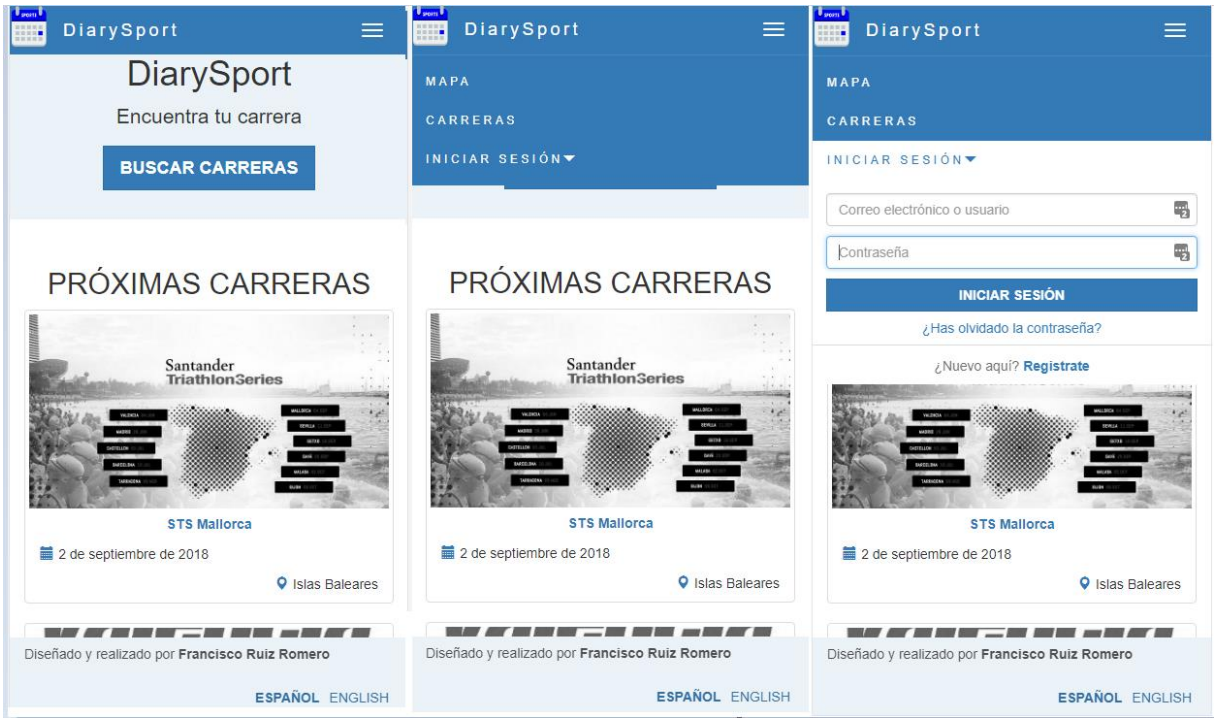


*Figura 69. Página principal con la funcionalidad inicio de sesión desde un dispositivo móvil*

# Capítulo 6: Conclusiones y Trabajos futuros

A lo largo de este apartado se hace una reflexión sobre el trabajo realizado, obteniendo una serie de conclusiones personales con la realización de este proyecto. Además, también se enumeran los objetivos planteados en este TFM y cómo se han alcanzado, junto a una serie de propuestas para complementar esta aplicación y dotarla de mayor funcionalidad.

## 6.1 Conclusiones

En este TFM se ha desarrollado una novedosa aplicación web que permite gestionar un listado unificado y global de carreras deportivas en toda España.

De forma más específica, se ha desarrollado una aplicación funcional y libre de fallos para la gestión de carreras deportivas. Además, esta aplicación incluye la gestión de cuentas de usuario con el objetivo de personalizar la aplicación a cada usuario que la utilice.

La aplicación permite crear alertas personalizadas para el usuario, de forma que se le notifique cuando se añada una carrera que coincida con un filtro de búsqueda empleado.

También se permite a los organizadores de carreras importar de forma manual o masiva los eventos deportivos, mediante un formulario en la aplicación o una plantilla Excel donde pueden introducir todas sus carreras.

Para facilitar su uso al usuario, la aplicación se encuentra internacionalizada en dos idiomas (castellano e inglés), es usable y accesible, y se adapta a cualquier tipo de dispositivo, siendo así *responsive*.

Por todo lo anterior, puede decirse que se han cumplido con éxito todos los objetivos propuestos en este trabajo.

## 6.2 Valoración personal

El desarrollo de este proyecto ha requerido poner en práctica muchos de los conocimientos obtenidos durante la realización del Grado y Máster. También ha sido necesario profundizar e investigar en el tema para conseguir alcanzar los objetivos requeridos. De la misma forma, ha sido necesario el aprendizaje de nuevas técnicas y lenguajes de programación.

