Universidad del País Vasco
Euskal Herriko Unibertsitatea

Facultad de Informática

Grado de Ingeniería Informática

▪ Proyecto Fin de Grado ▪

Ingeniería del Software

Juzgando las olas ++

Solución web sobre MEAN para el cálculo de *ranking* y *seeding* en campeonatos de Surf.

Andrés Zamora Soliz

Junio 2016

Director: José Miguel Blanco

informatika fakultatea
facultad de informática

Índice

AGRADECIMIENTOS

Me gustaría agradecer a las personas que me ayudaron al desarrollo de este proyecto o que de una u otra forma me han permitido llegar aquí.

Agradezco a mi director de proyecto José Miguel Blanco por orientarme en el desarrollo del proyecto y por motivarme en cada reunión para que siga trabajando más duro.

Agradezco a mi director de proyecto en empresa Adrián Izquierdo, quien me presentó este proyecto. Le estoy muy agradecido por darme la oportunidad de desarrollarlo y poder aprender más. Trabajar con él ha sido un placer.

Agradezco a mis padres, sin ellos no estaría escribiendo esta memoria. Toda mi educación se la debo a ellos, gracias por creer en mí siempre y apoyarme en las buenas y en las malas.

Agradezco a mi hermano, sin él la universidad no hubiera sido lo mismo y los trabajos en grupo tampoco.

Agradezco a Judith Sánchez, quien ha estado conmigo en todo momento, y por su ayuda en la revisión de la ortografía de esta memoria.

Agradezco a todos los profesores de la carrera que me han enseñado a lo largo de estos años, todos ellos me han dado los conocimientos necesarios para llegar a este momento. Les estaré agradecido siempre.

1

Introducción

El proyecto desarrollado trata de solucionar un problema real que existe en la actualidad en los campeonatos de surf. El problema es el cálculo de *ranking* y *seeding* en los Circuitos. Este problema se debe a que los cálculos realizados para obtener el *ranking* y *seeding* suelen ser demasiados y en la actualidad se realizan de forma manual. Mi proyecto buscará solucionar este problema creando las funcionalidades necesarias para realizar los cálculos de forma automática. De esta forma, lo que actualmente lleva tres o cuatro días en calcularse de forma manual, se podría tener resuelto en cuestión de segundos o minutos. Esto ahorraría mucho tiempo a las organizaciones de surf, las cuales tendrían un especial interés en estas funcionalidades para usarlas en sus campeonatos.

Este proyecto se desarrolla en una empresa llamada Wavescores Development S.L., creada por un exalumno de la universidad junto con 2 socios. Mi proyecto se construye sobre la aplicación web creada durante el proyecto fin de carrera de este exalumno, aplicación web que está actualmente en funcionamiento en un entorno real como es el de los campeonatos de surf.

En esta memoria trataré diferentes aspectos del proyecto, los cuales expondré en 11 capítulos más:

- **Antecedentes:** En este capítulo hablaré sobre el proyecto inicial, la empresa, las tecnologías implicadas en el proyecto inicial y los conocimientos del mundo del surf.
- **Objetivos:** En este capítulo expondré el alcance del proyecto y los objetivos concretos del mismo, y las tareas a realizar.
- **Adaptación:** En este capítulo expondré la adaptación a las tecnologías y al software durante el proyecto.
- **Análisis:** En este capítulo expondré el modelo del dominio, el diagrama de casos de uso y el flujo de eventos del proyecto.
- **Diseño:** En este capítulo expondré los diagramas de secuencia de cada caso de uso.
- **Implementación:** En este capítulo expondré la estructura del proyecto, los módulos y ficheros que se usan para el desarrollo del mismo.
- **Herramientas y Entorno de Desarrollo:** En este capítulo expondré las herramientas usadas y el entorno de desarrollo elegido para desarrollar la aplicación.
- **Pruebas:** En este capítulo expondré las pruebas realizadas para testear lo desarrollado y comprobar su correcto funcionamiento.
- **Gestión:** En este capítulo expondré la planificación y el seguimiento y control del proyecto.
- **Conclusiones:** En este capítulo expondré las conclusiones que se ha tenido después del desarrollo del proyecto.
- **Bibliografía:** En este capítulo expondré distintas referencias bibliográficas que se han usado para el desarrollo del proyecto.

2

Antecedentes

En este capítulo se explicarán los antecedentes del proyecto. En él se hablará sobre el proyecto inicial que originó el desarrollo del mío, también sobre la empresa donde he realizado el proyecto, llamada Wavescores Development S.L, las tecnologías implicadas de partida en el proyecto y unas nociones básicas sobre el mundo de los campeonatos de surf, necesarias para enmarcar los objetivos y el desarrollo del proyecto.

2.1. Proyecto Inicial

Mi proyecto está cimentado sobre el PFC[1] (Proyecto Fin de Carrera) realizado por un exalumno de la universidad llamado Adrián Izquierdo. Él ha sido mi DPE[1]. Su proyecto era una aplicación web para gestionar campeonatos de surf, crear campeonatos, categorías, mangas, registrar surfistas en los campeonatos, etc. El proyecto lo inició como PFC en la universidad, pero

[1] Director de Proyecto de Empresa, este término se usará a lo largo de la memoria.

a su vez lo empezó a utilizar en un entorno real, concretamente en campeonatos de surf en Guipúzcoa. A la EHSF[2] le interesó la aplicación web y contrató sus servicios.

De esta forma, su proyecto ha crecido desde que lo inició hace dos años. Ha añadido nuevas funcionalidades que mejoran el servicio prestado permitiendo facilitar la gestión tanto a los jueces en el transcurso de los campeonatos como a la parte administrativa antes y después de los campeonatos de surf.

2.2. Wavescores Development S.L.

Imagen 1: Logo de la empresa Wavescores Development S.L.

El proyecto inicial que empezó como PFC, con el tiempo, y viendo el potencial de crecimiento de la misma, llevó a la creación de la empresa Wavescores Development S.L. Fue creada el año pasado por el DPE junto con dos socios más. Estos socios se llaman Gorka y Mikel, son jueces en campeonatos de surf. Ellos se encargan de promocionar la aplicación y, gracias a su trabajo cercano con el mundo del surf, pueden ver las posibles mejoras que podrían implementarse de cara a mejorar la experiencia para los jueces y la parte administrativa en los campeonatos.

En 2016 la empresa ha obtenido beneficios, pero los socios decidieron usar ese dinero para invertirlo en la misma empresa y, de esta forma, poder mejorar ciertos equipos y poder paliar cualquier gasto imprevisto que hubiera.

La empresa ha expandido el ámbito de la aplicación web, ya no sólo gestiona campeonatos de surf, sino que también campeonatos de *bodyboard* y de *surf paddel*. De esta forma, la presencia de la aplicación y de la empresa ha crecido en Guipúzcoa y poco a poco lo va haciendo en España. Este año han empezado a tratar con la Federación de Surf de Cantabria para utilizar la aplicación en sus campeonatos.

[2] Euskal Herriko Surf Federazioa. http://www.euskalsurf.com

2.3. Tecnologías implicadas

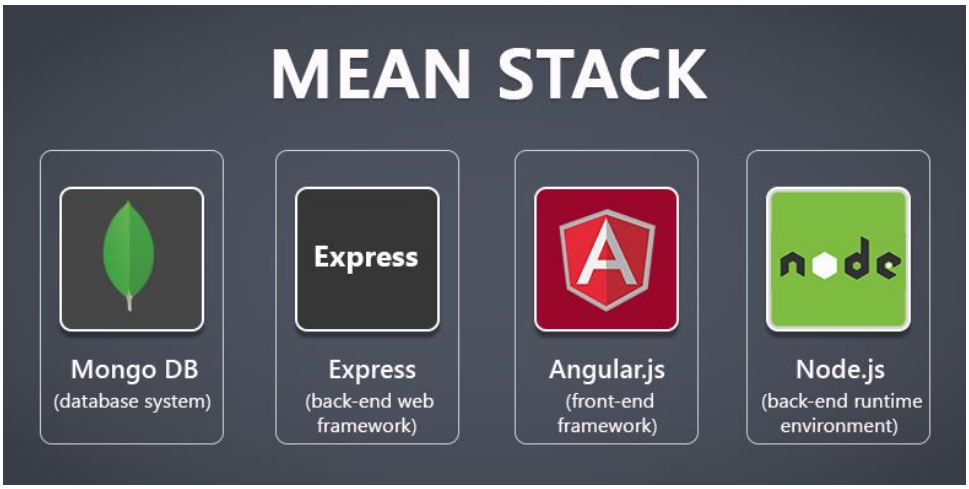


Imagen 2: *Stack* de MEAN con las tecnologías que la conforman.

Las tecnologías utilizadas para el proyecto inicial fueron las que forman el *stack* MEAN[3], que son: MongoDB, Express.js, Angular.js y Node.js. Mis conocimientos sobre dichas tecnologías eran pocos, solamente había tratado con Node.js debido a que fue una de las tecnologías principales utilizadas en unas prácticas de verano que realicé en MagnaSIS, empresa junior de la universidad. Dichas prácticas me permitieron aprender a usar Node.js y aprender a desarrollar el lado servidor en Javascript. Durante la carrera siempre hemos usado PHP para el lado servidor, aunque nos habían dicho que en la actualidad Javascript se usaba tanto para lado cliente como para lado servidor.

Lo realizado durante la práctica me dio confianza para aprender más tecnologías nuevas y embarcarme en el proyecto ofrecido por la empresa. Como informáticos siempre nos tocará aprender durante nuestra carrera profesional y cuanto antes empecemos mejor.

2.4. Conocimientos del mundo del surf

Mis conocimientos previos del mundo del surf al inicio del proyecto eran completamente nulos. Dicho deporte no me llamaba mucho la atención, por lo que no conocía el funcionamiento de los campeonatos ni las reglas de los mismos. Pero al ser un proyecto sobre ese ámbito tenía

[3] Página Oficial: http://mean.io

que aprender su funcionamiento para poder entender la aplicación web que había inicialmente, la cual sería la base sobre la que desarrollaría mi proyecto.

Me explicaron que los campeonatos de surf están formados por varias categorías, que pueden ser por ejemplo: Sub15, Sub18, etc. Un campeonato se puede desarrollar durante uno o dos días, esto depende de la cantidad de surfistas apuntados y de la cantidad de categorías en dicho campeonato. En una categoría se apuntan varios surfistas y está formada por rondas. Las rondas serían como en el mundo del fútbol (usaré similitudes con este deporte al ser el más conocido y fácil de entender) las fases de clasificación en un torneo, por ejemplo: fase de grupos, octavos de final, cuartos de final, semifinales y, por último, la final. Cada ronda está a su vez formada por mangas, las mangas serían como en el mundo del fútbol los partidos. En éstas un grupo de surfistas se disputan el poder pasar a la siguiente ronda de la categoría de un campeonato.

Al inicio de una categoría, los surfistas en la primera ronda se distribuyen en distintas mangas basados en el *seeding* que poseen (el *seeding* es la suma de los puntos que un surfista ha obtenido en las distintas categorías de algunos campeonatos y la suma de los puntos obtenidos en el *ranking* total del año pasado). Antes de iniciar mi proyecto, el *seeding* se ingresaba de forma manual, el poder calcular dichas sumas y añadir el *seeding* correspondiente a cada surfista es una de mis labores en este proyecto. Cada manga tiene una duración de 20 minutos para que a los surfistas les dé tiempo a mostrar sus habilidades tomando olas y que los jueces les puntúen. El tiempo de una manga en la final aumenta a 25 minutos. Una manga está conformada por varios surfistas, por lo general son 4, pero ese número puede variar dependiendo del formato que se ha decidido usar en la categoría. Al finalizar una ronda, los surfistas que pasan a la siguiente serán los dos mejores de cada manga, ese número de surfistas también puede variar debido al formato elegido. A medida que se vaya pasando de ronda en ronda el número de mangas se reducirá, como también lo hará el número de surfistas hasta llegar a la final, que es una única ronda con los surfistas que han pasado de la ronda anterior. Al terminar esta manga, la categoría se dará por finalizada y a los surfistas se les asignarán sus posiciones finales.

3

Objetivos

En este capítulo trataré los objetivos que he tenido a la hora de realizar el proyecto.

3.1. Alcance

El proyecto queda definido de la siguiente manera:

Desarrollar funcionalidades para una aplicación web que gestiona campeonatos de surf.

Estas funcionalidades son: crear Circuitos, calcular *ranking* en Circuitos y calcular *seeding* en Circuitos.

3.1.1. Descripción de los objetivos concretos del proyecto

Los objetivos serán realizar tres funcionalidades, también la integración en el software preexistente y la realización de pruebas exhaustivas.

- Funcionalidad 1 - Crear Circuitos:

  Esta funcionalidad abarcará la definición del Modelo tour (Circuito), la implementación del código para crear Circuitos y, además, la posibilidad de añadir Categories (es un conjunto de categorías que formarán parte del Circuito). Un circuito puede tener varios Categories.

  Modelo tour.js (circuito).

  Está formado por los siguientes campos:

  - Nombre.
  - Año.
  - Campeonatos:[Contest_ids]
  - Categories: [nombre, categorías_ids, *ranking*AñoPasado, tabla*Ranking*]
  - scores: [ ]

  Campos del formulario para Crear Circuitos:

  - Nombre.
  - Año.
  - Campeonatos.
  - Categories name.
  - Categorías.
  - Puntuaciones.

  Campos del formulario para añadir Categories:

  - Categories name.
  - Categorías.

- Funcionalidad 2 - Cálculo de *Ranking* en Circuitos:

  Esta funcionalidad se encargará de calcular el *ranking*, que es la suma de puntuaciones de un surfista en las distintas Categorías que forman parte de la Categories de un Circuito. Se creará una tabla para mostrar los surfistas y los cálculos realizados. Cada columna representará a un grupo de surfistas de una Categoría de la Categories de un Circuito. Se creará una columna por cada Categoría. El orden de los surfistas en las columnas se determinará por la posición final obtenida al finalizar el Campeonato de dicha Categoría. La última columna de la tabla representará el *Ranking* Total de cada surfista con la suma de los puntos obtenidos en cada Categoría.

- Funcionalidad 3 - Cálculo de *Seeding* en Circuitos:

  Esta funcionalidad se encargará de calcular el *seeding*, que es el resultado de la suma de puntuaciones de un surfista en las distintas Categorías que forman parte de la Categories de un Circuito más la suma de la puntuación obtenida del *ranking* total del año pasado.

  Se creará una tabla para mostrar los surfistas y los cálculos realizados. La primera columna representará el *ranking* total del año pasado, con los surfistas participantes, sus posiciones y puntos. Luego, se creará una columna por cada Categoría con sus surfistas y, finalmente, la última columna de la tabla representará el *Seeding* Total de cada surfista con la suma de los puntos obtenidos en cada Categoría.

- Integración en el software preexistente:

  Se deberá realizar previamente una integración en el software preexistente. Deberá adaptarse a las tecnologías y al software para poder entender el funcionamiento de ese software y poder integrar las nuevas funcionalidades que se van a realizar.

- Pruebas exhaustivas:

  Se realizarán muchas pruebas sistemáticas y completas. Serán de diversos tipos para probar la correcta ejecución de las funcionalidades desarrolladas en situaciones diferentes.

3.1.2. Entregables

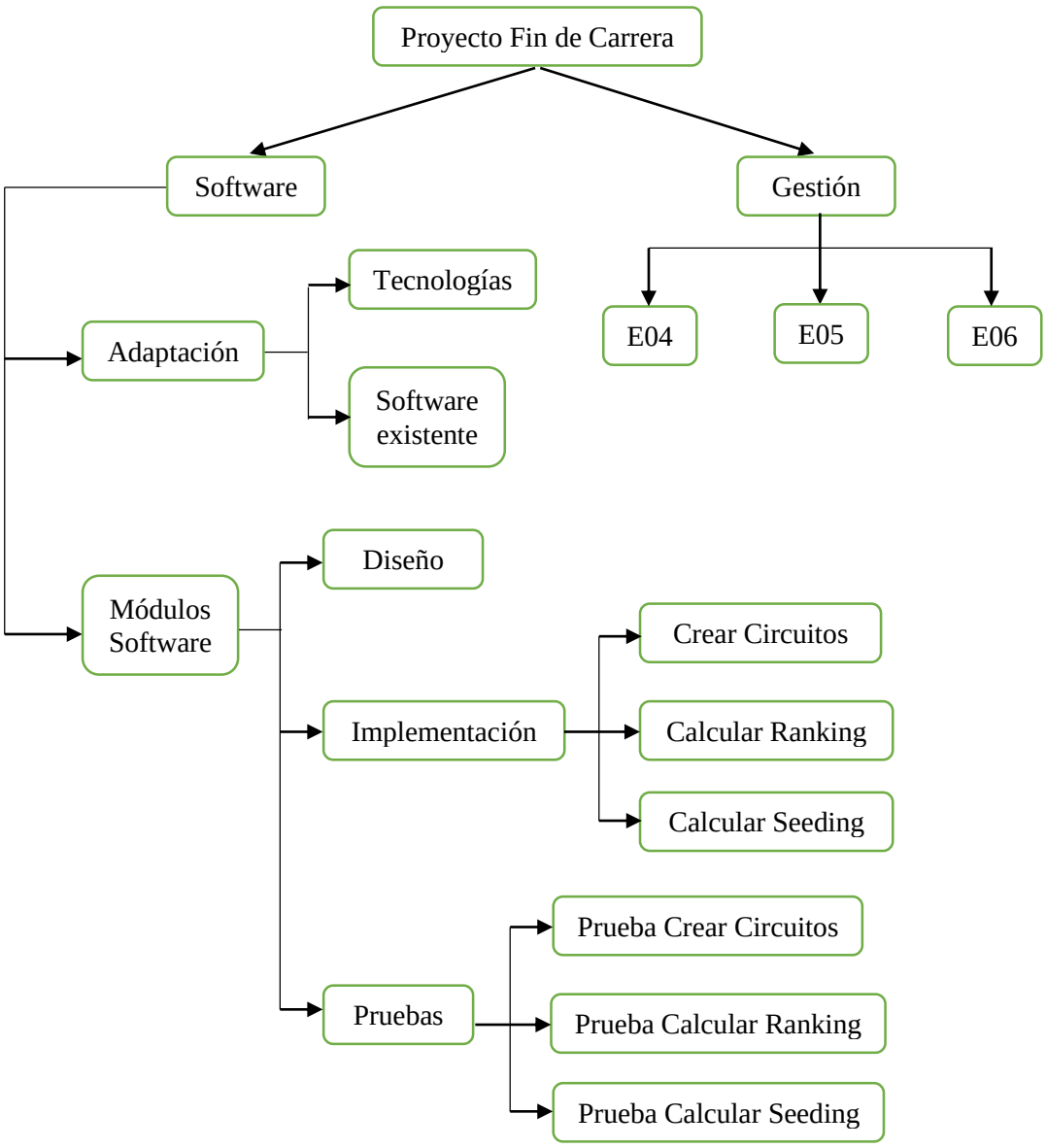
3.1.2.1 Relacionados con el objeto de proyecto en sí.

- E01. Funcionalidad Crear Circuitos.
- E02. Funcionalidad Calcular *Ranking* de un Circuito.
- E03. Funcionalidad Calcular *Seeding* de un Circuito.

3.1.2.2 Relacionados con la Gestión del proyecto.

- E04. Planificación.
- E05. Seguimiento y Control.
- E06. Memoria.

3.1.3. WBS/EDT



3.1.4. Descripción tareas a realizar:

3.1.4.1 Funcionalidad Crear Circuitos (E01).

- **T01.1:** Definir el modelo tour.js (Circuito).
- **T01.2:** Crear un elemento modal para mostrar el formulario para Crear Circuitos.
- **T01.3:** Guardar el Circuito en la base de datos.
- **T01.4:** Crear un elemento modal para mostrar el formulario para Añadir Categories de un Circuito.
- **T01.5:** Modificar el Circuito con el Categories añadido en la base de datos.

