Grado en Ingeniería Informática del Software

Trabajo Fin de Grado

Sistema multiusuario para el seguimiento on-line de partidos de baloncesto

AUTOR

Cristian Lado González

TUTORA

Mª José Suárez Cabal

NOVIEMBRE 2020

Agradecimientos

A mi familia por el apoyo constante que me han dado.

A todos los amigos que me han acompañado en este camino por la vida universitaria y, en especial a tres que me han hecho todo mucho más fácil.

A mi tutora M.ª José por estar siempre disponible y dispuesta a ayudarme en todo lo referido a este TFG.

Resumen

Un marcador de baloncesto en tiempo real para poder consultarlo en cualquier momento. Este es el objetivo de este Trabajo Fin de Grado. El sistema consta de dos principales funcionalidades, por un lado, la gestión de equipos y partidos desde una aplicación web, y, por otro, el propio marcador en sí, consistente en una aplicación móvil en la que se puede hacer de oficial de mesa o de mero espectador. En este documento se expondrá todo el trabajo realizado para llegar a la solución final.

Palabras clave: Baloncesto, marcador, tiempo real, aplicación móvil, aplicación web.

Abstract

A basketball scoreboard in real time to consult it at any time. This is the objective of this Final Degree Project. The system consists of two main functionalities, team and match management from a web application, and the scoreboard itself which consists of a mobile application in which you can be a table official or a spectator. This document will expose all the work done to achieve the final solution.

Keywords: Basketball, scoreboard, real time, mobile application, web application.

Contenido

Índice de figuras

Índice de tablas

Capítulo 1. Memoria

1.1 Introducción

Este Trabajo de Fin de Grado consiste en un proyecto de desarrollo software, concretamente de un sistema multiusuario para el seguimiento on-line de partidos de baloncesto, como bien indica el título de este documento.

En los apartados siguientes de este capítulo se comentarán las necesidades del cliente, es decir, lo que el cliente espera del software a desarrollar, el ámbito y alcance del proyecto, es decir, una definición del producto resultante con sus limitaciones, y, por último, los requisitos de usuario de alto nivel, es decir, lo que podrán realizar los usuarios con el sistema una vez terminado sin entrar en mucho detalle. Además, se incluirán en este capítulo un estudio de las alternativas existentes para llegar a la solución final y cuál de esas alternativas se ha elegido finalmente, las conclusiones y las ampliaciones posibles de este proyecto.

En los sucesivos capítulos se explicará cómo se ha llevado a cabo la planificación del sistema, la elaboración del presupuesto tanto de costes como del cliente, el análisis del sistema, el plan de pruebas seguido, el diseño y la construcción del sistema y, para terminar, se incluirán los manuales de usuario necesarios para poner el sistema en funcionamiento, entre estos manuales se incluye el de despliegue, el de instalación y el de usuario.

1.2 Objetivos

El objetivo principal es desarrollar un sistema multiusuario para el seguimiento on-line de partidos de baloncesto. El sistema debe proveer la posibilidad tanto de retransmitir las acciones de los partidos a través de una aplicación móvil como de poder visualizar dichas acciones en directo o diferido. Además, se debe proveer de una gestión de equipos y partidos desde una aplicación web.

1.3 Necesidades del cliente

Muchas personas que acuden a ver un partido de baloncesto no pueden acceder a la información que ofrece el marcador, bien porque no está encendido o bien porque no está visible desde la parte en la que se encuentran. Este problema se repite sobre todo en partidos de categorías inferiores. Para resolverlo se ha encargado la realización de un sistema informático multiusuario para el seguimiento on-line de partidos de baloncesto. El sistema debe permitir tanto introducir toda la información relacionada con los partidos como la visualización de dicha información por parte de varios usuarios. Además, deberá ser un sistema en tiempo real para que la información ofrecida esté lo más actualizada posible. Otra necesidad por cubrir es ver la información de partidos ya jugados con anterioridad, una vez ya terminados, y poder consultar clasificaciones de competiciones y categorías.

1.4 Ámbito y alcance

El sistema consistirá en un sitio web en tiempo real para el seguimiento on-line de partidos de baloncesto y una aplicación móvil para poder “retransmitir” el partido para que se pueda seguir a través de la propia aplicación.

Los partidos estarán organizados por categorías, ligas, competiciones, etc. Para poder acceder a alguno de ellos habrá que elegir primero el apartado en el que se encuentra. También podrá haber partidos que no pertenezcan a ninguna categoría, liga o competición y solo habrá que elegir alguno para poder acceder a ellos.

Lo utilizarán tres tipos de usuarios, los seguidores, los oficiales de mesa y los administradores que será el encargado de introducir toda la información relativa a los partidos para así poder dar información a los usuarios registrados que utilicen el sistema.

Los seguidores podrán hacer uso del sistema para registrarse y poder así pasar a ser usuarios registrados, los cuales, tras identificarse, podrán elegir qué partido quieren seguir y una vez elegido podrán visualizar todas las acciones del partido, como el tiempo, el cuarto en el que está, los puntos, las faltas personales de cada jugador, los tiempos muertos de cada equipo y las faltas de equipo.

El oficial de mesa, a través de la aplicación móvil, deberá poder poner en marcha el tiempo, sumar los puntos que anote cada equipo, restarlos en caso de que el árbitro lo indique tras el Instant Replay, añadir tiempo si así lo dice el árbitro, marcar las faltas personales y de equipo y los tiempos muertos de cada equipo.

El administrador, a través de la aplicación web, será el encargado de configurar todo lo relativo a las competiciones, categorías, equipos y partidos, como se ha mencionado anteriormente.

1.5 Requisitos de usuario de alto nivel

R1. El usuario no registrado deberá poder registrarse en el sistema.

R2. El usuario registrado deberá poder identificarse para poder entrar en el sistema.

R3. El seguidor podrá elegir el partido del que quiere hacer el seguimiento.

R4. El seguidor podrá ver todas las acciones del juego.

- R4.1. El cuarto en el que se encuentra el partido.
- R4.2. El tiempo que queda.
- R4.3. Los tiempos muertos de cada equipo.
- R4.4. Los puntos de cada equipo.
- R4.5. Las infracciones de ambos equipos.
- R4.6. Las faltas personales de cada jugador.
- R4.7. Los cambios de jugadores de ambos equipos.

R5. El seguidor podrá ver los datos de partidos ya finalizados anteriormente.

- R5.1. El seguidor podrá ver un resumen de los puntos y faltas personales de los jugadores de un partido ya finalizado.

R6. El seguidor podrá ver las clasificaciones de los equipos.

- R6.1. Las clasificaciones se actualizarán automáticamente una vez finalice un partido.

R7. El administrador podrá introducir los datos de una competición manualmente.

- R7.1. Si existe alguna fuente de datos con los datos de la competición se podrá importar automáticamente.

R8. El administrador podrá introducir los datos de una categoría manualmente.

R8.1. Si existe alguna fuente de datos con los datos de la categoría se podrá importar automáticamente.

R9. El administrador podrá introducir los equipos.

R9.1. Podrá introducir equipos que estén dentro de una competición y/o categoría.

R9.2. Podrá introducir equipos que no pertenezcan a una competición ni a una categoría.

R10. El administrador podrá crear un partido.

R10.1. Podrá crear un partido suelto sin pertenecer a ninguna competición.

R10.2. Podrá crear un partido para una competición y categoría en concreto.

R11. El administrador podrá configurar un partido creado manualmente.

R12. El administrador podrá activar un partido una vez esté creado.

R13. El administrador podrá modificar un partido que no esté activo.

R14. El administrador podrá eliminar un equipo que no esté participando en ningún partido.

R15. El oficial de mesa se encargará de recoger todas las acciones que el usuario puede visualizar.

R15.1. Empezar un partido.

R15.2. Poner en marcha el cronómetro del sistema.

R15.3. Añadir tiempo si es conveniente.

R15.4. Añadir o quitar puntos.

R15.5. Añadir o quitar las faltas personales y las de equipo.

R15.6. Añadir un tiempo muerto a cualquiera de los dos equipos.

R15.7. Añadir otras infracciones que puedan ocurrir en el partido.

R15.8. Finalizar un cuarto.

R15.9. Empezar un cuarto.

R15.10. Añadir un cambio de jugadores.

R15.11. Finalizar un partido.

R15.11.1. Una vez finalice el partido, si existe una clasificación para la competición a la que pertenece, se actualizará automáticamente.

R16. El oficial de mesa podrá suspender un partido que se esté jugando.

R17. El oficial de mesa podrá retomar un partido suspendido para continuar haciendo mesa en ese partido.

R18. El oficial de mesa podrá abandonar un partido para que otro oficial de mesa haga mesa en ese partido.

R18.1. El oficial de mesa que abandona le envía una invitación al otro oficial de mesa.

R18.2. El oficial de mesa abandona el partido y lo puede retomar cualquier otro oficial de mesa.

1.6 Análisis y selección de alternativas

Actualmente existen algunas páginas web en las que se pueden consultar resultados de baloncesto en directo. Algunas de ellas son mismarcadores.com, marcadoresonline.com y livescore.in, [Consultadas el 9 de febrero de 2020] pero solo partidos de grandes ligas o competiciones. En todas estas páginas mencionadas se puede elegir la liga que se quiere visualizar, hacer un seguimiento de los partidos en directo, consultar clasificaciones y consultar partidos ya finalizados en la fecha que se quiera. Una de las características de la que carecen estos sistemas es la de la posibilidad de que un administrador pueda incluir partidos. Además, son páginas en las que no solo se centran en resultados de este deporte, sino que hay otros deportes más, lo que puede ser un poco engorroso para aquellas personas que solo quieran centrarse en resultados de baloncesto.

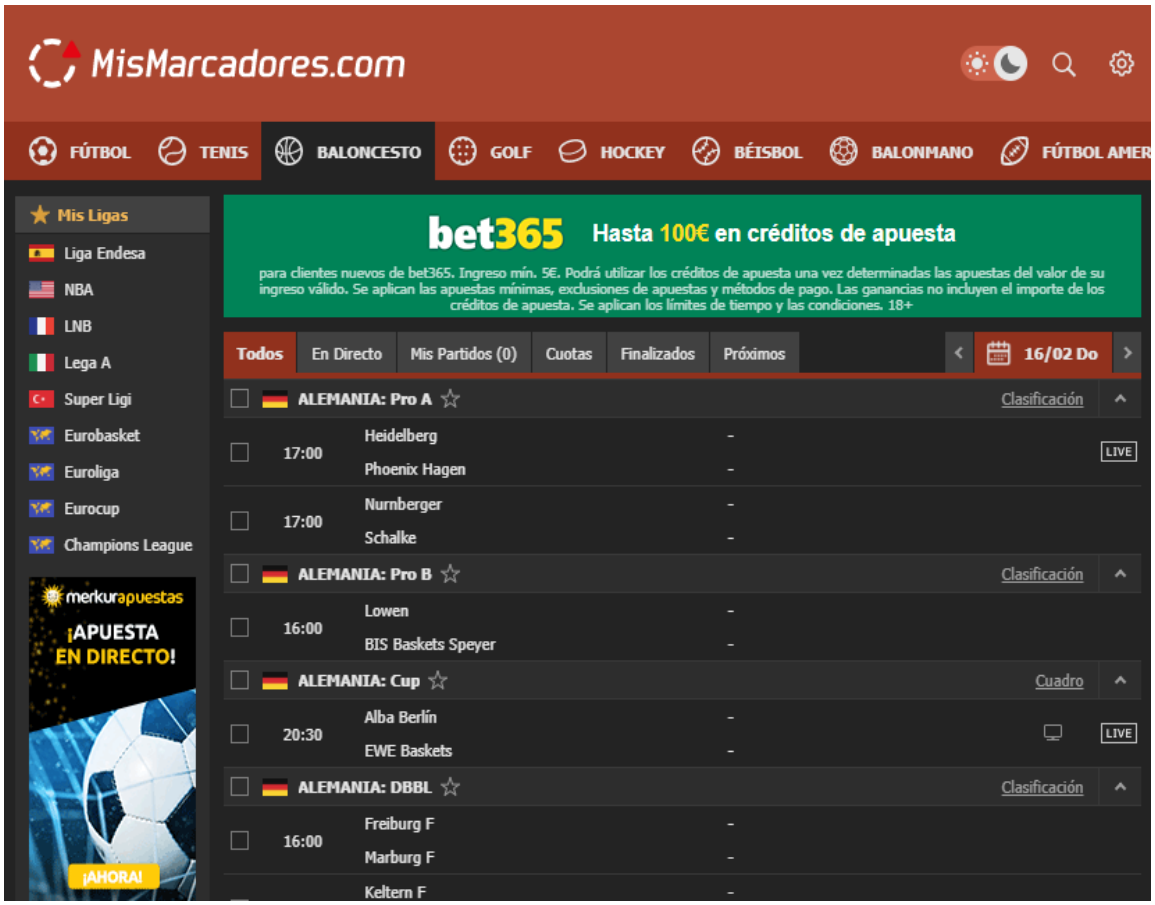


Figura 1. Visualización de los próximos partidos en la web mismarcadores.com

También existen aplicaciones móviles que sirven de marcador de baloncesto, pero sería para recoger la información de un único partido y monousuario, es decir, no se podría compartir con otros usuarios, solo se puede ver en el propio dispositivo que tiene instalada la aplicación.

Existe una aplicación móvil llamada Tanteo [12], disponible en Google Play y Apple Store parecida a lo que se quiere llegar a conseguir con este sistema. Alguna diferencia en cuanto a lo que se quiere implantar es la posibilidad de visualizar el cronómetro del partido y restar puntos,

ya que en esta aplicación existente no es posible como se puede ver en la Figura 2. Otra ausencia de funcionalidad de dicha aplicación con respecto a este sistema es la visualización de los datos del equipo rival, es decir, solo puedes administrar las acciones de uno de los equipos. En cuanto a los tiempos muertos, no se especifica un máximo por cuarto ni se puede marcar quién lo ha solicitado.

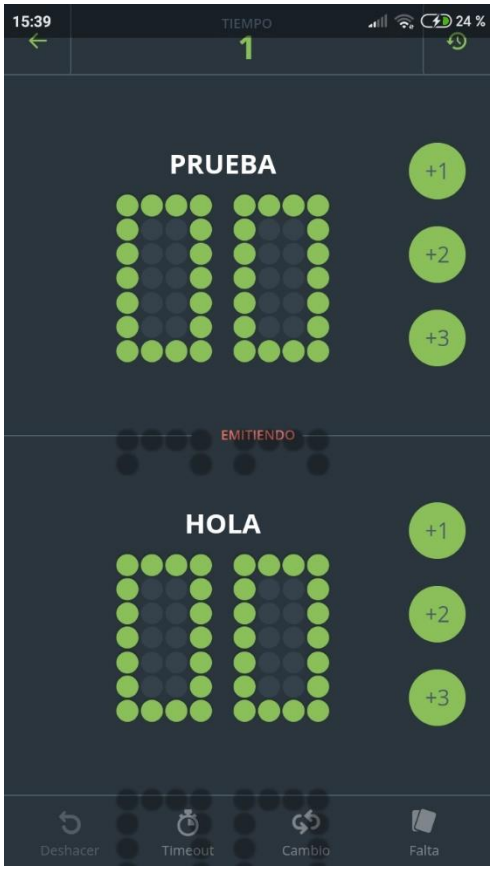


Figura 2. Seguimiento de un partido en directo en la aplicación Tanteo

En el caso del sistema que se va a implantar, se va a centrar más en partidos de categorías inferiores, aunque pueda haber partidos de grandes competiciones, por lo que no existiría ninguna alternativa conocida a este sistema en el mercado. Además, con el sistema final se podría llevar el tanteo de los partidos y poder compartirlo a más usuarios para que puedan seguir el resultado del partido, bien en directo o una vez ya finalizado.

1.7 Estudio de tecnologías y selección final

En este apartado se estudiarán las diferentes alternativas que se han barajado para poder llevar a cabo la realización del sistema, además de explicar por qué se rechazan unas en favor de otras y cuál ha sido la alternativa seleccionada finalmente para este sistema.

1.7.1 Tecnologías para aplicaciones móviles

Para empezar, se han estudiado las diferentes opciones que hay actualmente en la programación de aplicaciones móviles [1] ya que el administrador deberá utilizar una aplicación móvil. Lo ideal en este caso sería poder utilizar la aplicación móvil del sistema en las diferentes plataformas que existen, las principales son Android, iOS y Windows.

Xamarin

Una de las alternativas que existen es la herramienta Xamarin ya que se reutiliza gran parte del código para las diferentes plataformas. La interfaz de usuario es la nativa en cada plataforma y la lógica de negocio se realiza en el lenguaje C#, con lo que se puede aprovechar toda la lógica cuando se quiera desarrollar la aplicación en otra plataforma, únicamente modificando la interfaz de usuario. Un inconveniente a esta herramienta es el absoluto desconocimiento de su funcionamiento ya que en el grado no se da en ninguna asignatura que haya cursado.

PhoneGap

Otra de las alternativas es un desarrollo basado en HTML5 utilizando PhoneGap/Apache Cordova que permite crear la aplicación utilizando lenguajes conocidos como HTML, CSS y JavaScript. La desventaja es que genera aplicaciones híbridas y no tienen un rendimiento similar a las aplicaciones 100% nativas ya que se visualiza en web y no por interfaces específicas de los sistemas operativos [6].

Es una posibilidad buena para las personas con conocimientos en desarrollo web ya que se utilizan los mismos lenguajes y si lo que se quiere conseguir no es una interfaz nativa de cada sistema operativo móvil, sino que se conforma con un visionado web.

Android Studio

Por último, en cuanto a la aplicación móvil, existen herramientas nativas de cada sistema operativo como Android Studio para Android. Es cierto que no se llega a todas las plataformas de esta manera, pero como se menciona en el artículo [2] de las referencias bibliográficas, un 90% de los dispositivos móviles en todo el mundo utiliza el sistema operativo Android, por lo que utilizando la herramienta Android Studio se llegaría a una gran mayoría de personas. Además de esto, esta herramienta se puede utilizar en cualquier dispositivo para programar la aplicación. La mayor de las ventajas es el conocimiento, aunque muy básico, se adquirió en una optativa del grado.

El inconveniente, como se ha dicho anteriormente, es no llegar al 100% de los dispositivos móviles, lo cual reduce la utilización del sistema por parte del administrador a un dispositivo Android.

Swift

Para iOS existe Swift. Si se quiere programar para esta plataforma haría falta un ordenador MAC. La aplicación resultante solo se podría utilizar en dispositivos iOS lo cual reduce también la utilización del sistema, pero, en este caso, se reduciría mucho más que utilizando cualquiera de las otras herramientas mencionadas.

Resumen

El problema de las herramientas nativas es el no poder llegar a todos los públicos ya que el código no se podría reutilizar y habría que hacer dos aplicaciones distintas si se quiere utilizar distintas plataformas. Pero se gana en rendimiento ya que no es lo mismo utilizar herramientas ajenas al sistema operativo que utilizar las propias.

El resumen de las características de todas las herramientas mencionadas y que puede hacer que se decida por una o por otra se puede ver en la siguiente tabla:

	Xamarin	PhoneGap	Android Studio	Swift
Nativa	**Sí**	No	**Sí**	**Sí**
Lenguaje	C#	HTML, JS, CSS	Java	Swift
Lenguaje conocido	**Sí**	**Sí**	**Sí**	No
Cualquier plataforma	**Sí**	**Sí**	No	No
Reutilizar código	**Sí**	**Sí**	No	No
Herramienta conocida	No	No	**Sí**	No

Tabla 1. Comparación resumen entre herramientas móviles

1.7.2 Tecnologías para aplicaciones web

En cuanto a la aplicación web que utilizarán los usuarios no registrados, registrados e identificados, se han estudiado principalmente dos alternativas que se están usando mucho actualmente, Spring Boot y Node.js [3].

Spring Boot

Esta alternativa, cuyo lenguaje es Java, permite liberar la mayor parte del trabajo de configuración que llevan las aplicaciones Spring [7]. Es una buena elección para aplicaciones stand-alone, es decir, aplicaciones que se ejecutan por sí solas sin necesidad de otras aplicaciones, pero para este sistema no es el caso. Una ventaja de Spring Boot frente a Node.js es el multi-hilo, pero como este sistema no tendrá una gran carga computacional no sería muy necesaria esta característica.

Node.js

Es un entorno de ejecución basado en JavaScript [8]. Se utiliza para aplicaciones en tiempo real ya que es asíncrono y puede realizar una tarea mientras se realiza otra. También tiene un conjunto de módulos que, normalmente, se instalan con Node Package Manager (npm) que facilita la compilación, la instalación, la actualización de estos módulos y la gestión de dependencias. Uno de los módulos interesantes para este trabajo es Socket.io, que permite la comunicación en tiempo real entre servidor y cliente.

1.7.3 Tecnologías para almacenamiento de datos

Para almacenar los datos del sistema se han tenido en cuenta dos tipos de bases de datos, las relacionales y las NoSQL.

Bases de datos relacionales

Las bases de datos relacionales representan los datos en forma de tablas. Cada una de las tablas tiene unos campos en los que se almacenan los datos y todas ellas están relacionadas entre sí. Las principales ventajas de este tipo de bases de datos son la evitación de repetir registros duplicados, ya que los Sistemas de Gestión de Bases de Datos permiten configurar que no se puedan incluir datos repetidos, y garantizan la integridad ya que si se elimina un registro se eliminan también todos los relacionados con él. El principal inconveniente es el tener que rellenar todos los campos o dejarlos vacíos si un registro no lo posee, lo cual ocuparía espacio en memoria sin ser utilizado.

NoSQL

Este tipo de bases de datos no utilizan estructuras uniformes en las que almacenar los datos, sino que se basan en datos almacenados como un par clave-valor. La principal desventaja es que no garantizan las propiedades ACID (atomicidad, consistencia, aislamiento y durabilidad) pero la gran ventaja es la búsqueda rápida de registros y la no obligación de mantener todos los campos en todos los registros.

1.7.4 Selección final

Teniendo todo esto presente se ha decidido elegir una herramienta nativa para la aplicación móvil ya que se adapta mucho mejor al sistema operativo. En este caso se ha elegido Android Studio ya que se ha dado unas nociones muy básicas en la optativa de Realidad Aumentada y Accesibilidad, pero han servido para familiarizarse un poco con la herramienta. El mayor inconveniente es la limitación a dispositivos Android, pero, como se ha expuesto anteriormente, se calcula que el 90% de los dispositivos móviles cuentan con este sistema operativo. En cuanto a la aplicación web se ha decidido utilizar Node.js ya que se ha visto en la asignatura de Sistemas Distribuidos e Internet y es un entorno en el que resulta muy fácil consumir servicios web. En cuanto al almacenamiento de datos se ha optado por una MongoDB, una base de datos NoSQL, ya que el sistema no necesitará almacenar algunos datos de objetos del mismo tipo, además de soportar bastante bien el uso de Node.js.

1.8 Conclusiones

El objetivo de este Trabajo Fin de Grado era el desarrollo de un sistema multiusuario para el seguimiento on-line de partidos de baloncesto. Tras todo el trabajo realizado se ha conseguido alcanzar el objetivo básico propuesto satisfactoriamente.

La elección de las tecnologías adecuadas para la realización de este desarrollo ha sido clave ya que ha sido todo un poco más fácil, a pesar de haber encontrado algún contratiempo a la hora de empezar el desarrollo como, por ejemplo, la no integración de Android con una base de datos MongoDB [22]. Al final estos contratiempos se han salvado buscando soluciones, que, al fin y al cabo, es la labor de la Ingeniería, la búsqueda de soluciones a problemas reales.

En todo este trabajo que lleva detrás la realización de este TFG se puede ver reflejado el aprendizaje recibido en estos cuatro años de Grado y una gran ilusión por poder llegar a conseguir el objetivo propuesto.

1.9 Ampliaciones

Si bien es cierto que el objetivo básico del trabajo se ha alcanzado surgen algunas ampliaciones que podrían llegar a convertirlo en un sistema mucho más potente. Estas ampliaciones están descritas en el backlog ya que no se ha conseguido realizar todo lo que se especifica en él y se ha dejado alguna funcionalidad extra para futuras iteraciones del producto. Algunas de estas ampliaciones a modo de resumen podrían ser:

- Incluir categorías y competiciones para agrupar los partidos.
- Suspensión de partidos.

- Abandono de oficiales de mesa de los partidos.
- Utilizar datos de fuentes de información externas existentes para ampliar la información que ya se da.

Incluso este sistema podría servir para los oficiales de mesa reales para ir anotando las acciones en él y luego poder transmitirlas a la federación correspondiente. Para este caso se podrían reutilizar muchos de los componentes realizados.

Capítulo 2. Planificación

Para realizar este proyecto se seguirá una metodología ágil, SCRUM, en la que el desarrollo se dividirá en sprints en los que se va incrementando la funcionalidad del sistema.

La planificación de las tareas se divide en 6 grandes actividades. Para empezar, están las tareas iniciales que consisten en la preparación del proyecto en sí. Entre estas tareas están definir las necesidades del cliente, determinar el ámbito y el alcance, obtener los requisitos de alto nivel, el estudio de viabilidad del sistema, el planteamiento de la arquitectura, la preparación de entornos de trabajo y la creación del backlog.

Las siguientes 3 tares constan de los diferentes sprints a realizar, cada uno de ellos de unas 40 horas. Cada uno de estos sprints se dividen a su vez en análisis, desarrollo, integración, pruebas unitarias y pruebas de sistema. El análisis está formado por elegir las historias de usuario que se realizarán durante ese sprint y la estimación de dichas historias de usuario. La integración, por su parte, se subdivide en integración de componentes y pruebas de integración. Tanto estas pruebas como las unitarias y las de sistema se subdividen en diseño y ejecución. A su vez las de sistema se componen de funcionales y de seguridad.

Como quinta actividad están las pruebas de sistema que incluyen las pruebas de rendimiento y carga y las pruebas de usabilidad. Como se ha mencionado anteriormente estas pruebas también se dividen en diseño y ejecución.

Por último, se encuentran las pruebas de aceptación, en las que el cliente dará la aprobación del sistema en su conjunto o no.

Se ha considerado trabajar 4 horas al día durante 4 días a la semana, lo que conlleva un total de unos 5 meses, es decir un total de 320 horas. La fecha de comienzo está planificada para el 5 de enero de 2020 y la fecha de fin para el 30 de mayo de 2020.

En la siguiente figura se puede ver, en un diagrama de Gantt, el resumen de la planificación de cada una de estas actividades y tareas, así como las precedencias entre ellas.

Proyecto	**41,88 días**
Tareas iniciales	**18,5 días**
Definir necesidades del usuario	0,13 días
Determinar ámbito y alcance	0,25 días
Obtener requisitos de usuario de alto nivel	**0,38 días**
Estudio de viabilidad del sistema	**3 días**
Planteamiento de la arquitectura	0,06 días
Preparación de entornos de trabajo	0,13 días
Crear backlog	0,5 días
Sprint 1	**6,63 días**
Análisis	**0,19 días**
Desarrollo	5 días
Integración	**5,5 días**
Integración de componentes	5 días
Pruebas de integración	**5,5 días**
Pruebas unitarias	**6,25 días**
Pruebas funcionales	**6,25 días**
Pruebas de sistema	**6,63 días**
Pruebas funcionales	**6,63 días**
Pruebas de seguridad	**6,63 días**
Sprint 2	**6,81 días**
Sprint 3	**6,56 días**
Pruebas de sistema	**3 días**
Pruebas de rendimiento y carga	**1,5 días**
Pruebas de usabilidad	**2 días**
Pruebas de aceptación	3 días

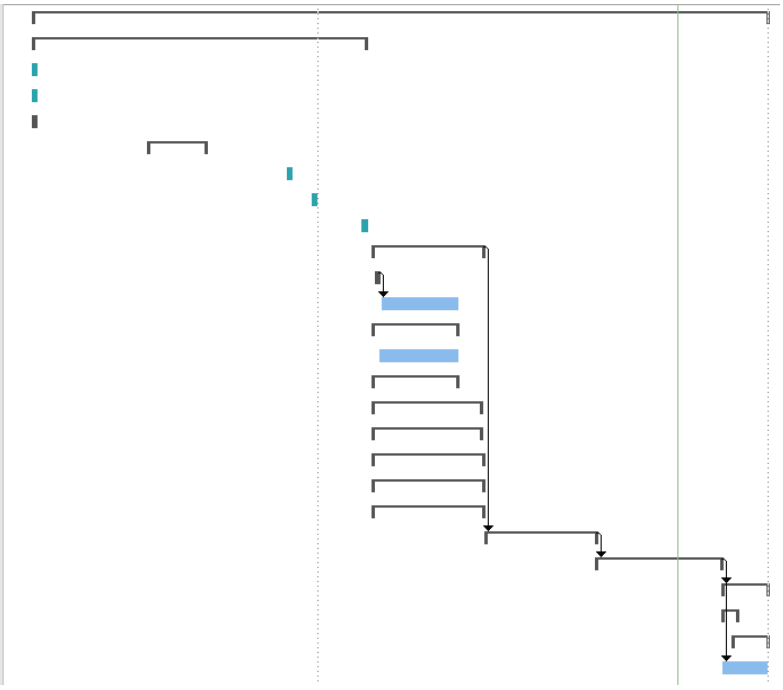


Figura 3. Planificación de las tareas y actividades del proyecto y diagrama de Gantt

Capítulo 3. Presupuesto

Para realizar el presupuesto se han tenido en cuenta 5 roles principales que intervienen en el proyecto:

- Ingeniero de software
- Consultor de tecnología
- Arquitecto del software
- Analista funcional
- Programador

Para cada uno de ellos se ha calculado su sueldo bruto al año tomando como referencia la media española de cada uno de estos perfiles proporcionada por indeed.es [14] y, a partir de ese sueldo bruto, se ha calculado el coste salarial de cada trabajador utilizando la calculadora disponible en la herramienta de billin.net [13]. Como trabajan 4 horas al día, 4 días a la semana se ha reducido el sueldo bruto a la mitad en comparación a la media española a jornada completa. En la siguiente tabla se muestran los resultados obtenidos.

Recursos de trabajo que intervienen en el proyecto				
Personal	Sueldo Bruto año	Coste salarial Año	Coste salarial mes	TOTAL
Ingeniero de software	16.556,00 €	21.560,24 €	1.796,69 €	1.796,69 €
Consultor de tecnología	16.202,50 €	21.047,05 €	1.753,92 €	1.753,92 €
Arquitecto de software	19.913,50 €	25.867,64 €	2.155,64 €	2.155,64 €
Analista funcional	16.419,00 €	21.328,28 €	1.777,36 €	1.777,36 €
Programador	13.485,00 €	17.517,01 €	1.459,75 €	1.459,75 €
TOTAL				8.943,35 €

Tabla 2. Recursos de trabajo que intervienen en el proyecto

Para calcular el total del proyecto de cada perfil se ha calculado la parte proporcional a los días que trabaja por semana durante los 5 meses del proyecto. A continuación, se ha calculado el porcentaje de la productividad en el proyecto de cada uno de los perfiles y, poder así, calcular el coste directo que supone ese trabajador en el proyecto y el coste indirecto. Los porcentajes de productividad se han calculado en base a las horas que trabaja cada perfil en el proyecto. Estos detalles se pueden ver en la tabla de Trabajos no productivos de los Apéndices.

Por otro lado, se han calculado los costes indirectos relativos a servicios y equipos y licencias que supone el proyecto.

En el caso de los servicios se ha calculado el coste al mes de cada uno de ellos, el plazo en meses que dura el proyecto, en este caso 5 meses, y el coste total durante el período de tiempo que dura el proyecto.

Para los equipos y licencias se han especificado las unidades necesarias de cada ítem, el precio, el coste total de todas las unidades, el plazo en meses de amortización para el proyecto, el coste al mes y el coste total que se imputa al proyecto.

Costes indirectos			
Servicio	Coste mes	Plazo (meses)	Coste total
Electricidad	70,00 €	5	350,00 €
Agua	15,00 €	5	75,00 €
Internet	53,00 €	5	265,00 €
Telefonía móvil	10,00 €	5	50,00 €
Material de oficina	10,00 €	5	50,00 €
TOTAL	158,00 €		**790,00 €**

Tabla 3. Costes indirectos de servicios del proyecto

Costes indirectos							
Equipo / Licencia	Unidades	Precio	Coste total unidades	Plazo (años)	Plazo (meses)	Coste mes	Coste total
Portátil	01	400,00 €	400,00 €	4	48	8,33 €	41,67 €
Teléfono móvil	01	150,00 €	150,00 €	2	24	150,00 €	750,00 €
Ratón	01	6,00 €	6,00 €	3	36	6,00 €	30,00 €
Silla	01	60,00 €	60,00 €	5	60	1,00 €	5,00 €
Mesa	01	120,00 €	120,00 €	5	60	2,00 €	10,00 €
Licencia de estudiante WebStorm	01	0,00 €	0,00 €	-	-	0,00 €	0,00 €
Licencia GNU AGPL de MongoDB	01	0,00 €	0,00€	-	-	0,00 €	0,00 €
Repositorio OneDrive	01	0,00 €	0,00 €	-	-	0,00 €	0,00 €
Repositorio GitHub	01	0,00 €	0,00 €	-	-	0,00 €	0,00 €
TOTAL			736,00 €			167,33 €	**836,67 €**

Tabla 4. Costes indirectos de equipos y licencias del proyecto

Con todo esto se obtiene un total de costes indirectos de 1.626,67€.

Servicios	790,00 €
Equipos/licencias	836,67 €
Costes indirectos	1.626,67 €

Tabla 5. Total de costes indirectos

Una vez calculados los costes directos que suponen los empleados y los costes indirectos se calculan los beneficios deseados, que en este caso serán del 25%, con lo que la cuantía de beneficios asciende a 54.038,51€.

Costes directos	35.773,41 €
Costes indirectos	1.626,67 €
Costes directos e indirectos	37.400,07 €
Beneficio deseado (25%)	9.350,02 €
Coste total (directos, indirectos y beneficios)	46.750,09 €

Tabla 6. Resumen del coste total del proyecto

En el apéndice Horas de trabajo y facturación se muestran los cálculos realizados y los precios/hora para obtener la facturación deseada que es de 46.751,66€.

Estos precios/hora calculados no son los que se utilizarán para calcular el presupuesto del proyecto ya que se han ajustado con los beneficios. Se utilizarán los precios/hora sin el 25% de beneficio como se muestra en la siguiente tabla.

Precio / hora (sin beneficios)	
Personal	Precio / hora
Ingeniero de software	47,85 €
Consultor de tecnología	41,33 €
Arquitecto de software	49,95 €
Analista funcional	41,25 €
Programador	33,75 €

Tabla 7. Precio/hora de los perfiles sin beneficios

A continuación, se muestra una tabla con un resumen de todos los cálculos realizados y con el margen entre el coste total y la facturación posible.

Concepto	Importe
Coste total (directos, indirectos y beneficios)	46.750,09 €
Facturación posible	46.751,66 €
Margen entre coste total y facturación	0,34%

Tabla 8. Resumen de todos los costes, facturación posible y margen entre el coste total y la facturación

El presupuesto se divide en 5 partidas principales de acuerdo con la planificación realizada:

- Tareas iniciales
- Sprint 1
- Sprint 2
- Sprint 3
- Pruebas finales

En la parte de Desglose del presupuesto de costes de los Apéndices se puede ver en más detalle los cálculos realizados para calcular los costes de cada una de las partidas.

El resumen del presupuesto interno de costes queda reflejado en la tabla que se muestra en el apartado Resumen del presupuesto de costes de los Apéndices.

En el presupuesto de costes no se incluye el beneficio deseado por lo que en el presupuesto del cliente se añade el 25% de beneficios deseado. En la siguiente tabla se muestra el presupuesto del cliente resumido con el beneficio incluido sin el 21% de IVA. Para una mayor información

sobre el desglose del presupuesto del cliente se puede consultar el apéndice Desglose del presupuesto del cliente.

Cod.	Partida	Total	Beneficio	Coste total
01	Tareas iniciales	1.183,50 €	295,88 €	1.479,38 €
02	Sprint 1	10.547,33 €	2.636,83 €	13.184,16 €
03	Sprint 2	10.547,33 €	2.636,83 €	13.184,16 €
04	Sprint 3	10.547,33 €	2.636,83 €	13.184,16 €
05	Pruebas finales	3.635,40 €	908,85 €	4.544,25 €
	TOTAL (sin IVA)	36.460,88 €	9.115,22 €	45.576,09 €

Tabla 9. Resumen del presupuesto del cliente

Aplicando el 21% de IVA que se debe incluir en este tipo de productos, el coste total sería de 55.147,07€.

Total (con IVA)	55.147,07 €

Capítulo 4. Análisis del sistema

4.1 Requisitos funcionales del sistema

Al seguir la metodología SCRUM para desarrollar el sistema, los requisitos funcionales se recogen en un backlog o pila del producto en la que los requisitos se representan mediante historias de usuario. Estas historias de usuario siguen un formato "Como <tipo de usuario> quiero <funcionalidad> para <objetivo>" y todas ellas componen la funcionalidad completa del sistema. Además, estas historias de usuario van acompañadas de criterios de aceptación en los que se establecen las restricciones que debe cumplir cada funcionalidad.

Para elaborar el backlog del producto con las historias de usuario y sus criterios de aceptación obtenidas a partir de los requisitos de usuario. Para definir estas historias de usuario se han identificado tres roles principales: administrador, oficial de mesa y seguidor. Todos ellos podrán registrarse en el sistema e iniciar sesión.

El administrador se encargará de toda la gestión de equipos, jugadores, partidos, competiciones y categorías.

El oficial de mesa será el encargado de introducir todas las acciones de los partidos para que así los seguidores puedan hacer el seguimiento de ellos.

El seguidor podrá hacer el seguimiento de partidos en directo, partidos que estén en el histórico o partidos que aún no se hayan jugado. Además, podrán elegir competiciones y/o categorías para ver los partidos pertenecientes a ellas. Por último, también tienen la posibilidad de consultar las clasificaciones de las distintas competiciones y categorías.

El backlog resultante con las historias de usuario y los criterios de aceptación es el siguiente:

HISTORIAS DE USUARIO	CRITERIOS DE ACEPTACIÓN
Como administrador quiero introducir equipos para poder crear todo tipo de partidos.	Información obligatoria de cada equipo: * nombre del equipo * lugar donde juega como local. * nombre del entrenador. Información opcional: * nombres de los jugadores con sus dorsales. No puede haber dos equipos con el mismo nombre. Si no se asignan dorsales, se asociarán al equipo los dorsales desde 00 hasta 99. No puede haber dos jugadores con el mismo dorsal en el mismo equipo.