Antes de iniciar este proyecto, no sabía sobre qué lo iba a hacer por lo que me puse a pensar alguna necesidad existente y que pudiese darle solución. Como soy una persona deportista aficionado a los triatlones y carreras populares, se me ocurrió la idea de crear la aplicación para resolver un problema que me había ocurrido en bastantes ocasiones, el no saber qué carreras se disputan, dónde y cuándo.

Una vez definida la idea, el siguiente paso era seleccionar qué tecnologías se iban a utilizar para llevarla a cabo y, con el objetivo de formarme más en el desarrollo web, elegí *MongoDB* y *NodeJS*, las cuales había utilizado muy por encima durante alguna asignatura del Máster, pero eran muy distintas a las aplicaciones *Java* y *MySQL* que había utilizado durante el Grado y el Trabajo Fin de Grado.

Además, he podido comprobar que, diseñando las interfaces de usuario con prototipos de alto nivel de abstracción, como los bocetos, es muy beneficioso ya que cualquier cambio no requiere prácticamente ningún coste, y, si ya se encontrasen implementadas, el cambio si llevaría más trabajo. *Bootstrap*, *HTML* y *CSS* ofrecen muchas funcionalidades para la creación de las interfaces y hubiera sido muy costoso desarrollar el *front-end* de la aplicación sin partir de un diseño claro.

De igual modo, para este proyecto se ha utilizado una nueva metodología de desarrollo nunca usada durante el Grado y Máster. Se ha podido comprobar que esta metodología es muy apta para proyectos no muy grandes que pueden tener requisitos o prioridades cambiantes.

A modo personal, siento una profunda satisfacción al haber conseguido terminar este proyecto, tanto por haber realizado una aplicación que da solución a un problema real, como por haberme ayudado a formarme más específicamente en las tecnologías y desarrollo web y poner un punto de partida en mi carrera como profesional.

## 6.3 Propuestas de trabajo futuro

En este apartado se exponen un conjunto de propuestas que complementarían a la aplicación para otorgarle mayor funcionalidad y aumentarían su ámbito de uso. Estas mejoras han ido surgiendo a lo largo del proyecto, tanto de ideas propias como recomendaciones y sugerencias realizadas por diferentes personas que han hecho uso de la aplicación:

- Incluir más categorías y distancias en la aplicación.
- Crear un apartado de carreras disputadas donde se pueda visualizar la clasificación, resultados o fotos.
- Añadir un sistema de comentarios o valoraciones de las carreras para poder establecer un ranking de pruebas deportivas.
- Dotar al sistema de un sistema para indicar la dificultad de las carreras.
- Incluir fechas de inscripciones a carreras y cambios de precio.
- Añadir nuevos filtros como precio de inscripción o si están homologadas o no.
- Agregar nuevas alarmas para enviar aviso por cambio de precio o cierre de inscripciones y permitir elegir frecuencia de las alarmas.
- Almacenar las imágenes de carreras en la nube.

# Referencias bibliográficas

[1] «Sportmaniacs — Calendario de carreras, inscripciones y clasificaciones». [En línea]. Disponible en: https://sportmaniacs.com/es. [Accedido: 11-sep-2018].

[2] «Calendario de Eventos», *https://web.rockthesport.com*. [En línea]. Disponible en: https://web.rockthesport.com. [Accedido: 11-sep-2018].

[3] «Atletas del Sol - Atletismo, Calendario y Resultados». [En línea]. Disponible en: http://www.atletasdelsol.com/. [Accedido: 11-sep-2018].

[4] «Portada - carreraspopulares.com | Calendario nacional de carreras populares». [En línea]. Disponible en: http://www.carreraspopulares.com/WR_01_index.asp. [Accedido: 11-sep-2018].

[5] «Correr en La Rioja - Noticias sobre el running en La Rioja», *Correr en La Rioja*. [En línea]. Disponible en: http://www.correrenlarioja.com/. [Accedido: 11-sep-2018].

[6] «Runedia». [En línea]. Disponible en: https://runedia.mundodeportivo.com/. [Accedido: 11-sep-2018].

[7] Ben Frain, *Responsive Web Design with HTML5 and CSS3*. Packt Publishing, 2012.

[8] J. Nielsen, *Designing Web Usability: The Practice of Simplicity*, First Edition. New Riders, 1999.

[9] Digité, «What is Kanban? Comprehensive Overview of the Kanban Method for Knowledge Teams», *Digité*. [En línea]. Disponible en: https://www.digite.com/kanban/what-is-kanban/. [Accedido: 28-jul-2018].

[10] Carmen Lasa Gómez, Alonso Álvarez García, y Rafael de las Heras del Dedo, *Métodos Ágiles. Scrum, Kanban, Lean*. ANAYA, 2017.

[11] «Qué es Kanban y cómo utilizarlo en el desarrollo de proyectos», *Blog de IEBSchool*, 31-jul-2013. [En línea]. Disponible en: https://www.iebschool.com/blog/metodologia-kanban-agile-scrum/. [Accedido: 28-jul-2018].