3.1.4.2 Funcionalidad Calcular *Ranking* de un Circuito (E02).

- **T02.1:** Crear una tabla con las Categorías de la Categories de un Circuito y con sus surfistas correspondientes.
- **T02.2:** Calcular el *Ranking*.
- **T02.3:** Guardar la tabla *Ranking* en la base de datos.

3.1.4.3 Funcionalidad Calcular *Seeding* de un Circuito (E03).

- **T03.1:** Elegir el modo en que se quiere añadir el *Ranking* Total del año pasado.
- **T03.2:** Crear una tabla con las Categorías de Categories de un Circuito y con sus surfistas correspondientes.
- **T03.3:** Guardar el *Ranking* Total del año pasado en la base de datos.
- **T03.4:** Calcular el *Seeding*.
- **T03.5:** Modificar el *Seeding* de una Categoría seleccionada en la base de datos.

4

Adaptación

En este capítulo trataré la adaptación al proyecto, la cual fue de menos a más. Se podría dividir en dos partes: adaptación a las tecnologías y adaptación al software.

Previamente al inicio de las funcionalidades del proyecto, se realizaron dos funcionalidades que ayudaron en el aprendizaje y a la adaptación al proyecto. La primera funcionalidad se podría poner en el apartado de adaptación a las tecnologías y la segunda en adaptación al software existente.

4.1. Adaptación Tecnologías

4.1.1. Descripción de las Tecnologías.

Como ya he mencionado anteriormente, las tecnologías empleadas en el desarrollo de esta práctica forman parte del *stack* MEAN. MEAN es un conjunto de tecnologías para el

desarrollo de páginas web dinámicas con el uso del lenguaje de programación JavaScript. Está conformado por:

- MongoDB
- Express.js
- Angular.js
- Node.js

MongoBD

Es un sistema de base de datos NoSQL multiplataforma orientado a documentos. Esto quiere decir que en lugar de guardar los datos en registro, los guarda en documentos. Dichos documentos son almacenados en BSON que es una representación binaria de JSON.

Los documentos se pueden agrupar en colecciones, que serían como las tablas en base de datos relacionales. La diferencia es que las colecciones pueden almacenar documentos con formatos muy distintos, no están sometidos a un esquema fijo.

Es una base de datos ágil, permite cambiar rápidamente los esquemas cuando las aplicaciones crecen y evolucionan. Proporcionan funcionalidades que los desarrolladores esperan de las bases de datos tradicionales.

Express.js

Es un *framework* de desarrollo de aplicaciones web minimalista y flexible para Node.js. Este *framework* proporciona los métodos suficientes para manejar las solicitudes o peticiones que se hacen por medio de los métodos del protocolo HTTP (GET, POST, etc). Ofrece un sistema de enrutamiento, dentro del *stack* MEAN es aprovechado por el lado servidor.

Angular.js

Es un *framework* de Javascript para el desarrollo web en el lado cliente. Permite crear y mantener aplicaciones web SPA (Single-Page Applications). Fue creado con el objetivo de permitir un desarrollo ordenado, sencillo y sobre todo fácil de mantener en un futuro. Para lograr ese objetivo, Angular usa el patrón de diseño MVC (Modelo, Vista, Controlador), de esta manera podemos separar el código en diferentes secciones y que sea más sencillo el desarrollo web.

Node.js

Es un entorno Javascript del lado servidor de código abierto. Tiene una arquitectura orientada a eventos. Node ejecuta Javascript utilizando el motor V8 que ha desarrollado Google

para usar en su navegador Chrome. El motor V8 proporciona a Node un entorno de ejecución del lado servidor que compila y ejecuta Javascript.

4.1.2. Primera Funcionalidad

Esta primera funcionalidad fue sugerida por el DPE. Me pidió que creara una funcionalidad en la plantilla que MEAN da por defecto para empezar a practicar con dichas tecnologías (la plantilla es una aplicación de gestión de artículos, permite crearlos, modificarlos, eliminarlos, etc.). La funcionalidad a crear por mí era poder poner un artículo como favorito.

Para poder desarrollar esta funcionalidad tuve que modificar el modelo MongoDB de Artículo y añadirle un campo favorito de tipo "booleano". Si el campo era "false" el artículo no era favorito, en caso contrario, sería favorito. Al ser una plantilla ya funcional desde el inicio, el servidor Node.js ya estaba definido y no se tuvo que modificar. A pesar de ello, los conocimientos de Node.js adquiridos en las prácticas realizadas en MagnaSIS me permitieron entender el código sin necesidad de indagar mucho en ello. El aprendizaje de Express.js se realizó al entender cómo usar el enrutamiento para poder realizar las solicitudes y peticiones entre el lado cliente y el lado servidor. El uso de Angular.js fue el que tuvo más impacto, pude comprobar cómo se utilizaba el patrón de diseño MVC en esa aplicación. La primera vez resultó un poco complicado entender dónde se encontraban los ficheros que se querían consultar. A pesar de ello, lo ordenado del código permitió distinguir dónde estaban los modelos, las vistas y los controladores, y mostraba el lado cliente y el lado servidor con claridad. Aprendí cómo usar directivas (componentes útiles que agregan valor al HTML añadiendo comportamientos diversos). También sobre módulos y controladores. Consulté frecuentemente esta página[6] para poder entender más sobre Angular.js.

En general, el desarrollar esta primera funcionalidad me permitió aprender mucho sobre las tecnologías de MEAN y poder adaptarme a ellas.

4.2. Adaptación al Software

Me tuve que adaptar al software ya implementado previamente. No era algo muy común durante la carrera el desarrollar sobre aplicaciones ya iniciadas, sino el crear desde cero las aplicaciones. A pesar de ello, se sabe que es algo común para los informáticos que se presente esta situación. Por ello, lo tomé como un reto y una oportunidad de aprendizaje.

La adaptación al software la realicé a lo largo del proyecto, pero haré hincapié en el aprendizaje inicial antes del inicio del proyecto mismo. Esto se hizo en la segunda funcionalidad implementada.

4.2.1. Segunda Funcionalidad

Esta segunda funcionalidad me fue encomendada por el DPE. Me pidió que arreglara un pequeño fallo que tenía en su aplicación al calcular la posición final y el *ranking* de los surfistas al finalizar la Categoría de un Campeonato (diferente al cálculo de *ranking* de Circuitos que debía implementar para el proyecto).

El desarrollar esta funcionalidad me permitió conocer el software del DPE, el funcionamiento de la aplicación y cómo utilizaba las tecnologías. Pude aprender los modelos MongoDB existentes, entender para qué servían y cuáles podrían ser útiles para el desarrollo de las funcionalidades que implementaría en el futuro. Aprendí el uso de una librería de Node.js llamada Mongose. Mongose es un ODM (Object Data Mapper) en Javascript, esto quiere decir que traduce los datos de la base de datos en objetos Javascript para utilizarlos en la aplicación. Durante el desarrollo de esta funcionalidad y del resto del proyecto se consultó frecuentemente la página oficial[13] de Mongose para poder aprender cómo realizar distintos tipos de consultas a la base de datos. Como en la primera funcionalidad, el hecho de tener una aplicación ya iniciada se tradujo en que no tenía que preocuparme por la creación del servidor con Node.js. Aprendí cómo se utilizaba el enrutamiento en la aplicación gracias a Express.js. Era igual que en el proyecto plantilla de MEAN, lo que facilitó el aprendizaje. Por último, entendí cómo se usaba Angular.js en el software existente gracias al patrón de diseño MVC, el entender dónde se encontraban los ficheros que necesitaba modificar me resultó más fácil. Comprendí cómo se usaban los controladores y las directivas en la aplicación.

El desarrollar esta funcionalidad me permitió entender el software existente y adaptarme al mismo con más facilidad. Lo cual sería de ayuda para la posterior implementación de las otras funcionalidades que se realizarían.

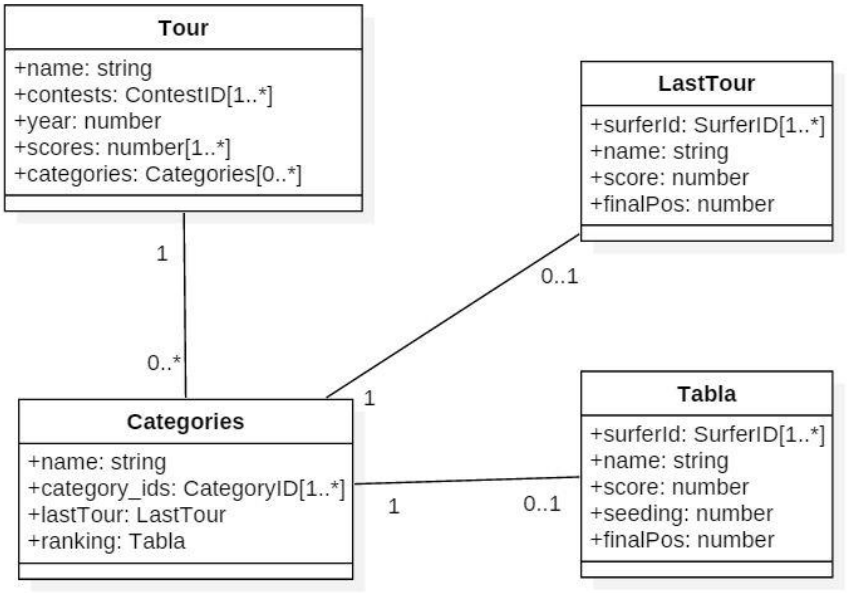
5

Análisis

En este capítulo se explicará el modelo del dominio, el diagrama de casos de uso y el flujo de eventos del proyecto desarrollado.

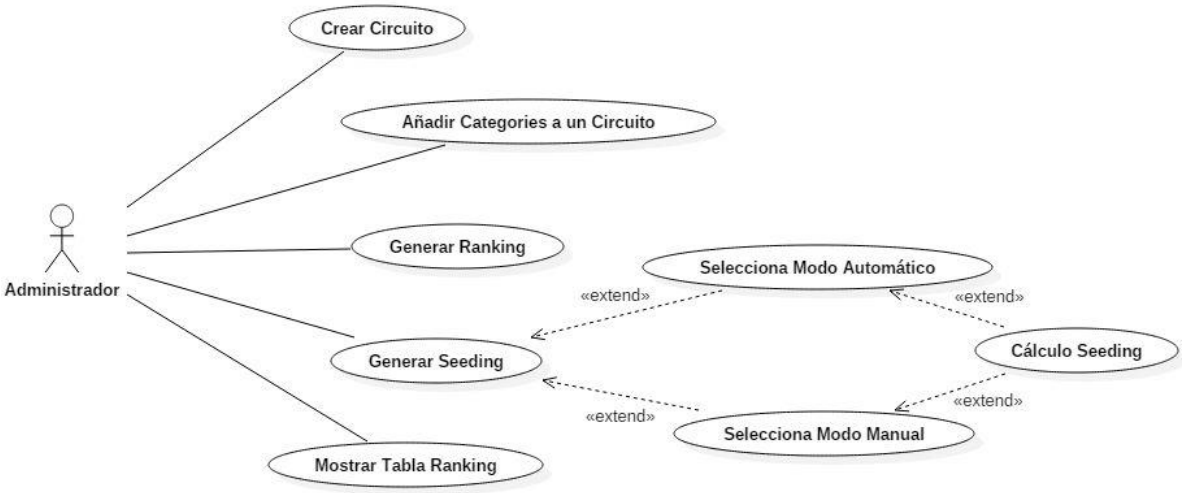
5.1. Modelo del Dominio

Para realizar el análisis de los modelos necesarios para desarrollar el proyecto hay que tener en cuenta que se parte de una aplicación web inicial y tiene un modelo definido. El modelo previo tenía que entenderlo para poder añadir mis modelos. Finalmente el modelo pensado fue el que se recoge a continuación.



5.2. Diagrama de Casos de Uso

En este apartado mostraré el diagrama de casos de uso de mi proyecto.



5.3. Flujo de Eventos

En este apartado explicaré detalladamente el comportamiento de cada uno de los casos de uso antes expuestos.

Caso de Uso:	Crear Circuito.
Descripción	Permitir la creación de un nuevo Circuito.
Actores:	Administrador.
Precondición:	El administrador debe estar autenticado como tal. Tener creados Campeonatos y Categorías en los mismos.
Flujo Normal:	1. El administrador da *click* al botón "New Tour". 2. El sistema muestra un formulario y carga en una lista desplegable todos los campeonatos de la base de datos. 3. El administrador rellena los campos: nombre del Circuito, año, nombre de Categories. 4. El administrador selecciona un campeonato de la lista desplegable. 5. (Optativo) El administrador da *click* al botón con el signo + para añadir otra lista desplegable con campeonatos. 6. El sistema carga en una lista desplegable todas las categorías del campeonato seleccionado previamente. 7. El administrador selecciona una categoría de la lista desplegable. 8. El administrador rellena el campo de puntos. 9. (Optativo) El administrador da *click* al botón con el signo + para añadir otro campo de entrada para meter más puntos. 10. El administrador da *click* al botón "Ok" para crear el Circuito. 11. El sistema comprueba que todos los datos estén completos y sean correctos, y crea el Circuito.
Flujo Alternativo:	2.A. En caso de no haber campeonatos, no se cargan y no se podrán crear Circuitos. 6.A. En caso de no haber categorías en un campeonato, no se cargan y no se podrán crear Circuitos.

	10.A. El sistema mostrará mensajes de error en caso de que algún dato esté cumplimentado de forma incorrecta.
Post Condiciones:	El Circuito se ha creado con éxito y se actualiza la lista de Circuitos.

Caso de Uso:	**Añadir Categories a Circuito.**
Descripción	Permitir la creación de un nuevo Categories en un Circuito.
Actores:	Administrador.
Precondición:	El administrador debe estar autenticado como tal. Tener creado un Circuito previamente.
Flujo Normal:	1. El administrador da *click* al botón "Admin Options" de un Circuito. 2. El sistema muestra un botón llamado "Add Categories". 3. El administrador da *click* al botón "Add Categories". 4. El sistema muestra un formulario y carga en listas desplegables todas las categorías de todos los campeonatos que tiene el Circuito. 5. El administrador rellena el campo Nombre de Categories. 6. El administrador selecciona una categoría de cada lista desplegable. 7. El administrador da *click* al botón "Ok" para crear el Categories del Circuito. 8. El sistema comprueba que todos los datos estén completos y sean correctos, actualiza el Circuito.
Flujo Alternativo:	5.A. El sistema mostrará mensajes de error en caso de que algún dato esté cumplimentado de forma incorrecta.
Post Condiciones:	Se actualiza el Circuito con el nuevo Categories, se actualiza la lista de Circuitos.

Caso de Uso:	**Generar *Ranking*.**
Descripción	Permitir calcular el *Ranking* de un Categories de un Circuito.
Actores:	Administrador.
Precondición:	El administrador debe estar autenticado como tal.

	Tener creado un Circuito con al menos un Categories previamente.
Flujo Normal:	1. El administrador da *click* encima del nombre de un Circuito. 2. El sistema muestra una lista con los Categories que posee dicho Circuito. 3. El administrador da *click* al botón "Generate *Ranking"* de un Categories de la lista. 4. El sistema nos dirige a una nueva ventana con nombre "Generar *Ranking"*. Se cargan las categorías del Circuito y se carga la tabla con los surfistas de cada categoría. El sistema ordena y asigna los puntos a los surfistas en la tabla. 5. El administrador da *click* al botón "Calcular Total". 6. El sistema calcula el *ranking* de los surfistas y asigna los valores a una lista. Se ordena la lista y se asigna sus valores a la última columna de la tabla, representará el *Ranking* Total. El sistema muestra un botón llamado "Push *Ranking* Table". 7. El administrador da *click* al botón "Push *Ranking* Table" para guardar la Tabla *Ranking*. 8. El sistema guarda la Tabla *Ranking* en el Categories del Circuito y muestra un mensaje de éxito para el administrador.
Flujo Alternativo:	4.A. Si alguna de las categorías cargadas no tiene surfistas, la columna que le corresponda quedará vacía. 4.B. Si todas las categorías cargadas no tienen surfistas, la tabla quedará vacía y no se podrá calcular el *ranking*. 6.A. Si en la tabla no hay surfistas, no se puede calcular el *ranking*. 6.B. Si en la tabla hay surfistas pero las categorías no están finalizadas, no se puede calcular el *ranking*.
Post Condiciones:	Se actualiza el Circuito con la Tabla *Ranking* del Categories elegido.

Caso de Uso:	Generar *Seeding*.
Descripción	Permitir calcular el *Seeding* de un Categories de un Circuito.
Actores:	Administrador.
Precondición:	El administrador debe estar autenticado como tal. Tener creado un Circuito con al menos un Categories previamente.

Flujo Normal:	1. El administrador da *click* encima del nombre de un Circuito. 2. El sistema muestra una lista con los Categories que posee dicho Circuito. 3. El administrador da *click* al botón "Generate *Seeding"* de un Categories de la lista. 4. El sistema nos dirige a una nueva ventana con nombre "Generar *Seeding"*. Se cargan las categorías del Circuito y se carga la tabla con los surfistas de cada categoría. El sistema ordena y asigna los puntos a los surfistas en la tabla. Se muestran dos botones: uno llamado "Usar *Ranking*Total guardado" u otro llamado "Usar otro *Ranking*Total". 5. El administrador da *click* al botón "Usar otro *Ranking*Total". 6. (Si se da *click* a otro *Ranking*Total) El sistema muestra dos botones: uno llamado "modo automático" y otro "modo manual". 7. El administrador da *click* botón "modo automático" y se va al caso de uso Seleccionar Modo Automático. 8. El administrador da *click* al botón "modo manual" y se va al caso de uso Seleccionar Modo Manual. 9. (Si se da click a *Ranking*Total guardado) El sistema muestra un botón para modo manual y una lista desplegable. 10. Si el administrador da *click* al botón "modo manual" iría al caso de uso Seleccionar Modo Manual. En caso de no dar *click* se va al caso de uso Cálculo *Seeding*.
Flujo Alternativo:	4.A. No se muestra el botón "Usar *Ranking*Total guardado", si el Categories elegido no tiene uno guardado previamente. La primera vez que se accede no se tiene. Al usar el modo manual se puede guardar este dato en el Circuito.
Post Condiciones:	Se ha seleccionado cómo usar el *Ranking*Total, usar uno guardado o usar otro. Se ha llenado la tabla con los surfistas de las categorías del Circuito.

Caso de Uso:	Seleccionar Modo Automático.
Descripción	Seleccionar el modo automático y sus repercusiones en el cálculo de *seeding*.
Actores:	Administrador.
Precondición:	Se debe haber clickado previamente el botón modo automático.
Flujo Normal:	1. El sistema muestra una lista desplegable y se carga con Circuitos que tenga Categories con *Ranking*Total. 2. El administrador selecciona un Circuito de la lista desplegable para elegir un *Ranking*Total. 3. El sistema carga el *Ranking*Total elegido y lo asigna en la primera columna de la tabla. Se pasa al caso de uso Cálculo de *Seeding*.
Flujo Alternativo:	1.A. Si no existen Circuitos que tenga Categories con *Ranking*Total en la base de datos, no se cargan. No se puede realizar los otros pasos ni calcular el *seeding*.
Post Condiciones:	Se ha seleccionado un *Ranking*Total. Se ha llenado la primera columna de la tabla.

Caso de Uso:	Seleccionar Modo Manual.
Descripción	Seleccionar el modo manual y sus repercusiones en el cálculo de *seeding*.
Actores:	Administrador.
Precondición:	Se debe haber clickado previamente el botón modo manual.
Flujo Normal:	1. El sistema muestra una lista vacía llamada *Ranking* Manual y tres botones: "Add *Ranking*Manual To *Ranking*Total", "Empty *Ranking*Total" y "Save *Ranking*Total". También se muestra un botón "Add" en cada surfista de la tabla. 2. El administrador da *click* al botón "Add", esto añade al surfista en la lista *Ranking* Manual. Se pueden añadir varios surfistas y no se pueden repetir en la lista. 3. Si el administrador da *click* al botón "Empty *Ranking*Total" se vacía la lista *Ranking* Manual y la primera columna de la tabla.