HISTORIAS DE USUARIO	CRITERIOS DE ACEPTACIÓN
Como administrador quiero crear un partido que no pertenezca a una competición para poder hacer el seguimiento.	Información obligatoria: * equipos participantes * número de cuartos * duración de cada cuarto * número de tiempos muertos máximo por cada equipo y cuarto * número máximo de faltas por jugador * fecha * hora * lugar * tiemo corrido o no Información opcional: * oficial de mesa El partido se crea con estado "creado".
Como administrador quiero activar un partido una vez creado para que los usuarios identificados puedan visualizarlo.	Listar partidos creados por el administrador en estado "creado" o "suspendido" Al seleccionar un partido, cambia a estado "activo".
Como oficial de mesa quiero elegir un partido activo para poder introducir las acciones.	Listar partidos en estado "activo" sin oficial de mesa o "activos" de ese oficial. Al seleccionar un partido, se asocia el oficial de mesa con dicho partido. Todas las acciones sobre el partido deben ser realizadas por este oficial de mesa.
Como oficial de mesa quiero introducir los nombres de los jugadores que falten.	El partido debe estar en estado "activo". Se introducen los nombres con los dorsales si el dorsal no tiene jugador asociado en el equipo.
Como oficial de mesa quiero poner en marcha el cronómetro para iniciar el partido.	El partido debe estar en estado "activo". Una vez iniciado el partido pasará al estado "en juego". El tiempo irá disminuyendo y se visulizará solo el tiempo que queda para terminar el cuarto. El tiempo se mostrará actualizado en todo momento en la pantalla del oficial de mesa. Mostrar evento de inicio de partido a los seguidores.
Como oficial de mesa quiero parar el cronómetro cuando se pare el partido sin informar de ninguna acción.	Se podrá parar si: * el cronómetro está corriendo * el partido no es a tiempo corrido Mostrar evento de parar cronómetro a los seguidores.
Como oficial de mesa quiero añadir tiempo al cronómetro.	Se puede añadir tiempo si: * el cronómetro está parado * el partido no es a tiempo corrido

HISTORIAS DE USUARIO	CRITERIOS DE ACEPTACIÓN
Como oficial de mesa quiero reanudar el cronómetro cuando se vuelva a reanudar el partido.	Se podrá volver a poner en marcha desde dónde se había parado si: * el cronómetro está parado * el partido no es a tiempo corrido Mostrar evento de reanudación a los seguidores.
Como oficial de mesa quiero añadir puntos a ambos equipos para ir actualizando el resultado.	Aumentar de uno en uno, de dos en dos o de tres en tres. Indicar jugador. Se actualizará el marcador y se mostrará a los seguidores automáticamente. Mostrar evento a los seguidores. Si no es a tiempo corrido parar el tiempo.
Como oficial de mesa quiero quitar puntos a ambos equipos para ir actualizando el resultado.	Quitar puntos de uno en uno a un equipo. Indicar jugador. Se actualizará el marcador y se mostrará a los seguidores automáticamente. Mostrar evento a los seguidores. Mostrar cronómetro a los seguidores.
Como oficial de mesa quiero añadir faltas personales a los jugadores que las cometan.	Se parará el crono cuando se cometa una falta. Al añadir una falta personal a un jugador se le añadirá una falta a su equipo. Un jugador podrá cometer como máximo el número de faltas personales asignadas en el partido. Mostrar evento de falta personal a los seguidores. Mostrar cronómetro a los seguidores.
Como oficial de mesa quiero quitar faltas personales a un jugador.	Eliminar de una en una faltas personales de un jugador. No se registra la eliminación. Mostrar evento a los seguidores. Mostrar cronómetro a los seguidores.
Como oficial de mesa quiero añadir tiempos muertos a cada equipo.	Se parará el cronómetro. Cuando se acabe el tiempo muerto el oficial de mesa pondrá en marcha el cronómetro como en una parada de tiempo normal. El número de tiempos muertos por cuarto de cada equipo no puede superar el número máximo. Mostrar evento a los seguidores.

HISTORIAS DE USUARIO	CRITERIOS DE ACEPTACIÓN
Como oficial de mesa quiero añadir otras infracciones para informar a los seguidores del partido de este tipo de acciones.	Se informará de: * pasos * dobles * campo atrás * posesión * 5 segundos * 3 segundos * 8 segundos * pie * antideportiva (añadir al jugador y al equipo, 2 son expulsión) * técnica (añadir al jugador o entrenador y al equipo, 2 son expulsión) Parar el cronómetro. No se incluyen en faltas personales ni de equipo salvo las mencionadas. Mostrar cronómetro a los seguidores.
Como oficial de mesa quiero acabar un cuarto.	El tiempo tiene que estar acabado. Mostrar evento a los seguidores.
Como oficial de mesa quiero iniciar un cuarto.	Se pone en marcha el cronómetro automáticamente al iniciarlo. El contador de faltas de equipo se inicializa a cero. El contador de tiempos muertos se inicializa a cero. Mostrar evento a los seguidores.
Como oficial de mesa quiero introducir un cambio de un jugador para mostrarlo a los seguidores.	El tiempo tiene que estar parado. Se indica el jugador que sale y el que entra. Mostrar evento a los seguidores. Mostrar cronómetro a los seguidores.
Como oficial de mesa quiero finalizar un partido para incluirlo en el histórico.	El partido pasa a estado "finalizado". Se tienen que haber acabado todos los cuartos. La clasificación se debe actualizar otorgando los puntos configurados en la creación de la competición al equipo ganador y al equipo perdedor. Mostrar evento a los seguidores.
Como seguidor quiero elegir un partido activo para seguirlo y ver todas las acciones que acontecen.	El seguidor recibirá información de todas las acciones que se produzcan (puntos, faltas, tiempos muertos...). Registrar seguidor en el partido.
Como administrador quiero modificar los datos de un partido.	Partido en estado "creado" y del administrador Se podrán modificar todos los datos.
Como administrador quiero eliminar un equipo para que no aparezca en el sistema.	El equipo no puede estar en ningún partido.

HISTORIAS DE USUARIO	CRITERIOS DE ACEPTACIÓN
Como usuario no registrado quiero registrarme para poder autenticarme en el sistema.	Información obligatoria: * nombre de usuario * contraseña * repetición de contraseña El nombre de usuario no puede estar repetido en la aplicación y la contraseña y la repetición de contraseña deben coincidir.
Como usuario registrado y no identificado quiero autenticarme en el sistema con mis datos de registro para acceder a la funcionalidad del sistema.	Información obligatoria: * nombre de usuario * contraseña
Como seguidor quiero elegir un partido en concreto ya finalizado para ver un resumen.	Se muestra: * equipos * marcador * faltas personales de jugadores * técnicas * antideportivas * puntos de jugadores
Como seguidor quiero ver los partidos por jugar para ver un resumen.	Se muestra: * equipos * fecha * hora * lugar
Como seguidor quiero ver el tiempo del cronómetro para saber cuánto queda.	Se pide el tiempo y se muestra el cronómetro como una captura en ese momento.
Como oficial de mesa quiero suspender un partido antes de acabar sus tiempos.	El partido pasa a estado "suspendido". Se desvincula el oficial de mesa del partido.
Como oficial de mesa quiero continuar el seguimiento de un partido que había sido suspendido previamente.	Se muestran los partidos en estado "activo" que no tienen oficial de mesa y los activos del oficial de mesa. Se retoma con los datos que se habían quedado.
Como administrador quiero introducir los datos de una competición para crearla manualmente.	Información obligatoria: * duración de cada cuarto * número de tiempos muertos por equipo y cuarto * número máximo de faltas personales por jugador * puntos clasificatorios del equipo perdedor/ganador/empate Información opcional: * equipos * categorías (prevalecen las normas de esta)

HISTORIAS DE USUARIO	CRITERIOS DE ACEPTACIÓN
Como administrador quiero introducir los datos de una categoría para crearla manualmente.	Información obligatoria: * duración de cada cuarto * número de tiempos muertos por equipo y cuarto * número máximo de faltas personales por jugador * puntos clasificatorios del equipo perdedor/ganador/empate Información opcional: * equipos
Como administrador quiero crear un partido automáticamente con los datos de la competición o categoría a la que pertenece y de los equipos.	Información obligatoria: * competición/categoría * equipos participantes * dorsales de los jugadores si no tienen asignado * fecha * hora * lugar
Como seguidor quiero ver los partidos de una competición y categoría para ver las acciones.	Seleccionar competición y categoría. Listar los partidos en orden de fecha. Para cada partido se indicará su estado "activo", "en juego", "finalizado" o "suspendido".
Como seguidor quiero ver los partidos de una fecha para visualizar los partidos jugados o por jugar.	Elegir una fecha ya pasada o futura. Se muestran los partidos en estado "activo", "en juego", "suspendido" o "finalizado" de esa fecha.
Como seguidor quiero ver los partidos de una jornada para visualizar los partidos jugador o por jugar.	Elegir una jornada ya pasada o futura. Se muestran los partidos en estado "activo", "en juego", "suspendido" o "finalizado" de esa fecha.
Como seguidor quiero ver la clasificación de los equipos.	Se podrá elegir la clasificación de que competición o categoría se quiere visualizar. Se muestran: * nombre de equipos ordenados por posición * puntos de cada equipo * número de partidos ganados y perdidos de cada equipo
Como oficial de mesa quiero abandonar un partido que se esté jugando para que otro pueda seguir en él.	El partido sigue en estado "en juego". El partido aparecerá en la lista de partidos activos para que un oficial de mesa recoja las acciones desde donde se dejó.
Como oficial de mesa quiero enviar una invitación a otro oficial de mesa para que pueda recoger las acciones de un partido.	Introducir nombre de usuario del otro oficial para traspasar la recogida de las acciones del partido cuando esté en estado "en juego" o "activo". El oficial de mesa invitado no puede estar en otro partido "en juego".
Como administrador quiero consultar una fuente de información disponible con la información de una competición y/o categoría para crearla automáticamente.	Si existe una fuente de información en la que esté la información necesaria para crear una competición o categoría se podrán crear automáticamente sin tener que introducir los datos manualmente.

Tabla 10. Backlog

Todas estas funcionalidades quedan recogidas de una forma más visual y específica en el story mapping de las siguientes figuras. Las fichas azules corresponden a los roles, las fichas amarillas, a las funcionalidades de más alto nivel y las rosas a las funcionalidades más específicas o historias de usuario. Por último, se muestran fichas grises que se dejarían para un desarrollo posterior.

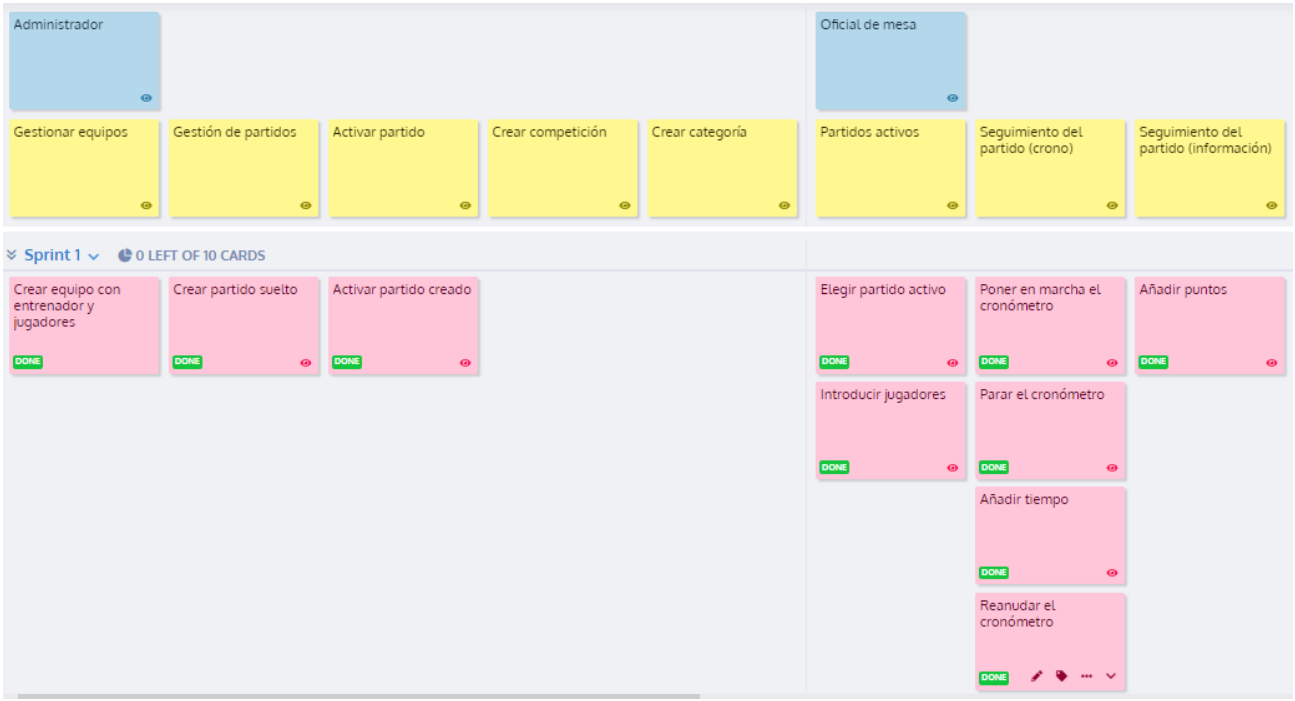


Figura 4. Story mapping con las historias de usuario del Sprint 1

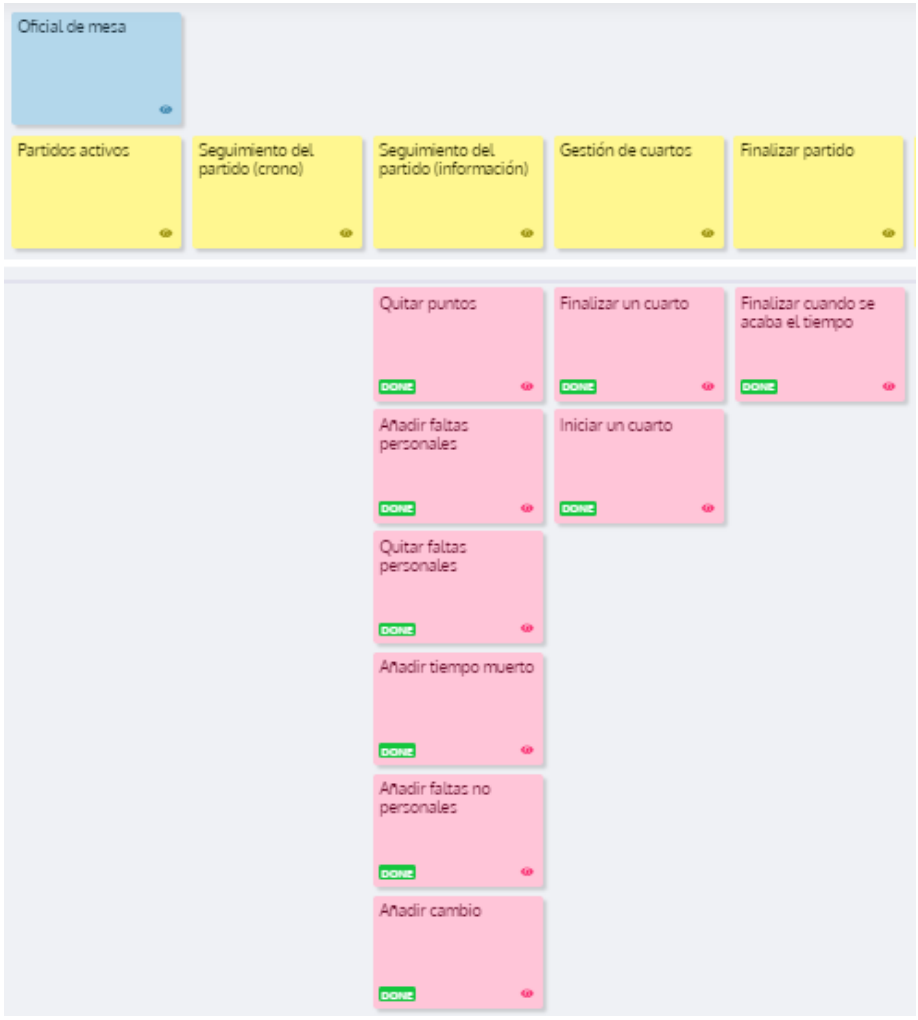


Figura 5. Story mapping con las historias de usuario del Sprint 2

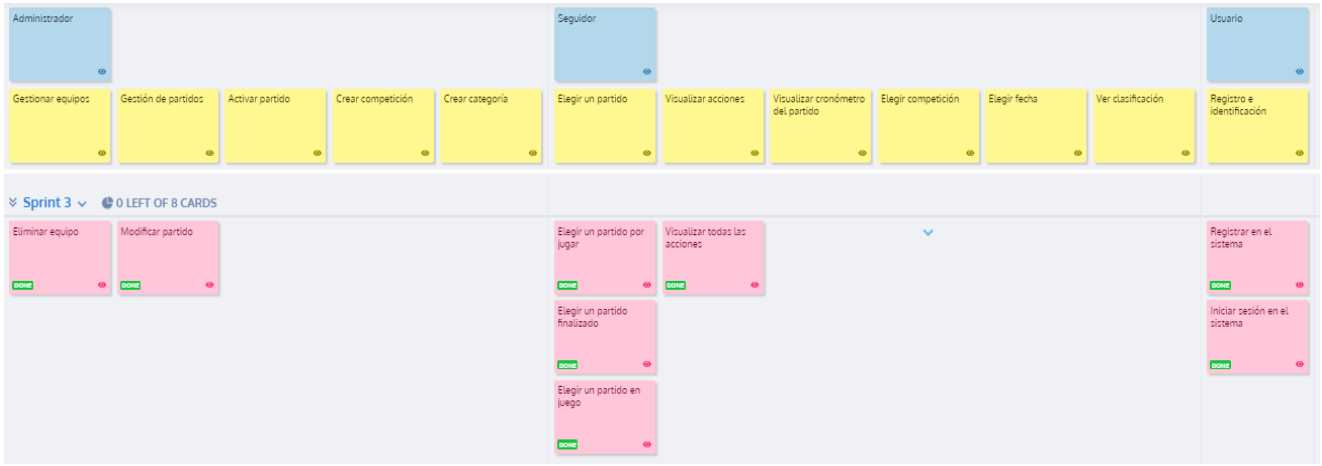


Figura 6. Story mapping con las historias de usuario del Sprint 3

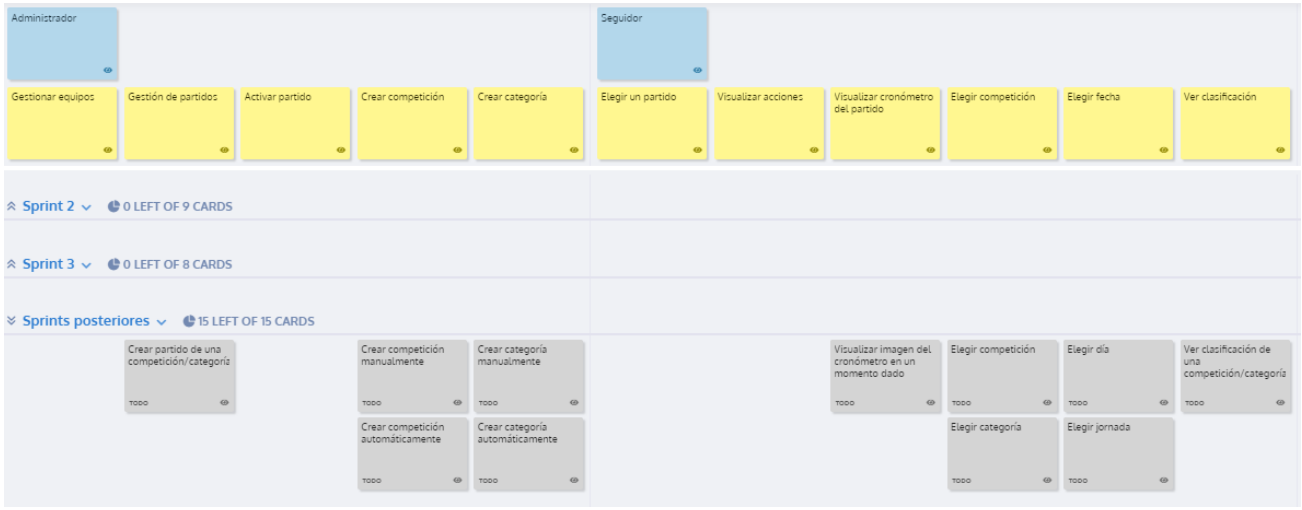


Figura 7. Story mapping con las historias de usuario pendientes (I)

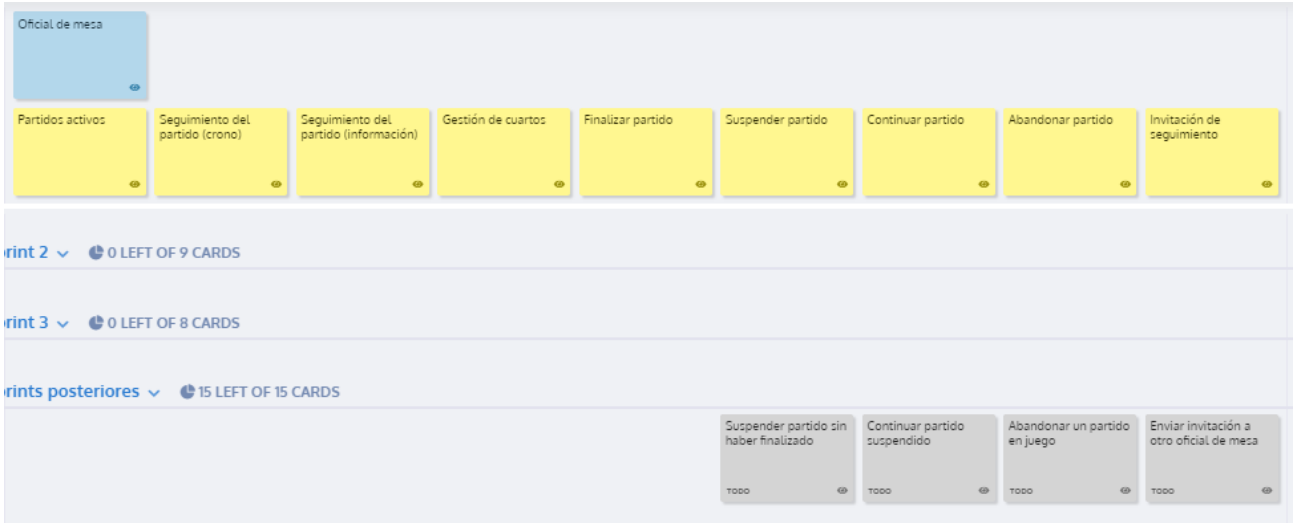


Figura 8. Story mapping con las historias de usuario pendientes (II)

4.2 Requisitos no funcionales

En este apartado se mostrarán los requisitos no funcionales que debe poseer el sistema para cumplir con atributos de calidad de rendimiento, eficiencia y seguridad entre otros.

RNF1. El sistema debe soportar una carga de 100 personas simultáneamente.

RNF2. Más del 65% de las respuestas del sistema deben realizarse en menos de 5 segundos.

RNF3. El acceso a la funcionalidad del sistema solo se podrá llevar a cabo por un usuario identificado.

RNF4. Todas las contraseñas deben estar encriptadas en la base de datos.

RNF5. La aplicación web se deberá poder visualizar correctamente en los navegadores Firefox, Microsoft Edge, Chrome y Opera.

RNF6. La aplicación web se desarrollará únicamente para visualizarse en navegadores con Javascript.

RNF7. La aplicación móvil se deberá poder visualizar correctamente en todo tipo de dispositivos Android.

RNF8. El sistema debe proveer mensajes de error informativos al usuario.

RNF9. La metodología de desarrollo será SCRUM.

RNF10. El sistema no permitirá la introducción incorrecta de datos.

4.3 Modelo de clases de dominio

El modelo de clases de dominio se representa utilizando el lenguaje UML.

Un partido tiene un creador siempre, que será el usuario que lo cree, y un partido siempre tiene dos equipos, el equipo local y el visitante. Además, un partido tiene también otros usuarios vinculados como el oficial de mesa y los seguidores. Los tipos de eventos representados pueden tener un equipo, un jugador o un infractor de la forma en que se puede observar en la figura.

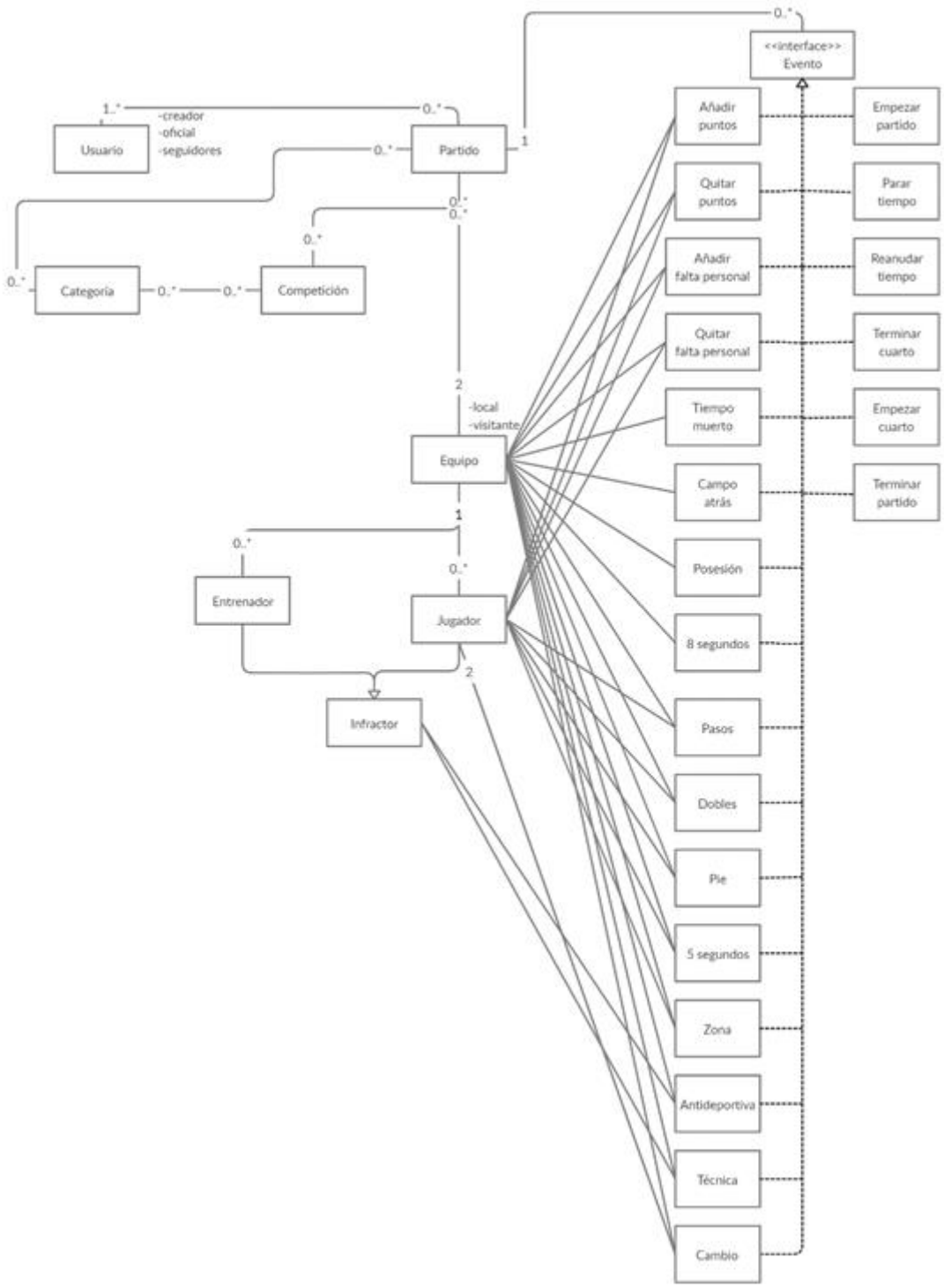


Figura 9. Modelo de clases de dominio

4.4 Diagrama de estados de Partido

Para entender mejor cómo funcionan algunas entidades del sistema se realiza un diagrama de estados.

Un partido pasa por una serie de estados que se detallarán a continuación.

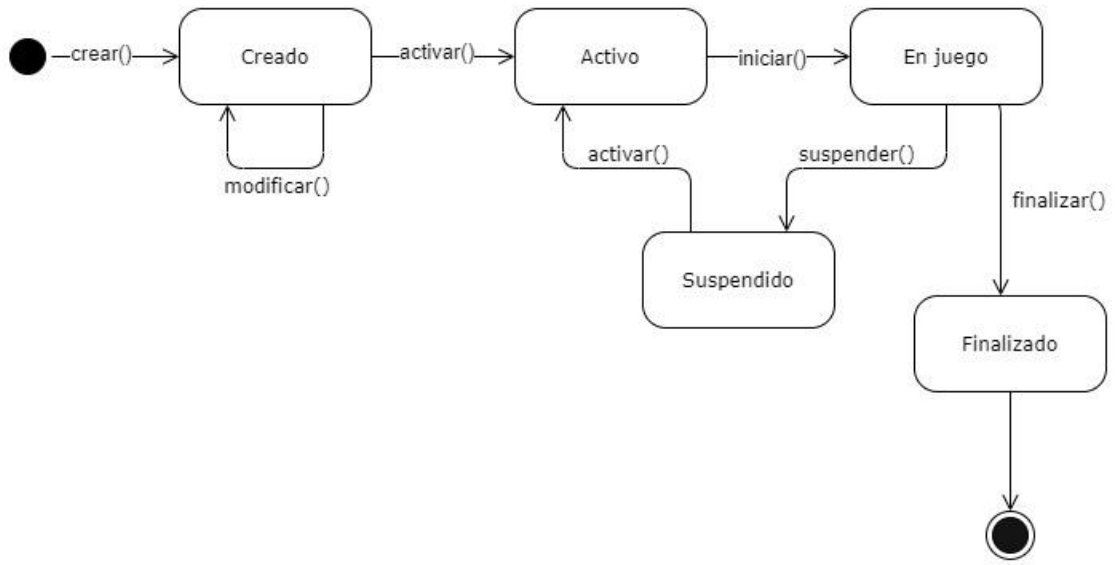


Figura 10. Diagrama de estados de la entidad partido

4.5 Prototipos de pantalla

En esta sección se encuentran los prototipos de pantalla realizados para el diseño del sistema, tanto de la aplicación web como de la aplicación móvil. Todos ellos se han realizado con la herramienta online draw.io [21].

Antes de mostrar las pantallas en detalle de cada una de las partes de las que consta el sistema se muestra un mapa orientativo de pantallas.

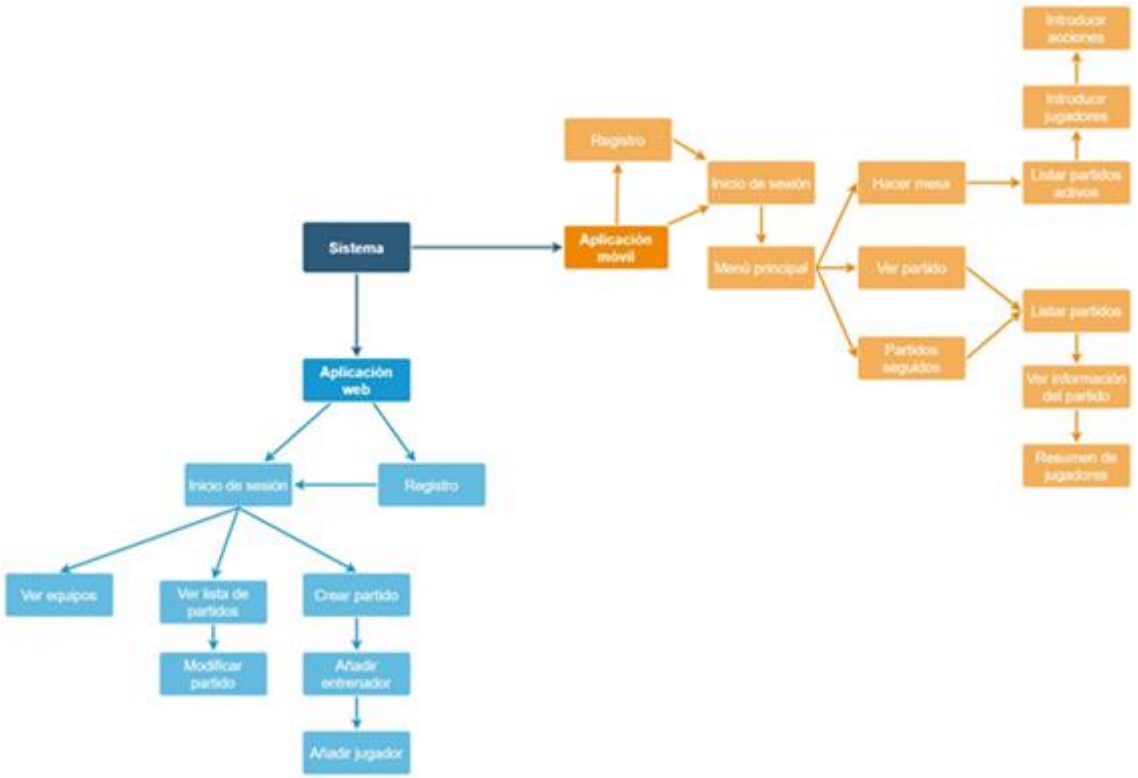


Figura 11. Mapa de pantallas del sistema

Aplicación web

Figura 12.Prototipo de pantalla de Registro en web

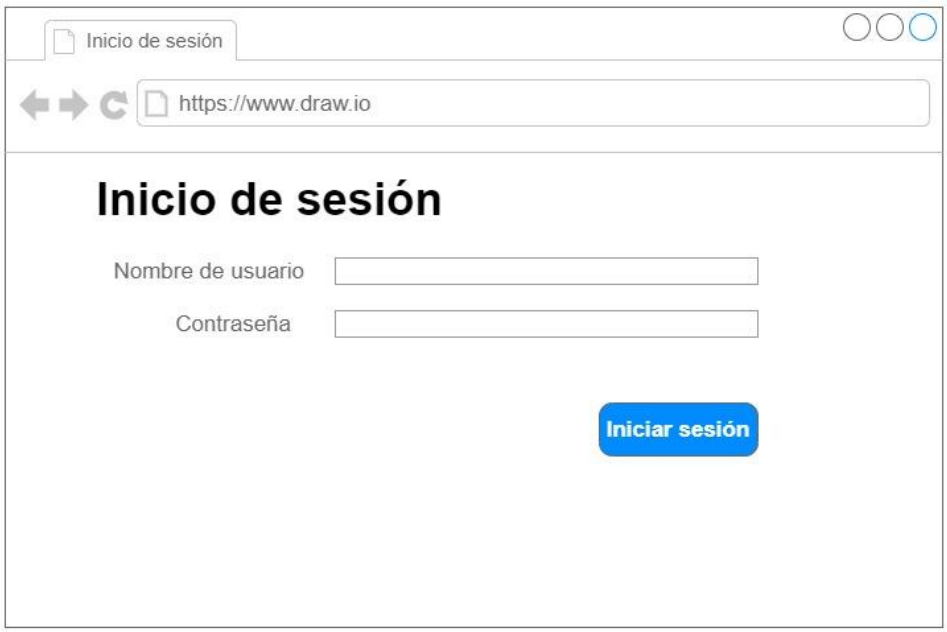


Figura 13. Prototipo de pantalla de Inicio de sesión en web

Figura 14. Prototipo de pantalla del menú de la aplicación web

Figura 15. Prototipo de pantalla de lista de partidos creados y activos

Figura 16. Prototipo de pantalla de navegación entre Inicio de sesión y lista de partidos

Figura 17. Prototipo de pantalla de creación de equipo

Figura 18. Prototipo de pantalla de inserción de entrenador

Figura 19. Prototipo de pantalla de inserción de jugador

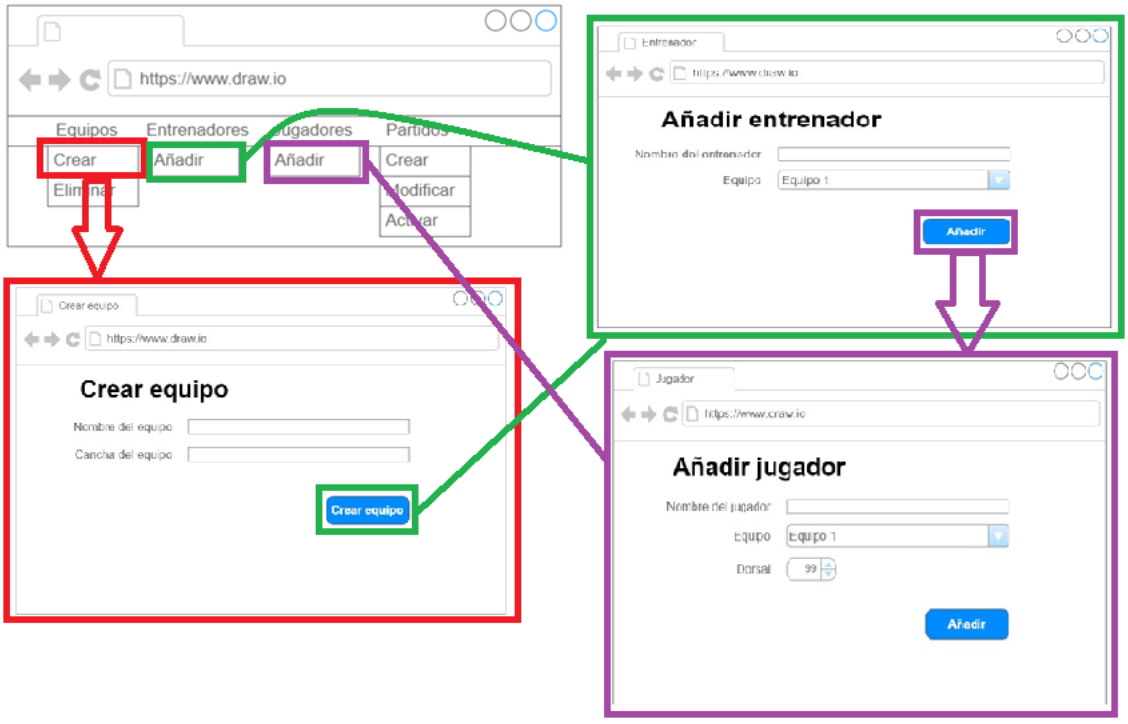


Figura 20. Prototipo de pantalla de navegación para crear un partido, añadir entrenador y añadir jugadores

