UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA
Escuela de Ingeniería Informática

App móvil Android de seguimiento GPS para carreras de orientación. Especialidad Rogainine

Gastón Martín Mora Madrid

Mayo 2018

Tutor: Modesto Castrillón Santana

Tutor: Javier Lorenzo Navarro

Agradecimientos

Finalizar este proyecto nunca hubiese sido posible sin antes haberlo empezado. Puede parecer algo trivial esta afirmación pero atrás han quedado muchos años, no quiero ni contarlos, desde que entré por primera vez por las puertas de la Facultad de Informática de la Universidad de Las Palmas de Gran Canaria. Un proyecto sobre un modelo numérico de aguas someras del Estrecho de Magallanes finalizado de intercambio en Punta Arenas, Chile en 2007 y un intento fallido posterior de adaptar este último a las aguas que rodean el archipiélago canario han sido las precuelas de este, El Definitivo... espero. Es mucho el tiempo que llevo cargando con este lastre a la espalda (el de no presentar el proyecto) y es ahora, cuando estoy escribiendo estas líneas, cuando empiezo a sentir el alivio de quitarme ese peso de encima.

Finalmente, pude encontrar el proyecto que me diese la motivación necesaria para seguir en casa unas horas más con el ordenador tras las jornadas de trabajo y el apoyo de unos tutores que compartían conmigo la afición a este deporte. Era un proyecto motivante, útil y divertido, pero aún así no ha sido hasta el último momento cuando he sacado fuerzas para dedicarme en cuerpo y alma a terminarlo definitivamente.

El apoyo de mi gente, su insistencia y comprensión año tras año para que lo terminase de una vez han sido determinantes para llegar a buen puerto y por mucho que escriba nunca será suficiente para agradecerles su paciencia y confianza.

Muchas gracias a todos, ¡por fin lo hemos conseguido!

Gastón Mora Madrid, Las Palmas de Gran Canaria. Junio 2018

Resumen

El presente proyecto se centra en proporcionar una solución para facilitar el seguimiento de los deportistas en las carreras de orientación de la modalidad Rogaining. Se busca crear una herramienta autogestionada, y asequible a todos los organizadores que permita el seguimiento en tiempo real y el análisis posterior de las rutas escogidas por los equipos. Se propone una herramienta basada en tres componentes: un portal Web, un *backend* a modo de servicio Web y una aplicación para dispositivos móviles.

En el primer capítulo se introduce de forma breve y concisa el deporte de la orientación, sus particularidades en general y las características específicas de la modalidad de Rogaining.

En el segundo capítulo, se realizará un repaso sobre el estado del arte actual en lo que al seguimiento de carreras de orientación y análisis de los recorridos de los participantes se refiere.

En los capítulos 3 y 4 se especifican las tecnologías y herramientas utilizadas, la metodología de trabajo y la planificación del presente proyecto.

A continuación, en el capítulo 5, se entra en profundidad en el análisis y diseño de la solución planteada, poniendo especial interés en los retos o problemas a resolver que se han considerado más relevantes. Se especifica también la estructura de la base de datos que dará soporte a los tres componentes.

Los capítulos 6,7 y 8 se introducen en los detalles significativos de la implementación de cada una de las partes del sistema: *backend*, *frontend* y la aplicación para *smartphones* respectivamente.

En el siguiente capítulo se analiza la batería de pruebas realizada y los resultados obtenidos, sobre la gestión completa de un evento, desde la creación del mismo hasta

el seguimiento mediante las posiciones reportadas por los dispositivos móviles.

En el capítulo 10 se exponen las conclusiones alcanzadas y las ideas de trabajo futuro propuestas por el autor a partir de las funcionalidades implementadas en este trabajo.

Adicionalmente en el anexo A, se dan las instrucciones necesarias para la puesta en marcha de la solución.

Índice general

Índice de figuras

Capítulo 1

Introducción

Con el fin de poder tener una primera toma de contacto, en este primer capítulo se tratará de introducir de forma breve y concisa el deporte de la orientación y sus particularidades.

1.1. La orientación deportiva

La orientación se trata de una modalidad deportiva, originaria de Escandinavia y cuya primera competición fue organizada en octubre de 1897 [1], bajo el amparo de la Federación Internacional de Orientación (IOF por sus siglas en inglés), cuyo fin, en su forma original, es visitar una serie de localizaciones geográficas de forma ordenada y en el menor tiempo posible con la ayuda de un mapa (elaborado con una simbología específica para este deporte) y una brújula [Figura 1.1].

Figura 1.1: Mapa de orientación y brújula

Figura 1.2: Baliza, pinza manual y estación de control electrónica (Sportident)

1.1.1. Reglas básicas

Los corredores y corredoras compiten de forma individual contra el crono tratando de localizar los puntos marcados sobre el mapa trasladándose a pie. El recorrido a realizar entre los distintos puntos de control es libre, sólo siendo obligatorio encontrarlos todos y dejando de mano del propio competidor la elección del que considere el trayecto más rápido para ir de un control a otro.

En las coordenadas del terreno correspondientes al centro exacto de cada uno de estos puntos, se ubican los controles. Estos están representados con una baliza de tela, en forma de prisma triangular y de color naranja y blanco a las que acompaña un sistema que permite a los deportistas justificar su paso por el mismo y que puede ser manual (pinza) o electrónico (Sportident o Emit[1]) [Figura 1.2]. La no localización o justificación de paso por cada una de las balizas en el orden establecido, supone la descalificación del competidor.

Los participantes se dividen en distintas categorías definidas en función del sexo, edad y capacidad técnica y física de los mismos. Cada categoría tiene un recorrido cuya longitud y dificultad se trata de adaptar a estos parámetros.

Una misma baliza puede pertenecer al recorrido de más de una categoría y en un orden distinto dentro de cada uno de ellas. Por ello, se etiqueta cada baliza con un código de control que es único para una misma carrera y que permite identificarla de forma unívoca.

A tal fin, a los corredores se les incluye junto con el mapa una *tabla de descripción de controles* en el que se le indica esta correspondencia de códigos de control para

[1]Si desea conocer más sobre el funcionamiento de estos sistemas puede consultar sus respectivas páginas web: Sportident https://www.sportident.com/ o Emit http://www.emit.no/en

MF14/16			4,7 km			150 m	
▷							
1	57						
2	53						
3	48						
4	54						
5	55						
6	52						
7	44						
8	33						
9	31						
10	41						
11	39						
12	42						
13	32						
14	36						
15	41						
16	35						
17	38						
18	40						
○			90 m				◎

Figura 1.3: Código de control (baliza 16) y descripcion de controles.

cada baliza del recorrido y además, mediante una serie de símbolos estandarizados [4], se les informa del elemento exacto donde se encuentra la misma [Figura 1.3].

Está totalmente prohibido el uso de dispositivos de ayuda a la navegación como smartphones, GPS, altímetros y cuenta pasos.

1.1.2. Variantes

Como alternativa a las carreras tradicionales de orientación, nacen otras modalidades asociadas a las mismas que comparten su filosofia (encontrar una serie de controles por medio de la utilización de un mapa y una brújula), pero en las que cambia ligeramente las normas o la forma de participación. Entre ellas encontramos:

- **Relevos**: Variante de orientación tradicional por equipos de 3 participantes (habitualmente) en el que cada uno realiza un recorrido distinto de forma secuencial, debiendo terminar cada miembro del equipo el suyo para que el siguiente relevo pueda comenzar [Figura 1.4].

- **Score**: En esta modalidad individual, el orden en el que se visitan los controles es libre, cada control tiene asignado una puntuación y no es obligatorio localizarlos todos. A cada categoría se le otorga un tiempo máximo de carrera, en el que debe tratar de sumar el mayor número de puntos posibles. Es una

Figura 1.4: Orientación en Relevos

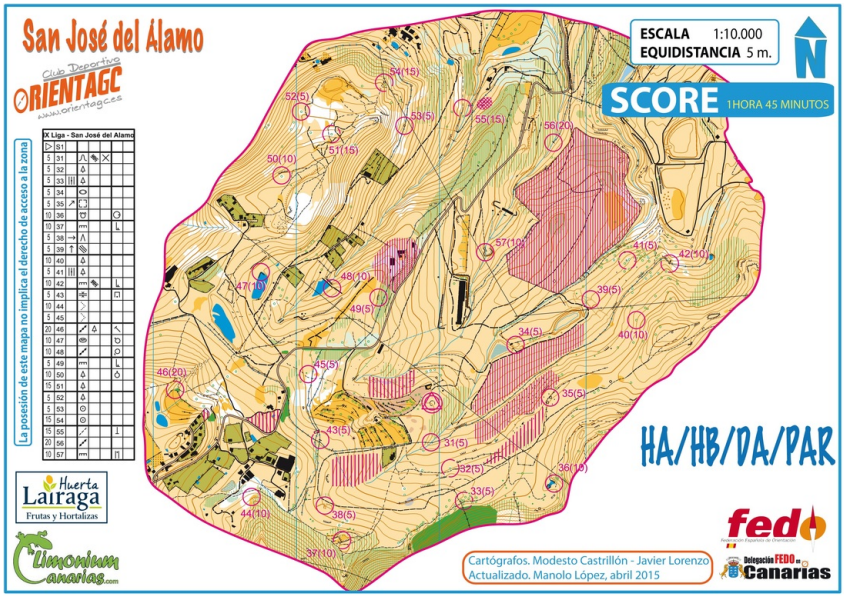


Figura 1.5: Orientación Score

modalidad donde además de las habilidades en orientación se trabaja también la estrategia y planificación de la carrera [Figura 1.5].

- **Micro-O**: Modalidad individual adaptada a la práctica de la orientación en espacios cerrados y reducidos, como una cancha deportiva, en la se utilizan todo tipo de elementos para confeccionar el recorrido (vallas, conos, taburetes, bancos,...) y en la que se busca trabajar la rápida toma de decisiones y lectura del mapa [Figura 1.6].

- **Parejas o grupos**: Si bien no se trata de una modalidad como tal, es una variante de la orientación muy utilizada para la iniciación en este deporte y que permite realizar los recorridos en pareja o grupos de más de dos participantes.

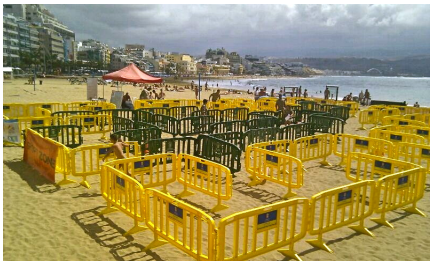


Figura 1.6: Micro orientación

1.1.3. Rogaining

El Rogaining se trata de una modalidad de orientación de larga distancia, en la que compiten equipos de entre 2 y 5 participantes, y en la que la dinámica es similar a las carreras tipo *Score*; siendo su fin tratar de conseguir el mayor número de puntos posibles visitando los controles elegidos en un tiempo limitado. Es una modalidad nueva, originaria de Australia y cuya primera carrera data de 1947 [2] y que cuenta con su propia federación, la Federación Internacional de Rogaining (IRF por sus siglas en inglés).

Equipos y categorías En una misma prueba pueden competir equipos de distinto número de participantes, edad y sexo, pudiendo ser mixtos. Los equipos se dividen en categorías en función del sexo y edad de sus componentes.

Duración Si bien la duración estipulada por la IRF para los campeonatos oficiales de Rogaining es de 24 horas, se permite la organización de pruebas de menor duración [3]. Tras haber seguido en los últimos años el calendario de pruebas organizadas en España, las más habituales suelen ser las de 3,6 y 12 horas de duración.

Además del tiempo máximo establecido, se fija un umbral de llegada tardía a meta por encima del tiempo máximo fijado en el que el equipo va perdiendo puntos (penalizando) a medida que pasa el tiempo. Una vez superado este umbral, el equipo queda descalificado, perdiendo todos los puntos que haya podido conseguir.

Avituallamientos y material obligatorio Si bien se trata de un deporte de autosuficiencia en el que cada equipo es responsable de llevar el agua y alimentos

que cree que va a necesitar durante la carrera, en todas las pruebas de Rogaining, es obligatorio que la organización cuente con al menos un campamento base (conocido como Hash House) donde los corredores puedan descansar, comer y reponer fuerzas.

Del mismo modo, aunque no obligatorio, es común que además del campamento base, se ubiquen distintos puntos de avituallamiento en el mapa donde el corredor pueda repostar al menos líquido.

Sistema de cronometraje y contabilización de los controles En este tipo de pruebas es imprescindible el uso de un sistema de cronometraje electrónico tipo Sportident o Emit, ya que para que un control sea contabilizado, todos los miembros del equipo deben justificar su paso por él en una franja de tiempo inferior a 3 minutos. A tal fin, los chips de cronometraje deben ir sujetos al corredor de forma que no pueda quitárselo sin dejar constancia de ello, para evitar que sea un sólo miembro del equipo el que lleve los chips del resto.

El mapa y el terreno En Rogaining, si bien es recomendable, no es obligatorio utilizar un mapa con simbología específica de orientación, pudiendo utilizarse a tal fin mapas topográficos normales. Se suelen usar mapas con escalas superiores a 1:15.000, siendo las más usadas 1:25.000 o 1:50.000.

La extensión del terreno donde se desarrolla una prueba de esta modalidad deportiva depende de la duración de la misma, el desnivel del terreno y la dificultad de desplazamiento por el mismo. Para las pruebas de entre 6 y 12 horas de duración, los mapas suelen abarcar una extensión de entre 20km^2y 50km^2.

La estrategia Como ya se puede haber intuido por las características de los Rogainings, la estrategia y la elección del número de controles que se desean visitar y el orden de los mismos es un factor muy importante. Es por ello, que el mapa con los controles se entrega siempre entre media hora y cuatro horas antes (dependiendo de la duración de la competición) con el fin de que los equipos planifiquen su recorrido.

Para ello, a la hora de planificar una ruta, se debe tener en cuenta la distancia y desnivel a recorrer así como el estado físico de todos los componentes del equipo; ya que una mala elección de la estrategia puede derivar en su descalificación por no

llegar a meta dentro del tiempo permitido.

Una estrategia óptima sobre el papel, será aquella que permita conseguir el mayor ratio de puntos por kilómetro recorrido y mayor ratio de puntos por desnivel acumulado de subida en función de las puntuaciones y distribución de los controles en el mapa.

Uso de elementos de ayuda a la navegación Al igual que ocurre en la orientación, está totalmente prohibido el uso durante la carrera de elementos que ayuden a la navegación como GPS, altímetros, cuenta pasos o smartphones, si bien estos últimos está permitido llevarlos precintados para utilizarlos en caso de emergencia.

1.2. Motivación

Desde que comencé a practicarla regularmente, no hace más de 10 años, la orientación se ha convertido para mi en el deporte por excelencia, mi hobby favorito. No sólo tiene los beneficios de cualquier actividad física, sino que además, supone un desafio mental muy estimulante. Cada vez que estoy perdido en plena montaña o en mitad de la ciudad con un mapa y una brújula, disfruto, me olvido de todo y sólo puedo pensar en cómo llegar a los siguientes puntos de control. Incluso podría afirmar que soy un "frikie" de la orientación: salgo de viaje y busco mapas de la zona a donde voy, carreras que coincidan con mis fechas de vacaciones o en su defecto, me hago mi propio mapa con los Servicios de Información Geográfica del lugar que visito para conocerlo; voy en carretera con el coche, de caminata o en bicicleta, y veo nuevas zonas interesantes para practicar orientación en cualquier rincón y en mis ratos libres, cuando no estoy practicándola, estoy buscando en Google Earth, lugares de mi isla o islas vecinas con las características adecuadas para elaborar un nuevo mapa donde practicar orientación.

Pero esto no acaba aquí. Este amor por la orientación me ha llevado, además de practicarla, a pertenecer al Club Deportivo OrientaGC con el cual tratamos de hacer llegar y dar a conocer este deporte al mayor número de personas posibles, convencidos de que una vez lo conozcan, no lo podrán dejar. Llevamos años organizando la Liga Gran Canaria de Orientación, varios raids de aventura y cinco ediciones hasta el

momento de la Rogaining Gran Canaria, una de las competiciones más antiguas en nuestro país de esta modalidad y que ha acogido el campeonato nacional de la Federación Española de Orientación en 2014 y el campeonato nacional de la Asociación Iberogaine en 2015.

Es en este último punto donde comienza a gestarse la idea de este proyecto. Como organizadores de varias pruebas de rogaining, edición tras edición, siempre hemos tratado de ir aumentando la participación en las mismas, mejorar en calidad, seguridad y visibilidad del evento. Para conseguir esto sin que lo costee el corredor, necesitamos a nuevos patrocinadores que, para darte su colaboración, exigen ser visibles a modo de promoción durante la prueba. Y es aquí donde surge la pregunta clave:

1.2.1. ¿Cómo se puede seguir una prueba de Rogaining?

A diferencia de la mayoría de deportes al aire libre donde la organización y el público sabe exactamente y de antemano por dónde van a pasar los corredores y es algo común ver a público durante el recorrido, pancartas publicitarias de los patrocinadores o fotógrafos y cámaras para inmortalizar los momentos que serán la imagen futura de la prueba; en el caso de la orientación en general y de la rogaining en particular, esto se torna bastante más complicado debido a dos características fundamentales:

- El mapa y la localización de los controles debe ser secreto hasta al instante justo de comienzo de la prueba y una vez comenzada, asegurarse de que siga siendo secreto para los corredores que aún no han tomado la salida.
- Es imposible conocer el recorrido exacto que realizará cada corredor o equipo ya que es totalmente libre.

Además en este último aspecto, la modalidad de rogaining tiene un hándicap añadido:

- En las carreras tradicionales de orientación el orden en el que se visitan las balizas es conocido por la organización de antemano, por lo que esta puede

intuir o aproximar cuál es el trazado o trazados óptimos o más probables para ir de un punto a otro, colocar algún control en una zona destinada al público por la que pasen todos los corredores (conocido como baliza del espectador), indicar a los fotógrafos donde se pueden colocar o, a nivel de pruebas nacionales e internacionales, ubicar cámaras en los puntos más interesantes. Esto además, al igual que se hace por ejemplo en las pruebas de contrarreloj en ciclismo, permite hacer un seguimiento de tiempos o clasificaciones intermedias de los corredores de una categoría mediante la toma de tiempos parciales en diferentes puntos del recorrido, lo que permite darle cierta emoción a la prueba durante su desarrollo y enganchar al espectador sin una inversión desproporcionada.

- En el caso de una Rogaining, desaparece totalmente esta posibilidad. El organizador no puede saber qué orden es el elegido por un equipo para visitar las balizas, qué balizas va a buscar y cuáles no ni qué camino elegirá para ir de un control al siguiente. Esto impide conocer cuántos puntos lleva cada equipo en cada instante de la prueba o la clasificación virtual. Poner cámaras fijas en determinados controles pierde el sentido, a no ser que se tuviese presupuesto para poner una cámara en cada uno de los controles del recorrido, que suelen ser más de 40 para pruebas a partir de 6 horas. Y aún así, aunque se tuviese esta posibilidad, la mayor parte del tiempo, las cámaras estarían grabando una zona de terreno sin corredores; al ser pruebas de larga duración y mucha mayor distancia que una prueba tradicional, la separación entre controles hace que la frecuencia por la que los equipos pasan por un control sea mucho menor.

Esta gran diferencia entre una modalidad y otra, no sólo afecta a la visibilidad o posibilidad de seguimiento de la prueba, sino también a la seguridad de la misma. En la primera, si un corredor no llega a meta y sufre un accidente, es posible con un equipo de búsqueda, hacer el recorrido en sentido inverso y tener altas probabilidades de encontrarlo. En una rogaining, si bien se obliga a los equipos a llevar un móvil por corredor con batería totalmente cargada al inicio de la carrera, un posible accidente en una zona sin cobertura con una extensión de más de 20km^2 y sin posibilidad de saber qué recorrido eligió es mucho más complicado, requiriendo muchos más efectivos para la búsqueda y alargándose por lo general el tiempo de localización del

accidentado, con el riesgo que ello conlleva.

1.2.2. El uso de los dispositivos de seguimiento por GPS en la orientación

A raíz de lo expuesto, en los últimos años y en pro de la visibilidad y seguridad de los eventos, se ha introducido los dispositivos de seguimiento GPS en el deporte de las carreras de orientación. Esto ha mejorado la posibilidad de seguimiento de las pruebas, su espectacularidad y entretenimiento para el espectador.

Como organizadores de rogaining lo hemos ido utilizando progresivamente en las 3 últimas ediciones, comenzando inicialmente con el seguimiento únicamente de los equipos más punteros de la competición y terminando en las últimas ediciones con el seguimiento de todos los equipos participantes. Sin embargo, sin apoyo externo (que suele ser lo habitual) el precio de uso de esta tecnología (10 euros proximadamente por dispositivo en 2015) supone un desembolso muy grande para los pequeños promotores como nosotros y es por eso, que su uso no sea lo habitual en la mayoría de pruebas de rogaining celebradas en nuestro país.

Además, la mayoría de plataformas existentes, cómo veremos en el capítulo del Estado del Arte, si bien cubren la necesidad básica de saber dónde está cada equipo en cada momento y recorrido realizado hasta ese instante, no son lo suficientemente completas para llegar a los patrocinadores y espectador.

Así, metiendo todos estos ingredientes en una batidora: mi devoción por este deporte, el afán por divulgar y acercar la orientación a más personas, la premisa de como organizador dar mayor visibilidad y seguridad a mis eventos y, como fan, el hacer esta posibilidad accesible económicamente a todas las empresas y clubes; unido a la necesidad personal de encontrar un proyecto motivante que me incentivase a seguir delante del ordenador tras mi jornada laboral, es como surge esta idea.

1.3. Objetivo

El objetivo principal del presente trabajo es confeccionar una solución que permita realizar el seguimiento por GPS de pruebas orientación en la modalidad de

Rogaining a través de un portal Web y que sea accesible a cualquier organizador. Para ello se plantean los siguientes subobjetivos:

Desarrollar una herramienta autogestionada Debe ser el propio organizador el que pueda gestionar su evento, cargar el mapa, incluir a los participantes, definir los equipos, establecer los controles, recorridos, puntuaciones y todos los ajustes relacionados con la carrera sin necesidad de ayuda externa por parte del equipo de desarrollo o de mantenimiento de la solución.

Permitir la utilización de los mapas de carrera georreferenciados Debe permitir cargar los mapas específicos utilizados para la carrera y georreferenciarlos para facilitar la impresión de los tracks GPS y posiciones sobre el mismo.

Utilizar de smartphones para transmitir la posición de cada equipo Con el fin de hacerla accesible a todos los organizadores y reducir los costes, se plantea que la solución incluya una App móvil instalable en los teléfonos inteligentes de los participantes para transmitir la posición de estos y no tener que alquilar o adquirir dispositivos de seguimiento GPS dedicados.

Reproducir en directo y diferido la competición Se debe poder seguir en tiempo real la competición o bien poder volver a reproducirla tras haber finalizado.

Permitir configurar los parámetros de seguimiento Como números de equipos a seguir, centrado automático del mapa en un equipo, longitud del track, visibilidad de distintos parámetros de información en el mapa, opacidad del mapa de carrera o velocidad de la reproducción (solo en diferido). También es importante ofrecer, de cara a asegurar la igualdad de condiciones de todos los participantes, la posibilidad de configurar el momento a partir del cuál se puede realizar el seguimiento por internet en modo público y no desvelar el mapa ni la posición de los controles antes de tiempo.

Mostrar clasificación virtual durante el seguimiento y otras estadísticas de los equipos Con el fin de hacer más interesante el seguimiento para el públi-

co, se debe mostrar al menos los puntos estimados de cada equipo en un momento dado y su posición en carrera. Se considera interesante para la captación de seguidores el cálculo de otra información estadística como kilometros recorridos, desnivel acumulado, puntos restantes, puntos por kilómetro recorrido, ...

Permitir multi-seguimiento en modo rejilla Para poder seguir a varios equipos de forma simultánea, se considera implementar una funcionalidad que permita dividir la pantalla mediante una rejilla para seguir en cada cuadrícula a un participante distinto.

Implementar un espacio visible para patrocinadores De cara a facilitar la búsqueda de apoyos y colaboradores a los organizadores.

Utilizar varios dispositivos para seguimiento de un mismo equipo Al estar un mismo equipo formado por dos o más corredores, se plantea implementar la lógica necesaria para que se utilicen de forma alternativa o simultánea los teléfonos inteligentes de las personas que lo componen (por ejemplo, en función del nivel de bateria, pérdida de señal de uno de ellos, u otros factores a determinar).

Capítulo 2

Estado del arte

En los últimos años, probablemente motivado principalmente por el auge de las carreras de montaña, han ido aparecido nuevas plataformas y herramientas que permiten el seguimiento de los partipantes en tiempo real por GPS. Si bien, esta tecnología se ha trasladado a las carreras de orientación tradicionales (en las que los recorridos son lineales) desarrollándose algunas soluciones, no se ha extendido a las carreras en modalidad Rogaining, una rama minoritaria de este deporte, siendo pocas las alternativas encontradas.

Además, es necesario destacar, que de todas las soluciones encontradas, ninguna utiliza el smartphone como dispositivo para transmitir la posición de los corredores, haciendo todas uso de dispositivos GPS específicos [Figura 2.1] para desempeñar esta tarea; más compactos, ligeros y precisos que un smarphone pero con un sobrecoste adicional por dispositivo para el organizador.

Estos dispositivos se componen de un receptor GPS, una batería con una duración aproximada de entre 6 horas y dos semanas (en función de la frecuencia de actualización de la posición que se le configure) y una tarjeta SIM con la que se emiten los datos por GSM o GPRS al servidor configurado. Son resistentes a golpes y al agua, y suelen disponer de botón de SOS para utilizar en caso de accidente.

Figura 2.1: Dispositivo de seguimiento GPS Tipo

2.1. Seguimiento por GPS en carreras de orientación

Ya es habitual y casi un requisito, que las competiciones internacionales de orientación tradicional, dispongan de la posibilidad de seguir a los corredores más destacados durante la celebración de la prueba. Aunque es posible encontrar varias soluciones válidas para este fin, son pocas las destinadas exclusivamente a orientación.

2.1.1. GPSSeuranta

Echando un repaso a la sección de "Software for Orienteering" del sitio Web de la Federación Internacional de Orientación, IOF por sus siglas en inglés, se encuentra esta aplicación entre el listado publicado. Es la única de este tipo que se anuncia en el mismo.

Se trata de una solución basada en dispositivos GPS dedicados específicamente para el seguimiento de carreras de orientación. Cuenta con una interfaz web sencilla pero muy completa [Figura 2.2], en la que destacan principalmente las capacidades de análisis de tiempos parciales entre controles y comparación entre los distintos corredores. Las principales funcionalidades de esta aplicación son:

- Alternar entre salida normal o salida en masa (para reproducción en diferi-

do): en la que nos permite ver la evolución de la carrera como si todos los participantes hubiesen salido al mismo tiempo.

- Selección individualizada de los corredores que se desea seguir.
- Función de resaltado de los corredores.
- Función de centrado automático en un corredor elegido.
- Clasificación automática en tiempo real en función de los tiempos de carrera.
- Posibilidad de alternar entre ver la ruta completa o solo el número de minutos indicados en la configuración, con selección independiente por corredor.
- Cambiar la velocidad de reproducción de la carrera (para el reproducción en diferido).
- Ver durante el seguimiento un recuadro con la diferencia entre dos corredores seleccionados.
- Mostrar u ocultar el nombre del corredor.
- Elegir la unidad en la que se desea que se muestren las velocidades.
- Alternar entre modo PC y modo dispositivo móvil.

La configuración de los eventos debe ser realizada por el desarrollador de la aplicación.

2.1.2. TracTrac

Plataforma de reciente creación, en la que el seguimiento se realiza por dispositivos GPS dedicados. Se trata de una plataforma desarrollada inicialmente para el seguimiento de eventos de embarcaciones a vela, que ha extendido su uso para deportivos al aire libre en general. A fecha de 30 de mayo de 2018, sólo cuenta en su listado de eventos de orientación con alrededor de diez pruebas de orientación celebradas entre 2017 y 2018.

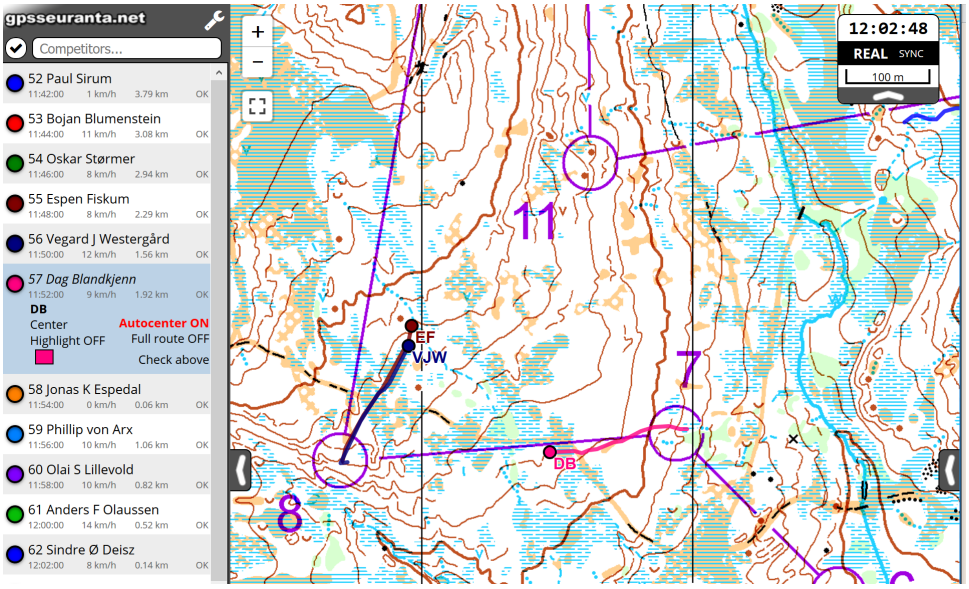


Figura 2.2: GPSSeraunta

Posee una interfaz moderna y visualmente atractiva [Figura 2.3], con posibilidades de configuración completas en lo relativo al *track*, tipo de mapa e información del corredor, pero sin funciones de análisis específicas de orientación aún como comparación de parciales entre otros. Entre las características a destacar, diferentes a las ya mencionadas en soluciones anteriores, encontramos:

- La posibilidad de exportar los resultados calculados por el seguimiento GPS a Excel o bien imprimirlos.
- La posibilidad de añadir comentarios sobre la carrera y compartirlo en redes sociales.
- Posibilidad de rotar el mapa
- Dispone de aplicación específica para Android e IOS desde la que visualizar el seguimiento del evento.

También en este caso, la configuración de los eventos debe ser realizada por el propietario de la plataforma.

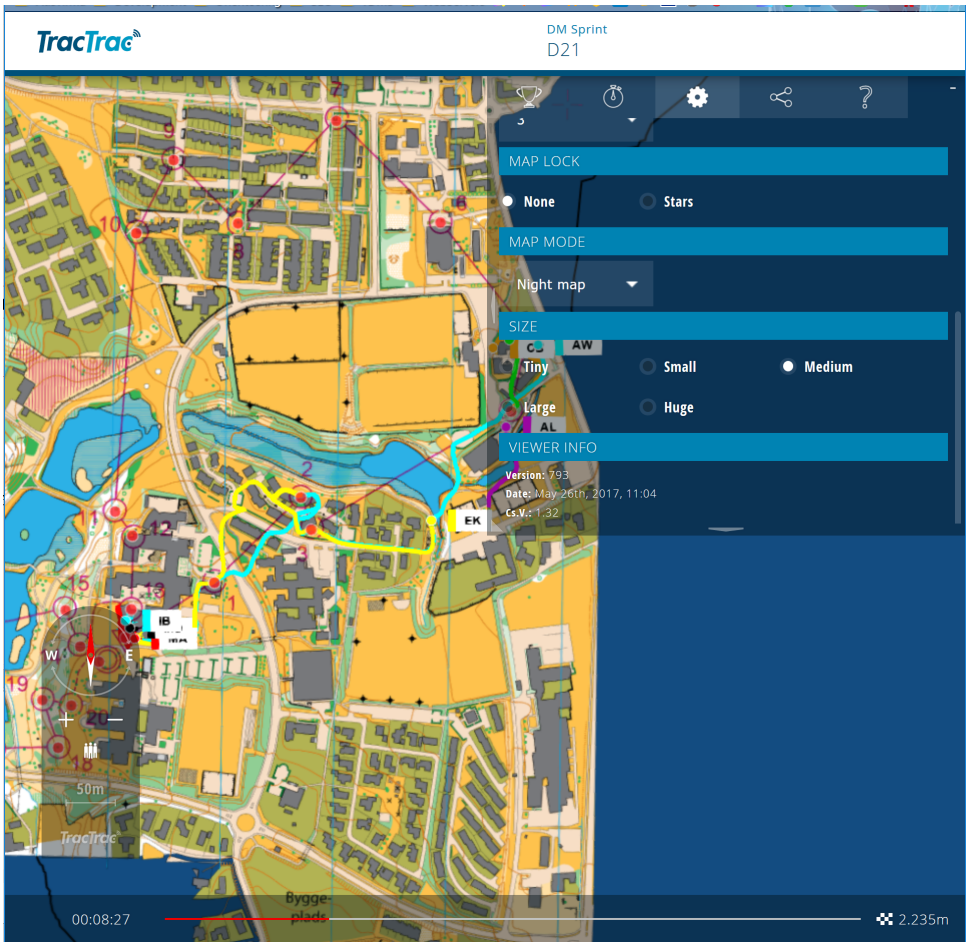


Figura 2.3: TracTrac

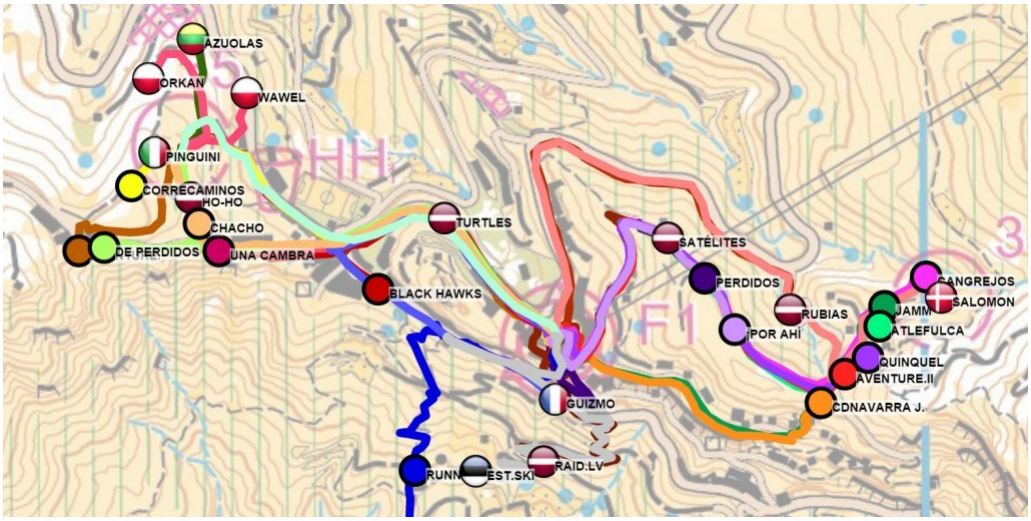


Figura 2.4: SportRec

2.1.3. SportRec

Esta solución, incluye entre las modalidades deportivas ofertadas, la Rogaining de forma específica [Figura 2.4]. El seguimiento es realizado utilizando dispositivos GPS dedicados y, en cuanto a las opciones de seguimiento se refiere, es la mas básica de las analizadas hasta el momento, sólo permitiendo ver el recorrido realizado y la distancia total acumulada por equipo (además de las configuraciones en relación al track, equipos visibles e información de los corredores).