	4. Si el administrador da *click* al botón "Add *Ranking*Manual To *Ranking*Total", se ordena la lista del *Ranking* Manual y se asigna a la primera columna de la tabla que representa el *Ranking* Total del año pasado. Se pasa al caso de uso Cálculo de *Seeding*.
Flujo Alternativo:	1.A. En caso de que la tabla esté vacía porque no existen surfistas en las categorías, no se puede colocar el botón "Add" en las celdas de los surfistas. No se podrá realizar el resto de pasos ni calcular el *seeding*. 4.A. Si la lista *Ranking* Manual está vacía, no se puede asignar a la primera columna de la tabla ni se podrá pasar al caso de uso Cálculo de *Seeding*.
Post Condiciones:	Se ha creado un *Ranking*Total de forma manual. Se ha llenado la primera columna de la tabla.

Caso de Uso:	Cálculo del *Seeding*.
Descripción	Se calcula el *seeding* de los surfistas de una categoría.
Actores:	Administrador.
Precondición:	Se debe tener un *Ranking*Total. La primera columna de la tabla debe estar llena.
Flujo Normal:	1. El sistema muestra una lista desplegable y se carga con las categorías no finalizadas del Circuito. 2. El administrador selecciona una categoría de la lista desplegable. 3. El sistema resalta la categoría seleccionada y muestra el *seeding* de los surfistas de dicha categoría. También muestra un botón llamado "Calcular *Seeding"*. 4. El administrador da *click* al botón "Calcular *Seeding"*. 5. El sistema calcula el *seeding* de los surfistas, se asigna la suma de los puntos de los surfistas en la última columna de la tabla. Se asigna el nuevo *seeding* a los surfistas de la categoría seleccionada previamente. Se muestra una lista desplegable para elegir si el *seeding* es definitivo o no.

	6. El administrador selecciona alguna opción de la lista, si es definitivo o no. 7. El sistema muestra un botón "Push *Seeding"*. 8. El administrador da *click* al botón "Push *Seeding"* y se guarda el nuevo *seeding* de los surfistas.
Flujo Alternativo:	1.A. Si no existe ninguna categoría no finalizada en el Circuito no se carga la lista desplegable. No se podrá realizar el resto de pasos ni calcular el *seeding*.
Post Condiciones:	Se actualiza la Categoría con el nuevo *seeding* de los surfistas.

Caso de Uso:	Mostrar Tabla *Ranking*.
Descripción	Permitir la Tabla del *Ranking* de un Categories de un Circuito.
Actores:	Administrador.
Precondición:	El administrador debe estar autenticado como tal. Tener creado previamente un Circuito con al menos un Categories.
Flujo Normal:	1. El administrador da *click* encima del nombre de un Circuito. 2. El sistema muestra una lista con los Categories que posee dicho Circuito. 3. El administrador da *click* al botón "Show *Ranking"* de un Categories de la lista. 4. El sistema nos dirige a una nueva ventana con nombre Tabla *Ranking*. Se cargan las categorías del Circuito y se carga la tabla *ranking* del Categories del Circuito elegido. Se cargan todos los valores de los surfistas de la tabla *ranking* en la tabla de la ventana.
Flujo Alternativo:	4.A. Si el Categories elegido no tiene una tabla *ranking* guardada previamente, la tabla de la ventana queda vacía y se muestra un mensaje por pantalla al administrador.
Post Condiciones:	Se muestra la Tabla *Ranking* del Categories elegido.

6

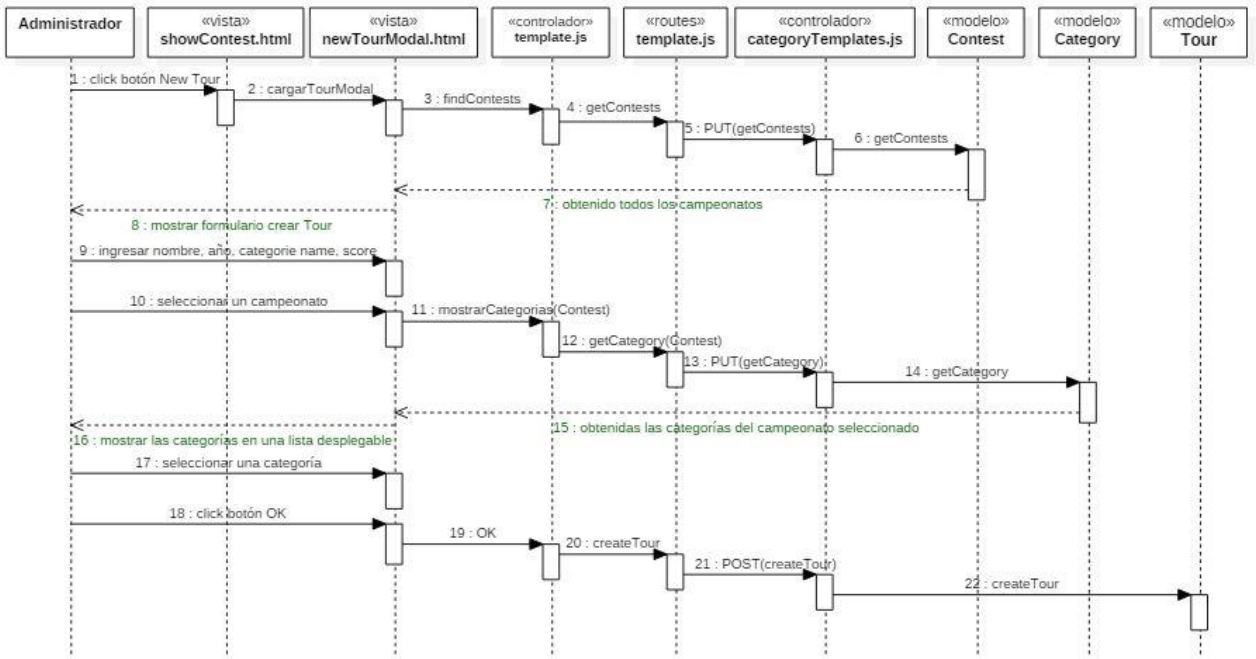
Diseño

En este capítulo se mostrarán los diagramas de secuencia de los casos de uso expuestos en el capítulo de Análisis.

6.1. Diagramas de Secuencia

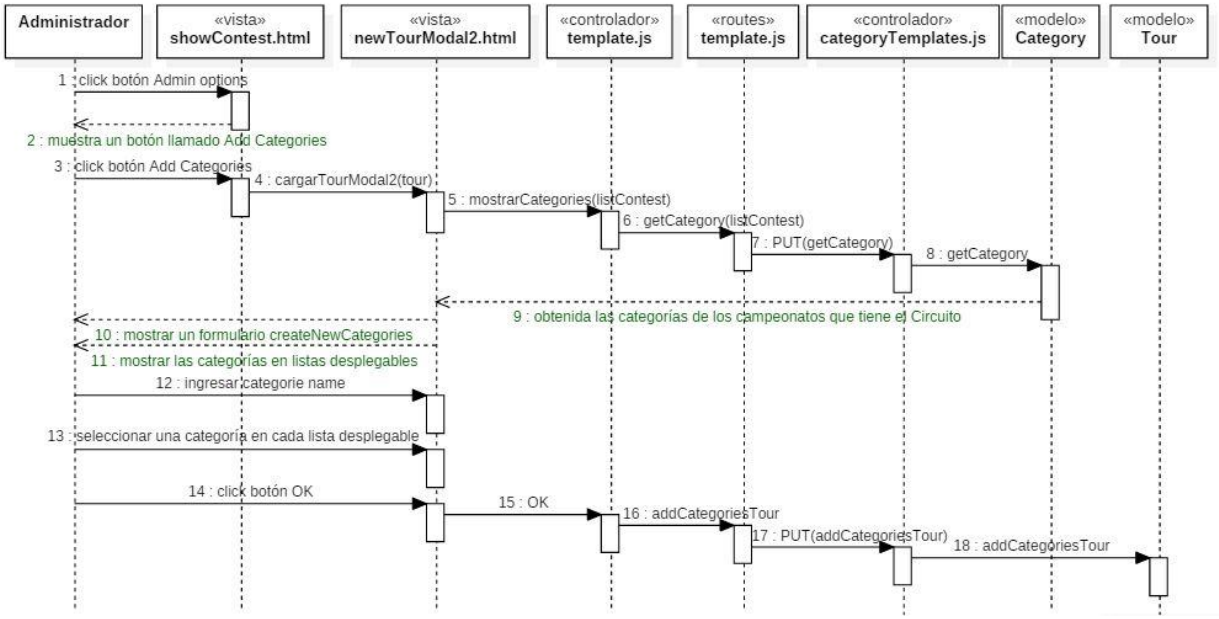
Se mostrarán los diagramas de secuencia para cada uno de los casos de uso que se implementaron y una breve explicación introductoria.

Crear Circuito:



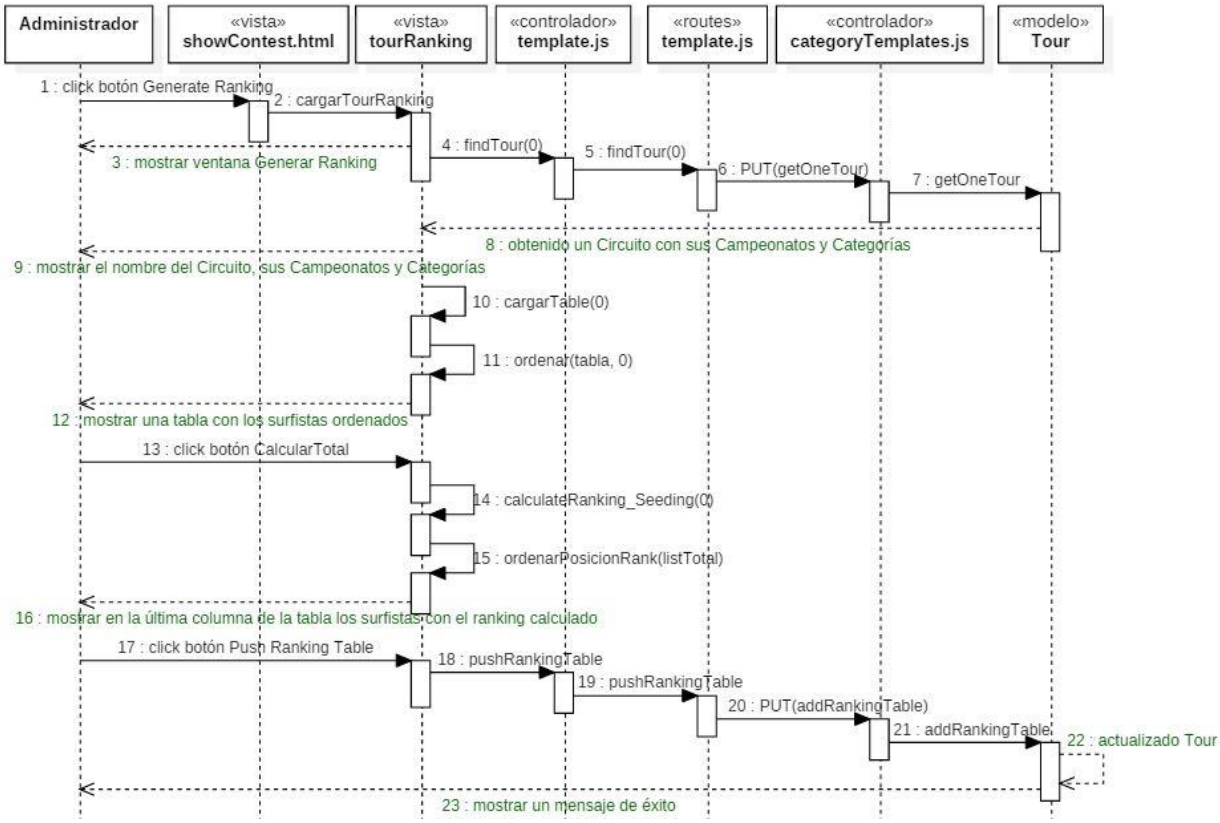
El caso Crear Circuitos se encarga de la creación de circuitos mediante el correcto ingreso de datos en el formulario mostrado. Se establece una conexión entre el cliente y el servidor para obtener los campeonatos en la base de datos y luego sus correspondientes categorías. Cuando todos los datos estén ingresados, se pasan todos al servidor y se crea el Circuito basándose en el modelo Tour.

Añadir Categories a un Circuito:



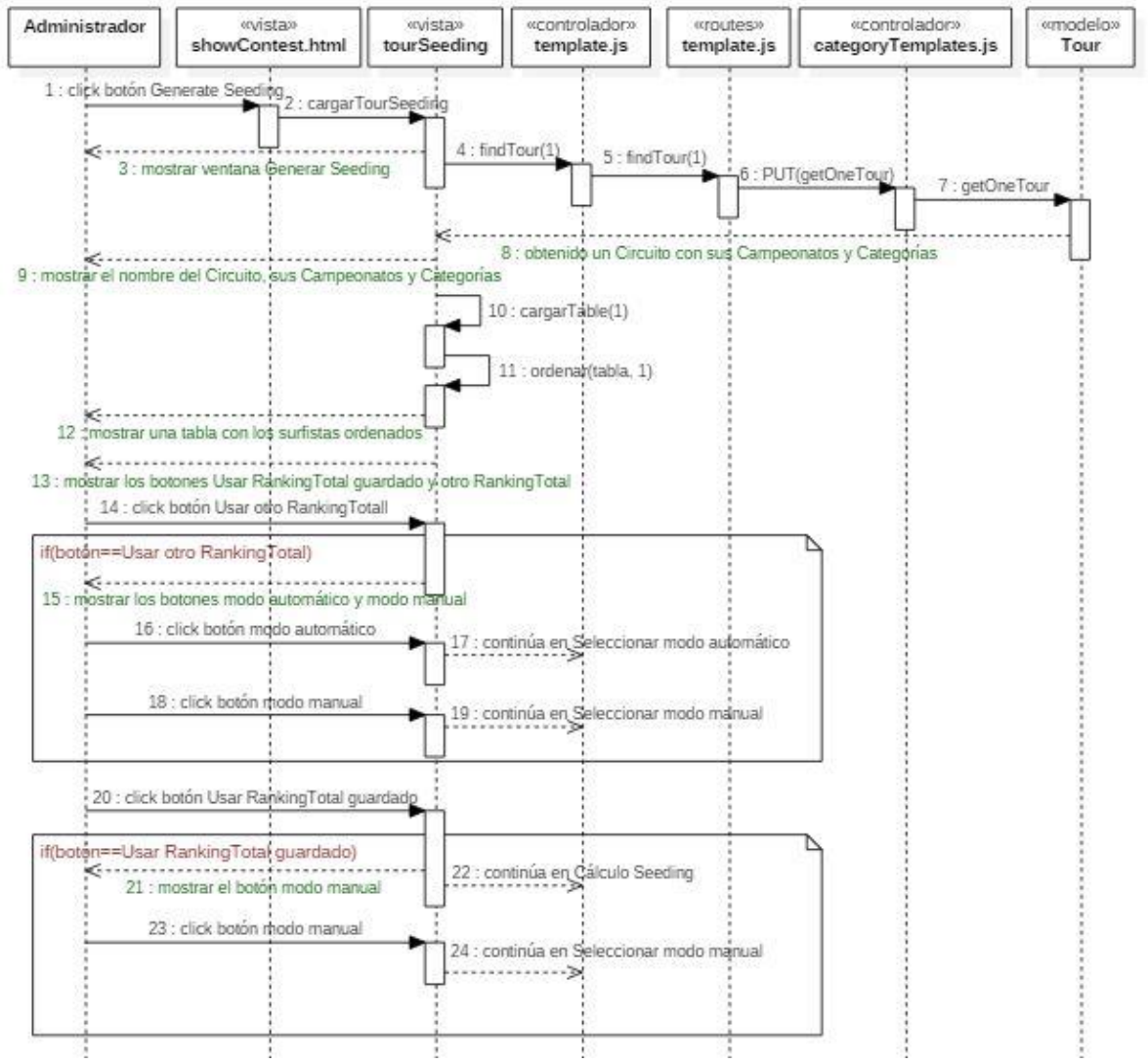
El caso Añadir Categories a un Circuito se encarga de la creación de un nuevo Categories. Para ello, se ingresan los datos en el formulario mostrado. Se establece una conexión entre el cliente y el servidor para obtener categorías de la base de datos. Esas categorías se eligen basándose en los campeonatos del Circuito elegido. Se seleccionan las categorías deseadas y se ingresa el nombre del Categories a crear. Cuando todos los datos estén ingresados, se pasan todos al servidor y se actualiza el Circuito con el nuevo Categories.

Generar *Ranking*:



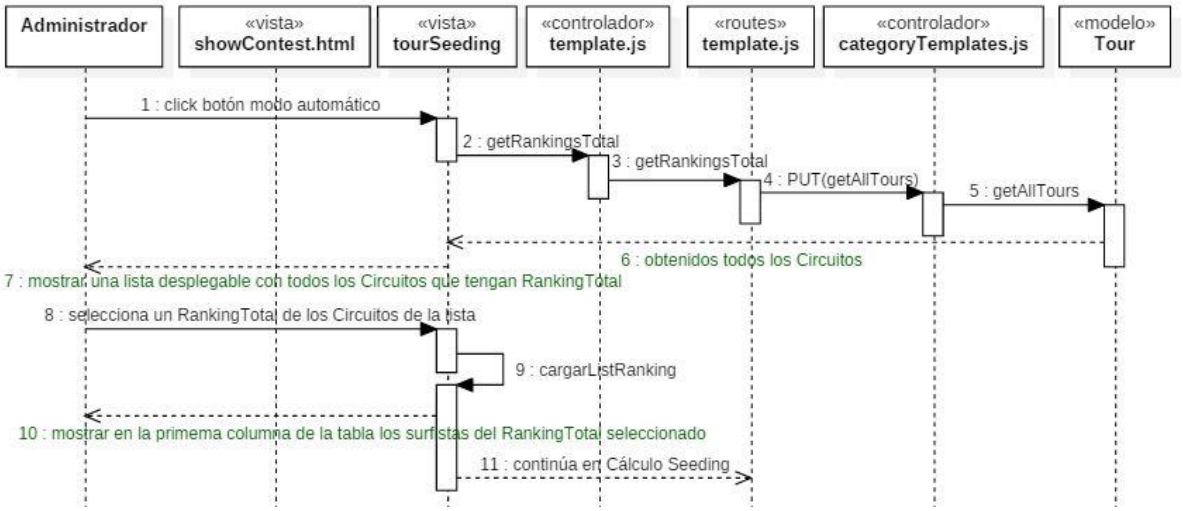
El caso Generar *Ranking* se encarga del cálculo del *ranking* del Categories de un Circuito. Para ello se establece una conexión entre el cliente y el servidor para obtener el Circuito de la base de datos. Se carga la tabla con los surfistas de las categorías del Circuito, se ordena la tabla y se muestra. A continuación, se calcula el *ranking* en el lado cliente y, por último, se guardan los cambios en el servidor. Se actualiza el Circuito con la tabla *ranking* calculada con anterioridad.