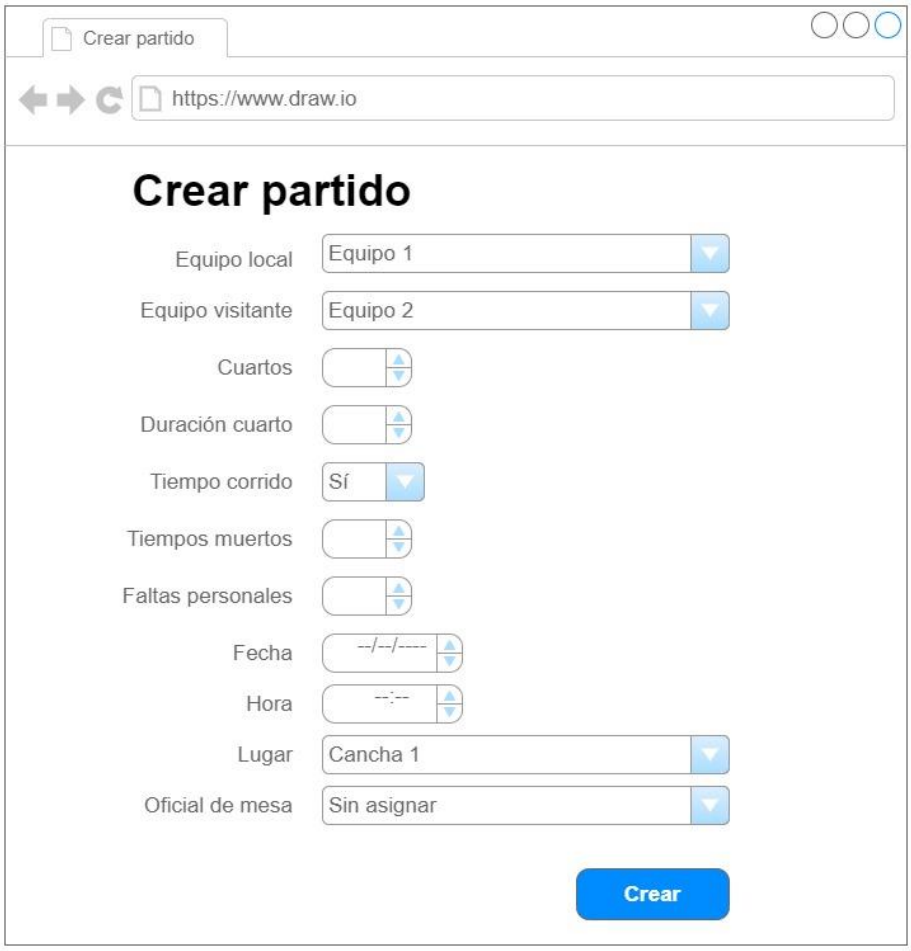


Figura 21. Prototipo de pantalla de creación de partido

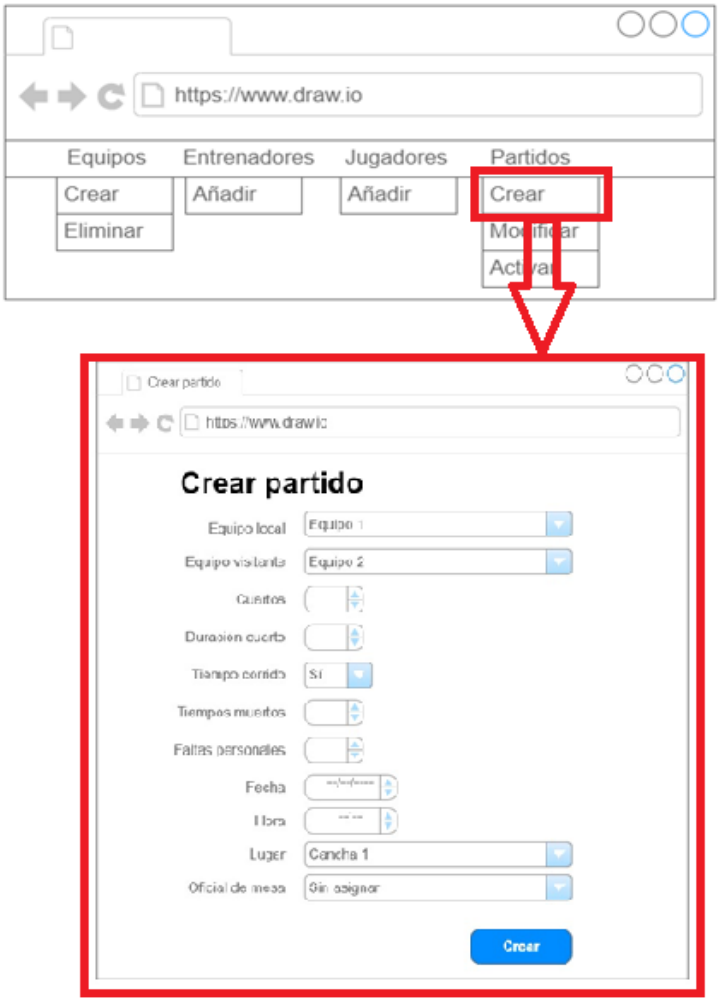


Figura 22. Prototipo de pantalla de navegación para crear partido

Figura 23. Prototipo de pantalla de modificación de partido

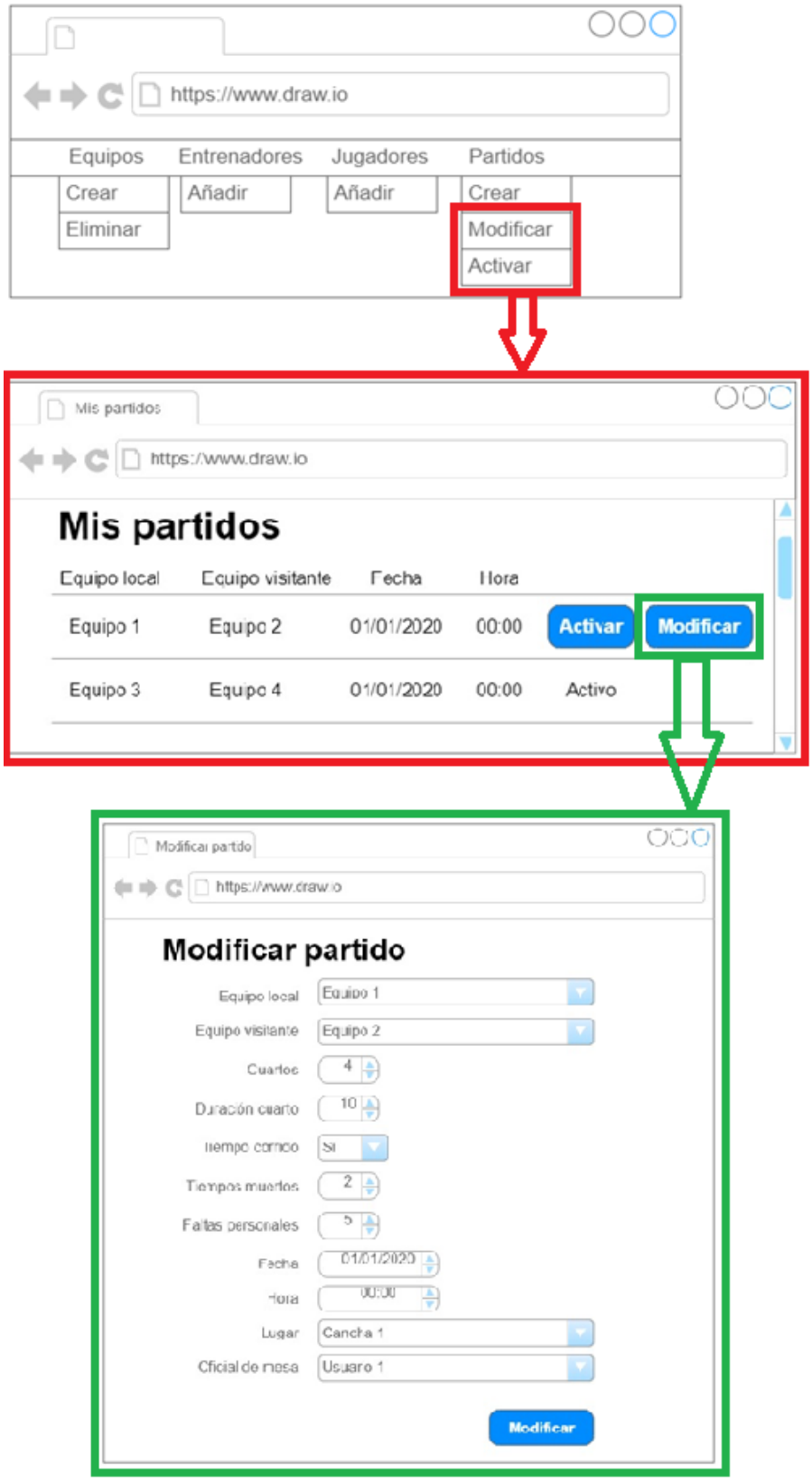


Figura 24. Prototipo de pantalla de navegación para modificar un partido

Aplicación móvil

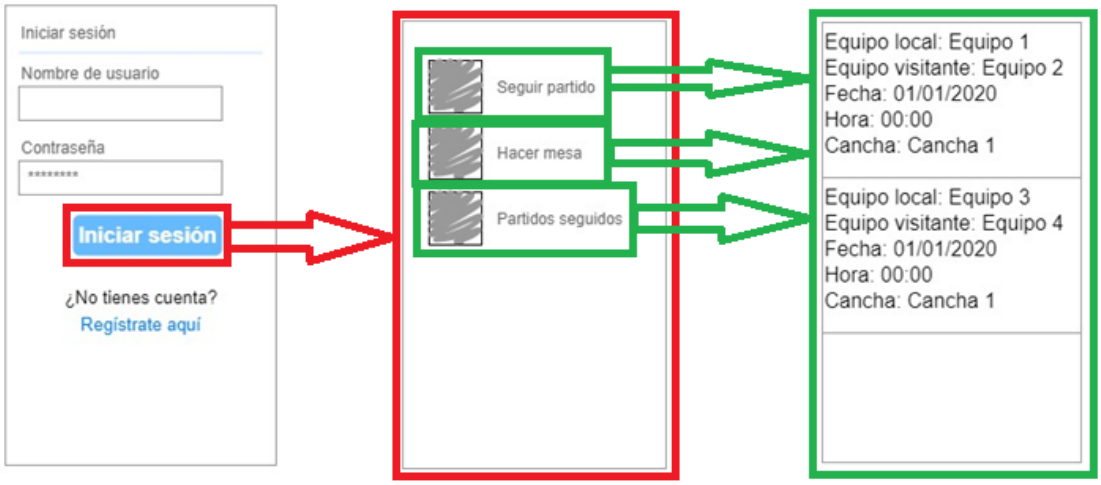


Figura 25. Prototipo de pantallas de navegación para listar partidos en la aplicación móvil

Figura 26. Prototipo de pantallas de navegación para hacer mesa de un partido

Figura 27. Prototipo de pantalla para elegir al jugador de una acción

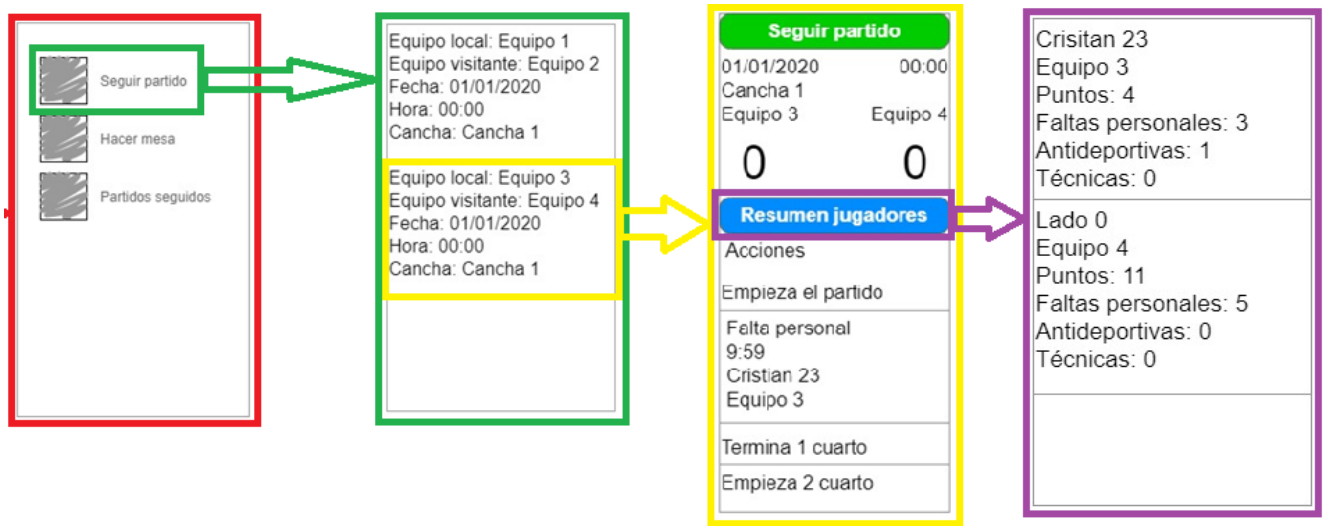


Figura 28. Prototipo de pantalla de navegación para hacer el seguimiento de un partido

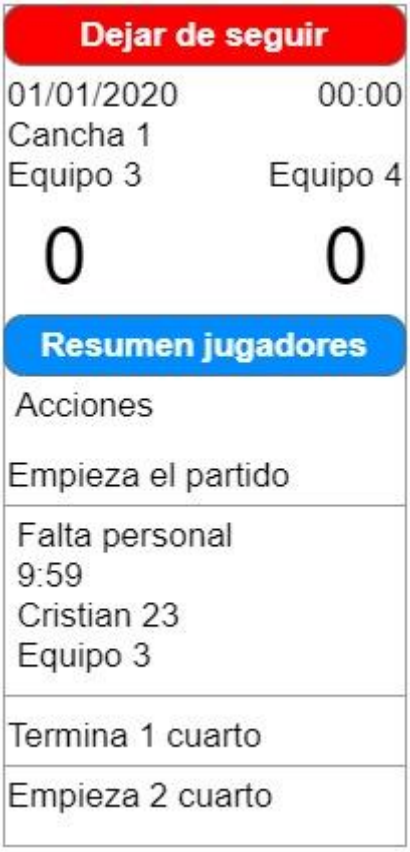


Figura 29. Prototipo de pantalla para el seguimiento de un partido seguido

4.6 Especificación del Plan de Pruebas

En este apartado se creará el plan de pruebas.

En este proyecto se contemplan 4 niveles de pruebas:

- **Pruebas unitarias:** Las pruebas unitarias están dirigidas a comprobar el correcto funcionamiento de cada uno de los módulos del sistema por separado. Sirven para comprobar que la lógica de negocio se comporta como se espera.
  Para realizar estas pruebas de la aplicación web se necesita ejecutar el servidor del que consta el sistema, mientras que en la parte de la aplicación móvil no es necesaria esta ejecución ya que se realizarán, principalmente, comprobaciones de validación del código.
  Para las pruebas realizadas en el entorno de desarrollo WebStorm, es decir, para las pruebas sobre la aplicación web, se utilizará Mocha [17], un framework de pruebas para Node js. Esta tecnología ya fue utilizada en la asignatura Arquitectura del Software. Para las pruebas en Android Studio se utilizará JUnit 4.12 [18].
- **Pruebas de integración:** Las pruebas de integración se realizarán para verificar el correcto funcionamiento de los módulos una vez integrados entre sí. Al igual que en las pruebas unitarias se utilizará Mocha y Chai-HTTP para probar las peticiones al servicio REST.
- **Pruebas de sistema:** Las pruebas de sistema sirven para comprobar el correcto funcionamiento del sistema en conjunto. Estas pruebas se basan en las funcionalidades

expuestas en la sección 4.1 Requisitos funcionales del sistema Dentro de este nivel de pruebas se han considerado 4 tipos de pruebas:

- **Pruebas funcionales:** En las que se verifica que el sistema cumple con la funcionalidad requerida y lo hace una forma correcta.
  Este tipo de pruebas van enfocadas a probar el sistema desde un punto de vista de la interfaz gráfica de usuario. Se ha decidido dividirlas en dos partes. Por un lado, las pruebas de la aplicación web realizadas automáticamente con Selenium, y por otro, las de la aplicación móvil realizadas manualmente. En este tipo de pruebas también encajan pruebas de validación de los formularios en las que se comprueba que cualquier manipulación del sistema de forma incorrecta o maliciosa no tiene consecuencias en su ejecución.
  Estas pruebas complementan a las demás de este tipo en cuanto a que se prueba todo lo que no se había probado en ellas con respecto a las unitarias.
  Existen principalmente tres tipos de pruebas a realizar, cambiando valores del html de la aplicación:
  1. Eliminar "require" de los campos de los formularios correspondientes.
  2. Cambiar el valor de los combo box por uno inadecuado.
  3. Cambiar el mínimo y máximo de los valores numéricos.

  Las pruebas de la aplicación web se han diseñado en base a las tablas de los dos niveles de prueba anteriores, es decir, basándose en clases de equivalencia, escogiendo lo más crítico desde el punto de vista de la interfaz de usuario.
  Las pruebas de la aplicación móvil se realizan manualmente siguiendo técnicas basadas en caminos y transiciones, en este caso concreto se ha elegido la técnica de pares de caminos para hacer algo más robustas las pruebas que con caminos simples. Esta técnica consiste en recorrer todos los pares de caminos que existen en un escenario determinado.
- **Pruebas de seguridad:** En las que se comprueba cualquier acceso denegado al sistema. Estas pruebas se realizarán de forma manual. Tanto en la aplicación web como en la aplicación móvil se comprueba que los usuarios no identificados no puedan acceder a la funcionalidad del sistema. En el caso de la aplicación web se intentará acceder a todas las rutas de las que se compone sin estar identificado. En la aplicación móvil se comprobará que solo se puede acceder a la pantalla de registro y de identificación.
- **Pruebas de rendimiento y carga:** En las que se comprueba la tolerancia del sistema a soportar cierta carga y que no afecte a su rendimiento.
- **Pruebas de usabilidad:** En las que se hace un estudio a varias personas seleccionadas con distintos perfiles sobre su manejo con el sistema. Se observarán las dificultades que presentan para realizar distintas tareas y así poder mejorar el sistema.
  Para la realización de estas pruebas se intentará elegir a personas de diferentes perfiles para encontrar facilidades y dificultades que puedan surgir al utilizar el sistema como usuario final.
  Son unas pruebas monitorizadas, es decir, el proveedor del sistema guiará al usuario sobre las acciones que debe realizar en el sistema, para poder así observar la manipulación en todas las partes del sistema contempladas.

- **Pruebas de aceptación:** Las pruebas de aceptación se realizan con el cliente. Se le muestra el sistema construido y muestra su conformidad o disconformidad con algunas partes del sistema, es decir, si era lo que esperaba o no. En este caso, se basarán en las

historias de usuario definidas en el backlog ya que es la funcionalidad acordada con el usuario desde un primer momento. Para realizarlas se irán revisando una a una las historias de usuario para ver si el sistema permite realizar lo esperado y de una manera adecuada para el cliente. Una vez se realizan las pruebas se obtiene una retroalimentación del cliente para saber si el sistema cumple con sus expectativas o se necesitan mejorar algunos aspectos.

Se intentará seguir una pirámide de test [20] en la que en la base están las pruebas unitarias, sobre ellas, las de integración y, en la cúspide, las de sistema. Por último, las de aceptación serían una nube encima de la pirámide ya que no se automatizan y es el culmen de las pruebas. Esta pirámide tiene un significado ya que la mayor parte de las pruebas se basan en las pruebas unitarias, que son mucho más críticas que las demás, y va disminuyendo la cantidad según se avanza hacia arriba en la pirámide.

Figura 30. Pirámide de test de Cohn adaptada al trabajo

Capítulo 5. Diseño del sistema

5.1 Arquitectura

El sistema tiene dos partes claramente diferenciadas: una aplicación móvil y una aplicación web.

Dentro del propio sistema, para comunicar ambas aplicaciones, se utiliza una base de datos MongoDB para almacenar datos fijos que quedarán en la aplicación un tiempo considerable. Estos datos son las competiciones, las categorías, los equipos, los jugadores y los partidos. Se utiliza una base de datos de este tipo ya que se adapta muy bien a las tecnologías que se van a utilizar, Android Studio y Node.js, ya que utiliza un formato JSON y es muy fácil de manejar en ambas. Como intermediario entre la base de datos MongoDB y la aplicación móvil se utiliza un servicio REST propio realizado con Node.js. Este servicio se utiliza para que la aplicación móvil pueda recibir información de la base de datos y enviarla ya que la tecnología utilizada, Android Studio, no soporta muy bien el driver de Mongo, por lo que se ha optado por esta solución.

En la parte de la aplicación móvil con Android Studio se utiliza un paquete llamado Socket.io que permite actualizar la aplicación cuando haya un cambio en el lugar en el que está "escuchando" dicho paquete. De esta forma, cuando la aplicación móvil produzca un cambio en el servicio REST, este cambio se actualizará automáticamente en la propia aplicación. Este paquete se utiliza en mensajería instantánea para actualizar los mensajes.

Por otra parte, como se ha comentado en otros apartados, se utiliza una fuente de información externa, en este caso una API, para introducir los datos de una competición y/o una categoría automáticamente. Esta API externa la utiliza la aplicación web, ya que es la encargada de crear las competiciones y categorías, para, a su vez, introducir estos datos en la base de datos.

En la Figura 3 se muestra un esquema de la arquitectura descrita en este apartado de una forma visual para que sea más fácil de entender.

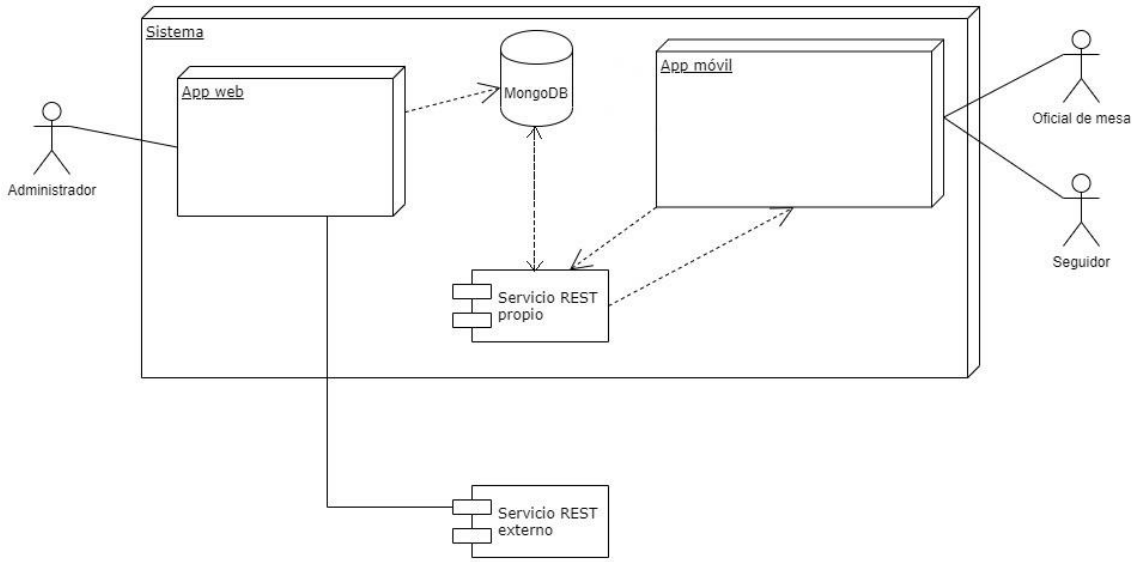


Figura 31. Esquema de la arquitectura inicial del sistema

La aplicación web sigue una arquitectura dividida en módulos. Cada módulo trata una funcionalidad del sistema. En este caso existen tres módulos divididos de tal forma que cada uno de ellos trata con los usuarios, los equipos y los partidos. Estos tres módulos se comunican con la base de datos a través de otro módulo que gestiona la base de datos. En la siguiente figura se muestra la interacción entre la interfaz gráfica de la aplicación y la base de datos con cada una de sus capas intermedias.

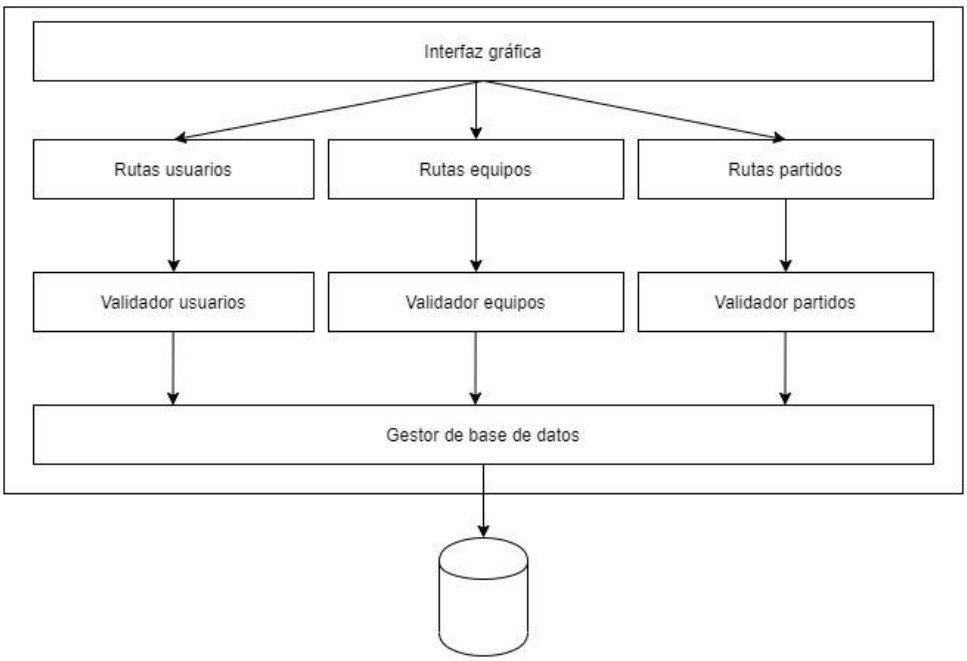


Figura 32. Esquema de la arquitectura de la aplicación web

A su vez, la aplicación móvil cuenta con una arquitectura propia. En esta arquitectura se ha separado la interfaz gráfica de usuario de la comunicación con el servicio REST propio, que, a su vez, se comunica con la base de datos. Entre estos dos niveles se incluyen las actividades, objetos que se encargan de comunicar la interfaz gráfica de usuario con las entidades o servicios.

En el caso de esta última capa en la que se establece la comunicación entre la actividad y el servicio REST, se ha decidido hacer de esta manera para que las entidades se encarguen de enviar y recibir información propia de la base de datos y, en los casos en los que no existen entidades propiamente dichas, sino que son varias las que se tendrían que manejar, se utilizan los servicios, que engloban varios objetos a incluir o recibidos del servicio REST.

En la siguiente figura se muestra esta comunicación entre los distintos componentes de la arquitectura de la aplicación móvil.

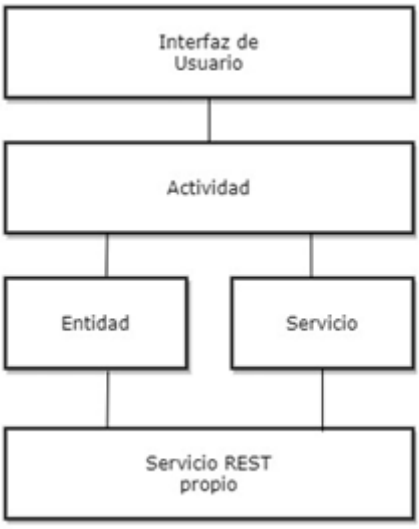


Figura 33. Esquema de la arquitectura de la aplicación móvil

5.2 Diagrama de paquetes

A continuación, se muestra el diagrama de paquetes dividido en dos figuras. En una se muestra el diagrama correspondiente a la estructura de la aplicación web y en otra el correspondiente a la estructura de la aplicación móvil.

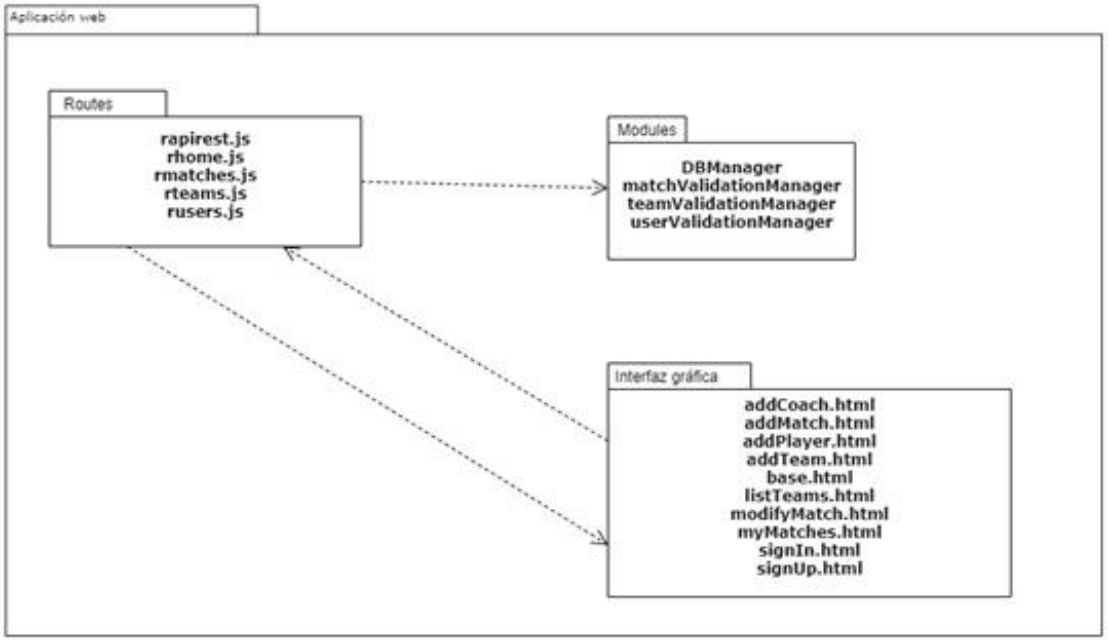


Figura 34. Diagrama de paquetes de la aplicación web

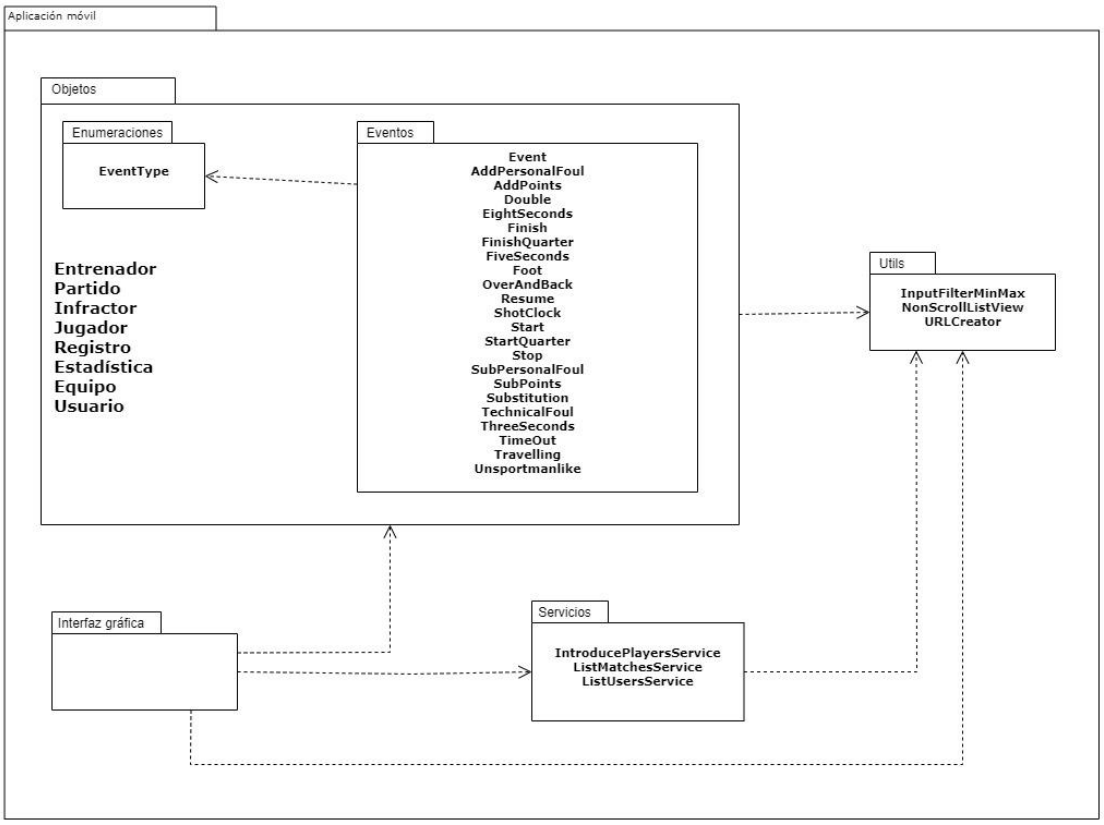


Figura 35. Diagrama de paquetes de la aplicación móvil

5.3 Diagrama de clases

Los diagramas que se muestran a continuación corresponden a las clases de la aplicación móvil ya que es la única parte del sistema en la que existen clases. En concreto se muestran las clases de los paquetes Objetos, Eventos y Servicios vistos en el apartado anterior.

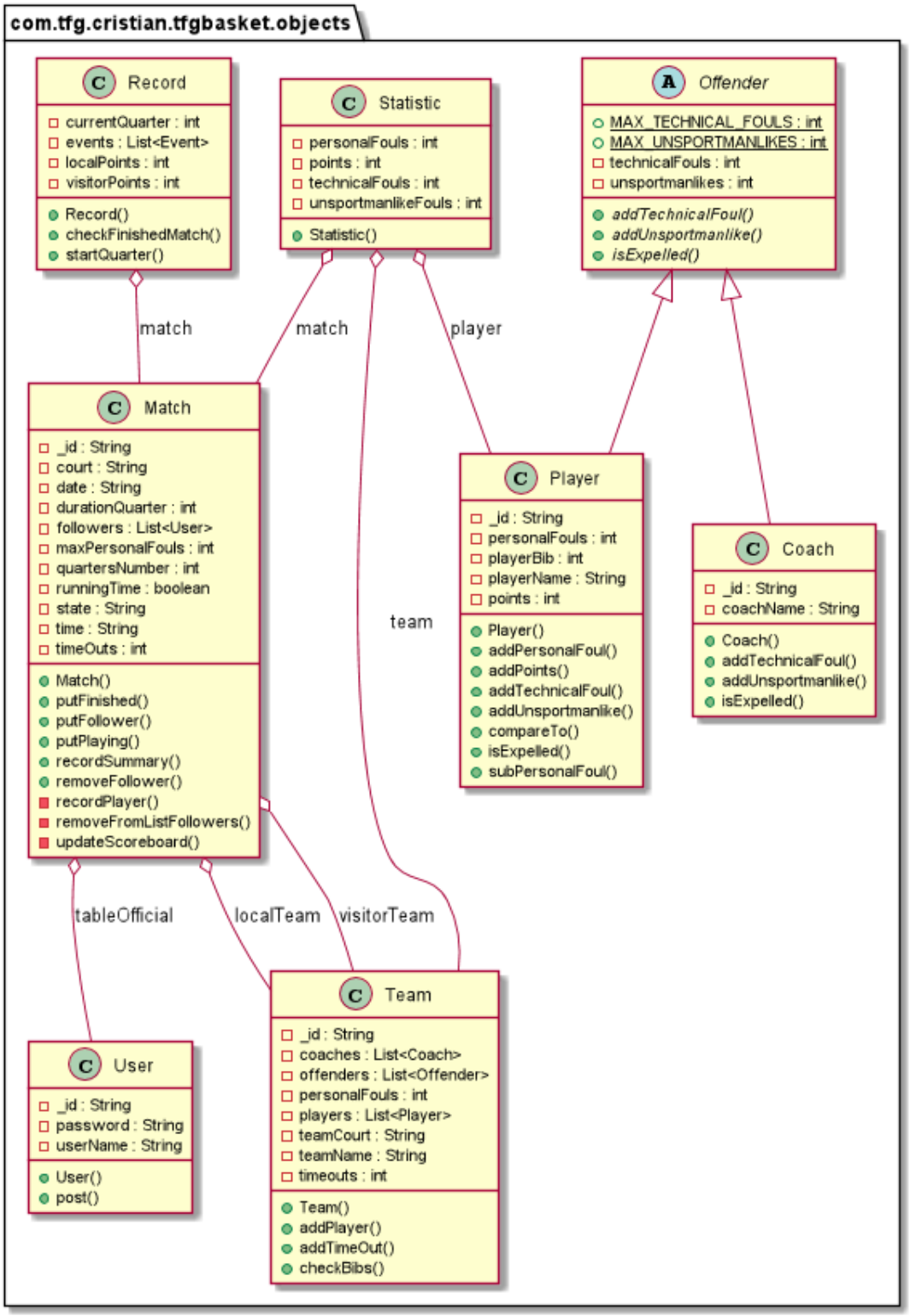


Figura 36. Diagrama de clases del paquete Objetos

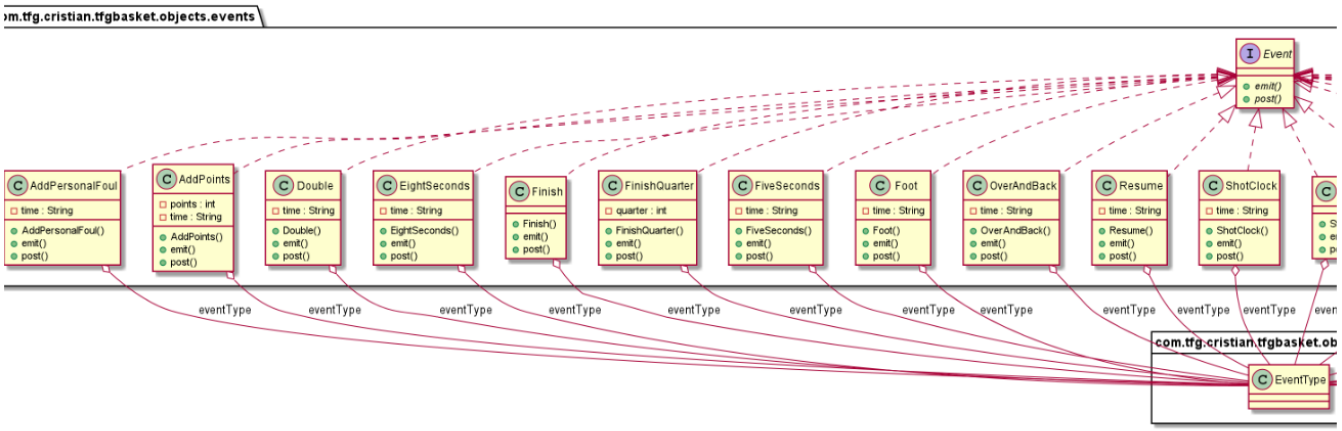


Figura 37. Diagrama de clases del paquete Eventos (I)

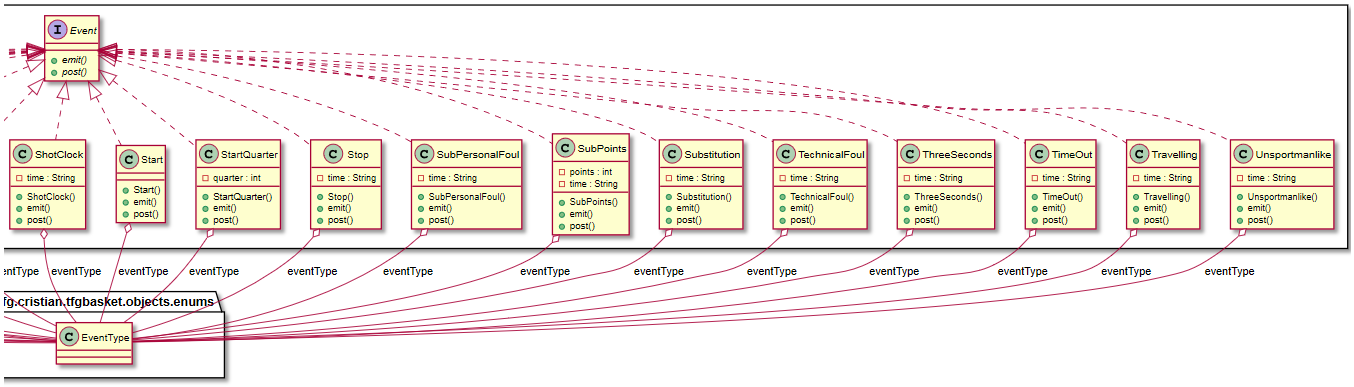


Figura 38. Diagrama de clases del paquete Eventos (II)

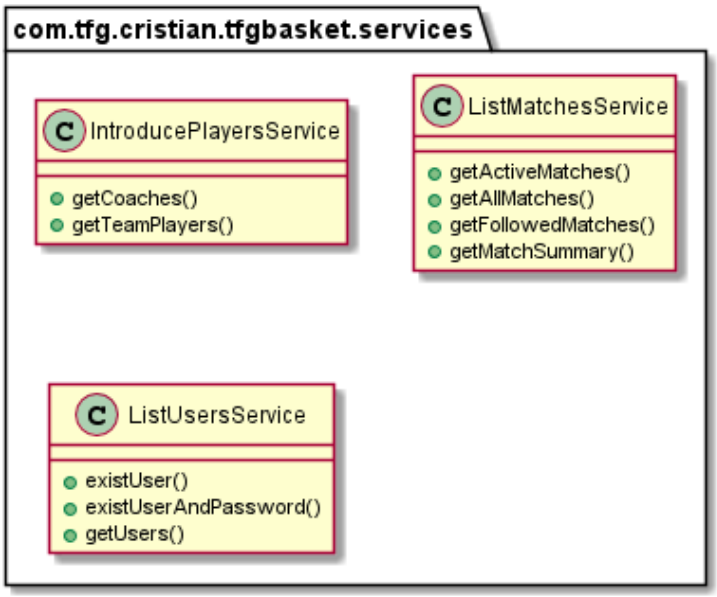


Figura 39. Diagrama de clases del paquete Servicios

5.4 Diseño de la base de datos

A partir del Modelo de clases de dominio se diseña la composición de la base de datos. Como se utiliza una base de datos MongoDB, la cual es un sistema NoSQL, las entidades de la figura representan las diferentes colecciones de esta. Al no ser un modelo de datos basado en registros y tablas, sino en documentos, no todos ellos tienen los mismos campos por lo que solo se han representado algunos campos fijos en algunas de las colecciones.