Como valor añadido en relación al resto, destacar que ofrece, en la ventana de seguimiento, un espacio para un carrusel de imágenes destinado a poner los logos de patrocinadores y publicidad, de cara a dar visibilidad a los mismos. Esto puede ayudar a los organizadores a conseguir apoyos para sufragar los gastos de la prueba o del propio servicio de seguimiento.

Para su uso, es necesario suministrar toda la información a la empresa responsable para que configure el evento y nos indique los enlaces donde poder hacerle el seguimiento.

2.1.4. Track The Race

Se trata de una herramienta que, en virtud de la cantidad de eventos a los que se le ha hecho seguimiento que se listan en su portal Web, tiene una gran aceptación

en el territorio nacional español y en pruebas similares a los Rogaining como son los raids de aventura [Figura 2.5]. También cuenta en su haber con el seguimiento de numerosas pruebas de orientación tradicional y alguna Rogaining.

Como puntos fuertes y novedosos con respecto al resto de aplicaciones encontramos:

- Posibilidad de asociar puntuación a cada uno de los controles y cálculo automático de la puntuación acumulada por cada en participante en función de las balizas encontradas.
- Información asociada a cada control: número de corredores que lo han encontrado, número de corredores pendientes por localizarlo y hora de la primera vez que se localizó.
- Concepto de equipo como un participante, pudiendo ver información de la lista de personas asociadas a un equipo.
- Posibilidad de enviar mensajes de ánimo a equipos concretos (que verán al reproducir en diferido la carrera).
- Relación de todos los controles encontrados por cada equipo.

Si bien, como en el caso de las anteriores, es el responsable de la aplicación el encargado de configurar el evento, el organizador tiene acceso a determinados parámetros de configuración del mismo como la descripción o el cartel del evento.

2.2. Herramientas de análisis para carreras de orientación

En este punto se analizará dos de las utilidades software para el análisis de carreras de orientación a posteriori, a través de los track GPS obtenidos por los corredores por medio de relojes, smartphones u otros dispositivos, con el fin de extraer información y funcionalidades que puedan resultar de interés para la solución que se plantea.

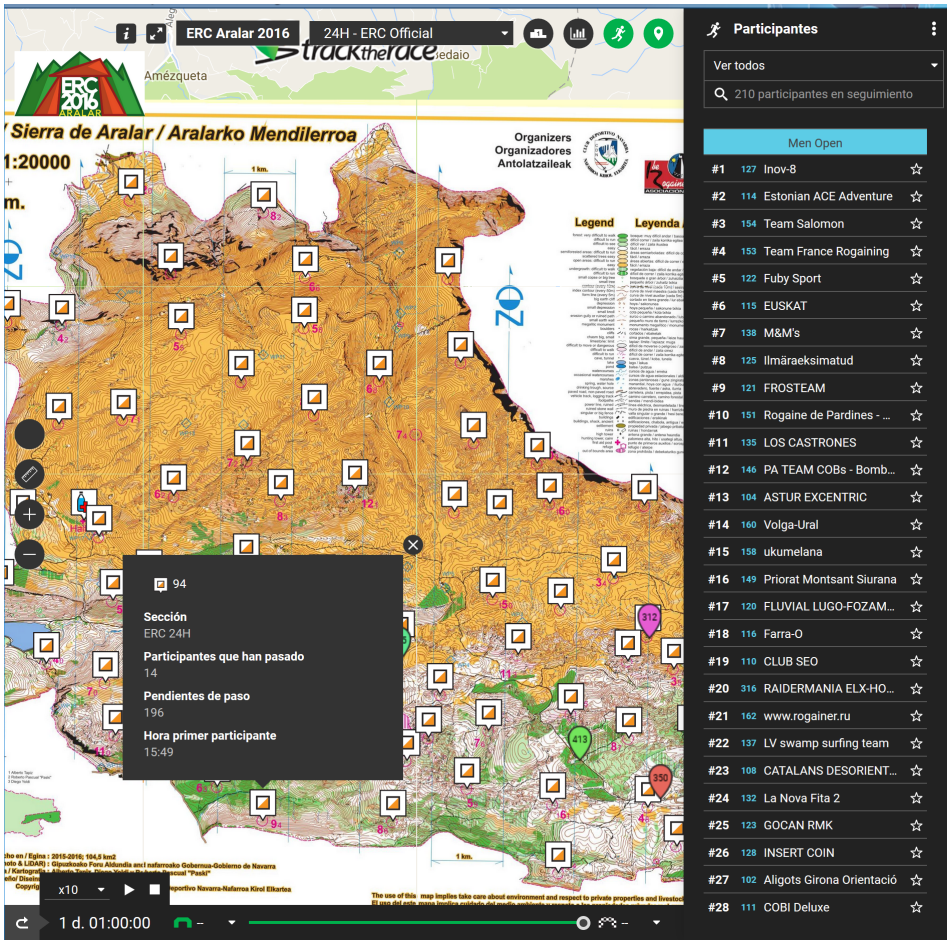


Figura 2.5: Track The Race

2.2.1. 2DReRun/3DReRun

Se trata de una herramienta Web que permite subir, previa creación de una cuenta gratuita en el portal, mapas, recorridos y tracks de carreras de orientación y entrenamientos para su análisis. Una vez cargada la información, ofrece una gran cantidad de parámetros que permiten el estudio de la carrera realizada de forma individualizada por corredor o en modo comparativo con los corredores seleccionados. Entre las funcionalidades más importantes destacan:

- Información completa en cada punto del track: Ritmo medio, tiempo de carrera, tiempo del parcial, altitud, etc.
- Herramienta para georreferenciación de mapas a partir de fotos o imágenes
- Vinculación de grabación de video con un recorrido existente para su posterior sincronización, lo que permite no sólo ver la posición en el mapa en cada momento, sino el terreno por el que se corría.

2.2.2. QuickRoute

Aplicación de escritorio freeware para Windows, que sobre un *track* y un mapa dados, añade otras funcionales interesantes de análisis de una carrera orientación, diferentes a lo ya propuesto por la solución anterior:

- Histograma de ritmo de carrera
- Gráfico de ritmo durante la sesión
- Integración con Google Earth, lo que permite tener una visión 3D del mapa utilizando la vista correspondiente de esta aplicación.
- Coloreado de ruta en función de distintos parámetros: velocidad, ritmo de carrera, ritmo cardíaco, altitud, desviación de dirección con respecto al siguiente control.
- Completo análisis de tiempos parciales entre controles: velocidad media entre controles, distancia en línea recta, distancia del track realizado, porcentaje

Figura 2.6: Glympse

de desviación entre ambas distancias , frecuencia cardíaca media (si el *track* cuenta con dicha información), etc.

2.3. App móviles con tecnología GPS aplicada a la orientación

2.3.1. Glympse

App móvil de seguimiento que permite compartir tu ubicación en tiempo real con otros usuarios y durante el tiempo que tu establezcas. Una vez un usuario comparte su ubicación, el seguimiento puede ser realizado desde un móvil, Tablet o su propia Web [Figura 2.6]. Ofrece SDK para usar las funcionalidades de Glympse con tu propia App. Está disponible para Android, IOS y Windows y dispone de una versión gratuita.

2.3.2. GPSOrienteering

App móvil disponible para Android que permite realizar todo tipo de carreras de orientación (orientación lineal, score, Rogaining, etc.) de forma virtual utilizando el GPS del dispositivo móvil y sin necesidad de que los controles estén colocados físicamente sobre el terreno.

Su funcionamiento se basa en la utilización de imágenes georreferenciadas de mapas de orientación y la colocación sobre estas de controles con coordenadas específicas. Cuando la aplicación detecta que la posición indicada por el GPS atraviesa el control (con un margen de error igual a la precisión en metros en ese momento de la señal GPS que se está recibiendo), lo marca como encontrado.

Al finalizar la carrera te permite realizar un análisis detallado de la misma informándote de los tiempos parciales entre controles, velocidad media, gráficos de desnivel, distancia recorrida, etc. Además te permite visualizar tu recorrido sobre el mapa y volver a reproducir la carrera realizada. También ofrece la posibilidad de subir tus resultados al servidor para crear una clasificación virtual entre todos los que hayan hecho el mismo recorrido, así como comparar su trayectos y tiempos.

Capítulo 3

Entorno de desarrollo

En el presente capítulo se describe los componentes lógicos: software, sistemas de gestión de bases de datos y lenguajes de programación y *frameworks*; y físicos: hardware, utilizados para el desarrollo de cada una de las partes de la solución de estudio. Como se detalla más adelante, dicha solución constará de tres partes bien diferenciadas: una aplicación Web, un *backend* a modo de servicio Web y una aplicación móvil para dispositivos con sistema operativo Android.

3.1. Componentes lógicos

3.1.1. Software

Se ha elegido aplicaciones de la compañia Jetbrains incluidas en su licencia gratuita para estudiantes por un año [5] para la implementación de los componentes de la solución y de la capa de datos de la misma.

Intellij Idea Ultimate v. 2016.1 - 2017.1 [6] Entorno de desarrollo integrado multiplataforma orientado al desarrollo de aplicaciones en lenguaje Java y lenguajes Web. Posee gran cantidad de funcionalidades de asistencia al desarrollo: generación automática de código, autocompletado, refactorización, análisis inteligente de código, etc. todas ellas destinadas a aumentar la productividad y velocidad al escribir aplicaciones en Java y el resto de lenguajes soportados. Además ofrece soporte nativo para los *frameworks* que se han elegido para implementar las partes del proyecto

como son Spring para Java, AngularJS para Javascript y desarrollo de aplicaciones para sistemas Android. También incluye soporte integrado para distintos sistemas de control de versiones y una amplia biblioteca de plugins que añaden funcionalidades adicionales que facilitan las tareas de desarrollo. Como aspecto final a destacar, es posible habilitar el seguimiento de tiempo durante el desarrollo lo que permite computar de forma sencilla el tiempo empleado en las tareas de implementación sin necesidad de ninguna herramienta adicional.

DataGrip v. 2016.1 - 2017.1 [7] IDE multiplataforma para trabajo con bases de datos SQL. Incluye soporte para los SGBDs SQL más comunes como: DB2, Derby, H2, HSQLDB, MySQL, Oracle, PostgreSQL, SQL Server, Sqlite o Sybase. Además de la ventaja que ofrece tener un único entorno para gestionar los distintos SGBDs de una aplicación, incluye muchas de las funcionalidades de análisis de código del resto de IDEs de la compañia, como autocompletado o análisis de código, lo que aumenta la productividad.

Toad Data Modeler 5.3 Freeware Edition [8] Herramienta para diseño de bases de datos que permite construir modelos de datos lógicos y físicos de forma visual y sencilla, permitiendo generar de forma automática los scripts SQL/DDL asociados a dichos modelos, haciendo más fácil el matenimiento y modificación de los mismos. Tiene soporte para los SGBDs SQL más comunes. La edición Freeware tiene una limitación de 25 entidades como máximo, que son suficientes para los modelos utilizados en el presente proyecto.

JetBrains Youtrack [9] Software para gestión de proyectos y seguimiento de tareas e incidencias con funcionalidades de planificación y gestión del tiempo por tareas. Tiene integración mediante plugin adicional con el IDE IntelliJ Idea Ultimate y con distintos repositorios de control de versiones del código, permitiendo asignar a cada tarea la relación de cambios de código asociados y el tiempo empleado en la misma. Es gratuita su uso para equipos de hasta 10 usuarios.

GitLab Community Edition v 9.x [10] Servicio Web de control de versiones y desarrollo colaborativo basado en Git. Adicionalmente ofrece otras funcionalidades, entre ellas el alojamiento de Wikis asociada a los distintos proyectos, que ha sido utilizada para anotaciones o aprendizajes adquiridos sobre distintas tecnologías u otros aspectos del presente proyecto. También, mediante plugin, es posible integrarlo con el IDE IntelliJ Idea Ultimate y la herramiento de gestión de proyectos YouTrack.

Herramientas del programa Microsoft MSDN Academic Alliance Se ha utilizado el **Microsoft Project 2016 [11],** como herramienta de planificación de proyectos, para la estimación inicial del presente trabajo y distribución por etapas, y el **Microsoft Visio Professional 2010 [12],** sofware de dibujo vectorial, para la elaboración de los distintos diagramas durante las fases de análisis y diseño.

3.1.2. SGBD (Sistemas de Gestión de Bases de Datos)

PostgreSQL [15] Se ha elegido **PostgreSQL** como SGBD (Sistema de Gestión de Bases de Datos) para dar cabida a la capa de datos principal de la solución, por ser un sistema de código abierto, libre y relacional.

SQLite3 [17] SGBD implementado de forma nativa en Android que permite la persistencia de información en los dispositivos de forma sencilla.

3.1.3. Lenguajes de programación y frameworks

3.1.3.1. Java

Spring Framework [13] Spring es un framework de código abierto para el desarrollo de aplicaciones Java. Proporciona un modelo completo multiplataforma de programación y configuración para aplicaciones empresariales modernas basadas en Java. Entre las principales caracteristicas de Spring encontramos:

- Inyección de dependencias
- Programación orientada a aspectos

- Marco para aplicaciones Web MVC (Modelo Vista Controlador) y servicios web RESTful
- Soporte básico para operaciones sobre bases de datos con JDBC, JPA, JMS Mucho más...

Android SDK [16] Conjunto de herramientas de desarrollo necesarias para la implementación de cualquier aplicación para Android Nativa

3.1.3.2. Javascript

Angular [18] Se trata de un framework de código abierto para Javascript, mantenido por Google, orientado a la creación de aplicaciones web de una sola página (SPA por sus siglas en inglés). Trata de acercar al desarrollo Web el modelo vista controlador, permite desarrollar gran cantidad de código reutilizable (componentes) y separa la manipulación directa del DOM de la lógica de la aplicación en la confección de páginas Web dinámicas, creando un doble enlace entre la vista (código HTML) y el controlador (código Javascript). Además integra el uso de la inyección de dependencias lo que permite trasladar servicios y otras funcionalidades al lado del cliente aligerando la carga de proceso del backend.

La versión del framework utilizada en el presente proyecto es la 1.x, conocida comúnente como AngularJS o Angular 1. A partir de la versión versión 2.x, el framework sufre una profunda transformación en su estructura y filosofía de programación, se pasa a Typescript como lenguaje principal (aunque sigue soportando Javascript) y modifica la filosofía de programacíon y funcionamiento del mismo pero conservando su esencia y objetivo. Esta segunda versión (y posteriores) es conocida como Angular 2 o Angular.

Google Maps JavaScript API V3 [19] Libreria javascript desarrollada por Google que permite la integración y trabajo con los Mapas de Google en nuestras aplicaciones.

Librerías Javascript de terceros Mediante la utilización de Bower (gestor de paquetes Javascript, HTML y CSS), se han utilizado una serie de libreria de terceros para la implementación de determinadas funcionalidades de la solución entre las que destacan:

- **cryptojslib**: Colección de algoritmos estándares y seguros de encriptación implementados en Javascript.
- **imgcache**: Librería Javascript que permite el almacenamiento local de imágenes de forma sencilla para su uso offline, mejorando de este modo la velocidad de carga de las aplicaciones.
- **moment**: Librería Javascript para el trabajo con Fechas.
- **api-check**: Librería Javascript que permite asociar validaciones a una API para asegurar el uso correcto de la misma.

Google Firebase - Firebase Cloud Messaging (FCM) [20] Plataforma propiedad de Google que ofrece un conjunto de herramientas que facilitan la implementación de determinadas funcionalidades en la aplicaciones Web y Móviles, como la autentificación, la gestión de bases de datos, el informe de errores o el intercambio de mensajes y la comunicación entre los distintos actores de un sistema.

Para la solución propuesta en el presente proyecto, se ha hecho uso de Firebase Cloud Messaging, un servicio que permite el intercambio de mensajes entre el servidor y los clientes de manera eficiente y sencilla.

Bootstrap [21] Framework de código abierto para diseño y maquetación de sitios y aplicaciones web. Incluye una gran variedad de componentes web: menús, botones, paneles, etc, prediseñados y con estilos definidos que permiten componer de una forma más rápida y sencilla la interfaz gráfica de la aplicación que se está desarrollando.

3.2. Hardware

El presente proyecto se ha desarrollando usando un portátil Dell modelo XPS 15 9550 con las siguientes características:

- Windows 10
- Procesador: I7 6700HQ a 3.5GHZ
- 16GB de memoria RAM.
- Disco Duro 500GB SSD

Las pruebas de la aplicación Android en un dispositivo real, se han efectuado en un smartphone Samsung Galaxy S5 con sistema operativo Android 6.0.1 (a la finalización del mismo).

Para las pruebas de la aplicación Web y el servicio Web se ha contratado un servidor virtual en la nube hospedado en Time4VPS con las siguientes características:

- CPU con 6 nucleos de 1,8GHz cada uno
- 4GB de Ram
- 160GB de espacio de almacenamiento
- Sistema Operativo CentOs 7

Por último se ha utilizado dos servidores virtuales en la nube hospedados en Time4VPS que alojan, respectivamente, las herramientas YouTrack y GitLab utilizadas. Las características de cada uno de estos servidores son las siguientes:

- 2 CPUs a 2.4GHz
- 2GB RAM
- 80GB de espacio de almacenamiento
- Sistema Operativo CentOs7

Capítulo 4

Planificación y metodología de trabajo

En el presente capítulo se expone la programación y temporalización del presente proyecto, así como la metodología de trabajo seguida durante el desarrollo del mismo.

4.1. Planificación inicial

En la figura 4.1 se muestra la planificación inicial basada en un calendario en el que se dedicarán 20 horas semanales a la elaboración del proyecto: dos horas cada día de lunes a viernes y cinco horas cada fin de semana.

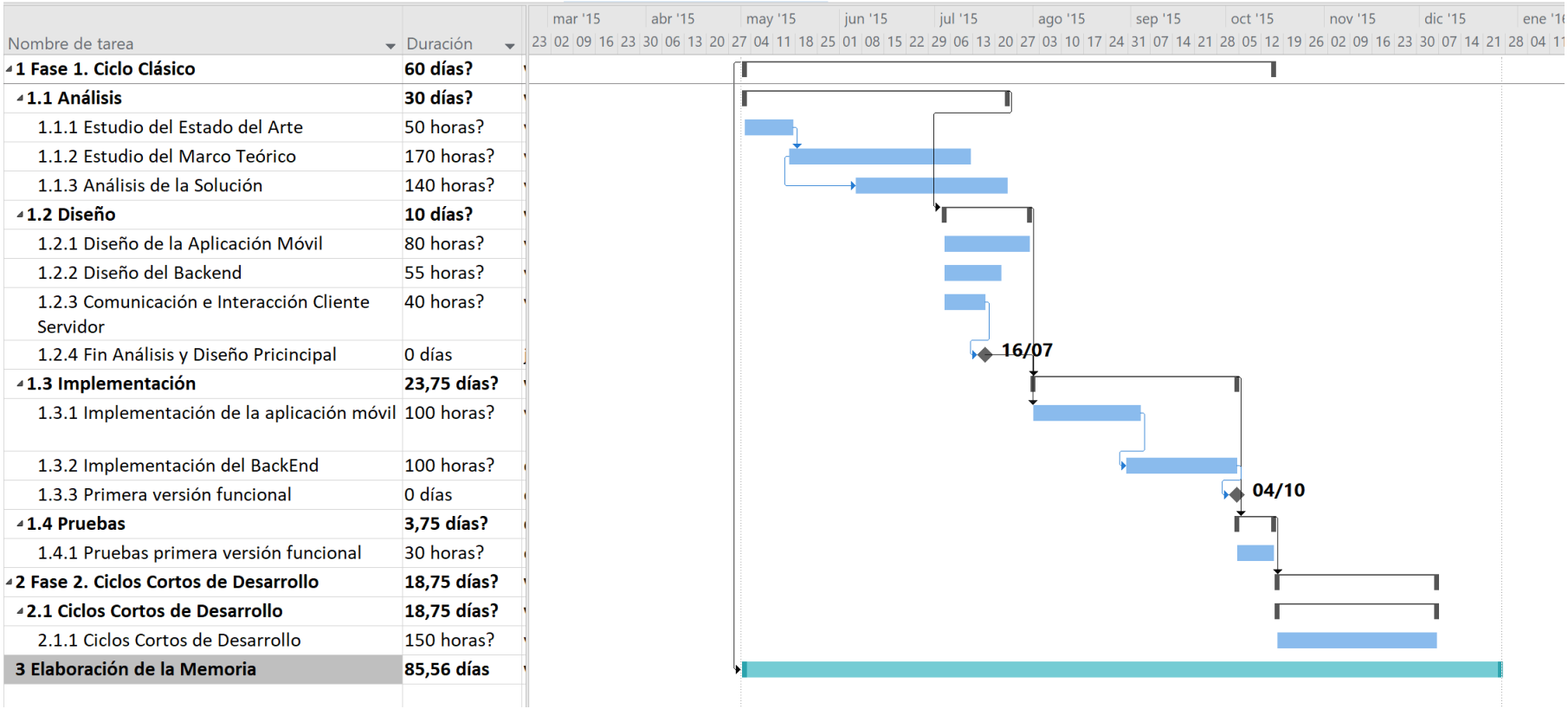


Figura 4.1: Planificación Inicial

4.2. Temporalización real del proyecto

La temporalización final del proyecto se extendió durante más de dos años, con un retraso en el inicio del mismo y con periodos de inactividad de más de 4 o 5 meses en diferentes etapas. La realización de la memoria no se realizó paralelamente a la finalización de las diferentes etapas, y se pospuso hasta la finalización de la fase de desarrollo.

4.3. Metodología de trabajo

Teniendo en cuenta el tiempo efectivo de trabajo en el proyecto, el desarrollo del mismo se ha realizado según lo estipulado inicialmente, pudiendo dividirse en dos fases bien diferenciadas con dos metodología de trabajo distintas:

- Una primera fase inicial, compuesta por un primer ciclo largo de desarrollo clásico, en la que se realizó un estudio del Estado del Arte en las soluciones de seguimiento existentes para esta modalidad deportiva o similares, un análisis exhaustivo de los requisitos de la aplicación y estudio del Marco teórico necesario para el desarrollo del proyecto y el diseño e implementación del *backend* (portal Web) y de una versión funcional de la aplicación móvil.

- A partir de esta versión funcional, se comenzón con la segunda fase en la que se realizó sucesivos ciclos de desarrollos cortos apoyados por pruebas de campo en los que se fue incluyendo sucesivamente más funcionalidades y complejidad y depurando los problemas encontrados.

Capítulo 5

Análisis y diseño

En este capítulo se explica en detalle a qué se pretende dar solución con el presente proyecto y cómo se espera conseguirlo, prestando especial atención a los retos más relevantes que se plantean en función de los objetivos expuestos en el apartado 1.3 del capítulo Introductorio:

- Desarrollar una herramienta autogestionada.
- Permitir la utilización de los mapas de carrera georreferenciados.
- Utilizar *smartphones* para transmitir la posición de cada equipo.
- Reproducir en directo y diferido la competición.
- Permitir configurar los parámetros de seguimiento.
- Mostrar clasificación virtual durante el seguimiento y otras estadísticas de los equipos.
- Permitir multi-seguimiento en modo rejilla.
- Implementar un espacio visible para patrocinadores.
- Utilizar varios dispositivos para seguimiento de un mismo equipo.

En base a los objetivos principales, se detectan a priori los siguientes retos a resolver que posteriormente explicaremos en detalle:

- Auto-gestión de evento por parte del organizador.
- Restricción de accesos a parte de la aplicación.
- Carga de mapa georreferenciado.
- Dibujado sobre mapa de objetos en función del zoom/escala.
- Control de hora de comienzo de visibilidad pública de seguimiento.
- Comunicar los móviles con la aplicación y viceversa.
- Gestionar la imposibilidad de enviar información por falta de cobertura u otros motivos.
- Control/Verificación de registros de los corredores a través de sus dispositivos.
- Configuración automática de comienzo del seguimiento a la hora establecida.
- Distinción de los equipos durante el seguimiento.
- Estimación de localización de controles en función de distancia al mismo.
- Gestión de problemas de batería en los dispositivos durante el seguimiento.
- Gestión de la reproducción del evento en tiempo real.
- Asociación de más de un dispositivo por equipo

5.1. Análisis: Los retos más importantes

5.1.1. Auto-gestión de evento por parte del organizador

Uno de los hitos principales del presente proyecto es que el software resultante permita a cada organizador tener una cuenta propia con la que gestionar sus propios eventos, sin necesidad de intervención por parte de los desarrolladores o un equipo de soporte adicional. Esto incluye:

- La **creación** de tantos **eventos nuevos** como desee:

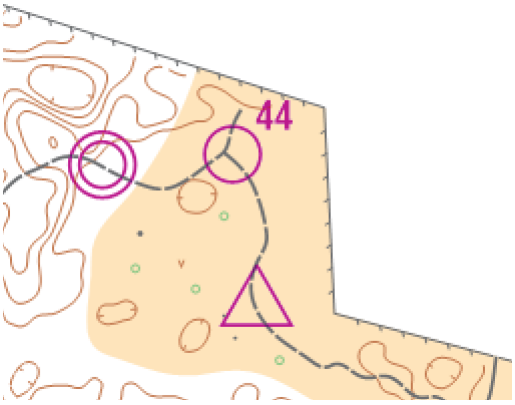


Figura 5.1: Salida, Meta y Control de Paso

  - Un evento se identifica por un nombre, una fecha de celebración, un lugar y un organizador.

- La definición de los **mapas** del evento:
  - Cada evento puede tener asociados uno o varios mapas.
  - La extensión, escala, equidistancia y localización de cada mapa del evento puede ser distinta.
- La definición de los **controles** asociados a un mapa:
  - Cada control está asociado a un único mapa y tiene una localización determinada.
  - Existen 3 tipos de controles básicos [Figura 5.1]:
    - Control de Salida o *Start*. Identificado por un triángulo de color magenta. Sólo puede haber una única salida por mapa.
    - Control de *Finish* o Meta. Identificados por un doble círculo concéntrico de color magenta. En cada mapa sólo puede haber un único control de meta.
    - Control de Paso: Identificados por un círculo y un código numérico adyacente de color magenta. Pueden existir tantos controles como se desee asociados a un mismo mapa.
- La definición de las distintas **modalidades** de cada evento:

- Un evento puede tener una o varias modalidades.
- Una modalidad se pude distinguir de otra por el tiempo máximo con el que cuentan los equipos para localizar el máximo número de controles posibles, por los controles válidos para dicha modalidad, o por ambas cosas.
- Cada modalidad puede tener una hora de comienzo distinta, y dada la premisa anterior, un tiempo de finalización distinto al resto.
- Diferentes modalidades pueden tener controles coincidentes, compartidos por varias modalidades y controles independientes, sólo presentes en dicha modalidad.
- Un mismo control coincidente, puede tener puntuación distinta para cada modalidad.
- Cada modalidad tiene un único mapa asociado. Los mapas de distintas modalidades pueden ser el mismo o distintos.
- El punto de salida y final de cada mapa puede variar.

- La definición de las distintas **categorías** asociadas a cada una de las modalidades de un evento:
  - Cada modalidad puede tener asociada una o más categorías.
  - Una categoría se distingue de otra por el sexo y edad de los componentes de los equipos.
  - Todas las categorías de una modalidad comparten la configuración establecida para la modalidad en la que se incluyen, esto es: tiempo máximo, mapa, controles y puntuación de los mismos.
- El registro de los **equipos y miembros** del mismo en el evento:
  - Un equipo se inscribe en una categoría, de una modalidad, de un evento dado.
  - El equipo se identifica por un dorsal y un nombre.

- Cada equipo puede estar formado por entre 2 y 5 corredores que deben cumplir los parámetros de edad y sexo asociados a la categoría en la que están inscritos.
- Cada equipo que desee ser seguido debe tener asociado al menos un dispositivo a uno de sus miembros.
- Cada miembro del equipo puede tener un dispositivo distinto asociado. Esta afirmación añade una complejidad adicional al sistema que será analizada en el apartado 5.1.14.

- Realizar **el seguimiento** de cada evento.

5.1.2. Restricción de accesos a parte de la aplicación

El portal Web de la solución, deberá tener dos partes diferenciadas:

- Una parte pública, desde donde poder ver la relación de eventos pasados, presentes y futuros disponibles para seguir así como información básica de los mismos que incluya al menos hora y fecha del evento, modalidades y hora de comienzo del seguimiento de cada una.
- Una parte de administración de los eventos, privada, accesible sólo a los organizadores.

Esto implica que:

- El portal deberá contar con un mecanismo de registro e inicio de sesión de usuarios para acceder a la consola de administración.
- Cada gestor de eventos deberá tener acceso sólo a administrar sus eventos y no ver los eventos del resto de gestores.
- No se considera necesario hacer distinción entre distintos roles de usuario en la parte de administración.

5.1.3. Carga de mapa georreferenciado

Uno de los primeros problemas que se presentan es la necesidad de poder utilizar, para realizar el seguimiento de los corredores, una imagen del mapa usado por los equipos en carrera. Esto implica que la aplicación debe ser capaz de mostrar, sobre dicha imagen, la posición exacta de cada equipo, así como su recorrido, en función de las coordenadas geográficas transmitidas por los dispositivos de los participantes.

Mapas georreferenciados

Los programas utilizados habitualmente para cartografía de orientación, así como cualquier programa SIG (Sistema de Información Geográfica), tienen integradas funcionalidades que permiten producir imágenes de los mapas georreferenciadas. Esta georreferenciación no es más que la generación, junto con la imagen del mapa, de un archivo adicional denominado archivo de mundo (de la traducción literal en inglés *World File*) o archivo de georreferenciación. Estos archivos, tienen el mismo nombre que la imagen asociada, siendo su extensión ligeramente distinta [Figura 5.2]. Se trata de archivos de texto cuyo contenido describe localización, escala y orientación de la imagen. Así y tomando como ejemplo un archivo de georreferenciación basado en el sistema de coordenadas UTM (Universal Transverse Mercator) [24] nos encontramos con los siguientes valores separados por línea [23]:

Ejemplos de nombres de archivos de georreferenciación

Archivo de datos ráster	Archivos de georreferenciación
image.tif	image.tfw o image.tifw
image.bil	image.blw o image.bilw
image.jpg	image.jgw o image.jpgw
image.raster	image.rasterw
image.bt	image.btw

Figura 5.2: Archivos de georreferenciación asociados a distintas extensiones [22]

Linea	Valor	Significado
1	2.116666666667	Longitud (en metros) en el eje X de cada píxel de la imagen
2	0.000000000000	Ángulo de giro respecto al eje Y (normalmente 0)
3	0.000000000000	Ángulo de giro respecto al eje X (normalmente 0)
4	-2.116666666667	Longitud (en metros) en el Y de cada píxel de la imagen (en valor absoluto)
5	437975.3083	Coordenada X (longitud) en metros (UTM) del píxel correspondiente a la esquina superior izquierda de la imagen
6	3097477.6920	Coordenada Y(latitud) en metros (UTM) del píxel correspondiente a la esquina superior izquierda de la imagen

Como dato adicional, dado que el sistema de coordenadas UTM divide la tierra en 60 husos de 6^{o} cada uno, es necesario conocer el huso horario en el que se localiza la esquina superior izquierda del mapa, ya que la coordenada indicada en las líneas 5 y 6 se repite en los 60 husos del sistema UTM.

Una vez georreferenciado el mapa, será prácticamente trivial el mostrar la posi-

ción reportada por los dispositivos en el lugar del mapa que corresponde.

Por lo tanto, si se impone como premisa de entrada al sistema que el usuario deba aportar una imagen georreferenciada con sus dos archivos y un huso del sistema UTM, se podrá utilizar esta información para integrarla en el sistema y dar solución a este problema.

5.1.4. Dibujado sobre mapa de objetos en función del zoom/escala

Una vez cargado el mapa en la aplicación, se desea:

- Mostrar sobre el mismo todos los controles, así como la posición y *track* de cada equipo cuando corresponda.
- Permitir que el usuario pueda hacer zoom sobre el mapa durante el seguimiento o su posterior reproducción.
- Seleccionar la posición de un equipo en concreto para acceder a información adicional sobre el mismo como distancia recorrida, puntos alcanzados hasta el momento, etc.

Por este motivo, el dibujado de estos elementos tendrá que ser proporcional a la parte visible del mapa de modo que sea legible en todo momento y redimensionando su tamaño en función de la parte del mapa visible en cada momento para que siempre ocupe la misma superficie en el mismo.

5.1.5. Control de hora de comienzo de visibilidad pública de seguimiento

Dado que el secreto mejor guardado de las carreras de orientación es el mapa y la ubicación de los controles, esta información debe ser secreta al menos hasta la misma hora de salida, la solución debe proveer un mecanismo que permita controlar la hora a partir de la cuál, el público en general podrá acceder a la parte pública del evento para comenzar a seguirlo.

5.1.6. Comunicar los móviles con la aplicación y viceversa

La solución debe permitir la conversación y paso de información entre los dispositivos móviles de los participantes y el servidor en los dos sentidos, por lo que se identifican dos escenarios en función del sentido del mensaje:

- **Escenario 1. Envío de información desde los dispositivos móviles al servidor:** Este escenario es el más trivial de los dos expuestos, ya que el servidor es una parte estática y conocida del sistema, identificada claramente por una URI y que mediante la exposición de una API con un contrato conocido por los dispositivos, estará siempre a la escucha y dispuesto a recibir los mensajes que todos los clientes necesiten enviarle.

- **Escenario 2. Envío de información desde el servidor a los dispositivos móviles:** Es en este sentido de la comunicación donde se plantea la mayor dificultad. El servidor debe poder comunicarse en determinados momentos con clientes con las siguientes características:

  - Se trata de clientes sin una URI conocida y ni con dirección IP estática que hagan posible la comunicación de la misma forma que en el escenario anterior.

  - Se trata de dispositivos en los que el recurso de acceso a Internet es limitado por los planes de datos contratados con su operador. Además, no siempre están accesibles desde internet debido a problemas de cobertura, desconexiones voluntarias temporales de la red (apagado durante la noche, modo avión, ahorro de datos, etc.).

  - Esto implica que la comunicación debe ser pro-activa, iniciada desde el lado del servidor, no pudiendo estar los dispositivos consultando constantemente al servidor (modo pasivo) en busca de nueva información debido a los problemas de rendimiento, uso de datos y consumo de batería que esto supondría.