Generar *Seeding*:



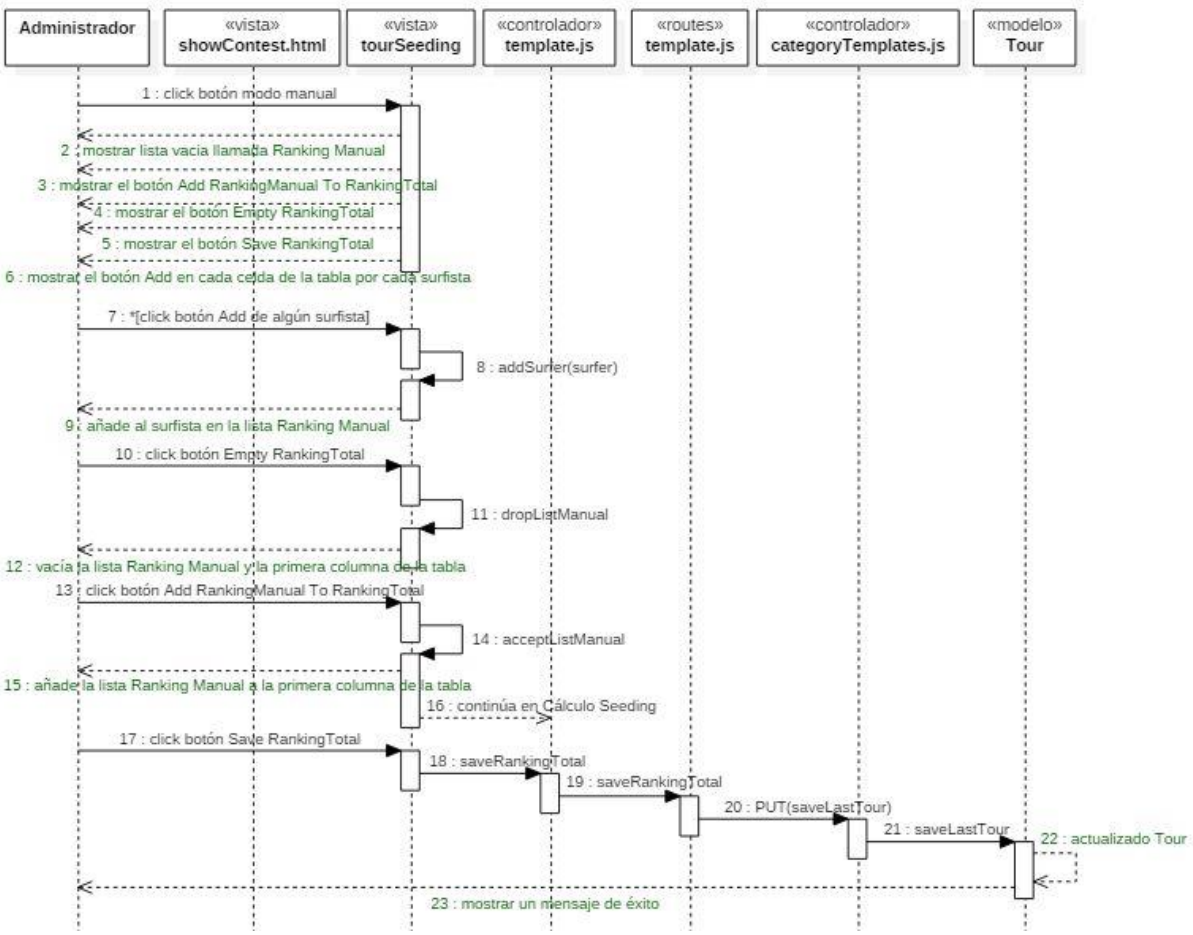
El caso Generar *Seeding* se encarga de cargar los surfistas en la tabla y de elegir cómo usar el *Ranking* Total. Para ello, se establece una conexión entre el cliente y el servidor para obtener el Circuito de la base de datos. Se carga la tabla con los surfistas de las categorías del Circuito, se ordena la tabla y se muestra. A continuación, se muestran los botones para elegir cómo usar el *Ranking*Total, ya sea utilizando uno guardado previamente (la primera vez nunca se tiene) o elegir utilizar otro. Al elegir una u otra opción se puede seguir por diferentes vías que encaminarán a los casos de uso expuestos más adelante.

Seleccionar Modo Automático:



El caso Seleccionar Modo Automático se encarga de seleccionar el *Ranking*Total que se va a usar y, además, añadir dicho *ranking* a la primera columna de la tabla. Para ello, se establece una conexión entre el cliente y el servidor para obtener todos los Circuitos de la base de datos. Se carga la lista desplegable con los Circuitos que tengan *Ranking*Total para poder usarlos. Luego, se selecciona un *Ranking*Total y se carga la primera columna de la tabla con los surfistas del *Ranking*Total elegido. A continuación, seguirá el caso de uso Cálculo *Seeding*.

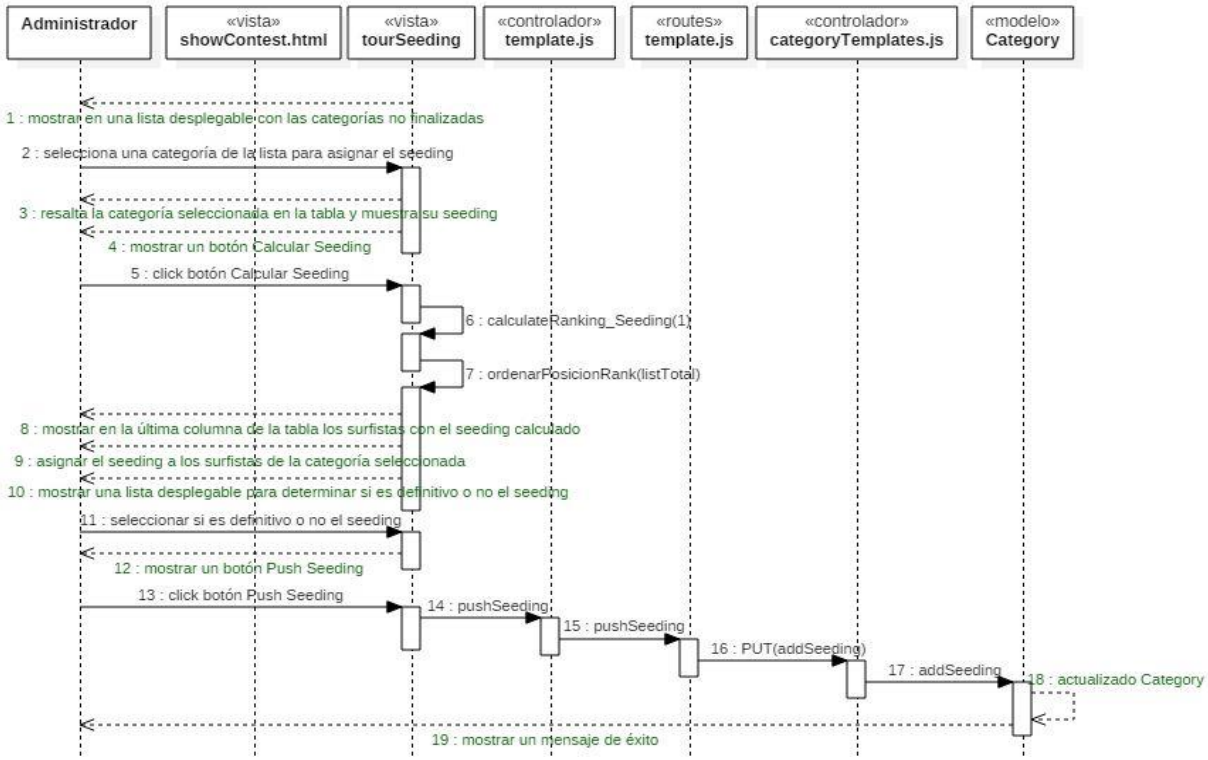
Selecciona Modo Manual:



El caso Seleccionar Modo Manual se encarga de seleccionar el *Ranking*Total que se va a usar y, además, añadir dicho *ranking* a la primera columna de la tabla. Para ello, se crea una lista *Ranking*Manual que nos permitirá crear el *Ranking*Total que nosotros queramos utilizando los surfistas disponibles en la tabla. Si se le da al botón "Add" de los surfistas, éstos son añadidos en la lista *Ranking*Manual. El botón "Empty *Ranking*Total" vaciará dicha lista y la primera columna de la tabla. El botón "Add*Ranking*Manual To *Ranking*Total" cargará el *Ranking*Manual en la primera columna de la tabla. Con esto, se podría continuar al caso de uso Cálculo *Seeding*.

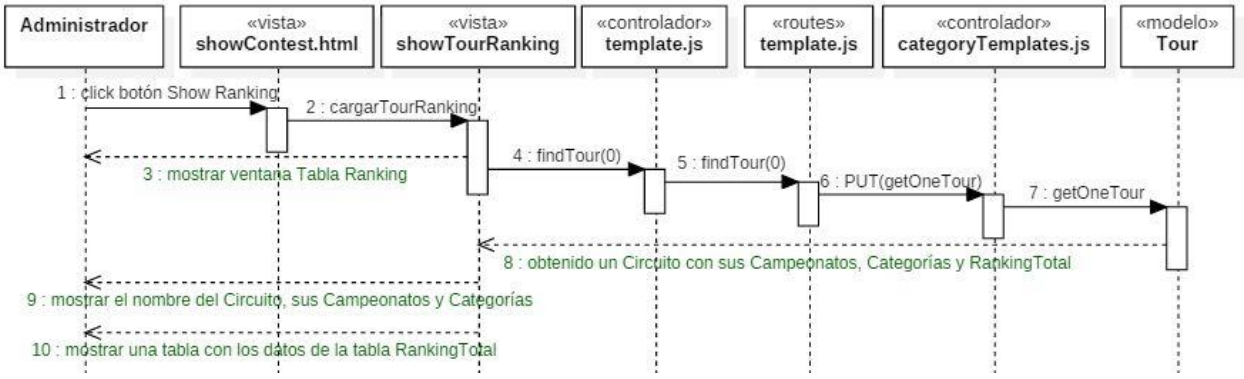
Si se desea se puede guardar el *Ranking*Total creado en el Circuito, este *Ranking*Total representará el *Ranking* del año pasado, de esta manera, la próxima vez que se ingrese en la funcionalidad se puede optar por la opción de Usar un *Ranking*Total guardado.

Cálculo *Seeding*:



El caso Cálculo *Seeding* se encarga de realizar el cálculo del *seeding* del Categories de un Circuito. Para ello, primeramente, se ha cargado una lista desplegable con las categorías no finalizadas del Circuito actual. Se selecciona una categoría de esa lista, esta categoría será a la que se le asigne el *seeding*. La categoría elegida se resalta y aparece el *seeding* que tiene por defecto. Al darle *click* al botón "Calcular *Seeding"* se empiezan a realizar los cálculos y, finalmente, el resultado se carga en la última columna de la tabla. A su vez, se asigna el nuevo *seeding* a los surfistas de la categoría elegida. Se selecciona si se quiere poner el *seeding* como definitivo o aún no. Y finalmente, se guardan los cambios estableciendo una conexión con el lado servidor. Se actualiza la Categoría elegida con el nuevo *seeding* en la base de datos.

Mostrar Tabla *Ranking*:



El caso Mostrar *Ranking* se encarga de mostrar la tabla *ranking* que tiene el Categories de un Circuito. Para ello, se establece una conexión entre el lado cliente y el lado servidor para obtener un Circuito con todos sus datos. Finalmente, se carga la tabla con todos los datos de la tabla *ranking* que posee el Categories del Circuito.

7

Implementación

En este capítulo explicaré cómo es la estructura de la aplicación, los módulos usados, la identificación de los ficheros existentes necesarios para el desarrollo del proyecto y la creación de nuevos ficheros para conseguir dicho objetivo.

7.1. Estructura de la aplicación

Para la implementación de las funcionalidades tuve que entender la estructura de la aplicación, pero gracias al patrón MVC que Angular.js utiliza por defecto, facilitó la organización del código y una mejor comprensión del mismo.

Para el desarrollo de las funcionalidades a crear no se necesitaba entender todo el software preexistente, pero sí lo necesario para poder utilizar funciones ya creadas y no replicar código de forma innecesaria. Después del proceso de adaptación tanto a las tecnologías como al software, el cual se habla en el capítulo de Adaptación, empecé a desarrollar las funcionalidades del proyecto.

El software preexistente tiene un total de ocho módulos o carpetas donde se distribuyen diversas funcionalidades ya implementadas por el DPE. Se identificó que solamente se usarían dos módulos para el desarrollo del proyecto, uno llamado *contests* y otro llamado *templates*. Entre estos dos módulos se desarrollaría casi toda la implementación en el segundo.

7.1.1. Módulo *Contests*

Fue necesario usar este módulo ya que en el mismo se encontraba la definición de los modelos que se usan en la aplicación. Y como expliqué en el alcance del capítulo Objetivos, se tenía que crear un nuevo modelo *Tour* que representaría a los Circuitos. Se creó un fichero con nombre tour.js y se definió el modelo con sus correspondientes datos.

```
var mongoose = require('mongoose'),
    Schema = mongoose.Schema;

/**
 * Tour Schema
 */
var TourSchema = new Schema({
    update: {type: Boolean, default: true},
    messages: [],
    contests: [{ type: Schema.ObjectId, ref: 'Contest' }],
    name: {type: String, default: ''},
    year: Number,
    scores:[],
    categories: [{
        name: String,
        category_ids: [{ type: Schema.ObjectId, ref: 'Category' }],
        lastTour: [{
            surferId: { type: Schema.ObjectId, ref: 'Surfer' },
            name: String,
            score: Number,
            finalPos: Number
        }],
        ranking: {
            row: [{
                    col: [{
                        surferId: { type: Schema.ObjectId, ref: 'Surfer' },
                        name: String,
                        score: Number,
                        seeding: Number,
                        finalPos: Number
                    }]
            }]
        }
    }]
});
```

Imagen 3: Definición del modelo *Tour* en tour.js.

7.1.1.1 Identificación de ficheros existentes necesarios.

También se identificaron los ficheros existentes necesarios para realizar las funcionalidades deseadas. Éstos serían:

- contest.js (*server*, modelo): Modelo que representa un campeonato.
- category.js (*server*, modelo): Modelo que representa una categoría.

7.1.1.2 Ficheros creados necesarios.

Como se explicó previamente, se creó un fichero para definir el modelo circuito. Éste es:

- tour.js (*server*, modelo): Modelo que representa un circuito.

7.1.2. Módulo *Templates*

Este módulo representa todas las funcionalidades que necesita el administrador para realizar la creación de campeonatos, sus categorías y la asignación del *seeding* de los surfistas de forma manual, entre otras funciones a las que solamente tiene acceso el administrador o el informático responsable de tener el control durante una competición de surf. Esto quiere decir que las funcionalidades que desarrollaría para el proyecto estaban dirigidas a que fueran usadas solamente por el administrador y no por los clientes de la aplicación, aunque los resultados obtenidos de estas funcionalidades sí serían accesibles para los clientes de otras maneras, como, por ejemplo, el ver la tabla del *ranking* en la aplicación web o consultando el *seeding* que tiene un surfista en el siguiente campeonato.

En este módulo tuve que identificar las vistas y los controladores que iba necesitar. Los modelos a usar estaban identificados previamente ya que se encontraban en el módulo *contests*. Gracias al patrón MVC de Angular.js, la organización de las carpetas dentro del módulo era intuitiva y permitía encontrar los ficheros que se buscaban con suma facilidad. También, me permitió identificar los ficheros que se usaban en el lado cliente como en el lado servidor.

Imagen 4: Organización del módulo *templates*. La carpeta *public* representa el lado cliente y *server* el lado servidor.

7.1.2.1 Identificación de ficheros existentes necesarios.

Se identificaron los ficheros existentes necesarios para realizar las funcionalidades deseadas. Éstos serían:

- showContest.html (*public*, vista): Esta vista muestra una lista con todos los campeonatos de la base de datos. Se añadió código para permitir mostrar también una lista con todos los circuitos de la base de datos. Además, se añadieron los botones que darían acceso a otras funcionalidades.

```
<div collapse="!tour.isCollapsed">
    <div ng-repeat="category in tour.categories">
        <h3>
            <a data-ng-href="#!/tour/goToEditTour/tour/{{tour._id}}/categories/{{category._id}}">{{category.name}} (
            {{category.type}})</a>
            <a data-ng-href="#!/tour/goToEditTour/tour/{{tour._id}}/categories/{{category._id}}" class="btn btn-default btn
            -sm" role="button">Generate Ranking</a>
            <a data-ng-href="#!/tour/generateSeeding/tour/{{tour._id}}/categories/{{category._id}}" class="btn btn-default
            btn-sm" role="button">Generate Seeding</a>
            <a data-ng-href="#!/tour/showRanking/tour/{{tour._id}}/categories/{{category._id}}" class="btn btn-default btn-
            sm" role="button">Show Ranking</a>
        </h3>
    </div>
</div>
```

Imagen 5: Fragmento del código que muestra los botones para realizar otras funcionalidades.

- template.js (*public*, controlador): Este fichero representa el controlador de la aplicación en el lado cliente. Contiene funciones que ayudan a las vistas a realizar distintos comportamientos. En este fichero se añadieron todas las funciones necesarias para la creación de tour, el cálculo de *ranking* y el cálculo de *seeding* que correspondían con el lado cliente y, además, las funciones que solicitaban datos al lado servidor.

```
/**
 * Función que hace una llamada al servidor para obtener un tour.
 * Autor: Andres
 */
    $scope.findTour = function(caso){
        $scope.tourId = $stateParams.tourId;
        $scope.categoriesId = $stateParams.categoriesId;

            $http.put('/tour/goToEditTour/tour/:tourId/categories/:categoriesId',{
                param1: $scope.tourId,
                param2: $scope.categoriesId
            })
                .success(function(response) {
                    $scope.tour = response.data;
                    $scope.categories = response.data1;
                    $scope.rankingTotal2 = response.data3;
                    // Bucle for para añadir en las categorias el contestName para poder usar ese dato.
                    for (var x = 0; x < response.data2.length; x++) {
                        for (var i = 0; i < response.data2.length; i++) {
                            if($scope.categories.category_ids[x].contestId == response.data2[i].contestId){
                                $scope.categories.category_ids[x].contestName = response.data2[i].contestName;
                            }
                        }
                    }
                    $scope.cargarTable(caso);
                })
                .error(function(err) {
                  console.log('error', err);
                });
    };
```

Imagen 6: Función *findTour* del controlador del lado cliente.

- **template.js (*server*, *routes*):** Este fichero representa el enrutamiento o direccionamiento que proporciona Express.js. Permite atender solicitudes del lado cliente y direccionar éstas al controlador correspondiente. Se añadieron nuevas rutas para atender las solicitudes que las funcionalidades creadas por mí realizarían al lado servidor.

```
app.route('/tours/createTour')
  .post(auth.requiresLogin, categoriesTemplates.createTour)
  .put(auth.requiresLogin, categoriesTemplates.addCategoriesTour);
app.route('/tours/getCategories')
  .put(auth.requiresLogin, categoriesTemplates.getCategoryContest);
app.route('/tours/getTours')
  .get(auth.requiresLogin, categoriesTemplates.getAllTours);
```

Imagen 7: Fragmento de código con algunas rutas y sus direccionamientos.

- **categoryTemplates.js (*server*, controlador):** Este fichero representa el controlador de la aplicación en el lado servidor. Contiene funciones que sirven para realizar consultas a la base de datos por medio de los modelos. Se añadieron varias funciones para realizar las consultas que necesitaban las funcionalidades creadas.

```
/**
 * Guardar lastTour en la categories del Tour dado.
 * Autor: Andres
 */
exports.saveLastTour = function(req, res) {
    Tour.findOne({_id: req.body.param1}).
            exec(function (err, tour) {
                    if (err) return handleError(err);

                    for (var i = 0; i < tour.categories.length; i++) {
                        if(tour.categories[i]._id==req.body.param2._id){
                            tour.categories[i].lastTour = req.body.param3;
                        }
                    }
                    tour.save(function(err) {
                        if (err) return er.handler(res, err, 'Error al añadir lastTour el tour, vuelva a intentarlo.');
                        res.jsonp(tour);
                    });
            });
};
```

Imagen 8: Función *saveLastTour* del controlador del lado servidor.