[12] «Windows 10 - Microsoft Store España». [En línea]. Disponible en: https://www.microsoft.com/es-es/store/b/windows. [Accedido: 10-sep-2018].

[13] «MongoDB Hosting: Database-as-a-Service by mLab», *mLab*. [En línea]. Disponible en: https://mlab.com/. [Accedido: 10-sep-2018].

[14] «EvenNode - Node.js and Python web hosting». [En línea]. Disponible en: https://www.evennode.com/. [Accedido: 10-sep-2018].

[15] «WebStorm: The Smartest JavaScript IDE by JetBrains», *JetBrains*. [En línea]. Disponible en: https://www.jetbrains.com/webstorm/. [Accedido: 10-sep-2018].

[16] F. de Node.js, «Node.js», *Node.js*. [En línea]. Disponible en: https://nodejs.org/es/. [Accedido: 10-sep-2018].

[17] «Free JavaScript training, resources and examples for the community». [En línea]. Disponible en: https://www.javascript.com/. [Accedido: 10-sep-2018].

[18] J. F.- js.foundation, «jQuery». .

[19] M. O. contributors Jacob Thornton, and Bootstrap, «Bootstrap». [En línea]. Disponible en: https://getbootstrap.com/. [Accedido: 10-sep-2018].

[20] «MongoDB for GIANT Ideas», *MongoDB*. [En línea]. Disponible en: https://www.mongodb.com/index. [Accedido: 10-sep-2018].

[21] «Introduction - i18next documentation». [En línea]. Disponible en: https://www.i18next.com/. [Accedido: 26-ago-2018].

[22] «Jade - Template Engine». .

[23] «Balsamiq. Rapid, effective and fun wireframing software. | Balsamiq». [En línea]. Disponible en: https://balsamiq.com/. [Accedido: 10-sep-2018].

[24] «Create UML diagrams online in seconds, no special tools needed.» [En línea]. Disponible en: https://yuml.me/. [Accedido: 10-sep-2018].

[25] «Word 2016 - Microsoft Store España». [En línea]. Disponible en: https://www.microsoft.com/es-es/store/b/word_2016. [Accedido: 10-sep-2018].

[26] «Sublime Text - A sophisticated text editor for code, markup and prose». [En línea]. Disponible en: https://www.sublimetext.com/. [Accedido: 10-sep-2018].

[27] «Zotero | Your personal research assistant». [En línea]. Disponible en: https://www.zotero.org/. [Accedido: 10-sep-2018].

[28] «Git». [En línea]. Disponible en: https://git-scm.com/. [Accedido: 10-sep-2018].

[29] Atlassian, «Bitbucket | The Git solution for professional teams», *Bitbucket*. [En línea]. Disponible en: https://bitbucket.org. [Accedido: 10-sep-2018].

[30] «MVC (Model, View, Controller) explicado.», *CódigoFacilito*. [En línea]. Disponible en: https://codigofacilito.com/articulos/mvc-model-view-controller-explicado. [Accedido: 28-jul-2018].

[31] A. S. for P. Affairs, «System Usability Scale (SUS)», 06-sep-2013. [En línea]. Disponible en: /how-to-and-tools/methods/system-usability-scale.html. [Accedido: 28-ago-2018].

[32] «How To Use The System Usability Scale (SUS) To Evaluate The Usability Of Your Website», *Usability Geek*, 13-jul-2015. [En línea]. Disponible en: https://usabilitygeek.com/how-to-use-the-system-usability-scale-sus-to-evaluate-the-usability-of-your-website/. [Accedido: 28-ago-2018].

[33] «TAW | Servicios de accesibilidad y movilidad web». [En línea]. Disponible en: https://www.tawdis.net/. [Accedido: 26-ago-2018].

[34] «WAVE Web Accessibility Tool». [En línea]. Disponible en: https://wave.webaim.org/. [Accedido: 26-ago-2018].

[35] «IDI Web Accessibility Checker : Web Accessibility Checker». [En línea]. Disponible en: https://achecker.ca/checker/index.php. [Accedido: 26-ago-2018].

[36] «Web Content Accessibility Guidelines (WCAG) 2.0». [En línea]. Disponible en: https://www.w3.org/TR/WCAG20/. [Accedido: 26-ago-2018].

[37] «The W3C Markup Validation Service». [En línea]. Disponible en: https://validator.w3.org/. [Accedido: 27-ago-2018].

[38] «Template Engines for Node.js». [En línea]. Disponible en: http://www.tutorialsteacher.com/nodejs/template-engines-for-nodejs. [Accedido: 26-ago-2018].

[39] «Overview | Maps JavaScript API», *Google Developers*. [En línea]. Disponible en: https://developers.google.com/maps/documentation/javascript/tutorial. [Accedido: 25-ago-2018].