  - Se plantean casos de uso en los que la información que se necesite enviar, deba estar disponible y actualizada lo antes posible en los dispositivos

móviles (esto es desde que tengan conexión a internet) como es el caso de modificaciones de horas de comienzo e inicio de seguimiento en una categoría o modalidad dada, actualización de la información de un evento, o comunicaciones de urgencia desde la organización a los participantes.

- No se debe enviar siempre la misma información a todos los clientes ya que depende de la prueba, modalidad y categoría en la que vayan a participar, por lo que el mecanismo de comunicación debe permitir discriminar en función de estos parámetros.

La solución a este segundo escenario, pasa por lo tanto, por usar una implementación basada en una estrategia de comunicación *push* (comunicación iniciada por el servidor) en contraposición a la estrategia *pull* (comunicación iniciada por el cliente).

5.1.7. Gestionar la imposibilidad de enviar información por falta de cobertura u otros motivos

Al tratarse de clientes corriendo en dispositivos móviles es necesario contemplar la desconexión de estos de internet por pérdida de cobertura, apagado de los mismos, modo avión, etc. Por esta razón debe existir un mecanismo que permita el encolamiento de los mensajes a enviar, la discriminación de los destinatarios de los mensajes y el acuse de recibo de los mismos por parte de los receptores para asegurarse, de una forma óptima, de que cada mensaje llegue a todos los clientes a los que vaya dirigido.

5.1.8. Control/Verificación de registros de los corredores a través de sus dispositivos

Se desea que el registro en cada evento para el seguimiento, se realice de forma activa directamente por los corredores a través de la aplicación para *smartphone*. De este modo, el corredor autoriza expresamente a ser seguido durante la competición.

Sin embargo, la aplicación debe controlar la inscripción indiscriminada en un evento por usuarios que no participen en el mismo, para asegurase de que todos los

dispositivos que aparezcan en el mapa de carrera correspondan a corredores reales del evento.

5.1.9. Configuración automática de comienzo del seguimiento a la hora establecida

Una vez registrados los dispositivos en un evento, el sistema debe permitir configurarlos para que el día y la hora establecida por el organizador, el *smartphone* del corredor comience a transmitir la posición de forma automática sin necesidad de ninguna acción por parte de los mismos.

El sistema debe comprobar el estado de todos los dispositivos registrados en el evento unos minutos antes del inicio del mismo e informar a los organizadores si alguno de los dispositivos no está transmitiendo la posición con el fin de que éste pueda actuar y comunicarse con el corredor para hacer las verificaciones oportunas: móvil encendido, con batería, datos habilitados, etc.

Como funcionalidad añadida se propone que el sistema envié una o varias notificaciones a los corredores con anterioridad al inicio del evento recordando que deben llevar su dispositivo cargado, con datos habilitados a la salida para el correcto funcionamiento de la aplicación.

5.1.10. Distinción de los equipos durante el seguimiento

Tanto durante el proceso de seguimiento de un evento en tiempo real como en su reproducción a posteriori, debe ser posible identificar claramente a cada equipo, así como separarlos, al menos, en función de la modalidad en la que participan. Es necesario, por lo tanto, durante el proceso de registro en el evento mencionado anteriormente, identificar al dispositivo que se está registrando mediante un dato único asociado al mismo y que deberá ser transmitido cada vez que se reporte información al servidor para poder vincular dicha información al equipo correspondiente.

5.1.11. Estimación de localización de controles en función de distancia al mismo

La posibilidad de estimar la cantidad de puntos conseguidos por los equipos en cada momento de carrera y, en base a ello, realizar una clasificación virtual, en tiempo real, se considera una funcionalidad importante ya que ofrece un valor añadido tanto al espectador como al organizador y medios informativos de la carrera:

- Al primero, saber cómo va su equipo favorito o quién es el líder de la categoría o modalidad, añade un plus de emoción necesario para conseguir mantenerlo durante más tiempo pendiente del evento.
- A los segundos, tener información estimada de la marcha de la prueba, permite hacer una mejor difusión de la misma en el centro de competición y a través de los canales de comunicación que utilicen como redes sociales, página Web, prensa, etc.

Para esta estimación, se plantea definir un umbral de distancia, con respecto a la ubicación de cada control, para el cuál el sistema considere la baliza como encontrada y sume los puntos correspondientes a la puntuación total del equipo. Este umbral, deberá adaptarse dinámicamente en función del margen de error de la posición reportada en cada momento (esta información, por lo tanto, se deberá incluir en la información enviada desde los dispositivos al servidor), ya que puede variar en función de la densidad de la vegetación alta, la orografía del terreno u otros factores.

5.1.12. Gestión de problemas de batería en los dispositivos durante el seguimiento

Uno de los puntos débiles del sistema radica en que su funcionamiento depende de la utilización de dispositivos de terceros, los *smartphones* de los corredores; cuyo óptimo estado de funcionamiento no es verificado. Sin embargo, aún partiendo de la premisa de que todos los dispositivos utilizados para un evento funcionan correctamente, esto es:

- El sensor GPS funciona según se espera y está activado.

- El usuario tiene contratada una tarifa de datos con suficiente tráfico restante para poder enviar la información al servidor durante la carrera.
- La antena de recepción de señal móvil funciona correctamente, el proveedor de servicios del participante ofrece cobertura en la zona de competición y tiene el envío y recepción de datos habilitado.
- La batería no está deteriorada y está completamente cargada al inicio de la prueba.

la correcta gestión de esta última una vez en carrera es un elemento crítico para el éxito de la solución. Se trata de pruebas con una duración media de entre 6 y 12 horas y un dispositivo que se apague por pérdida de batería ni registra ni reporta información al servidor. Para ello habrá que considerar:

- La frecuencia de registro de la posición GPS: Será necesario determinar una frecuencia lo suficientemente buena para poder seguir su recorrido con cierta precisión y evitar un consumo excesivo de batería. La gestión dinámica de esta frecuencia en función de la velocidad del equipo, la batería del dispositivo, el tiempo restante para acabar la prueba y el consumo medio de batería durante su funcionamiento, sería un valor añadido adicional que optimizaría la solución.
- La frecuencia de envío de información al servidor: Es el otro factor determinante del consumo de batería. Enviar la posición al servidor constantemente, drenaría la batería en poco tiempo. Por eso la información de posición debe enviarse en paquetes de datos con una frecuencia establecida que permita tenerla disponible en el servidor en pseudo tiempo real, y no suponga un consumo excesivo de batería.

5.1.13. Gestión de la reproducción en tiempo real

Si bien, en primera instancia, puede parecer que el tratamiento de la reproducción de un evento en diferido o en tiempo real conlleva la misma complejidad, una óptima gestión de la batería como se expone en el punto 5.1.12, sumado a los posibles

problemas y retrasos por falta de cobertura (entre otros) al transmitir la información de los dispositivos durante la carrera, implica una dificultad adicional.

Así de forma simplificada, la reproducción de un evento en diferido, no es más que el envío desde el *backend* a la interfaz Web de toda la secuencia de posiciones temporales de cada equipo para su dibujado secuencial en pantalla sobre el mapa de carrera.

Sin embargo, en la reproducción en tiempo real se presentan una serie de dificultades:

- No se dispondrá información de todos los equipos en cada instante del seguimiento,
- Los equipos no enviarán posición por posición al servidor, si no que lo harán en forma paquetes de información con las últimas posiciones no reportadas
- La información no llegará al servidor desde todos los dispositivos ordenada cronológicamente.

Por lo tanto, será necesario, en base a estas premisas, diseñar una lógica adecuada que permita mostrar un seguimiento lo más fluido posible, con el mínimo número de cortes o 'saltos' de posición de los equipos y, en el caso de que se produzca pérdida de señal de un equipo, permita diferenciar visualmente a los equipos que están *online* y los que no han transmitido su posición.

5.1.14. Asociación de más de un dispositivo por equipo

Al estar un mismo equipo formado por dos o más corredores, cabe la posibilidad de tener más de un dispositivo de seguimiento asociado a un mismo equipo. Esta funcionalidad además de ser un elemento diferenciador del resto de soluciones existentes, que usan dispositivos específicos pero muy costosos que normalmente hace inviable ofrecer más de uno por equipo, brinda una serie de ventajas al funcionamiento del sistema. Estas ventajas se pasan a analizar desde dos planteamientos distintos.

Funcionamiento exclusivo

- Sólo puede haber un dispositivo por equipo reportando información de la posición durante la carrera, que se denomina principal.
- El resto estaría en modo reposo, a la espera de entrar en funcionamiento.
- Ante cualquier problema detectado en el dispositivo principal: baja batería, pérdida prolongada de señal o cortes continuos de la misma, poca precisión del GPS, etc. el servidor iniciará una comunicación con el resto de dispositivos del equipo solicitando información sobre batería, estado de señal, precisión GPS, etc.
- Si uno de los dispositivos reporta condiciones que mejoran el problema detectado en el dispositivo principal, el servidor envía una señal al principal para que pase a modo reposo, asignando el rol de principal al nuevo dispositivo.

Este modo, óptimo para pruebas de larga duración, permite prolongar el tiempo de seguimiento al poder usar la batería de todos los dispositivos del equipo. Además, es posible cuando los miembros de un equipo tienen operadores diferentes, paliar problemas como la pérdida de cobertura por zonas de operadores determinados o ante fallo de funcionamiento del GPS de un dispositivo usar otro que ofrezca mejor precisión.

Funcionamiento simultáneo

- Todos los dispositivos de un mismo equipo, están reportando información al servidor al mismo tiempo.
- El servidor se encarga de recoger esa información, procesarla y filtrarla, pudiendo ser utilizada para tareas como mejorar la precisión de la posición de un equipo o facilitar a la organización información para saber si se cumple medidas del reglamento de carrera como separación máxima permitida entre los corredores de un mismo equipo.

5.2. Análisis: Actores del sistema

Se identifican tres tipos de usuarios principales de la solución [Figura 5.3]:

- ***Race manager***: Usuario registrado en el sistema, administrador de uno o varios eventos, encargado de crear, configurar y administrar sus propios eventos. Es el rol que desempeña los organizadores de eventos.
- ***Supporter*:** Usuario anómino, familiar, amigo de un corredor o aficionado general que utilizará la parte pública del portal Web para seguirlo en tiempo real a posteriori.
- ***Runner*:** Usuario del aplicación móvil, miembro de un equipo participante en un evento que desea activar el seguimiento durante la carrera. Si bien no tiene que ser un usuario registrado en el sistema con el método habitual de usuario y contraseña, debe estar identificado por alguna información única que permita asociar la información enviada por su dispositivo a su equipo.

Se pude contemplar la existencia de un cuarto tipo de usuario que será el **administrador del portal**, encargado de supervisar y gestionar la actividad de los ***Rogaine managers***. Si bien este tipo de usuario no prioritario para alcanzar el objetivo deseado.

Por último, se identifican dos actores necesarios asociados a procesos: uno en el servidor y otro en cada *smartphone* que serán los encargados de comunicarse entre ellos, si intervención de un usuario, para transmitir y procesar la información de seguimiento en tiempo real del evento [Figura 5.4].

5.3. Análisis: Principales casos de uso

Definidos los actores del sistema y en función del análisis básico y de retos más importantes realizado, se plasma en diagramas de casos de uso los escenarios principales del proyecto.

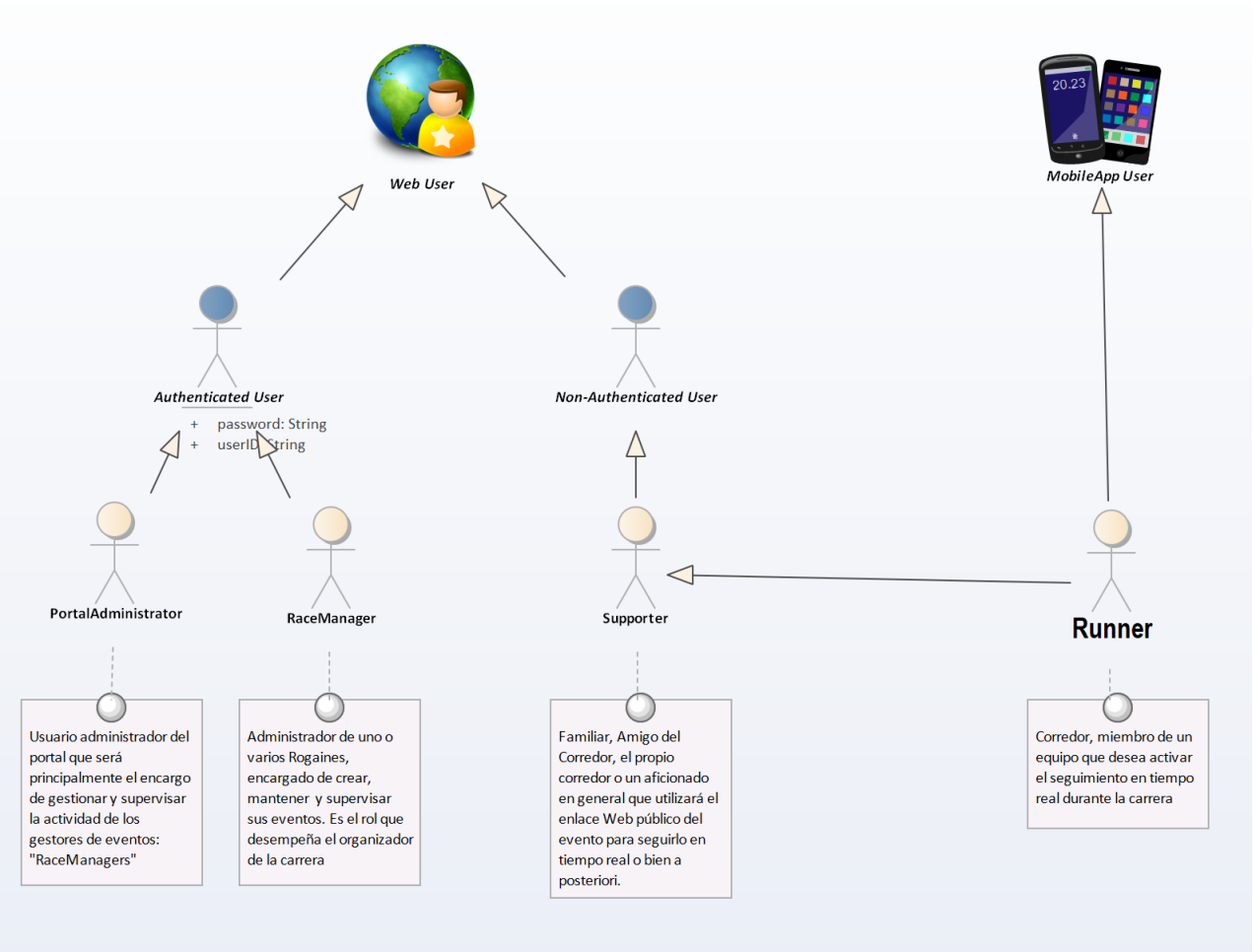


Figura 5.3: Usuarios del sistema

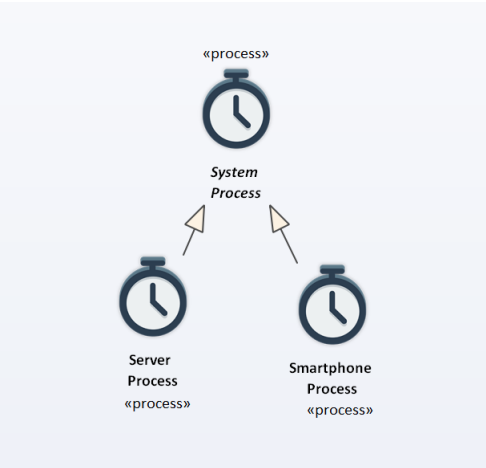


Figura 5.4: Actores no asociados a usuarios

5.3.1. Visión general

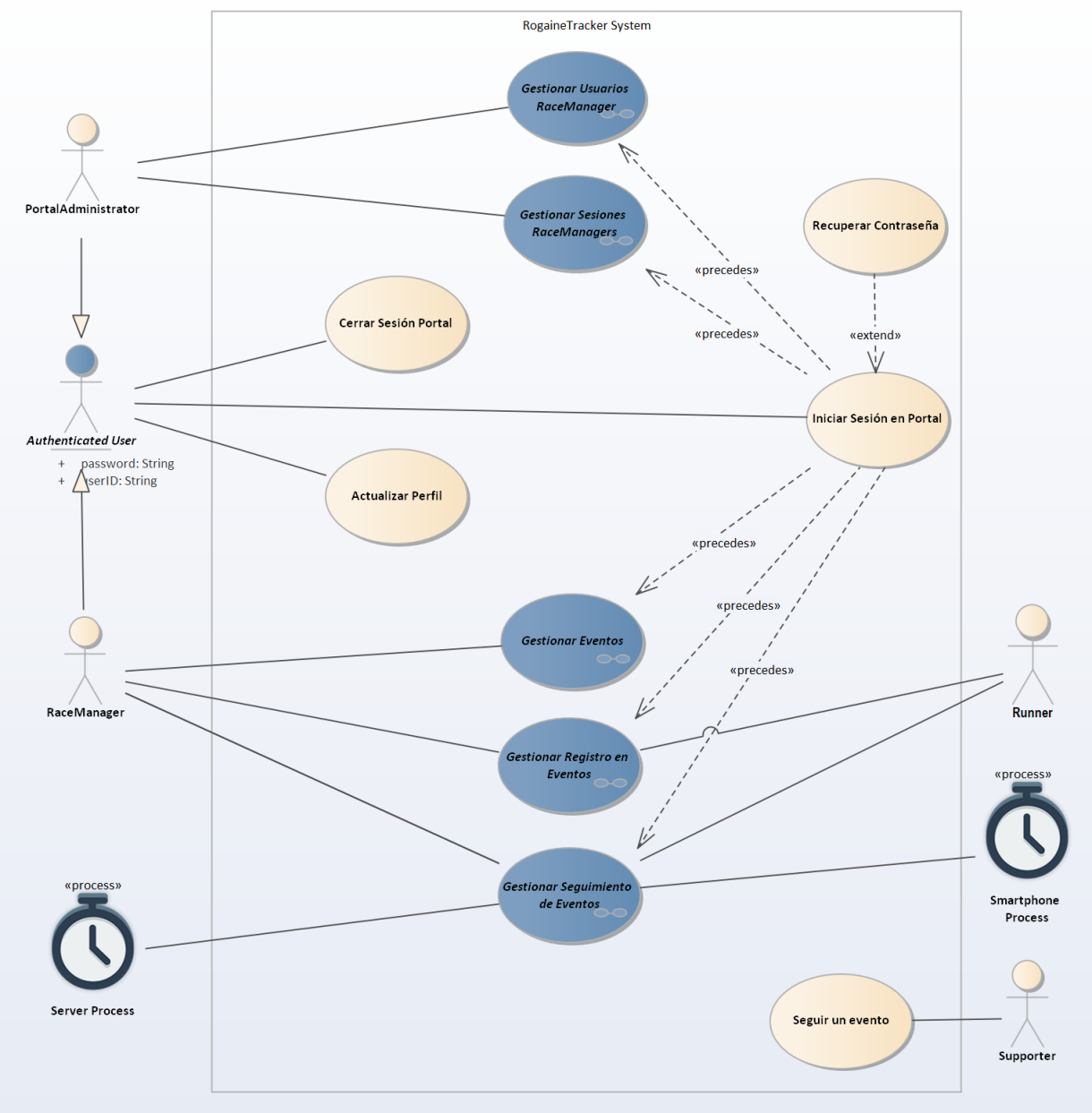


Figura 5.5: Casos de uso. Visión general

5.3.2. Administración del portal

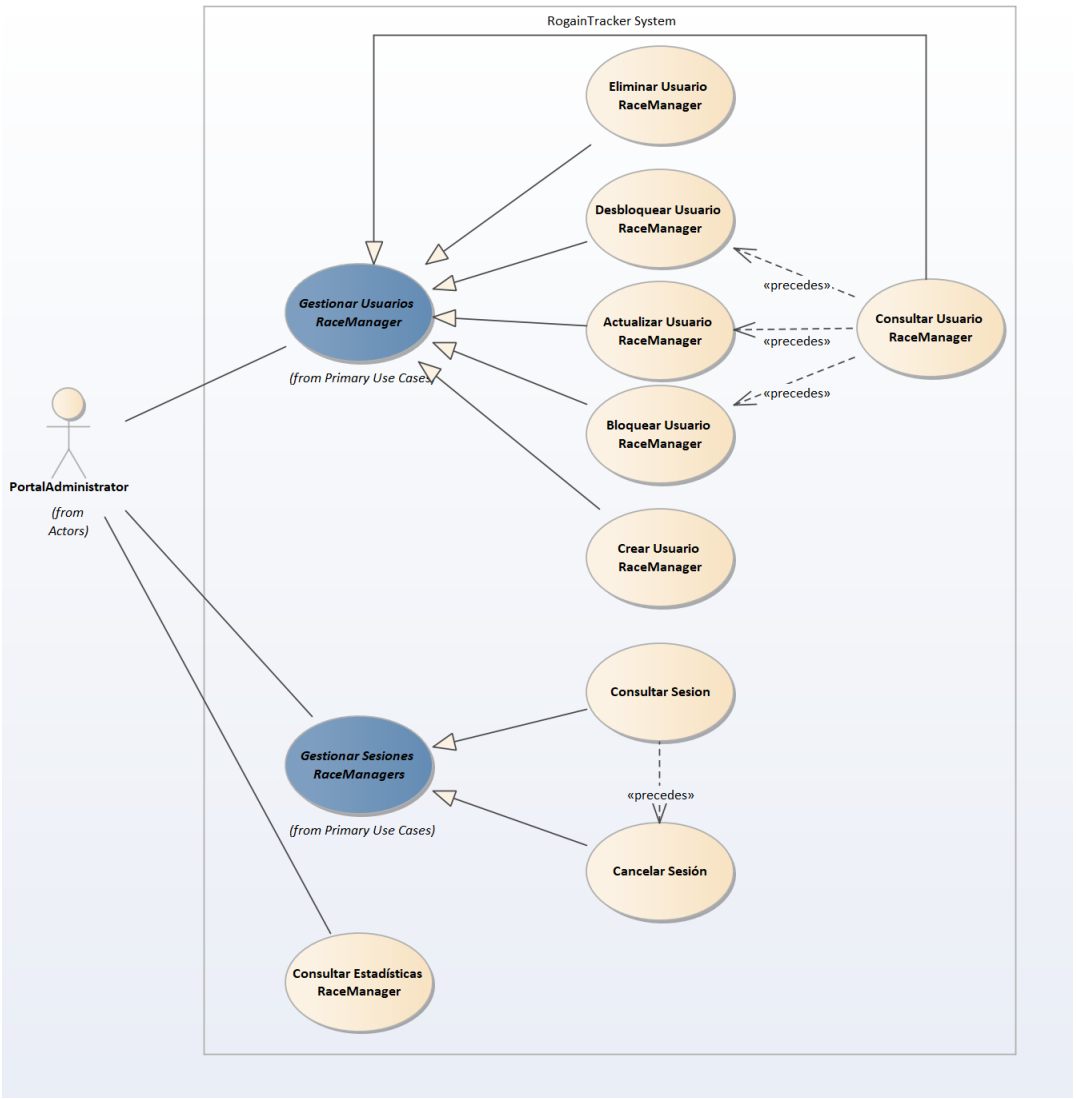


Figura 5.6: Casos de uso: Administración del portal

5.3.3. Gestión de eventos

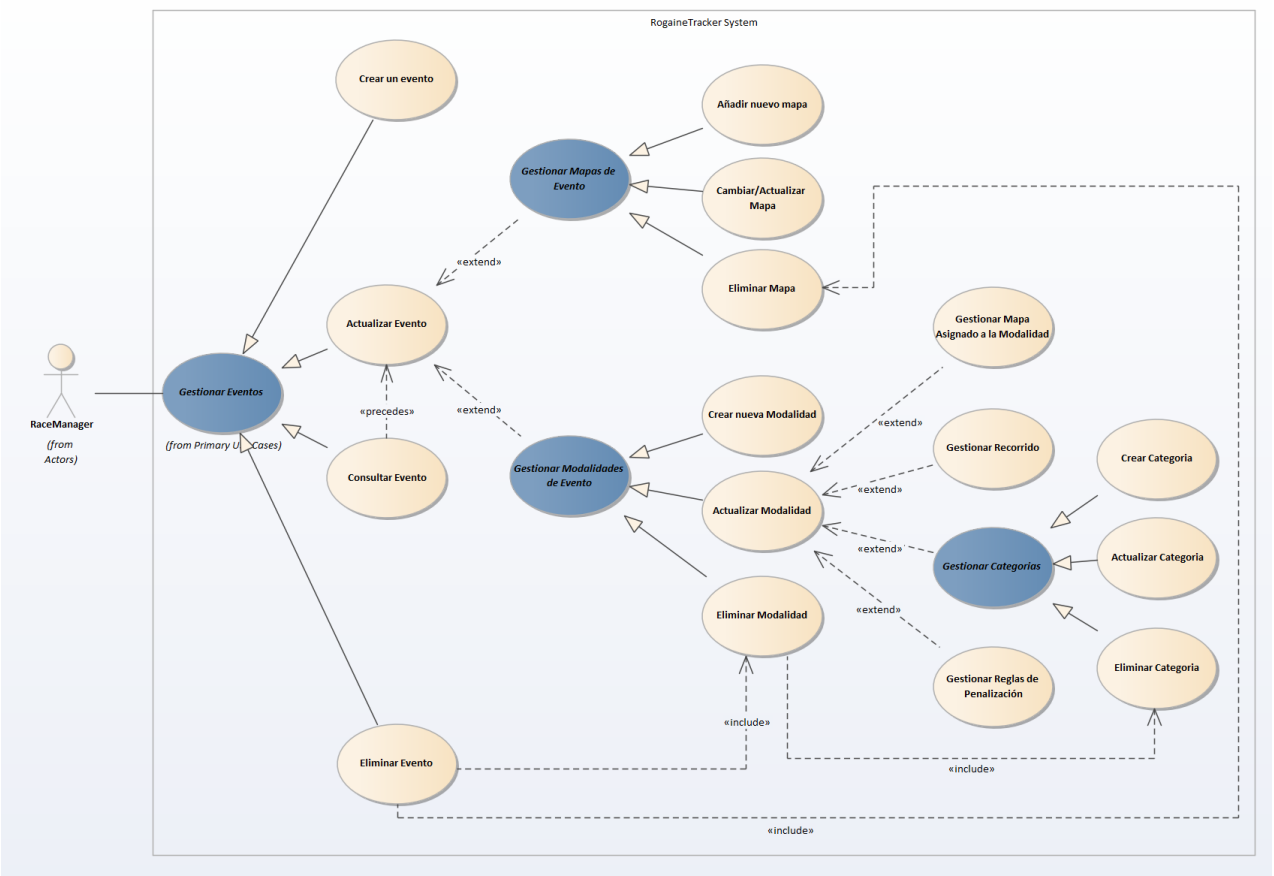


Figura 5.7: Casos de uso: Gestión de eventos

5.3.4. Gestionar registro en eventos de los participantes

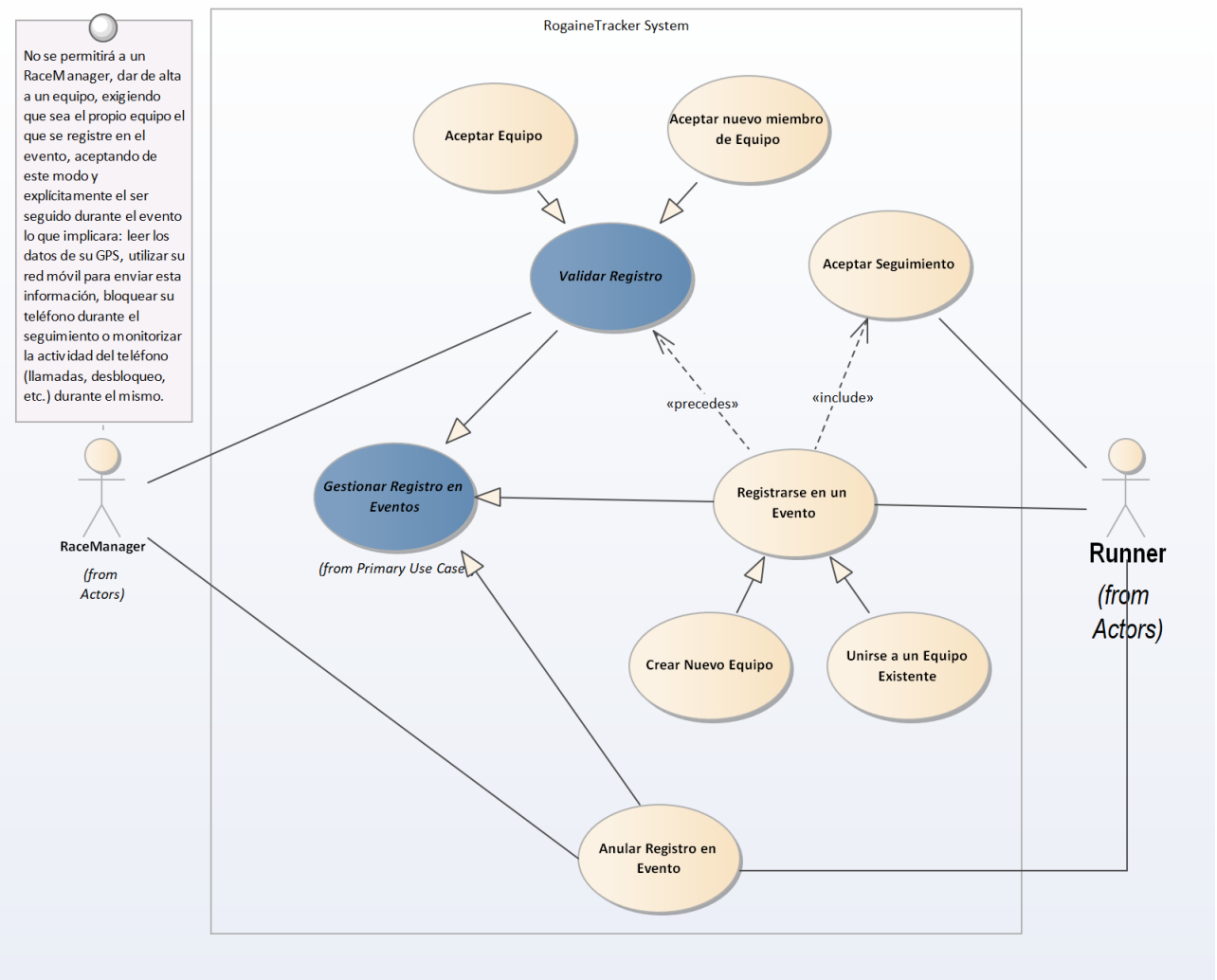


Figura 5.8: Casos de uso: Gestión de registro en eventos

5.3.5. Gestionar seguimiento de evento

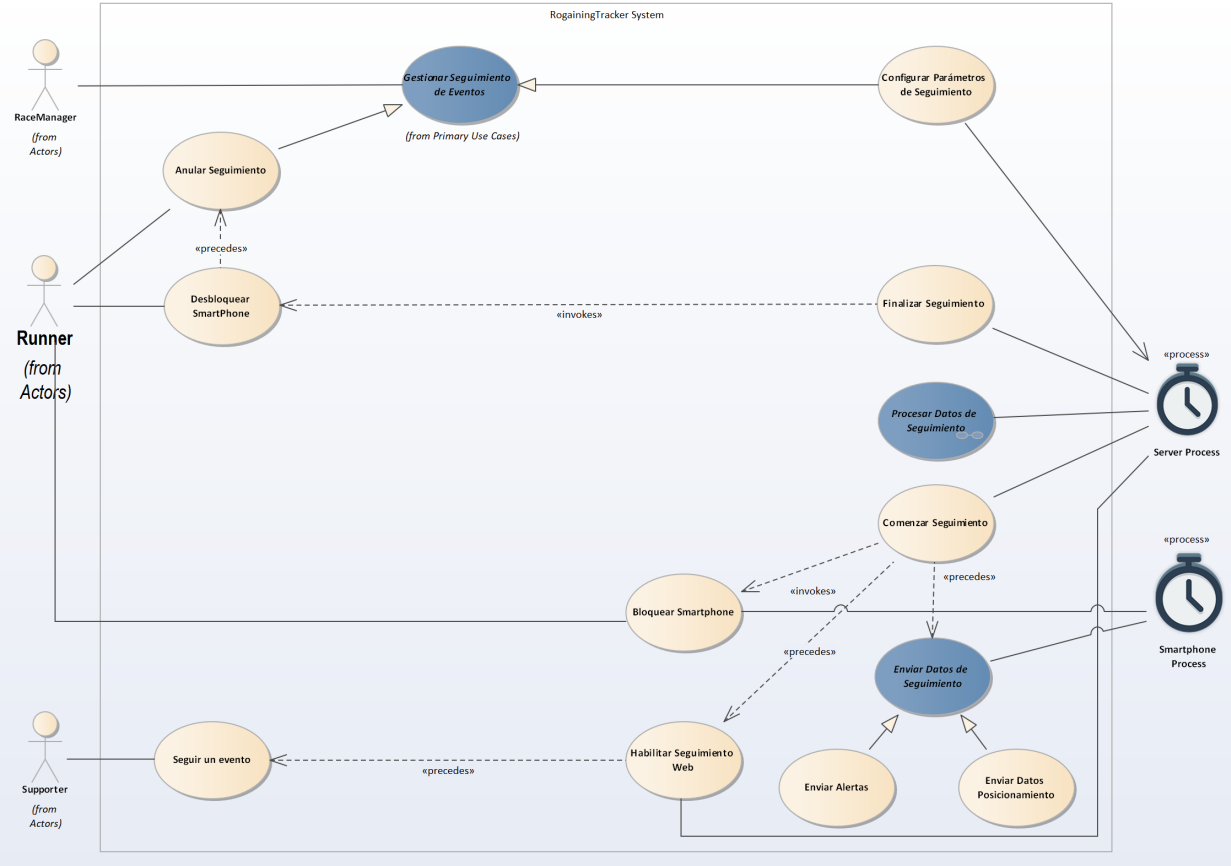


Figura 5.9: Casos de uso: Gestión de seguimiento de evento

Procesar datos de seguimiento

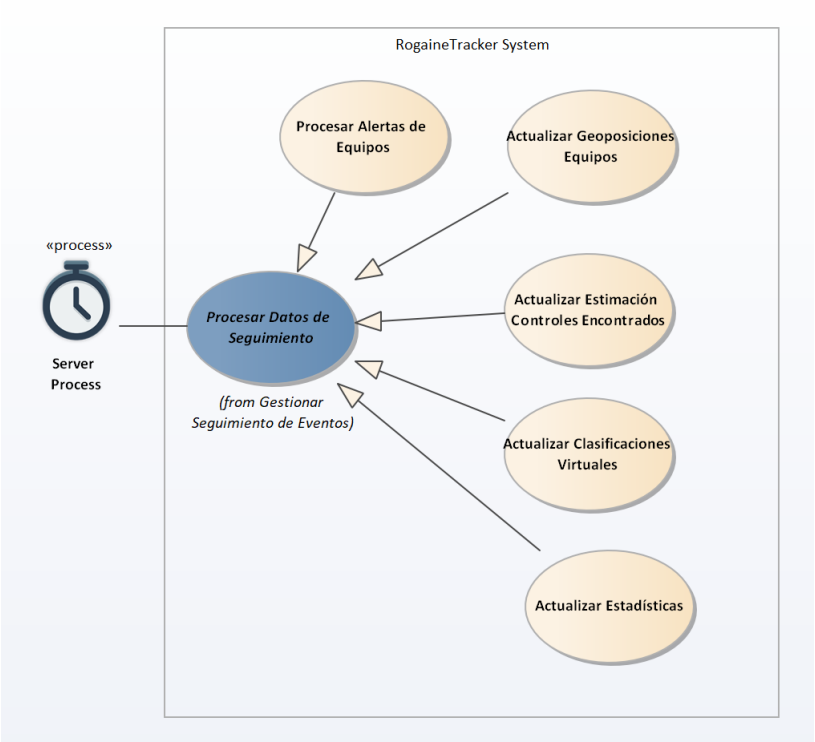


Figura 5.10: Casos de uso: Procesar datos de seguimiento

5.4. Diseño: Base de datos

Un correcto diseño de la base de datos que permita dar cabida a todos los requisitos de multiplicidad y cardinalidad en base al análisis realizado, es fundamental para el éxito de la solución. El sistema requiere de dos bases de datos diferenciadas:

- Una en el lado del servidor que será la base de datos principal donde se almacene y persista toda la información asociada a la solución.
- Otra para la aplicación instalada en los dispositivos móviles de los corredores que permitirá almacenar configuración, eventos registrados y servirá de almacenamiento temporal para los datos a enviar al servidor durante el seguimiento.