7.1.2.2 Ficheros creados necesarios.

También se crearon nuevos ficheros que ayudarían a realizar las funcionalidades deseadas. Éstos serían:

- **newTourModal.html (*public*, vista):** Esta vista es un formulario que solicitará los datos necesarios para la creación de un circuito.

```
<div class="col-md-6" data-ng-controller="CategoriesTemplatesController" data-ng-init="findContests()" data-ng-show="
contentLoaded">
    <label>Contests: </label>
    <select class='form-control' ng-repeat='cp in model.contests' ng-change="mostrarCategories(model.contests)" data-ng-model='
    cp.contestId'>
        <option ng-repeat='contest in contests' value='{{contest._id}}'>{{contest.name}}</option>
    </select>
</div>
```

Imagen 9: Fragmento de código. Muestra cómo se carga el elemento *select*.

- **newTourModal2.html (*public*, vista):** Esta vista es un formulario que solicitará los datos necesarios para añadir un nuevo Categories en el circuito.

- tour*Ranking*.html (*public*, vista): Esta vista mostrará una tabla que contendrá todos los surfistas de las categorías del circuito con sus correspondientes puntos. Se podrá calcular el *ranking* del circuito si se cumplen las condiciones adecuadas, se hablará más de ello en el capítulo de Pruebas. Además, se podrá guardar la tabla *ranking* calculada en la base de datos.

```
<tbody>
    <tr ng-repeat="row in tableData">
        <td class="text-center col-sm-1"><b>{{$index+1}}</b></td>
        <td class="text-center col-sm-1" ng-repeat="cell in row track by $index">
            <span data-ng-show="cell"><b>{{cell.finalPos}}</b>- {{cell.name}} <b>{{cell.score}}</b></span>
        </td>
    </tr>
</tbody>
```

Imagen 10: Fragmento de código. Muestra cómo se rellenan algunas celdas de la tabla.

- tour*Seeding*.html (*public*, vista): Esta vista mostrará una tabla que contendrá todos los surfistas de las categorías del circuito con sus correspondientes puntos. Se podrá:
    - Elegir cómo usar el *ranking* total, si uno guardado u otro.
    - Elegir cómo añadir dicho *ranking* total, de forma manual o automática.
    - Elegir la categoría de los surfistas a los que se desee añadir el *seeding*.
    - Calcular el *seeding* del circuito si se cumplen las condiciones adecuadas, se hablará más de ello en el capítulo de Pruebas. Además, se podrá guardar el nuevo *seeding* calculado de los surfistas en la base de datos.

- showTour*Ranking*.html (*public*, vista): Esta vista mostrará una tabla que contendrá todos los datos de la tabla *ranking* que posee el Categories elegido en la base de datos.

7.2. Número de líneas de código al inicio y al final del proyecto

Mostraré el número de líneas de código, tanto de los ficheros existentes que fueron necesarios pero con código añadido, como de los ficheros creados desde cero.

FICHEROS EXISTENTES			
Nombre	Líneas de código al inicio	Líneas de código añadidas	Líneas de código totales
showContests.html (*public*, vista)	118	44	162
template.js (*public*, controlador)	432	843	1275

template.js (*server*, router)	64	19	83
categoryTemplates.js (*server*, cont..)	714	279	993
FICHEROS CREADOS			
tour.js (*public*, vista)	0	42	42
newTourModal.html (*public*, vista)	0	69	69
newTourModal2.html (*public*, vista)	0	40	40
tour*Ranking*.html (*public*, vista)	0	37	37
tour*Seeding*.html (*public*, vista)	0	131	131
showTour*Ranking*.html (*public*, vista)	0	51	51
TOTAL	**1328**	**1555**	**2883**

Esto refleja la aportación que han tenido las funcionalidades desarrolladas en los ficheros de la aplicación que había previamente. Para la implementación, tuve que entender bastante el software existente, para qué servían ciertas funciones, qué ficheros me eran útiles para el desarrollo, mirar cómo funcionaban ciertas funciones donde parte del código me podría ser de interés, reutilización de funciones, etc. Hubo un gran trabajo de adaptación al software.

8

Herramientas y Entorno de desarrollo

En este capítulo se expondrán las herramientas y el entorno de desarrollo utilizado durante el proyecto para la correcta implementación de las funcionalidades deseadas.

8.1. Entorno de Desarrollo

Desde el inicio del proyecto decidí utilizar Windows como entorno de trabajo, ya que fue el entorno elegido por el DPE y, además, era un entorno cómodo para mí y de fácil manejo.

Al ser este proyecto un desarrollo web, en principio iba a utilizar como editor de texto "Notepad++", pero el DPE me recomendó el uso de "Sublime Text2" ya que tiene resaltador de sintaxis y generador automático de código.

8.2. Herramientas

Robomongo

Imagen 11: Logo de Robomongo.

Es una herramienta multiplataforma con la que podemos administrar gráficamente nuestras bases de datos MongoDB. Muy útil para ver los datos de forma más gráfica. Tuve que descargar una versión anterior de Robomongo ya que la versión actual tenía un problema, cuando quería eliminar un elemento de la base de datos, los cambios no se aplicaban, por eso descargué la versión anterior para probarlo y vi que en esa sí funcionaba la eliminación de elementos.

SourceTree

Imagen 12: Logo SourceTree.

Es una GUI[4] para manejar repositorios Git. Desde la aplicación se nos permite:

- Crear y clonar repositorios Git. Además, permite integrarse perfectamente con Bitbucket o Github.
- Realizar commit, push, pull y merge de nuestros archivos.
- Detectar y resolver conflictos.
- Consultar el historial de cambios de nuestros repositorios.

Bitbucket

Imagen 13: Logo de Bitbucket.

[4] Interfaz Gráfica de Usuario.

Es un servicio de alojamiento web para proyectos que usen un sistema de control de versiones como puede ser Git. Es similar a Github, pero la razón que me llevó a utilizar este servicio en vez de Github, el cual ya conocía y había usado durante la carrera, fue que nos permitía crear repositorios privados sin tener pagar, ya que con Github para poder crear dichos repositorios hay que crear una cuenta de pago. El uso del repositorio privado era necesario debido a que parte del código de la aplicación web ya creada por la empresa estaría expuesta y solamente debía tener acceso yo.

Git

Imagen 14: Logo de Git.

Es un software de control de versiones. La buena experiencia del uso de Git durante la carrera en la asignatura de Sistemas Web y su uso en las prácticas realizadas en MagnaSIS, me mostraron la utilidad del control sobre las distintas versiones que genero mientras programo.

9

Pruebas

En este capítulo se explicarán las distintas pruebas realizadas para verificar el correcto funcionamiento de las funcionalidades desarrolladas durante el proyecto.

9.1. Pruebas de la Funcionalidad Crear Circuito

Para crear un Tour (Circuito) se deben rellenar los datos del formulario correctamente, en caso contrario aparecerán mensajes de error y no se creará el Tour.

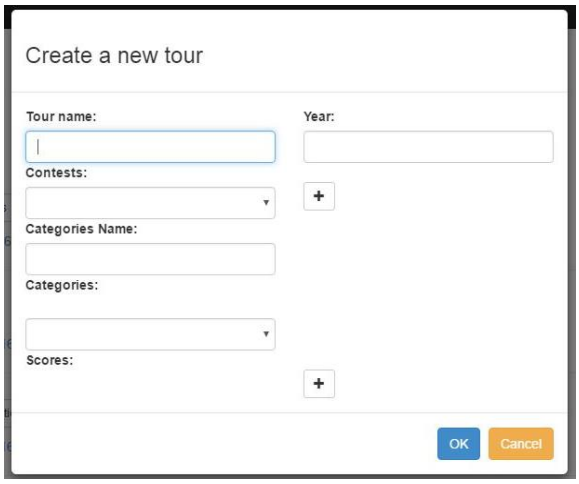


Imagen 15: Formulario para Crear Tour con los campos vacíos.

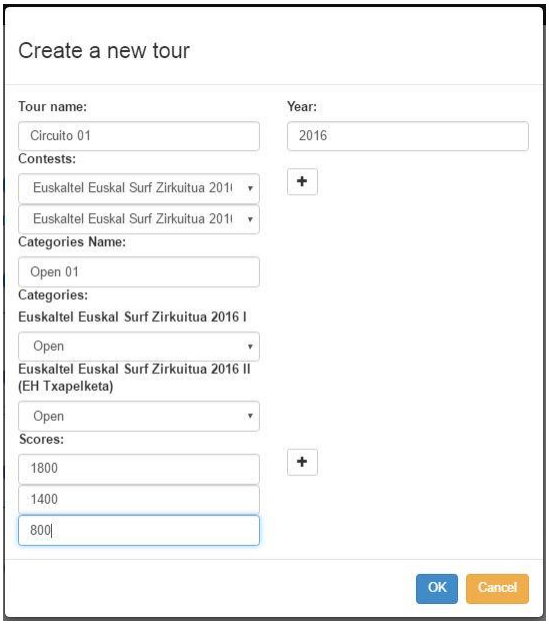


Imagen 16: Formulario para Crear Tour con los campos llenos.

Se han realizado varias validaciones:

- El campo *Tour Name* es un campo *input* de tipo texto. Permite escribir tanto caracteres como números, pero se no pueden añadir los dos caracteres “<”, “>”. De esta manera, trato de evitar que se añada código Javascript en el campo.
- El campo *Year* es un campo de tipo *number* y no permite el ingreso de caracteres. La entrada será el año en que se realiza el campeonato.
- El campo *Contests* es una lista desplegable que muestra todos los campeonatos.
- El campo *Categories Name* es un campo *input* de tipo texto, permite escribir tanto caracteres como números, pero se no pueden añadir los dos caracteres “<”, “>”.
- El campo Categories es una lista desplegable que muestra todas las categorías del campeonato seleccionado anteriormente en el campo *Contests*.
- El campo *Scores* es un campo de tipo *number* y no permite el ingreso de caracteres, la entrada será una de las puntuaciones de la lista de puntuaciones que se debe asignar al Circuito.

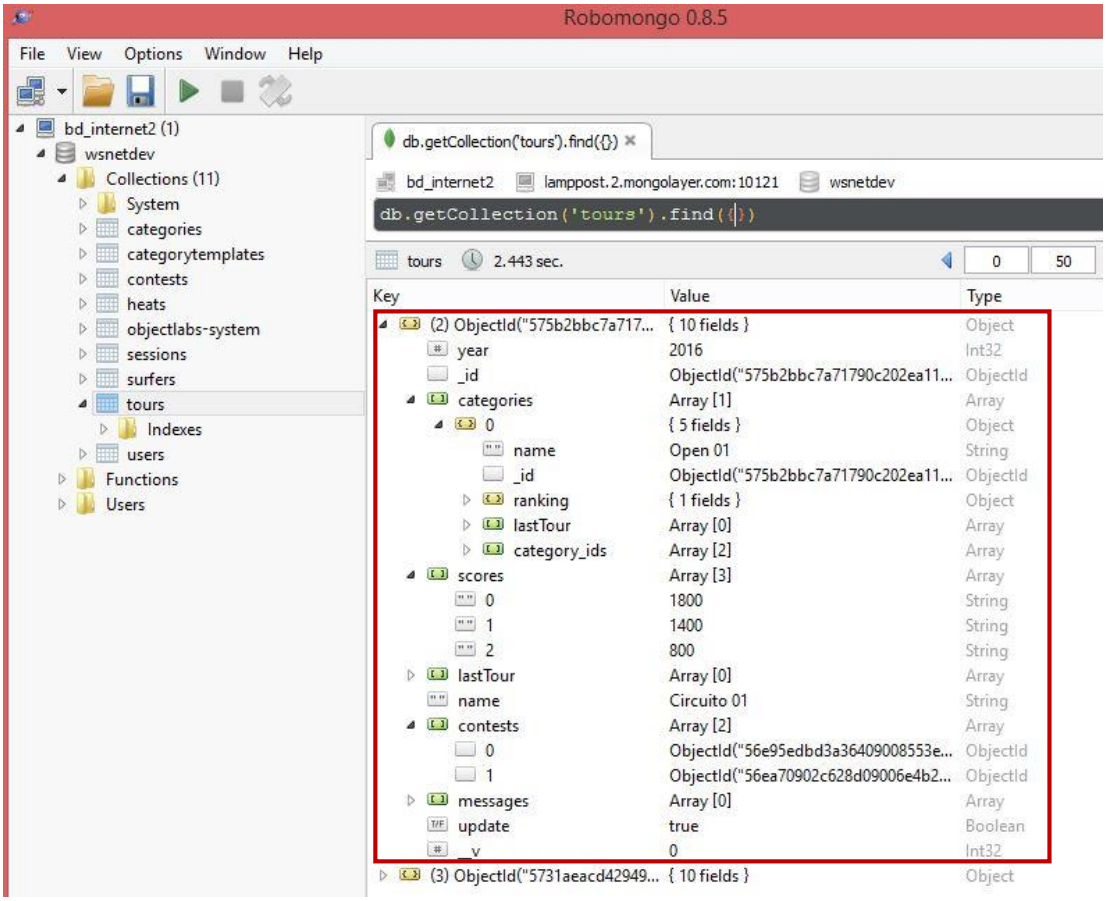


Imagen 17: Visualización en Robomongo del Tour creado correctamente.

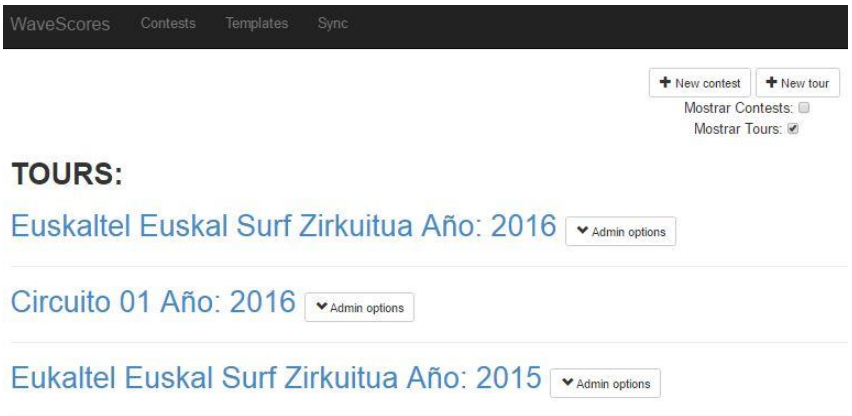


Imagen 18: Lista con todos los Tours y con el recién creado llamado Circuito 01.

Estas dos últimas imágenes muestran que se ha realizado correctamente la creación de un Circuito.

Añadir Categories en un Tour.

Para añadir una Categories en un Tour se deben rellenar los datos del formulario correctamente, en caso contrario, aparecerán mensajes de error y no se creará el Tour.

Se han realizado varias validaciones:

- El campo *Tour Name* está deshabilitado y no se puede modificar.
- El campo *Year* está deshabilitado y no se puede modificar.
- El campo *Categories Name* es un campo *input* de tipo texto, permite escribir tanto caracteres como números, pero se no pueden añadir los dos caracteres “<”, “>”.
- El campo Categories es una lista desplegable que muestra todas las categorías del campeonato seleccionado al crear el Tour.

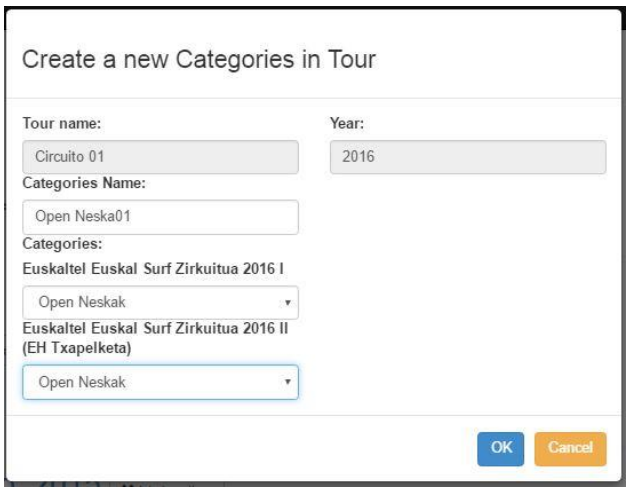


Imagen 19: Formulario para crear una Categories en un Tour con los campos llenos.

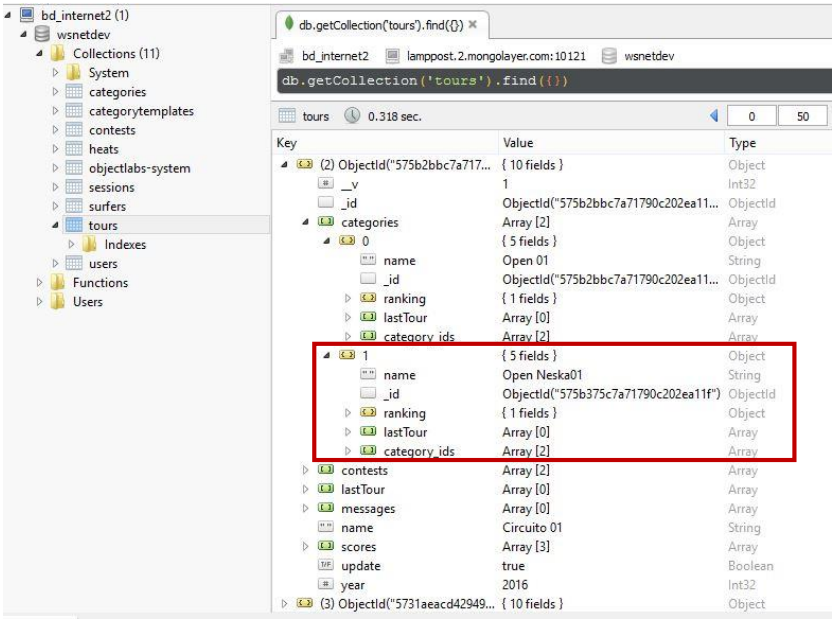


Imagen 20: Visualización en Robomongo de un Tour de la nueva Categories añadida correctamente.

La imagen muestra que se ha añadido la nueva Categories en la base de datos correctamente.

9.2. Pruebas de la Funcionalidad Generar *Ranking*

Se realizaron diversas pruebas para probar si la funcionalidad se realizaba correctamente en distintos casos.

Requisitos para el Cálculo del *Ranking*:

1. Es necesario que haya surfistas en alguna categoría.
2. Es necesario que haya por lo menos una categoría finalizada.

Una categoría finalizada es una donde el campeonato ya ha concluido, en cambio una categoría sin finalizar es una donde el campeonato aún no ha iniciado.

Caso 01 - Generar *Ranking* sin surfistas en las categorías:

- **Datos de entrada:**
  La entrada será un Circuito con sus correspondientes datos. En las Categorías añadidas no habrá surfistas.

CIRCUITO		
Nombre:	Circuito Prueba01.	
Año:	2015	
Campeonatos:	Campeonato01 2015, Campeonato02 2015, Campeonato03 2015.	
Categories*:	**Categories name:**	Sub18
	Categorías:	Sub18_C01, Sub18_C02, Sub18_C03.
Puntos:	2000, 1800, 1600, 1400, 800, 200.	

* Categories es un *array* que un su interior contiene varios Categories formados por varios campos como *categories name* o categorías, entre otros. Para estas pruebas se usará solamente un categories.

Sub18_C01, Sub18_C02 y Sub18_C03: Sin surfistas añadidos.

- Datos de Salida:

  La salida será una tabla sin surfistas ya que no hay ninguno añadido en las categorías. Como se expuso previamente, para el cálculo del *ranking* se necesita cumplir una serie de requisitos. En este caso, no se cumplen ni el primero ni el segundo. Al no haber surfistas en alguna categoría no se puede realizar el cálculo del *ranking*.