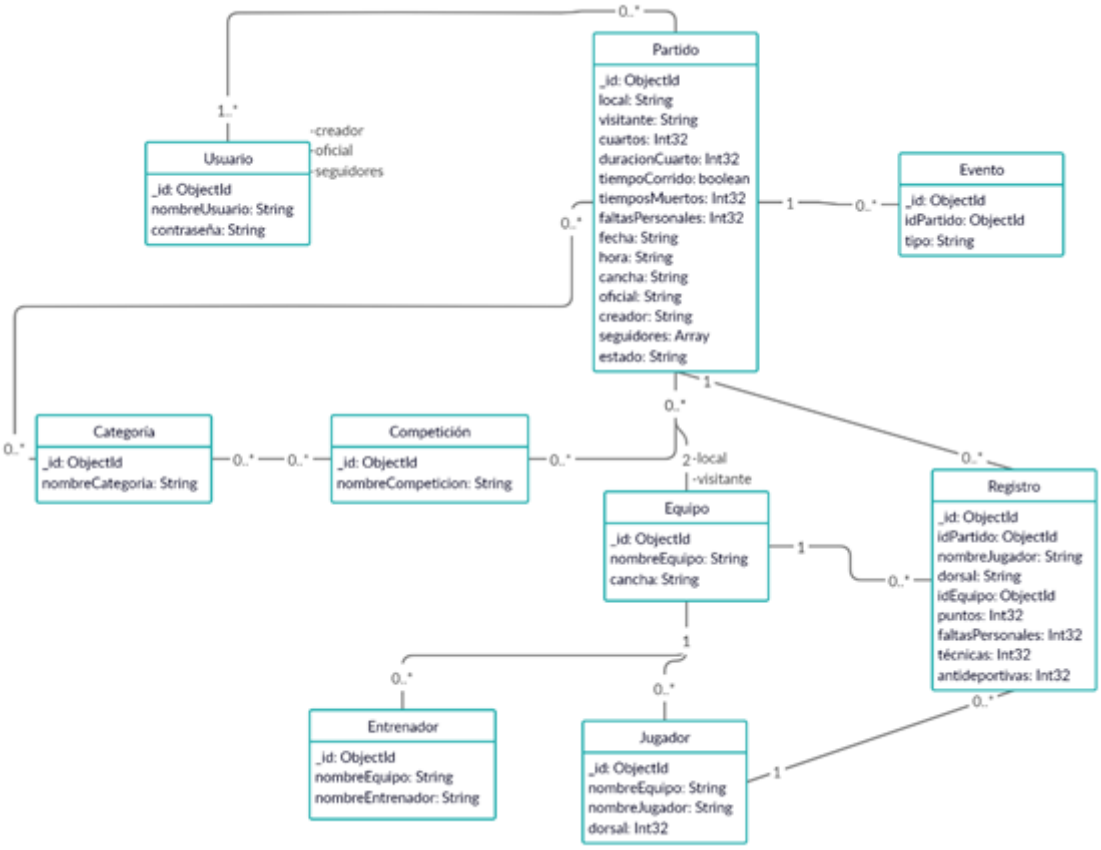


Figura 40. Diagrama del diseño de la base de datos

5.5 Diseño de las pruebas

En este apartado se diseñarán las pruebas tal y como se ha especificado en el plan de pruebas en el apartado Especificación del Plan de Pruebas.

En el caso de las pruebas relativas a la aplicación web se realizarán sobre un entorno con un sistema operativo Windows 10 Home, mientras que para las pruebas de la aplicación móvil se ha utilizado un dispositivo con un sistema operativo Android 7.1.2.

El orden en el que se realizarán las pruebas será el mismo que se sigue aquí, realizando las pruebas unitarias, las de integración y algunos tipos de las de sistema durante los sprints planificados a la vez que se desarrollan cada una de las historias de usuario.

5.5.1 Pruebas unitarias

En el diseño de las pruebas unitarias se han utilizado clases de equivalencia.

Se han dividido en tres partes claramente diferenciadas. Por un lado, están las pruebas relativas a los equipos, por otro, las relativas a los partidos, y, por último, las relativas a los usuarios. Cada una de estas partes necesitan ser ejecutadas por separado ya que, al realizarse de manera asíncrona cada una de ellas, pueden mezclarse datos que no deberían.

Todos los casos de prueba son independientes por lo que se podrían ejecutar de forma separada.

No se incluye el diseño de estas pruebas ya que es el programador el que las va realizando a medida que avanza el desarrollo del sistema para probar métodos y funciones. Se tiene que ir probando componente a componente.

5.5.2 Pruebas de integración

Las pruebas de integración se realizan para comprobar el correcto funcionamiento de todos los componentes entre sí. En este caso estas pruebas se han centrado en probar el servicio REST propio que comunica la aplicación móvil con la base de datos. Para ello se ha utilizado Mocha, al igual que para las pruebas unitarias, y Chai-HTTP, una librería para Node js que sirve para realizar este tipo de pruebas [16].

La condición inicial para realizar estas pruebas es rellenar la base de datos a través del script de rellenado de datos de los Anexos. Las pruebas a realizar son las siguientes:

Integración de la aplicación móvil con el servicio REST para obtener partidos activos			
Id	Descripción	Procedimiento	Salida esperada
PI1	Obtener todos los partidos en estado “activo” del sistema	Acceder a la ruta /api/activeMatches mediante GET	Se obtienen un partido con equipo local “Sanfer”, equipo visitante “ADBA”, número de cuartos 4, duración de cada cuarto 1, tiempo corrido “false”, número de tiempos muertos 2, número máximo de faltas personales 5, fecha “2020-04-10”, hora “20:11”, cancha “Polideportivo Quirinal”, oficial de mesa vacío, nombre de usuario “test”, seguidores “[]” y estado “active”

Tabla 11. Pruebas de integración de la aplicación móvil con el servicio REST para obtener partidos activos

Integración de la aplicación móvil con el servicio REST para obtener jugadores de un equipo			
Id	Descripción	Procedimiento	Salida esperada
PI2	Obtener jugadores de un equipo inexistente	Acceder a la ruta /api/players/test1234 mediante GET	Un array de jugadores vacío
PI3	Obtener jugadores de un equipo que no tiene jugadores	Acceder a la ruta /api/players/test2 mediante GET	Un array de jugadores vacío
PI4	Obtener jugadores de un equipo con jugadores	Acceder a la ruta /api/players/test mediante GET	Un jugador con nombre de equipo “test”, nombre de jugador “Test” y dorsal 0

Tabla 12. Pruebas de integración de la aplicación móvil con el servicio REST para obtener jugadores de un equipo

Integración de la aplicación móvil con el servicio REST para obtener entrenadores de un equipo			
Id	Descripción	Procedimiento	Salida esperada
PI5	Obtener entrenadores de un equipo inexistente	Acceder a la ruta /api/coaches/test1234 mediante GET	Un array de entrenadores vacío
PI6	Obtener entrenadores de un equipo que no tiene entrenadores	Acceder a la ruta /api/coaches/test2 mediante GET	Un array de entrenadores vacío
PI7	Obtener entrenadores de un equipo con entrenadores	Acceder a la ruta /api/coaches/Sanfer mediante GET	Un entrenador con nombre de equipo “Sanfer” y nombre de entrenador “Pablo”

Tabla 13. Pruebas de integración de la aplicación móvil con el servicio REST para obtener entrenadores de un equipo

Integración de la aplicación móvil con el servicio REST para empezar partido			
Id	Descripción	Procedimiento	Salida esperada
PI8	Empezar un partido inexistente	Acceder a la ruta /api/start/match/555555555555555555 mediante PUT	No se actualiza ningún partido de los existentes
PI9	Empezar un partido existente	Acceder a la ruta /api/start/match/”id de un partido existente” mediante PUT	Se actualiza el partido con el id introducido en la ruta a estado “playing” y no aparece entre los partidos activos

Tabla 14. Pruebas de integración de la aplicación móvil con el servicio REST para empezar partido

Integración de la aplicación móvil con el servicio REST para añadir eventos			
Id	Descripción	Procedimiento	Salida esperada
PI10	Añadir evento de empezar partido	Acceder a la ruta /api/add/start mediante POST enviando el id de un partido existente	Se añade el evento
PI11	Añadir evento de parar tiempo	Acceder a la ruta /api/add/stop mediante POST enviando el id de un partido existente y el tiempo “9:59”	Se añade el evento
PI12	Añadir evento de reanudar tiempo	Acceder a la ruta /api/add/resume mediante POST enviando el id de un partido existente y el tiempo “9:59”	Se añade el evento
PI13	Añadir evento de añadir puntos	Acceder a la ruta /api/add/points mediante POST enviando el id de un partido existente, el tiempo “9:59”, id del jugador vacío, nombre del jugador “Cristian”, id del equipo “555 55555555555555555555” dorsal 23 y puntos 3	Se añade el evento
PI14	Añadir evento de eliminar puntos	Acceder a la ruta /api/sub/points mediante POST enviando el id de un partido existente, el tiempo “9:59”, id del jugador vacío, nombre del jugador “Cristian”, id del equipo “555 55555555555555555555”, dorsal 23 y puntos -1	Se añade el evento
PI15	Añadir evento de añadir falta personal	Acceder a la ruta /api/add/personalFoul mediante POST enviando el id de un partido existente, el tiempo “9:59”, id del jugador vacío, nombre del jugador “Cristian”, id del equipo “555 55555555555555555555” y dorsal 23	Se añade el evento
PI16	Añadir evento de eliminar falta personal	Acceder a la ruta /api/sub/personalFoul mediante POST enviando el id de un partido existente, el tiempo “9:59”, id del jugador vacío, nombre del jugador “Cristian”, id del equipo “555 55555555555555555555” y dorsal 23	Se añade el evento
PI17	Añadir evento de tiempo muerto	Acceder a la ruta /api/add/timeout mediante POST enviando el id de un partido existente, el tiempo “9:59” e id del equipo “5555 55555555555555555”	Se añade el evento
PI18	Añadir evento de pasos	Acceder a la ruta /api/add/travelling mediante POST enviando el id de un partido existente, el tiempo “9:59”, id del jugador vacío, nombre del jugador “Cristian”, id del equipo “555 55555555555555555555” y dorsal 23	Se añade el evento

Id	Descripción	Procedimiento	Salida esperada
PI19	Añadir evento de dobles	Acceder a la ruta /api/add/double mediante POST enviando el id de un partido existente, el tiempo “9:59”, id del jugador vacío, nombre del jugador “Cristian”, id del equipo “555 55555555555555555555” y dorsal 23	Se añade el evento
PI20	Añadir evento de campo atrás	Acceder a la ruta /api/add/overAndBack mediante POST enviando el id de un partido existente, el tiempo “9:59” e id del equipo “5555555555555555 55555555”	Se añade el evento
PI21	Añadir evento de posesión	Acceder a la ruta /api/add/shotClock mediante POST enviando el id de un partido existente, el tiempo “9:59” e id del equipo “5555555555555555 55555555”	Se añade el evento
PI22	Añadir evento de 5 segundos	Acceder a la ruta /api/add/fiveSeconds mediante POST enviando el id de un partido existente, el tiempo “9:59”, id del jugador vacío, nombre del jugador “Cristian”, id del equipo “555 55555555555555555555” y dorsal 23	Se añade el evento
PI23	Añadir evento de zona	Acceder a la ruta /api/add/threeSeconds mediante POST enviando el id de un partido existente, el tiempo “9:59”, id del jugador vacío, nombre del jugador “Cristian”, id del equipo “555 55555555555555555555” y dorsal 23	Se añade el evento
PI24	Añadir evento de 8 segundos	Acceder a la ruta /api/add/eightSeconds mediante POST enviando el id de un partido existente, el tiempo “9:59” e id del equipo “5555555555555555 55555555”	Se añade el evento
PI25	Añadir evento de pie	Acceder a la ruta /api/add/foot mediante POST enviando el id de un partido existente, el tiempo “9:59”, id del jugador vacío, nombre del jugador “Cristian”, id del equipo “555 55555555555555555555” y dorsal 23	Se añade el evento
PI26	Añadir evento de antideportiva	Acceder a la ruta /api/add/unsportmanlike mediante POST enviando el id de un partido existente, el tiempo “9:59”, id del jugador vacío, nombre del jugador “Cristian”, id del equipo “555 55555555555555555555” y dorsal 23	Se añade el evento

Id	Descripción	Procedimiento	Salida esperada
PI27	Añadir evento de técnica	Acceder a la ruta /api/add/technicalFoul mediante POST enviando el id de un partido existente, el tiempo “9:59”, id del jugador vacío, nombre del jugador “Cristian”, id del equipo “555 55555555555555555555” y dorsal 23	Se añade el evento
PI28	Añadir evento de finalizar cuarto	Acceder a la ruta /api/add/finishQuarter mediante POST enviando el id de un partido existente y cuarto 1	Se añade el evento
PI29	Añadir evento de empezar cuarto	Acceder a la ruta /api/add/startQuarter mediante POST enviando el id de un partido existente y cuarto 2	Se añade el evento
PI30	Añadir evento de finalizar partido	Acceder a la ruta /api/add/finish mediante POST enviando el id de un partido existente	Se añade el evento
PI31	Añadir evento de cambio	Acceder a la ruta /api/add/substitution mediante POST enviando el id de un partido existente, el tiempo “9:59”, id del jugador vacío, nombre del jugador “Cristian”, dorsal 23, id del sustituto vacío, nombre del sustituto “Adrián”, dorsal del sustituto 15 e id del equipo “55555555555555555555 555”	Se añade el evento

Tabla 15. Pruebas de integración de la aplicación móvil con el servicio REST para añadir eventos

Los casos de prueba sobre añadir eventos se replican con un id de partido inexistente.

Integración de la aplicación móvil con el servicio REST para finalizar partido			
Id	Descripción	Procedimiento	Salida esperada
PI32	Finalizar un partido inexistente	Acceder a la ruta /api/finish/match/555555555 55555555555555 mediante PUT	No se actualiza ningún partido de los existentes
PI33	Finalizar un partido existente	Acceder a la ruta /api/finish/match/”id de un partido existente” mediante PUT	Se actualiza el partido con el id introducido en la ruta a estado “finished” y no aparece entre los partidos activos

Tabla 16. Pruebas de integración de la aplicación móvil con el servicio REST para finalizar partido

Integración de la aplicación móvil con el servicio REST para la gestión de usuarios			
Id	Descripción	Procedimiento	Salida esperada
PI34	Obtener todos los usuarios del sistema	Acceder a la ruta /api/users mediante GET	Se obtiene un usuario con nombre de usuario “test” y contraseña “1392542397501e1158418adae09d694ffb8ed833a3a5e8a017e15ba565d28c70”
PI35	Añadir un usuario correctamente	Acceder a la ruta /api/add/user mediante POST enviando nombre de usuario “test2” y contraseña “test2” encriptada mediante sha256	Se añade el usuario correctamente al sistema
PI36	Obtener un usuario que existe en el sistema	Acceder a la ruta /api/user mediante POST enviando nombre de usuario “test” y contraseña “test”	Se obtiene un usuario con nombre de usuario “test”
PI37	Obtener un usuario que no existe en el sistema	Acceder a la ruta /api/user mediante POST enviando nombre de usuario “1234” y contraseña “1234”	Se obtiene un array de usuarios vacío

Tabla 17. Pruebas de integración de la aplicación móvil con el servicio REST para la gestión de usuarios

Integración de la aplicación móvil con el servicio REST para la gestión de seguidores			
	Descripción	Procedimiento	Salida esperada
PI38	Añadir un seguidor a un partido	Acceder a la ruta /api/add/follower/”id de un partido existente” mediante PUT enviando el nombre de usuario “test2”	Se añade el seguidor al array de seguidores del partido cuyo id se indica en la ruta
PI39	Obtener seguidores de un partido existente	Acceder a la ruta /api/allMatchesVisible mediante GET	Se obtienen un partido con equipo local “Sanfer”, equipo visitante “ADBA”, número de cuartos 4, duración de cada cuarto 1, tiempo corrido “false”, número de tiempos muertos 2, número máximo de faltas personales 5, fecha “2020-04-10”, hora “20:11”, cancha “Polideportivo Quirinal”, oficial de mesa vacío, nombre de usuario “test”, seguidores “[test2]” y estado “finished”
PI40	Eliminar un seguidor de un partido	Acceder a la ruta /api/remove/follower/”id de un partido existente” mediante PUT enviando el nombre de usuario “test2”	Se elimina el seguidor del array de seguidores del partido cuyo id se indica en la ruta

Tabla 18. Pruebas de integración de la aplicación móvil con el servicio REST para la gestión de seguidores

5.5.3 Pruebas de sistema

Este nivel de pruebas se ha dividido en cuatro tipos: funcionales, de seguridad, de rendimiento y carga y de usabilidad.

Pruebas funcionales

En el caso de la aplicación web se obtienen los casos de prueba detallados a continuación. La condición inicial para todos los casos de prueba es la ejecución del script de rellenado de datos de los Apéndices.

A todos y cada uno de estos casos de prueba les precede un inicio de sesión con el usuario "test" y contraseña "test", a excepción del Registro de usuario y el Inicio de sesión.

Creación de equipo			
Id	Descripción	Procedimiento	Salida esperada
PSF1	Creación correcta de un equipo	Introducir un equipo con nombre "correctTeam" y nombre de la cancha "correctCourt"	Tras insertar el equipo debe mostrar la página de añadir entrenador
PSF2	Creación incorrecta de un equipo con el nombre del equipo que ya existe	Introducir un equipo con nombre "test" y nombre de la cancha "correctCourt"	Se muestra el mensaje "El nombre de equipo ya existe"
PSF3	Introducción correcta de entrenador	Introducir un entrenador con nombre "correctCoach" y nombre de equipo "Llaranes"	Se muestra el mensaje "Entrenador añadido"
PSF4	Introducción incorrecta de jugador con dorsal repetido en el equipo	Introducir un jugador con nombre "correctPlayer", doral 10 y nombre de equipo "Llaranes"	Se muestra el mensaje "El dorsal ya existe en ese equipo"
PSF5	Introducción correcta de jugador	Introducir un jugador con nombre "correctPlayer", dorsal 10 y nombre de equipo "Llaranes"	Se muestra el mensaje "Jugador añadido"

Tabla 19. Pruebas de sistema funcionales de creación de equipo

Creación de partido			
Id	Descripción	Procedimiento	Salida esperada
PSF6	Creación correcta de un partido	Crear un partido con equipo local “Llaranes”, equipo visitante “Real Madrid”, número de cuartos 4, duración de cada cuarto 10, tiempo corrido “true”, número de tiempos muertos 2, número máximo de faltas personales 5, día “23-10-1998”, hora “13:30”, cancha “La Toba” y oficial de mesa “Sin asignar”	Se muestra el mensaje “Partido creado correctamente”
PSF7	Creación incorrecta de un partido con equipos iguales	Crear un partido con equipo local “Llaranes”, equipo visitante “Llaranes”, número de cuartos 4, duración de cada cuarto 10, tiempo corrido “true”, número de tiempos muertos 2, número máximo de faltas personales 5, día “23-10-1998”, hora “13:30”, cancha “La Toba” y oficial de mesa “Sin asignar”	Se muestra el mensaje “Los nombres de los equipos son iguales”
PSF8	Creación correcta de un partido con tiempo corrido “false”	Crear un partido con equipo local “Llaranes”, equipo visitante “Real Madrid”, número de cuartos 4, duración de cada cuarto 10, tiempo corrido “false”, número de tiempos muertos 2, número máximo de faltas personales 5, día “23-10-1998”, hora “13:30”, cancha “La Toba” y oficial de mesa “Sin asignar”	Se muestra el mensaje “Partido creado correctamente”
PSF9	Creación correcta de un partido con tiempo corrido “true”	Crear un partido con equipo local “Llaranes”, equipo visitante “Real Madrid”, número de cuartos 4, duración de cada cuarto 10, tiempo corrido “true”, número de tiempos muertos 2, número máximo de faltas personales 5, día “23-10-1998”, hora “13:30”, cancha “La Toba” y oficial de mesa “Sin asignar”	Se muestra el mensaje “Partido creado correctamente”

Tabla 20. Pruebas de sistema funcionales de creación de partido

Modificación de partido			
Id	Descripción	Procedimiento	Salida esperada
PSF10	Modificación correcta de un partido	Modificar el primer partido de la lista modificable con equipo local “Llaranes”, equipo visitante “Real Madrid”, número de cuartos 4, duración de cada cuarto 10, tiempo corrido “true”, número de tiempos muertos 2, número máximo de faltas personales 5, día “23-10-1998”, hora “13:30”, cancha “La Toba” y oficial de mesa “Sin asignar”	Se muestra el mensaje “Partido modificado correctamente”
PSF11	Modificación incorrecta de un partido con equipos iguales	Modificar el primer partido de la lista modificable con equipo local “Llaranes”, equipo visitante “Llaranes”, número de cuartos 4, duración de cada cuarto 10, tiempo corrido “true”, número de tiempos muertos 2, número máximo de faltas personales 5, día “23-10-1998”, hora “13:30”, cancha “La Toba” y oficial de mesa “Sin asignar”	Se muestra el mensaje “Los nombres de los equipos son iguales”

Id	Descripción	Procedimiento	Salida esperada
PSF12	Modificación correcta de un partido con tiempo corrido "false"	Modificar el primer partido de la lista modificable con equipo local "Llaranes", equipo visitante "Real Madrid", número de cuartos 4, duración de cada cuarto 10, tiempo corrido "false", número de tiempos muertos 2, número máximo de faltas personales 5, día "23-10-1998", hora "13:30", cancha "La Toba" y oficial de mesa "Sin asignar"	Se muestra el mensaje "Partido modificado correctamente"
PSF13	Modificación correcta de un partido con tiempo corrido "true"	Modificar el primer partido de la lista modificable con equipo local "Llaranes", equipo visitante "Real Madrid", número de cuartos 4, duración de cada cuarto 10, tiempo corrido "true", número de tiempos muertos 2, número máximo de faltas personales 5, día "23-10-1998", hora "13:30", cancha "La Toba" y oficial de mesa "Sin asignar"	Se muestra el mensaje "Partido modificado correctamente"

Tabla 21. Pruebas de sistema funcionales de modificación de partido

Activación de partido			
	Descripción	Procedimiento	Salida esperada
PSF14	Activar un partido en estado creado	Activar el primer partido posible de la lista	El partido se muestra en la lista como activo

Tabla 22. Pruebas de sistema funcionales de activación de partido

Eliminación de equipo			
Id	Descripción	Procedimiento	Salida esperada
PSF15	Eliminar correctamente un equipo que no está en ningún partido	Eliminar el equipo en la posición séptima de la lista	La lista de equipos contiene 7 equipos y no aparece el eliminado
PSF16	Eliminar incorrectamente un equipo que está en algún partido	Eliminar el equipo en la posición segunda de la lista	La lista de equipos contiene 7 equipos y se muestra el mensaje "El equipo está en algún partido"

Tabla 23. Pruebas de sistema funcionales de eliminación de equipo

Registro de usuario			
Id	Descripción	Procedimiento	Salida esperada
PSF17	Registro incorrecto de un usuario con nombre de usuario repetido en el sistema	Ir a la página de registro e introducir un usuario con nombre de usuario “test”, contraseña “1234” y repetición de contraseña “1234”	Se muestra el mensaje “El nombre de usuario ya existe”
PSF18	Registro correcto de un usuario	Ir a la página de registro e introducir un usuario con nombre de usuario “test2”, contraseña “test2” y repetición de contraseña “test2”	Se muestra la página con la lista de partidos vacía
PSF19	Registro incorrecto de un usuario con repetición de contraseña distinto a contraseña	Ir a la página de registro e introducir un usuario con nombre de usuario “test2”, contraseña “test2” y repetición de contraseña “1234”	Se muestra el mensaje “Las contraseñas no coinciden”

Tabla 24. Pruebas de sistema funcionales de registro de usuario

Inicio de sesión			
Id	Descripción	Procedimiento	Salida esperada
PSF20	Inicio de sesión incorrecto con nombre de usuario no existente	Introducir nombre de usuario “test3” y contraseña “test”	Se muestra el mensaje “Usuario o password incorrecto”
PSF21	Inicio de sesión correcto	Introducir nombre de usuario “test” y contraseña “test”	Aparece la lista de partidos del usuario
PSF22	Inicio de sesión incorrecto con contraseña incorrecta	Introducir nombre de usuario “test” y contraseña “1234”	Se muestra el mensaje “Usuario o password incorrecto”

Tabla 25. Pruebas de sistema funcionales de inicio de sesión

Cierre de sesión			
Id	Descripción	Procedimiento	Salida esperada
PSF23	Cerrar sesión con un usuario en sesión	Cerrar sesión	Se muestra la página de inicio de sesión

Tabla 26. Pruebas de sistema funcionales de cierre de sesión

Dentro de este tipo de pruebas también se encuadran pruebas de validación de los formularios que existen en la aplicación web. Los casos de prueba se especifican a continuación.

Creación de equipo			
Id	Descripción	Procedimiento	Salida esperada
PSF24	Creación incorrecta de un equipo con el nombre del equipo vacío	Eliminar require del campo nombre del equipo. Introducir un equipo con nombre vacío y nombre de la cancha “correctCourt”	Se muestra el mensaje “Campo nombre de equipo vacío”
PSF25	Creación incorrecta de un equipo con el campo cancha vacío	Eliminar require del campo cancha. Introducir un equipo con nombre “correctTeam” y nombre de la cancha vacío	Se muestra el mensaje “Campo cancha vacío”
PSF26	Introducción incorrecta de entrenador de un equipo inexistente	Cambiar el valor del atributo value de la opción seleccionada por “hola”. Introducir un entrenador con nombre “correctCoach” y nombre de equipo “nonexistentTeam”	Se muestra el mensaje “El equipo no existe”
PSF27	Introducción incorrecta de entrenador con nombre vacío	Eliminar require del campo nombre del entrenador. Introducir un entrenador con nombre vacío y nombre de equipo “test”	Se muestra el mensaje “Campo nombre de entrenador vacío”
PSF28	Introducción incorrecta de entrenador con nombre de equipo vacío	Eliminar require del campo nombre del equipo. Introducir un entrenador con nombre “correctCoach” y nombre de equipo vacío	Se muestra el mensaje “El equipo no existe”
PSF29	Introducción incorrecta de jugador con dorsal negativo	Cambiar valor del mínimo del campo dorsal por -1. Introducir un jugador con nombre “correctPlayer”, dorsal -1 y nombre de equipo “test”	Se muestra el mensaje “Dorsal negativo”
PSF30	Introducción incorrecta de jugador con dorsal mayor que 99	Cambiar valor del máximo del campo dorsal por 100. Introducir un jugador con nombre “correctPlayer”, dorsal 100 y nombre de equipo “test”	Se muestra el mensaje “Dorsal mayor que 99”
PSF31	Introducción incorrecta de jugador con nombre vacío	Eliminar require del campo nombre del jugador. Introducir un jugador con nombre vacío, dorsal 99 y nombre de equipo “test”	Se muestra el mensaje “Campo nombre de jugador vacío”
PSF32	Introducción incorrecta de jugador de un equipo inexistente	Cambiar el valor del atributo value de la opción seleccionada por “hola”. Introducir un jugador con nombre “correctPlayer”, dorsal 99 y nombre de equipo vacío	Se muestra el mensaje “El equipo no existe”
PSF33	Introducción incorrecta de jugador con nombre de equipo vacío	Eliminar require del campo nombre del equipo. Introducir un jugador con nombre “correctPlayer” y nombre de equipo vacío	Se muestra el mensaje “El equipo no existe”

Tabla 27. Pruebas de validación de creación de equipo

Creación de partido			
Id	Descripción	Procedimiento	Salida esperada
PSF34	Creación incorrecta de un partido con el número de cuartos menor que 1	Cambiar valor del mínimo del campo número de cuartos por 0. Introducir un partido con equipo local “test”, equipo visitante “test2”, número de cuartos 0, duración de cada cuarto 1, tiempo corrido “false”, número de tiempos muertos 0, número máximo de faltas personales 0, día “2020-01-01”, hora “00:00”, cancha “test”, oficial de mesa “test”	Se muestra el mensaje “El número de cuartos debe ser mínimo 1”
PSF35	Creación incorrecta de un partido con el número de cuartos vacío	Eliminar require del campo número de cuartos. Introducir un partido con equipo local “test”, equipo visitante “test2”, número de cuartos vacío, duración de cada cuarto 1, tiempo corrido “false”, número de tiempos muertos 0, número máximo de faltas personales 0, día “2020-01-01”, hora “00:00”, cancha “test”, oficial de mesa “test”	Se muestra el mensaje “Número de cuartos vacío”
PSF36	Creación incorrecta de un partido con duración de cada cuarto menor que 1	Cambiar valor del mínimo del campo duración de cuarto por 0. Introducir un partido con equipo local “test”, equipo visitante “test2”, número de cuartos 4, duración de cada cuarto 0, tiempo corrido “false”, número de tiempos muertos 0, número máximo de faltas personales 0, día “2020-01-01”, hora “00:00”, cancha “test”, oficial de mesa “test”	Se muestra el mensaje “La duración de cada cuarto debe ser mínimo de 1 minuto”
PSF37	Creación incorrecta de un partido con duración de cada cuarto mayor que 60	Cambiar valor del mínimo del campo duración de cuarto por 61. Introducir un partido con equipo local “test”, equipo visitante “test2”, número de cuartos 4, duración de cada cuarto 61, tiempo corrido “false”, número de tiempos muertos 0, número máximo de faltas personales 0, día “2020-01-01”, hora “00:00”, cancha “test”, oficial de mesa “test”	Se muestra el mensaje “La duración de cada cuarto debe ser máximo de 60 minutos”
PSF38	Creación incorrecta de un partido con duración de cada cuarto vacío	Eliminar require del campo duración de cuarto. Introducir un partido con equipo local “test”, equipo visitante “test2”, número de cuartos 4, duración de cada cuarto vacío, tiempo corrido “false”, número de tiempos muertos 0, número máximo de faltas personales 0, día “2020-01-01”, hora “00:00”, cancha “test”, oficial de mesa “test”	Se muestra el mensaje “Duración de cada cuarto vacío”

Id	Descripción	Procedimiento	Salida esperada
PSF39	Creación incorrecta de un partido con equipo local inexistente	Cambiar el valor del atributo value de la opción seleccionada por “hola”. Introducir un partido con equipo local vacío, equipo visitante “test2”, número de cuartos 4, duración de cada cuarto 60, tiempo corrido “false”, número de tiempos muertos 0, número máximo de faltas personales 0, día “2020-01-01”, hora “00:00”, cancha “test”, oficial de mesa “test”	Se muestra el mensaje “El nombre del equipo local no existe”
PSF40	Creación incorrecta de un partido con equipo visitante inexistente	Cambiar el valor del atributo value de la opción seleccionada por “hola”. Introducir un partido con equipo local “test”, equipo visitante vacío, número de cuartos 4, duración de cada cuarto 60, tiempo corrido “false”, número de tiempos muertos 0, número máximo de faltas personales 0, día “2020-01-01”, hora “00:00”, cancha “test”, oficial de mesa “test”	Se muestra el mensaje “El nombre del equipo visitante no existe”
PSF41	Creación incorrecta de un partido con tiempo corrido diferente a “true” o “false”	Cambiar el valor del atributo value de la opción seleccionada por “hola”. Introducir un partido con equipo local “test”, equipo visitante “test2”, número de cuartos 4, duración de cada cuarto 60, tiempo corrido vacío, número de tiempos muertos 0, número máximo de faltas personales 0, día “2020-01-01”, hora “00:00”, cancha “test”, oficial de mesa “test”	Se muestra el mensaje “La opción de tiempo corrido no está bien seleccionada”
PSF42	Creación incorrecta de un partido con número de tiempos muertos negativo	Cambiar valor del mínimo del campo número de tiempos muertos por -1. Introducir un partido con equipo local “test”, equipo visitante “test2”, número de cuartos 4, duración de cada cuarto 60, tiempo corrido “true”, número de tiempos muertos -1, número máximo de faltas personales 0, día “2020-01-01”, hora “00:00”, cancha “test”, oficial de mesa “test”	Se muestra el mensaje “El número de tiempos muertos no puede ser negativo”
PSF43	Creación incorrecta de un partido con número de tiempos muertos vacío	Eliminar require del campo número de tiempos muertos. Introducir un partido con equipo local “test”, equipo visitante “test2”, número de cuartos 4, duración de cada cuarto 60, tiempo corrido “true”, número de tiempos muertos vacío, número máximo de faltas personales 0, día “2020-01-01”, hora “00:00”, cancha “test”, oficial de mesa “test”	Se muestra el mensaje “Número de tiempos muertos vacío”

Id	Descripción	Procedimiento	Salida esperada
PSF44	Creación incorrecta de un partido con número máximo de faltas personales negativo	Cambiar valor del mínimo del campo número de faltas personales por -1. Introducir un partido con equipo local “test”, equipo visitante “test2”, número de cuartos 4, duración de cada cuarto 60, tiempo corrido “true”, número de tiempos muertos 1, número máximo de faltas personales -1, día “2020-01-01”, hora “00:00”, cancha “test”, oficial de mesa “test”	Se muestra el mensaje “El número máximo de faltas personales no puede ser negativo”
PSF45	Creación incorrecta de un partido con número máximo de faltas personales vacío	Eliminar require del campo número de faltas personales. Introducir un partido con equipo local “test”, equipo visitante “test2”, número de cuartos 4, duración de cada cuarto 60, tiempo corrido “true”, número de tiempos muertos 1, número máximo de faltas personales vacío, día “2020-01-01”, hora “00:00”, cancha “test”, oficial de mesa “test”	Se muestra el mensaje “Máximo de faltas personales vacío”
PSF46	Creación incorrecta de un partido con nombre de cancha inexistente	Cambiar el valor del atributo value de la opción seleccionada por “hola”. Introducir un partido con equipo local “test”, equipo visitante “test2”, número de cuartos 4, duración de cada cuarto 60, tiempo corrido “true”, número de tiempos muertos 1, número máximo de faltas personales 1, día “2020-01-01”, hora “00:00”, cancha vacío, oficial de mesa “test”	Se muestra el mensaje “El nombre del lugar no existe”
PSF47	Creación incorrecta de un partido con fecha vacía	Eliminar require del campo fecha. Introducir un partido con equipo local “test”, equipo visitante “test2”, número de cuartos 4, duración de cada cuarto 60, tiempo corrido “true”, número de tiempos muertos 1, número máximo de faltas personales 1, día vacío, hora “00:00”, cancha “test”, oficial de mesa “test”	Se muestra el mensaje “La fecha está vacía”
PSF48	Creación incorrecta de un partido con hora vacía	Eliminar require del campo hora. Introducir un partido con equipo local “test”, equipo visitante “test2”, número de cuartos 4, duración de cada cuarto 60, tiempo corrido “true”, número de tiempos muertos 1, número máximo de faltas personales 1, día “2020-01-01”, hora vacío cancha “test”, oficial de mesa “test”	Se muestra el mensaje “La hora está vacía”

Id	Descripción	Procedimiento	Salida esperada
PSF49	Creación incorrecta de un partido con oficial de mesa inexistente	Cambiar el valor del atributo value de la opción seleccionada por “hola”. Introducir un partido con equipo local “test”, equipo visitante “test2”, número de cuartos 4, duración de cada cuarto 60, tiempo corrido “true”, número de tiempos muertos 1, número máximo de faltas personales 1, día “2020-01-01”, hora vacío cancha “test”, oficial de mesa “test”	Se muestra el mensaje “La hora está vacía”

Tabla 28. Pruebas de validación de creación de partido

Modificación de partido			
Id	Descripción	Procedimiento	Salida esperada
PSF50	Modificación incorrecta de un partido con el número de cuartos menor que 1	Cambiar valor del mínimo del campo número de cuartos por 0. Modificar un partido con equipo local “test”, equipo visitante “test2”, número de cuartos 0, duración de cada cuarto 1, tiempo corrido “false”, número de tiempos muertos 0, número máximo de faltas personales 0, día “2020-01-01”, hora “00:00”, cancha “test”, oficial de mesa “test”	Se muestra el mensaje “El número de cuartos debe ser mínimo 1”
PSF51	Modificación incorrecta de un partido con el número de cuartos vacío	Eliminar require del campo número de cuartos. Modificar un partido con equipo local “test”, equipo visitante “test2”, número de cuartos vacío, duración de cada cuarto 1, tiempo corrido “false”, número de tiempos muertos 0, número máximo de faltas personales 0, día “2020-01-01”, hora “00:00”, cancha “test”, oficial de mesa “test”	Se muestra el mensaje “Número de cuartos vacío”
PSF52	Modificación incorrecta de un partido con duración de cada cuarto menor que 1	Cambiar valor del mínimo del campo duración de cuarto por 0. Modificar un partido con equipo local “test”, equipo visitante “test2”, número de cuartos 4, duración de cada cuarto 0, tiempo corrido “false”, número de tiempos muertos 0, número máximo de faltas personales 0, día “2020-01-01”, hora “00:00”, cancha “test”, oficial de mesa “test”	Se muestra el mensaje “La duración de cada cuarto debe ser mínimo de 1 minuto”
PSF53	Modificación incorrecta de un partido con duración de cada cuarto mayor que 60	Cambiar valor del mínimo del campo duración de cuarto por 61. Modificar un partido con equipo local “test”, equipo visitante “test2”, número de cuartos 4, duración de cada cuarto 61, tiempo corrido “false”, número de tiempos muertos 0, número máximo de faltas personales 0, día “2020-01-01”, hora “00:00”, cancha “test”, oficial de mesa “test”	Se muestra el mensaje “La duración de cada cuarto debe ser máximo de 60 minutos”

Id	Descripción	Procedimiento	Salida esperada
PSF54	Modificación incorrecta de un partido con duración de cada cuarto vacío	Eliminar require del campo duración de cuarto. Modificar un partido con equipo local “test”, equipo visitante “test2”, número de cuartos 4, duración de cada cuarto vacío, tiempo corrido “false”, número de tiempos muertos 0, número máximo de faltas personales 0, día “2020-01-01”, hora “00:00”, cancha “test”, oficial de mesa “test”	Se muestra el mensaje “Duración de cada cuarto vacío”
PSF55	Modificación incorrecta de un partido con equipo local inexistente	Cambiar el valor del atributo value de la opción seleccionada por “hola”. Modificar un partido con equipo local vacío, equipo visitante “test2”, número de cuartos 4, duración de cada cuarto 60, tiempo corrido “false”, número de tiempos muertos 0, número máximo de faltas personales 0, día “2020-01-01”, hora “00:00”, cancha “test”, oficial de mesa “test”	Se muestra el mensaje “El nombre del equipo local no existe”
PSF56	Modificación incorrecta de un partido con equipo visitante inexistente	Cambiar el valor del atributo value de la opción seleccionada por “hola”. Modificar un partido con equipo local “test”, equipo visitante vacío, número de cuartos 4, duración de cada cuarto 60, tiempo corrido “false”, número de tiempos muertos 0, número máximo de faltas personales 0, día “2020-01-01”, hora “00:00”, cancha “test”, oficial de mesa “test”	Se muestra el mensaje “El nombre del equipo visitante no existe”
PSF57	Modificación incorrecta de un partido con tiempo corrido diferente a “true” o “false”	Cambiar el valor del atributo value de la opción seleccionada por “hola”. Modificar un partido con equipo local “test”, equipo visitante “test2”, número de cuartos 4, duración de cada cuarto 60, tiempo corrido vacío, número de tiempos muertos 0, número máximo de faltas personales 0, día “2020-01-01”, hora “00:00”, cancha “test”, oficial de mesa “test”	Se muestra el mensaje “La opción de tiempo corrido no está bien seleccionada”
PSF58	Modificación incorrecta de un partido con número de tiempos muertos negativo	Cambiar valor del mínimo del campo número de tiempos muertos por -1. Modificar un partido con equipo local “test”, equipo visitante “test2”, número de cuartos 4, duración de cada cuarto 60, tiempo corrido “true”, número de tiempos muertos -1, número máximo de faltas personales 0, día “2020-01-01”, hora “00:00”, cancha “test”, oficial de mesa “test”	Se muestra el mensaje “El número de tiempos muertos no puede ser negativo”