En ambos casos, tal como se indica en el apartado 3.1.2, se ha optado por un sistema de bases de datos relacional para el diseño de las mismas.

5.4.1. Base de datos principal

Del análisis realizado en los puntos anteriores, se puede identificar dos etapas diferenciadas de almacenamiento de información y recogida de datos en el uso de la aplicación.:

- Una primera fase de gestión del evento, donde este se configura por parte del organizador y se registran los distintos equipos.
- Una segunda etapa que se da durante el seguimiento del mismo, tanto en tiempo real como en diferido.

Partiendo de esta premisa, si bien ambas fases de recogida de información coexistirán en la misma base de datos, se mostrará la estructura de tablas que soportará cada una de ellas por separado.

Gestión de eventos y registro de equipos

En los retos analizados en las secciones 5.1.1 y 5.1.2 se exponen prácticamente todos los requisitos de información necesarios para la primera fase. En la [Figura 5.11], se muestra el esquema de tablas, claves primarias y secundarias y tipos de relaciones principales entre ellas, para soportar estas necesidades de información.

A continuación se explica brevemente la función e información almacenada en cada tabla:

- **t001RogaineManager:** Es la tabla que contiene la información básica de los usuarios registrados y contra la que se comprueba las credenciales de autenticación. Almacena información básica del usuario como nombre, apellidos y correo electrónico.
- **t001Event:** Persiste la información principal de cada evento: nombre, organizador, web, fecha, ubicación, logo, etc. Además almacena una serie de modificadores *booleanos* que permiten controlar si el evento es visible en la lista de eventos disponibles en el portal y en la aplicación móvil o si está abierto el registro en el mismo. Cada evento esta asociado a un único *RogaineMana-*

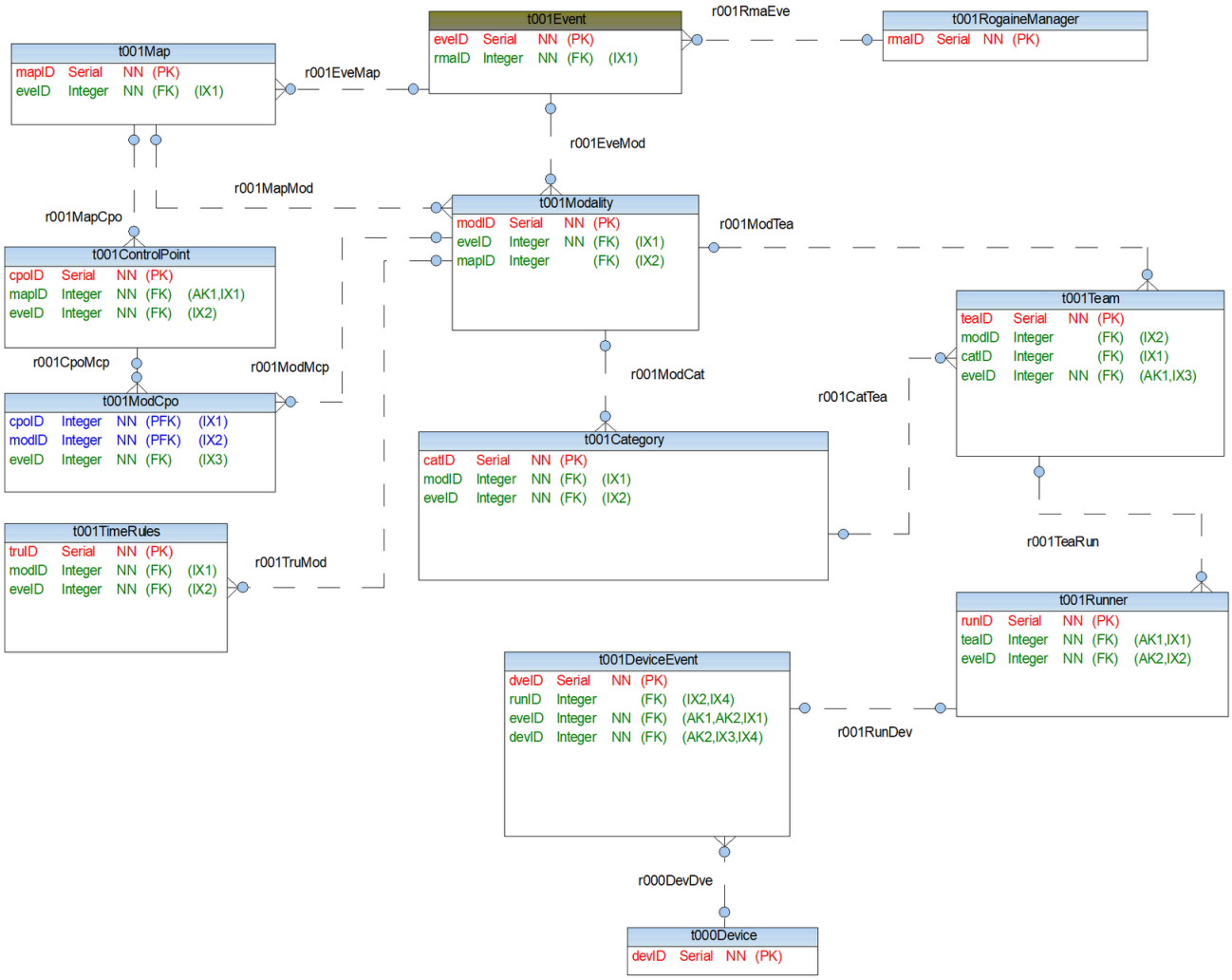


Figura 5.11: Estructura de tablas para gestión de eventos y registro de equipos

ger por lo que de este sencillo modo se puede controlar que cada organizador pueda gestionar sólo sus eventos.

- **t001Map:** Registra información de cada mapa: nombre, ubicación en el servidor de la imagen de mapa correspondiente, escala y equidistancia, propietario, etc. También guarda la información de georreferenciación del propio mapa, almacenando en dos columnas las coordenadas geográficas de la esquina superior izquierda y de la esquina inferior derecha del mismo. Cada mapa puede estar incluido en un único evento.

- **t001ControlPoint:** Almacena la ubicación y el código de todos los controles asociados a un determinado mapa.

- **t001Modality:** La información sobre cada una de las modalidades de un evento se encuentra en esta tabla: nombre, duración, hora de comienzo de la modalidad, hora de habilitación de seguimiento público del evento, mapa asociado y localización del punto de salida de la misma. Además se almacenan dos URL:

  - Una para el seguimiento público que será accesible a cualquier persona que lo posea y disponible únicamente a partir de la hora de configurada de visibilidad pública del evento.
  - Otra de seguimiento privado, sólo accesible para el gestor del evento en cualquier momento antes, durante y después del mismo.

- **t001ModCpo:** Guarda información sobre los controles incluidos en cada modalidad y la puntuación asignada a los mismos.

- **t001Category:** Nombre de cada categoría y modalidad en la que se incluye. Si bien se ha podido incluir información sobre restricciones de edad y sexo en cada categoría para utilizarla con fines de validación de inscripción de los equipos, en esta primera aproximación se ha decido no incluirlo ya que será el organizador el que se encargue de incluir las inscripciones en el sistema y asignar la categoría a cada equipo.

- **t001Team:** Información básica de cada equipo como el nombre, el dorsal, un correo de contacto, la nacionalidad y club al que pertenecen. Cada equipo está asociado a una categoría y modalidad de un evento.

- **t001Runners:** Nombre, apellidos, prioridad de seguimiento (pensado para la implementación de la funcionalidad que permite la asignación de varios dispositivos de seguimiento por equipo) y un número de teléfono. Este número de teléfono debe ser el asociado al *smartphone* que se use para el seguimiento, ya que será el dato que se use para la validación del registro. Un corredor pertenece a un único equipo y un equipo puede tener varios corredores. No se ha implementado restricción a nivel de base de datos para el número de corredores permitidos para cada equipo, dejando el control de esa particularidad al lado del servidor.

- **t000Device:** Almacena la información de cada uno de los dispositivos que han solicitado, desde la aplicación móvil, registrarse a un evento. Cada dispositivo tendrá una única entrada en esta tabla. Un dispositivo se identifica por: un Id de dispositivo, un token de FCM(Firebase Cloud Messaging) que se introduce más adelante y un número de teléfono.

- **t001DeviceEvent:** Asocia los dispositivos de la tabla anterior a un corredor de un evento dado. Cada dispositivo puede estar asociado uno o varios corredores de eventos distintos no pudiendo estar un mismo dispositivo asociado a dos corredores del mismo evento. Guarda información como el estado del mismo, si está *online*, si está reportando la posición o si está deshabilitado.

Por otra parte, en el diseño de la figura 5.11, si bien no se muestra todas las relaciones con la tabla *T001Evento* por motivos de claridad, si se analiza las claves secundarias de todas las entidades se observa como en la mayoría de ellas se ha establecido una relación 1 a N con la misma. Esta decisión, tomada por motivos de optimización y rendimiento en la gestión de la información al reducir el número de cruces (*joins)* necesarios entre tablas para llegar desde la tabla *T001Evento* a cualquiera de sus tablas hijos, implica la necesidad de controlar explícitamente, en el momento de la implementación de la parte de la aplicación encargada de gestionar el acceso a

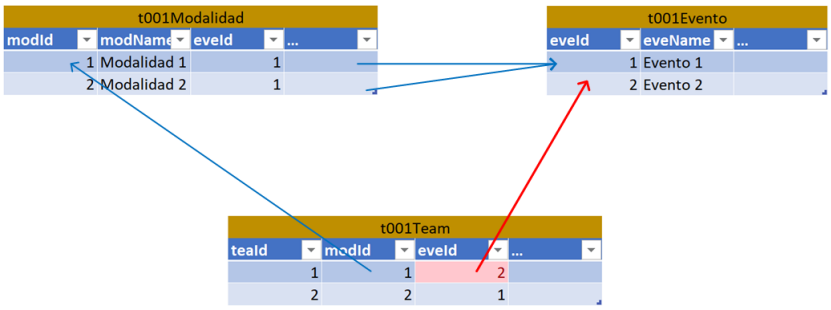


Figura 5.12: Ejemplo de incoherencia de información posible en la base de datos diseñada

los datos, la coherencia de la información almacenada. Si no se hiciese, se podrían dar casos como el mostrado en la figura 5.12 en la que un equipo registrado en una modalidad de un evento y apuntando a otro en el que no está incluida dicha modalidad. Esto mismo ocurre en otras relaciones y tablas de la base de datos por el mismo motivo.

Seguimiento de los equipos

Del resto de retos y casos de uso analizados en los apartados anteriores se deducen la necesidades de persistencia de información para soportar el seguimiento de los equipos. Así, en la [Figura 5.13], se puede ver el diagrama de las tablas y relaciones de la base de datos principal implicadas en la gestión de esta tarea:

- **t002EventTrackLog:** Esta tabla almacena en bruto la información transmitida por los dispositivos del evento: identificador del dispositivo, equipo al que pertenece, *timestamp*, posición, altitud, velocidad y precisión. Además tiene tres columnas para almacenar, asociado a cada dato, la distancia, desnivel positivo y desnivel negativo por el remitente hasta ese momento. Esta información deberá ser calculada por el servidor o enviada por los *smartphones* con el resto de datos. En una primera aproximación, en la que los datos no son procesados y mejorados por el servidor, la información de esta tabla será suficiente para mostrar la posición del equipo.

- **t002ProcesedTrackLog:** En una segunda versión, aquí se almacenaría la

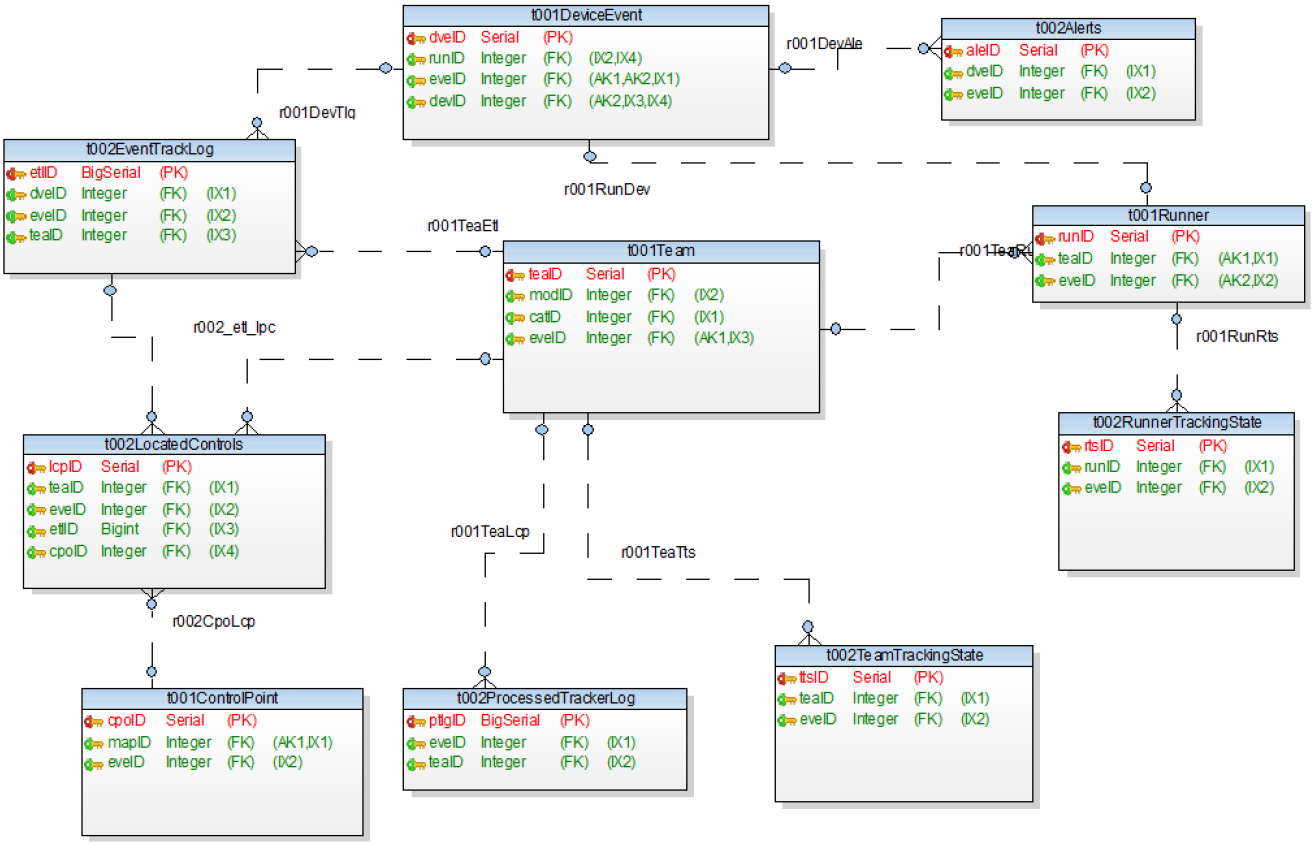


Figura 5.13: Estructura de tablas para el seguimiento del evento

información procesada y mejorada por el servidor de los datos transmitidos por cada equipo que permitiría, entre otros: el suavizado del *track* mediante alguna técnica de interpolación, eliminar ruido obviando datos con mucho margen de error o la mejora de la precisión mediante la utilización de los datos de varios miembros del equipo como se proponer en el escenario planteado en el apartado 5.1.14.

- **t002LocatedControls:** Utilizando la información de cualquiera de las tablas anteriores, registra la relación de controles que el sistema ha estimado que el equipo ha encontrado, según las ubicaciones reportadas. Guarda la posición a partir de la cuál se ha dado por localizado, el tiempo de carrera, la puntuación conseguida y los puntos totales acumulados hasta el momento. Además, contempla la posibilidad de que el organizador decida asignar de forma manual un control al equipo e incluye para ello una columna que indica si el control ha sido asignado manualmente.

- **t002TeamTrackingState y t002RunnerTrackingState:** Sólo utilizadas durante el seguimiento en tiempo real, guarda información sobre si el equipo y el corredor respectivamente, está transmitiendo información en un momento dado.

- **t002Alerts:** Pensada para almacenar las alertas transmitidas por la aplicación al servidor móvil como baja batería, mala precisión prolongada del GPS, acciones no permitidas con el dispositivo por parte del corredor, etc. No es utilizada en esta primera aproximación.

5.4.2. Base de datos para dispositivos móviles

Las necesidades de persistencia de información en la aplicación móvil son simples, siendo la estructura de tablas necesaria muy simple como se puede observar en la [Figura 5.14]:

- **s001Event:** Almacena la información básica de todos los eventos marcados como visibles públicamente por los organizadores, controlando si están o no

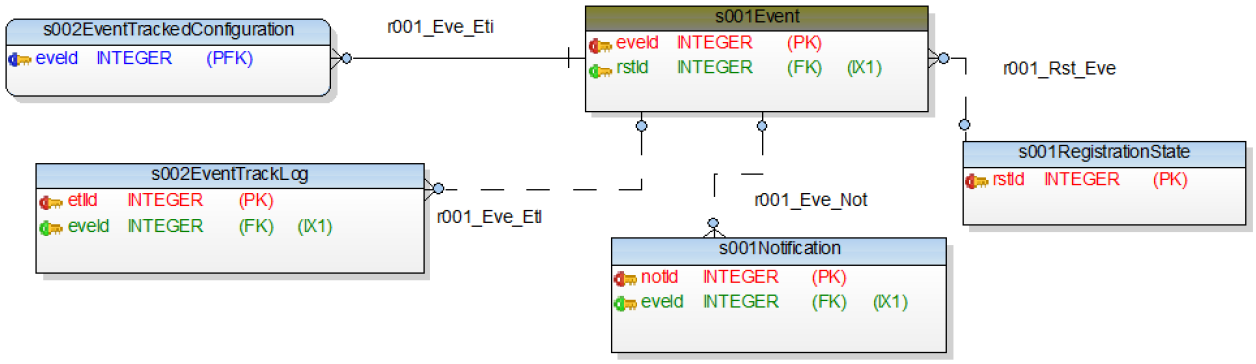


Figura 5.14: Estructura de tablas aplicación móvil

disponibles para el registro y el estado de registro del corredor en dicho evento: no registrado, registrado pendiente de validación, etc.

- **s001RegistrationState:** Almacena una lista de los posibles estados de registro en los que se puede encontrar el dispositivo.
- **s002EventTrackedConfiguration:** Contiene la configuración de seguimiento de los eventos en los que el usuario se ha registrado con el dispositivo como: modalidad y categoría asignada, comienzo y final del seguimiento, si ya ha terminado o si ha sido cancelado.
- **s002EventTrackLog:** Tabla donde se almacena los datos de posicionamiento capturados por el sensor GPS y que se transmitirán al servidor. Cada dato incluirá información sobre posición, velocidad, altitud y precisión, registrando si ha sido sincronizado con el servidor y cuando.
- **s001Notification:** Donde se guarda las comunicaciones recibidas desde el servidor en relación a los eventos en los que se ha registrado como: confirmación de registro, modificación de categoría o cambio en la hora de comienzo o final de la modalidad en la que está registrado. Permite distinguir las notificaciones que han sido leídas y las que no.

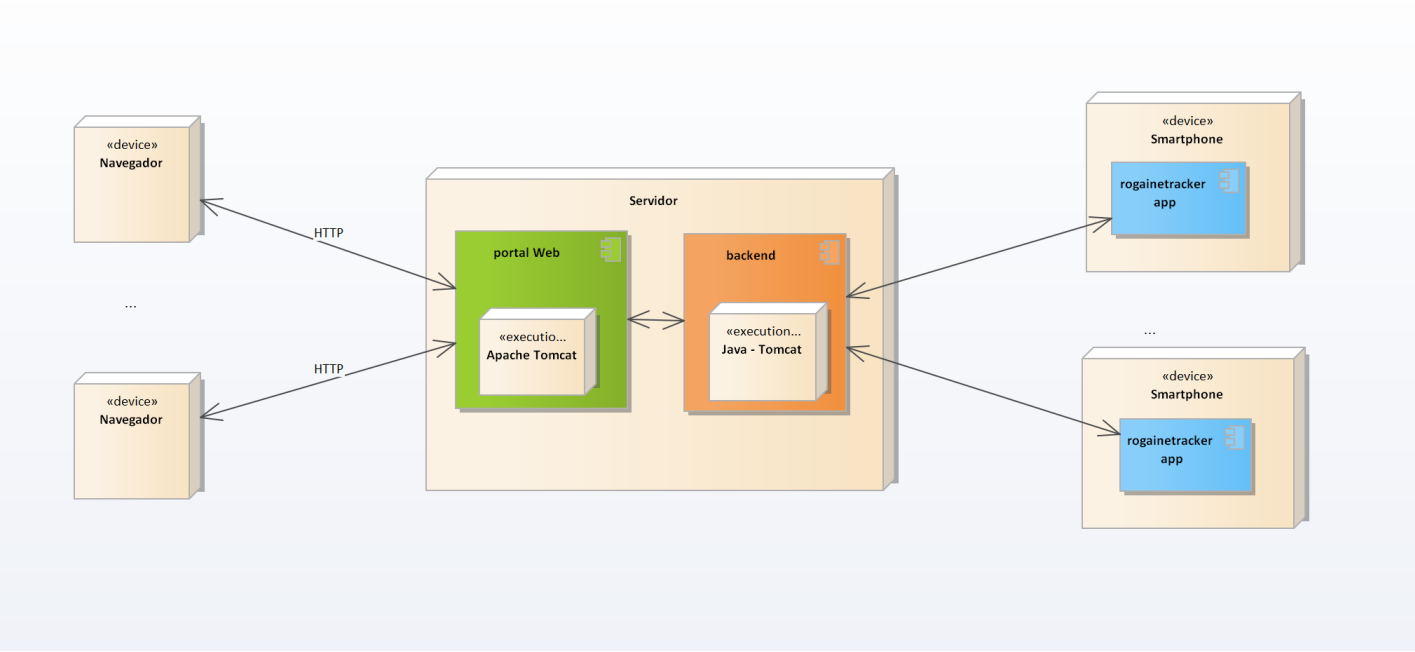


Figura 5.15: Arquitectura básica del sistema

5.5. Diseño: Arquitectura del sistema

Como se ha ido introduciendo en el resto de secciones, la solución se plantea en base a tres componentes [Figura 5.15]:

- Un servicio Web que es el punto de unión entre el resto de componentes y el encargado de supervisar y gestionar el acceso a los datos.
- Un portal Web para gestión, configuración y seguimiento de los eventos.
- Una aplicación para dispositivos móviles cuya principal misión es la de capturar las posiciones de los participantes y transmitirlas.

En la figura 5.16 se muestra las principales tecnologías y librerías que se ha decidido utilizar en cada componente de la solución.

Servicio Web

Se decide diseñar un servicio web *RESTFul,* es decir, basado en la arquitectura REST [26]. La filosofía REST basa su funcionamiento en 4 principios:

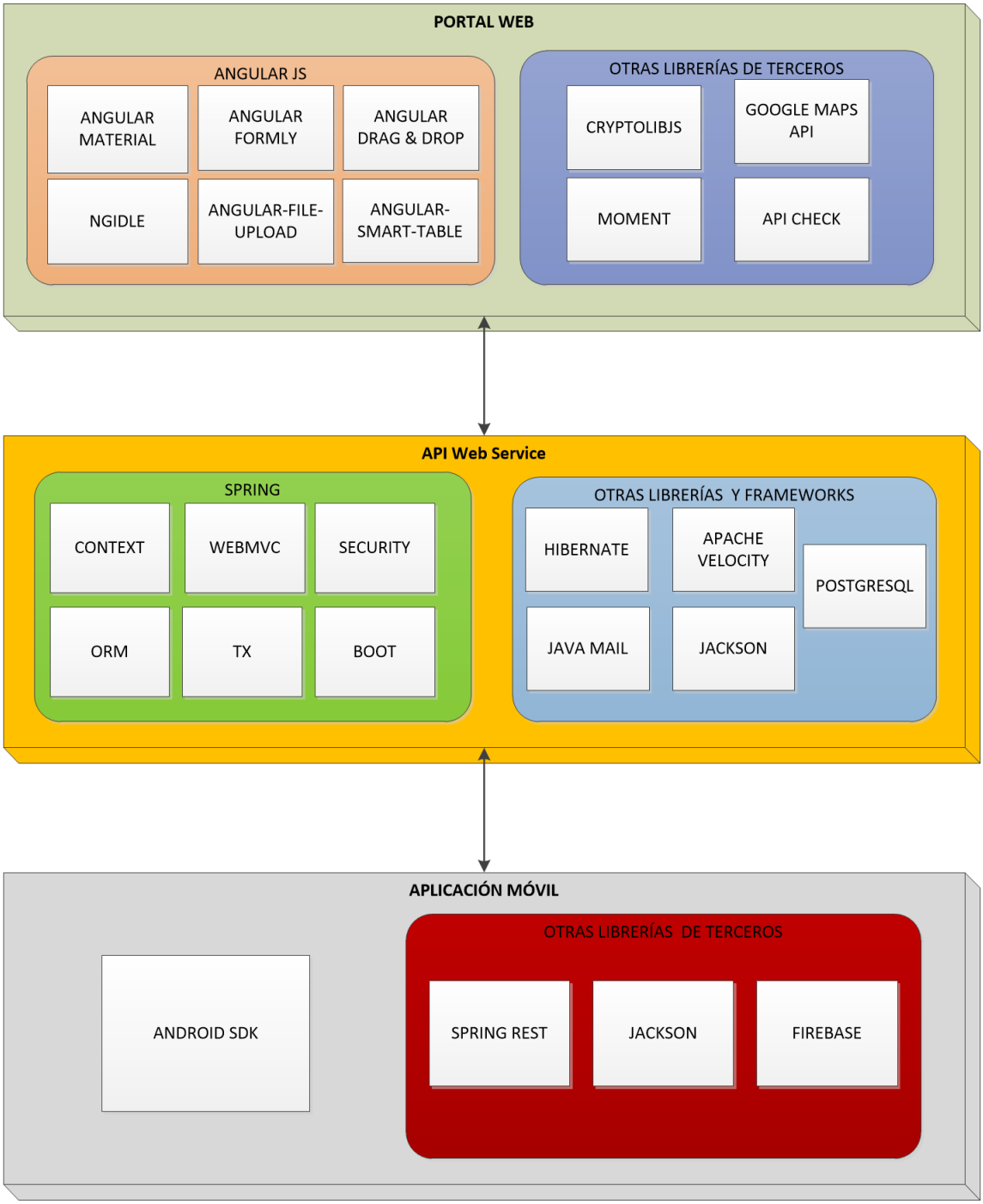


Figura 5.16: Resumen de tecnologías y librería del proyecto

- Utiliza los métodos HTTP de manera explícita y su semántica es siempre la misma. Así, para los más importantes:
  - GET: Obtiene un recurso.
  - POST: Crea un recurso.
  - PUT: Actualiza el estado de un recurso.
  - DELETE: Elimina el recurso
- No mantiene estados *(stateless),* por lo que debe ser el cliente al enviar las peticiones el que incluya todos los datos necesarios para que se pueda gestionar la misma, siendo el servicio Web únicamente encargado de combinarlos para ofrecer la respuesta esperada. Esto incluye, los datos de autentificación, la sesión o los parámetros de la solicitud a efectuar.
- Usa una sintaxis universal para identificar los recursos, siendo estos identificables unívocamente a través de su URI. Las URIs de acceso a los recursos se exponen en forma de directorios, con una estructura jerárquica.
- Transfiere los datos en formato XML o JSON entre otros.

Para la implementación del *backend* se elige utilizar el lenguaje de programación Java mediante dos *frameworks* principales:

- **Spring**: Para la implementación de la API. Mediante las características expuestas en 3.1.3.1, permite, de forma simple y mediante anotaciones y algunas clases de configuración, el desarrollo de la lógica para creación de URIs de acceso a recursos, el mapeado de las respuestas, la autenticación de usuarios y la gestión de los filtros de seguridad de cada uno de los *endpoints* de la API.
- **Hibernate:** Para el acceso y manipulación a los datos. Este *framework* facilita el mapeo de bases de datos relacionales a clases Java mediante anotaciones y unos sencillos archivos de configuración. Permite abstraerse del motor de base de datos utilizado, pudiendo de forma simple, cambiar de motor de base de datos sin tener que modificar la lógica del programa.

Portal Web

Se plantea la realización de un portal con una interfaz sencilla y fluida en el que haya dos partes diferenciadas:

- Parte pública, compuesta por:
  - Con un *dashboard* como página principal donde se liste la relación de próximos eventos para su seguimiento, pudiendo visualizar el detalle de cada uno: fechas, categorías, etc. así como su enlace de seguiento público.
  - Una pantalla de seguimiento donde el principal elemento sea el mapa de carrera y en la que se pueda seguir el evento seleccionado según los parámetros de visibilidad establecidos para el mismo. Además se plantea incluir en esta pantalla un espacio para logos de patrocinadores que permitan dar visibilidad a estos últimos.
- Parte restringida, compuesta por:
  - Con un panel de control que permita la gestión y configuración de todos los eventos de un organizador dado.
  - Una pantalla de seguimiento del evento solo accesible al organizador con el que poder hacer pruebas previas al evento, visualizar el aspecto del mapa u otros factores.

Para la implementación de esta parte del sistema, se decide utilizar Javascript a través de su *framework* AngularJS. Este *framework*, según lo indicado en el apartado 3.1.3.2, permite desarrollar aplicaciones Web SPA en modelo vista-controlador, con componentes reutilizables y contenido dinámico, liberando al programador de gran parte de las tareas de manipulación del DOM y permitiendo centrarse en la lógica de la solución.

En la parte visual, se decide utilizar el patrón de diseño Material desarrolado por Google [27], por ser visualmente atractivo y dar a la aplicación un aspecto moderno. AngularJS, permite añadir un conjunto de librerías de componentes que siguen este patrón y que incluye tablas, formularios o botones entre otros.

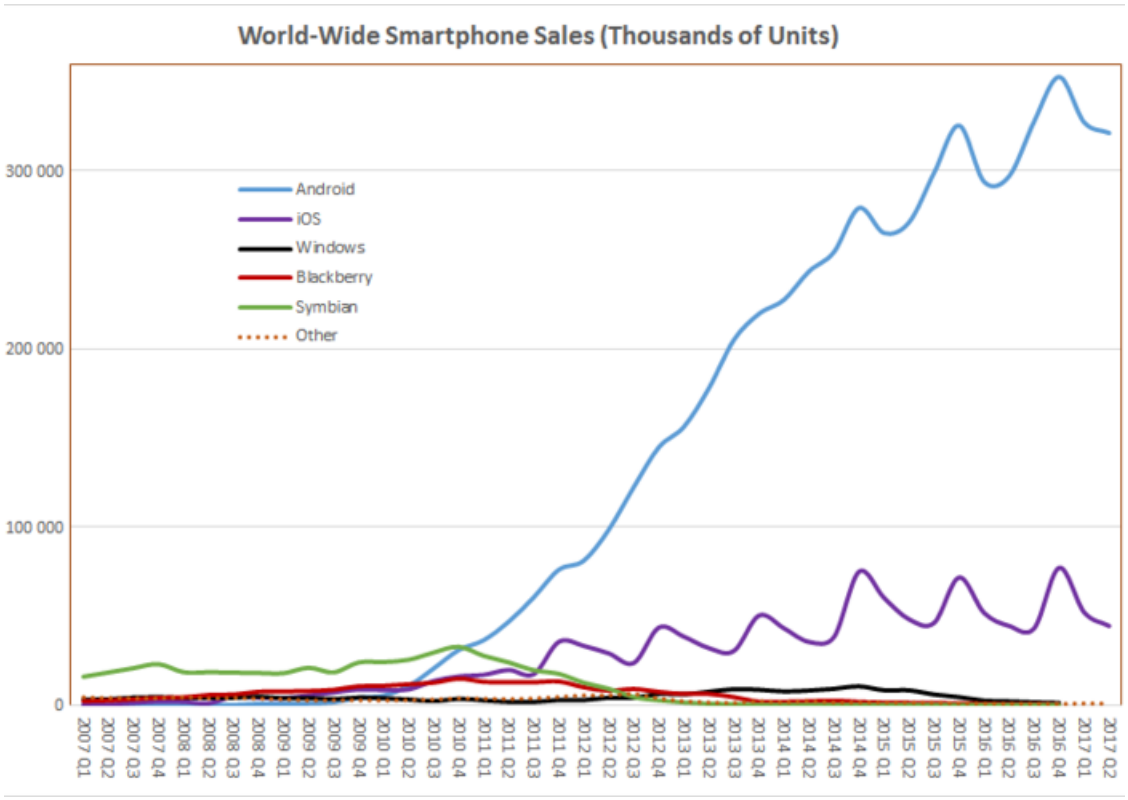


Figura 5.17: Distribución de sistema operativo en dispositivos móviles [28]

Aplicación móvil

Tratando de alcanzar un mayor número de destinatarios iniciales, se decide desarrollar la solución para *smartphones* con sistema operativo Android, ya que es el sistema operativo para dispositivos móviles más extendido con una amplia diferencia [Figura 5.17].