Campeonato01 2015 Sub18_C01	Campeonato02 2015 Sub18_C02	Campeonato03 2015 Sub18_C03	*Ranking* Total

- Fallos descubiertos:

  Ningún fallo relevante. La prueba realizada fue satisfactoria.

Caso 02 - Generar *Ranking* con surfistas en categorías no finalizadas:

- Datos de entrada:

  La entrada será un Circuito con sus correspondientes datos. En las Categorías habrá surfistas pero éstas no se habrán desarrollado aún, esto se traduce en que los surfistas no tendrán una posición final asociada ya que ésta se asigna al finalizar las Categorías. Cuando no se ha finalizado una Categoría, por defecto su posición final asignada es -1.
  El Circuito a usar será el mismo que se usó en el Caso 01.

Posición / Nombre_Surfista

CATEGORIAS		
Sub18_C01	Sub18_C02	Sub18_C03
-1 / Andrés Zamora	-1 / Miracleman	-1 / Batman
-1 / Sandman	-1 / Deadpool	-1 / Joe Kelly
-1 / Alan Moore	-1 / Andrés Zamora	-1 / Sandman
-1 / Neil Gaiman	-1 / Superman	-1 / Andrés Zamora

- Datos de Salida:

  La salida será una tabla con los surfistas de las categorías. Por ser categorías no finalizadas se les asigna 0 puntos a los surfistas.

  Siguiendo los requisitos para el cálculo del *ranking*, se cumple el primero pero no el segundo. Al no haber al menos una categoría finalizada no se puede realizar el cálculo del *ranking*.

Posición / Nombre_Surfista / Puntos

Campeonato01 2015 Sub18_C01	Campeonato02 2015 Sub18_C02	Campeonato03 2015 Sub18_C03	*Ranking* Total
-1 / Andrés Zamora 0	-1 / Miracleman 0	-1 / Batman 0	
-1 / Sandman 0	-1 / Deadpool 0	-1 / Joe Kelly 0	
-1 / Alan Moore 0	-1 / Andrés Zamora 0	-1 / Sandman 0	
-1 / Neil Gaiman 0	-1 / Superman 0	-1 / Andrés Zamora 0	

- Fallos descubiertos:

  Ningún fallo relevante. La prueba realizada fue satisfactoria.

Caso 03 - Generar *Ranking* con surfistas en las categorías finalizadas:

- Datos de entrada:

  La entrada será un Circuito con sus correspondientes datos. En las Categorías habrá surfistas y éstas estarán finalizadas.

  El Circuito a usar será el mismo que se usó en el Caso 01.

  Prueba 01 - Entrada:

  En esta prueba las categorías tienen la misma cantidad de surfistas.

CATEGORIAS		
Sub18_C01	Sub18_C02	Sub18_C03
0 / Andrés Zamora	0 / Miracleman	0 / Batman
1 / Sandman	1 / Deadpool	1 / Joe Kelly
2 / Alan Moore	2 / Andrés Zamora	2 / Sandman
3 / Neil Gaiman	3 / Superman	3 / Andrés Zamora

4 / Frank Miller	4 / Joe Kelly	4 / Miracleman
5 / Batman	5 / Alan Moore	5 / Frank Miller
6 / Deadpool	6 / Batman	6 / Spiderman
7 / Spiderman	7 / Stan Lee	7 / Alan Moore

Prueba 02 - Entrada:

En esta prueba las categorías tienen distintas cantidades de surfistas: Sub18_C01 son ocho surfistas, Sub18_C02 son cinco surfistas, Sub18_C03 son seis surfistas.

CATEGORIAS		
Sub18_C01	**Sub18_C02**	**Sub18_C03**
0 / Andrés Zamora	0 / Miracleman	0 / Batman
1 / Sandman	1 / Deadpool	1 / Joe Kelly
2 / Alan Moore	2 / Andrés Zamora	2 / Sandman
3 / Neil Gaiman	3 / Superman	3 / Andrés Zamora
4 / Frank Miller	4 / Joe Kelly	4 / Miracleman
5 / Batman		5 / Frank Miller
6 / Deadpool		
7 / Spiderman		

Prueba 03 - Entrada:

En esta prueba las categorías tienen distintas cantidades de surfistas: Sub18_C01 son cuatro surfistas, Sub18_C02 son seis surfistas, Sub18_C03 son tres surfistas.

CATEGORIAS		
Sub18_C01	**Sub18_C02**	**Sub18_C03**
0 / Andrés Zamora	0 / Miracleman	0 / Batman
1 / Sandman	1 / Deadpool	1 / Joe Kelly
2 / Alan Moore	2 / Andrés Zamora	2 / Sandman
3 / Neil Gaiman	3 / Superman	
	4 / Joe Kelly	
	5 / Alan Moore	

Prueba 04 - Entrada:

En esta prueba las categorías tienen distintas cantidades de surfistas: Sub18_C01 son tres surfistas, Sub18_C02 son cinco surfistas, Sub18_C03 son ocho surfistas.

CATEGORIAS		
Sub18_C01	Sub18_C02	Sub18_C03
0 / Andrés Zamora	0 / Miracleman	0 / Batman
1 / Sandman	1 / Deadpool	1 / Joe Kelly
2 / Alan Moore	2 / Andrés Zamora	2 / Sandman
	3 / Superman	3 / Andrés Zamora
	4 / Joe Kelly	4 / Miracleman
		5 / Frank Miller
		6 / Spiderman
		7 / Alan Moore

- Datos de Salida:

  La salida será una tabla con los surfistas de las categorías. Por ser categorías finalizadas, a los surfistas se les asignan los puntos del Circuito dependiendo de su posición final en la categoría.

  Siguiendo los requisitos para el cálculo del *ranking*, se cumplen todos en todas las pruebas. Así, se puede realizar el cálculo del *ranking*.

  Para las pruebas del cálculo de *ranking*s comprobamos que se han realizado las sumas de los puntos que un surfista ha obtenido en cada una de las categorías en las que ha participado. Finalmente, la última columna se ordena por medio de los puntos sumados de cada surfista.

Prueba 01 - Salida:

Posición / Nombre_Surfista / Puntos

Campeonato01 2015 Sub18_C01	Campeonato02 2015 Sub18_C02	Campeonato03 2015 Sub18_C03	*Ranking* Total
0 / Andrés Zamora **2000**	0 / Miracleman **2000**	0 / Batman **2000**	1 / Andrés Zamora **5000**
1 / Sandman **1800**	1 / Deadpool **1800**	1 / Joe Kelly **1800**	2 / Sandman **3400**

2 / Alan Moore 1600	2 / Andrés Zamora 1600	2 / Sandman 1600	3 / Batman 3000
3 / Neil Gaiman 1400	3 / Superman 1400	3 / Andrés Zamora 1400	4 / Miracleman 2800
4 / Frank Miller 800	4 / Joe Kelly 800	4 / Miracleman 800	5 / Alan Moore 2600
4 / Batman 800	4 / Alan Moore 800	4 / Frank Miller 800	6 / Joe Kelly 2600
6 / Deadpool 200	6 / Batman 200	6 / Spiderman 200	7 / Deadpool 2000
6 / Spiderman 200	6 / Stan Lee 200	6 / Alan Moore 200	8 / Frank Miller 1600
			9 / Neil Gaiman 1400
			10 / Superman 1400
			11 / Spiderman 400
			12 / Stan Lee 200

Prueba 02 - Salida:

Campeonato01 2015 Sub18_C01	Campeonato02 2015 Sub18_C02	Campeonato03 2015 Sub18_C03	*Ranking* Total
0 / Andrés Zamora 2000	0 / Miracleman 2000	0 / Batman 2000	1 / Andrés Zamora 5000
1 / Sandman 1800	1 / Deadpool 1800	1 / Joe Kelly 1800	2 / Sandman 3400
2 / Alan Moore 1600	2 / Andrés Zamora 1600	2 / Sandman 1600	3 / Batman 2800
3 / Neil Gaiman 1400	3 / Superman 1400	3 / Andrés Zamora 1400	4 / Miracleman 2800
4 / Frank Miller 800	4 / Joe Kelly 800	4 / Miracleman 800	5 / Joe Kelly 2600
4 / Batman 800		5 / Frank Miller 200	6 / Deadpool 2000
6 / Deadpool 200			7 / Alan Moore 1600
6 / Spiderman 200			8 / Neil Gaiman 1400
			9 / Superman 1400
			10 / Frank Miller 1000
			11 / Spiderman 200

Prueba 03 - Salida:

Campeonato01 2015 Sub18_C01	Campeonato02 2015 Sub18_C02	Campeonato03 2015 Sub18_C03	*Ranking* Total
0 / Andrés Zamora 2000	0 / Miracleman 2000	0 / Batman 2000	1 / Andrés Zamora 3600
1 / Sandman 1800	1 / Deadpool 1800	1 / Joe Kelly 1800	2 / Sandman 3400
2 / Alan Moore 1600	2 / Andrés Zamora 1600	2 / Sandman 1600	3 / Joe Kelly 2600
3 / Neil Gaiman 1400	3 / Superman 1400		4 / Miracleman 2000
	4 / Joe Kelly 800		5 / Batman 2000
	5 / Alan Moore 200		6 / Alan Moore 1800
			7 / Deadpool 1800
			8 / Neil Gaiman 1400
			9 / Superman 1400

Prueba 04 - Salida:

Campeonato01 2015 Sub18_C01	Campeonato02 2015 Sub18_C02	Campeonato03 2015 Sub18_C03	*Ranking* Total
0 / Andrés Zamora 2000	0 / Miracleman 2000	0 / Batman 2000	1 / Andrés Zamora 5000
1 / Sandman 1800	1 / Deadpool 1800	1 / Joe Kelly 1800	2 / Sandman 3400
2 / Alan Moore 1600	2 / Andrés Zamora 1600	2 / Sandman 1600	3 / Miracleman 2800
	3 / Superman 1400	3 / Andrés Zamora 1400	4 / Joe Kelly 2600
	4 / Joe Kelly 800	4 / Miracleman 800	5 / Batman 2000
		4 / Frank Miller 800	6 / Alan Moore 1800
		6 / Spiderman 200	7 / Deadpool 1800
		6 / Alan Moore 200	8 / Superman 1400
			9 / Frank Miller 800
			10 / Spiderman 200

- Fallos descubiertos:

  Gracias a las pruebas se descubrieron fallos. La prueba 01 se realizaba de forma correcta pero las pruebas 02, 03 y 04 no. El error se debía a que al recorrer la lista de surfistas de cada categoría siempre se tomaba como longitud para el recorrido la de la primera lista de la primera categoría y se usaba la misma para el recorrido de todas las categorías.
  Esto no era ningún problema para categorías que tuvieran la misma longitud (cantidad de surfistas) como en la prueba 01, pero el resto de pruebas con categorías de distintas longitudes no permitían el cálculo correcto del *ranking* y no pasaban las pruebas.
  Gracias a las pruebas, se localizó el problema y se resolvió. Se volvieron a realizar las 4 pruebas del Caso03 y los resultados coincidían con las salidas esperadas, las pruebas fueron satisfactorias.

Caso 04 - Generar *Ranking* con surfistas en algunas categorías no finalizadas y otras finalizadas:

- Datos de entrada:

  La entrada será un Circuito con sus correspondientes datos. En las Categorías habrá surfistas. Algunas de éstas estarán finalizadas y otras no.
  El Circuito a usar será el mismo que se usó en el Caso 01.

Prueba 01 - Entrada:

Las Categorías Sub18_C01 y Sub18_C02 están finalizadas, pero la Sub18_C03 no.

CATEGORIAS		
Sub18_C01	Sub18_C02	Sub18_C03
0 / Andrés Zamora	0 / Miracleman	-1 / Batman
1 / Sandman	1 / Alan Moore	-1 / Joe Kelly
2 / Alan Moore	2 / Andrés Zamora	-1 / Sandman
3 / Neil Gaiman	3 / Superman	-1 / Andrés Zamora

Prueba 02 - Entrada:

Las Categorías Sub18_C02 y Sub18_C03 están finalizadas, pero la Sub18_C01 no.

CATEGORIAS		
Sub18_C01	Sub18_C02	Sub18_C03
-1 / Andrés Zamora	0 / Miracleman	0 / Batman
-1 / Sandman	1 / Alan Moore	1 / Joe Kelly

-1 / Alan Moore	2 / Andrés Zamora	2 / Sandman
-1 / Neil Gaiman	3 / Superman	3 / Andrés Zamora

Prueba 03 - Entrada:

Las Categorías Sub18_C01 y Sub18_C03 están finalizadas, pero la Sub18_C02 no.

CATEGORIAS		
Sub18_C01	**Sub18_C02**	**Sub18_C03**
0 / Andrés Zamora	-1 / Miracleman	0 / Batman
1 / Sandman	-1 / Alan Moore	1 / Joe Kelly
2 / Alan Moore	-1 / Andrés Zamora	2 / Sandman
3 / Neil Gaiman	-1 / Superman	3 / Andrés Zamora

- Datos de Salida:

  La salida será una tabla con los surfistas de las categorías.

  En las categorías finalizadas se les asignan a los surfistas los puntos del Circuito dependiendo de su posición final en la categoría. En las categorías no finalizadas se les asigna a los surfistas 0 puntos.

  Siguiendo los requisitos para el cálculo del *ranking*, se cumplen todos en todas las pruebas, se puede realizar el cálculo del *ranking*.

Prueba 01 - Salida:

Las Categorías Sub18_C01 y Sub18_C02 están finalizadas, pero la Sub18_C03 no.

Campeonato01 2015 Sub18_C01	Campeonato02 2015 Sub18_C02	Campeonato03 2015 Sub18_C03	*Ranking* Total
0 / Andrés Zamora **2000**	0 / Miracleman **2000**	-1 / Batman **0**	1 / Andrés Zamora **3600**
1 / Sandman **1800**	1 / Alan Moore **1800**	-1 / Joe Kelly **0**	2 / Alan Moore **3400**
2 / Alan Moore **1600**	2 / Andrés Zamora **1600**	-1 / Sandman **0**	3 / Miracleman **2000**
3 / Neil Gaiman **1400**	3 / Superman **1400**	-1 / Andrés Zamora **0**	4 / Sandman **1800**
			5 / Neil Gaiman **1400**
			6 / Superman **1400**

Prueba 02 - Salida:

Las Categorías Sub18_C02 y Sub18_C03 están finalizadas, pero la Sub18_C01 no.

Campeonato01 2015 Sub18_C01	Campeonato02 2015 Sub18_C02	Campeonato03 2015 Sub18_C03	*Ranking* Total
-1 / Andrés Zamora 0	0 / Miracleman 2000	0 / Batman 2000	1 / Andrés Zamora 3000
-1 / Sandman 0	1 / Alan Moore 1800	1 / Joe Kelly 1800	2 / Miracleman 2000
-1 / Alan Moore 0	2 / Andrés Zamora 1600	2 / Sandman 1600	3 / Batman 2000
-1 / Neil Gaiman 0	3 / Superman 1400	3 / Andrés Zamora 1400	4 / Alan Moore 1800
			5 / Joe Kelly 1800
			6 / Sandman 1600
			7 / Superman 1400

Prueba 03 - Salida:

Las Categorías Sub18_C01 y Sub18_C03 están finalizadas, pero la Sub18_C02 no.

Campeonato01 2015 Sub18_C01	Campeonato02 2015 Sub18_C02	Campeonato03 2015 Sub18_C03	*Ranking* Total
0 / Andrés Zamora 2000	-1 / Miracleman 0	0 / Batman 2000	1 / Andrés Zamora 3400
1 / Sandman 1800	-1 / Alan Moore 0	1 / Joe Kelly 1800	2 / Sandman 3400
2 / Alan Moore 1600	-1 / Andrés Zamora 0	2 / Sandman 1600	3 / Batman 2000
3 / Neil Gaiman 1400	-1 / Superman 0	3 / Andrés Zamora 1400	4 / Joe Kelly 1800
			5 / Alan Moore 1600
			6 / Neil Gaiman 1400

- Fallos descubiertos:

  Gracias a las pruebas se descubrió un fallo. La prueba 01 se realizaba de forma correcta pero las pruebas 02 y 03 no. El error se debía a que en la implementación el código solo

estaba pensado para que se realizara de forma ordenada, es decir que siempre debían estar finalizadas las primeras categorías y las no finalizadas deberían seguirles después.
El orden correcto sería, por ejemplo, tener la categoría Sub18_C01 finalizada y las siguientes no finalizadas, o tener las Sub18_C01 y Sub18_C02 finalizadas y la Sub18_C03 no finalizada. En esos casos, las pruebas hubieran sido superadas como en la prueba 01.
Pero si se intercalaban categorías no finalizadas o se empezaba por una no finalizada, las pruebas no serían superadas como ocurrió en las pruebas 02 y 03.
Gracias a las pruebas se localizó el problema y se resolvió. Se volvieron a realizar las 3 pruebas del Caso04 y los resultados coincidían con las salidas esperadas, las pruebas fueron satisfactorias.

Caso 05 - Generar *Ranking* sin surfistas en unas Categorías y con surfistas en otras:

- **Datos de entrada:**

  La entrada será un Circuito con sus correspondientes datos. En algunas Categorías habrá surfistas y en otras no. Todas las categorías con surfistas están finalizadas.
  El Circuito a usar será el mismo que se usó en el Caso 01.

Prueba 01 - Entrada:

CATEGORIAS		
Sub18_C01	**Sub18_C02**	**Sub18_C03**
0 / Andrés Zamora		
1 / Sandman		
2 / Alan Moore		

Prueba 02 - Entrada:

CATEGORIAS		
Sub18_C01	**Sub18_C02**	**Sub18_C03**
	0 / Miracleman	
	1 / Alan Moore	
	2 / Andrés Zamora	

Prueba 03 - Entrada:

CATEGORIAS		
Sub18_C01	Sub18_C02	Sub18_C03
		0 / Batman
		1 / Joe Kelly
		2 / Sandman

Prueba 04 - Entrada:

CATEGORIAS		
Sub18_C01	Sub18_C02	Sub18_C03
0 / Andrés Zamora		0 / Batman
1 / Sandman		1 / Joe Kelly
2 / Alan Moore		2 / Sandman

Prueba 05 - Entrada:

CATEGORIAS		
Sub18_C01	Sub18_C02	Sub18_C03
0 / Andrés Zamora	0 / Miracleman	
1 / Sandman	1 / Alan Moore	
2 / Alan Moore	2 / Andrés Zamora	

Prueba 06 - Entrada:

CATEGORIAS		
Sub18_C01	Sub18_C02	Sub18_C03
	0 / Miracleman	0 / Batman
	1 / Alan Moore	1 / Joe Kelly
	2 / Andrés Zamora	2 / Sandman

- Datos de Salida:

  La salida será una tabla mostrando los surfistas dentro de su Categoría correspondiente. Las columnas de las Categorías que no tengan surfistas quedarán vacías.

  Siguiendo los requisitos para el cálculo del *ranking*, se cumplen todos en todas las pruebas, se puede realizar el cálculo del *ranking*.