Id	Descripción	Procedimiento	Salida esperada
PSF59	Modificación incorrecta de un partido con número de tiempos muertos vacío	Eliminar require del campo número de tiempos muertos. Modificar un partido con equipo local “test”, equipo visitante “test2”, número de cuartos 4, duración de cada cuarto 60, tiempo corrido “true”, número de tiempos muertos vacío, número máximo de faltas personales 0, día “2020-01-01”, hora “00:00”, cancha “test”, oficial de mesa “test”	Se muestra el mensaje “Número de tiempos muertos vacío”
PSF60	Modificación incorrecta de un partido con número máximo de faltas personales negativo	Cambiar valor del mínimo del campo número de faltas personales por -1. Modificar un partido con equipo local “test”, equipo visitante “test2”, número de cuartos 4, duración de cada cuarto 60, tiempo corrido “true”, número de tiempos muertos 1, número máximo de faltas personales -1, día “2020-01-01”, hora “00:00”, cancha “test”, oficial de mesa “test”	Se muestra el mensaje “El número máximo de faltas personales no puede ser negativo”
PSF61	Modificación incorrecta de un partido con número máximo de faltas personales vacío	Eliminar require del campo número de faltas personales. Modificar un partido con equipo local “test”, equipo visitante “test2”, número de cuartos 4, duración de cada cuarto 60, tiempo corrido “true”, número de tiempos muertos 1, número máximo de faltas personales vacío, día “2020-01-01”, hora “00:00”, cancha “test”, oficial de mesa “test”	Se muestra el mensaje “Máximo de faltas personales vacío”
PSF62	Modificación incorrecta de un partido con nombre de cancha inexistente	Cambiar el valor del atributo value de la opción seleccionada por “hola”. Modificar un partido con equipo local “test”, equipo visitante “test2”, número de cuartos 4, duración de cada cuarto 60, tiempo corrido “true”, número de tiempos muertos 1, número máximo de faltas personales 1, día “2020-01-01”, hora “00:00”, cancha vacío, oficial de mesa “test”	Se muestra el mensaje “El nombre del lugar no existe”
PSF63	Modificación incorrecta de un partido con fecha vacía	Eliminar require del campo fecha. Modificar un partido con equipo local “test”, equipo visitante “test2”, número de cuartos 4, duración de cada cuarto 60, tiempo corrido “true”, número de tiempos muertos 1, número máximo de faltas personales 1, día vacío, hora “00:00”, cancha “test”, oficial de mesa “test”	Se muestra el mensaje “La fecha está vacía”

Id	Descripción	Procedimiento	Salida esperada
PSF64	Modificación incorrecta de un partido con hora vacía	Eliminar require del campo hora. Modificar un partido con equipo local “test”, equipo visitante “test2”, número de cuartos 4, duración de cada cuarto 60, tiempo corrido “true”, número de tiempos muertos 1, número máximo de faltas personales 1, día “2020-01-01”, hora vacío cancha “test”, oficial de mesa “test”	Se muestra el mensaje “La hora está vacía”
PSF65	Modificación incorrecta de un partido con oficial de mesa inexistente	Cambiar el valor del atributo value de la opción seleccionada por “hola”. Modificar un partido con equipo local “test”, equipo visitante “test2”, número de cuartos 4, duración de cada cuarto 60, tiempo corrido “true”, número de tiempos muertos 1, número máximo de faltas personales 1, día “2020-01-01”, hora vacío cancha “test”, oficial de mesa “test”	Se muestra el mensaje “La hora está vacía”

Tabla 29. Pruebas de validación de modificación de partido

Registro de usuario			
Id	Descripción	Procedimiento	Salida esperada
PSF66	Registro incorrecto de un usuario con nombre de usuario vacío	Eliminar require del campo nombre de usuario. Introducir un usuario con nombre de usuario vacío, contraseña “1234” y repetición de contraseña “1234”	Se muestra el mensaje “Campo nombre de usuario vacío”
PSF67	Registro incorrecto de un usuario con contraseña vacía	Eliminar require del campo contraseña. Introducir un usuario con nombre de usuario “test2”, contraseña vacía y repetición de contraseña vacía	Se muestra el mensaje “Campo contraseña vacío”
PSF68	Registro incorrecto de un usuario con repetición de contraseña vacía	Eliminar require del campo repetición de contraseña. Introducir un usuario con nombre de usuario “test2”, contraseña “1234” y repetición de contraseña vacía	Se muestra el mensaje “Las contraseñas no coinciden”

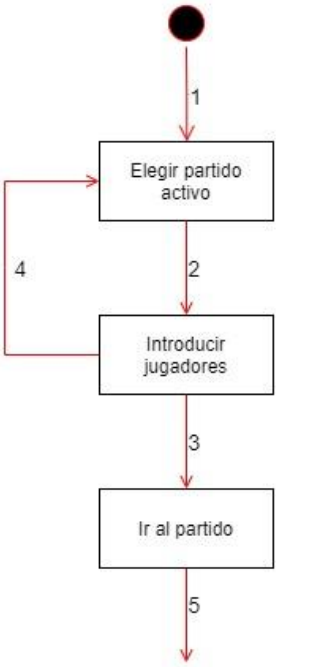
Tabla 30. Pruebas de validación de registro de usuario

Inicio de sesión			
Id	Descripción	Procedimiento	Salida esperada
PSF69	Inicio de sesión incorrecto con nombre de usuario vacío	Eliminar require del campo nombre de usuario. Introducir nombre de usuario vacío y contraseña “test”	Se muestra el mensaje “Campo nombre de usuario vacío”
PSF70	Inicio de sesión incorrecto con contraseña vacía	Eliminar require del campo contraseña. Introducir nombre de usuario “test” y contraseña vacía	Se muestra el mensaje “Campo contraseña vacío”

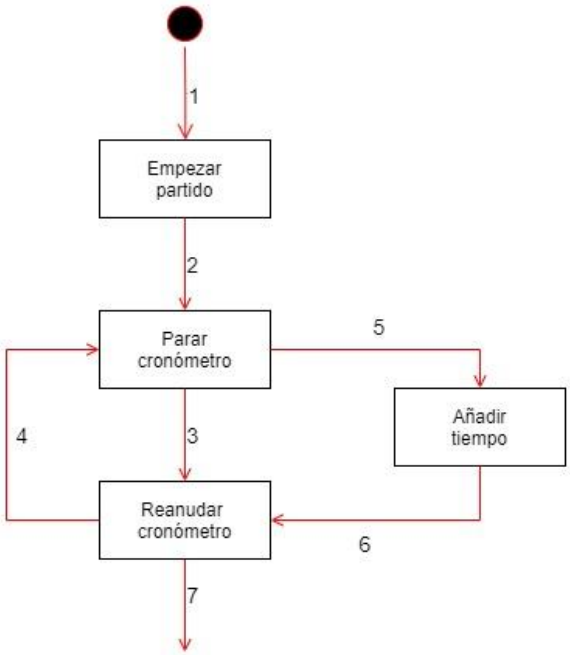
Tabla 31. Pruebas de validación de inicio de sesión

En el caso de la aplicación móvil, aplicando la técnica de pares de caminos se obtienen los pasos que hay que seguir para completar la prueba. A continuación, se muestran los diagramas que se utilizan para realizar estas pruebas y los caminos seguidos para verificar la correcta funcionalidad de la aplicación.

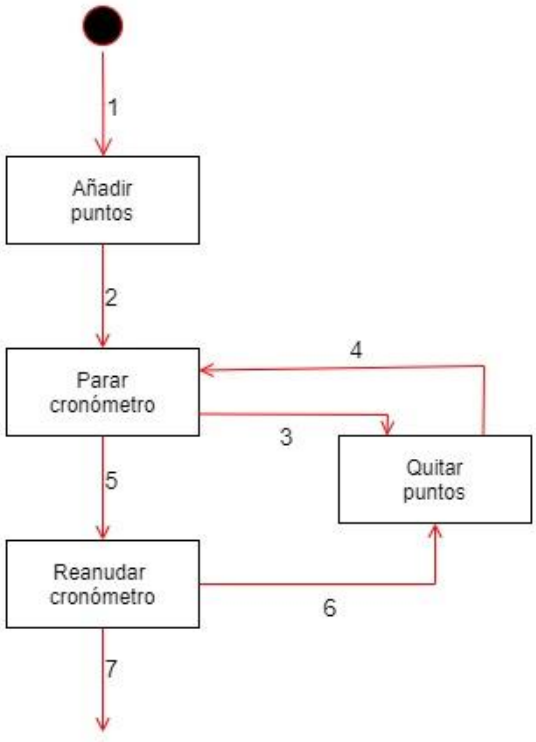
Hacer mesa de partido activo		
Id	Procedimiento	Salida esperada
PSF71	1-2-4-2-3-5 Elegir partido activo, introducir jugadores, volver a elegir partido activo, introducir jugadores, ir al partido	Se muestra la pantalla para empezar a hacer mesa del partido elegido. Aparece la acción en el seguimiento del partido



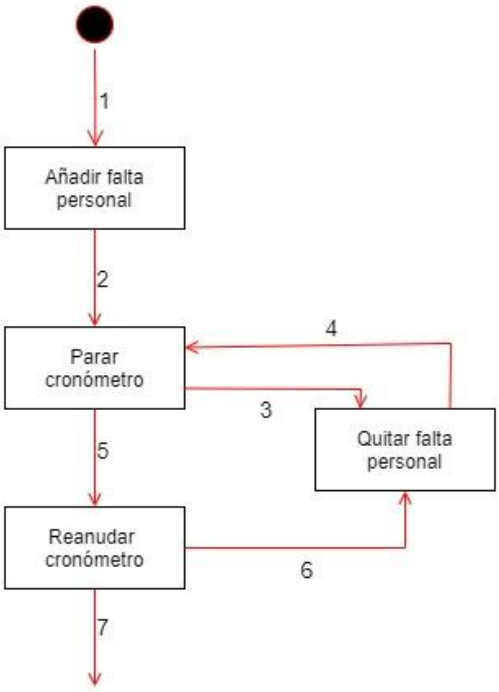
Cronómetro		
Id	Procedimiento	Salida esperada
PSF72	1-2-5-6-4-3-4-5-6-7 Empezar partido, parar cronómetro, añadir 1 segundo, reanudar cronómetro, parar cronómetro, reanudar cronómetro, parar cronómetro, añadir 1 segundo, reanudar cronómetro	El partido está empezado y el tiempo corriendo. El botón de parar cronómetro está habilitado y el de reanudar está deshabilitado. Aparecen las acciones en el seguimiento del partido
PSF73	1-2-3-7 Empezar partido, parar cronómetro, reanudar cronómetro	El partido está empezado y el tiempo corriendo. El botón de parar cronómetro está habilitado y el de reanudar está deshabilitado. Aparecen las acciones en el seguimiento del partido



Marcador		
Id	Procedimiento	Salida esperada
PSF74	1-2-3-4-3-4-5-6-4-5-7 Añadir 2 puntos, parar cronómetro, quitar 1 punto, parar cronómetro, reanudar cronómetro, quitar 1 punto, parar cronómetro, reanudar cronómetro	El marcador del equipo tiene los mismos puntos que antes de realizar el caso de prueba y el tiempo está corriendo. Aparecen las acciones en el seguimiento del partido
PSF75	1-2-5-7 Añadir 3 puntos, parar cronómetro, reanudar cronómetro	El marcador del equipo tiene 3 puntos más y el tiempo está corriendo. Aparece la acción en el seguimiento del partido



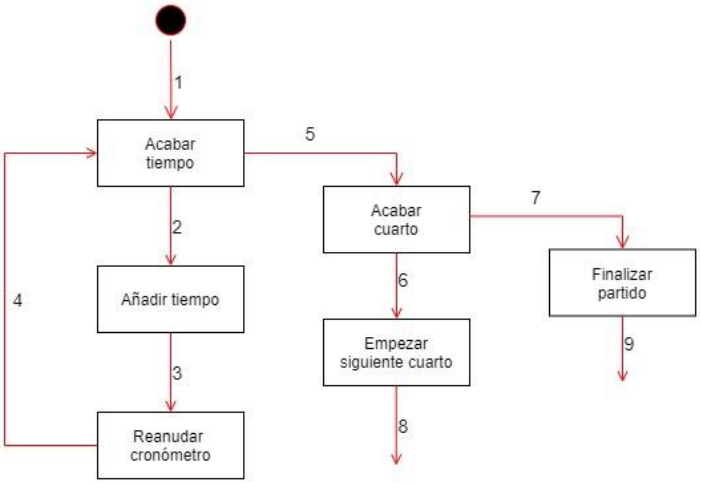
Faltas personales		
Id	Procedimiento	Salida esperada
PSF76	1-2-3-4-3-4-5-6-4-5-7 Añadir falta personal, parar cronómetro, quitar falta personal, parar cronómetro, reanudar cronómetro, quitar falta personal, parar cronómetro, reanudar cronómetro	El equipo tiene una falta personal menos que antes de realizar el caso de prueba y el tiempo está corriendo. Aparecen las acciones en el seguimiento del partido
PSF77	1-2-5-7 Añadir falta personal, parar cronómetro, reanudar cronómetro	El equipo tiene una falta personal más que antes de realizar el caso de prueba y el tiempo está corriendo. Aparece la acción en el seguimiento del partido



Tiempo muerto		
Id	Procedimiento	Salida esperada
PSF78	1-2-3-4 Añadir tiempo muerto, parar cronómetro, reanudar cronómetro	El equipo tiene un tiempo muerto más que antes de realizar el caso de prueba y el tiempo está corriendo. Aparece la acción en el seguimiento del partido

Otras infracciones		
Id	Procedimiento	Salida esperada
PSF79	1-2-3-4 Añadir otras infracciones, parar cronómetro, reanudar cronómetro	El tiempo está corriendo y se ha añadido la infracción. Aparece la acción en el seguimiento del partido

Acabar tiempo		
Id	Procedimiento	Salida esperada
PSF80	1-2-3-4-2-3-4-5-6-8 Acabar tiempo, añadir 1 segundo, reanudar cronómetro, acabar tiempo, añadir 1 segundo, reanudar cronómetro, acabar tiempo, acabar cuarto, empezar siguiente	El cuarto actual es el siguiente al que estaba en juego antes de realizar el caso de prueba y los botones están habilitados, excepto el de reanudar. Aparecen las acciones en el seguimiento del partido
PSF81	1-5-7-9 Acabar tiempo, acabar cuarto, finalizar partido	El cuarto que finalizó era el último y el partido está finalizado. Aparecen las acciones en el seguimiento del partido

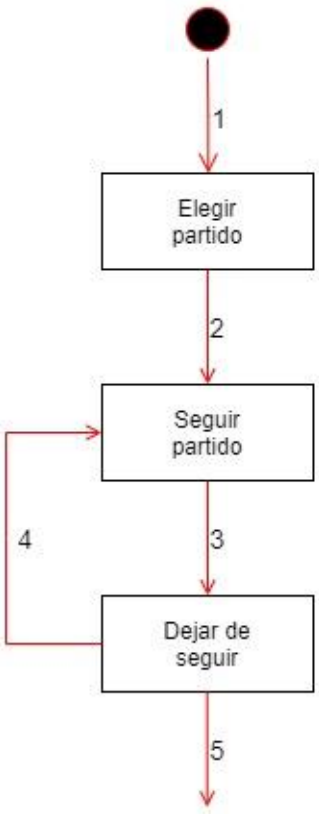


Cambio		
Id	Procedimiento	Salida esperada
PSF82	1-2-3-4-5-6 Añadir cambio, parar cronómetro, elegir jugador a sustituir, elegir sustituto, reanudar cronómetro	El tiempo está corriendo y se ha realizado el cambio. Aparece la acción en el seguimiento del partido

Partido por jugar		
Id	Procedimiento	Salida esperada
PSF83	1-2-3 Elegir partido por jugar, ver equipos, fecha, hora y lugar	Se muestran los equipos, la fecha, la hora y el lugar del partido elegido

Partido finalizado		
Id	Procedimiento	Salida esperada
PSF84	1-2-3-4 Elegir partido finalizado, ver eventos, ver resumen de jugadores	Se muestran todas las acciones del partido finalizado y el resumen de los jugadores

Seguir partido		
Id	Procedimiento	Salida esperada
PSF85	1-2-3-4-3-5 Elegir partido, seguir partido, dejar de seguir	Aparece el botón de seguir partido y no se muestra el partido en partidos seguidos



Partido en juego		
Id	Procedimiento	Salida esperada
PSF86	1-2-3-4-5-6 Elegir partido en juego, ver eventos pasados, ver eventos en directo, finaliza partido, ver resumen de jugadores	Se muestran todas las acciones del partido finalizado y el resumen de los jugadores

Pruebas de seguridad

Id	Descripción	Procedimiento	Salida esperada
PSS1	Acceder a la ruta /team/add sin estar identificado	Cerrar sesión si está abierta y escribir en el navegador la ruta /team/add	Redirige a la pantalla de inicio de sesión
PSS2	Acceder a la ruta /coach/add sin estar identificado	Cerrar sesión si está abierta y escribir en el navegador la ruta /coach/add	Redirige a la pantalla de inicio de sesión
PSS3	Acceder a la ruta /player/add sin estar identificado	Cerrar sesión si está abierta y escribir en el navegador la ruta /player/add	Redirige a la pantalla de inicio de sesión
PSS4	Acceder a la ruta /team/list sin estar identificado	Cerrar sesión si está abierta y escribir en el navegador la ruta /team/list	Redirige a la pantalla de inicio de sesión
PSS5	Acceder a la ruta /team/delete/id sin estar identificado	Cerrar sesión si está abierta y escribir en el navegador la ruta /team/delete/id	Redirige a la pantalla de inicio de sesión
PSS6	Acceder a la ruta / sin estar identificado	Cerrar sesión si está abierta y escribir en el navegador la ruta /	Redirige a la pantalla de inicio de sesión
PSS7	Acceder a la ruta /match/add sin estar identificado	Cerrar sesión si está abierta y escribir en el navegador la ruta /match/add	Redirige a la pantalla de inicio de sesión
PSS8	Acceder a la ruta /match/mine sin estar identificado	Cerrar sesión si está abierta y escribir en el navegador la ruta /match/mine	Redirige a la pantalla de inicio de sesión
PSS9	Acceder a la ruta /match/activate/id sin estar identificado	Cerrar sesión si está abierta y escribir en el navegador la ruta /match/actívate/id	Redirige a la pantalla de inicio de sesión
PSS10	Acceder a la ruta /match/modify/id sin estar identificado	Cerrar sesión si está abierta y escribir en el navegador la ruta /match/modify/id	Redirige a la pantalla de inicio de sesión
PSS11	Acceder a la aplicación móvil sin estar identificado	Abrir la aplicación móvil y darle al botón de iniciar sesión	La aplicación no deja visualizar otra pantalla que no sea la de inicio de sesión y la de registro

Tabla 32. Pruebas de seguridad

Pruebas de rendimiento y carga

Para realizar las pruebas de rendimiento y carga se utiliza Gatling, una herramienta de pruebas de rendimiento y carga [19]. Sirven para comprobar cómo se comporta el sistema sometiéndolo a varias peticiones a la vez a modo de uso de varios usuarios simultáneos utilizándolo.

La realización de las pruebas será en local por lo que podría existir alguna diferencia de resultados dependiendo de la plataforma o servidor en el que finalmente se implante el sistema.

Para el diseño de estas pruebas se han tenido en cuenta las posibles acciones que podría realizar un usuario estándar en la aplicación web para simular el comportamiento real que tendría el

sistema una vez implantado. Por lo tanto, los pasos que seguirá el usuario simulado son las siguientes:

- Inicio de sesión.
- Crear equipo.
- Añadir entrenador.
- Añadir 5 jugadores.
- Crear partido.
- Activar partido.
- Cerrar sesión.

No se contemplan todas las funcionalidades de la aplicación web ya que, como se ha mencionado anteriormente, se simula el posible uso de un usuario en el sistema real.

Por último, cabe destacar que la carga a la que se somete al sistema será escalonada, viniendo determinado el intervalo de tiempo entre un usuario y otro por 5 segundos, hasta alcanzar los 100 usuarios conectados para satisfacer el requisito no funcional RNF1.

Pruebas de usabilidad

Antes de realizar el seguimiento, el usuario deberá rellenar un cuestionario con cinco preguntas que ayuden a conocer más sobre el perfil con el que se va a trabajar: edad, nivel de informática, uso de aplicaciones web, uso de aplicaciones móvil y familiarización con el tema.

Una vez obtenidos los datos del cuestionario se le guiará para realizar las siguientes tareas:

- Registrarse en la web.
- Cerrar sesión en la web.
- Iniciar sesión en la web.
- Crear equipo.
- Añadir entrenador.
- Añadir los jugadores que quiera.
- Crear partido.
- Modificar partido creado.
- Activar partido creado.
- Iniciar sesión en móvil.
- Hacer mesa en el partido activado.
- Completar con 6 jugadores en cada equipo.
- Iniciar partido.
- Añadir puntos al equipo local.
- Reanudar.
- Añadir puntos al equipo visitante.
- Quitar un punto al equipo visitante.
- Reanudar.
- Añadir falta personal al equipo local.
- Añadir un segundo al tiempo.
- Añadir falta personal al equipo visitante.
- Quitar falta personal previa.
- Reanudar.
- Añadir tiempo muerto.

- Reanudar.
- Añadir pie del equipo local.
- Reanudar.
- Añadir técnica al entrenador local.
- Reanudar.
- Hacer un cambio de jugadores.
- Reanudar.
- Acabar cuarto.
- Empezar cuarto.
- Finalizar partido.
- Seguir el partido.
- Ver resumen de los jugadores.
- Dejar de seguir el partido.

Durante el transcurso de estas pruebas con cada usuario se apuntan algunas observaciones para poder mejorar el sistema de cara a la usabilidad.

5.5.4 Pruebas de aceptación

Para diseñar los casos de prueba de este nivel se parte de las historias de usuario descritas en los requisitos funcionales del sistema, ya que es la funcionalidad que espera el cliente del sistema.

Id	Descripción	Procedimiento	Salida esperada
PA1	Como administrador quiero introducir equipos para poder crear todo tipo de partidos.	Entrar en la aplicación web con un usuario correcto. Ir a Gestión de equipos -> Crear equipo. Introducir los datos del equipo y crear equipo	Se pueden introducir el nombre del equipo, la cancha, el entrenador y los jugadores. Se crea el equipo
PA2	Como administrador quiero crear un partido que no pertenezca a una competición para poder hacer el seguimiento.	Entrar en la aplicación web con un usuario correcto. Ir a Gestión de partidos -> Crear partido. Introducir los datos del partido y crear partido	Se pueden introducir los equipos, número de cuartos, duración de cada cuarto, número de tiempos muertos, número máximo de faltas personales, fecha, hora, lugar, tiempo corrido y oficial de mesa. Se crea el partido
PA3	Como administrador quiero activar un partido una vez creado para que los usuarios identificados puedan visualizarlo.	Entrar en la aplicación web con un usuario correcto. Ir a Gestión de partidos -> Mis partidos. Activar un partido que no está activo	Aparece la opción de activar partido y cuando se activa ya no se puede volver a activar

Id	Descripción	Procedimiento	Salida esperada
PA4	Como oficial de mesa quiero elegir un partido activo para poder introducir las acciones.	Entrar en la aplicación móvil con un usuario correcto. Ir a Hacer mesa. Elegir un partido de los disponibles	Aparecen los partidos activos que tiene asignados ese usuario como oficial de mesa o los que no tienen oficial de mesa asignado
PA5	Como oficial de mesa quiero introducir los nombres de los jugadores que falten.	Elegir un partido activo para hacer mesa. Introducir los nombres y los dorsales de los jugadores del partido si no están ya asignados al equipo	Aparecen los jugadores de los equipos y se pueden añadir más con dorsales diferentes a los ya existentes en el mismo equipo
PA6	Como oficial de mesa quiero poner en marcha el cronómetro para iniciar el partido.	Pulsar el botón de iniciar partido	Aparece el primer cuarto y el tiempo va corriendo hacia atrás con el tiempo que queda para finalizar el cuarto. Se habilitan todos los botones
PA7	Como oficial de mesa quiero parar el cronómetro cuando se pare el partido sin informar de ninguna acción.	Pulsar el botón de parar el cronómetro	Se para el tiempo si el tiempo está corriendo y el partido no es a tiempo corrido. Se deshabilita el botón de parar y se habilitan los de reanudar y de añadir tiempo
PA8	Como oficial de mesa quiero añadir tiempo al cronómetro.	Parar el cronómetro. Pulsar el botón más de la derecha para añadir un segundo y el botón de la izquierda para añadir un minuto	El tiempo se incrementa en un minuto y un segundo
PA9	Como oficial de mesa quiero reanudar el cronómetro cuando se vuelva a reanudar el partido.	Pulsar el botón de reanudar el cronómetro	Se reanuda el tiempo si estaba parado. Se deshabilitan los botones de reanudar y de añadir tiempo y se habilita el de parar
PA10	Como oficial de mesa quiero añadir puntos a ambos equipos para ir actualizando el resultado.	Pulsar el botón de añadir 1, 2 y 3 puntos en cada uno de los equipos y elegir el jugador anotador	Se actualiza el marcador con la suma de los puntos indicados

Id	Descripción	Procedimiento	Salida esperada
PA11	Como oficial de mesa quiero quitar puntos a ambos equipos para ir actualizando el resultado.	Pulsar el botón de quitar un punto en los dos equipos y elegir un jugador que hubiera anotado	Se actualiza el marcador restando los puntos indicados
PA12	Como oficial de mesa quiero añadir faltas personales a los jugadores que las cometan.	Pulsar el botón de Añadir falta personal y elegir un jugador del equipo	Se actualizan las faltas del equipo sumando una falta personal
PA13	Como oficial de mesa quiero quitar faltas personales a un jugador.	Pulsar el botón de Quitar falta personal y elegir un jugador que haya cometido alguna falta	Se actualizan las faltas del equipo restando una falta personal
PA14	Como oficial de mesa quiero añadir tiempos muertos a cada equipo.	Pulsar el botón de Tiempo muerto de uno de los equipos	Se para el tiempo si el partido no es a tiempo corrido y el equipo cuenta con un tiempo muerto más
PA15	Como oficial de mesa quiero añadir otras infracciones para informar a los seguidores del partido de este tipo de acciones.	Pulsar el botón de Otras infracciones de uno de los equipos. Elegir la infracción y, dependiendo de si es de jugador o de equipo, elegir jugador	Si cuenta como una falta personal se suma a las faltas del equipo, si no se añade la infracción y se para el tiempo si el partido no es a tiempo corrido
PA16	Como oficial de mesa quiero acabar un cuarto.	Pulsar el botón de Finalizar cuarto cuando el cronómetro está a 00:00	Se deshabilitan todos los botones menos el de Empezar cuarto
PA17	Como oficial de mesa quiero iniciar un cuarto.	Pulsar el botón de Empezar cuarto	Se actualiza el número de cuarto y se habilitan los botones
PA18	Como oficial de mesa quiero introducir un cambio de un jugador para mostrarlo a los seguidores.	Pulsar el botón de cambio, elegir el jugador a sustituir y elegir el jugador sustituto	Se para el tiempo si el partido no es a tiempo corrido

Id	Descripción	Procedimiento	Salida esperada
PA19	Como oficial de mesa quiero finalizar un partido para incluirlo en el histórico.	Pulsar el botón de Finalizar partido cuando el cronómetro está a 00:00	Se acaba el partido y se muestra el menú principal
PA20	Como seguidor quiero elegir un partido activo para seguirlo y ver todas las acciones que acontecen.	Elegir Ver partido en el menú principal. Elegir un partido y pulsar en el botón de Seguir partido	Se muestran las acciones del partido y se habilita el botón de Dejar de seguir. El partido aparece en Partidos seguidos
PA21	Como administrador quiero modificar los datos de un partido.	Entrar en la aplicación web con un usuario correcto. Ir a Gestión de partidos -> Modificar partido. Pulsar en el botón de Modificar en la fila del partido. Introducir los datos del partido y modificar partido	Se pueden modificar los equipos, número de cuartos, duración de cada cuarto, número de tiempos muertos, número máximo de faltas personales, fecha, hora, lugar, tiempo corrido y oficial de mesa. Se modifica el partido
PA22	Como administrador quiero eliminar un equipo para que no aparezca en el sistema.	Entrar en la aplicación web con un usuario correcto. Ir a Gestión de equipos -> Ver equipos. Pulsar en el botón Eliminar del equipo que se quiere eliminar	Si el equipo no está en ningún partido se elimina y no aparece en la lista. Si está en algún partido no se elimina y se muestra un mensaje
PA23	Como usuario no registrado quiero registrarme para poder autenticarme en el sistema.	Entrar en la aplicación web e ir al formulario de Registro. Introducir nombre de usuario, contraseña y repetición de contraseña	Si el usuario no está repetido y la contraseña y la repetición de contraseña coinciden se registra el usuario y se entra en el sistema
PA24	Como usuario registrado y no identificado quiero autenticarme en el sistema con mis datos de registro para acceder a la funcionalidad del sistema.	Entrar en la aplicación web e ir al formulario de Inicio de sesión. Introducir nombre de usuario y contraseña	Si el usuario y la contraseña son correctos se entra en el sistema

Id	Descripción	Procedimiento	Salida esperada
PA25	Como seguidor quiero elegir un partido en concreto ya finalizado para ver un resumen.	Entrar en Ver partido y elegir un partido finalizado. Para ver el resumen de los jugadores pulsar en el botón Resumen de jugadores	Se muestran los equipos del partido, el marcador y las acciones. En el resumen de los jugadores se muestran los puntos, las faltas personales, las técnicas y las antideportivas de cada uno
PA26	Como seguidor quiero ver los partidos por jugar para ver un resumen.	Entrar en Ver partido y elegir un partido por jugar	Se muestran los equipos, la fecha, la hora y el lugar

Tabla 33. Pruebas de aceptación

Capítulo 6. Implementación del sistema

6.1 Preparación de entornos

Para realizar este trabajo se necesitan principalmente dos entornos de desarrollo: Android Studio y WebStorm.

Android Studio

Para instalar el primero de ellos hay que acudir a la página de Android para desarrolladores y hacer click en el apartado de Android Studio [10]. Una vez dentro del mencionado sitio sólo hay que darla al botón de descargar como se muestra, a continuación, en la Figura 4.

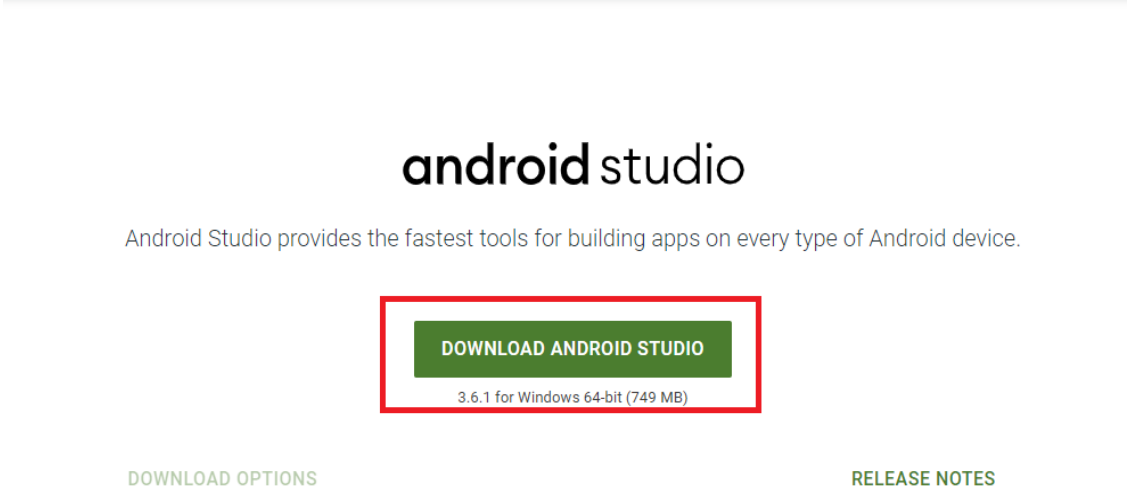


Figura 41. Botón de descarga de Android Studio remarcado en rojo dentro de la página web

También existen más opciones de descarga para otros sistemas operativos. Para acceder a ellas habrá que ir a *Download Options*. Para la instalación en Windows solo hace falta ejecutar el .exe descargado anteriormente.

Una vez descargado e instalado el entorno es el momento de crear un nuevo proyecto. Para ello se abre la aplicación Android Studio y se crea un nuevo proyecto. En el diálogo para crear un nuevo proyecto pide tres campos: el nombre de la aplicación, el dominio de la compañía y la ubicación del proyecto. En este caso, el nombre de la aplicación es TFGBasket, el dominio de la compañía es cristian.tfg.com, este campo simplemente determina la jerarquía de carpetas del proyecto, y, por último, se elige una ubicación del PC. No se incluye ni soporte para C++ ni para Kotlin porque no se van a utilizar. En la Figura 5 se muestra esta información.

A continuación, se decide en qué dispositivos Android se va a poder visualizar la aplicación, en este caso solo se selecciona para teléfono y Tablet ya que es lo que utilizará el administrador cuando utilice la app. En cuanto a las versiones de Android que soportarán la aplicación, lo ideal sería en todas, pero si se elige una versión inferior para que todos los dispositivos la soporten, las características disponibles se reducirán. Por tanto, se intenta buscar una versión intermedia que tengan la gran mayoría de dispositivos y no disponga de una funcionalidad muy básica. El propio Android Studio muestra una aproximación sobre los dispositivos que la soportan. Se elige un nivel API 16 para Android 4.1 (Jelly Bean) para que la aplicación se pueda ejecutar en el 99,6% de los dispositivos como se muestra en la Figura 6.

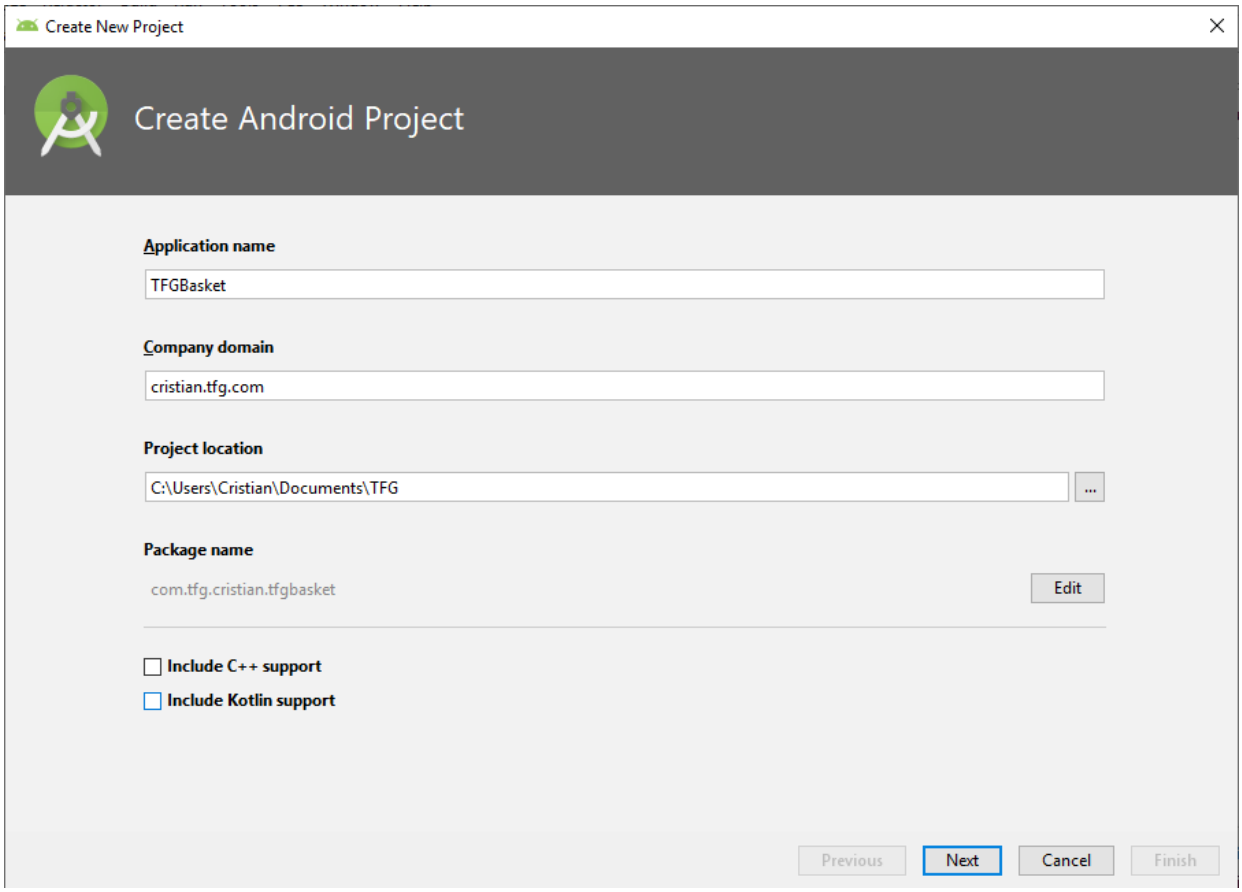


Figura 42. Diálogo para crear un nuevo proyecto en Android Studio

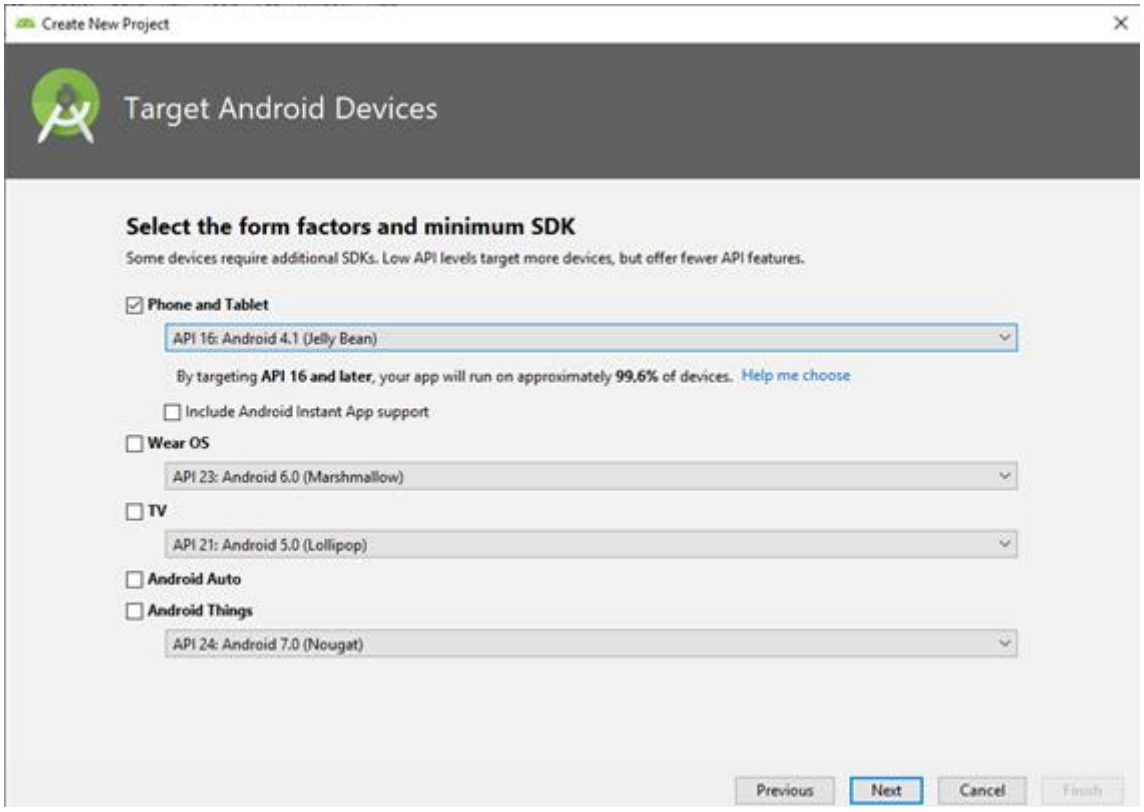


Figura 43. Elección de los dispositivos para los cuales está disponible la app y de la versión de Android

Por último, se pide elegir una actividad y se elige una vacía (Empty Activity en la Figura 7). A esta actividad principal se le da un nombre a la propia actividad y al diseño (layout) de la misma, en este caso el nombre de la actividad es MainActivity y el nombre del Layout es activity_main. Ambos valores son los que vienen por defecto. Para terminar, se hace click en el botón Finish y ya está creado el proyecto sobre el que se trabajará.