[40] «Geocoding Service | Maps JavaScript API», *Google Developers*. [En línea]. Disponible en: https://developers.google.com/maps/documentation/javascript/geocoding. [Accedido: 25-ago-2018].

[41] «xlsx - npm». [En línea]. Disponible en: https://www.npmjs.com/package/xlsx. [Accedido: 28-ago-2018].

[42] T. Fisher, «What's an ICS File and How Do You Open One?», *Lifewire*. [En línea]. Disponible en: https://www.lifewire.com/ics-file-2622749. [Accedido: 28-ago-2018].

[43] «Importar eventos a Google Calendar - Ayuda de Calendario de Google». [En línea]. Disponible en: https://support.google.com/calendar/answer/37118?hl=es. [Accedido: 28-ago-2018].

[44] «ics», *npm*. [En línea]. Disponible en: https://www.npmjs.com/package/ics. [Accedido: 28-ago-2018].

# Anexo A: Motor de plantillas Jade

Un motor de plantillas es una herramienta que permite generar código HTML en aplicaciones web con el mínimo código necesario [38]. Su funcionamiento se basa en inyectar datos en la plantilla para generar la página HTML final. El motor de plantillas elegido es Jade, que, al no utilizar la sintaxis HTML, el código es más legible, intuitivo y fácil de escribir, además de ser uno de los más usados para *NodeJS*. Tal como indica en su página web, *Jade* es un lenguaje breve para escribir plantillas HTML, produciendo HTML, suportando código dinámico y reutilizable [22]. La Figura 70 muestra un ejemplo simple de este motor de plantillas.

```
!!! 5
html(lang="en")
  head
    title= pageTitle
    :javascript
      | if (foo) {
      |    bar()
      | }
  body
    h1 Jade - node template engine
    #container
      - if (youAreUsingJade)
         You are amazing
      - else
         Get on it!
         Get on it!
         Get on it!
         Get on it!
```

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Jade</title>
    <script type="text/javascript">
      //<![CDATA[
      if (foo) {
        bar()
      }
      //]]>
    </script>
  </head>
  <body>
    <h1>Jade - node template engine</h1>
    <div id="container">
      <p>You are amazing</p>
    </div>
  </body>
</html>
```

*Figura 70. Ejemplo motor de plantillas Jade (Fuente [22])*

En la parte izquierda del ejemplo, se observa el código *Jade* que, una vez procesador, genera el código HTML de la parte derecha. La sintaxis es muy sencilla, por cada elemento, en lugar de escribir la etiqueta correspondiente, sólo se escribe el nombre del elemento y, en caso de tener atributos, se escriben de igual forma entre paréntesis. Para otorgar un valor a un elemento existen dos formas: con texto plano predefinido en la plantilla (como el elemento *h1*) o insertando el valor de una variable (como el elemento *title*).

Además, como se puede observar en el ejemplo, otra de las ventajas de *Jade* consiste en la introducción de condicionales en la propia plantilla, para, según el valor de una variable, generar un código u otro. Esta funcionalidad es perfecta para el proyecto,

porque existen determinadas páginas (como la de perfil de usuario), que según el tipo de usuario que acceda a ella (atleta u administrador), se debe mostrar unas opciones u otras.

Otra ventaja, muy útil para el proyecto, consiste en la reutilización del código, definiendo bloques de contenido en archivos para importarlos posteriormente. De esta forma, el código que siempre sea fijo en la aplicación se almacenará en un fichero con su nombre y se importará cuando sea necesario, consiguiendo reutilizar y optimizar el código.

```
doctype html
html(lang=i18n.getLocale() == 'es' ? 'es' : 'en')
    head...
    body
        include header
        block content
        include footer
```

*Figura 71. Ejemplo código reutilizable en Jade*

La Figura 71 muestra el contenido del archivo *layout.jade* utilizado para el proyecto. En él se observa una página muy simple donde se incluye tanto la cabecera como el pie de página (definidos en archivos separados para no repetir código) y se define un bloque llamado *content* que se sustituirá por el contenido de cada una de las páginas. En la Figura 72 se puede observar que la página inicial *index.jade* extiende de *layout.jade* e indica cual es el contenido del bloque *content* que debe ser sustituido para formar la página final.

index.jade

```
extends auxPages/layout
block content
    .jumbotron.text-center
        h1 DiarySport
        p #{i18n.__("index.title")}
```

*Figura 72. Ejemplo bloque content en Jade*

Por último, para generar la página que se debe mostrar, es necesario utilizar la función *render* e indicar el fichero a renderizar y la lista de variables para que el motor de plantillas pueda acceder a ellas al generar el código. La Figura 73 muestra un ejemplo del uso de esta función para generar la página inicial.