Prueba 01 - Salida:

Campeonato01 2015 Sub18_C01	Campeonato02 2015 Sub18_C02	Campeonato03 2015 Sub18_C03	*Ranking* Total
0 / Andrés Zamora 2000			1 / Andrés Zamora 2000
1 / Sandman 1800			2 / Sandman 1800
2 / Alan Moore 1600			3 / Alan Moore 1600

Prueba 02 - Salida:

Campeonato01 2015 Sub18_C01	Campeonato02 2015 Sub18_C02	Campeonato03 2015 Sub18_C03	*Ranking* Total
	0 / Miracleman 2000		1 / Miracleman 2000
	1 / Alan Moore 1800		2 / Alan Moore 1800
	2 / Andrés Zamora 1600		3 / Andrés Zamora 1600

Prueba 03 - Salida:

Campeonato01 2015 Sub18_C01	Campeonato02 2015 Sub18_C02	Campeonato03 2015 Sub18_C03	*Ranking* Total
		0 / Batman 2000	1 / Batman 2000
		1 / Joe Kelly 1800	2 / Joe Kelly 1800
		2 / Sandman 1600	3 / Sandman 1600

Prueba 04 - Salida:

Campeonato01 2015 Sub18_C01	Campeonato02 2015 Sub18_C02	Campeonato03 2015 Sub18_C03	*Ranking* Total
0 / Andrés Zamora 2000		0 / Batman 2000	1 / Sandman 3400
1 / Sandman 1800		1 / Joe Kelly 1800	2 / Andrés Zamora 2000

2 / Alan Moore **1600**		2 / Sandman **1600**	3 / Batman **2000**
			4 / Joe Kelly **1800**
			5 / Alan Moore **1600**

Prueba 05 - Salida:

Campeonato01 2015 Sub18_C01	Campeonato02 2015 Sub18_C02	Campeonato03 2015 Sub18_C03	*Ranking* Total
0 / Andrés Zamora **2000**	0 / Miracleman **2000**		1 / Andrés Zamora **3600**
1 / Sandman **1800**	1 / Alan Moore **1800**		2 / Alan Moore **3400**
2 / Alan Moore **1600**	2 / Andrés Zamora **1600**		3 / Miracleman **2000**
			4 / Sandman **1800**

Prueba 06 - Salida:

Campeonato01 2015 Sub18_C01	Campeonato02 2015 Sub18_C02	Campeonato03 2015 Sub18_C03	*Ranking* Total
	0 / Miracleman **2000**	0 / Batman **2000**	1 / Miracleman **2000**
	1 / Alan Moore **1800**	1 / Joe Kelly **1800**	2 / Batman **2000**
	2 / Andrés Zamora **1600**	2 / Sandman **1600**	3 / Alan Moore **1800**
			4 / Joe Kelly **1800**
			5 / Andrés Zamora **1600**
			6 / Sandman **1600**

- Fallos descubiertos:

  Ningún fallo relevante. Las pruebas realizadas fueron satisfactorias.

Caso 06 - Generar *Ranking* mezclando casos anteriores:

- Datos de entrada:

  La entrada será un Circuito con sus correspondientes datos. En algunas Categorías habrá surfistas y en otras no. Algunas categorías con surfistas están finalizadas, otras no. La cantidad de surfistas en cada categoría será diferente.

  El Circuito a usar será el mismo que se usó en el Caso 01.

  Prueba 01 - Entrada:

  - La categoría Sub18_C01 tiene surfistas (4) y no está finalizada.
  - La categoría Sub18_C02 no tiene surfistas.
  - La categoría Sub18_C03 tiene surfistas (3) y no está finalizada.

CATEGORIAS		
Sub18_C01	Sub18_C02	Sub18_C03
-1 / Andrés Zamora		-1 / Batman
-1 / Sandman		-1 / Joe Kelly
-1 / Alan Moore		-1 / Sandman
-1 / Neil Gaiman		

  Prueba 02 - Entrada:

  - La categoría Sub18_C01 tiene surfistas (3) y está finalizada.
  - La categoría Sub18_C02 no tiene surfistas.
  - La categoría Sub18_C03 tiene surfistas (4) pero no está finalizada.

CATEGORIAS		
Sub18_C01	Sub18_C02	Sub18_C03
0 / Andrés Zamora		-1 / Batman
1 / Sandman		-1 / Joe Kelly
2 / Alan Moore		-1 / Sandman
		-1 / Andrés Zamora

  Prueba 03 - Entrada:

  - La categoría Sub18_C01 tiene surfistas (5) y no está finalizada.

- La categoría Sub18_C02 tiene surfistas (4) y está finalizada.
- La categoría Sub18_C03 no tiene surfistas.

CATEGORIAS		
Sub18_C01	Sub18_C02	Sub18_C03
-1 / Andrés Zamora	0 / Miracleman	
-1 / Sandman	1 / Deadpool	
-1 / Alan Moore	2 / Andrés Zamora	
-1 / Neil Gaiman	3 / Superman	
-1 / Frank Miller		

Prueba 04 - Entrada:

- La categoría Sub18_C01 tiene surfistas (4) y no está finalizada.
- La categoría Sub18_C02 tiene surfistas (2) y está finalizada.
- La categoría Sub18_C03 tiene surfistas (3) y está finalizada.

CATEGORIAS		
Sub18_C01	Sub18_C02	Sub18_C03
-1 / Andrés Zamora	0 / Miracleman	0 / Batman
-1 / Sandman	1 / Joe Kelly	1 / Joe Kelly
-1 / Alan Moore		2 / Sandman
-1 / Neil Gaiman		

- Datos de Salida:

  La salida será una tabla mostrando los surfistas dentro de su Categoría correspondiente. Las columnas de las Categorías que no tengan surfistas quedarán vacías.

  Siguiendo los requisitos para el cálculo del *ranking*, en la prueba 01 se cumple el primero pero no el segundo. Al no haber al menos una categoría finalizada no se puede realizar el cálculo del *ranking*. Por otro lado, en el resto de pruebas sí se cumplen todos los requisitos, se puede realizar el cálculo del *ranking* en esas pruebas.

Prueba 01 - Salida:

Campeonato01 2015 Sub18_C01	Campeonato02 2015 Sub18_C02	Campeonato03 2015 Sub18_C03	*Ranking* Total
-1 / Andrés Zamora 0		-1 / Batman 0	
-1 / Sandman 0		-1 / Joe Kelly 0	
-1 / Alan Moore 0		-1 / Sandman 0	
-1 / Neil Gaiman 0			

Prueba 02 - Salida:

Campeonato01 2015 Sub18_C01	Campeonato02 2015 Sub18_C02	Campeonato03 2015 Sub18_C03	*Ranking* Total
0 / Andrés Zamora 2000		-1 / Batman 0	1 / Andrés Zamora 2000
1 / Sandman 1800		-1 / Joe Kelly 0	2 / Sandman 1800
2 / Alan Moore 1600		-1 / Sandman 0	3 / Alan Moore 1600
		-1 / Andrés Zamora 0	

Prueba 03 - Salida:

Campeonato01 2015 Sub18_C01	Campeonato02 2015 Sub18_C02	Campeonato03 2015 Sub18_C03	*Ranking* Total
-1 / Andrés Zamora 0	0 / Miracleman 2000		1 / Miracleman 2000
-1 / Sandman 0	1 / Deadpool 1800		2 / Deadpool 1800
-1 / Alan Moore 0	2 / Andrés Zamora 1600		3 / Andrés Zamora 1600
-1 / Neil Gaiman 0	3 / Superman 1400		4 / Superman 1400
-1 / Frank Miller 0			

Prueba 04 - Salida:

Campeonato01 2015 Sub18_C01	Campeonato02 2015 Sub18_C02	Campeonato03 2015 Sub18_C03	*Ranking* Total
-1 / Andrés Zamora **0**	0 / Miracleman **2000**	0 / Batman **2000**	1 / Joe Kelly **3600**
-1 / Sandman **0**	1 / Joe Kelly **1800**	1 / Joe Kelly **1800**	2 / Miracleman **2000**
-1 / Alan Moore **0**		2 / Sandman **1600**	3 / Batman **2000**
-1 / Neil Gaiman **0**			4 / Sandman **1600**

- Fallos descubiertos:

  Ningún fallo relevante. Las pruebas realizadas fueron satisfactorias.

9.3. Pruebas de la Funcionalidad Generar *Seeding*

Se realizaron diversas pruebas para probar si la funcionalidad se realizaba correctamente en distintos casos. La funcionalidad Generar *Seeding* es muy parecida a la de Generar *Ranking*, parte del código usado para el *Ranking* se usa en la de *Seeding*, al ser pruebas muy parecidas no me extenderé tanto como en el apartado anterior y solo expondré algunos casos.

Requisitos para el Cálculo del *Seeding*:

1. Es necesario el *Ranking*Total del año pasado.
2. Es necesario que haya surfistas en alguna categoría.
3. Es necesario que haya por lo menos una categoría sin finalizar.

El *seeding* es un dato importante, sirve para poder determinar el orden y la distribución de los surfistas en las mangas de la primera ronda de una categoría. Es un dato que hay que obtener antes del inicio de un campeonato.

Por esta razón, para calcular el *seeding,* es necesario elegir una categoría sin finalizar y en esa categoría a los surfistas se les asigna el *seeding* correspondiente.

Caso 01 - Generar *Seeding* sin surfistas en las categorías:

- Datos de entrada:

  La entrada será un Circuito con sus correspondientes datos. En las Categorías añadidas no habrá surfistas.

CIRCUITO		
Nombre:	Circuito Prueba02.	
Año:	2016	
Campeonatos:	Campeonato01 2016, Campeonato02 2016, Campeonato03 2016.	
Categories:	Categories name:	Sub18
	Categorías:	Sub18_C01, Sub18_C02, Sub18_C03.
	*Ranking*Total Año Pasado:	*Ranking*Total_Sub18_2015
Puntos:	2000, 1800, 1600, 1400, 800, 200.	

*Ranking*Total_Sub18_2015
1 / Batman **2000**
2 / Minsky **1500**
3 / Turing **1400**
4 / Alan Moore **1000**
5 / Sandman **800**

Los puntos de la lista del *Ranking* Total del año pasado no tienen que concordar necesariamente con los puntos asignados al nuevo Circuito Prueba02. Los puntos de un Circuito pueden variar de año en año, no es algo común pero puede ocurrir.

Sub18_C01, Sub18_C02 y Sub18_C03: Sin surfistas añadidos.

- Datos de Salida:

  La salida será una tabla sin surfistas en las categorías, pero la primera columna mostrará a los surfistas del *ranking* total del año pasado.

  Como se expuso previamente, para el cálculo del *seeding* se necesita cumplir una serie de requisitos. En este caso, cumplimos el primero, pero el segundo no. Al no haber surfistas en alguna categoría no se puede realizar el cálculo del *seeding*.

*Ranking*Total 2015	Campeonato01 2016 / Sub18_C01	Campeonato02 2016 / Sub18_C02	Campeonato03 2016 / Sub18_C03	*Seeding* Total
1 / Batman 2000				
2 / Minsky 1500				
3 / Turing 1400				
4 / Alan Moore 1000				
5 / Sandman 800				

- Fallos descubiertos:

  Ningún fallo relevante. La prueba realizada fue satisfactoria.

Caso 02 - Generar *Seeding* con surfistas en categorías finalizadas:

- Datos de entrada:

  La entrada será un Circuito con sus correspondientes datos. Las categorías tendrán surfistas y estarán finalizadas.

  El Circuito a usar será el mismo que se usó en el Caso 01.

CATEGORIAS		
Sub18_C01	Sub18_C02	Sub18_C03
0 / Andrés Zamora	0 / Miracleman	0 / Batman
1 / Sandman	1 / Deadpool	1 / Joe Kelly
2 / Alan Moore	2 / Andrés Zamora	2 / Sandman
3 / Neil Gaiman	3 / Superman	3 / Andrés Zamora

- Datos de Salida:

  La salida será una tabla con los surfistas de las categorías que tengan surfistas. La primera columna mostrará a los surfistas del *ranking* total del año pasado.

  Siguiendo los requisitos para el cálculo del *seeding*, se cumplen el primer y segundo requisito, pero no el tercero. Al no haber al menos una categoría no finalizada, no se puede realizar el cálculo del *seeding*.

*Ranking*Total 2015	Campeonato01 2016 / Sub18_C01	Campeonato02 2016 / Sub18_C02	Campeonato03 2016 / Sub18_C03	*Seeding* Total
1 / Batman 2000	0 / Andrés Zamora 2000	0 / Miracleman 2000	0 / Batman 2000	
2 / Minsky 1500	1 / Sandman 1800	1 / Deadpool 1800	1 / Joe Kelly 1800	
3 / Turing 1400	2 / Alan Moore 1600	2 / Andrés Zamora 1600	2 / Sandman 1600	
4 / Alan Moore 1000	3 / Neil Gaiman 1400	3 / Superman 1400	3 / Andrés Zamora 1400	
5 / Sandman 800				

- Fallos descubiertos:

  Ningún fallo relevante. La prueba realizada fue satisfactoria.

Caso 03 - Generar *Seeding* con surfistas en categorías no finalizadas:

- Datos de entrada:

  La entrada será un Circuito con sus correspondientes datos. Las categorías tendrán surfistas y no estarán finalizadas.

  El Circuito a usar será el mismo que se usó en el Caso 01.

  Las categorías no finalizadas tienen un *seeding* por defecto que es el orden en que los surfistas fueron añadidos en las categorías. Dicho valor cambia al calcular el *seeding* y guardar los cambios en la base de datos.

  Prueba 01 - Entrada:

  Se puede elegir entre todas las categorías ya que son no finalizadas.

  En esta prueba, la categoría que se elegirá para asignarle el *seeding* será Sub18_C01.

Posición / Nombre_Surfista / Puntos / *Seeding*

CATEGORIAS		
Sub18_C01	Sub18_C02	Sub18_C03
-1 / Andrés Zamora 0 *Seeding*: 0	-1 / Miracleman	-1 / Batman
-1 / Sandman 0 *Seeding*: 1	-1 / Deadpool	-1 / Joe Kelly
-1 / Alan Moore 0 *Seeding*: 2	-1 / Andrés Zamora	-1 / Sandman
-1 / Neil Gaiman 0 *Seeding*: 3	-1 / Superman	-1 / Andrés Zamora

Prueba 02 - Entrada:

Solo se puede elegir entre Sub18_C02 y Sub18_C03 al ser categorías no finalizadas.

En esta prueba la categoría que se elegirá para asignarle el *seeding* será Sub18_C03.

CATEGORIAS		
Sub18_C01	Sub18_C02	Sub18_C03
	-1 / Miracleman	-1 / Batman 0 *Seeding*: 0
	-1 / Deadpool	-1 / Joe Kelly 0 *Seeding*: 1
	-1 / Andrés Zamora	-1 / Sandman 0 *Seeding*: 2
	-1 / Superman	-1 / Andrés Zamora 0 *Seeding*: 3

- Datos de Salida:

  La salida será una tabla con los surfistas de las categorías que tengan surfistas. La primera columna mostrará los surfistas del *ranking* total del año pasado.

  Siguiendo los requisitos para el cálculo del *seeding*, en todas las pruebas se cumplen todos los requisitos, se puede realizar el cálculo del *seeding*.

  El valor del *seeding* se consigue de la suma de puntos de un surfista en todas las categorías más el *ranking* total del año pasado. El *Seeding* total es una lista con esos

puntos, sólo se muestran en esa lista los surfistas de la categoría seleccionada previamente. Y el nuevo *seeding* que se les asigna a los surfistas de la categoría seleccionada se determina por la posición final en la lista del *Seeding* total.

Prueba 01 - Salida:

Posición / Nombre_Surfista / Puntos / *Seeding*

*Ranking*Total 2015	Campeonato01 2016 / Sub18_C01	Campeonato02 2016 / Sub18_C02	Campeonato03 2016 / Sub18_C03	*Seeding* Total
1 / Batman 2000	-1 / Andrés Zamora 0 *Seeding*: 2	-1 / Miracleman 0	-1 / Batman 0	1 /Alan Moore 1000
2 / Minsky 1500	-1 / Sandman 0 *Seeding*: 1	-1 / Deadpool 0	-1 / Joe Kelly 0	2 / Sandman 800
3 / Turing 1400	-1 / Alan Moore 0 *Seeding*: 0	-1 / Andrés Zamora 0	-1 / Sandman 0	3 / Andrés Zamora 0
4 / Alan Moore 1000	-1 / Neil Gaiman 0 *Seeding*: 3	-1 / Superman 0	-1 / Andrés Zamora 0	4 / Neil Gaiman 0
5 / Sandman 800				

Prueba 02 - Salida:

*Ranking*Total 2015	Campeonato01 2016 / Sub18_C01	Campeonato02 2016 / Sub18_C02	Campeonato03 2016 / Sub18_C03	*Seeding* Total
1 / Batman 2000		-1 / Miracleman 0	-1 / Batman 0 *Seeding*: 0	1 / Batman 2000
2 / Minsky 1500		-1 / Deadpool 0	-1 / Joe Kelly 0 *Seeding*: 2	2 / Sandman 800
3 / Turing 1400		-1 / Andrés Zamora 0	-1 / Sandman 0 *Seeding*: 1	3 / Joe Kelly 0

4 / Alan Moore **1000**		-1 / Superman **0**	-1 / Andrés Zamora **0** *Seeding*: 3	4 / Andrés Zamora **0**
5 / Sandman **800**				

- **Fallos descubiertos:**

  Ningún fallo relevante. Las pruebas realizadas fueron satisfactorias.

Caso 04 - Generar *Seeding* con surfistas mezclando categorías finalizadas y no finalizadas:

- **Datos de entrada:**

  La entrada será un Circuito con sus correspondientes datos. Las Categorías podrán tener surfistas o no y algunas de éstas estarán finalizadas, otras no.

  La cantidad de surfistas de las categorías son distintas para probar que también funciona en esos casos.

  El Circuito a usar será el mismo que se usó en el Caso 01.

  Prueba 01 - Entrada:

  Solo se puede elegir la categoría Sub18_C03 al ser la única no finalizada.

  En esta prueba la categoría que se elegirá para asignarle el *seeding* será Sub18_C03.

CATEGORIAS		
Sub18_C01	**Sub18_C02**	**Sub18_C03**
0 / Andrés Zamora	0 / Miracleman	-1 / Batman **0** *Seeding*: 0
1 / Sandman	1 / Alan Moore	-1 / Neil Gaiman **0** *Seeding*: 1
	2 / Andrés Zamora	-1 / Sandman **0** *Seeding*: 2
		-1 / Andrés Zamora **0** *Seeding*: 3

Prueba 02 - Entrada:

Sólo se puede elegir la categoría Sub18_C02 al ser la única no finalizada.

En esta prueba la categoría que se elegirá para asignarle el *seeding* será Sub18_C02.

CATEGORIAS		
Sub18_C01	**Sub18_C02**	**Sub18_C03**
	-1 / Miracleman **0** *Seeding*: 0	0 / Batman
	-1 / Alan Moore **0** *Seeding*: 1	1 / Joe Kelly
	-1 / Andrés Zamora **0** *Seeding*: 2	2 / Sandman
		3 / Andrés Zamora

- **Datos de Salida:**

  La salida será una tabla con los surfistas de las categorías que tengan surfistas. La primera columna mostrará a los surfistas del *ranking* total del año pasado.

  Siguiendo los requisitos para el cálculo del *seeding*, en todas las pruebas se cumplen todos los requisitos, se puede realizar el cálculo del *seeding*.

Prueba 01 - Salida:

***Ranking*Total 2015**	**Campeonato01 2016 / Sub18_C01**	**Campeonato02 2016 / Sub18_C02**	**Campeonato03 2016 / Sub18_C03**	***Seeding* Total**
1 / Batman **2000**	0 / Andrés Zamora **2000**	0 / Miracleman **2000**	-1 / Batman **0** *Seeding*: 2	1 / Andrés Zamora **3600**
2 / Minsky **1500**	1 / Sandman **1800**	1 / Alan Moore **1800**	-1 / Neil Gaiman **0** *Seeding*: 3	2 / Sandman **2600**
3 / Turing **1400**		2 / Andrés Zamora **1600**	-1 / Sandman **0** *Seeding*: 1	3 / Batman **2000**
4 / Alan Moore **1000**			-1 / Andrés Zamora **0** *Seeding*: 0	4 / Neil Gaiman **0**

5 / Sandman **800**				

Prueba 02 - Salida:

*Ranking*Total 2015	Campeonato01 2016 / Sub18_C01	Campeonato02 2016 / Sub18_C02	Campeonato03 2016 / Sub18_C03	*Seeding* Total
1 / Batman **2000**		-1 / Miracleman **0** *Seeding*: 2	0 / Batman **2000**	1 / Andrés Zamora **1400**
2 / Minsky **1500**		-1 / Alan Moore **0** *Seeding*: 1	1 / Joe Kelly **1800**	2 / Alan Moore **1000**
3 / Turing **1400**		-1 / Andrés Zamora **0** *Seeding*: 0	2 / Sandman **1600**	3 / Miracleman **1000**
4 / Alan Moore **1000**			3 / Andrés Zamora **1400**	
5 / Sandman **800**				

- Fallos descubiertos:

  Ningún fallo relevante. Las pruebas realizadas fueron satisfactorias.