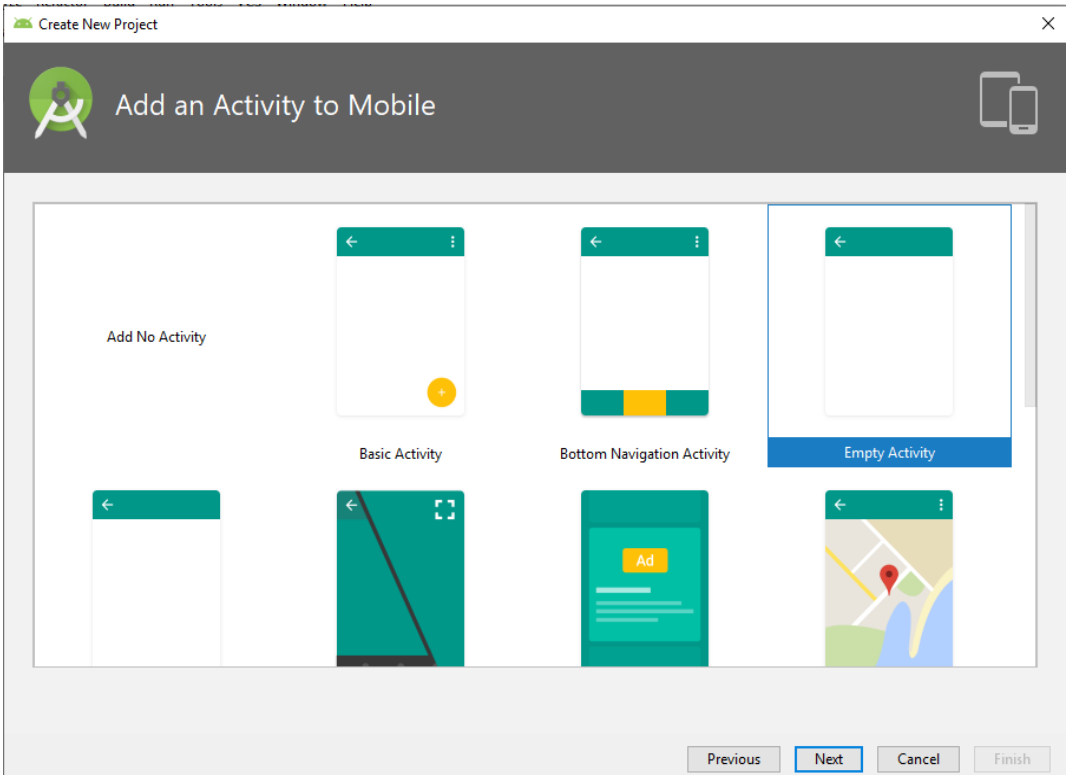


Figura 44. Diálogo para seleccionar el tipo de actividad

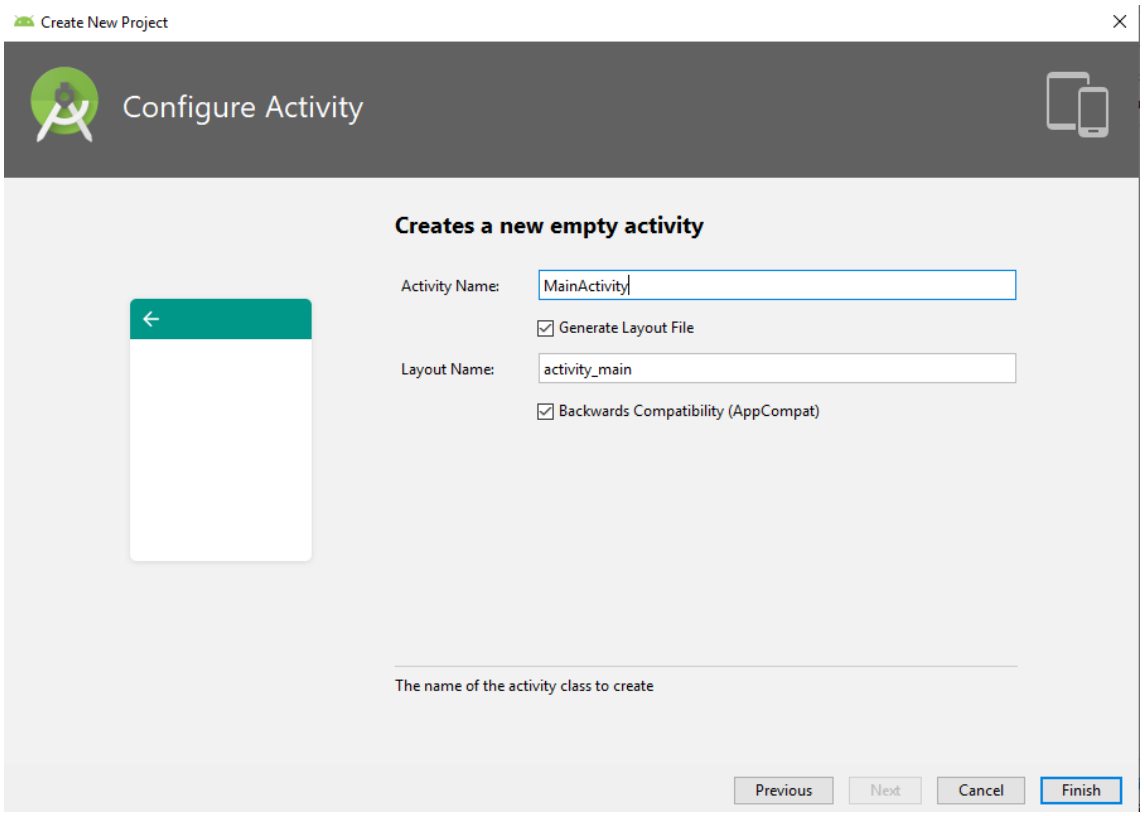


Figura 45. Diálogo para dar nombre a la actividad elegida y a su diseño (layout)

Para ejecutar la aplicación durante el desarrollo se utilizará un emulador virtual. De entre los dispositivos disponibles se eligen un Nexus 5X de 5,2" y una Tablet Nexus 10 de 10,05". Ambos dispositivos utilizan un nivel de API 29, es decir, superior al que soporta la aplicación, por lo que debería de visualizarse la aplicación en ellos.

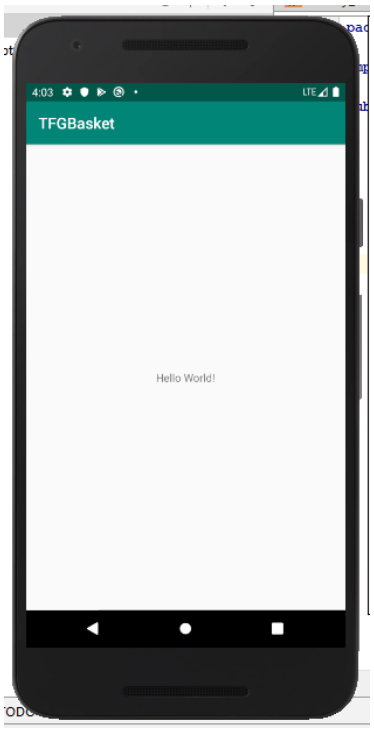


Figura 46. Visualización de la aplicación en el emulador Nexus 5X

WebStorm

Para instalar este segundo entorno se acude a la página de JetBrains, se busca WebStorm y se hace click en el botón de descargar. En la referencia [11] se encuentra el enlace para la descarga. Una vez descargado el instalador, se procede a instalar la herramienta ejecutándolo.

La creación de un nuevo proyecto en este entorno es mucho más simple que en Android Studio. Cuando ya está instalado, se crea un nuevo proyecto Node.js. Para ello se elige una ubicación donde se va a guardar y el gestor de paquetes, que en este caso se utilizará npm, como se muestra en la Figura 10. Una vez hacho esto el proyecto ya estaría creado para poder empezar a trabajar en él.

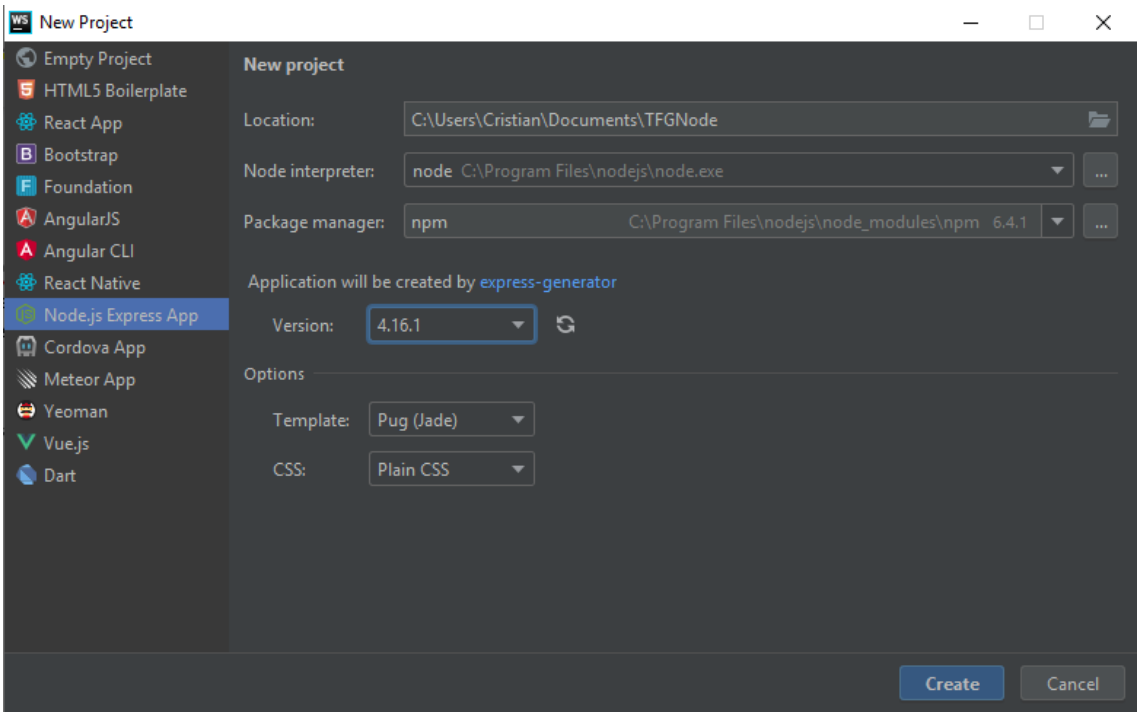


Figura 47. Crear un nuevo proyecto en WebStorm

GitHub

Para almacenar los proyectos y llevar un control de versiones se utilizará la herramienta GitHub. Para ello se crean dos repositorios, uno para cada proyecto (móvil y web).

En ambos casos es muy fácil de compartir el proyecto en GitHub. Para el entorno Android Studio se sigue la siguiente ruta: VCS > Import into Version Control > Share Project on GitHub. Se introducen las credenciales de GitHub y se elige un nombre para el repositorio, como se muestra en la Figura 11. En el caso de WebStorm habría que seguir los mismos pasos.

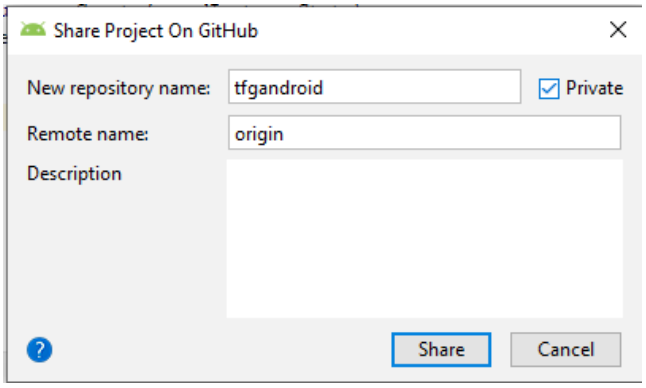


Figura 48. Compartir proyecto de Android Studio en GitHub

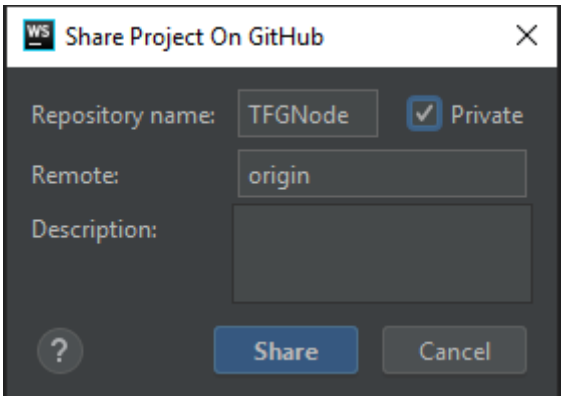


Figura 49. Compartir proyecto de WebStorm en GitHub

6.2 Generación del código

Para tener una referencia más clara del código generado se incluyen comentarios en cada uno de los métodos. Se muestran las partes más importantes del código de los módulos y clases más relevantes en el proyecto en el apéndice Documentación del código. En el caso del código generado para Node js la herramienta utilizada para generar la documentación propia de cada uno de los métodos es jsdoc y en el caso de la documentación de las clases java realizadas en Android Studio se utiliza Javadoc.

Los módulos más importantes realizados mediante Node js son el gestor de la base de datos, el validador de equipos, el validador de partidos y el validador de usuarios. Todos los demás módulos se han utilizado para referenciar las rutas de la aplicación web y utilizan estos cuatro mencionados para llevar a cabo la funcionalidad de la aplicación. En las siguientes figuras se muestra el módulo al que se hace referencia, una breve descripción y los métodos con su descripción y sus parámetros. El código y la documentación están en inglés ya que la mayoría de documentación software disponible se encuentra en ese idioma y en este ámbito de la informática es un idioma muy utilizado.

En el caso del código generado en Java con Android Studio las clases más relevantes son los eventos, que para abreviar se mostrará únicamente la documentación generada para la interfaz y que es extensible a todos los eventos, la clase 'partido', la clase 'usuario' y los servicios. Las demás clases que representan entidades tienen funcionalidad muy básica que trata sobre todo con sus propios atributos, y las clases de tipo 'Activity' hacen de intermediarias entre la interfaz

gráfica de usuario y los demás componentes como se ha explicado en el diseño de la arquitectura.

6.3 Ejecución de las pruebas

En este apartado se mostrarán los resultados de cada una de las pruebas especificadas anteriormente en 5.3 Diseño de las pruebas. Antes de ejecutar algunas de las pruebas se debe llenar la base de datos con unos datos específicos. Para ello, existe un script en el entorno de pruebas al que se accede mediante Node js y se deja la base de datos en estado consistente con los datos que se tienen en cuenta en las pruebas. En cada uno de los subapartados siguientes se explicará cuando realizar este relleno.

6.3.1 Pruebas unitarias

Este nivel de pruebas no necesita el rellenado de datos mencionado anteriormente.

Todas las pruebas unitarias realizadas pasan.

A continuación, se muestran los resultados de la ejecución de las pruebas unitarias de la aplicación web. Se muestra una figura en la que se muestra un mensaje de finalización al acabar.

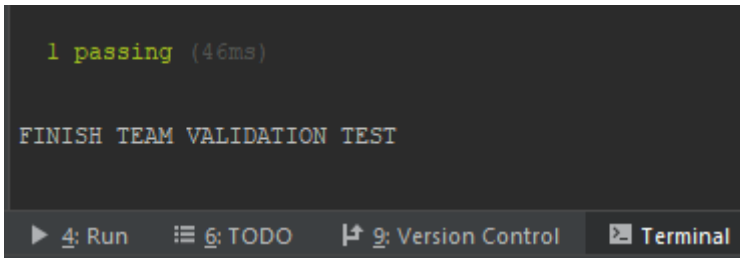


Figura 50. Resultado de las pruebas unitarias del módulo teamValidationManager

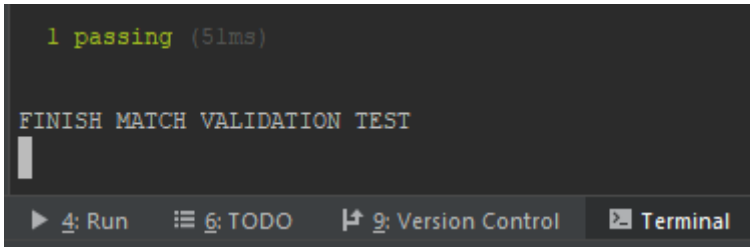


Figura 51. Resultado de las pruebas unitarias del módulo matchValidationManager

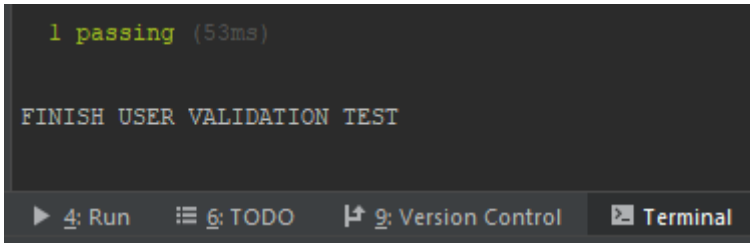


Figura 52. Resultado de las pruebas unitarias del módulo userValidationManager

En el caso de la aplicación móvil se han ejecutado los casos de prueba diseñados en el diseño de las pruebas y los resultados se muestran a continuación.

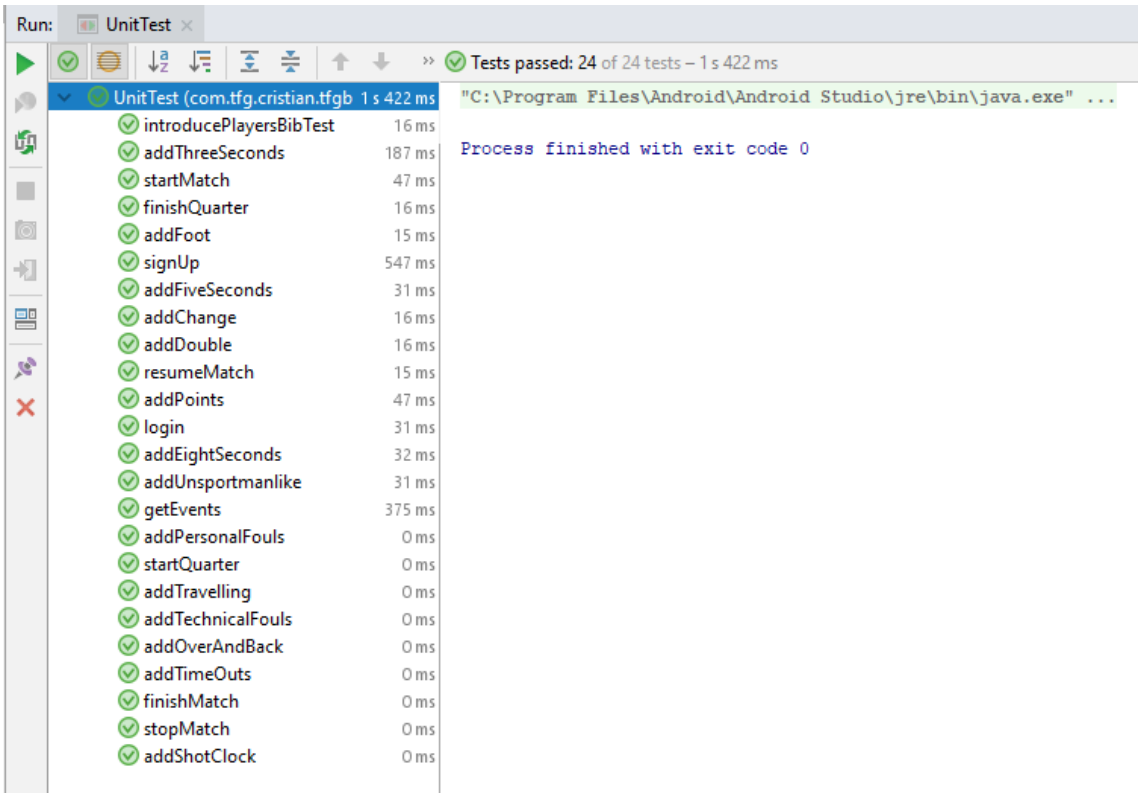


Figura 53. Resultado de las pruebas unitarias de la aplicación móvil

6.3.2 Pruebas de integración

Este nivel de pruebas sí que necesita el rellenado de datos.

Se han dividido de acuerdo con el diseño realizado y se obtienen tantos casos como en el diseño. El resultado de la ejecución de las pruebas es el siguiente:

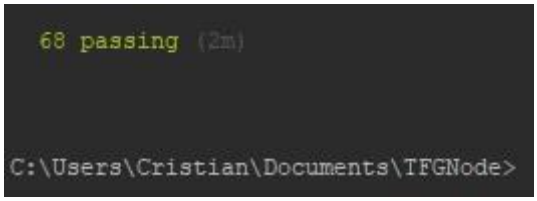


Figura 54. Resultado de las pruebas de integración

Todos los casos de prueba diseñados para las pruebas de integración pasan.

6.3.3 Pruebas de sistema

Estas pruebas y sus distintos tipos se han ejecutado de forma muy diversa. A continuación, se explicará con más detalla la ejecución de cada tipo de pruebas realizado.

Pruebas funcionales

Para la ejecución de estas pruebas también es necesario realizar el rellenado de datos y que el servidor del entorno de pruebas esté disponible.

Por una parte, las pruebas funcionales de la aplicación web se han realizado con Selenium y con ayuda del geckodriver de Firefox. Antes de realizar las pruebas se debe rellenar la base de datos

con los datos apropiados como ya se ha explicado y, una vez que se terminan todas las pruebas el propio caso de test JUnit navega hasta el script que hace el rellenado de nuevo.

En total estas pruebas se componen de 23 métodos en los que se comprueba la funcionalidad del sistema desde el punto de vista del usuario. El resultado de la ejecución de todos estos métodos se muestra a continuación. Como se puede observar todas pasan.

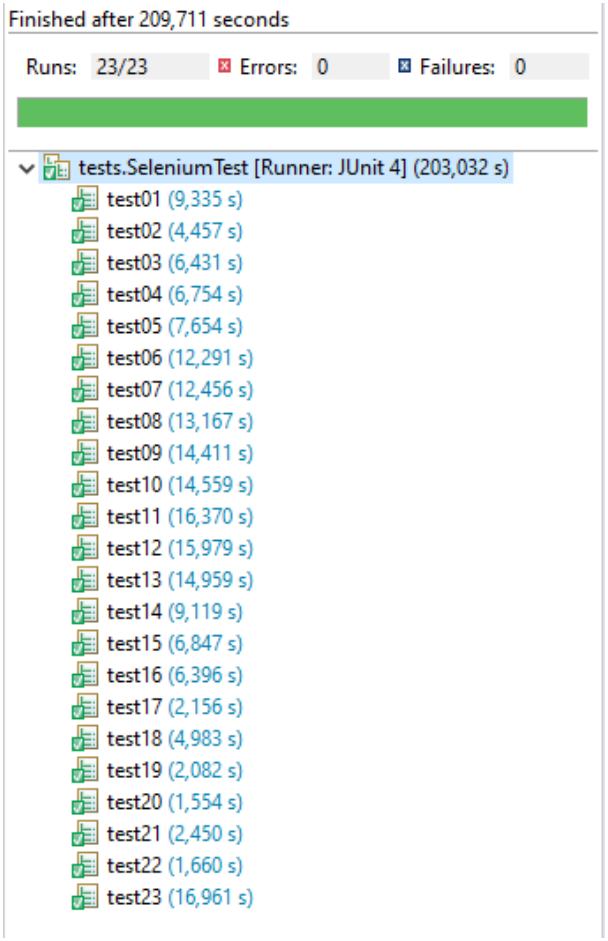


Figura 55. Resultado de las pruebas de sistema funcionales de la aplicación web con Selenium

Las pruebas de validación de formularios realizadas manualmente pasan todas.

En el caso de la aplicación móvil se han diseñado pruebas basadas en la técnica de pares de caminos y se han realizado manualmente. Todos los casos han resultado favorables.

Pruebas de seguridad

Este tipo de pruebas se han realizado manualmente ya que requería de la manipulación del html para intentar buscar vulnerabilidades que dejaran el sistema en un estado inconsistente. Todas estas pruebas se han realizado favorablemente.

Pruebas de rendimiento y carga

Este tipo de pruebas se han realizado utilizando Gatling y siguiendo los pasos determinados en el diseño del Plan de Pruebas. El resultado de estas pruebas se ha recogido en un fichero html cuyo contenido se muestra más adelante.

Se han realizado un total de 7684 peticiones, de las cuales 4023 se han realizado en menos de 800 milisegundos, 238 entre 800 y 1200 milisegundos, 3201 en más de 1200 milisegundos y 222 han fallado. Con esto se puede comprobar que la mayor parte de las peticiones se realizan en menos de 800 milisegundos o tardan más de 1200 milisegundos. Cabe destacar que el porcentaje de peticiones que han fallado es muy bajo. A continuación, se muestra un gráfico en el que se ve la proporción de la duración de las peticiones.

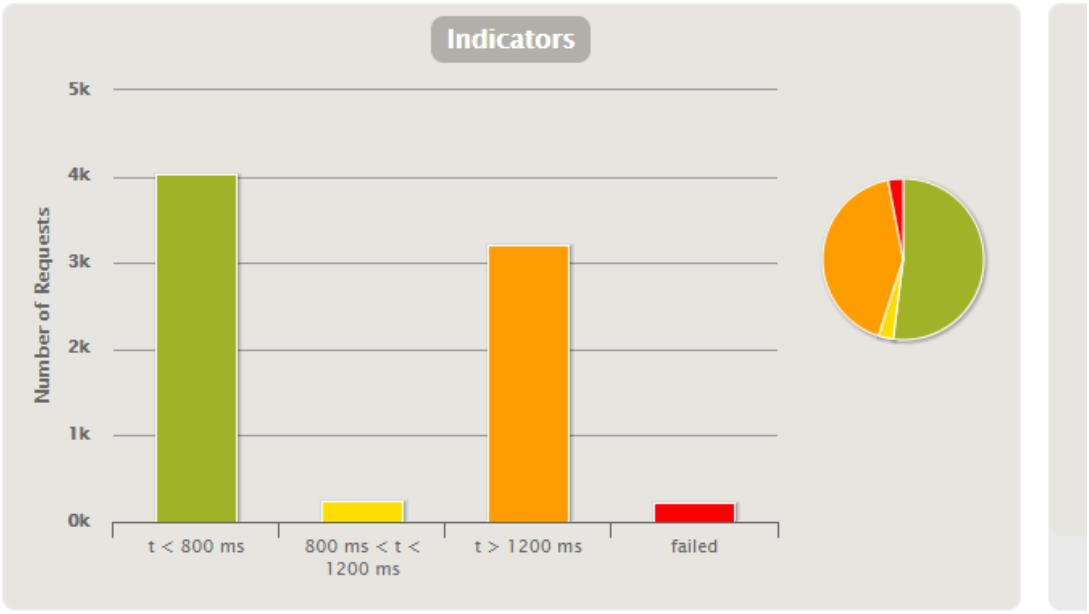


Figura 56. Gráfica de la duración de las peticiones realizadas en las pruebas de rendimiento y carga

Durante casi todo el transcurso de las pruebas se ha mantenido el número de usuarios en 100. Al principio subiendo muy rápido y al final reduciéndose el número poco a poco.

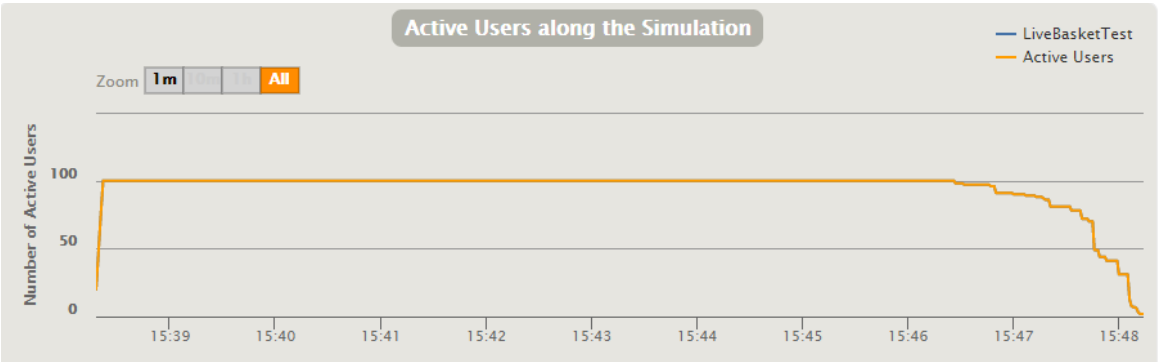


Figura 57. Gráfica de los usuarios activos durante las pruebas de rendimiento y carga

Por último, el último dato destacable de la ejecución de estas pruebas es la distribución del tiempo de respuesta, la cual es muy favorable ya que muestra un pico a la izquierda que va disminuyendo conforme aumenta el tiempo de respuesta. En la gráfica siguiente se puede ver que algo más del 50% de las respuestas se han realizado en 303 milisegundos.

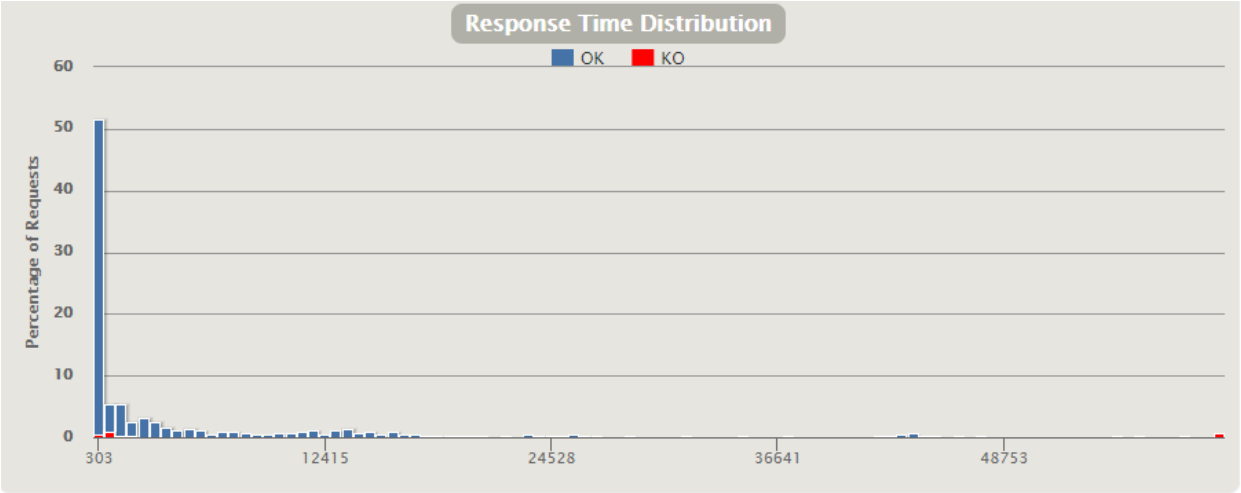


Figura 58. Gráfica de los tiempos de respuesta durante las pruebas de rendimiento y carga

Pruebas de usabilidad

Para realizar estas pruebas se ha escogido a 8 personas de diferentes perfiles para observar cómo se desenvuelven con el sistema. En el cuestionario de los Apéndices se pueden encontrar los resultados individuales de cada una de las personas a las que se han sometido estas pruebas. Como ya se ha especificado en el diseño del Plan de Pruebas, antes de realizar unas acciones monitorizadas en el sistema, estas personas deben rellenar un formulario.

No se han observado grandes diferencias en el uso del sistema por parte de los diferentes perfiles y, en general, las pruebas han sido satisfactorias. Si bien es cierto que existen algunas pequeñas cuestiones que se deben tener en cuenta. A modo de resumen se muestran en las siguientes tablas los resultados obtenidos de forma agregada por perfiles.

Edad	Observaciones
20-30	En general esta franja de edad utiliza el calendario y el reloj que ofrece el navegador a la hora de introducir la fecha y hora de un partido. En la pantalla en la que se hace mesa de un partido no descubrieron que haciendo scroll había más opciones. El registro lo realizaron correctamente sin intentar iniciar sesión antes.
45-55	Para realizar el registro primero se dieron cuenta de que no pueden iniciar sesión ya que es la primera pantalla que aparece.

Tabla 34. Pruebas de usabilidad por rango de edad

Conocimiento del tema	Observaciones
Bajo	Dificultad al añadir los jugadores del partido porque no sabían cómo ni dónde añadir el nombre y el dorsal.
Medio	Dificultad al añadir acciones de un nivel superior de conocimiento como añadir pie de un jugador.
Alto	No se observó ninguna dificultad en lo relacionado con la temática del sistema.

Tabla 35. Pruebas de usabilidad por conocimiento de la temática

Una cuestión que se observó repetidamente en todos los usuarios es que, en el menú principal de la aplicación móvil, pulsaban en las letras que indican la opción del menú, mientras que, para

realizarlo de forma correcta, hay que pulsar en el botón con dibujo que hay a la izquierda de las letras.

Con relación al nivel de informática y del uso de aplicaciones web y móvil no se observaron diferencias significativas entre los distintos usuarios.

A partir de estos resultados se pueden llegar a hacer algunas acciones correctoras, entre ellas:

- Añadir enlace en las letras del menú para no tener que hacer click en el botón.
- Añadir un enlace para ir al registro en el formulario de inicio de sesión de la web de forma similar a la aplicación móvil.
- Buscar alguna forma de mostrar todas las opciones de un simple golpe de vista al hacer mesa para no tener que hacer scroll.

La puntuación media que los usuarios otorgaron al sistema es de 8,5 sobre 10.

6.3.4 Pruebas de aceptación

Los resultados de las pruebas de aceptación son favorables, todos los casos de prueba diseñados pasan.

Capítulo 7. Manuales

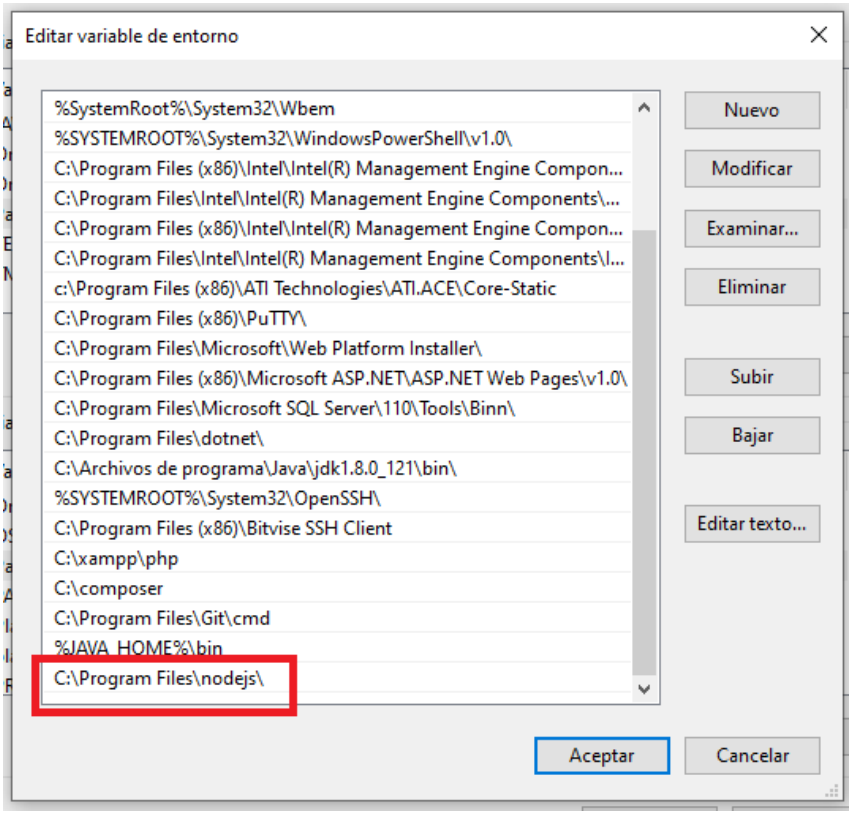
7.1 Introducción

A continuación, se muestran la creación de los manuales de despliegue en el que se especifican los pasos necesarios para el despliegue de la parte de la aplicación web, de instalación en el que se especifican los pasos necesarios para instalar el sistema en el dispositivo móvil, y de usuario que sirve de ayuda a los usuarios para saber cómo funciona el sistema.

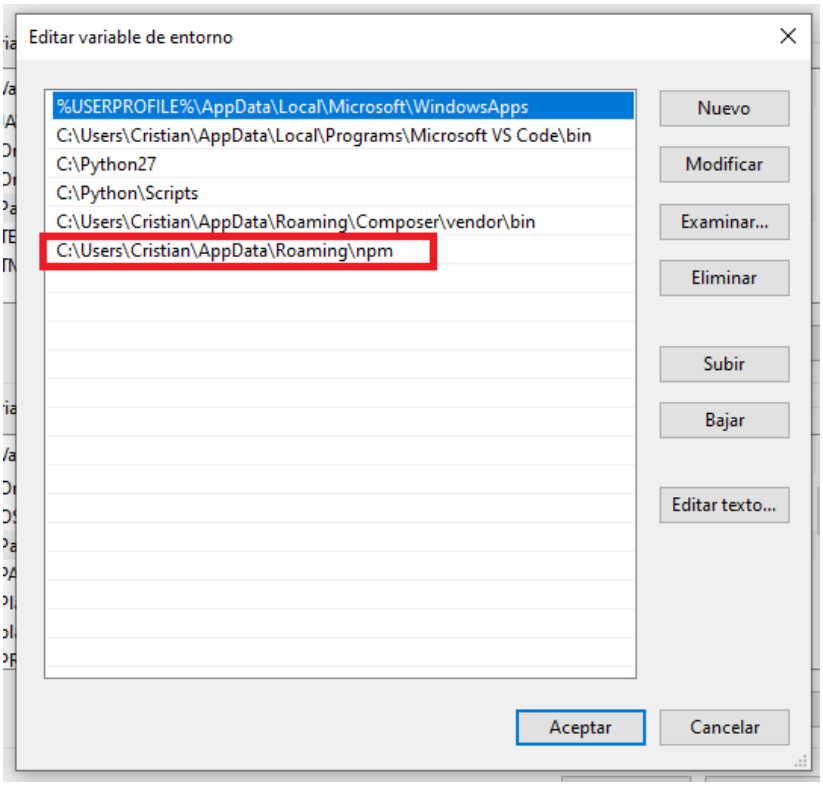
7.2 Manual de despliegue

Para el despliegue de la aplicación web y del servicio REST propio en conjunto, se necesita la instalación de Node js. Para instalar este entorno es necesario descargarlo de la web oficial https://nodejs.org/es/download/ y, una vez descargado, es necesario instalarlo.

Cuando ya está instalado se debe asegurar que la ruta en la que se encuentra está en las variables de entorno del sistema para, así, no tener que copiar toda la ruta cuando se quiera ejecutar.