```
/* GET Index page. */
router.get('/', function(req, res, next) {
    res.render('index',{
        i18n:res,
        session:req.session,
        feedback:req.flash("feedback")
    });
});
```

*Figura 73. Ejemplo funcion render en Jade y NodeJS*

# Anexo B: Internacionalización i18next

I18next es un *framework* de internacionalización escrito en y para JavaScript [21] que proporciona una solución completa para internacionalizar una aplicación web, con *plugins* para detectar el lenguaje del usuario o cargar las traducciones, siendo flexible, escalable, fácil de usar y multiplataforma, es decir, se puede utilizar para multitud de *frameworks* como *React, AngularJS, NodeJS, PHP* o *Android.*

Para utilizarlo, una vez instalado mediante el comando *npm install i18n --save*, es necesario indicar a la aplicación que se quiere usar. La Figura 74 muestra el código necesario para establecer la configuración, indicando qué idiomas puede tener la aplicación, el directorio con los archivos de traducciones y el idioma por defecto.

```
//i18next middleware and configuration
var session = require('express-session');
var i18n = require('i18n');
i18n.configure({
    locales:['es', 'en'],
    directory: __dirname + '/locales',
    defaultLocale: 'es',
    objectNotation: true,
    cookie: 'i18n'
});
app.use(cookieParser("i18next"));
app.use(session({
    secret: "my session",
    resave: true,
    saveUninitialized: true,
    cookie: { maxAge: 60000 }
}));
app.use(i18n.init);
```

*Figura 74. Ejemplo inicializar i18n en NodeJS*

Una vez configurada la librería, es necesario crear los archivos de traducciones. Cada archivo, cuyo nombre corresponde con el idioma, contiene un objeto *JSON* con pares clave-valor para cada traducción. La Figura 75 muestra una parte de los archivos de traducciones del proyecto.

es.json

```
{
    "header": {
        "races": "Carreras",
        "login": "Iniciar sesión",
        "forget-pwd": "¿Has olvidado la contraseña?",
        "lbl-email": "Te enviaremos un correo electrónico",
        "btn-recover": "Recuperar contraseña",
        "lbl-hide": "Ocultar ",
        "new-here": "¿Nuevo aquí?",
```

en.json

```
{
    "header": {
        "races": "Races",
        "login": "Log in",
        "forget-pwd": "Have you forgotten your password?",
        "lbl-email": "We will send you an email",
        "btn-recover": "Recover password",
        "lbl-hide": "Hide ",
        "new-here": "New here?",
```

*Figura 75. Ejemplo ficheros de traducciones i18n*

Para permitir al usuario cambiar de idioma cuando lo necesite, habilita una opción en la página para que envíe una petición al servidor (véase parte izquierda Figura 76) a la dirección **/idioma**. El servidor (véase parte derecha Figura 76), al recibir la petición, establece en las *cookies* y en la variable *i18n* el idioma seleccionado por el usuario.

```
ul.list-inline.text-uppercase
    li(class=i18n.getLocale() == 'es' ? 'active' : '')
        a(href='/es') Español
    li(class=i18n.getLocale() == 'en' ? 'active' : '')
        a(href='/en') English
```

```
//Cambiar idioma a castellano
router.get('/es', function (req, res) {
    res.cookie('i18n', 'es');
    i18n.setLocale('es');
    res.redirect('back');
});
//Cambiar idioma a inglés
router.get('/en', function (req, res) {
    res.cookie('i18n', 'en');
    i18n.setLocale('en');
    res.redirect('back');
});
```

*Figura 76. Ejemplo cambio de lenguaje en NodeJS y HTML*

La mayoría de las traducciones se utilizan en los ficheros de plantillas *Jade* para crear la página HTML según el idioma seleccionado por el usuario. Para poder acceder al fichero de traducciones y recuperar el valor se emplea el siguiente código: *#{i18n.__(clave)}*. Un ejemplo de ello se puede observar en la Figura 72. Aunque también es posible utilizar las traducciones en el lado del servidor, para, por ejemplo, enviar un mensaje al usuario al recibir una petición AJAX. Para ello, el código es ligeramente distinto al anterior, siendo *res.__(clave)* la sentencia a utilizar. La Figura 77 muestra el código desarrollado para la función de iniciar sesión: si existe algún error en la autenticación, se devuelve un mensaje JSON con el código 401 y una cadena con información del error, que dependerá del lenguaje que tenga seleccionado el usuario.

```
User.authenticate(req.body.emailUsername, req.body.password, function (error, user) {
    if (error || !user) {
        return res.status(401).json({
            result:false,
            msg: res.__("users_js.errorLogIn")
        });
```

*Figura 77. Ejemplo traducción en el lado del servidor con i18n*

# Anexo C: API de Google Maps y Geocode

En este anexo se justifica cómo y porqué se utilizan las API de Google Maps y Geocode en el proyecto, que permiten incluir mapas, en este caso de Google Maps, y obtener las coordenadas cartesianas dada una dirección postal.

## Google Maps

La API de Google Maps para JavaScript permite a un usuario personalizar mapas con su propio contenido e imágenes para mostrarlos en su página web o en dispositivos móviles [39]. Para ello, es necesario crear una cuenta en *Google Cloud Platform Console*, registrar un proyecto o seleccionar uno existente, activar la API de Google Maps para JavaScript y obtener una clave para poderlo utilizar.

Una vez se disponga de la clave, se debe crear una nueva página (o incluir en una existente) con un elemento *div* cuyo *ID* sea *#map*, crear una función llamada *initMap()* que inicialice el mapa y, por último, importar el *script* de la API (véase Figura 78). Al inicializar el mapa, se permiten especificar diversos valores, como el centro del mapa (especificado por latitud y longitud) y el *zoom*. Todo este proceso se lleva a cabo con computación en el lado del cliente.