10

Gestión

En este capítulo trataré la gestión que he tenido del proyecto a lo largo del mismo.

10.1. Planificación

10.1.1. Requisitos

Al inicio del proyecto mantuve una reunión con el DPE para que me expusiera su idea de proyecto, lo que deseaba que hiciera, y ver si me interesaba. Me comentó distintas propuestas de proyectos para realizar y me interesé por la del cálculo de *ranking*s. Me explicó brevemente que su aplicación lleva la gestión de campeonatos de surf, el registro de usuarios, la creación de Campeonatos, sus correspondientes Categorías, las Mangas, las Rondas, las puntuaciones que le asignan los jueces a los surfistas en el transcurso de la Categoría de un Campeonato, cómo es el paso a las siguientes rondas hasta llegar a la final, etc. Sin entrar en detalles, me comentó la funcionalidad que quería que desarrollara: hacer el cálculo de *ranking*s de los surfistas en un

Circuito, junto con las tecnologías que utilizaba, su entorno de trabajo y las herramientas que usaba.

Acepté el proyecto y quedamos para la instalación de su aplicación web en mi ordenador y las herramientas necesarias para desarrollarla. Me propuso reunirme con el resto de socios de la empresa WaveScores para charlar sobre la empresa, su intención con el proyecto y profundizar un poco más sobre las funcionalidades a desarrollar. Me reuní con ellos y me explicaron cómo se conocieron, cómo empezó todo como un proyecto fin de carrera hasta llegar a ser una aplicación web en explotación en algunos campeonatos de surf de Guipúzcoa. Los dos socios aparte del DPE son jueces de campeonatos de surf de España y también de campeonatos en la WSL[5], por lo cual conocen el mundo del surf y están actualizados en lo que se busca en una aplicación con las características del DPE y le mantienen informado de los últimos cambios en el reglamento para estar siempre a la última. Me explicaron las funcionalidades que deseaban que desarrollara, a grandes rasgos querían calcular el *ranking* de los surfistas en los Circuitos, pero hablando más con ellos pude identificar tres casos: Creación de Circuitos, Cálculo de *Ranking* y Cálculo de *Seeding*.

10.1.2. Viabilidad

La viabilidad del proyecto la basé en lo aprendido durante las dos funcionalidades realizadas previamente al desarrollo del proyecto, se habló más sobre estas funcionalidades en el capítulo 4 llamado Adaptación.

Estas dos funcionalidades me sirvieron para poder tener una idea más clara de si podría realizar las tres funcionalidades requeridas en el apartado anterior en el tiempo disponible y concluí que sí era viable.

10.1.3. Dependencia entre funcionalidades

Las funcionalidades tienen dependencias claras. Una de ellas es que hay que implementar la creación de circuitos, sin un Circuito las funcionalidades restantes no pueden funcionar. La otra dependencia clara es entre las otras dos funcionalidades. No se puede desarrollar la funcionalidad Calcular *Seeding* sin haber implementado antes la de Cálculo de *Ranking*, esto es porque para poder calcular el *seeding* es necesario el *ranking* total del año pasado, sin haber hecho antes el Cálculo de *Ranking* no se puede conseguir hacer la otra funcionalidad.

[5] World Surf League. http://www.worldsurfleague.com

10.1.4. Periodo de desarrollo de las funcionalidades

Las fechas previstas para el desarrollo de las tres funcionalidades eran:

- Funcionalidad Crear Circuitos.
  - **Inicio:** 07/03/16
  - **Fin:** 25/03/16
- Funcionalidad Calcular *Ranking* de un Circuito.
  - **Inicio:** 27/03/16
  - **Fin:** 11/04/16
- Funcionalidad Calcular *Seeding* de un Circuito.
  - **Inicio:** 12/04/16
  - **Fin:** 24/04/16

10.1.5. Gestión de Riesgos

Había algunos riesgos al iniciar el proyecto:

- El aprendizaje de nuevas tecnologías. Durante la carrera aprendí varias, pero éstas eran completamente nuevas. El riesgo era no poder entender las tecnologías o que costara demasiado y, por ello, no poder cumplir los plazos para desarrollar todas las funcionalidades requeridas.

- Leer el código del DPE y entender lo realizado por él previamente. Algunos informáticos tenemos la costumbre de escribir el código a nuestra manera, sin comentarios, ya que no pensamos que otro programador vaya a tocar el código, esto sucede más en proyectos personales como era el del DPE, ya que se inició como un proyecto fin de carrera y no se pensaba que alguna vez alguien más fuera a revisar su código. Es un hándicap a tener en cuenta, pero es algo con lo que a los informáticos nos toca lidiar alguna vez en nuestra profesión. Muchas veces no haremos las aplicaciones desde cero, sino que nos tocará añadir funcionalidades a aplicaciones ya desarrolladas, entenderlas y adaptarnos a ellas. Es un riesgo, pero a la vez lo veo como una oportunidad de aprendizaje.

10.1.6. Calidad

Calidad Mínima:

- Completada la funcionalidad Crear Circuitos.
- Completada la funcionalidad Calcular *Ranking* de un Circuito.
- Completada la funcionalidad Calcular *Seeding* de un Circuito.

Calidad Añadida:

- La funcionalidad de mostrar la tabla de *ranking*s del Categories de un Circuito, se tomará dicha tabla de la base de datos.

10.2. Seguimiento y Control

10.2.1. Tareas realizadas en el proyecto

- Respecto al entregable E01.

TAREAS	DESCRIPCIÓN	ESTADO
T01.1	Definir el modelo tour.js (Circuito).	Finalizado
T01.2	Crear un elemento modal para mostrar el formulario para Crear Circuitos.	Finalizado
T01.3	Guardar el Circuito en la base de datos.	Finalizado
T01.4	Crear un elemento modal para mostrar el formulario para Añadir Categories de un Circuito.	Finalizado
T01.5	Modificar el Circuito con el Categories añadido en la base de datos.	Finalizado

- Respecto al entregable E02.

TAREAS	DESCRIPCIÓN	ESTADO
T02.1	Crear una tabla con las Categorías de Categories de un Circuito y sus surfistas correspondientes.	Finalizado
T02.2	Calcular el *Ranking*.	Finalizado
T02.3	Guardar la tabla *Ranking* en la base de datos.	Finalizado

- Respecto al entregable E03.

TAREAS	DESCRIPCIÓN	ESTADO
T03.1	Elegir el modo en que se quiere añadir el *Ranking* Total del año pasado.	Finalizado
T03.2	Crear una tabla con las Categorías de Categories de un Circuito y con surfistas correspondientes.	Finalizado
T03.3	Guardar el *Ranking* Total del año pasado en la base de datos.	Finalizado
T03.4	Calcular el *Seeding*.	Finalizado
T03.5	Modificar el *Seeding* de una Categoría seleccionada en la base de datos.	Finalizado

10.2.2. Duración y fechas finales del proyecto

- Comparación de Fechas Estimadas y Dedicadas:

	Fechas Estimadas	Fechas Dedicadas	Desviaciones
Crear Circuitos	**Inicio:** 07/03/16 **Fin:** 25/03/16 **Duración:** 19 días.	**Inicio:** 15/03/16 **Fin:** 28/03/16 **Duración:** 14 días.	**Inicio:** 8 días después. **Fin:** 3 días después. Duración: 5 días menos.
Calcular *Ranking* de un Circuito	**Inicio:** 26/03/16 **Fin:** 11/04/16 **Duración:** 17 días.	**Inicio:** 29/03/16 **Fin:** 15/04/16 **Duración:** 18 días.	**Inicio:** 3 días después. **Fin:** 4 días después. Duración: 1 día más.
Calcular *Seeding* de un Circuito	**Inicio:** 12/04/16 **Fin:** 24/04/16 **Duración:** 13 días.	**Inicio:** 20/04/16 **Fin:** 29/04/16 **Duración:** 10 días.	**Inicio:** 8 días después. **Fin:** 5 días después. Duración: 3 días menos.

- Horas dedicadas al proyecto:

Categoría	Descripción	Horas
Aprendizaje	Funcionalidad de artículo favorito. Funcionalidad arreglada de cálculo de *ranking*. Tecnologías del *stack* MEAN.	15 horas 48 horas 36 horas
Gestión del Proyecto	Reunión con los socios. Reuniones con el director del proyecto.	3 horas 5 horas

	Reuniones con el DPE.	15 horas
Diseño	Modelo.	2 horas
Implementación	Modelo de Datos. *Backend*. *Frontend*. Pruebas	2 horas 122 horas 68 horas 12 horas
Corrección de errores		31 horas
Memoria	Redacción. Corrección.	51 horas 13 horas
TOTAL		423 horas

10.2.3. Desviaciones significativas del proyecto

Hubo varias desviaciones a lo largo del proyecto por diversos motivos.

Al iniciar el proyecto con la primera funcionalidad, la desviación fue de ocho días. Esto se debió a que arreglar el fallo en la aplicación del DPE que me pidió como segunda funcionalidad para practicar me tomó más tiempo del previsto, ya que era la primera vez que miraba su código y tenía que aprender su modelo de datos, cómo están distribuidas las vistas, rutas y lo que hacían ciertas variables, ya que no estaba el código comentado y eso hacía más complicada la tarea.

Otra desviación importante fue cuando se inició la tercera funcionalidad de cálculo de *Seeding*. Se puede apreciar que hubo un periodo de 5 días entre la finalización de la segunda funcionalidad y el inicio de la tercera, esto se debió a que después de finalizar la segunda funcionalidad, el DPE me pidió que hiciera un pequeño manual de usuario para explicarle el funcionamiento de las dos primeras funcionalidades a Gorka, uno de los socios, para que diera su visto bueno y mirara si había algo que cambiar o mejorar. Esta fue la principal razón de haber iniciado más tarde la tercera funcionalidad.

A pesar de las desviaciones de inicio y finalización más tardías, se puede contemplar que la duración en días fue menor en la primera y tercera funcionalidad, sólo en la segunda se demoró un día más su finalización. La segunda funcionalidad fue complicada, ya que tuve que resolver muchos problemas de implementación y esto originó la demora. Pero en la tercera funcionalidad usaba parte del código utilizado en la segunda y esto hizo que la tercera tuviera una duración menor, además del hecho de que ya tenía más dominadas las tecnologías y me costaba menos realizar ciertas tareas.

Las desviaciones fueron significativas con respecto a la planificación inicial, aunque le entregué el proyecto al DPE con la antelación necesaria para que lo incorporara en el suyo y lo pudiera utilizar en los campeonatos venideros.

10.3. Gestión de Interesados

Los interesados los he dividido en dos partes: la universidad y la empresa.

10.3.1. Universidad

En la universidad, el primer interesado fue mi director de proyecto José Miguel Blanco. Desde que estuve en su asignatura de Sistemas de Gestión de Seguridad de Sistemas de Información le expresé mi intención de que fuera mi director. También le expuse mis dudas iniciales sobre qué tema hacer el proyecto y me aconsejó que me enfocara primero en aprobar todas las asignaturas que me faltaban, y a principios de enero hablaríamos más sobre el proyecto. Al llegar enero y haber aprobado todas las asignaturas, me reuní con él y le comenté mi intención de hacer el proyecto en una empresa para ganar experiencia profesional. Me dijo que un exalumno de la universidad, el DPE, había formado una empresa y estaba interesado en aceptar de prácticas a un estudiante. También, me expuso las tecnologías que utilizaba. Me interesó el tema de una empresa emprendedora y las tecnologías usadas, así que accedí a reunirme con el DPE para hablar de sus ideas de proyecto en su empresa. La universidad también tomó parte para que pudiera matricularme en el proyecto fin de carrera, inscribir el proyecto y, finalmente, firmar el convenio entre la universidad, la empresa y yo.

10.3.2. Empresa

La empresa para la que realizaría el proyecto se llama Wavescores Development S.L y está formada por tres socios. Uno de ellos es el DPE, el exalumno de la universidad del que me habló José Miguel. Es el Ingeniero Informático a cargo del desarrollo de la aplicación web.

Después de haberme reunido con él y aceptar su propuesta de proyecto, nos reunimos con el resto de socios para conocernos, exponerme la situación actual de la empresa, cómo habían llegado a donde están y decirme las funcionalidades que deseaban que realizara para su aplicación web. Los otros dos socios se llaman Gorka y Mikel, son hermanos y trabajan como jueces en campeonatos de surf en España y a nivel mundial. Su labor es promocionar la aplicación en los distintos campeonatos de surf donde no se utiliza, como alternativa a tener en cuenta, ya

que otras aplicaciones para gestión de campeonatos de surf son mucho más caras y en algunos casos menos completas que la aplicación de su empresa. Al ser jueces de surf, también están a la última en lo que buscan los jueces y los cambios en los reglamentos entre otros temas. Este conocimiento es importante ya que permite la mejora de la aplicación y estar actualizados siempre.

11

Conclusiones

En este capítulo expondré las conclusiones a las que he llegado después de la realización del proyecto.

11.1. Aprendizaje de nuevas tecnologías

Al iniciar este proyecto, una de las cosas que más me llamaron la atención fueron las tecnologías involucradas para el desarrollo del mismo. Durante la carrera he aprendido una variedad de tecnologías que son útiles tanto para desarrollar aplicaciones web como aplicaciones software tradicionales, pero no había tratado con el *stack* MEAN, ésta era nueva para mí. Me interesó el detalle de desarrollar tanto el lado cliente como el lado servidor usando Javascript.

Puedo concluir que he aprendido mucho de las tecnologías utilizadas durante el proyecto y que me han aportado conocimientos que me serán de utilidad en mi futuro profesional.

11.2. Solucionar un problema real

Las funcionalidades implementadas durante el desarrollo del proyecto se creaban con el fin de solucionar un problema real existente hoy en día en los campeonatos de surf, el cual es el cálculo de *ranking*s y *seeding*s.

El gasto excesivo de tiempo que toma hacer dichos cálculos es un problema, el mundo del surf no estaba hace unos años tan informatizado como está en la actualidad y, aun así, hay problemas que pueden ser resueltos por medio de la informática de forma más sencilla y óptima que como se está resolviendo actualmente. Hoy en día, para calcular el *ranking* y *seeding* de los surfistas en los circuitos se hace todo a mano, lo cual lleva un tiempo considerable si hablamos de que una persona suma todas las puntuaciones, pudiendo llegar a ser en ocasiones hasta más de 100 surfistas diferentes entre dos o más campeonatos. Por esta razón se desarrollaron las funcionalidades en la aplicación web, ya que serían de gran ayuda a las organizaciones de surf que quisieran ahorrar tiempo y esfuerzo para obtener estos cálculos de forma inmediata y que de otra forma llevaría días obtener.

Puedo concluir que el haber desarrollado funcionalidades para solucionar un problema real me mostró la importancia que puede tener la informática en ciertos entornos donde no se usa actualmente y que su inclusión podría aportar mejoras considerables.

11.3. La importancia de tener un software testeado

Un apartado muy importante en el proyecto fueron las pruebas. Se crearon distintos tipos de pruebas para diferentes casos. Las funcionalidades desarrolladas debían ejecutarse correctamente al pasar a fase de explotación, de ahí la importancia de realizar muchas pruebas mientras se estuviera en fase de producción. En fase de producción se encontraron diversos errores que gracias a las pruebas fueron localizados y solucionados.

Puedo concluir que he aprendido la importancia de realizar pruebas en aplicaciones que están en explotación, como era el caso de ésta. Un software sin testear tiene muchas probabilidades de fallo y es algo que no se puede permitir en aplicaciones que están en funcionamiento, ya que los fallos acarrearían realizar pruebas extras, resolver los fallos dedicando más tiempo del previsto y dando mala imagen a la aplicación.

11.4. Obtener experiencia en una empresa

Mi interés inicial en hacer mi proyecto fin de carrera en una empresa me llevó a desarrollarlo en la empresa Wavescores Development S.L. Me motivó la idea de trabajar con un exalumno de la universidad y seguir ayudando con el proyecto fin de carrera que él inició hace unos años y que está en funcionamiento en la actualidad en un entorno real, como son los campeonatos de surf.

Puedo concluir que la experiencia obtenida ha sido grata y significativa. He comprendido la importancia de las reuniones con los clientes y la importancia de realizar una captura de requisitos adecuada para el correcto desarrollo de las funcionalidades pedidas. Pero la mayor aportación por parte de la empresa ha sido presenciar cómo una aplicación web que empezó hace dos años como un PFC ha crecido tanto en este tiempo hasta llegar a ser lo que es hoy en día. Estar en esta empresa emprendedora me ha servido de motivación para comprobar que los informáticos podemos llegar a hacer grandes cosas con esfuerzo y dedicación.

12

Bibliografía

[1] Adrián Izquierdo Blanco, Memoria Proyecto Fin de Carrera, "Juzgando las olas, sistema de puntuaciones para campeonatos de surf". Junio 2014.

[2] Santiago Ceria, Casos de Uso Un Método Práctico para Explorar Requerimientos.
http://www-2.dc.uba.ar/materias/isoft1/2001_2/apuntes/CasosDeUso.pdf

[3] Junta de Andalucía, Guía para la redacción de casos de uso.
http://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/416

[4] Demián Gutiérrez, Casos de uso Diagramas de Casos de Uso.
http://www.codecompiling.net/files/slides/UML_clase_02_UML_casos_de_uso.pdf

[5] Demián Gutiérrez, UML Diagrama de Secuencia.
http://www.codecompiling.net/files/slides/UML_clase_06_UML_secuencia.pdf

[6] DesarrolloWeb.com, Manual de AngularJS.
http://www.desarrolloweb.com/manuales/manual-angularjs.html

[7] W3schools.com, *AngularJS Tutorial.*
http://www.w3schools.com/angular/

[8] egghead.io, videotutoriales de Angular.js.

https://egghead.io/technologies/angularjs?order=desc&page=6

[9] kendar.org, *AngularJS Tutorial - 3.*

http://www.kendar.org/?p=/tutorials/angularjs/part03

[10] stackoverflow.com, *nested arrays with ng-repeat and table html.*

http://stackoverflow.com/questions/29321117/nested-arrays-with-ng-repeat-and-table-html

[11] jsfiddle.net, ejemplo de código que rellena una tabla html.

https://jsfiddle.net/6nh100ca/11/

[12] MongoDB, Reinventando la gestión de datos.

https://www.mongodb.com/es

[13] Mongoose, página oficial.

http://mongoosejs.com/

[14] Modulus, *Node.Js And Mongodb - Getting Started With Mongoose.*

http://blog.modulus.io/getting-started-with-mongoose

[15] Genbetadev, MongoDB: qué es, cómo funciona y cuándo podemos usarlo (o no).

http://www.genbetadev.com/bases-de-datos/mongodb-que-es-como-funciona-y-cuando-podemos-usarlo-o-no

[16] Genbetadev, Una introducción a MongoDB.

http://www.genbetadev.com/bases-de-datos/una-introduccion-a-mongodb

[17] Blog de NetConsulting, Node.js: ¿Qué es y para qué sirve NodeJS?

http://www.netconsulting.es/blog/nodejs/

[18] Express, Direccionamiento.

http://expressjs.com/es/guide/routing.html

[19] Roger Dudler, git - la guía sencilla.

http://rogerdudler.github.io/git-guide/index.es.html

[20] Bootstrap, página oficial.

http://getbootstrap.com/

[21] Librosweb, Bootstrap 3 el manual oficial.

http://librosweb.es/libro/bootstrap_3/

[22] W3schools.com, *Bootstrap 3 Tutorial.*

http://www.w3schools.com/bootstrap/