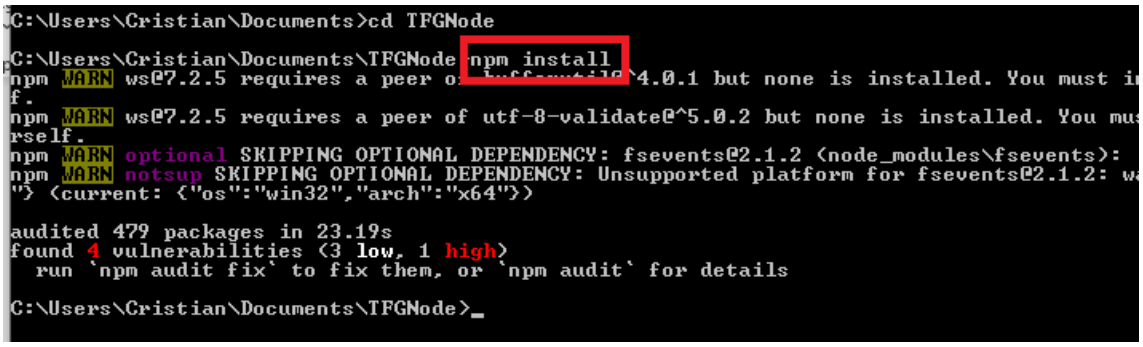


Otro aspecto que puede ser útil es añadir también a las variables de entorno del sistema la gestión de paquetes de Node, npm. Este sistema permite la instalación de todos los módulos necesarios para ejecutar la aplicación.



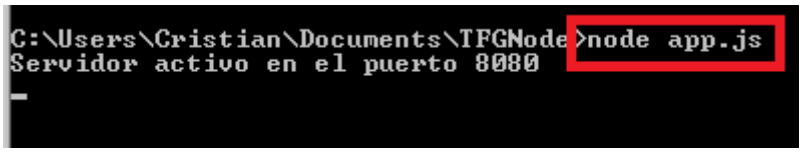
Una vez está el entorno de Node js preparado se procede al despliegue de la aplicación. Para ello se utiliza la consola de comandos de Windows. Se sitúa la ruta actual en la ubicación del

fichero de dependencias “package.json” y se instalan todos los módulos necesarios para la ejecución de la aplicación mediante el comando “npm install”.

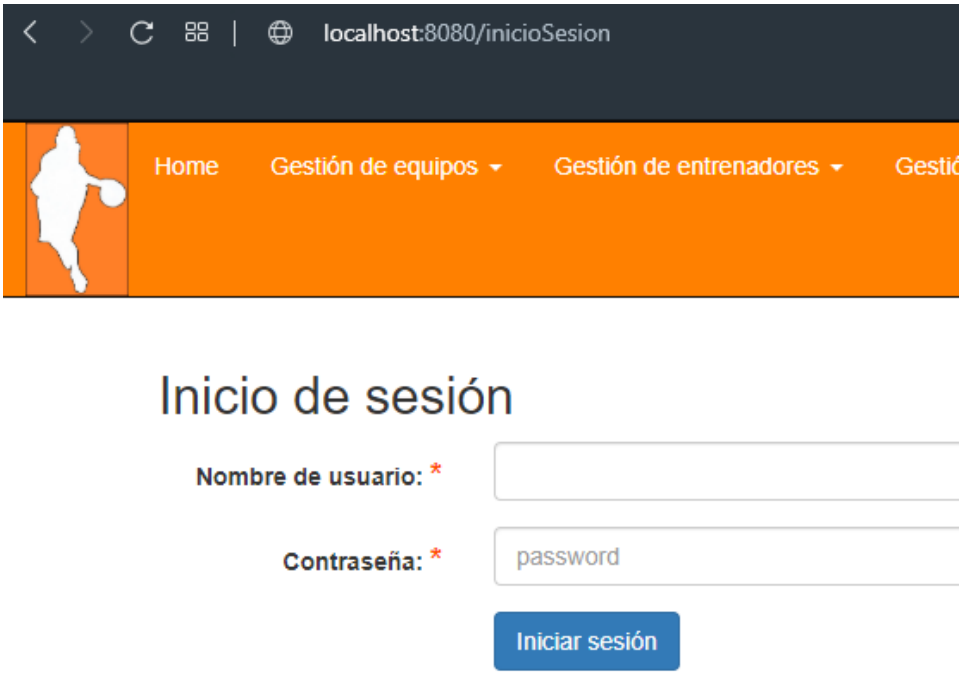


Por motivos de seguridad se elimina el archivo que contiene el script de rellenado de datos que se encuentra en routes/test/fillDB.js. Este archivo solo se utiliza para las pruebas. En el caso de querer ejecutar las pruebas realizadas o añadir más, lo más adecuado sería realizar una copia de seguridad de la base de datos actual con los datos del sistema en producción, realizar las pruebas con el script de rellenado de datos y volver a restaurar la copia de seguridad.

Para la ejecución del sistema se sitúa la ruta en el núcleo de la aplicación que, en este caso, es el fichero “app.js”, y se ejecuta el comando “node app.js”.



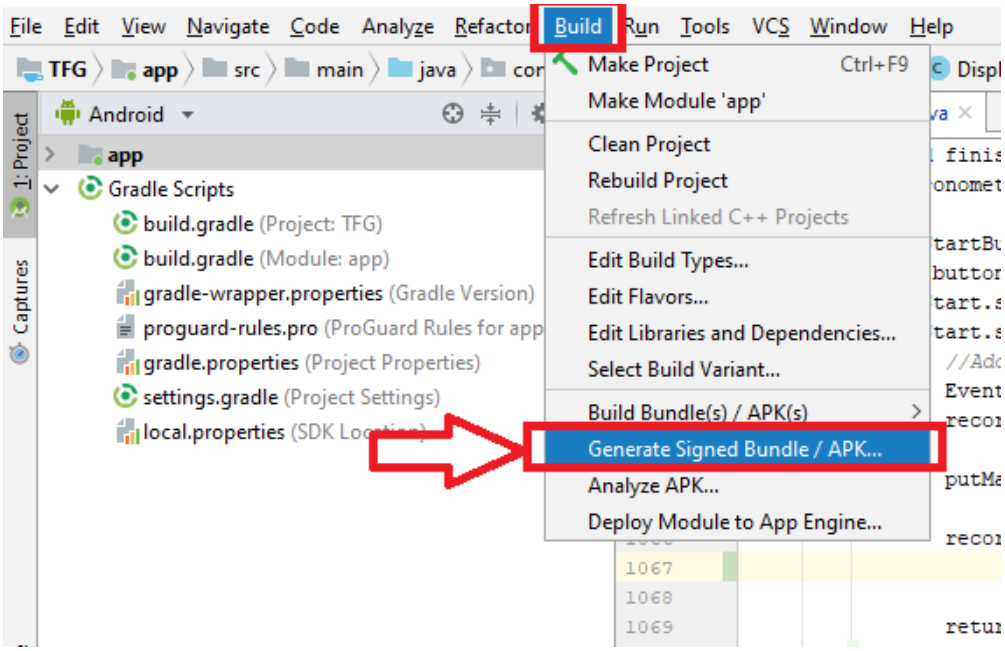
La línea de comandos muestra un mensaje de “Servidor activo en el puerto 8080” por lo que si se accede a la dirección ip mediante el puerto indicado 8080 de la máquina en la que se está ejecutando la aplicación se puede visualizar.



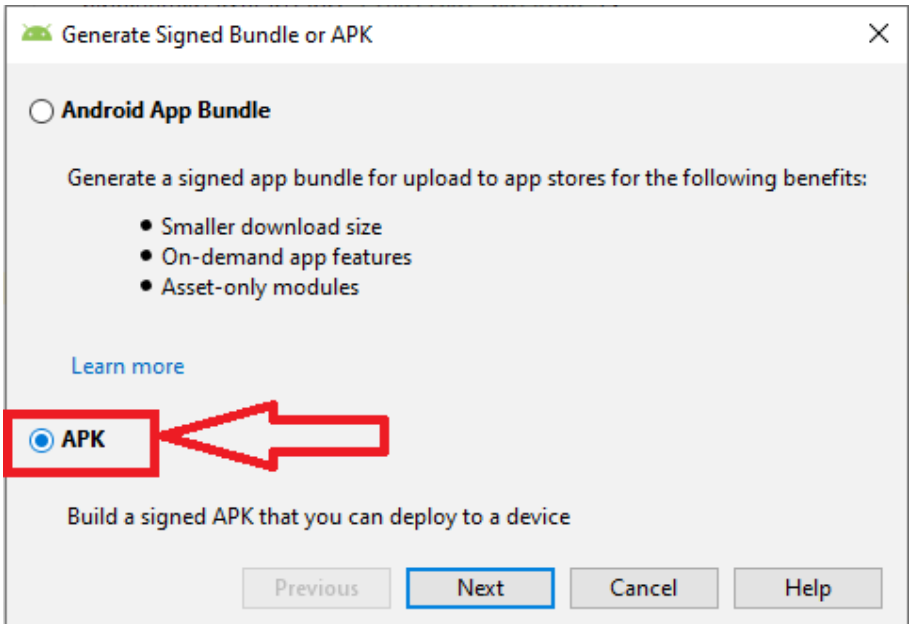
7.3 Manual de instalación

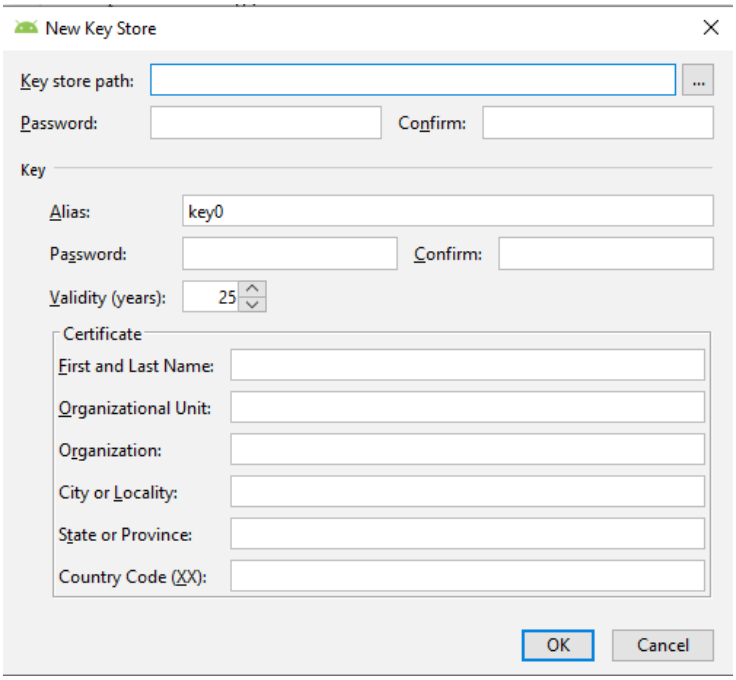
Este manual se centra en la aplicación móvil. Su instalación es bastante sencilla ya que solo se necesita descargar la apk generada desde Android Studio en el dispositivo móvil e instalarla como se hace normalmente con las demás aplicaciones móviles.

Lo primero que se debe hacer es generar el archivo apk que, en este caso, estará firmado digitalmente ya que si, en un futuro, se quiere subir a Google Play esta es una condición necesaria. Para ello, hay que elegir la opción "Generate Signed Bundle/APK…" dentro de la opción "Build" del menú superior de Android Studio.



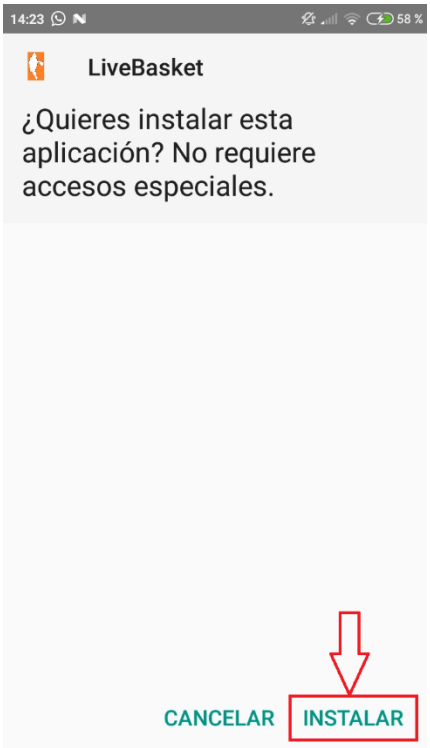
Una vez elegida esta opción se elige generar APK en el diálogo que se muestra y en el siguiente diálogo se crea una nueva clave o firma.





Cuando ya está creada la nueva firma se continúa con los demás pasos que se indican hasta generar el fichero apk con la aplicación.

El fichero apk generado se copia en el almacenamiento del dispositivo móvil y a partir de ese momento ya está disponible la aplicación para instalarse.



7.4 Manual de usuario

El sistema construido es un sistema multiusuario de seguimiento de partidos de baloncesto online. Consta de dos partes diferenciadas, una aplicación web y una aplicación móvil. En este

manual se explicará cómo funcionan ambas partes y lo que se puede llegar a realizar a través del sistema.

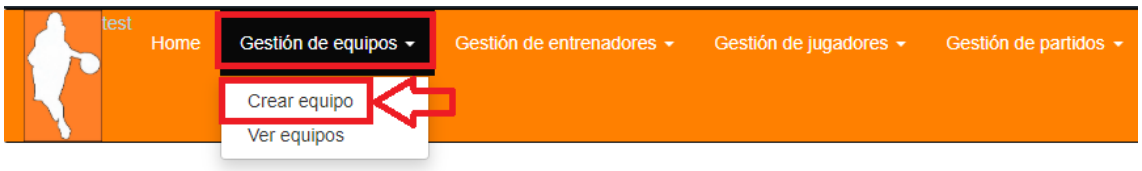
Para acceder a la aplicación web es necesario estar registrado en el sistema. Si el usuario no lo está deberá acceder al formulario de registro a través del enlace que existe para ello en la barra de menú de la cabecera.

Una vez en el formulario de registro se debe introducir el nombre de usuario, la contraseña y la repetición de la contraseña. El nombre de usuario no podrá coincidir con alguno ya registrado en el sistema.

Si el usuario ya está registrado no debe realizar este paso y debe introducir directamente en el formulario de inicio de sesión su nombre de usuario y la contraseña.

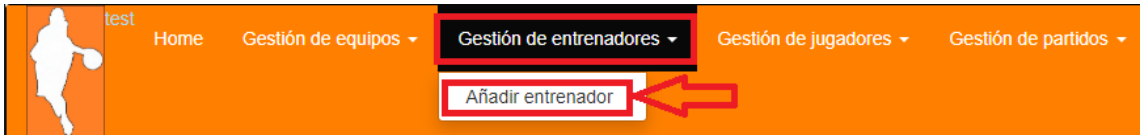
Cuando el usuario se ha identificado bien mediante el registro o bien mediante inicio de sesión el sistema muestra una tabla con los partidos creados por el usuario. Si hay algún partido sin activar existe la opción de activarlo o modificarlo a través de unos botones que aparecen en la misma fila que los datos del partido. Una vez activado un partido ya no se puede modificar.

Para crear un equipo se necesita acceder al enlace de la barra de navegación superior de “Gestión de equipos”. Al hacer click se despliega un submenú con la opción de crear equipo.



Se rellena el formulario de creación del equipo con el nombre del equipo y la cancha donde juega como local y se hace click en el botón “Crear equipo”. Una vez hecho esto el equipo se crea y aparece una nueva pantalla con la opción de añadir entrenador.

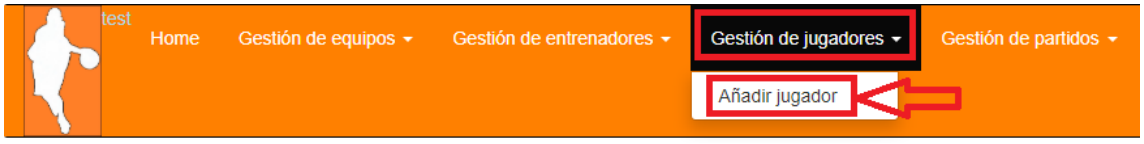
Para añadir un entrenador a un equipo se puede acceder de dos maneras. Una de ellas es creando un equipo como se ha mencionado anteriormente y la otra es accediendo a través de la "Gestión de entrenadores" de la barra de navegación superior.



Se rellena el formulario con el nombre del entrenador y se escoge el equipo de entre los disponibles en el sistema. Si se accede desde la creación del equipo, el equipo seleccionado por defecto es el que se había creado. Una vez rellenos los datos se añade el entrenador mediante el botón "Añadir entrenador". Un equipo puede tener todos los entrenadores que se quiera en este sistema.

Cuando se añade el entrenador el sistema redirige a la introducción de jugadores en el equipo. Si el equipo tiene unos dorsales fijos para cada jugador se pueden añadir desde esta pantalla. Al igual que el número de entrenadores, el número de jugadores que puede tener un equipo no está limitado.

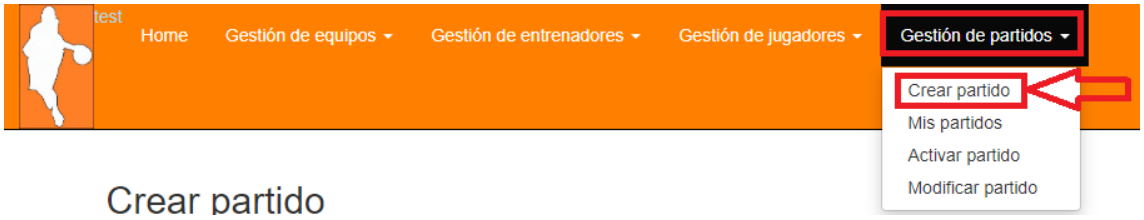
También se pueden añadir jugadores desde el menú superior a través de la "Gestión de jugadores".



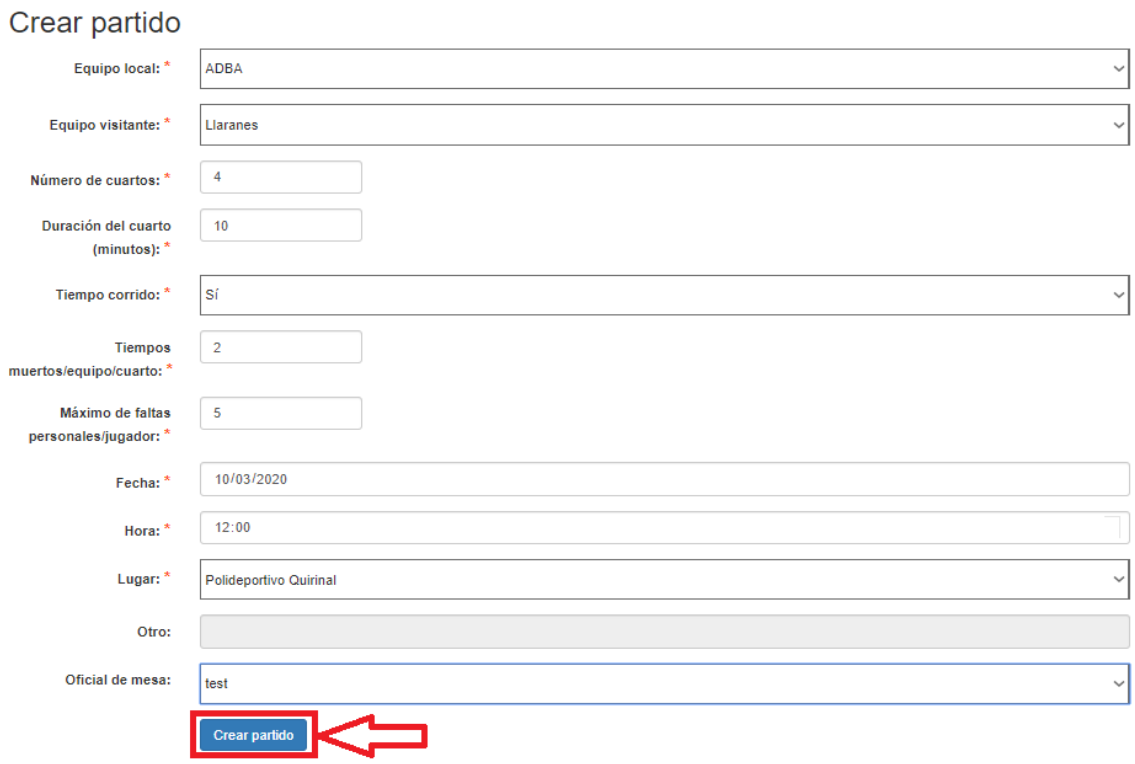
Para añadir a los jugadores se introduce el nombre, el dorsal y se escoge el equipo de entre los disponibles en el sistema. Una vez introducidos los datos se añade mediante el botón "Añadir jugador".

Un equipo creado puede ser eliminado por cualquier usuario mientras el equipo no esté participando en ningún partido creado, activo, en juego o finalizado. Para ello se accede a la lista de equipos a través de la "Gestión de equipos" y eligiendo la opción de "Ver equipos". Cada equipo tiene un botón de eliminar que al pulsar elimina al equipo si no está como participante en ningún partido o muestra un mensaje de error si sí lo está haciendo.

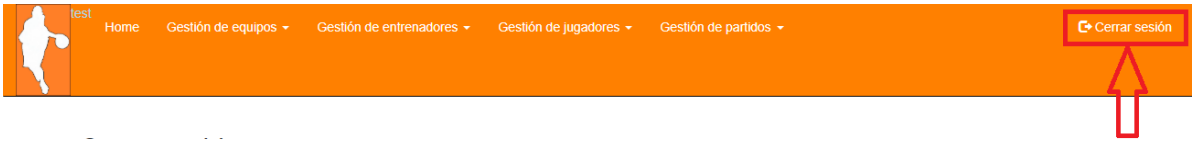
Mediante la aplicación web también es posible crear partidos. Para ello se accede a través de la "Gestión de partidos".



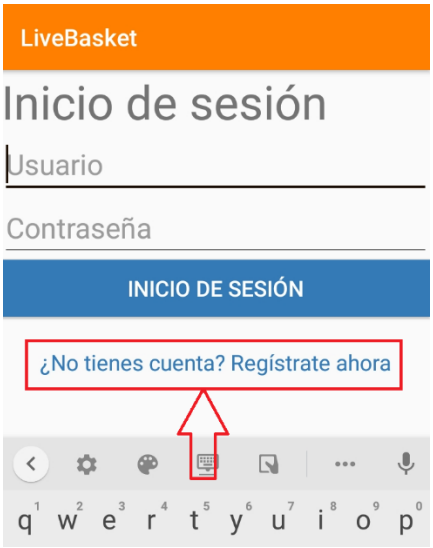
Para crear un partido se deben introducir los datos en el formulario. Estos datos se muestran en la siguiente figura. Una vez introducidos se crea el partido a través del botón "Crear partido". En el campo para elegir "Lugar" existe la posibilidad de elegir "Otro" y escribir el nombre del lugar en el campo "Otro" ya que se puede jugar un partido en un lugar diferente a las canchas de los equipos introducidas.



Por último, para cerrar sesión en la aplicación web se debe hacer click en la opción del menú superior “Cerrar sesión”.



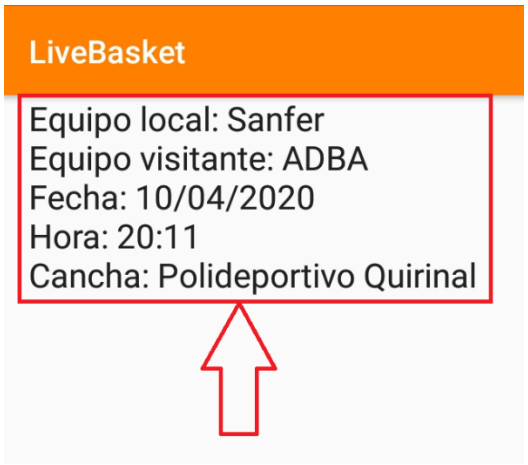
Para acceder a la aplicación móvil también es necesario estar registrado en el sistema. Si el usuario no tiene cuenta existe la posibilidad de registrarse desde el móvil rellenando el nombre de usuario, la contraseña y la repetición de la contraseña.



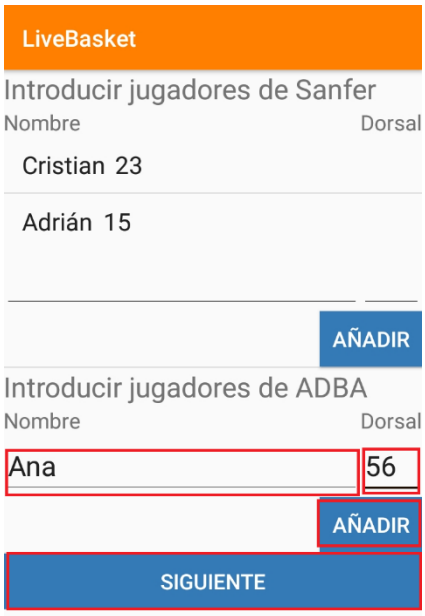
Si el usuario ya está registrado solo tiene que iniciar sesión con su nombre de usuario y su contraseña.

Cuando el usuario inicia sesión en la aplicación móvil se muestra un menú principal con tres opciones: hacer mesa, ver partido, partidos seguidos.

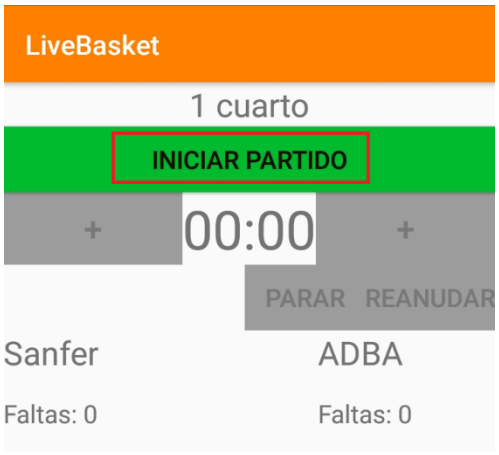
La primera de ellas sirve para introducir las acciones que vayan sucediendo en un partido. Cuando se elige esta opción se muestra una lista con los partidos activos que no tienen oficial de mesa asignado o cuyo oficial de mesa es el usuario identificado. En esta lista se escoge el partido del cual se quiere hacer mesa.



Una vez elegido el partido se deben completar los jugadores que participarán en él por parte de ambos equipos. Se muestran los jugadores ya introducidos en los equipos mediante la aplicación web que tienen un dorsal fijo y se da la opción de introducir más con el nombre y el dorsal. Debajo del hueco para añadir a los jugadores de cada equipo hay un botón “Añadir” que los añade al equipo correspondiente. Cuando ya se han completado los equipos se accede al partido mediante el botón “Siguiente”.



Para iniciar el partido se hace click en el botón “Iniciar partido” y el tiempo empieza a correr hacia atrás mostrando los minutos y segundos que quedan para el final del cuarto.

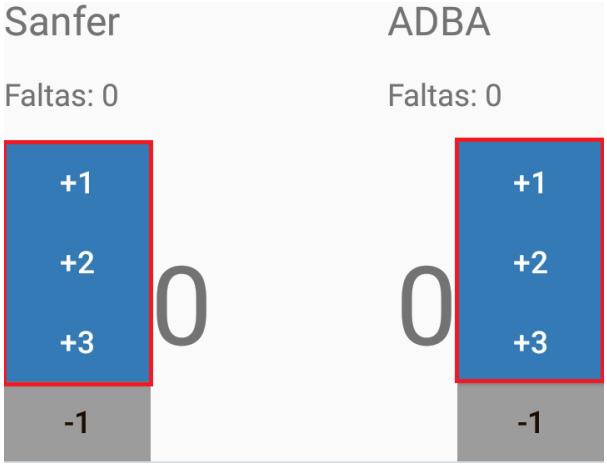


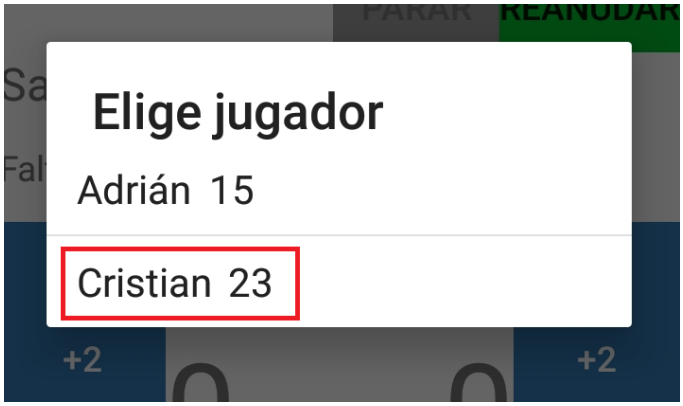
Cuando se inicia el partido ya se pueden introducir las acciones que vayan sucediendo. El tiempo se puede parar de forma voluntaria sin que haya ocurrido ninguna acción y se puede volver a reanudar si el partido no es a tiempo corrido.

Cuando el tiempo está parado se pueden añadir minutos o segundos. El botón "+" de la izquierda sirve para aumentar los minutos y el botón "+" de la derecha sirve para aumentar los segundos.

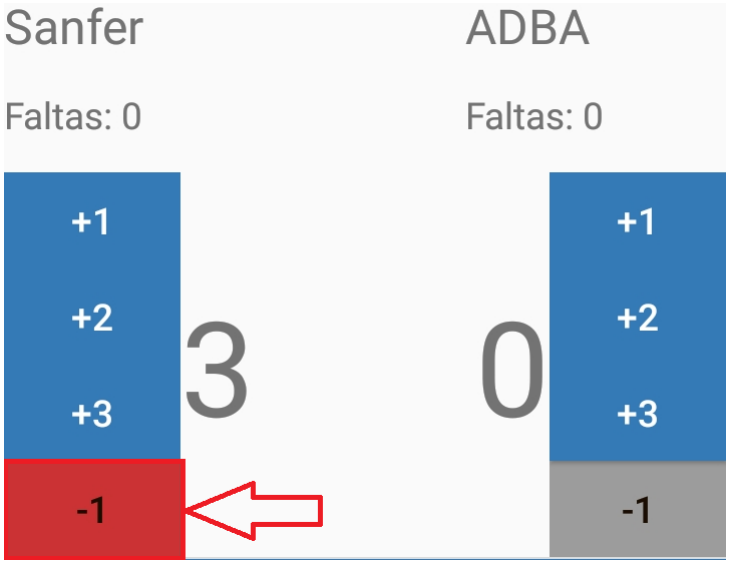
Al indicar cualquier otra acción el tiempo se para y se tendrá que reanudar para que siga corriendo si el partido no es a tiempo corrido.

Los puntos se añaden en los botones "+1", "+2" y "+3". Si los puntos corresponden al equipo local se utilizarán los botones de la izquierda, si, por el contrario, corresponden al equipo visitante se utilizarán los de la derecha. Cuando se pulsa uno de estos botones se debe elegir el jugador que ha anotado. El marcador se irá actualizando al lado de los botones.





Para quitar un punto a un equipo y jugador se utiliza el botón "-1". Al igual que para añadir puntos, el de la izquierda está destinado al equipo local y el de la derecha al equipo visitante.

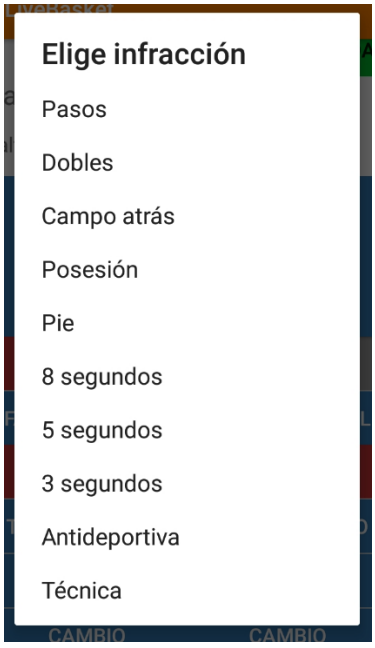


Para añadir faltas personales se utiliza el botón “Añadir falta personal” de un lado u otro dependiendo del equipo y se debe elegir el jugador que la ha cometido.

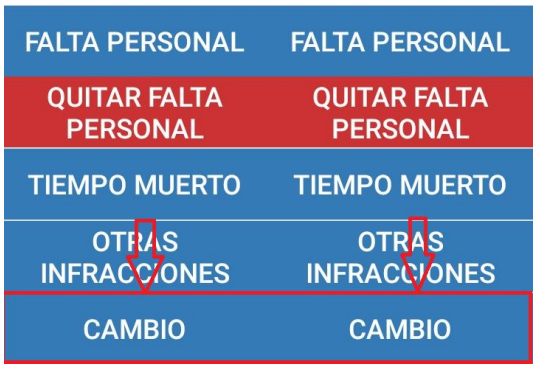
Para quitar una falta personal a un jugador se utiliza el botón “Quitar falta personal” de la misma manera que para añadirla.

Los tiempos muertos de cada equipo se añaden con los botones de “Tiempo muerto”. Cuando el equipo agota sus tiempos muertos en un cuarto el botón se deshabilita.

En el botón de “Otras infracciones” se puede elegir entre las siguientes: “Pasos”, “Dobles”, “Campo atrás”, “Posesión”, “Pie”, “8 segundos”, “5 segundos”, “3 segundos”, “Antideportiva” o “Técnica”. Una vez elegida una de estas opciones, dependiendo de si se asigna a un jugador o no se elige al infractor. En el caso de la técnica también se puede elegir a un entrenador para anotársela.

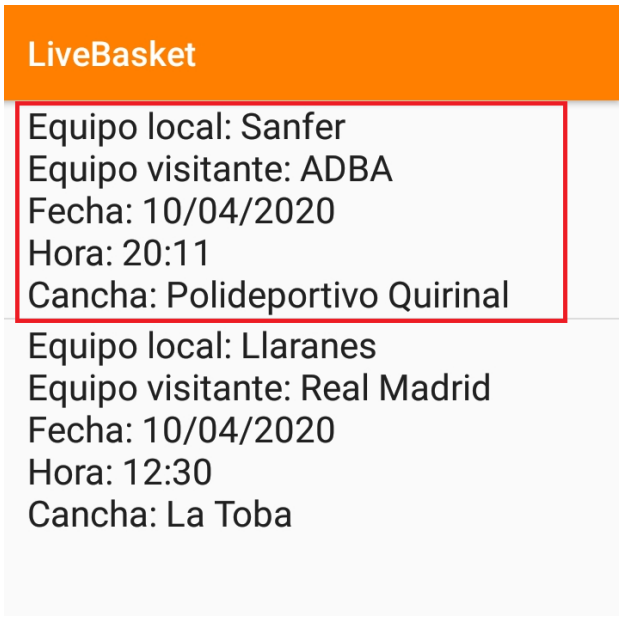


En el caso de los cambios se utiliza el botón de “Cambio” y después se debe elegir en primer lugar el jugador que va a ser sustituido y después a su sustituto.

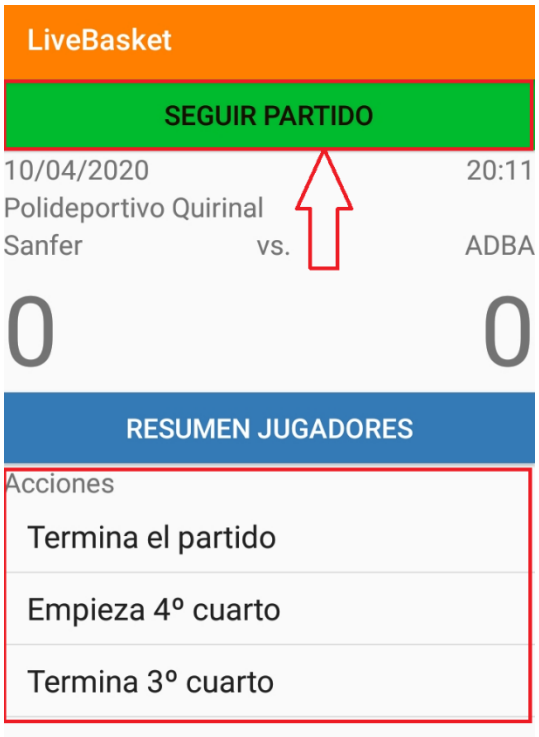


Cuando se acaba el tiempo se puede finalizar el cuarto si hay más cuartos o finalizar el partido si ya era el último cuarto. Si se finaliza un cuarto aparece la opción de empezar el siguiente.

La segunda opción del menú principal "Ver partido" sirve para poder ver las acciones que acontecen en un partido en tiempo real. Cuando se elige esta opción aparecen todos los partidos activos, en juego y finalizados.

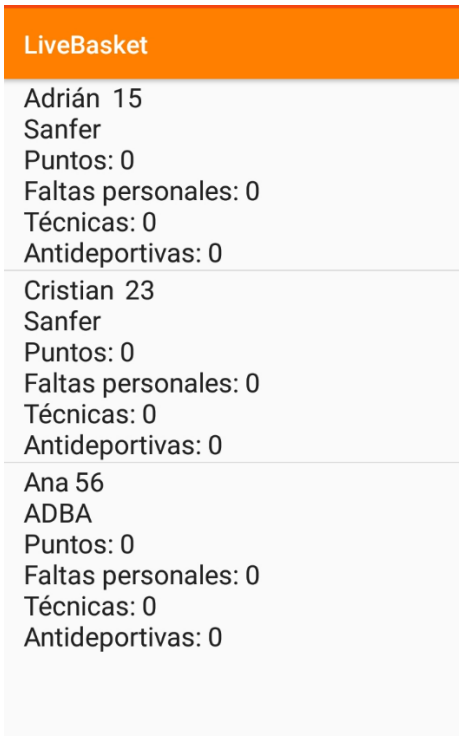


Al elegir un partido se muestran las acciones que acontecieron en el partido y aparece la opción de "Seguir partido". Una vez se empieza a seguir el partido se muestra en la lista de la opción del menú "Partidos seguidos".



Al empezar a seguir un partido aparece la opción “Dejar de seguir” para que ya no aparezca en los partidos seguidos por el usuario. También existe la opción ver el resumen de los jugadores del partido una vez el partido finaliza.

En el resumen de los jugadores se muestra cada jugador con su dorsal, el equipo, los puntos anotados en el partido, las faltas personales, las antideportivas y las técnicas cometidas.