```
<div id="map"></div>
<script>
  var map;
  function initMap() {
    map = new google.maps.Map(document.getElementById('map'), {
      center: {lat: -34.397, lng: 150.644},
      zoom: 8
    });
  }
</script>
<script src="https://maps.googleapis.com/maps/api/js?key=YOUR_API_KEY&callback=initMap"
async defer></script>
```

*Figura 78. Código para incluir Google Maps JavaScript API. Fuente [39]*

Para añadir los marcadores en el mapa, es necesario crearlos mediante el constructor *Marker*, indicando su posición, el título y el mapa al que se quiera añadir. Además, para que al pinchar sobre uno se despliegue su información, se le asocia un *listener* sobre el evento *click* para que muestre el nombre de la carrera (con un enlace a su página), el tipo de deporte y la fecha.

```
function addMarker(race) {
    var marker = new google.maps.Marker({
        map: map,
        position: new google.maps.LatLng(race.coordinates.latitude, race.coordinates.longitude),
        title: race.name
    });
    var infoWindow = new google.maps.InfoWindow({
        content: '<div class="info_content">' +
        '<h4><a href="/races/'+race.name+'/info" target="_blank">'+race.name+'</a></h4>' +
        '<p>'+race.sport+'</p>' +
        '<p>' + race.dateTimeFormat + '</p>' +
        '</div>'
    });
    google.maps.event.addListener(marker, 'click', function () {
        if (lastMarker) lastMarker.close();
        infoWindow.open(map, marker);
        lastMarker=infoWindow;
    });
}
```

*Figura 79. Código para añadir marcador al mapa (Google Maps JavaScript API)*

Por último, para que sólo se encuentre una ventana de información de carrera abierta, se guarda la última ventana abierta y, al pinchar para abrir otra, se cierra esta última. El resultado sería el siguiente:

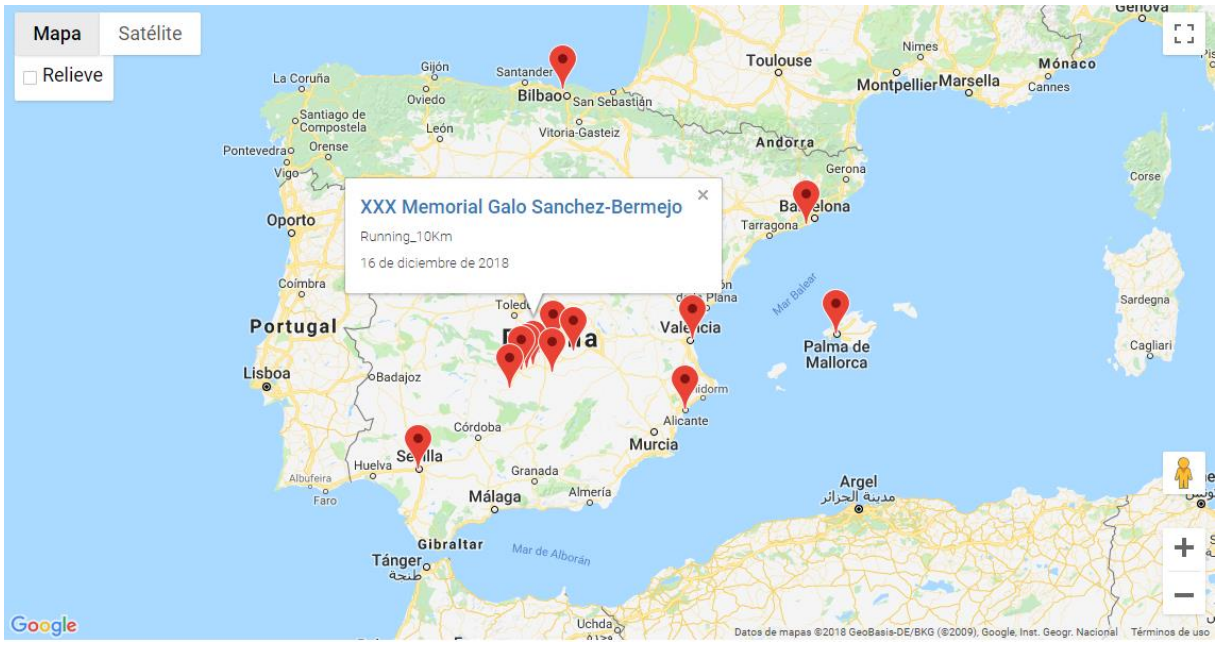


*Figura 80. Mapa resultante con marcadores (Google Maps JavaScript API)*

## Geocode

*Geocoding* es el proceso de convertir direcciones en coordenadas geográficas, que pueden ser usadas para añadir marcadores o posiciones en los mapas [40]. Para utilizar este servicio, es necesario activar su API en *Google Cloud Platform Console* y, al haberla obtenido anteriormente, no es necesario obtener una nueva clave, se puede usar la misma.

Aunque este servicio cuenta con una interfaz HTTP para realizar peticiones con las conversiones, existen también librerías para *NodeJs* como *node-geodecoder*. Esta librería simplifica mucho el proceso, iniciando el constructor con la clave (véase Figura 81), podemos utilizar la función *geocode* para obtener las coordenadas geográficas.