Por otra parte, en el momento de comienzo de desarrollo de la aplicación móvil, junio 2016, la versión 15 de la API de Android SDK que corresponde con la versión ICECREAM SANDWICH (4.0.3) del sistema operativo era la que ofrecía compatibilidad con más del 95 % de los dispositivos Android existentes [Figura 5.18], por lo que se decide establecer esta como versión mínima compatible con la aplicación. El desarrollo de la misma se va actualizando a lo largo del desarrollo acabando por ser la versión objetivo la 6.0 (API 23 del SDK de Android) pero manteniendo la compatibilidad con la versión antes indicada.

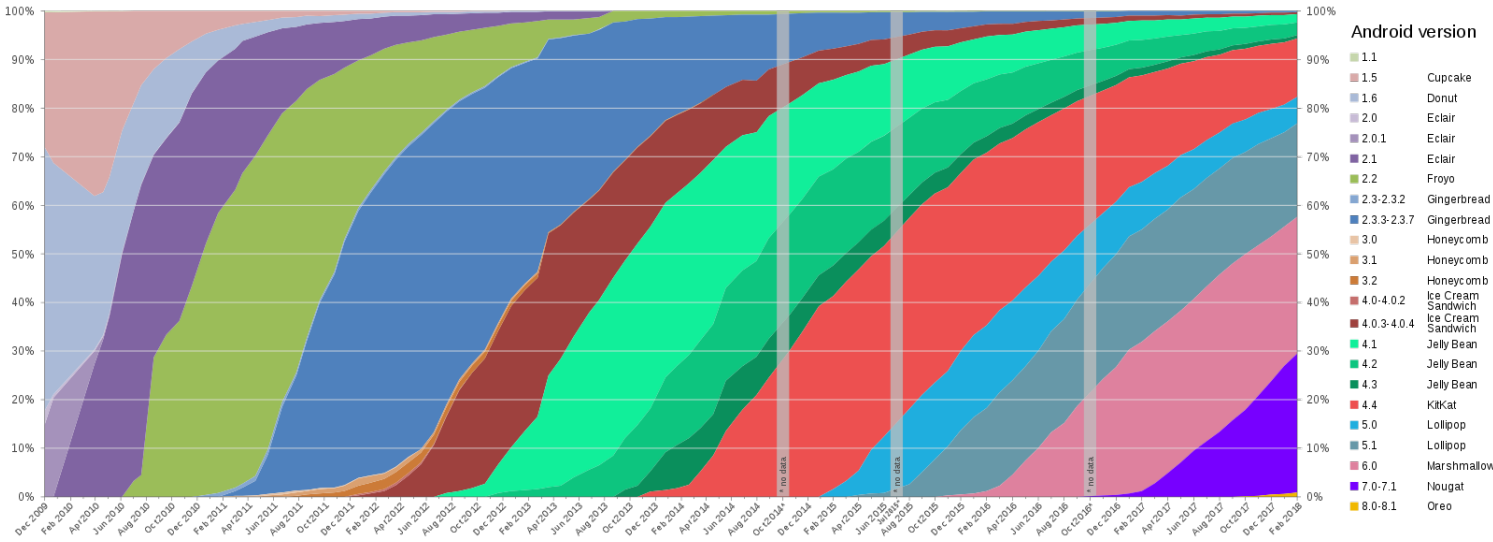


Figura 5.18: Distribución histórica de versiones del S.O. Android [29]

5.6. Diseño: Los retos más importantes

Aunque en el punto anterior, mediante el diseño de las base de datos, algunos de los retos planteados en la parte de análisis del proyecto quedan resueltos, otros requieren tomar otras decisiones adicionales a la estructura de datos que dará soporte al sistema para cerrar su solución. Dichas decisiones de diseño, son las que se pasa a exponer en este apartado.

5.6.1. Restricción de accesos a parte de la aplicación

Para proteger la parte privada del portal Web se diseña un mecanismo de registro abierto estándar en el que cualquier organizador interesado puede solicitar crear un usuario realizando los siguientes pasos:

1. Introducir dirección de correo electrónico con la que registrarse y contraseña deseada.

2. Al realizar la solicitud recibe un correo con la URL de activación de la cuenta en la dirección especificada.

3. Al seguir en el enlace, la cuenta queda activada de forma automática.

Una vez registrado el usuario, este tiene total acceso a crear y configurar sus propios eventos, sin posibilidad de ver el resto de eventos de otros organizadores.

La gestión de accesos se debe gestionar desde el *backend* que al ser *stateless* se debe enviar los datos de autenticación en cada llamada a la API que se realice.

5.6.2. Carga de mapa georreferenciado y dibujado sobre mapa de objetos en función del zoom/escala

Para dar respuesta a lo expuesto en los apartados 5.1.3 y 5.1.4, se decide utilizar, en el portal Web, la API de Google Maps que permite integrar, de forma relativamente sencilla el servidor de aplicaciones de mapas Google Maps en cualquier aplicación [25]. Entre las funcionalidades que ofrece la API, está la posibilidad de sobreimpresionar una imagen en una posición concreta del mapa con unas dimensiones definidas, modificar la opacidad de dicha imagen, dibujar polígonos, líneas y otros elementos sobre el mapa, mostrar el relieve, alejar o acercar el mapa, centrar el mapa en un punto dado o elegir el tipo de mapa a utilizar como capa base: satélite, híbrido, callejero, etc.

Además ofrece desencadenadores de eventos que permiten controlar los cambios de estado de múltiples propiedades del mapa y reaccionar frente a ellos. Esto, por ejemplo, facilita redimensionar los controles y otros elementos dibujados ante un cambio de zoom o centrar el mapa en la posición de un equipo dado a medida que se va desplazando.

5.6.3. Control de hora de comienzo de visibilidad pública de seguimiento

Se ha añadido en la tabla correspondiente de la base de datos, una columna asociada a cada modalidad para indicar la hora de comienzo de visibilidad pública de seguimiento del evento que será establecida por el organizador en el momento de configuración de la prueba, no teniendo que preocuparse posteriormente. Se propone que cuando un seguidor trate de acceder la URL de seguimiento público, si dicho acceso se realiza antes de la hora configurada, se mostrará un reloj con una cuenta

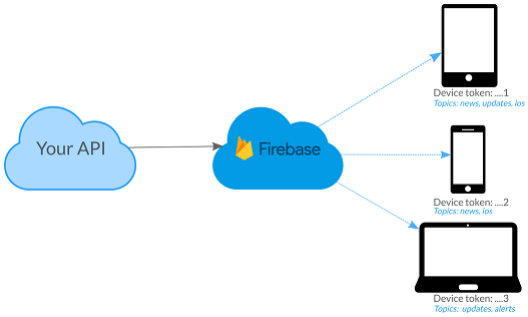


Figura 5.19: Funcionamiento Firebase Cloud Messaging

atrás indicando el tiempo que queda para poder ver el evento en vivo. Si lo hace después de la hora configurada, podrá seguir el evento.

La hora configurada será la hora local de la salida según la zona horaria donde se celebre el evento. La solución deberá tener en cuenta este extremo para gestionar accesos al sistema desde lugares en distintos husos horarios.

5.6.4. Comunicar los móviles con la aplicación y viceversa

Para dar solución al reto planteado en la comunicación desde el servidor hacia los dispositivos móviles se hace uso del conjunto de herramientas para desarrolladores Google Firebase, más concretamente el servicio Firebase Cloud Messaging [20], que permite gestionar de forma sencilla las comunicaciones entre distintos componentes de un sistema.

En líneas generales, el funcionamiento de este servicio se basa en la asignación de un *token* de identificación único asociado a cada dispositivo y la suscripción de este dispositivo a uno o varios canales o tópicos de información relacionados con un proyecto en concreto. El *backend* envía un mensaje a FCM indicando el canal o canales a los que va dirigido y este lo hace llegar a todos los dispositivos registrados en la aplicación y suscritos a ese canal, tan pronto como estén disponibles [Figura 5.19]. También es posible remitir un mensaje hacia un destinatario específico a través de su *token*. En el caso de dispositivos móviles, estos mensajes llegan a los destinatarios en forma de notificación *push*.

Así, cada dispositivo que se registre en un evento, se suscribirá al canal asociado a ese evento para recibir sólo información de los eventos en los que está registrado.

Cuando se actualice información del evento como horas de salida de las modalidades, fecha del mismo u otros parámetros, el servidor enviará un mensaje a los dispositivos informando qué datos son los que tienen que actualizar y estos invocarán el *path* correspondiente de la API del *backend* para actualizar la información que ha cambiado.

5.6.5. Gestionar la imposibilidad de enviar información por falta de cobertura u otros motivos

Para dar solución a este reto, se propone como primera aproximación un servicio en la aplicación móvil, que remita a intervalos programados, todos aquellos registros de posición no remitidos. A este fin, y como se presentó en el apartado 5.4.2 se ha incluido una columna, en la tabla de la base de datos de los dispositivos móviles que almacena las posiciones registradas, que indica si la posición ha sido o no sincronizada.

Si la sincronización en un ciclo de envío de información falla, los datos que no han sido enviados se incluirán en el siguiente ciclo de sincronización programado.

5.6.6. Control/Verificación de registros de los corredores a través de sus dispositivos

Para dar solución a las circunstancias expuestas en el apartado 5.1.8, se diseña un proceso de registro en el evento basado en tres pasos [Figura 5.20]:

1. En primer lugar, es el propio corredor el que a través de la aplicación selecciona el evento deseado y se registra en él, solicitando así expresamente ser seguido. Como premisa para poder registrarse en un evento, se exige a los participantes que asocien su número de teléfono a la aplicación, que será el dato único que se utilice para identificarlos. Este número teléfono deberá coincidir con el indicado en el momento de la inscripción. Al ejecutar este paso, se enviará la solicitud junto con el *token* FCM y el teléfono al servidor y se quedará en estado “registro pendiente de confirmar”.

2. Una vez solicitado se pueden dar dos circunstancias:

   - Que el teléfono facilitado coincida con el de uno de los participantes de los equipos inscritos en el evento, en cuyo caso el registro se confirmará de forma automática
   - Que el teléfono facilitado no coincida con ninguno de los incluidos por el organizador en el registro de participantes del evento. En este caso, el organizador desde el portal Web deberá asociar el dispositivo a un corredor del evento para confirmar el mismo.

3. La confirmación o no del registro se le enviará de vuelta al dispositivo a través del *token* FCM asociado.

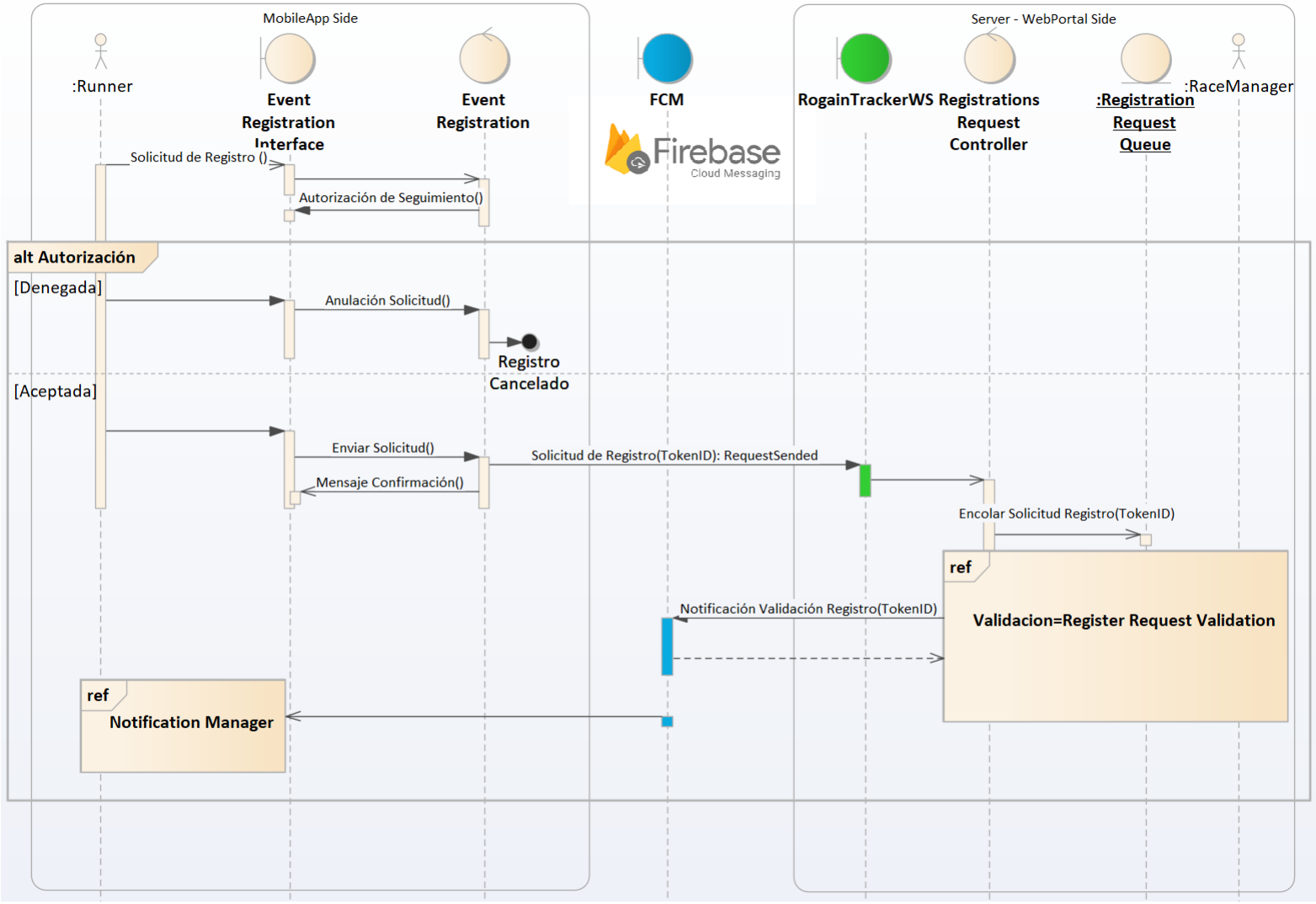


Figura 5.20: Diagrama de secuencia para registro de corredores desde aplicación móvil

5.6.7. Configuración automática de comienzo del seguimiento a la hora establecida

Una vez recibida la confirmación de registro desde el servidor, la aplicación móvil se descargará la información de seguimiento del evento: modalidad y categoría inscrita, hora de comienzo y finalización del mismo. Con estos datos, se programará automáticamente una tarea para que se ejecute el día y hora señalados, que marcará cuando el sistema comience a transmitir la posición. Igualmente, se programará la hora de finalización de dicho seguimiento.

De este modo, el participante sólo deberá tener el dispositivo encendido y el GPS y los datos habilitados en el momento de comienzo de la carrera ya que el dispositivo comenzará automáticamente a la hora programada a enviar información de posición al servidor.

5.6.8. Distinción de los equipos durante el seguimiento

Para lograr un distinción visual de los equipos durante el seguimiento se propone definir unas paletas de colores bien diferenciados que asignar de forma aleatoria y dinámica en función del número de equipos a seguir.

Se utiliza a tal fin la librería *open source* de Google *pallete.js* que permite trabajar con paletas de colores de forma simple.

5.6.9. Estimación de localización de controles en función de distancia al mismo

Se decide que sea el servidor, a partir de unos umbrales definidos, el que se encargue de realizar el cálculo de determinar si un control ha sido o no localizado. Este cálculo se efectuará cada vez que un dispositivo transmita un paquete de posiciones al *backend* teniendo en cuenta para el mismo también la precisión reportada por el GPS del dispositivo. Si se localizan varias posiciones que permitan dar por encontrado un mismo control, solo deberá registrarse una de ellas como válida.

5.6.10. Gestión de la reproducción del evento en tiempo real

Para sortear los posibles problemas derivados del retraso en el envío de la información de los dispositivos móviles y en busca de mostrar un seguimiento en tiempo real lo más fluido posible, se propone, que durante el mismo se muestre la posición de los equipos de forma intencionada con un tiempo de retraso lo suficientemente superior a la frecuencia de envío de información definida para los dispositivos móviles. De esta manera se añade una tolerancia a fallos en la transmisión de la información de los móviles de varios ciclos y se consigue el efecto de fluidez o continuidad en el movimiento de los equipos.

Por otra parte se decide añadir controles que permitan:

- Centrar el seguimiento en un equipo dado.
- Aumentar la velocidad de reproducción cuando se visualiza el evento en diferido.
- Mostrar u ocultar información de los equipos.
- Elegir la longitud, en minutos, del track mostrado para cada equipo.

5.6.11. Gestión de batería y asociación de más de un dispositivo por equipo

En esta primera versión de la solución del sistema no se han enfocado los esfuerzos en optimización de la batería según los parámetros indicados en el análisis y ni en la gestión de la asociación de más de un dispositivo por equipo.

Capítulo 6

Backend: Servicio Web

En el presente capítulo se pasa a describir las aspectos más relevantes de la implementación de la parte correspondiente al servicio Web. Para la construcción y gestión del proyecto y sus dependencia se utiliza Apache Maven [30].

Durante todo el proyecto se utiliza la inyección de dependencias incluida en Spring que permite gestionar de manera eficiente y sencilla el acoplamiento entre clases. Se encontrará durante el mismo numerosas anotaciones *@Autowired* asociada a atributos de una clase. Esta anotación indica que dicho atributo debe ser inicializado automáticamente por el inyector de dependencias de Spring, que, a grandes rasgos, se encargará de buscar un componente (*bean*) cuyo tipo coincida con el del atributo para poder utilizarlo en el momento de ser necesitado sin creación explícita por parte del programador de un objeto de dicha clase.

```
@Autowired
private IEventDAO dao;
```

Figura 6.1: Ejemplo de atributo gestionado por el injector de dependencias de Spring

6.1. Estructura del proyecto y aspectos relevantes del código

El proyecto se estructura según se muestra en la figura 6.2, donde se incluye cinco paquetes diferenciados:

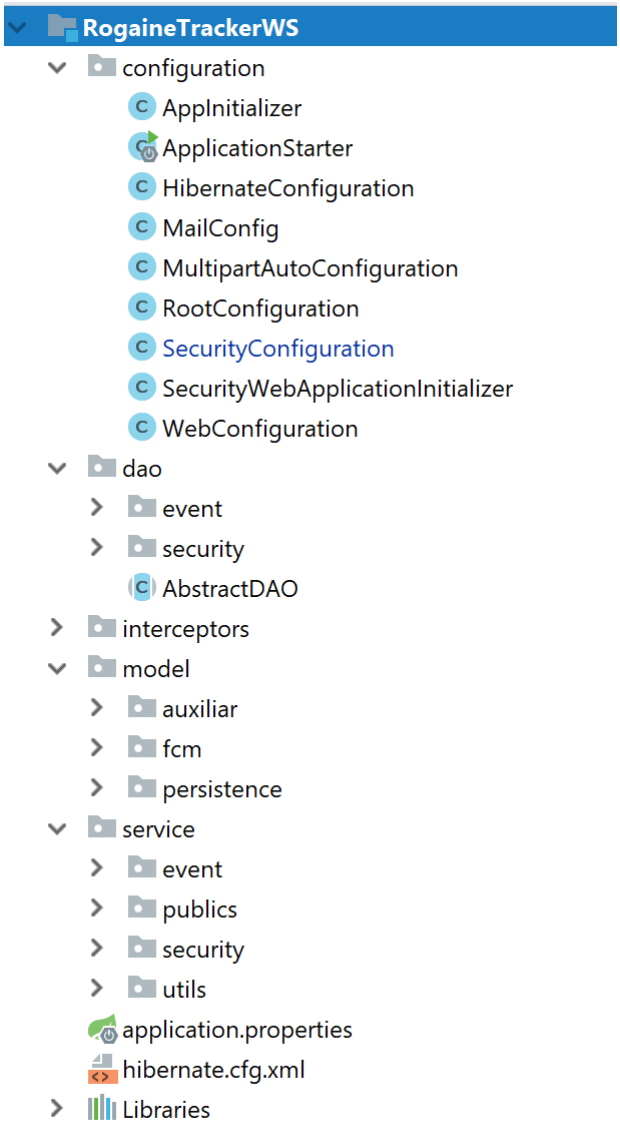


Figura 6.2: Estructura proyecto del *backend*

Configuration

Donde se incluyen todas las clases encargadas de la configuración e inicialización de la aplicación: establecimiento de la conexión a datos, seguridad de los *endpoints* del API, conexión a servidor de envío de correos y definición del método y entidad de autenticación.

Model

Que contiene todas las clases que definen las distintas entidades que forman parte de la solución. A su vez se dividen en tres packetes:

- ***Persistence:*** clases para el mapeado de las tablas de la base de datos en la aplicación. Se incluye una clase por cada tabla de la aplicación y el mapeado se define mediante anotaciones de Hibernate de la forma indicada en la figura 6.3. Así:
  - Para cada tabla se incluye una anotación *@Entity* y otra anotación *@Table* que tiene como parámetros el nombre de la tabla que corresponde a la clase, el *schema* de la base de datos y el nombre de la base de datos a la que pertenece la tabla.
  - Para cada propiedad, se incluye una anotación *@Column* donde se le indica el nombre de la columna de la tabla que se está mapeando y otros parámetros, como por ejemplo: si permite valores *null* o el número de caracteres máximos de la columna si se trata de tipo *varchar*.
  - Por último, también es posible integrar las relaciones mediante anotaciones *@OneToMany, @ManyToOne, @OneToOne* en función de la cardinalidad de la misma, que permite incluir enlaces de navegación directa desde una entidad a sus entidades relacionadas.
- **FCM:** clases para el proceso de comunicación con los dispositivos móviles a través del servicio Firebase Cloud Messaging.
- ***Auxiliar*:** entidades auxiliares para representación de las coordenadas geográficas o los errores de la aplicación entre otros.

```java
@Entity
@Table(name = "t000Device", schema = "public", catalog = "RogaineTracker")
public class T000DeviceEntity {
    private int devId;
    private String devFCMToken;
    private String devTelephoneNumber;
    private Timestamp devDateCreated;
    private Timestamp devLastUpdate;
    private Timestamp devLastAccess;
    private String devDeviceIdNumber;

    @JsonView(RTView.Authentificated.class)
    @Id
    @Column(name = "devId", nullable = false)
    @GeneratedValue(strategy = GenerationType.AUTO, generator = "auto_gen")
    @SequenceGenerator(name = "auto_gen", sequenceName = "t000device_devid_seq",allocationSize = 1)
    public int getDevId() { return devId; }

    public void setDevId(int devid) { this.devId = devid; }

    @JsonView(RTView.Basic.class)
    @Basic
    @Column(name = "devFCMtoken", nullable = false, length = 2048)
    public String getDevFCMToken() { return devFCMToken; }

    public void setDevFCMToken(String devfcmtoken) { this.devFCMToken = devfcmtoken; }

    @JsonView(RTView.Basic.class)
    @Basic
    @Column(name = "devTelephoneNumber", nullable = true, length = 50)
    public String getDevTelephoneNumber() { return devTelephoneNumber; }

    public void setDevTelephoneNumber(String devtelephonenumber) { this.devTelephoneNumber = devtelephonenumber; }
```

Figura 6.3: Ejemplo de mapeado Hibernate de la tabla T000Device

DAO (*Data Access Object*)

Son los encargados de gestionar el acceso a la base de datos. Se define una clase DAO por cada entidad de persistencia del paquete anterior y todas ellas implementan la clase abstracta *AbstracDAO* [Figura 6.4], que incluye las tareas comunes a todos los DAO : control de la sesión y las operaciones de CRUD (*Create, Read, Update y Delete*). Cada DAO, se debe anotar con *@Repository* (para que pueda ser utililizada por el injector de dependencias de Spring) y además de las operaciones básicas de crear, actualizar y eliminar, implementa una serie de métodos de lectura de datos (que se traducen en consultas *SELECT* sobre la base de datos) en función de diversas variables como se puede ver en la figura 6.5.

```
@Transactional(noRollbackFor = RuntimeException.class)
public abstract class AbstractDAO<PK extends Serializable, T> {

    private final Class<T> persistentClass;

    @SuppressWarnings("unchecked")
    public AbstractDAO(){...}

    @Autowired
    private SessionFactory sessionFactory;

    protected Session getSession() { return sessionFactory.getCurrentSession(); }

    /unchecked/
    public T getByKey(PK key) { return (T) getSession().get(persistentClass, key); }

    public void batchPersist(ArrayList<T> entities) {...}

    public void persist(T entity) { getSession().persist(entity); }

    public void update(T entity) { getSession().merge(entity); }

    public void delete(T entity) { getSession().delete(entity); }

    protected Criteria createEntityCriteria() { return getSession().createCriteria(persistentClass); }

}
```

Figura 6.4: Clase *AbstracDAO* para la gestión del acceso a datos

Services

Donde se incluyen todas las clases que implementan controladores y servicios que gestionan toda la lógica del *backend*. Se distinguen dos tipos de servicios básicos:

- **Controladores del API REST**: cuya misión es la de especificar los *endpoints* sobre los que se atenderá las peticiones de los clientes. Cada controlador, esté marcado con la anotación *@RestControler* y la anotación *@RequestMapping* asociada a la ruta base de las peticiones que atenderá dicho controlador. Cada método de la clase puede a su vez, ir anotado como *@RequestMapping* indicando la el método HTTP y el *path* relativo, respecto al indicado para la clase, sobre el que atenderá las peticiones. Este *path* puede incluir a su vez partes variables y otros parámetros de la ruta [Figura 6.6].

- **Resto de servicios**: que controlan otras tareas como la authenticación, el envío de correos (que se realiza utilizando plantillas *velocity*), la gestión de

```java
@Repository
public class EventDAO extends AbstractDAO<Integer, T001EventEntity> implements IEventDAO {

    @Autowired
    IRogaineManagerDAO rmaDAO;

    public T001EventEntity findById(String rmaEmail, int eveId) {

        Criteria criteria = createEntityCriteria();

        criteria
                .add(Restrictions.idEq(eveId))
                .createCriteria("t001RogaineManagerByRmaId")
                .add(Restrictions.eq( propertyName: "rmaEmail",rmaEmail));

        return (T001EventEntity) criteria.uniqueResult();
    }


    public T001EventEntity findById(int eveId) {...}

    public ArrayList<T001EventEntity> findAllEvents(String rmaEmail) {...}

    public ArrayList<T001EventEntity> findAllEvents() {...}

    public void create(String rmaEmail, T001EventEntity event) {...}

    public void update(String rmaEmail, int eveId, T001EventEntity event) {...}

    public void delete(String rmaEmail, int eveId) {...}
}
```

Figura 6.5: Clase EventDAO para la gestión de acceso a datos de la tabla T001Event

ficheros o el envío de mensajes a los dispositivos móviles a través del servicio FCM de Google.

Por otra parte, un aspecto importante de la implementación de los controladores es poder diferenciar la información, que se devuelve asociada a una entidad dada, en función del destino de los datos. Por ejemplo, no se debe devolver la misma información de una entidad del tipo *T001EventEntity* para que sea mostrada en la parte pública del portal Web donde, entre otros, la columna que indica si el evento es visible no tiene sentido, que para ser mostrado en la parte de gestión del mismo donde se debe tener acceso a todos los campos. Para dar solución a esta circunstancia, se utiliza la anotación *@JsonView* del paquete *com.fasterxml.jackson* que permite definir distintas vistas de los datos para la respuesta dada por cada *endpoint*. Así, anotando las propiedades de la clase *T001EventEntity* con la vista en la que se incluyen y posteriormente anotando los controladores con la vista de la

```java
@RestController
@RequestMapping(value = "/event")
public class EventService implements IEventService {

    @Autowired
    private IEventDAO dao;

    @JsonView(RTView.Authentificated.class)
    @RequestMapping(value="{eveId}",method = RequestMethod.GET) //gestiona el endpoint "/event/{eveId}"
    public T001EventEntity findById(@AuthenticationPrincipal UserDetails rmaUser, @PathVariable("eveId") int eveId) {...}

    @JsonView(RTView.TrackingPrivate.class)
    @RequestMapping(value="{eveId}/track",method = RequestMethod.GET) //gestiona el endpoint "/event/{eveId}/track"
    public T001EventEntity findByIdTrackingEventPrivate(@AuthenticationPrincipal UserDetails rmaUser, @PathVariable("eveId")

    @JsonView(RTView.Authentificated.class)
    @RequestMapping(method = RequestMethod.GET) //gestiona el endpoint "/event"
    public ArrayList<T001EventEntity> findAllEvents(@AuthenticationPrincipal UserDetails rmaUser){...}

    @RequestMapping(method = RequestMethod.POST) //gestiona el endpoint "/event"
    public void create(@AuthenticationPrincipal UserDetails rmaUser, @RequestBody T001EventEntity event) {...}

    @RequestMapping(value="{eveId}",method = RequestMethod.PUT) //gestiona el endpoint "/event/{eveId}"
    public void update(@AuthenticationPrincipal UserDetails rmaUser, @PathVariable("eveId")int eveId, @RequestBody T001EventE

    @RequestMapping(value="{eveId}",method = RequestMethod.DELETE)//gestiona el endpoint "/event/{eveId}"
    public void delete(@AuthenticationPrincipal UserDetails rmaUser, @PathVariable("eveId")int eveId) {...}
}
```

Figura 6.6: Clase EventService. *Endpoints* asociados a la ruta */event/**

entidad que deben devolver, se consigue abordar esta diferenciación. Las distintas vistas posibles se definen en la clase *RTView* del paquete *model*. En la figura 6.8 se muestra toda la lógica para el ejemplo aquí expuesto para *T001EventEntity*.

Interceptors

Paquete donde se incluyen las clases que utilizan Programación Orientada a Aspectos (AOP por sus siglas en inglés), para realizar diferentes tareas. Este paradigma de programación permite separar y encapsular las distintas responsabilidades de los controladores de la aplicación, liberándolos de tener que ocuparse de las obligaciones transversales del sistema (un ejemplo claro es el registro de un *log* de acciones del programa donde en el paradigma tradicional cada uno de los controladores se tiene que preocupar de guardar el registro de la acción que a realizado), logrando que cada parte de la misma se ocupe sólo de la tarea encomendada y no de la gestión de otros aspectos no relacionados directamente con su misión [31].

En esta primera versión, únicamente se incluye un interceptor que es el encargado de supervisar los cambios que afectan al seguimiento de los equipos para, desde que

se produzcan, transmitir la información a los dispositivos móviles. Estos cambios se pueden dar en diferentes partes de la solución por lo que se trata de una obligación transversal (aquella que afecta a varias partes del sistema) donde la AOP tiene cabida. En Spring cualquier aspecto se identifica mediante la anotación *@Aspect* que permite definir puntos de intersección en distintas partes de la aplicación. Así por ejemplo, en la figura 6.7 se observa que en este caso se definen puntos de intersección para las operaciones de:

- **Actualización de una modalidad**: concretamente cuando se cambia la hora de comienzo de la misma es necesario enviar esta información a los dispositivos móviles para que se reprogramen.
- **Actualización de un equipo:** cuando se modifica la modalidad o categoría de un equipo, es necesario comunicar la nueva información a los dispositivos móviles ya que distintas modalidades pueden tener horas de comienzo distinto y deben reprogramarse.
- **Actualización de un corredor:** cuando se modifica el teléfono de un participante y al ser este el identificador usado para asociar un dispositivo a un corredor, debe comprobarse si el nuevo número de teléfono permite vincular el corredor a un dispositivo y desvincularlo de un dispositivo anterior si es que lo estaba.

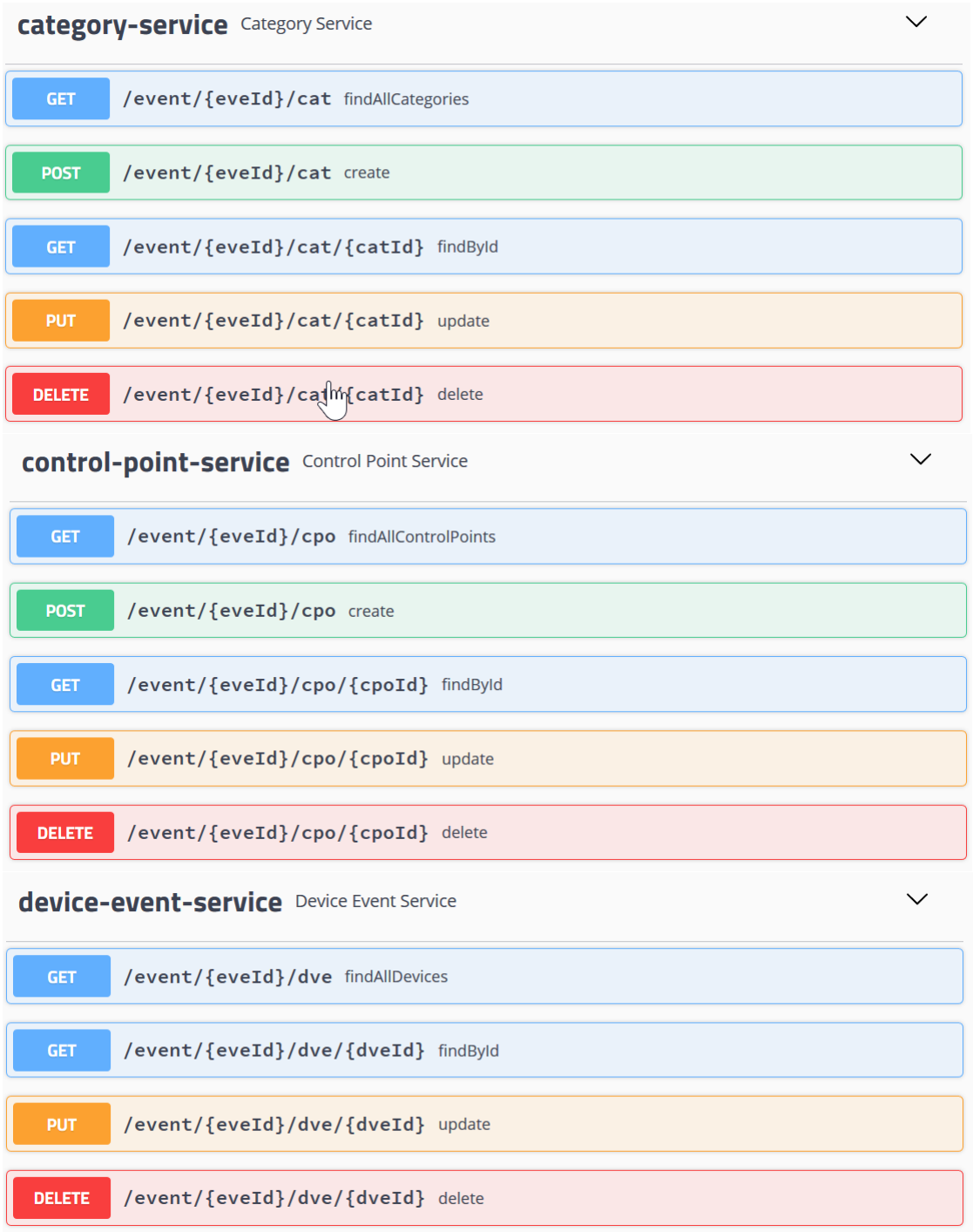
```
@Component
@Aspect
public class FCMInterceptor {

    /**...*/
    @Around("execution(* dao.event.IModalityDAO.update(*,*,*,*)) && args(rmaEmail, eveId, modId,mod)")
    public Object modalityUpdateAdvice(ProceedingJoinPoint pjp, String rmaEmail, int eveId, int modId, T001ModalityEntity mod) throws Throwable {...}

    @Around("execution(* dao.event.ITeamDAO.update(*,*,*,*)) && args(rmaEmail, eveId, teaId, tea)")
    public Object teamUpdateAdvice(ProceedingJoinPoint pjp, String rmaEmail, int eveId, int teaId, T001TeamEntity tea) throws Throwable {...}

    @Around(value = "execution(* dao.event.IRunnerDAO.update(*,*,*,*)) && args(rmaEmail, eveId, runId, run)")
    public Object runnerUpdateAdvice(ProceedingJoinPoint pjp, String rmaEmail, int eveId, int runId, T001RunnerEntity run) throws Throwable {...}
```

Figura 6.7: Intercepciones de operaciones que afectan al seguimiento de los equipos

6.2. Especificación API

Para generar la documentación de la API del *backend,* se utiliza el paquete de *io-springfox swagger-ui* que, mediante un pequeño fichero de configuración y algunas anotaciones, permite hacerlo de forma automática para cada uno de los *endpoints* del servicio Web.