LiveBasket
Adrián 15
Sanfer
Puntos: 0
Faltas personales: 0
Técnicas: 0
Antideportivas: 0
Cristian 23
Sanfer
Puntos: 0
Faltas personales: 0
Técnicas: 0
Antideportivas: 0
Ana 56
ADBA
Puntos: 0
Faltas personales: 0
Técnicas: 0
Antideportivas: 0

Capítulo 8. Referencias bibliográficas

[1] "Programación móvil: Qué herramienta y lenguaje elegir". Disponible en https://www.campusmvp.es/recursos/post/Programacion-movil-Que-herramienta-y-lenguaje-elegir.aspx [Consultado el 30 de enero de 2020]
[2] "Así es como Android se ha comido el mercado en diez años". Disponible en https://www.xatakamovil.com/sistemas-operativos/asi-como-android-se-ha-comido-mercado-diez-anos [Consultado el 30 de enero de 2020]
[3] "Node.js vs. Spring Boot – Which should you choose?". Disponible en https://medium.com/better-programming/node-js-vs-spring-boot-which-should-you-choose-2366c2f76587 [Consultado el 31 de enero de 2020]
[4] "Node.js vs Spring Boot". Disponible en https://stackshare.io/stackups/nodejs-vs-spring-boot [Consultado el 31 de enero de 2020]
[5] "¿Qué es Xamarin?". Disponible en https://docs.microsoft.com/es-es/xamarin/get-started/what-is-xamarin [Consultado el 16 de febrero de 2020]
[6] "PhoneGap". Disponible en https://es.wikipedia.org/wiki/PhoneGap [Consultado el 16 de febrero de 2020]
[7] "Aspectos básicos de Spring Boot". Disponible en https://www.ibm.com/developerworks/ssa/library/j-spring-boot-basics-perry/index.html [Consultado el 17 de febrero de 2020]
[8] "Acerca de Node.js". Disponible en https://nodejs.org/es/about/ [Consultado el 17 de febrero de 2020]
[9] "Socket.io 2.0 is here". Disponible en https://socket.io [Consultado el 17 de febrero de 2020]
[10] "Descarga Android Studio". Disponible en https://developer.android.com/studio#downloads [Consultado el 29 de febrero de 2020]
[11] "Descarga WebStorm". Disponible en https://www.jetbrains.com/webstorm/download/#section=windows [Consultado el 1 de marzo de 2020]
[12] "Tanteo". Disponible en http://www.tanteoapp.com/es/ [Consultado el 5 de marzo de 2020]
[13] "¿Qué cuesta contratar a un trabajador?". Disponible en https://www.billin.net/blog/cuesta-contratar-trabajador/ [Consultado el 6 de mayo de 2020]
[14] "Buscar y comparar sueldos". Disponible en https://www.indeed.es/salaries [Consultado el 6 de mayo de 2020]
[15] "Creating a realtime chat app with android, NodeJs and Socket.io". Disponible en https://dev.to/medaymentn/creating-a-realtime-chat-app-with-android--nodejs-and-socketio-4o55 [Consultado el 26 de abril de 2020]
[16] "Testeo de API REST con Mocha y Chai-HTTP". Disponible en https://www.paradigmadigital.com/dev/testeo-api-rest-mocha-chai-http/ [Consultado el 12 de mayo de 2020]
[17] "Mocha, simple, flexible, fun". Disponible en https://mochajs.org [Consultado el 23 de mayo de 2020]
[18] "JUnit". Disponible en https://junit.org/junit4/ [Consultado el 23 de mayo de 2020]
[19] "Gatling". Disponible en https://gatling.io [Consultado el 23 de mayo de 2020]
[20] "'Vamos a automatizar pruebas'. ¿Qué significa esto? ¿Realmente por dónde deberíamos empezar a automatizar?". Disponible en https://www.javiergarzas.com/2015/01/automatizacion-pruebas.html [Consultado el 23 de mayo de 2020]
[21] "Draw.io". Disponible en https://app.diagrams.net [Consultado el 29 de mayo de 2020]

[22] “mongodb 3.x driver Android compatibility”. Disponible en https://stackoverflow.com/questions/32529484/mongodb-3-x-driver-android-compatibility/32531278#32531278 [Consultado el 30 de junio de 2020]

Apéndices

Trabajos no productivos

Trabajos no productivos					
Personal	TOTAL Proyecto	Prod (%)	Coste Directo	CI (%)	Coste Indirecto
Ingeniero de software	7.186,75 €	96,00%	6.899,28 €	4,00%	287,47 €
Consultor de tecnología	7.015,68 €	5,94%	416,56 €	94,06%	6.599,13 €
Arquitecto de software	8.622,55 €	60,63%	5.227,42 €	39,38%	3.395,13 €
Analista funcional	7.109,43 €	16,88%	1.199,72 €	83,13%	5.909,71 €
Programador	5.839,00 €	70,94%	4.142,04 €	29,06%	1.696,96 €
TOTAL	35.773,41 €		17.885,01 €		17.888,40 €

Tabla 36. Trabajos no productivos

Horas de trabajo y facturación

Horas de trabajo y facturación					
Personal	Prod (%)	Horas / proyecto	Horas productivas / proyecto / perfil	Precio / hora	Facturación
Ingeniero de software	96,00%	320	307	63,80 €	19.599,36 €
Consultor de tecnología	5,94%	320	19	55,10 €	1.046,90 €
Arquitecto de software	60,63%	320	194	66,60 €	12.920,40 €
Analista funcional	16,88%	320	54	55,00 €	2.970,00 €
Programador	70,94%	320	227	45,00 €	10.215,00 €
TOTAL					46.751,66 €

Tabla 37. Horas de trabajo y facturación

Desglose del presupuesto de costes

En este anexo se muestran las tablas con las que se ha calculado el coste total de cada una de las partidas del presupuesto.

En la parte del precio se han utilizado los precios/hora sin beneficios como se ha mencionado en el Capítulo 7 de Presupuesto.

Partida Tareas iniciales

Descripción	Cantidad	Unidades	Precio	Subtotal (5)	Subtotal (4)	Subtotal (3)	Subtotal (2)	Subtotal (1)
Tareas iniciales								1.183,50 €
Definir necesidades del usuario							41,25 €	
Analista funcional	1	horas	41,25 €			41,25 €		
Determinar ámbito y alcance							82,50 €	

Descripción	Cantidad	Unidades	Precio	Subtotal (5)	Subtotal (4)	Subtotal (3)	Subtotal (2)	Subtotal (1)
Analista funcional	2	horas	41,25 €			82,50 €		
Obtener requisitos de usuario de alto nivel							123,75 €	
Definir EPICs						123,75 €		
Analista funcional	3	horas	41,25 €		123,75 €			
Estudio de viabilidad del sistema							454,58 €	
Análisis de alternativas ya existentes						247,95 €		
Estudio de las alternativas					206,63 €			
Consultor de tecnología	5	horas	41,33 €	206,63 €				
Elección de la alternativa final					41,33 €			
Consultor de tecnología	1	horas	41,33 €	41,33 €				
Análisis de tecnologías						206,63 €		
Estudio de las tecnologías					82,65 €			
Consultor de tecnología	2	horas	41,33 €	82,65 €				
Elección de las tecnologías					123,98 €			
Consultor de tecnología	3	horas	41,33 €	123,98 €				
Planteamiento de la arquitectura							91,28 €	
Arquitecto del software	1	horas	49,95 €			49,95 €		
Consultor de tecnología	1	horas	41,33 €			41,33 €		
Preparación de entornos de trabajo							33,75 €	
Programador	1	horas	33,75 €			33,75 €		
Crear backlog							356,40 €	
Ingeniero de software	4	horas	47,85 €			191,40 €		
Anlaista funcional	4	horas	41,25 €			165,00 €		

Tabla 38. Presupuesto de costes de la partida Tareas iniciales

Partida Sprint 1

Descripción	Cantidad	Unidades	Precio	Subtotal (5)	Subtotal (4)	Subtotal (3)	Subtotal (2)	Subtotal (1)
Sprint 1								10.547,33 €
Análisis							50,63 €	
Escoger HU del sprint						33,75 €		
Programador	1	horas	33,75 €		33,75 €			
Estimar HU						16,88 €		
Programador	0,5	horas	33,75 €		16,88 €			
Desarrollo							1.350,00 €	
Programador	40	horas	33,75 €			1.350,00 €		
Integración							5.601,60 €	
Integración de componentes						3.348,00 €		
Arquitecto del software	40	horas	49,95 €		1.998,00 €			
Programador	40	horas	33,75 €		1.350,00 €			
Pruebas de integración						2.253,60 €		
Diseño					1.173,60 €			
Arquitecto del software	12	horas	49,95 €	599,40 €				
Ingeniero de software	12	horas	47,85 €	574,20 €				
Ejecución					1.080,00 €			
Programador	32	horas	33,75 €	1.080,00 €				
Pruebas unitarias							1.856,70 €	
Pruebas funcionales						1.856,70 €		
Diseño					574,20 €			
Ingeniero de software	12	horas	47,85 €	574,20 €				
Ejecución					1.282,50 €			
Programador	38	horas	33,75 €	1.282,50 €				
Pruebas de sistema							1.688,40 €	
Pruebas funcionales						844,20 €		

Descripción	Cantidad	Unidades	Precio	Subtotal (5)	Subtotal (4)	Subtotal (3)	Subtotal (2)	Subtotal (1)
Diseño					574,20 €			
Ingeniero de software	12	horas	47,85 €	574,20 €				
Ejecución					270,00 €			
Programador	8	horas	33,75 €	270,00 €				
Pruebas de seguridad						844,20 €		
Diseño					574,20 €			
Ingeniero de software	12	horas	47,85 €	574,20 €				
Ejecución					270,00 €			
Programador	8	horas	33,75 €	270,00 €				

Tabla 39. Presupuesto de costes de la partida Sprint 1

Partida Sprint 2

Descripción	Cantidad	Unidades	Precio	Subtotal (5)	Subtotal (4)	Subtotal (3)	Subtotal (2)	Subtotal (1)
Sprint 2								10.547,33 €
Análisis							50,63 €	
Escoger HU del sprint						33,75 €		
Programador	1	horas	33,75 €		33,75 €			
Estimar HU						16,88 €		
Programador	0,5	horas	33,75 €		16,88 €			
Desarrollo							1.350,00 €	
Programador	40	horas	33,75 €			1.350,00 €		
Integración							5.601,60 €	
Integración de componentes						3.348,00 €		
Arquitecto del software	40	horas	49,95 €		1.998,00 €			
Programador	40	horas	33,75 €		1.350,00 €			
Pruebas de integración						2.253,60 €		
Diseño					1.173,60 €			
Arquitecto del software	12	horas	49,95 €	599,40 €				
Ingeniero de software	12	horas	47,85 €	574,20 €				
Ejecución					1.080,00 €			
Programador	32	horas	33,75 €	1.080,0 €				

Descripción	Cantidad	Unidades	Precio	Subtotal (5)	Subtotal (4)	Subtotal (3)	Subtotal (2)	Subtotal (1)
Pruebas unitarias							1.856,70 €	
Pruebas funcionales						1.856,70 €		
Diseño					574,20 €			
Ingeniero de software	12	horas	47,85 €	574,20 €				
Ejecución					1.282,50 €			
Programador	38	horas	33,75 €	1.282,50 €				
Pruebas de sistema							1.688,40 €	
Pruebas funcionales						844,20 €		
Diseño					574,20 €			
Ingeniero de software	12	horas	47,85 €	574,20 €				
Ejecución					270,00 €			
Programador	8	horas	33,75 €	270,00 €				
Pruebas de seguridad						844,20 €		
Diseño					574,20 €			
Ingeniero de software	12	horas	47,85 €	574,20 €				
Ejecución					270,00 €			
Programador	8	horas	33,75 €	270,00 €				

Tabla 40. Presupuesto de costes de la partida Sprint 2

Partida Sprint 3

Descripción	Cantidad	Unidades	Precio	Subtotal (5)	Subtotal (4)	Subtotal (3)	Subtotal (2)	Subtotal (1)
Sprint 3								10.547,33 €
Análisis							50,63 €	
Escoger HU del sprint						33,75 €		
Programador	1	horas	33,75 €		33,75 €			
Estimar HU						16,88 €		
Programador	0,5	horas	33,75 €		16,88 €			
Desarrollo							1.350,00 €	
Programador	40	horas	33,75 €			1.350,00 €		
Integración							5.601,60 €	
Integración de componentes						3.348,00 €		

Descripción	Cantidad	Unidades	Precio	Subtotal (5)	Subtotal (4)	Subtotal (3)	Subtotal (2)	Subtotal (1)
Arquitecto del software	40	horas	49,95 €		1.998,00 €			
Programador	40	horas	33,75 €		1.350,00 €			
Pruebas de integración						2.253,60 €		
Diseño					1.173,60 €			
Arquitecto del software	12	horas	49,95 €	599,40 €				
Ingeniero de software	12	horas	47,85 €	574,20 €				
Ejecución					1.080,00 €			
Programador	32	horas	33,75 €	1.080,00 €				
Pruebas unitarias							1.856,70 €	
Pruebas funcionales						1.856,70 €		
Diseño					574,20 €			
Ingeniero de software	12	horas	47,85 €	574,20 €				
Ejecución					1.282,50 €			
Programador	38	horas	33,75 €	1.282,50 €				
Pruebas de sistema							1.688,40 €	
Pruebas funcionales						844,20 €		
Diseño					574,20 €			
Ingeniero de software	12	horas	47,85 €	574,20 €				
Ejecución					270,00 €			
Programador	8	horas	33,75 €	270,00 €				
Pruebas de seguridad						844,20 €		
Diseño					574,20 €			
Ingeniero de software	12	horas	47,85 €	574,20 €				
Ejecución					270,00 €			
Programador	8	horas	33,75 €	270,00 €				

Tabla 41. Presupuesto de costes de la partida Sprint 3

Partida Pruebas finales

Descripción	Cantidad	Unidades	Precio	Subtotal (5)	Subtotal (4)	Subtotal (3)	Subtotal (2)	Subtotal (1)
Pruebas finales								3.635,40 €
Pruebas de sistema							1.497,00 €	
Pruebas de rendimiento y carga						461,40 €		
Diseño					191,40 €			
Ingeniero de software	4	horas	47,85 €	191,40 €				
Ejecución					270,00 €			
Programador	8	horas	33,75 €	270,00 €				
Pruebas de usabilidad						1.035,60 €		
Diseño					382,80 €			
Ingeniero de software	8	horas	47,85 €	382,80 €				
Ejecución					652,80 €			
Ingeniero de software	8	horas	47,85 €	382,80 €				
Programador	8	horas	33,75 €	270,00 €				
Pruebas de aceptación							2.138,40 €	
Analista funcional	24	horas	41,25 €			990,00 €		
Ingeniero de software	24	horas	47,85 €			1.148,40 €		

Tabla 42. Presupuesto de costes de la partida Pruebas finales

Resumen del presupuesto de costes

Cod.	Partida	Total
01	Tareas iniciales	1.183,50 €
02	Sprint 1	10.547,33 €
03	Sprint 2	10.547,33 €
04	Sprint 3	10.547,33 €
05	Pruebas finales	3.635,40 €

TOTAL	36.460,88 €

Tabla 43. Resumen del presupuesto de costes

Desglose del presupuesto del cliente

Partida	Item	Partida	Importe	Total
01		Tareas iniciales		1.479,38 €
	01	Definir necesidades del usuario	51,56 €	
	02	Definir ámbito y alcance	103,13 €	
	03	Obtener requisitos de usuario de alto nivel	154,69 €	
	04	Estudio de viabilidad del sistema	568,22 €	
	05	Planteamiento de la arquitectura	114,09 €	
	06	Preparación de entornos de trabajo	42,19 €	
	07	Crear backlog	445,50 €	
02		Sprint 1		13.184,16 €
	01	Análisis	63,28 €	
	02	Desarrollo	1.687,50 €	
	03	Integración	7.002,00 €	
	04	Pruebas unitarias	2.320,88 €	
	05	Pruebas de sistema	2.110,50 €	
03		Sprint 2		13.184,16 €
	01	Análisis	63,28 €	
	02	Desarrollo	1.687,50 €	
	03	Integración	7.002,00 €	
	04	Pruebas unitarias	2.320,88 €	
	05	Pruebas de sistema	2.110,50 €	
04		Sprint 3		13.184,16 €
	01	Análisis	63,28 €	
	02	Desarrollo	1.687,50 €	
	03	Integración	7.002,00 €	
	04	Pruebas unitarias	2.320,88 €	
	05	Pruebas de sistema	2.110,50 €	
05		Pruebas finales		4.544,25 €
	01	Pruebas de sistema	1.871,25 €	
	02	Pruebas de aceptación	2.673,00 €	

Total	45.576,09 €

Tabla 44. Desglose del presupuesto del cliente

Documentación del código

Module: modules/DBManager

Database module manager

Source: modules/DBManager.js, line 1

Methods

(static) deleteTeam(criterion, funcionCallback)

Delete a team by a given criteria

Parameters:

Name	Type	Description
criterion	JSON	for filtering
funcionCallback	function	

Source: modules/DBManager.js, line 232

Figura 59. Código del módulo DBManager (I)

(static) getCoaches(criterion, funcionCallback)

Get coaches by a given criteria

Parameters:

Name	Type	Description
criterion	JSON	for filtering
funcionCallback	function	

Source: modules/DBManager.js, line 140

(static) getEvents(criterion, funcionCallback)

Get events by a given criteria

Parameters:

Name	Type	Description
criterion	JSON	for filtering
funcionCallback	function	

Source: modules/DBManager.js, line 375

Figura 60. Código del módulo DBManager (II)

(static) getMatches(criterion, funcionCallback)

Get matches by a given criteria

Parameters:

Name	Type	Description
criterion	JSON	for filtering
funcionCallback	function	

Source: modules/DBManager.js, line 257

(static) getPlayers(criterion, funcionCallback)

Get players by a given criteria

Parameters:

Name	Type	Description
criterion	JSON	for filtering
funcionCallback	function	

Source: modules/DBManager.js, line 186

Figura 61. Código del módulo DBManager (III)

(static) getRecords(criterion, funcionCallback)

Get summary of a player from a match

Parameters:

Name	Type	Description
criterion	JSON	for filtering
funcionCallback	function	

Source: modules/DBManager.js, line 398

(static) getTeams(criterion, funcionCallback)

Get teams by a given criteria

Parameters:

Name	Type	Description
criterion	JSON	for filtering
funcionCallback	function	

Source: modules/DBManager.js, line 94

Figura 62. Código del módulo DBManager (IV)

(static) getUsers(criterion, funcionCallback)

Get users by a given criteria

Parameters:

Name	Type	Description
criterion	JSON	for filtering
funcionCallback	function	

Source: modules/DBManager.js, line 46

(static) insertCoach(coach, funcionCallback)

Insert a coach

Parameters:

Name	Type	Description
coach	JSON	to insert
funcionCallback	function	

Source: modules/DBManager.js, line 163

Figura 63. Código del módulo DBManager (V)

(static) insertEvent(event, funcionCallback)

Insert an event from a match

Parameters:

Name	Type	Description
event	JSON	to insert
funcionCallback	function	

Source: modules/DBManager.js, line 352

(static) insertMatch(match, funcionCallback)

Insert a match

Parameters:

Name	Type	Description
match	JSON	to insert
funcionCallback	function	

Source: modules/DBManager.js, line 280

Figura 64. Código del módulo DBManager (VI)

(static) insertPlayer(player, funcionCallback)

Insert a player

Parameters:

Name	Type	Description
player	JSON	to insert
funcionCallback	function	

Source: modules/DBManager.js, line 209

(static) insertTeam(team, funcionCallback)

Insert a team

Parameters:

Name	Type	Description
team	JSON	to insert
funcionCallback	function	

Source: modules/DBManager.js, line 117

Figura 65. Código del módulo DBManager (VII)

(static) insertUser(user, funcionCallback)

Insert an user

Parameters:

Name	Type	Description
user	JSON	to insert
funcionCallback	function	

Source: modules/DBManager.js, line 69

(static) modifyMatch(criterion, match, funcionCallback)

Modify a match

Parameters:

Name	Type	Description
criterion	JSON	to find the match
match	JSON	to modify
funcionCallback	function	

Source: modules/DBManager.js, line 304

Figura 66. Código del módulo DBManager (VIII)

(static) recordMatch(record, funcionCallback)

Insert summary of a player in a match

Parameters:

Name	Type	Description
record	JSON	summary to insert
funcionCallback	function	

Source: modules/DBManager.js, line 327

Figura 67. Código del módulo DBManager (IX)

Module: modules/matchValidationManager

Match validator

Source: modules/matchValidationManager.js, line 1

Methods

(static) matchCreation(match, funcionCallback)

Validation for creating a match

Parameters:

Name	Type	Description
match	JSON	to create
funcionCallback	function	

Source: modules/matchValidationManager.js, line 22

Figura 68. Código del módulo matchValidationManager

Module: modules/teamValidationManager

Team validator

Source: modules/teamValidationManager.js, line 1

Methods

(static) coachAddition(coach, funcionCallback)

Validation for creating a coach

Parameters:

Name	Type	Description
coach	JSON	to create
funcionCallback	function	

Source: modules/teamValidationManager.js, line 40

Figura 69. Código del módulo teamValidationManager (I)

(static) playerAddition(player, funcionCallback)

Validation for creating a player

Parameters:

Name	Type	Description
player	JSON	to create
funcionCallback	function	

Source: modules/teamValidationManager.js, line 58

(static) teamCreation(team, funcionCallback)

Validation for creating a team

Parameters:

Name	Type	Description
team	JSON	to create
funcionCallback	function	

Source: modules/teamValidationManager.js, line 22

Figura 70. Código del módulo teamValidationManager (II)

Module: modules/userValidationManager

User validator

Source: modules/userValidationManager.js, line 1

Methods

(static) login(user, funcionCallback)

Validation for login an user

Parameters:

Name	Type	Description
user	JSON	to login
funcionCallback	function	

Source: modules/userValidationManager.js, line 42

Figura 71. Código del módulo userValidationManager (I)

(static) userRegister(user, funcionCallback)

Validation for register an user

Parameters:

Name	Type	Description
user	JSON	to register
funcionCallback	function	

Source: modules/userValidationManager.js, line 22

Figura 72. Código del módulo userValidationManager (II)

com.tfg.cristian.tfgbasket.objects.events

Interface Event

public interface **Event**

Method Summary

All Methods | Instance Methods | Abstract Methods

Modifier and Type	Method and Description
void	**emit**(com.github.nkzawa.socketio.client.Socket socket) Send the event to the socket that is listening
com.tfg.cristian.tfgbasket.objects.enums.EventType	**getEventType**() Return the type of event
void	**post**(com.android.volley.RequestQueue queue) Use the REST service to insert the event into the database

Figura 73. Código de la interface Event

com.tfg.cristian.tfgbasket.objects

Class Match

java.lang.Object
  com.tfg.cristian.tfgbasket.objects.Match

All Implemented Interfaces:

java.io.Serializable

public class **Match**
extends java.lang.Object
implements java.io.Serializable

See Also:

Serialized Form

Figura 74. Código de la clase Match (I)

Field Summary

Fields

Modifier and Type	Field and Description
private java.lang.String	_id
private java.lang.String	court
private java.lang.String	date
private int	durationQuarter
private java.util.List<User>	followers
private com.tfg.cristian.tfgbasket.objects.Team	localTeam
private int	maxPersonalFouls
private int	quartersNumber
private boolean	runningTime
private java.lang.String	state
private User	tableOfficial
private java.lang.String	time
private int	timeOuts
private com.tfg.cristian.tfgbasket.objects.Team	visitorTeam

Figura 75. Código de la clase Match (II)

Constructor Summary

Constructors

Constructor and Description
Match(java.lang.String _id, int quartersNumber, int durationQuarter, boolean runningTime, int timeOuts, int maxPersonalFouls, java.lang.String date, java.lang.String time, java.lang.String court, User tableOfficial, java.lang.String state) Constructor that creates a match without teams
Match(java.lang.String _id, com.tfg.cristian.tfgbasket.objects.Team localTeam, com.tfg.cristian.tfgbasket.objects.Team visitorTeam, int quartersNumber, int durationQuarter, boolean runningTime, int timeOuts, int maxPersonalFouls, java.lang.String date, java.lang.String time, java.lang.String court, User tableOfficial, java.lang.String state) Overloaded contructor that creates a match with the two teams

Figura 76. Código de la clase Match (III)

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method and Description
java.lang.String	**get_id**()
java.lang.String	**getCourt**()
java.lang.String	**getDate**() Transform date in format yyyy-MM-dd into dd/MM/yyyy
int	**getDurationQuarter**()
void	**getEvents**(com.android.volley.RequestQueue queue, android.widget.ArrayAdapter<java.lang.String> adapter, java.util.List<java.lang.String> events) Obtain all events of a match from database
java.util.List<User>	**getFollowers**()
com.tfg.cristian.tfgbasket.objects.Team	**getLocalTeam**()
int	**getMaxPersonalFouls**()
int	**getQuartersNumber**()
void	**getScoreboard**(com.android.volley.RequestQueue queue, android.widget.TextView tvLocalPoints, android.widget.TextView tvVisitorPoints) Obtain the scoreboard of a match at one point
java.lang.String	**getState**()

Figura 77. Código de la clase Match (IV)

User	**getTableOfficial**()
java.lang.String	**getTime**()
int	**getTimeOuts**()
com.tfg.cristian.tfgbasket.objects.Team	**getVisitorTeam**()
boolean	**isRunningTime**()
void	**putFinished**(com.android.volley.RequestQueue queue, com.tfg.cristian.tfgbasket.objects.Record record) Put the state of the match in database as finished and add the scoreboard of the match
void	**putFollower**(com.android.volley.RequestQueue queue, User user) Put one follower to a match
void	**putPlaying**(com.android.volley.RequestQueue queue) Put the state of the match in database as playing
private void	**recordPlayer**(com.android.volley.RequestQueue queue, java.lang.String url, com.tfg.cristian.tfgbasket.objects.Player player, com.tfg.cristian.tfgbasket.objects.Team team) Save the summary of one match player into database
void	**recordSummary**(com.android.volley.RequestQueue queue) Save the summary of the match players into database
void	**removeFollower**(com.android.volley.RequestQueue queue, User user) Remove one follower from a match
private void	**removeFromListFollowers**(User user)
void	**set_id**(java.lang.String _id)

Figura 78. Código de la clase Match (V)

void	**setCourt**(java.lang.String court)
void	**setDate**(java.lang.String date)
void	**setDurationQuarter**(int durationQuarter)
void	**setFollowers**(java.util.List<**User**> followers)
void	**setListening**(com.github.nkzawa.socketio.client.Socket socket, com.tfg.cristian.tfgbasket.FollowMatch context, android.widget.ArrayAdapter<java.lang.String> adapter, java.util.List<java.lang.String> events, android.widget.TextView tvLocalPoints, android.widget.TextView tvVisitorPoints) Set listening of the socket to communicate the server with the app in real time
void	**setLocalTeam**(com.tfg.cristian.tfgbasket.objects.Team localTeam)
void	**setMaxPersonalFouls**(int maxPersonalFouls)
void	**setQuartersNumber**(int quartersNumber)
void	**setRunningTime**(boolean runningTime)
void	**setState**(java.lang.String state)
void	**setTableOfficial**(**User** tableOfficial)
void	**setTime**(java.lang.String time)
void	**setTimeOuts**(int timeOuts)
void	**setVisitorTeam**(com.tfg.cristian.tfgbasket.objects.Team visitorTeam)
java.lang.String	**toString**()

Figura 79. Código de la clase Match (VI)

private void	**updateScoreboard**(int points, java.lang.String teamId, android.widget.TextView tvLocalPoints, android.widget.TextView tvVisitorPoints) Update scoreboard and show it

Figura 80. Código de la clase Match (VII)

com.tfg.cristian.tfgbasket.objects

Class User

java.lang.Object
    com.tfg.cristian.tfgbasket.objects.User

All Implemented Interfaces:
java.io.Serializable

```
public class User
extends java.lang.Object
implements java.io.Serializable
```

See Also:
Serialized Form

Field Summary

Fields

Modifier and Type	Field and Description
private java.lang.String	**_id**
private java.lang.String	**password**
private java.lang.String	**userName**

Figura 81. Código de la clase User (I)

Constructor Summary

Constructors

Constructor and Description
`User(java.lang.String userName)` Constructor to create an user with only user name
`User(java.lang.String _id, java.lang.String userName)` Overloaded constructor to create an user with id and user name
`User(java.lang.String _id, java.lang.String userName, java.lang.String password)` Overloaded constructor to create an user with id, user name and password

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method and Description
java.lang.String	`getPassword()`
java.lang.String	`getUserName()`
void	`post(com.android.volley.RequestQueue queue)` Add the user to the database

Figura 82. Código de la clase User (II)

com.tfg.cristian.tfgbasket.services

Class IntroducePlayersService

java.lang.Object
  com.tfg.cristian.tfgbasket.services.IntroducePlayersService

---

```
public class IntroducePlayersService
extends java.lang.Object
```

Constructor Summary

Constructors

Constructor and Description
`IntroducePlayersService()`

Figura 83. Código del Servicio IntroducePlayersService (I)

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method and Description
void	`getCoaches(com.android.volley.RequestQueue queue, com.tfg.cristian.tfgbasket.objects.Team team)` Obtain coaches from a team
void	`getTeamPlayers(com.android.volley.RequestQueue queue, android.widget.ArrayAdapter<com.tfg.cristian.tfgbasket.objects.Player> adapterLocal, android.widget.ArrayAdapter<com.tfg.cristian.tfgbasket.objects.Player> adapterVisitor, com.tfg.cristian.tfgbasket.objects.Team team)` Obtain players from a team

Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Figura 84. Código del Servicio IntroducePlayersService (II)

com.tfg.cristian.tfgbasket.services

Class ListMatchesService

java.lang.Object
com.tfg.cristian.tfgbasket.services.ListMatchesService

```
public class ListMatchesService
extends java.lang.Object
```

Constructor Summary

Constructors
Constructor and Description
`ListMatchesService()`

Figura 85. Código del Servicio ListMatchesService (I)

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method and Description
void	`getActiveMatches(com.android.volley.RequestQueue queue, android.widget.ArrayAdapter<Match> adapter, java.util.List<Match> matches, User user)` Obtain matches which its state is active and assigned table official is the user of the parameter
void	`getAllMatches(com.android.volley.RequestQueue queue, android.widget.ArrayAdapter<Match> adapter, java.util.List<Match> matches)` Obtain all matches which its state is not created
void	`getFollowedMatches(com.android.volley.RequestQueue queue, android.widget.ArrayAdapter<Match> adapter, java.util.List<Match> matches, User user)` Obtain all matches followed by an user
void	`getMatchSummary(com.android.volley.RequestQueue queue, android.widget.ArrayAdapter<com.tfg.cristian.tfgbasket.objects.Statistic> adapter, java.util.List<com.tfg.cristian.tfgbasket.objects.Statistic> statistics, Match match)` Obtain a summary of players' match

Figura 86. Código del Servicio ListMatchesService (II)

com.tfg.cristian.tfgbasket.services

Class ListUsersService

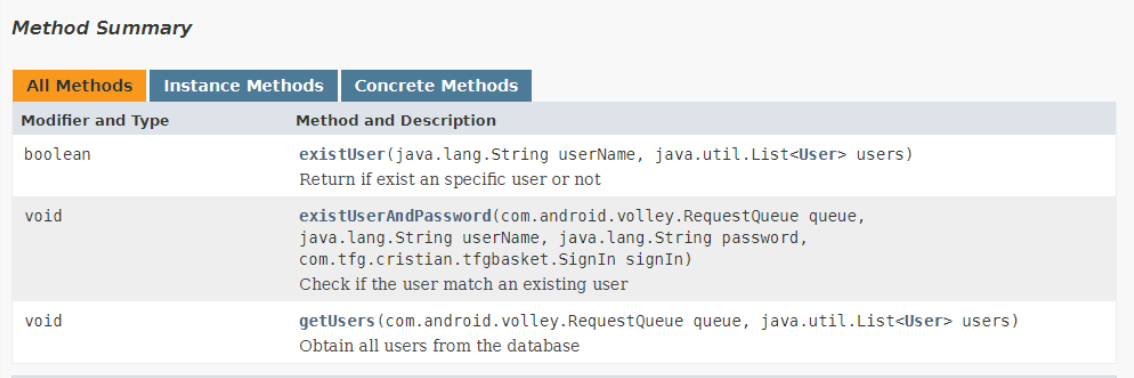
java.lang.Object
com.tfg.cristian.tfgbasket.services.ListUsersService

```
public class ListUsersService
extends java.lang.Object
```

Constructor Summary

Constructors
Constructor and Description
`ListUsersService()`

Figura 87. Código del Servicio ListUsersService (I)



Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method and Description
boolean	**existUser**(java.lang.String userName, java.util.List<**User**> users) Return if exist an specific user or not
void	**existUserAndPassword**(com.android.volley.RequestQueue queue, java.lang.String userName, java.lang.String password, com.tfg.cristian.tfgbasket.SignIn signIn) Check if the user match an existing user
void	**getUsers**(com.android.volley.RequestQueue queue, java.util.List<**User**> users) Obtain all users from the database

Figura 88. Código del Servicio ListUsersService (II)

Cuestionario para pruebas de usabilidad

Edad	52
Nivel de informática (del 1 al 10)	5
Uso de aplicaciones web (del 1 al 10)	5
Uso de aplicaciones móvil (del 1 al 10)	6
Familiarización con la temática del sistema (baja, media, alta)	Alta
Puntuación del sistema (del 1 al 10)	8
Comentarios (al finalizar la prueba)	La interfaz gráfica es un poco pobre.
Observaciones	Click en las letras del menú de la aplicación móvil.

Tabla 45. Cuestionario previo al usuario 1 de pruebas de usabilidad

Edad	50
Nivel de informática (del 1 al 10)	7
Uso de aplicaciones web (del 1 al 10)	8
Uso de aplicaciones móvil (del 1 al 10)	8
Familiarización con la temática del sistema (baja, media, alta)	Baja
Puntuación del sistema (del 1 al 10)	9
Comentarios (al finalizar la prueba)	
Observaciones	Click en las letras del menú de la aplicación móvil. Dificultad al añadir los jugadores de un partido.

Tabla 46. Cuestionario previo al usuario 2 de pruebas de usabilidad

Edad	48
Nivel de informática (del 1 al 10)	8
Uso de aplicaciones web (del 1 al 10)	8
Uso de aplicaciones móvil (del 1 al 10)	10
Familiarización con la temática del sistema (baja, media, alta)	Media
Puntuación del sistema (del 1 al 10)	7
Comentarios (al finalizar la prueba)	Tantos botones lían un poco.
Observaciones	Dificultad al añadir los jugadores de un partido. No encuentra algunas acciones.

Tabla 47. Cuestionario previo al usuario 3 de pruebas de usabilidad

Edad	22
Nivel de informática (del 1 al 10)	10
Uso de aplicaciones web (del 1 al 10)	10
Uso de aplicaciones móvil (del 1 al 10)	9
Familiarización con la temática del sistema (baja, media, alta)	Baja
Puntuación del sistema (del 1 al 10)	9
Comentarios (al finalizar la prueba)	Está muy bien porque tiene bastante funcionalidad.
Observaciones	Click en las letras del menú de la aplicación móvil. Usó el calendario para la fecha de los partidos. Añadió los jugadores bien. No hace scroll para ver más opciones. Dificultad al añadir un segundo.

Tabla 48. Cuestionario previo al usuario 4 de pruebas de usabilidad

Edad	22
Nivel de informática (del 1 al 10)	6
Uso de aplicaciones web (del 1 al 10)	6
Uso de aplicaciones móvil (del 1 al 10)	7
Familiarización con la temática del sistema (baja, media, alta)	Media
Puntuación del sistema (del 1 al 10)	8
Comentarios (al finalizar la prueba)	
Observaciones	No encuentra el registro en la web. No hace scroll para ver más opciones.

Tabla 49. Cuestionario previo al usuario 5 de pruebas de usabilidad

Edad	20
Nivel de informática (del 1 al 10)	9
Uso de aplicaciones web (del 1 al 10)	9
Uso de aplicaciones móvil (del 1 al 10)	10
Familiarización con la temática del sistema (baja, media, alta)	Alta
Puntuación del sistema (del 1 al 10)	8

Comentarios (al finalizar la prueba)	Posibilidad de incluir un enlace al registro en el formulario de inicio de sesión en web como está en móvil.
Observaciones	Click en las letras del menú de la aplicación móvil.

Tabla 50. Cuestionario previo al usuario 6 de pruebas de usabilidad

Edad	24
Nivel de informática (del 1 al 10)	7
Uso de aplicaciones web (del 1 al 10)	8
Uso de aplicaciones móvil (del 1 al 10)	9
Familiarización con la temática del sistema (baja, media, alta)	Baja
Puntuación del sistema (del 1 al 10)	9
Comentarios (al finalizar la prueba)	Aunque no entienda del tema me parece intuitivo.
Observaciones	Click en las letras del menú de la aplicación móvil. No hace scroll para ver más opciones.

Tabla 51. Cuestionario previo al usuario 7 de pruebas de usabilidad

Edad	54
Nivel de informática (del 1 al 10)	9
Uso de aplicaciones web (del 1 al 10)	8
Uso de aplicaciones móvil (del 1 al 10)	10
Familiarización con la temática del sistema (baja, media, alta)	Media
Puntuación del sistema (del 1 al 10)	10
Comentarios (al finalizar la prueba)	Falta alguna imagen para llamar más la atención pero está muy bien.
Observaciones	

Tabla 52. Cuestionario previo al usuario 8 de pruebas de usabilidad

Script de rellenado de datos

```
module.exports = function(app, swig, DBManager) {
    app.get("/fillDB", function (req, res) {
        DBManager.resetDB(function () {
            res.send({reset: true});
        });

        //Users
        var user1 = {
            userName: "test",
            password:
"1392542397501e1158418adae09d694ffb8ed833a3a5e8a017e15ba565d28c70"
        };

        //Teams
        var team1 = {
            teamName: "Barça Lassa",
            teamCourt: "Palau"
```

```
};

var team2 = {
    teamName: "Real Madrid",
    teamCourt: "Palacio de los Deportes"
};

var team3 = {
    teamName: "Sanfer",
    teamCourt: "Colegio San Fernando"
};

var team4 = {
    teamName: "ADBA",
    teamCourt: "Polideportivo Quirinal"
};

var team5 = {
    teamName: "Llaranes",
    teamCourt: "La Toba"
};

var team6 = {
    teamName: "test",
    teamCourt: "test"
};

var team7 = {
    teamName: "test2",
    teamCourt: "test2"
};

//Coaches
var coach1 = {
    teamName: "Sanfer",
    coachName: "Pablo"
};

var coach2 = {
    teamName: "Llaranes",
    coachName: "Suso"
};

//Players
var player1 = {
    teamName : "Sanfer",
    playerName : "Cristian",
    playerBib : 23
};

var player2 = {
    teamName : "Sanfer",
    playerName : "Adrián",
    playerBib : 15
};

var player3 = {
    teamName : "Llaranes",
```

```
    playerName : "Dani",
    playerBib : 10
};

var player4 = {
    teamName : "test",
    playerName : "Test",
    playerBib : 0
};

//Matches
var match1 = {
    localTeam : "ADBA",
    visitorTeam : "Barça Lassa",
    quartersNumber : 4,
    durationQuarter : 10,
    runningTime : false,
    timeOuts : 2,
    maxPersonalFouls : 5,
    date : "2020-10-23",
    time : "18:00",
    matchCourt : "Niemeyer",
    tableOfficial : "",
    userName : "test",
    followers : [],
    state : "created"
};

var match2 = {
    localTeam : "Sanfer",
    visitorTeam : "ADBA",
    quartersNumber : 4,
    durationQuarter : 1,
    runningTime : false,
    timeOuts : 2,
    maxPersonalFouls : 5,
    date : "2020-04-10",
    time : "20:11",
    matchCourt : "Polideportivo Quirinal",
    tableOfficial : "",
    userName : "test",
    followers : [],
    state : "active"
};

var match3 = {
    localTeam : "Llaranes",
    visitorTeam : "Real Madrid",
    quartersNumber : 4,
    durationQuarter : 1,
    runningTime : false,
    timeOuts : 2,
    maxPersonalFouls : 5,
    date : "2020-04-10",
    time : "12:30",
    matchCourt : "La Toba",
    tableOfficial : "test",
    userName : "test",
```

```
            followers : [],
            state : "finished"
        };

        //Events

        //Records

        DBManager.insertUser(user1, function() {
            DBManager.insertTeam(team1, function () {
                DBManager.insertTeam(team2, function () {
                    DBManager.insertTeam(team3, function () {
                        DBManager.insertTeam(team4, function () {
                            DBManager.insertTeam(team5, function ()
{
                                DBManager.insertTeam(team6,
function () {
                                    DBManager.insertTeam(team7,
function () {

DBManager.insertCoach(coach1, function () {

DBManager.insertCoach(coach2, function () {

DBManager.insertPlayer(player1, function () {

DBManager.insertPlayer(player2, function () {

DBManager.insertPlayer(player3, function () {

DBManager.insertPlayer(player4, function () {

DBManager.insertMatch(match1, function () {

DBManager.insertMatch(match2, function () {

DBManager.insertMatch(match3, function() {

res.send({

users: "usuarios insertados",

teams: "equipos insertados",

coaches: "entrenadores insertados",

players: "jugadores insertados",

matches: "partidos insertados",

events: "eventos insertados",

records: "historial insertado"

});

});
```

```
});
                                                                                    });
                                                                                });
                                                                            });
                                                                        });
                                                                    });
                                                                });
                                                            });
                                                        });
                                                    });
                                                });
                                            });
                                        });
                                    });
                                });
                            });
                        });
                    });
                });
            });
        });
};
```