```
getGeocoder:function(){
    var options = {
        provider: 'google',
        httpAdapter: 'https', // Default
        apiKey: 'AIzaSyCAZyysF63-DJC_iaZZHjdPo4lut0H3tR0',
        formatter: null
    };
    var geocoder = nodeGeocoder(options);
    return geocoder;
}
```

*Figura 81. Código para crear el constructor geocoder*

En la Figura 82 se muestra el código utilizado para ello. Para simplificar el proceso, se ha creado una regla en el esquema *Race* de la base de datos, para que cada vez que guarde algo en ella (tanto si es al crear como si es por modificar), convierta la dirección especificada en coordenadas geográficas y las guarde en el campo *coordinates*.

```
RaceSchema.pre('save', function (next) {
    var race = this;
    utils.getGeocoder().geocode(race.place+" "+race.province,function(err,res){
        if (err) {
            return next(err);
        }
        race.coordinates={'latitude':res[0].latitude,'longitude':res[0].longitude};
        next();

    });
});
```

*Figura 82. Código para obtener coordenadas a partir de una dirección (Geocode API)*

# Anexo D: Importación masiva de datos

Con el objetivo de facilitar la tarea a los usuarios organizadores que quieran añadir varias carreras a la aplicación, en lugar de añadirlas una a una a través del formulario de crear carrera, se le permite descargar una plantilla Excel para introducir todos los datos de sus carreras y, posteriormente, importarlos en la aplicación.

Para ello, se han preparado dos archivos Excel, uno en inglés y otro en castellano, para que cualquier usuario pueda hacer uso de esta opción en el idioma que desee.

La Figura 84 muestra la plantilla Excel donde el usuario organizador debe introducir todos los datos de sus carreras. Además de contar con una fila dónde se informa qué datos va a contener cada columna (y el formato de alguno de ellos), en la segunda fila se incluye un ejemplo para ayudar al usuario. Por último, los campos de deporte o provincia, al igual que en el formulario de crear carrera, despliegan una lista para que el usuario seleccione la provincia y el deporte de la carrera.

Para procesar el archivo en el servidor y poder importar las carreras se hace uso del módulo *xlsx* [41] que permite extraer los datos de un fichero Excel. La función *sheet_to_json* de este módulo convierte los datos de la hoja que se le indique por parámetros en un objeto *JSON*. En primer lugar, se comprueba que la longitud sea mayor que uno (véase Figura 83), para verificar que se han añadido carreras y, en caso afirmativo, se elimina la primera fila (que corresponde con la carrera de ejemplo). Posteriormente se comprueba, al igual que al crear una carrera, que todos los datos sean válidos antes de añadirlos a la base de datos. Si hubiera algún error se informa al usuario de ello para que pueda corregirlo.