A continuación se muestran todos los *endpoints* que expone la API:



device-public-service Device Public Service

- GET `/public/dev` findByDeviceId
- POST `/public/dev` createOrUpdate
- POST `/public/dev/dve` register
- POST `/public/dev/event/{eveId}/track` updateTrackLog

event-public-service Event Public Service

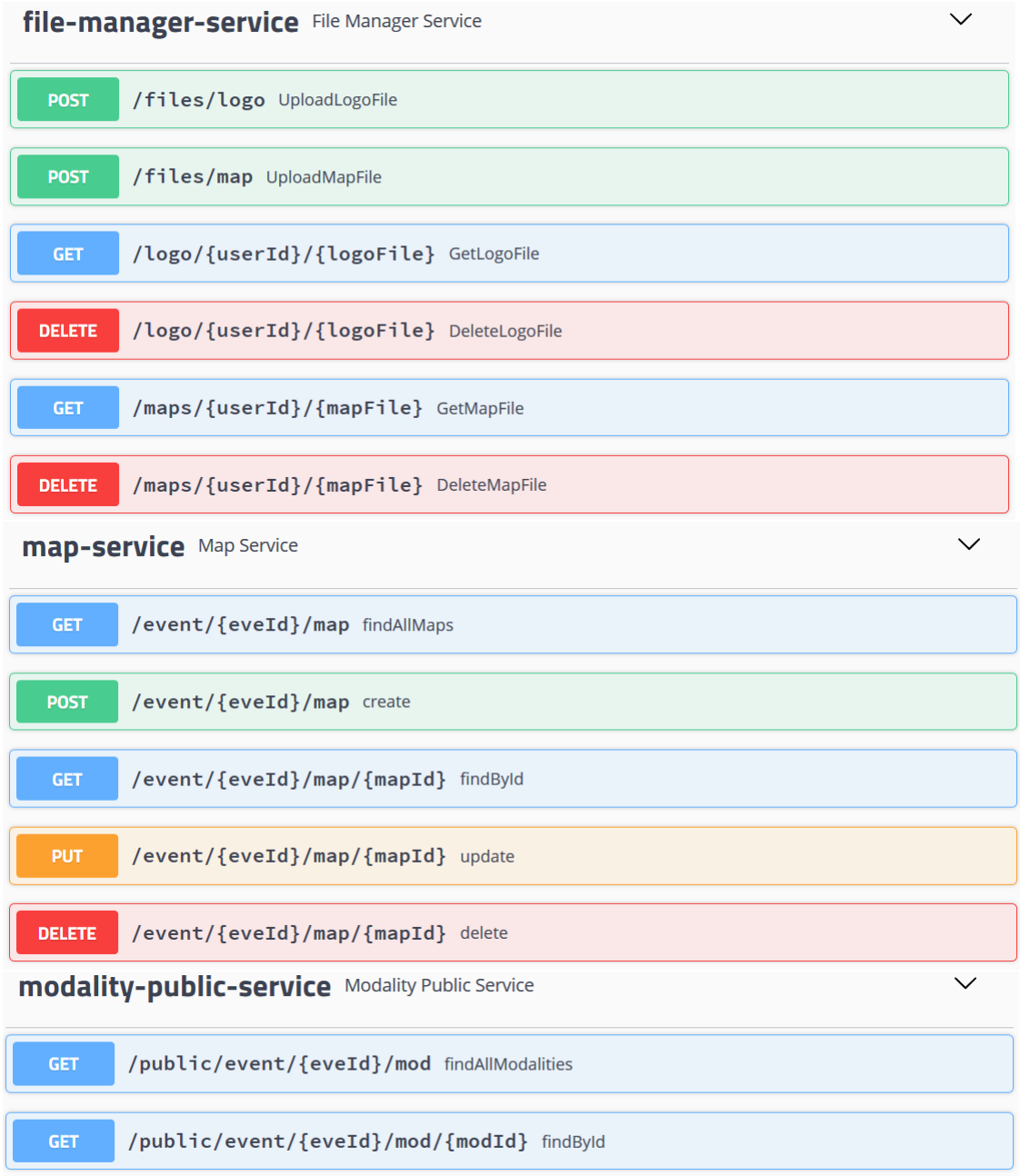
- GET `/public/event` findAllEvents
- GET `/public/event/{eveId}` findById
- GET `/public/event/{eveId}/run/{runId}` findEventTrackedConfigrationByrunId

event-service Event Service

- GET `/event` findAllEvents
- POST `/event` create
- GET `/event/{eveId}` findById
- PUT `/event/{eveId}` update
- DELETE `/event/{eveId}` delete
- GET `/event/{eveId}/track` findByIdTrackingEventPrivate

event-track-service Event Track Service

- GET `/event/{eveId}/track/mod/{modId}` getTrackLogUpdatesBetweenDates


file-manager-service File Manager Service
POST /files/logo UploadLogoFile
POST /files/map UploadMapFile
GET /logo/{userId}/{logoFile} GetLogoFile
DELETE /logo/{userId}/{logoFile} DeleteLogoFile
GET /maps/{userId}/{mapFile} GetMapFile
DELETE /maps/{userId}/{mapFile} DeleteMapFile
map-service Map Service
GET /event/{eveId}/map findAllMaps
POST /event/{eveId}/map create
GET /event/{eveId}/map/{mapId} findById
PUT /event/{eveId}/map/{mapId} update
DELETE /event/{eveId}/map/{mapId} delete
modality-public-service Modality Public Service
GET /public/event/{eveId}/mod findAllModalities
GET /public/event/{eveId}/mod/{modId} findById

modality-service Modality Service

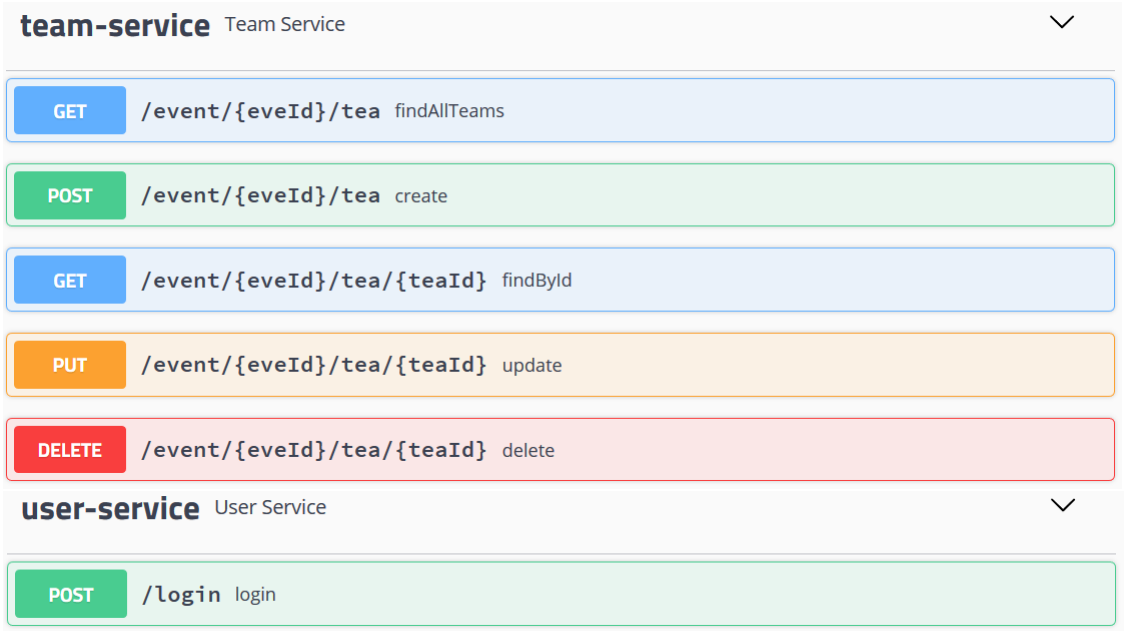
GET	/event/{eveId}/mod	findAllModalities
POST	/event/{eveId}/mod	create
GET	/event/{eveId}/mod/{modId}	findById
PUT	/event/{eveId}/mod/{modId}	update
DELETE	/event/{eveId}/mod/{modId}	delete
GET	/event/{eveId}/mod/{modId}/course	findCourse
PUT	/event/{eveId}/mod/{modId}/course	updateCourse
DELETE	/event/{eveId}/mod/{modId}/course	deleteCourse
GET	/event/{eveId}/mod/course	findAllCourses

registration-service Registration Service

POST	/confirmRegistration	confirmRegistration
POST	/register	register

runner-service Runner Service

GET	/event/{eveId}/run	findAllRunners
POST	/event/{eveId}/run	create
GET	/event/{eveId}/run/{runId}	findById
PUT	/event/{eveId}/run/{runId}	update
DELETE	/event/{eveId}/run/{runId}	delete



6.3. Funciones no implementadas

En el presente apartado se indica la relación de funcionalidades del *backend* estudiadas durante fase de análisis pero no implementadas en esta primera versión funcional:

- Uso de más de un dispositivo del mismo equipo para el seguimiento; bien en simultáneo para mejora de precisión o bien secuencial, uno tras otro, para ampliar el tiempo durante el cuál un equipo puede ser seguido.
- Análisis sobre las posiciones reportadas por los equipos para cálculo y actualización de clasificaciones en tiempo real, distancia recorrida, desnivel acumulado, etc.
- Suavizado del *track* reportado por los equipos mediante distintas función de interpolación.
- Control de hora de comienzo de visibilidad pública de seguimiento.
- Correos de registro con formato atractivo.
- Programación de clases de prueba para testear y validar el correcto comportamiento de, al menos, los casos de uso principales de esta parte del sistema.

```java
public class RTView {
    /**
     * Vista básico con los campos a mostrar en todas las vistas
     */
    public interface Basic {}
```

(a) Definición de la vista *Basic*

```java
/**
 * Campos específicos de la vista authenticated
 */
public interface Authentificated extends NonBasic_MobileApp_Authenticated_Shared, NonBasic_Authentificated_TrackingPublic_Shared, Basic {}
```

(b) Definición de la vista *Authenticated* (que incluye todos los de la *Basic*)

```java
    @JsonView(RTView.Basic.class)
    @Basic
    @Column(name = "eveWebPage")
    public String getEveWebPage() { return eveWebPage; }

    public void setEveWebPage(String eveWebPage) { this.eveWebPage = eveWebPage; }

    @JsonView(RTView.Basic.class)
    @Basic
    @Column(name = "eveDescription")
    public String getEveDescription() { return eveDescription; }

    public void setEveDescription(String eveDescription) { this.eveDescription = eveDescription; }

    @JsonView(RTView.Authentificated.class)
    @Basic
    @Column(name = "eveIsVisible", nullable = false)
    public Boolean getEveIsVisible() { return eveIsVisible; }

    public void setEveIsVisible(Boolean eveIsVisible) { this.eveIsVisible = eveIsVisible; }
```

(c) Asignación de la vista a la que pertenece de cada propiedad de *T001EventEntity*

```java
@RestController
@RequestMapping(value = "/event")
public class EventService implements IEventService {

    @Autowired
    private IEventDAO dao;

    @JsonView(RTView.Authentificated.class)
    @RequestMapping(method = RequestMethod.GET) //gestiona el endpoint "/event"
    public ArrayList<T001EventEntity> findAllEvents(@AuthenticationPrincipal UserDetails rmaUser){...}
```

(d) Devolución de la lista de eventos del controlador *EventService*. Vista *Authenticated*

```java
    @RestController
    @RequestMapping("/public/event")
    public class EventPublicService implements IEventPublicService {

        @Autowired
        IEventDAO dao;
        @Autowired
        IRunnerDAO runDAO;

        @JsonView(RTView.Basic.class)
        @RequestMapping(method = RequestMethod.GET)
        public ArrayList<T001EventEntity> findAllEvents() { return dao.findAllEvents(); }
```

(e) Devolución de la lista de eventos del controlador *EventPublicService*. Vista *Basic*

Figura 6.8: Gestión de vista de datos para *T001EventEntity*

Capítulo 7

Frontend: Portal Web

En el presente capítulo se pasa a describir las aspectos más relevantes de la implementación de la parte correspondiente al portal Web, a través del cual se gestionan los distintos eventos y se visualiza el seguimiento de los equipos.

7.1. Estructura del proyecto y aspectos relevantes del código

El proyecto se estructura según se muestra en la figura 7.1, donde se distinguen cinco paquetes diferenciados:

Views y Controllers

En AngularJS un componente o parte de la aplicación se define normalmente por un archivo HTML que contiene los elementos de la vista [Figura 7.2] y un archivo Javascript que implementa el controlador [Figura 7.3] asociado a la misma [Figura 7.4].

Adicionalmente, AngularJS permite la definición de directivas, que básicamente consiste en la asignación de una etiqueta HTML a un componente que al incluirlo en el archivo *.html* de una vista determinada, AngularJS se encarga de sustituirla por el código correspondiente. De este modo, un componente puede hacer uso dentro de él de otros componentes definidos en la aplicación lo que permite encapsularlos para

hacerlos reutilizables.

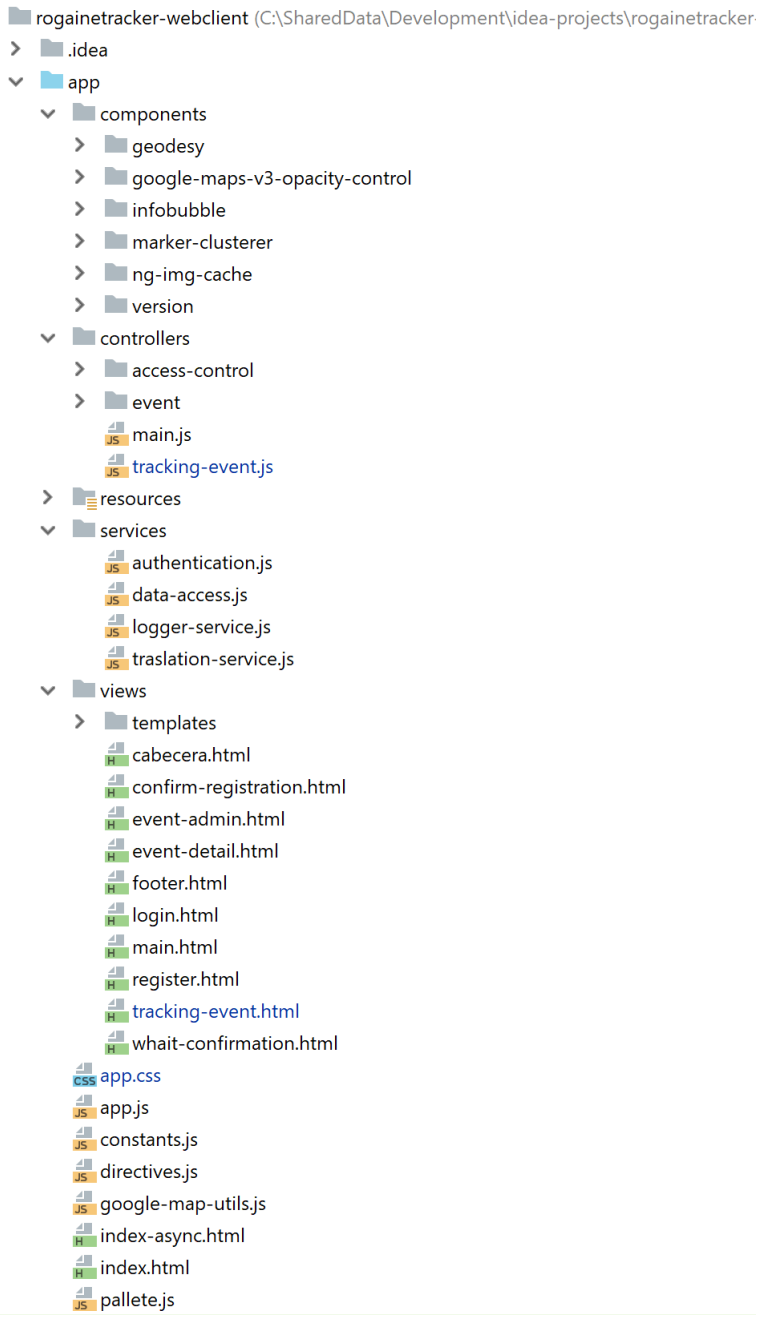


Figura 7.1: Estructura proyecto del componente correspondiente al portal Web

Esta ventaja, se ha tratado de explotar al máximo en este proyecto, minimizando la cantidad de código necesario para implementar la solución y facilitando su mantenimiento.

Por lo tanto, en los paquetes *views* y *controllers*, se incluye la definición de todas vistas y controladores del portal respectivamente.

```
<img src="resources/images/icons/rogaine_tracker_full_logo.svg" style="clear: both; width:15%; margin:3% auto;margin-bottom: 50px;"/>
<div  ng-cloak layout="row" layout-align="center center" layout-fill layout-margin>
<form class="login" name="login.form" ng-submit="login.onSubmit()" novalidate>
    <formly-form model="user" fields="login.fields"></formly-form>
    <md-button aria-label="loginLogin" class="md-raised md-primary" type="submit" ng-disabled="login.form.$invalid">{{translation.LOGIN_LOGIN}}</
    <md-button aria-label="loginCancel" class="md-raised md-warn" ng-click="login.onCancel()">{{translation.CANCEL}}</md-button>
</form>
</div>
```

Figura 7.2: AngularJS. Ejemplo de vista: *login.html*

```
app.controller('LoginController',['$scope',
    '$location',
    'AuthenticationService',
    '$rootScope',
    function($scope, $location,AuthenticationService,$rootScope){

    var user = {};
    $scope.login = [];
    $scope.login.onSubmit = onSubmit;
    $scope.login.onCancel = onCancel;
    $scope.user = user;

    $scope.login.fields = [...];

    //Reiniciamos los posibles credenciales que puedan existir
    var authenticationService = new AuthenticationService();

    (function initController() {...})();

    function onSubmit(){...}

    function onCancel(){...}
    }]
);
```

Figura 7.3: AngularJS. Ejemplo de controlador: *login.js*

```
app.config(["$routeProvider","$locationProvider","templateMaterialFormUrl",function($routeProvider,$locationProvider,templateUrl){

    $routeProvider.when("/login", {
        templateUrl: "views/login.html",
        controller: "LoginController"
    });
}]);
```

Figura 7.4: AngularJS. Vinculación de la vista al controlador asociado

Sin embargo, si se observa el nombre los archivos de la estructura de la figura 7.1, se puede ver que no todos los controladores tienen su vista correspondiente asociada. Esto es porque existen componentes cuya vista tiene una implementación específica y componentes cuya vista es común y sólo cambia el contenido de la misma. Para este tipo de elementos se han definido directivas para hacerlos reutilizables dentro del paquete *templates.* Estos elementos son los más interesantes para su estudio desde el punto de vista de la implementación entre los que destacan:

- **NswDynamicTable:** se trata de un componente basado en la librería de terceros *SmartTable* [32] que implementa una tabla con definición de columnas de modo dinámico y con funcionalidades de filtrado, ordenación y paginación, a la que se le ha añadido funcionalidades de:
  - Formateo de distintos tipos de datos: moneda, fechas, etc.
  - Vinculación a operaciones de CRUD del modelo subyacente asociado a cada fila.
  - Asociación a la tabla de un formulario en forma modal que es el que se encarga de gestionar la entrada de datos para las operaciones de creación y actualización.
  - Posibilidad de asociación a su vez a cada columna de componentes HTML complejos.
- **NswFormPopup:** basado en la librería de terceros *Angular-Formly [33],* implementa un formulario con campos y contenido dinámico en modo modal, que se utiliza, entre otras funciones para las operaciones de creación y actualización de las filas de la tabla dinámica. Sobre este, se definen múltiples tipos de campos admitidos [Figura 7.5] que enriquecen la funcionalidad del formulario: selectores de fecha, selectores de tiempo, listas desplegables, selectores de archivo, controles *Drag&Drop*, selectores de posición en un mapa, etc.
- **NswLocationPicker:** directiva que asocia a un botón una ventana modal con la que seleccionar la posición de un elemento sobre un mapa de Google Maps.

```
angular.module("nsw.modules")
    .directive('elastic', [...])
    .config(['formlyConfigProvider',
        '$windowProvider',
        '$mdDateLocaleProvider',
        function(formlyConfigProvider,$windowProvider,$mdDateLocaleProvider){

        formlyConfigProvider.setType({name: "datetimepicker"...});

        formlyConfigProvider.setType({name: "timepicker"...});

        formlyConfigProvider.setType({name: "locationpicker"...});

        formlyConfigProvider.setType({name: "uploadfile"...});

        formlyConfigProvider.setType({name: "dndlist"...});

        formlyConfigProvider.setType({name: "memo"...});

        formlyConfigProvider.setType({name: "dynamicsizeinput"...});
        }])
```

Figura 7.5: Directiva para formulario dinámico: *nsw-form-popup.js*

En base a estas tres directivas se definen prácticamente todos los controladores de la parte de gestión del evento: modalidades, categorías, mapas, controles, recorridos, equipos, corredores y dispositivos. Configurando el modelo subyacente, las rutas asociadas a las operaciones CRUD y definiendo cómo se mostrará el dato asociado a cada columna se obtiene el resultado deseado. En la figura 7.6, se puede ver un ejemplo para el controlador de la tabla de balizas de un evento.

Definidas cada una de las tablas, para utilizarlas basta con asociarlas a un atributo del controlador de la gestión de un evento y vincularlo a su vista mediante la directiva de la tabla creada [Figuras 7.7 y 7.8].

En último lugar, dentro de estos paquetes, el componente encargado de la parte de la aplicación que permite hacer el seguimiento de un evento tanto en tiempo real como en diferido requiere especial atención. Se detalla a continuación las funcionalidades implementadas más relevantes:

- **Superposición de la imagen de mapa en la posición adecuada.** Google Maps integra la funcionalidad de superponer capas de imágenes sobre la ventana del mapa (*overlays)*, permitiéndo indicar los límites de la misma. De este modo, teniendo la información de georreferenciación de la imagen es sen-

```
app.factory('ControlPoint',["NswDynamicTableModel",
    "remoteResource",
    "Map",
    "nswLocationPickerControlMarkerImageUrl",
    "$rootScope",
    function(NswDynamicTableModel,remoteResource, Map, nswLocationPickerControlMarkerImageUrl, $rootScope){

    return function(eventoId, MapParent) {

        var ControlPoint = new NswDynamicTableModel();

        ControlPoint.eveId = eventoId;
        ControlPoint.showSearchBox = true;
        ControlPoint.showPagination = true;
        ControlPoint.showAddItem = true;
        ControlPoint.IdItemName="cpoId"; //Se define la columna que actua de identificador de la tabla
        ControlPoint.rtHeaders= [...];//Se definen las cabeceras de la tablay sus propiedades
        ControlPoint.addingFormTitle = $rootScope.translation.CONTROL_POINT_ADD_NEW;
        ControlPoint.updateFormTitle = $rootScope.translation.CONTROL_POINT_UPDATE;
        ControlPoint.auxiliarFormFields = [...]; //Se definen los campos del Formulario asociado
        //Se establece el modelo estandar para la creación de una nueva fila
        ControlPoint.addingFormModel = {eveId: eventoId...};

        //Se definen las rutas para las operaciones CRUD
        ControlPoint.getItemsPath = function(){...};
        ControlPoint.getItemPath = function(elementId){...}
        ControlPoint.addItemPath =  function(){...};
        ControlPoint.updateItemPath = function(elementId){...};
        ControlPoint.deleteItemPath = function(elementId){...};

        ControlPoint.parents.push({name:"Map", object: MapParent}); //Lo enlazamos con su padre
        //Añadimos un campo al modelo que nos permita elegir una posición sobre el mapa asociado
        ControlPoint.extraModelFields = [
            {getParentMap:function(mapId){
                return MapParent.rowCollection.find(x=> x["mapId"] === mapId);
            }},
            {icon:nswLocationPickerControlMarkerImageUrl
            },
            {label:  ControlPoint.addingFormModel.cpoControlCode.toString()
            },
        ];

        ControlPoint.refresh();

        return ControlPoint;
    }
}]);
```

Figura 7.6: Controlador para la tabla de controles de un evento: *control-point.js*

cillo añadirla y ubicarla donde corresponde. Como valor añadido Google Maps permite cambiar la opacidad de la imagen superpuesta pudiendo combinar la vista del mapa de carrera con la vista de las capas propias de Google Maps.

- **Dibujado de los elementos en el mapa mediante el uso de marcadores**. A cada marcador, en función del tipo de elemento que se trata: controles, punto de salida o equipos se le asigna una imagen diferente:

    - En el caso de los equipos, se utiliza una imagen en formato SVG que se construye en el momento del dibujado de cada equipo y que permite

```
//Código para la gestión de un evento
if(evento.eveId !== "new") {
    $scope.maps = new Map(evento.eveId);
    $scope.modalities = new Modality(evento.eveId, $scope.maps);
    $scope.categories = new Category(evento.eveId, $scope.modalities);
    $scope.controlpoints = new ControlPoint(evento.eveId, $scope.maps);
    $scope.courses = new Course(evento.eveId, $scope.modalities, $scope.controlpoints);
    $scope.teams = new Team(evento.eveId, $scope.modalities, $scope.categories);
    $scope.runners = new Runner(evento.eveId, $scope.teams);
    $scope.deviceevents = new DeviceEvent(evento.eveId, $scope.runners);
}
```

Figura 7.7: Inclusión de las tablas en el controlador

```
<md-tab ng-if="!event.IsNew && (teams.rowCollection.length > 0)" label="{{$root.translation.RUNNERS}}">
    <md-content class="md-padding">
        <nsw-dynamic-table ng-model="runners"></nsw-dynamic-table>
    </md-content>
</md-tab>
<md-tab ng-if="!event.IsNew" label="{{$root.translation.DEVICES}}">
    <md-content class="md-padding">
        <nsw-dynamic-table ng-model="deviceevents"></nsw-dynamic-table>
    </md-content>
</md-tab>
```

Figura 7.8: Inclusión de las tablas en la vista

de esta manera asociar a cada marcador de equipo el número de dorsal asignado al mismo y una combinación de colores que lo diferencie del resto.

- Por otra parte, el dibujado de los elementos en el mapa se hace en función del zoom del mismo y la escala definida para el mapa subyacente, modificando su tamaño cuando el zoom cambia.
- Por último, se le asocia a cada marcador de equipo una serie de etiquetas con información de adicional como el nombre, la categoría o los puntos (opcional y controlable desde la interfaz) y un evento al pinchar con el ratón sobre él, que muestra información detallada sobre el mismos en forma de ventana emergente.

- **Centrado del mapa en un punto elegido.** Permitiendo de este modo seguir a un equipo a medida que se desplaza por el terreno, manteniendo el mapa centrado en él.

- **Manejo del *slider*.** Si bien puede parecer trivial la implementación del *slider*,

esté requiere tener en cuenta una serie de circunstancias que añaden complejidad a la tarea de la implementación de la lógica asociada:

- El *slider* no es más que la representación de un conjunto discreto de valores sobre una barra lineal. Por esto, es necesario traducir esta serie de valores a momentos de tiempo correspondientes con el transcurso de la prueba teniendo en cuenta que:
  - El valor mínimo debe corresponder con el momento de inicio del seguimiento y el valor máximo con el momento de finalización del mismo. Al tener las modalidades de los distintos eventos distinta duración y momento de celebración, esta traducción debe ser dinámica y calcularse en función de estos parámetros.
  - El *slider,* debe ir incrementando su valor a medida que avanza el tiempo. Dado que la unidad mínima de tiempo que se usará para mostrar el avance de la prueba es el segundo, el incremento de una unidad del valor del *slider* debe corresponder con un segundo de tiempo para asociar de forma unívoca una posición del mismo con un instante de tiempo determinado. Implementando esto permite al usuario moverlo de forma manual hasta el instante deseado. Por lo tanto, también el número de valores posibles del *slider* será dinámico e irá en función de la duración de la prueba.
- Permitir mover el *slider* de forma manual por parte del usuario implica que el sistema debe responder ante el cambio de valor del mismo, recalculando la información asociada como recorrido de los equipos, puntos conseguidos o posición de los mismos. Por otra parte, si este movimiento se hace durante el seguimiento en tiempo real de la prueba, se debe controlar que no se pueda avanzar más allá del momento actual.
- Por último, permitir modificar la velocidad de reproducción implica que cambia la cantidad de información que se actualiza en el mapa cada segundo de tiempo real. Además se hace necesario controlar durante la reproducción en tiempo real, que cuando el *slider* llegue a la posición

correspondiente al momento actual, vuelva a velocidad normal ya que no es posible reproducir a mayor velocidad algo que aun no ha pasado.

- **Obtención de datos de seguimiento.** Obtener los datos de seguimiento para reproducción en diferido, requiere de una única petición al *backend* para traerse todos los datos del evento e ir dibujándolos en pantalla a medida que el *slider* simula el paso del tiempo. Sin embargo, durante el seguimiento en tiempo real se deben realizar y programar múltiples llamadas para ir obteniendo los datos de los que va disponiendo el servidor, donde se tiene en cuenta que:

  - No se puede realizar una llamada al servidor por cada segundo que pase ya que la latencia de la propia llamada, sumando al tiempo empleado en efectuar los cálculos necesarios en los dispositivos de los clientes no permitiría hacer el seguimiento de forma correcta. Para este fin, se tomó la decisión en fase de diseño de mostrar en el seguimiento en tiempo real las posiciones con X minutos de retraso de forma intencionada. De esta manera se puede programar llamadas para la obtención de información en bloques de varios minutos ofreciendo una sensación de fluidez y seguimiento en real. En cada nueva llamada se pide únicamente los datos no obtenidos desde la última solicitud de datos de seguimiento al servidor.
  - Al comenzar el seguimiento se realiza una llamada inicial que trae un bloque de información lo suficientemente grande para permitir tolerancia a fallos ante varias llamadas al servidor sin datos.

En la figura 7.9 se muestra un extracto de la estructura del código de controlador del componente de seguimiento.

```
control fab menu

opciones de seguimiento

inicialización y control del mapa

//region funciones para gestionar el seguimiento de los equipos

/** Función encargada de gestionar el comienzo del seguimiento al presionar el botón correspondiente ...*/
$scope.startTracking = startTracking;
function startTracking() {...}

let mapJob;
/** Función encargada de realizar la actualización continua de la posición de los equipos en el mapa ...*/
function startMapJob() {...}

/**
 * Función encargada de detener el hilo de seguimiento y todos los intervalos generados por la función startMapJob
 * @param stopPlayingFlag
 */
function stopMapJob(stopPlayingFlag = true) {...}

let trackerJob;
/**
 * Proceso encargado de ir recopilando la información de forma continua en la BBDD en el caso de que estemos
 * en seguimiento en tiempo real
 */
function jobDataRetrieve() {...}

/** Detiene el seguimiento en tiempo real ...*/
function stopTrackerJob() {...}

/** Detiene todos los procesos relacionados con el seguimiento ...*/
$scope.stopTracking = stopTracking;
function stopTracking() {...}

/** Devuelve el tiempo actual de carrera ...*/
$scope.getRaceTime = function () {...}

/** Centra el Mapa en el equipo indicado ...*/
$scope.centerMapOnTeam = function (team){...}

/** Determina si el evento ha comenzado ya ...*/
function isEventModalityStarted(){...}

/** Determina si el evento ha terminado ya ...*/
function isEventModalityFinished(){...}

/** Indica si el evento está siendo seguido en tiempo real; ...*/
$scope.isRealTimeTracking = isRealTimeTracking;
function isRealTimeTracking(){...}
//endregion
```

Figura 7.9: Extracto del código del controlador de la vista de seguimiento

Services

En este paquete se incluye los servicios transversales de la solución no asociados directamente a ninguna vista de la aplicación:

- ***Authentication.js:*** Encargado de gestionar el proceso de autenticación contra el servicio Web y de incrustar nuestras credenciales en cada llamada al mismo una vez *loggeados* para nos autorice a realizar las operaciones solicita-

das.

- ***Data-Access.js:*** Actúa de intermediario para todas las llamadas http al *backend*. Implementa los métodos básicos GET, POST, PUT y DELETE.

- ***Logger-Service.js:*** Servicio que permite realizar funciones de registro de acciones para depuración en fase de desarrollo.

- ***Translation-Service.js:*** Servicio que permite traducir la aplicación al idioma deseado mediante un archivo JSON donde se asocie los distintos términos a la traducción en el idioma correspondiente. Así, en cada parte de la solución donde se muestre el texto de la forma `$rootScope.translation.MODALITY_ADD_NEW;` , puede ser traducida al idioma deseado creando un archivo JSON con el nombre *translation-xx.json* donde XX representa el código internacional de dos letras del idioma al que se desea traducir. En el caso de que el archivo de traducción creado no contenga la traducción de todas las claves definidas, se utilizará para los términos no traducidos el idioma por defecto. Como ejemplo, se muestra en la figura 7.10 un extracto del archivo *translation-es.json* que es el utilizado por defecto. En esta primera versión la configuración del idioma utilizado se realiza en el archivo *app.js* si bien es posible implementarlo para permitir cambiarlo de forma dinámica desde el explorador.

```
"LOGIN_LOGIN":"ACCEDER",
"LOGIN_OR_REGISTER":"ACCEDER/REGISTRARSE",
"LOGIN_SIGIN":"Registrarse",
"LOGIN_SIGIN_ERROR_DUPLICATED":"Ya existe un usuario con el email indicado.",
"LOGOUT":"Salir",

"MAKING":"Elaboración",
"MAP":"Mapa",
"MAP_ADD_NEW":"Añadir nuevo mapa",
"MAP_AND_TRACKING":"Mapa y Seguimiento",
"MAP_AREA":"Área (Km*Km)",
"MAP_CHANGE":"Cambiar Mapa",
"MAP_DATA":"Datos del Mapa",
```

Figura 7.10: Extracto de archivo de traducción

Components

Paquete donde se incluye una serie de componentes adicionales, de librerías de terceros, utilizados en distintas partes del *frontend,* principalmente en la vista de seguimiento.

Resources

Conjunto de recursos de estáticos utilizados por la aplicación como imágenes y archivos de traducción.

7.2. La interfaz Web

A continuación se hará un recorrido general por las diferentes vistas implementadas desde el punto de vista del usuario.

7.2.1. Pantalla principal

Incluye una lista de los eventos disponibles para seguir, tanto en directo como en diferido, con información básica de los mismos, un enlace a su sitio Web y un enlace para el seguimiento. Además en la esquina superior derecha se añade el acceso para entrar a la parte privada del portal o solicitar el registro [Figura 7.11].

7.2.2. Pantalla de administración de eventos

Esta pantalla está formada por dos partes diferenciadas: un *dashboard* [Figura 7.12] donde se puede visualizar el listado de eventos vinculados a la cuenta que está autenticada, y una vista de detalle de evento desde donde se gestiona los mismos.

Figura 7.11: Pantalla de bienvenida

La vista de gestión de eventos se divide en pestañas donde en la primera, diferente al resto, se introduce los datos básicos del mismo [Figura 7.13]. El resto, implementadas a través de la directiva *nsw-dynamic-table* como se explica en el apartado anterior, tienen todas el mismo formato y funcionalidades, siendo a priori más sencillo el aprendizaje para el usuario final.

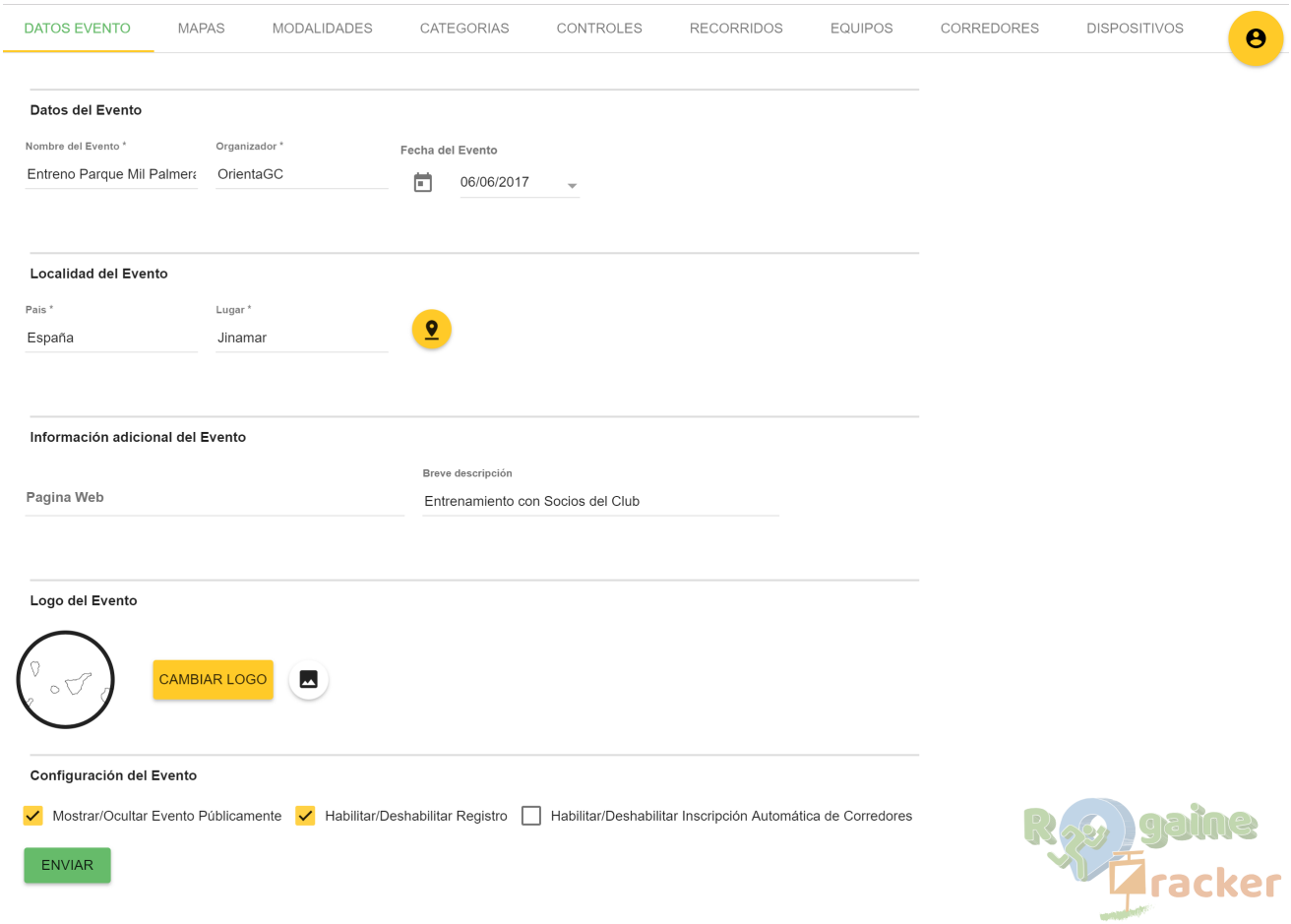


Figura 7.13: Datos generales del evento

Cada pestaña se compone de una tabla con el listado de datos relacionados, un recuadro de búsqueda, un botón para añadir un nuevo dato vinculado a la tabla, dos botones de edición y eliminación asociado a cada fila de la tabla y unos botones de navegación para avanzar o retroceder entre pestañas [Figura 7.14].

Figura 7.14: Pestaña de configuración de modalidades

Al pulsar en los botones de adición de una nueva fila a la tabla o de edición de la misma se muestra una ventana emergente con el modelo de datos definido [Figura 7.15]. Este formulario puede contener campos complejos como el de incluir un nuevo mapa georreferenciado [Figura 7.16] o la definición de los controles de un recorrido mediante elementos *Drag&Drop* [Figura 7.17]

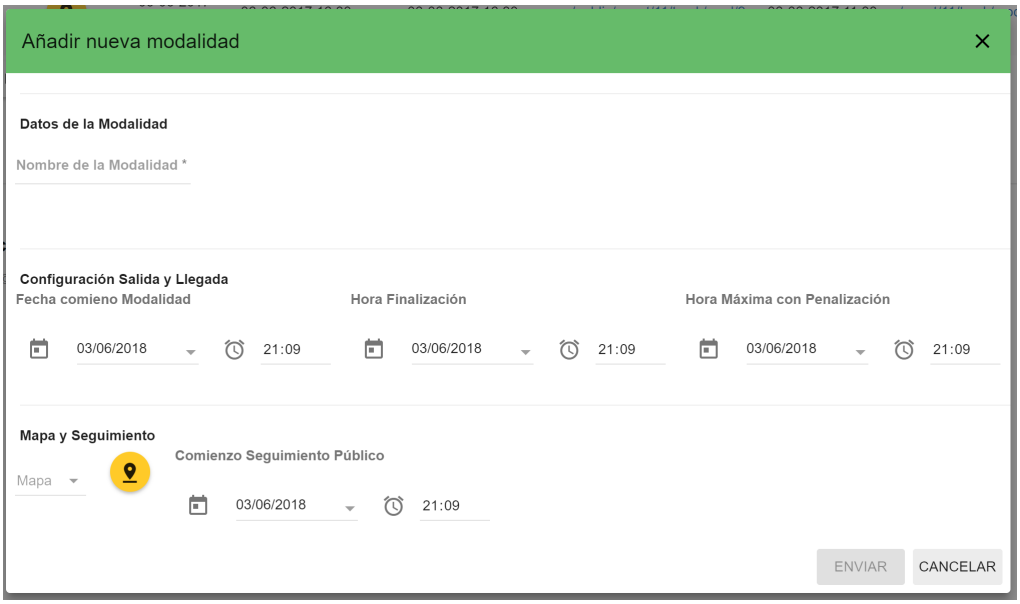


Figura 7.15: Formulario emergente de adición de una nueva modalidad

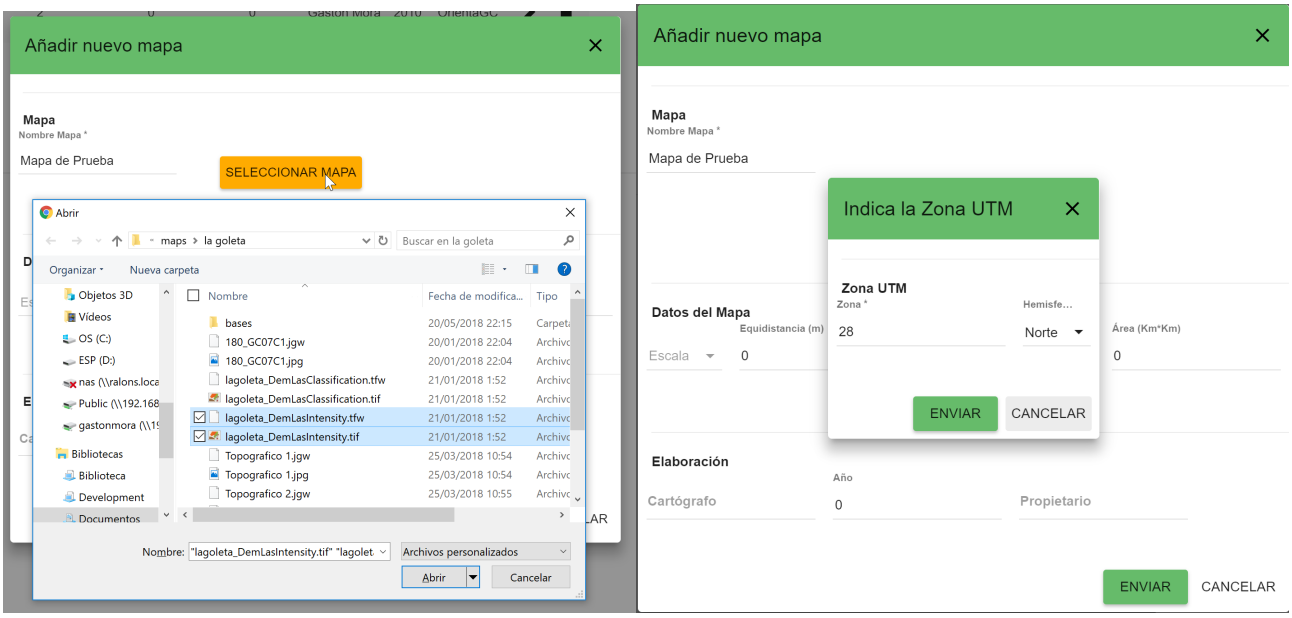


Figura 7.16: Formulario emergente inclusión de nuevo mapa

Figura 7.17: Componente *Drag&Drop* para definición de recorrido

Como se indica en el apartado 7.1, a cada columna se le puede asociar un modelo de representación específico. De todos ellos el más relevante es el asociado a columnas cuyo dato es una coordenada geográfica y que muestra en una ventana emergente sobre el mapa de carrera [Figura 7.18].

Figura 7.18: Columna con ventana de selección de localización

7.2.3. Vista de seguimiento de eventos

Por último, se encuentra la vista de seguimiento de la modalidad de un evento [Figura 7.19]. Cada modalidad tiene una URL de seguimiento distinta.

En ella se incluyen controles para la reproducción del evento y el avance manual a cualquier momento del mismo, la posibilidad de ver la información asociada a cada equipo y el track del mismo. Muestra el tipo de seguimiento que se está haciendo (en real o en diferido) y el nombre de la modalidad que está siendo visualizada. Además en la esquina superior izquierda implementa un menú desplegable con dos sub-menús [Figura 7.20]:

- El de **configuración del seguimiento** que permite realizar algunas configuraciones adicionales como: el tipo de mapa base a mostrar, la velocidad de seguimiento, la opacidad del mapa de competición, la longitud en minutos del *track* para cada equipo o la información mostrada junto al marcador de este último.

- La **lista de equipos** desde la que, en esta primera versión, se puede buscar en el mapa el equipo seleccionado o marcar uno de ellos para seguirlo manteniendo

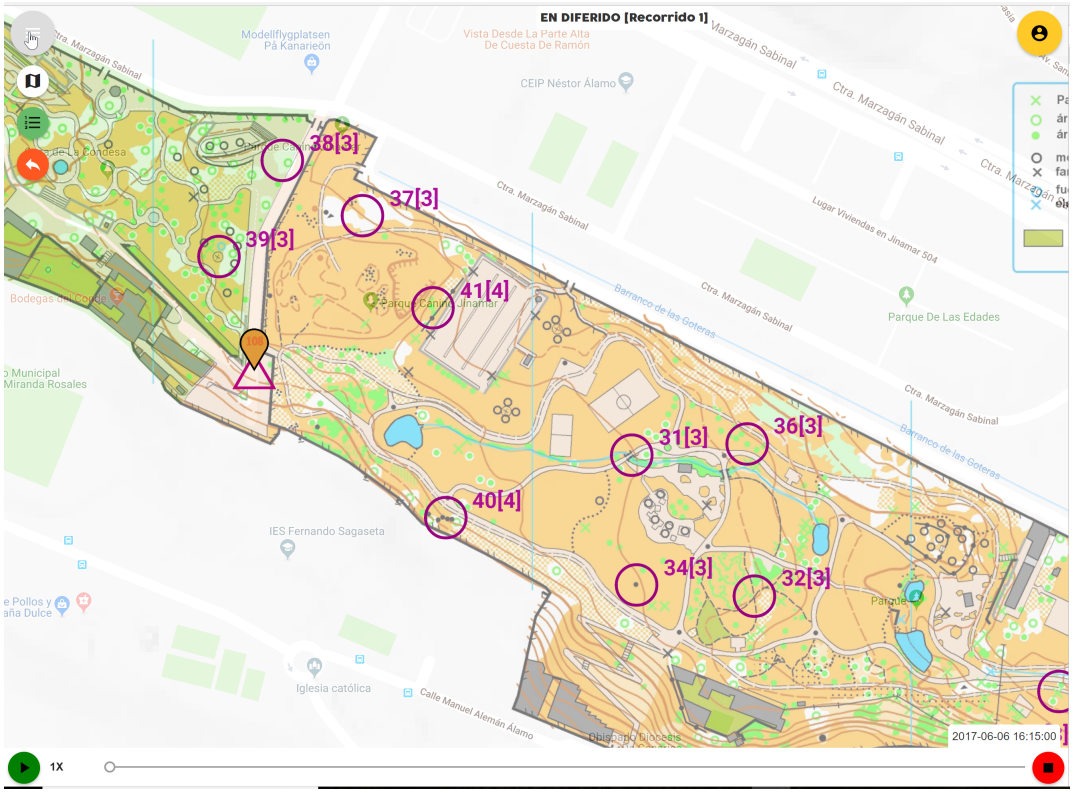


Figura 7.19: Vista de seguimiento de evento

el mapa centrado sobre él. Posteriormente, este menú será el que muestre las clasificaciones provisionales y otra información sobre los resultados de carrera como distancia recorrida por los equipos, desnivel acumulado, etc.

7.3. Funciones no implementadas

En el presente apartado se indica la relación de funcionalidades asociadas al *frontend* estudiadas durante fase de análisis pero no implementadas en esta primera versión funcional:

- Habilitación de vista de seguimiento a través del enlace público con temporizador para visualizar el tiempo hasta la apertura del la vista. Sólo es posible realizar el seguimiento a través del enlace privado accesible únicamente al gestor del evento.

- Inclusión en vista de seguimiento de clasificaciones en tiempo real y más es-

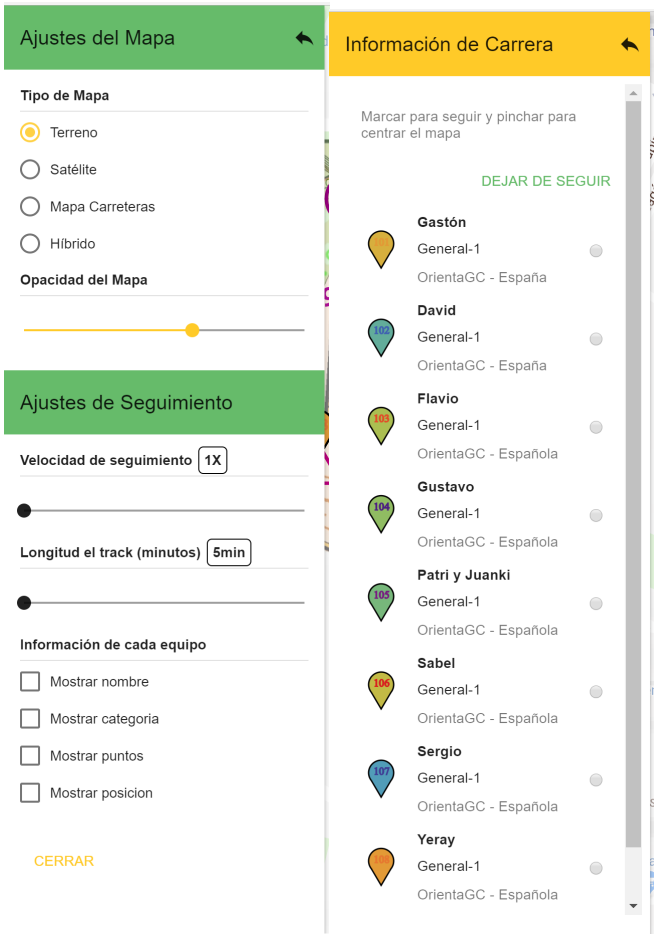


Figura 7.20: Menús de configuración de seguimiento y listado de equipos

tadísticas asociadas a los equipos y controles como puntuación acumulada, distancia recorrida, etc.

- Inclusión en vista de seguimiento de espacio para mostrar información de los patrocinadores y colaboradores del evento.
- Implementación del seguimiento en modo rejilla para tener varias ventanas abiertas al mismo tiempo, con diferentes partes del mapa o siguiendo a diferentes equipos.
- Filtrar equipos por categoría.
- Captura de errores reportados por el servidor e informe al usuario.
- Optimización del portal para visualización en dispositivos móviles.

- Documentación o tutorial de uso del sistema.
- Programación de clases de prueba para testear y validar el correcto comportamiento de, al menos, los casos de uso principales de esta parte del sistema.

Capítulo 8

Aplicación móvil

En el presente capítulo se pasa a describir las aspectos más relevantes de la implementación de la parte correspondiente a la aplicación móvil que es la encargada de recoger y reportar la posición de los equipos durante la carrera y permitir el registro de los corredores para ser seguidos durante la misma.

8.1. Estructura del proyecto y aspectos más relevantes del código

El proyecto se estructura según se muestra en la figura 8.1 y se ha realizado siguiendo las convenciones establecidas por la guía de desarrollo de Android:

Componentes visuales

Cada componente visual en Android que conforma una pantalla de la aplicación se denomina *activity* y consta de tres elementos :

- Una o varias definiciones (en función de la resolución y orientación de la pantalla del dispositivo) en XML de la vista cuyo archivo se almacena en la carpeta *layout* del paquete *resources.*
- Un clase de tipo *Activity* que actúa de controlador de dicha vista y que se ubica en el paquete *activities.*

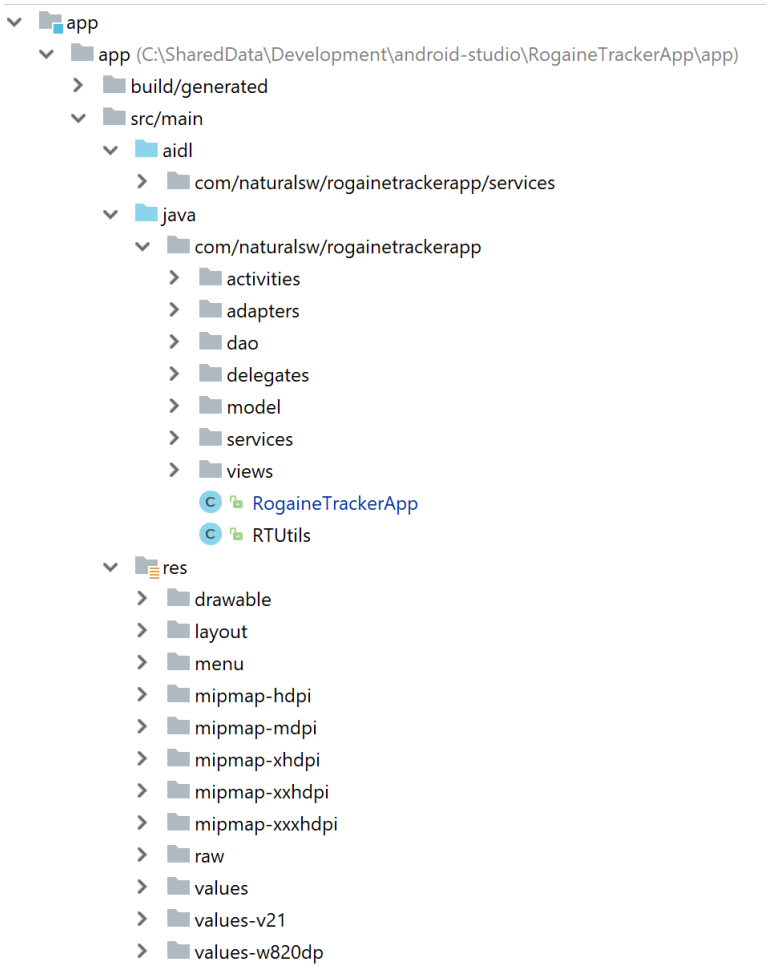


Figura 8.1: Estructura proyecto aplicación móvil

- Un adaptador que se encarga de implementar la lógica para incrustar los datos del modelo subyacente en los elementos de la vista y que se ubican en el paquete *adapters*.

La aplicación móvil de estudio se compone de cuatro componentes visuales:

- El listado de eventos disponibles.
- El detalle de cada uno de ellos.
- Un listado de las notificaciones recibidas.
- Una pantalla informativa mientras está activa la tarea de seguimiento.

Modelos de datos y objetos de acceso a datos

Al igual que en el *backend*, el acceso, mapeado y manipulación de datos de la aplicación para *smartphones* se ha organizado en los paquetes *model y dao*:

- El primero de ellos contiene el mapeado de las entidades de la base datos subyacente así como otras entidades necesarias para la comunicación con el servidor o la recepción de mensajes desde el servicio Firebase Cloud Messaging.
- En el segundo se incluye las clases encargadas de manejar las operaciones CRUD de las entidades anteriores sobre la base de datos.

Services

En este paquete se incluye las clases que conforman el núcleo de proceso de la aplicación. destacando:

- ***AsyncRestService y SyncRestService:*** Encargados de manejar las operaciones HTTP que se ejecutan contra la API del *backend,* de forma asíncrona y síncrona respectivamente. Para la gestión de las operaciones asíncronas se ha definido además, en el paquete **delegates** una clase abstracta *AsyncTaskOperations* [Figura 8.2] que expone una serie de métodos que el servicio usará para notificar o responder a los cambios de estado de las peticiones asíncronas realizadas. Derivadas de la clase anterior, se han implementado 5 tipos de operaciones asíncronas utilizadas en diferentes partes del código del programa.
- ***FCMInstanceIDService y FCMMessagingService:*** Para el registro y gestión de mensajes entrantes desde Firebase Cloud Message, implementado según lo indicado en [34].
- ***GeoLocationLogService:*** Gestiona la captura de las posiciones reportadas por el sensor GPS del *smartphone* durante la fase de seguimiento.
- ***GeoLocationReportService:*** Gestiona el envío recurrente, hacia el servidor, de las posiciones capturadas y almacenadas en la base de datos local.

- ***RTNotificationManager:*** Encargado de gestionar las notificaciones que se le deben mostrar a los usuarios de la aplicación.

- ***RTTaskManager:*** Utilizado para programar las tareas de comienzo y finalización del seguimiento según los parámetros de los eventos en los que el usuario se ha registrado

```
public abstract class AsyncTaskOperations<T, U extends Context> {

    protected U mAppContext;

    AsyncTaskOperations(U appContext) { this.mAppContext = appContext; }

    public void OnPreExecute(){};
    public void OnProgressUpdate(Integer... values){};
    public abstract void OnSuccessMethod(T t);
    public void OnErrorMethod(Exception e){};
    public void OnPostExecute(){};
}
```

Figura 8.2: Clase *AsyncTaskOperations* para peticiones asíncronas

8.2. La interfaz de la aplicación

Tal y como se ha introducido en el apartado anterior, la interfaz de este componente es sencilla y consta de 4 vistas básicas que pasan a mostrarse en el presente apartado.

8.2.1. Lista de eventos y pantalla de registro

A continuación, se analiza el flujo básico de uso de la aplicación haciendo un recorrido por la transición de vistas que se produce desde que se ingresa por primera vez en la misma hasta que se completa el registro en un evento. Como se muestra en la figura 8.3, este proceso pasa por ocho transiciones:

1. Al abrirla por primera vez, el usuario puede ver el listado de eventos disponibles con información básica de los mismos como son el nombre, la fecha de celebración, el lugar y el organizador del mismo así como si está o no disponible

para poder registrarse en él. Sin embargo, se le informa al usuario mediante un aviso que hasta que no asocie el número de teléfono a la aplicación, no podrá registrarse en ningún evento.

2. Al solicitar asociar el número de teléfono, se abre una ventana emergente donde se debe introducir el código de país y el número de teléfono.

3. Tras enviar al servidor esta información, desaparece el aviso anterior.

4. Cuando en la lista de eventos, el usuario pulsa sobre uno de ellos, accede a la vista con la información detallada del mismo donde se muestra los datos básicos del evento, un botón para solicitar el seguimiento en el evento y el listado de modalidades disponibles.

5. Al pulsar sobre una de las modalidades se puede ver información detallada de la misma como la URL de seguimiento, las categorías y la hora de comienzo y fin.

6. Al pulsar sobre el botón de registro, se abre una ventana emergente solicitando el nombre con el que se desea registrar el dispositivo.

7. Tras confirmar el registro, si todo ha ido correcto, recibe una notificación confirmándolo e informando al corredor de la modalidad y categoría donde está inscrito.

8. Tras volver a la lista de eventos, se observa en distinto color y estado, aquellos en los que se está registrado para ser seguido.

8.2.2. Pantalla de notificaciones

En esta vista se muestra el historial de las notificaciones recibidas por el usuario desde el servidor de la aplicación [Figura 8.4].

8.2.3. Pantalla de indicación de seguimiento

Esta pantalla se activa de forma automática cuando comienza la tarea programada para empezar a remitir las posiciones del dispositivo al servidor. Puede ser

anulada manualmente por el participante. Si no es así, se desactiva de forma automática a la hora de finalización programada [Figura 8.5].

8.3. Funciones no implementadas

En el presente apartado se indica la relación de funcionalidades estudiadas asociadas a los dispositivos móviles durante fase de análisis pero no implementadas en esta primera versión funcional:

- Lectura del nivel de batería de los dispositivos para reportar al servidor.
- Monitorización de operaciones del corredor sobre el dispositivo: desbloqueo de pantalla, uso de otras aplicaciones, etc.
- Programación de clases de prueba para testear y validar el correcto comportamiento de, al menos, los casos de uso principales de esta parte del sistema.

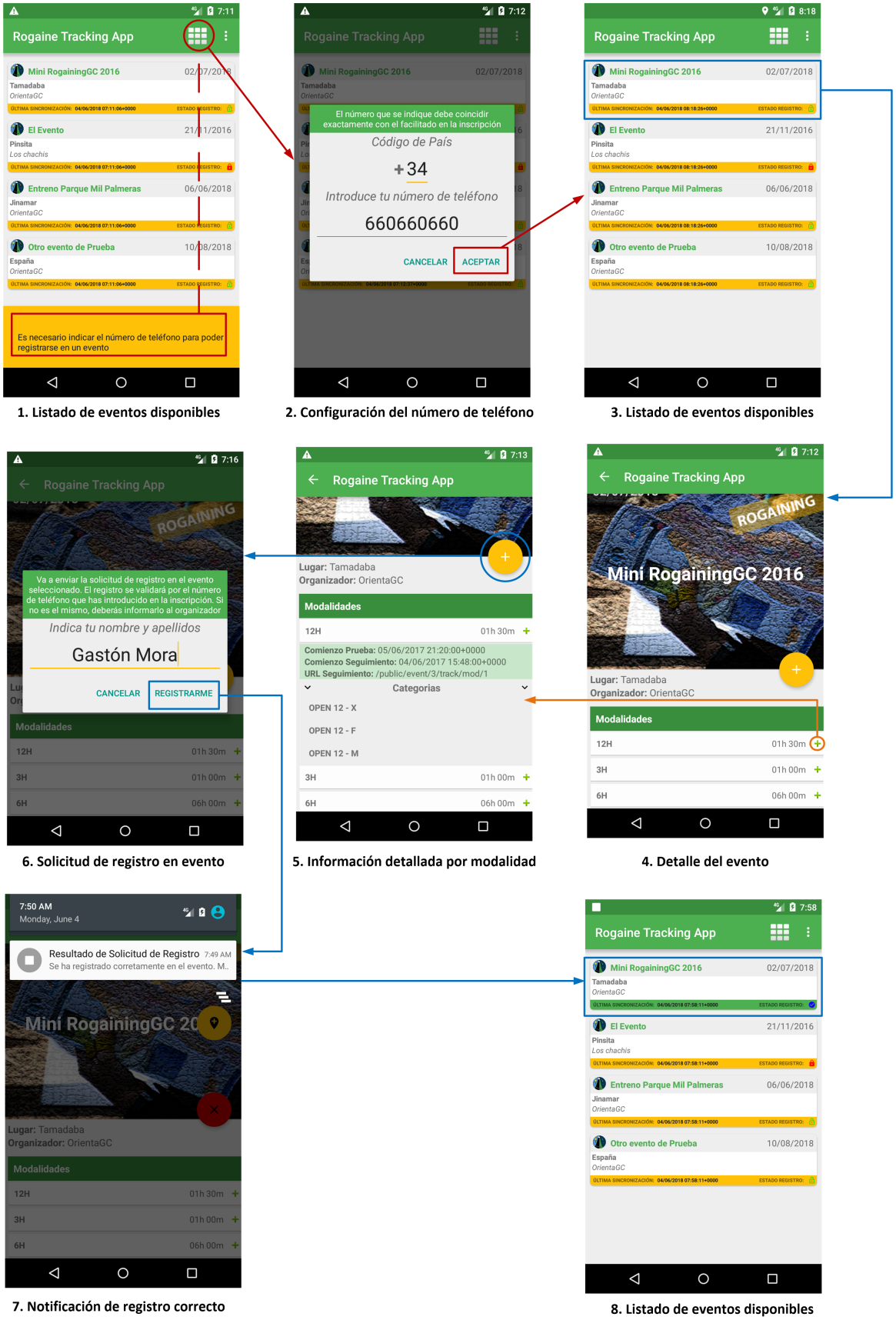


Figura 8.3: Flujo básico de registro en un evento

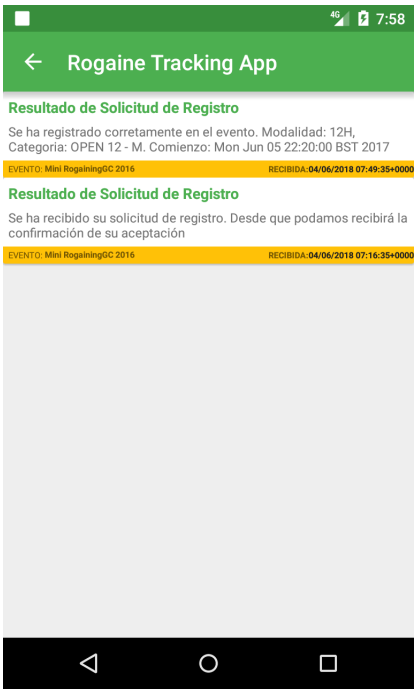


Figura 8.4: Vista de notificaciones

Figura 8.5: Vista de seguimiento activo

Capítulo 9

Pruebas y resultados

En el presente capítulo se expone las diferentes pruebas realizadas con la primera versión funcional del sistema propuesto. Las pruebas se dividen en 3 fases:

1. Ciclo completo de creación de un evento en la plataforma, desde el registro del usuario hasta la publicación del mismo.
2. Pruebas de registro en evento y seguimiento desde emulador Android.
3. Pruebas de registro en evento y seguimiento desde dispositivo real con sistema operativo Android.

9.1. Ciclo completo de gestión de evento

Para esta batería de pruebas se ha utilizado la aplicación desplegada en el servidor con las características indicadas en el anexo A.2, ubicado en la dirección http://server.naturalsw.com y un equipo del lado del cliente con la siguiente configuración :

- Portátil Dell cuyas características han sido descritas en el apartado 3.2.
- Navegador Google Chrome versión 67.

El ciclo testado comienza en el registro por parte de un usuario en el portal y finaliza en la creación de un evento totalmente configurado a excepción del registro de dispositivos que se realiza desde la aplicación móvil. Se ha pedido a un voluntario, con

conocimiento de esta modalidad deportiva, pero sin conocimiento de la aplicación, que sea quien realice esta prueba. La secuencia de acciones realizadas se indica en la figura 9.1.