```
var workbook = XLSX.readFile(path);
var json = XLSX.utils.sheet_to_json(workbook.Sheets[workbook.SheetNames[0]]);
if(json.length>1){
    json.shift();
    var err=excelHasErrors(json,req.session.user.id);
    if(!err){
        Race.create(json, function (err) {
```

*Figura 83. Código para leer el fichero Excel e insertar las carreras en la base de datos*

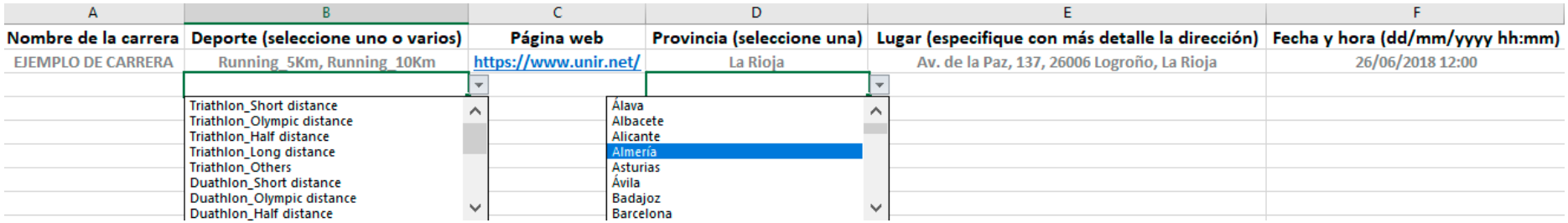


*Figura 84. Plantilla Excel para importar carreras de forma masiva*

# Anexo E: Exportación del calendario

En la actualidad existen multitud de aplicaciones calendario que permiten a un usuario organizarse día a día para ser más productivos y no olvidar citas o eventos que tienen planteado asistir, como *Google Calendar* o *Calendario de Outlook*.

Estas aplicaciones permiten crear eventos de forma manual a través de su aplicación o importarlos desde otras aplicaciones. Para poder importarlos, es necesario seguir un tipo de formato y estructura específica, como es el caso de los ficheros *iCalendar*. Un fichero *iCalendar* es un fichero de texto plano que contiene información de determinados eventos y posee una extensión *.ics,* aunque pueden existir también con extensión *ical* o *icalendar* [42]. Para construir estos ficheros es necesario seguir la estructura propuesta, porque si no, al importarlo en otras aplicaciones, surgirán errores. La Figura 85 muestra la estructura básica para crear un fichero de este tipo: en él se incluye una cabecera con cierta información y una lista de eventos con todos sus detalles.

```
BEGIN:VCALENDAR
VERSION:2.0
PRODID:< [escribe aquí los datos de la ID] >
(aquí vienen los demás datos de la cabecera)
BEGIN:VEVENT
(detalles del evento)
END:VEVENT
BEGIN:VEVENT
(detalles del evento)
END:VEVENT
END:VCALENDAR
```

*Figura 85. Estructura básica de un fichero ics.[43]*

Aunque estos archivos se pueden crear añadiendo cada evento dentro de la estructura, existen diferentes módulos o utilidades que ayudan a esta tarea, para que sea menos tediosa. En este proyecto se ha utilizado el módulo *ics* [44].

La Figura 86 muestra el código necesario para generar este archivo. En un vector se van añadiendo todos los eventos en formato JSON, indicando pares clave-valor. Para este proyecto los datos utilizados son el nombre de la carrera, la ubicación, fecha y, como descripción del evento, el tipo de deporte y la dirección web de la carrera. Una vez se han

añadido todas las carreras al vector, se debe llamar a la función *createEvents* del módulo *ics*, que devuelve un objeto cuyo atributo *value* es la cadena de texto que debe contener el fichero *.ics.* Por último, se genera un archivo de texto con esa cadena y se le asigna la extensión correcta para que el usuario pueda descargarlo e importarlo en su aplicación de calendario preferida.

```
exportCalendar:function(races,res){
    var events=[];
    for (var i=0;i<races.length;i++){
        var event={
            title:races[i].name,
            location:races[i].place+" ("+races[i].province+")",
            start:moment(moment(races[i].dateTime)).format('YYYY-M-D').split("-"),
            end:moment(moment(races[i].dateTime)).format('YYYY-M-D').split("-"),
            description:res.__("shared.sport")+": "+races[i].sport+"\n\n"+res.__("newRace.url")+": "+races[i].url
        };
        events.push(event);
    }
    var calendar=ics.createEvents(events);
    return calendar.value;
},
```

*Figura 86. Función desarrollada para crear el archivo con ayuda del módulo ics*

Una vez importado el calendario, en *Google Calendar* por ejemplo, se añade un evento por cada carrera a la que el usuario esté apuntado. La Figura 87 muestra un ejemplo de ello, en este caso el usuario se ha apuntado a la carrera de 10 kilómetros de Socuellamos, el día 28 de octubre de 2018.

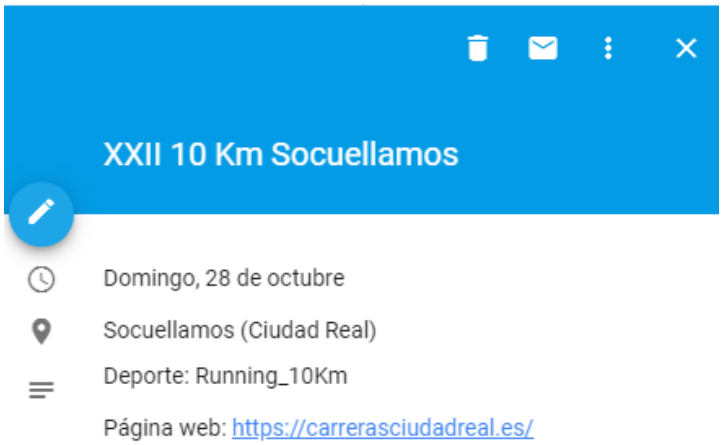


*Figura 87. Evento creado en Google Calendar al importar el calendario de carreras*