Se genera un evento con la siguiente configuración (no se indica el valor de todos los campos):

- **Datos Básicos del Evento**
  - Nombre: San José del Álamo
  - Fecha: 10/06/2018
  - Organizador: Club Deportivo OrientaGC
- **Mapas del Evento**
  - Mapa Georreferenciado de San José del Álamo
  - Zona UTM: 28N
  - Escala: 1:10.000
  - Equidistancia: 5m
- **Modalidades**
  - Oscore-S1:
    - Mapa: San José del Álamo
    - Inicio: 10/06/2018 10:30
    - Fin: 10/06/2018 11:30
    - Hora máxima con penalización: 10/06/2018 12:00
    - Punto de Salida: https://goo.gl/maps/GuGnjCUGr3y
  - Oscore-S2:
    - Mapa: San José del Álamo
    - Inicio: 10/06/2018 10:35
    - Fin: 10/06/2018 11:40
    - Hora máxima con penalización: 10/06/2018 12:10

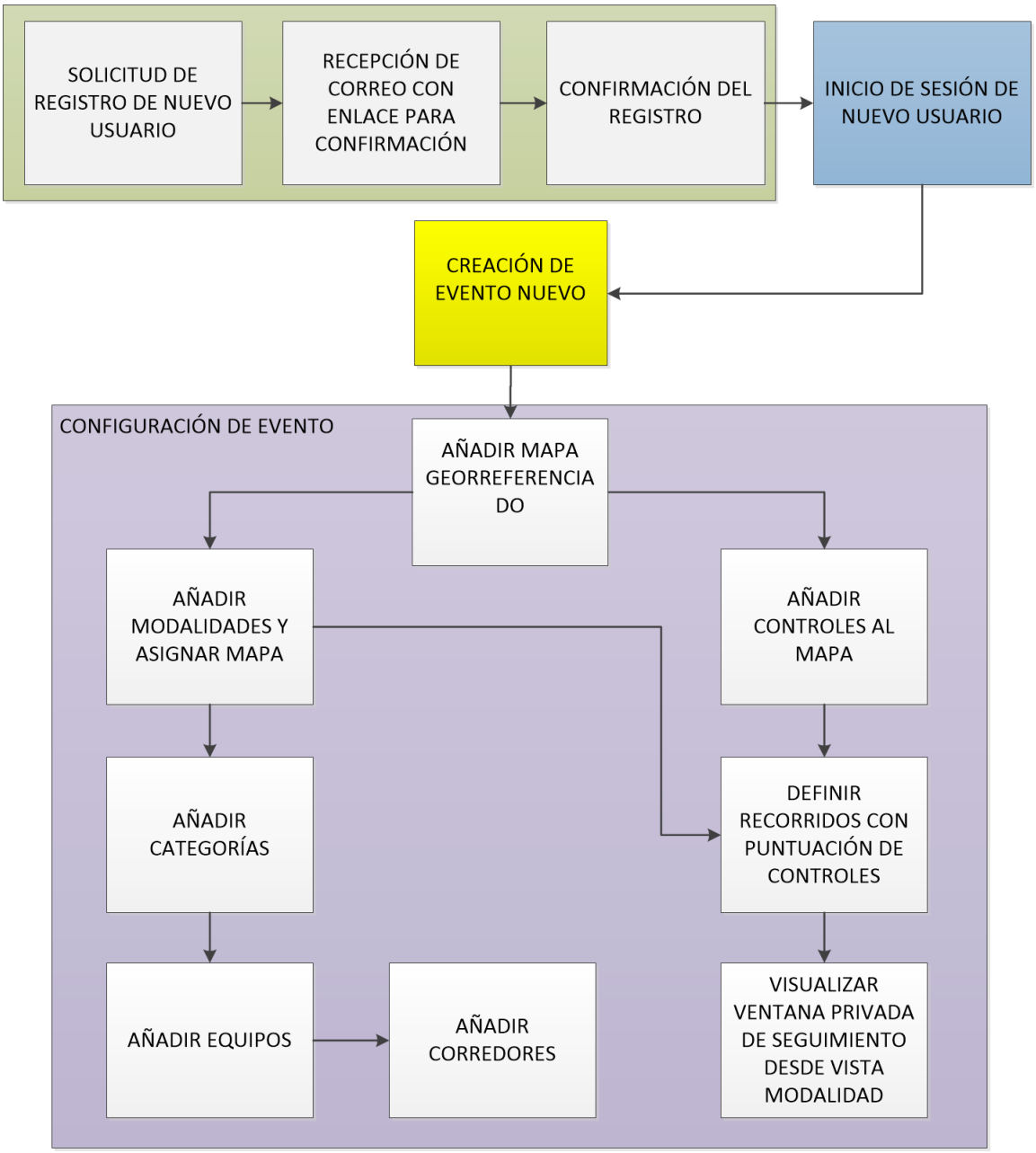


Figura 9.1: Ciclo completo creación de evento

- Punto de Salida: https://goo.gl/maps/GuGnjCUGr3y

- **Categorías**

  - ME: modalidad Oscore-S1
  - M21: modalidad Oscore-S2
  - M35: modalidad Oscore-S3
  - M45: modalidad Oscore-S4

- **Controles**

  - Cuatro controles 31, 32, 33 y 34

- **Recorridos**

  - Un recorrido por modalidad con distinta puntuación para cada uno de los controles figura 9.2

(a) Oscore-S1 (b) Oscore-S2

Figura 9.2: Recorridos ciclo de pruebas

- **Equipos**

  - Equipo 1:
    - Dorsal: 101
    - Categoría: ME

    - Modalidad: Oscore-S1
    - Corredores: Gastón Mora
- Equipo 2:
    - Dorsal: 102
    - Categoría: ME
    - Modalidad: Oscore-S1
    - Corredores: Manuel López

Resultados

El resultado de esta primera prueba con la aplicación en producción ha sido, en líneas generales, exitoso. Se ha podido realizar todo el ciclo planteado sin problemas graves ni errores bloqueantes (estos son los que impiden la finalización de dicha configuración) del evento completo, pudiendo ver como resultado final cada recorrido con sus controles configurados en la vista de seguimiento privada. Sin embargo, se detectan algunas deficiencias de usabilidad, control de lógica y otros aspectos:

- Si bien, la persona que ha testado el portal tiene conocimientos de Rogaining, se le ha tenido que dar explicaciones sobre algunos conceptos y el uso de determinados controles. Para una versión final, es imprescindible añadir una ayuda, manual de uso o similar que guíe al usuario principiante en la utilización de la herramienta.
- Problemas con la codificación de caracteres de los correos de registro y confirmación enviados: Se observa en la lectura desde dispositivos móviles del correo enviado que la codificación del mismo genera problemas de visualización de caracteres acentuados.
- Se detectan deficiencia de control de coherencia de datos en algunos puntos de la herramienta:
    - Al añadir una modalidad permite introducir fechas de inicio y fin de seguimiento, inicio de seguimiento público y finalización máxima permitida

no coherentes entre si ni con la fecha de celebración del evento. Por ejemplo: Un fecha de fin, anterior a una fecha de inicio. Esto tampoco genera un error en el servidor.

- Permite añadir dos corredores a un equipo con la misma prioridad de seguimiento. Esto genera un error en el servidor que no es informado al usuario.
- Permite añadir dos controles con el mismo código. Esto genera un error en el servidor que no es informado al usuario.

- Al añadir un nuevo control en el mapa no se ve la localización del resto de controles ya añadidos. Esta característica facilitaría el uso de la herramienta para planificación de recorridos y no solo implementación.
- Es necesario, cuando el proceso de carga de información se ralentiza, de incluir algún indicador de carga para que el usuario sepa que se está recibiendo información del servidor.
- Se detecta problemas en la vista de seguimiento al acceder a la ventana de seguimiento privado de dos modalidades distintas, los controles de la primera no son eliminados del array de controles y se muestra juntos con los de la segunda. Es necesario refrescar la pantalla para solucionar este *bug.*
- En líneas generales se ha detectado algunos errores de AngularJS y del servidor con alguna de las operaciones en la que si no se tiene la consola de depuración abierta, el usuario no obtiene ninguna información sobre el error. Es necesario la implementación de un controlador de errores que informe al usuario cada vez que se produzca uno.

9.2. Pruebas de registro en evento y seguimiento desde emulador Android

En esta batería de pruebas se utiliza el emulador de Android integrado en el entorno de desarrollo con un dispositivo virtual Nexus 5 API 23 (Android 6.0 Mars-

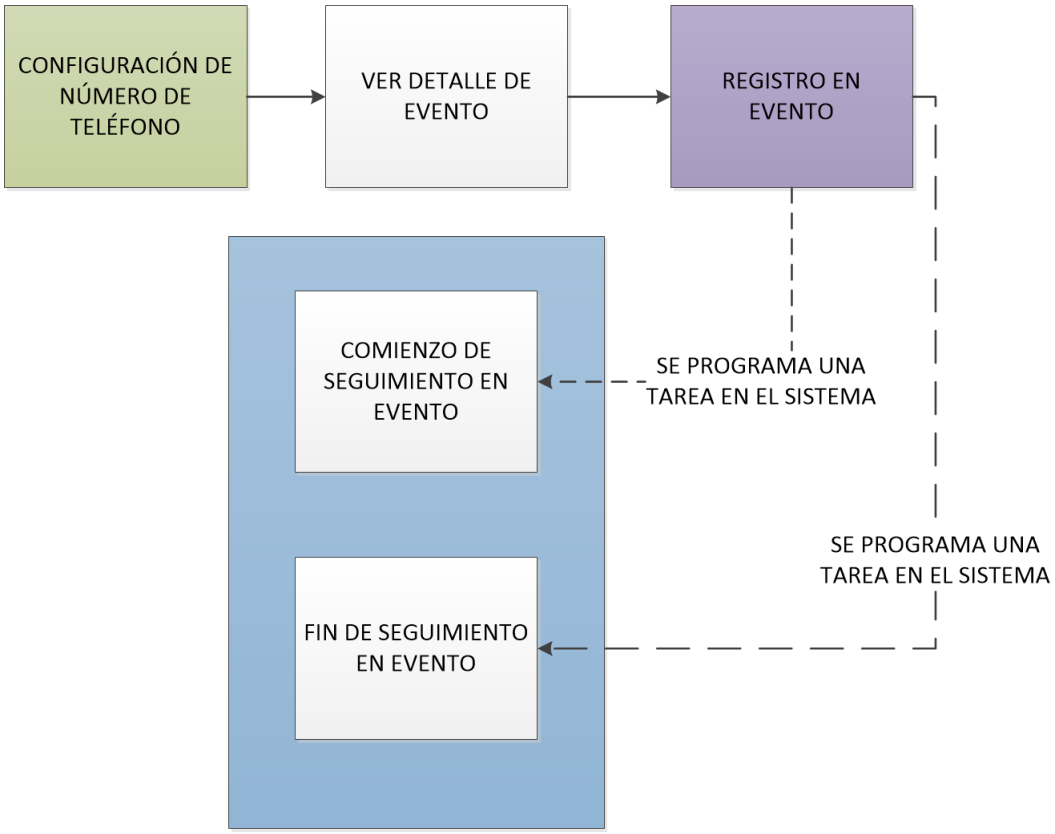


Figura 9.3: Ciclo de registro en evento desde dispositivo virtual Android

hallow) x86, con la aplicación conectada al servidor http://server.naturalsw.com.

Los pasos seguidos para la realización de las pruebas son los indicados en la figura 9.3

Se instala la aplicación, se indica el número de teléfono y se registra en el evento San José del Álamo con el número de teléfono asociado al corredor Manolo López en la batería de pruebas anterior y se espera a que salte automáticamente las tareas programadas de comienzo de seguimiento y fin de seguimiento respectivamente.

Para simular el envío de posiciones en tiempo real se utiliza la funcionalidad del emulador de Android que permita la carga de un *track* en formato *.gpx* para simular la adquisición de posiciones por el GPS del dispositivo. El *track* empleado corresponde a la zona de carrera.

Al mismo tiempo se abre desde el portal Web la vista de seguimiento privado para comprobar que es posible realizar el seguimiento [Figura 9.4].

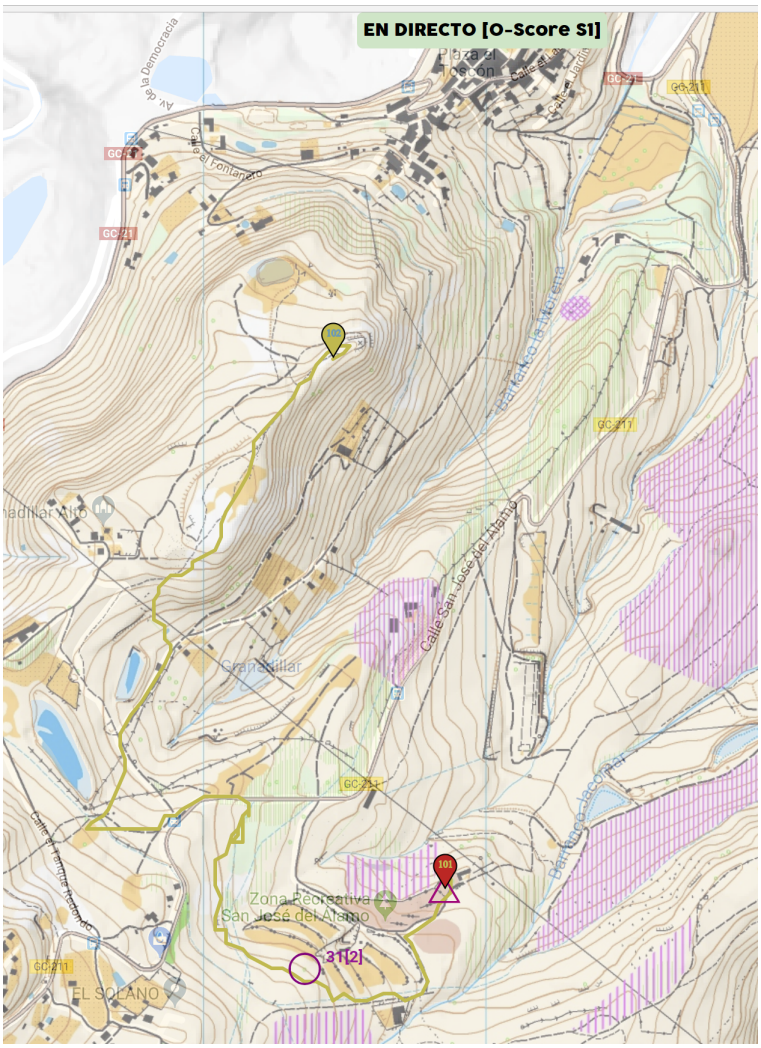


Figura 9.4: Vista de seguimiento en tiempo real exitosa

Resultados

La simulación con un solo dispositivo virtual ha sido exitosa pudiendo, desde la vista de seguimiento del portal Web, ver el desplazamiento del Equipo 2 por el mapa a medida que avanzaba el tiempo. Además se ha podido manipular con éxito los controles de la vista de seguimiento como avanzar y retroceder el *slider,* cambiar la velocidad del seguimiento o la longitud del *track* mostrado, alternar el mapa base utilizado, añadir o quitar información asociada al equipo o centrar el mapa en un equipo concreto.

Sin embargo se ha detectado algunas deficiencias que deben ser subsanadas:

- La asociación de un dispositivo a un corredor cuando no coincide el teléfono registrado en el dispositivo con ninguno de los almacenados para los corredores del evento. En este caso, si bien tras la modificación desde el portal Web del teléfono de un corredor para que coincida con el del dispositivo registrado se produce la asociación, no siempre llega la notificación al dispositivo del corredor, no completándose el registro en el mismo y no configurándose correctamente las tareas de seguimiento.

- Se detectan excepciones que se producen en determinadas secuencias de registro y cancelación de registro en un evento.

- La duración de las modalidades indicada en la aplicación móvil difiere de la real cuando la zona horaria del dispositivo no es la misma. Se debe revisar el tratamiento de las fechas para subsanar este problema.

9.3. Pruebas de registro en evento y seguimiento desde dispositivo real con sistema operativo Android

Se realiza la misma secuencia de pruebas de la sección anterior en un *smartphone* real Samsung Galaxy S5 con sistema operativo Android Marshallow 6.0.1. Por falta de tiempo, la prueba con el dispositivo real se han realizado utilizando la app Lockito

[36] y el modo desarrollador del móvil que permiten definir una ruta previa y utilizar las posiciones de dicha ruta para simular posiciones capturadas por el GPS del dispositivo.

Resultados

La simulación con un dispositivo real para un solo equipo ha funcionado de manera similar a la realizada con el emulador en el paso anterior, permitiendo realizar el seguimiento de un equipo en real durante la prueba.

Hasta el momento de elaboración de esta memoria, no ha sido posible la realización de pruebas con una muestra significativa de corredores y dispositivos que permita determinar la capacidad real del sistema, los posibles problemas con diferentes versión del sistema operativo o fallos por incompatibilidad con determinadas aplicaciones instaladas en el terminal. Queda por lo tanto, como posible trabajo futuro, la realización de test del sistema con dispositivos reales en un evento con un número significativo de participantes.

Capítulo 10

Conclusiones y trabajo futuro

En líneas generales y tras la implementación de una primera versión funcional que ha permitido pasar las pruebas realizadas con éxito, el nivel de satisfacción con el resultado del presente proyecto es bueno, aunque no se ha conseguido presentar una versión de la herramienta lista para ser utilizada en producción por organizadores de eventos, ya que requiere pulir algunas deficiencias detectadas durante la fase de pruebas y funcionalidades no implementadas.

Sin embargo, el planteamiento de la solución basado en tres componentes independientes cuyo nexo de unión lo forma un contrato común definido a través de la API, dota de flexibilidad a la aplicación pudiendo centrarse en modificar una parte sin afectar al resto. Con la idea de hacer realidad en un futuro este proyecto, esta arquitectura permite poder dividir el trabajo en tres equipos de desarrollo especializados en cada una de las tecnologías y lenguajes utilizados.

La elección de las herramientas de trabajo para el desarrollo de las tres partes del sistema ha sido un acierto, aumentando la productividad y permitiendo utilizar un marco de trabajo común. Del mismo modo, si bien el autor ya tenía experiencia en la utilización de Java y Spring para el desarrollo de aplicaciones de escritorio, Web *services* y aplicaciones Web, la elección del *framework* AngularJS , no utilizado anteriormente, para la implementación del portal Web ha abierto una nueva ventana de posibilidades en el desarrollo de este tipo de aplicaciones, simplificando mucho la complejidad que suponía la manipulación del DOM sin la utilización del mismo.

Como se ha expuesto anteriormente y especificado en los apartados 6.3, 7.3 y

8.3, ha habido una serie de características que no han podido implementarse en esta primera versión funcional. De cara a un posible trabajo futuro, el desarrollo de dichas funcionalidades es prioritario para tener una base robusta sobre la que seguir trabajando. De todas ellas, la peor carencia detectada es la falta de atención (por la limitación de tiempo expuesta) a una gestión automatizadas de las pruebas y premisas de funcionamiento del sistema que permitan ir lanzando versiones estables y testadas progresivamente, debiendo ser prioritario en el caso de continuidad del proyecto.

Por otra parte, a lo largo del desarrollo del presente proyecto, han ido surgiendo ideas de nuevas funcionalidades para la aplicación que se consideran interesantes para una futuro sistema real en producción, desatendido y destinado a la organización de pruebas reales. Se exponen a continuación algunas de las más relevantes:

- **Añadir posibilidad de importaciones de datos desde ficheros estándar**: Prácticamente todos los programas empleados en el cronometraje de pruebas de orientación, utilizan una serie de formatos de hojas de cálculo o XML para poder intercambiar información entre ellos o facilitar la tarea de los organizadores a la hora de trasladar inscripciones o recorridos generados desde otra plataforma. A tal fin, y para agilizar la labor de configuración de eventos y reducir la duplicidad de trabajo para el organizador, se considera interesante el añadir la característica de poder importar desde unos tipos de fichero de hojas de cálculo o XML definidos la siguiente información:
  - Inscripciones de equipos y corredores
  - Categorías y Modalidades
  - Controles del Mapa
  - Recorridos
- **Posibilidad de asignación manual por parte del administrador de control localizado:** Añadir la posibilidad de que aunque el sistema no haya considerado que un control ha sido localizado a través de sus cálculos de umbrales, el gestor del evento, pueda asignar un control de forma manual a un equipo para que se considere como localizado.

- **Sistema de mensajes para comunicación con el corredor a través de la aplicación:** Aprovechando las funcionalidades que se han implementado a través del servicio Firebase Cloud Messaging para enviar a los dispositivos notificaciones automáticas sobre el cambio de su estado de registro, cambios en la modalidad o categoría inscritos, etc. se plantea implementar un mecanismo desde el portal Web y pasando por el *backend* a través del cuál el organizador pueda mandar mensajes personalizados a distintos grupos de destinatarios: un corredor específico, todos los inscritos en un evento o sólo los corredores de una modalidad o categoría.

- **Desarrollo de la aplicación móvil para otros Sistemas Operativos:** Si bien los teléfonos con sistema operativo Android tienen una cuota de mercado muy importante, si se desea llegar a un mayor número de destinatarios y convertir el sistema en una alternativa real de uso al seguimiento con dispositivos GPS específicos, es imprescindible implementar la aplicación móvil al menos para dispositivos con sistema operativo IOS.

- **Integración del sistema para admitir también dispositivos de seguimiento dedicado:** Aunque para pruebas no oficiales puede llegar a ser una alternativa, la autorización a utilizar este sistema en pruebas oficiales del campeonato del mundo organizadas por la IRF puede llegar a complicarse por la prohibición explícita que existe en el reglamento de la federación internacional de utilizar dispositivos *smartphone* durante la carrera. Con el *frontend* y el *backend* implementados y funcionando, añadir la posibilidad al sistema de capturar la posición reportada por dispositivos de seguimiento GPS dedicados como hacen los sistemas actuales se intuye relativamente sencillo. Esto dotaría de un valor añadido a la solución al permitir convivir ambas alternativas.

- **Control de cierre de la aplicación y/o utilización del dispositivo móvil:** La implementación de estas funcionalidades para poder supervisar el uso fraudulento del dispositivo durante la carrera con el fin de obtener una ventaja respecto a otros competidores, permitiría acercar la posibilidad de autorización por parte de la IRF de utilización de este sistema en pruebas oficiales.

- **Posibilidad de importar el *track* de los corredores:** Se considera interesante el permitir importar el *track* registrado por los corredores desde otro dispositivo ajeno a la solución, como un reloj GPS. Este *track* podría o bien sustituir al reportado por el dispositivo móvil si es más fiable, o bien servir de alternativa a un equipo que no ha podido ser seguido durante la carrera en tiempo real pero, que integrando esta información, puede ser visualizado en diferido junto al resto de equipos.

- **Crear un portal privado para los corredores:** Crear un área privada para los corredores añadiría valor a la aplicación. En este área privada el corredor podría tener un historial de las carreras realizadas, los track en todas ellas, clasificaciones, etc. Implementada esta característica, se podría incluso gestionar las inscripciones en los eventos desde la propia plataforma pudiendo convertirse en una herramienta de gestión completa de este tipo de pruebas.

- **Actualización del Framework Web:** Desde el comienzo hasta la finalización del presente proyecto Angular ha evolucionado desde su versión 1 hasta la versión 6 que actualmente es la última vigente. En esta evolución, el *framework* ha cambiado radicalmente. Desde la forma de organización del código, implementación de componentes, servicios directivas, hasta el lenguaje principal de programación, que ha pasado de Javascript a Typescript. La actualización del código del *frontend* a la nueva versión de Angular, es necesaria para poder seguir evolucionando al mismo tiempo que lo hace el *framework* que da soporte a la misma, por lo que debe plantearse como trabajo a realizar si se desea continuar con el proyecto.

A modo de conclusión final, se considera el objetivo principal planteado al inicio de este proyecto cumplido, habiendo creado los cimientos para una alternativa asequible, viable y real a los métodos de seguimiento de carreras de orientación utilizados hasta la fecha.

Bibliografía

[1] *Orientación (deporte).* Wikipedia, la enciclopedia libre en Español[en línea].

[2] *Rogaining.* Wikipedia, la enciclopedia libre en Inglés [en línea].

[3] *Whats is Rogaining?.* Federación Internacional de Rogaining [en línea].

[4] *Especificación Internacional de Descripción de Controles.* Federación Internacional de Orientación [documento descargable] en el siguiente enlace http://orienteering.org/wp-content/uploads/2010/12/IOF-Control-Descriptions-2004.pdf

[5] *JetBrains s.r.o.* Conjunto de herramientas para el desarrollo incluidas en la Licencia gratuita para estudiantes [en línea].

[6] *IntelliJ Idea Ultimate.* IDE para Java [en línea].

[7] *DataGrip.* IDE para gestión de Bases de Datos SQL [en línea].

[8] *Toad Data Modeller.* Herramienta para diseño y modelados de bases de datos [en línea].

[9] *YouTrack.* Software de seguimiento de incidencias y gestión de proyectos [en línea]

[10] *GitLab.* Controle de Versiones y Desarrollo de Software Colaborativo [en línea]. Información en wikipedia [en línea]

[11] *Microsoft Project.* Software de planificación de proyectos [en línea].

[12] *Microsoft Visio.* Software de dibujo vectorial [en línea].

[13] *Spring Framework.* Framework para desarrollo de aplicaciones Java [en línea].

[14] *Amazon Web Services.* Capa de uso gratuito durante 12 meses [en línea].

[15] *PostgreSQL.* Sistema de base de datos relacional [en línea].

[16] *Android SDK [en línea].*

[17] *SQLite3.* Motor de base de datos [en línea].

[18] *Angular.io* Framework de desarrollo Web [en línea].

[19] *Google Maps Javascript API.* [en línea].

[20] *Google Firebase.* Plataforma de desarrollo web y movil [en línea].

[21] *Bootstrap.* Framework para diseño de sitos y aplicaciones web [en línea]

[22] *ArcGis.* Archivos de georreferenciación para datasets ráster*[en línea]*

[23] *WorldFile.* [en línea]

[24] *UTM.* Sistema de Coordenadas UTM [en línea]

[25] *Custom Overlay.* Sobre impresión de imágenes en Google Maps [en línea]

[26] *REST.* Transferencia de Estado Representacional [en línea]

[27] *Material.* Normativa de diseño desarrollada por Google [en línea]

[28] *Mobile Operating System.* Distribución de sistemas operativos en dispositivos móviles [en línea].

[29] *Android Version History.* Historial de distribución de versiones del sistema operativo Android [en línea].

[30] *Apache Maven.* Herramienta de gestión y construccion de proyectos [en línea].

[31] *AOP.* Programación Orientada a Aspectos [en línea]

[32] *SmartTable.* [en línea]

[33] *Angular Formly.* [en línea]

[34] *FCM y Android.* Configurar FCM en Android [en línea]

[35] *Nginx.* Servidor web/proxy ligero [en línea]

[36] *Lockito.* Fake GPS Itinerary [en línea]

Apéndice A

Despliegue e instalación

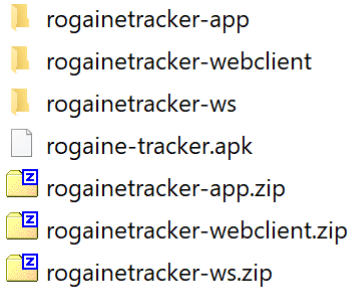
En el presente capítulo se indican las instrucciones para el despliegue de la solución en un equipo con sistema operativo Linux, distrubución CentOS 7. Las instrucciones el despliegue pueden variar de una distribución de Linux a otra. El entorno de despliegue utilizado ha sido un servidor virtualizado con las siguientes características (sección 3.2):

- CPU con 6 nucleos de 1,8GHz cada uno
- 4GB de Ram
- 160GB de espacio de almacenamiento

Se parte de la premisa que se tiene permisos de administrador en el servidor de despliegue, así como conexión a internet que permita instalar correctamente todo el software necesario. Del mismo modo, no se va a entrar en detalles de configuración de seguridad asociada al servidor y las distintas aplicaciones como reglas del firewall, certificados SSL, distintos usuarios de la base de datos, etc. ya que se desvía del objetivo del presente proyecto.

A.1. Contenido del DVD

En el DVD adjunto a la presente memoria se incluye el siguiente contenido:



- ***rogainetracker-app*** *y* ***rogainetracker-app.zip****:* Código fuente de la aplicación para dispositivos móviles con sistema operativo Android. Además se incluye un apk compilado y firmado apuntando al servidor *http://server.naturalsw.com* con el que se ha realizado las pruebas.
- ***rogainetracker-webclient*** y ***rogainetracker-webclient.zip:*** Código fuente del *frontend.*
- ***rogainetracker-ws*** y ***rogainetracker-ws.zip:*** Código fuente del *backend.*

A.2. Instalación y despliegue de la base de datos

El SGBD (Sistema de Gestión de Bases de Datos) elegido para la persistencia de los datos del servidor, como se indica en la sección 3.1.2, es PostgreSQL, por su naturaleza *SQL* y ser una herramienta *open source* y gratuita. En el momento de redacción de este capítulo, la versión estable más actual es la 10.4. Para su instalación hay que seguir los siguientes pasos:

```
#1. Añadir el repositorio que permitire instalar la última versión del SGBD
>rpm -Uvh https://yum.postgresql.org/10/redhat/rhel-7-x86_64/pgdg-centos10-10-2.
    noarch.rpm

#2. Instalar el servidor de postgresql
>yum install postgresql10-server postgresql10

#3. Inicializar la base de datos
>/usr/pgsql-10/bin/postgresql-10-setup initdb

#4. Arrancar el servicio y configurarlo para que se inicie automáticamente en el
    arranque
> service postgresql-10 start
```

```
> chkconfig postgresql-10 on

#5. Configurar una contraseña para el usuario postgres de la base datos
#5.1 Cambio de usuario
> su - postgres
#5.2 Iniciar la consola de PostgreSQL
> psql
#5.3 Definir contraseña:
>postgres=#/password postgres
>/q
```

A continuación es necesario crear la base de datos *RogaineTracker* y lanzar el script *PostgreSQLModel.sql* incluido en el DVD para la generación de todas las tablas de la aplicación. Partiendo del estado en el que queda la consola tras ejecutar los comandos del paso anterior:

```
#1. Creación de la base de datos
>createdb RogaineTracker

#2. Inicialización de la base de datos a partir del script
>psql -U postgres -d RogaineTracker -a -f PostgreSQLModel.sql
```

Por último, es necesario configurar el método de autenticación de los clientes sobre la base de datos para poder utilizar la autenticación por contraseña configurada en el *Web service*. Así:

```
#1. Editar el archivo pg_hba.conf de la carpeta /var/lib/pgsql/10/data
>cd /var/lib/pgsql/10/data
>vim
#2. En el final del archivo se reemplaza la linea:
    host    all             all             127.0.0.1/32            ident
# Por
    host    all             all             127.0.0.1/32            md5
#3. Reiniciar el servicio postgresql
> service postgresql-10 restart
```

A.3. Instalación y despliegue del servidor: *backend* y *frontend*

Instalación y despliegue del *backend*

En primer lugar, para poder compilar el proyecto, es necesario instalar el JDK de Java 8 (o superior) y Apache Maven:

```
#1. Instalar JDK Java (debe ser la versión devel para poder utilizar el plugin de
    maven con el servidor tomcat integrado)
>yum install java-1.8.0-openjdk-devel

#2. Instalar Apache Maven 3
>cd /usr/local/src
>wget http://www-us.apache.org/dist/maven/maven-3/3.5.3/binaries/apache-maven
    -3.5.3-bin.tar.gz
>tar -xf apache-maven-3.5.3-bin.tar.gz
>cd apache-maven-3.5.3

#3. Configurar Apache Maven
>cd /etc/profile.d
>vi maven.sh
#3.1 Copiar dentro del archivo la configuración de la carpeta donde se almacenará
    las dependencias descargadas
    export M2_HOME=/usr/local/src/apache-maven-3.5.3
    export PATH=${M2_HOME}/bin:${PATH}
#3.2 Guardar el archivo, asignar permisos de ejecución y configurar como source
>chmod +x maven.sh
>source /etc/profile.d/maven.sh
```

A continuación se debe subir al servidor el archivo *rogainetracker-ws.zip* de la carpeta */source*. Una vez subido, se debe descomprimir, configurar y compilar:

```
#Partiendo de la premisa de que el archivo rogainetracker-ws.zip se encuentra en la
     carpeta /usr/local/src
#1. Descomprimir el archivo
>cd /usr/local/src
>unzip rogainetracker-ws.zip
>cd rogainetracker-ws
```

Antes de compilar se debe editar el archivo *application.properties* modificando los valores de las siguientes variables:

- *jdbc.password* con la contraseña del usuario postgres establecida en la sección

anterior

- *email.** con los parámetros del servidor y la cuenta de correo que se desea utilizar para enviar los correos de registro predefinidos en la aplicacion

```
#1. Editar el archivo application.properties
>cd /usr/local/src/rogainetracker-ws/src/main/resources
>vim application.properties
#2. Cambiar el valor de las variables indicadas anteriormente
```

Y el archivo *WebConfiguration* añadiendo la URL del servidor donde se desplegará la aplicación al método *addCorsMappings*:

```
@Override
public void addCorsMappings(CorsRegistry registry) {
        registry.addMapping("/**").
             allowedOrigins("http://SERVERNAME")
            .allowedMethods("GET","POST","PUT", "DELETE","OPTIONS","HEAD");
}
```

Por último, se compila y arranca el servidor Tomcat integrado en el plugin de Maven

```
#1. Acceder a la carpeta del archivo pom.xml
>cd /usr/local/src/rogainetracker-ws/
#2. Compilar el proyecto
>mvn tomcat7:run
```

Para probar que todo funciona correctamente se puede poner en el explorador la siguiente url *http://**SERVERNAME**:8080/rogainetrackerws/public/event* donde ***SERVERNAME*** se debe sustituir por el nombre de la máquina donde se desplegará la aplicación. Si como respuesta a dicha petición obtenemos una matriz en formato JSON vacía, todo funciona correctamente.

Instalación y despliegue del *frontend*

Para poder compilar el proyecto, es necesario instalar *NodeJs*:

```
#1. Añadir el repositorio
>yum install epel-release
#2. Ejecutar el script de node para comprobar si se cumplen las condiciones para su
     instalación
>curl --silent --location https://rpm.nodesource.com/setup_6.x | bash -
```

```
#3. Instalar Nodejs
>yum install -y nodejs
```

A continuación se debe subir al servidor el archivo *rogainetracker-webclient.zip* de la carpeta */source.* Una vez subido, se debe descomprimir, configurar y compilar:

```
#Partiendo de la premisa de que el archivo rogainetracker-webclient.zip se
    encuentra en la carpeta /usr/local/src
#1. Descomprimir el archivo
>cd /usr/local/src
>unzip rogainetracker-webclient.zip
>cd rogainetracker-webclient
```

Se instalan las dependencias del proyecto

```
>npm install
```

El próximo paso es modificar el archivo *constants*.js de la carpeta *app,* para configurar la URL de la API que se ha desplegado en la sección anterior:

```
>cd /usr/local/src/rogainetracker-webclient/app
>vim constants.js
```

Por último, se inicia el *frontend:*

```
>npm start
```

Si todo ha ido correctamente, accediendo desde un navegador a la URL *http://* ***SERVERNAME,*** donde ***SERVERNAME*** se debe sustituir por el nombre de la máquina donde se desplegará la aplicación, debería aparece la ventana de bienvenida del *frontend.*

A.4. Instalación y despliegue aplicación móvil

Para despliegue de la aplicación móvil es necesario generar el archivo *.apk* e instalarlo en un dispositivo móvil. Para ello se debe:

1. Abrir el proyecto del DVD *rogainetracker-app* en Android Studio.
2. Modificar el valor de la variable *serverIP* del archivo *paths.xml* ubicado en *src/main/res/values* que contiene la URL del servidor.

3. Compilar el proyecto y generar el empaquetado de la *app* firmado para que permita su posterior instalación en un dispositivo móvil.

4. Instalar la *app* en un dispositivo móvil, y concederle los permisos solicitados. Si no aparece solicitud de permisos, se debe conceder explícitamente los permisos que necesita desde la gestión de permisos del dispositivo.