Facultad
de
Ciencias

Aplicación móvil para generar colaborativamente apuestas deportivas

Smartphone App for collaboratively sports betting

Trabajo de Fin de Grado para
acceder al

GRADO EN INGENIERÍA INFORMÁTICA

Autor: MARIO VIADERO HIDALGO

Director: RAFAEL DUQUE MEDINA

SEPTIEMBRE – 2018

RESUMEN

El sector de las apuestas deportivas on-line es uno de los más importantes modelos de negocio a través de la red. Con el paso del tiempo y el aumento del número de casas de apuestas, los usuarios tienen a su disposición mejores tipos de ofertas y promociones como pueden ser bonos de bienvenida, devolución de dinero en caso de fallar el pronóstico y mejores beneficios en el caso de apuestas combinadas.

Esto está suponiendo un gran aumento en el número de usuarios, la aparición de blogs con pronósticos recomendados, también de apostadores profesionales que se ganan la vida apostando en estas plataformas y de grupos de personas que comparten cuenta y pronósticos con el fin de obtener un beneficio común, o en el caso contrario, minimizar las pérdidas.

Cada vez está más extendido apostar colaborativamente en grupo, un símil puede ser las peñas quinielísticas o algo tan tradicional como comprar lotería con los compañeros de trabajo, amigos o familiares. El problema es que las apuestas deportivas online cambian continuamente, las cuotas no son las mismas en función de cómo se sucedan los acontecimientos. Por todo ello es necesario tener velocidad de decisión y consenso con el resto de compañeros para apostar con las mejores cuotas, y por tanto obtener el mayor beneficio de nuestros pronósticos.

En este contexto de negocio nace BETSHARE, un proyecto que plantea desarrollar una aplicación móvil para facilitar y agilizar este tipo de apuestas grupales, donde se puedan formar grupos de personas, apostar conjuntamente, recibir pronósticos de otros compañeros de grupo para dar nuestro visto bueno con un simple clic y chatear en grupo para opinar sobre los pronósticos

Por último la aplicación tendrá el apoyo de un sistema web para facilitar la administración de los eventos, sus cuotas y resultados, además de los usuarios registrados.

Palabras clave: Aplicación colaborativa, apuestas deportivas online, web, app, smartphone

Abstract

The field of online sports betting is one of the most important business models across the network. With the passage of time and the increasing number of gamblers, users have at their disposal better types of offers and promotions such as bonuses, money back in case of lost bets and better benefits in the case of combined bets.

This is meaning a large increase in the number of users, the appearance of blogs with odds recommended, also professional gamblers who make a living betting on these platforms and groups of people who share account and bets in order to make a profit common, or otherwise minimize losses.

It is increasingly widespread bet collaboratively in groups, a comparison can be the 1x2 bets groups or something as traditional as buying lottery with co-workers, friends or family. The problem is that online sports betting continually changing, odds are not the same depending on how events happen. Therefore, it is necessary to have speed decision-making and consensus with the rest of companions to bet with the best fares, and therefore get the most benefit from our forecasts.

In this business context BETSHARE was born, a project which aims to develop a mobile application to facilitate and expedite this type of group bets, where they can form groups of people, bet together, receive bets from other groupmates to give our approval with a simple click and chat in groups to review forecasts.

Finally, the application will be supported by a web system to facilitate the administration of events, odds, results, and registered users.

Keywords: collaborative application, online sports betting, web, app, smartphone

ÍNDICE

RESUMEN 1
Abstract 2
ÍNDICE 3
ÍNDICE DE FIGURAS 4
ÍNDICE DE TABLAS 4
1. INTRODUCCIÓN 5
1.1 Motivación y contexto tecnológico 5
1.2 Objetivos 6
1.3 Estructura del documento 6
2. MATERIAL Y MÉTODOS 7
2.1 Herramientas y tecnologías 7
2.2 Metodología 10
3. ANÁLISIS DE REQUISITOS 11
3.1 Requisitos funcionales 11
3.1.1 Requisitos funcionales para el invitado 13
3.1.2 Requisitos funcionales para el usuario 15
3.1.3 Requisitos funcionales para el administrador 19
3.2 Requisitos no funcionales 23
3.2.1 Usabilidad y apariencia 23
3.2.2 Requisitos Hardware 23
3.2.3 Requisitos legales 23
3.2.4 Requisitos de seguridad 23
4. DISEÑO E IMPLEMENTACIÓN 24
4.1 Diseño arquitectónico 24
4.2 Diseño e implementación la app 24
4.2.1 Diseño e implementación de la capa de persistencia 25
4.2.2 Diseño e implementación de la capa de negocio 28
4.2.3 Diseño e implementación de la capa de presentación 33
4.3 Diseño e implementación de la web de administración 39
5. Pruebas 41
5.1 Pruebas a la web de administración 41
5.2 Pruebas a la aplicación móvil 42
6. Conclusiones 46
6.1 Consecución de objetivos 46
6.2 Trabajos futuros 48
Referencias 49
Anexo I. Acrónimos 50
Anexo II. Prototipos Axure RP 51

ÍNDICE DE FIGURAS

Figura 1 Ciclo de vida del desarrollo iterativo incremental [16] 10
Figura 2 Diagrama de casos de uso 12
Figura 3 Modelo Entidad-Relación 25
Figura 4 Diseño de la base de datos 27
Figura 5 Estructura de ficheros de la aplicación Android 28
Figura 6 Vista registro de nuevo usuario 29
Figura 7 Función doRegistrarse() de la clase registro.js 29
Figura 8 Función sendData de la clase Service.js 30
Figura 9 Función doRegistrarse de la clase Service.js 30
Figura 10 Traducción de URLs en routes.php de CodeIgniter 30
Figura 11 Carpeta controllers (pantalla izquierda) - Carpeta api (pantalla derecha) 31
Figura 12 Código de la función de registro de usuario del controlador User_controller 31
Figura 13 Función registro_usuario del modelo usuario_model.php 32
Figura 14 Configuración de conexión a la base de datos en el fichero database.php 32
Figura 15 Vistas de la aplicación móvil 33
Figura 16 Pantalla de login (izquierda) - Pantalla de registro (derecha) 34
Figura 17 Logueado donde vemos los eventos (izquierda) - Filtrado de eventos por competición (derecha) 34
Figura 18 Sección de código de la vista logueado.html para la parte de los eventos 35
Figura 19 Ejecución de la función getInitData() en logueado.html 35
Figura 20 Carga del script logueado.js en la vista logueado.html 35
Figura 21 Menú lateral (izquierda) - Cupón Apuestas (centro) - Pantalla Grupos (derecha) 36
Figura 22 Información del grupo (izquierda) - Comentarios (centro) - Historial de apuestas (derecha) 37
Figura 23 Input para introducir la cantidad a apostar en la clase logueado.html 38
Figura 24 Código con funciones del cupón de apuestas en la clase logueado.js 38
Figura 25 Formulario de la vista de creación de competición 39
Figura 26 Sección de código de la función comprobar_admin() 39
Figura 27 Funciones check_nombre y check_deporte dentro del controlador Competiciones_controller 40
Figura 28 Web de administración. Formulario de creación de nuevo evento 40
Figura 29 Prueba de la web de administración con Selenium 42
Figura 30 Métodos de prueba test_doEnviarCupon() y test_getCupones() 43
Figura 31 Cobertura de las pruebas en la clase apuestas_model.php 44
Figura 32 Cobertura de las pruebas en la clase usuario_model.php 44
Figura 33 Fragmento del método doEnviarCupon() 45
Figura 34 Código del método test_doEnviarCupon() 45

ÍNDICE DE TABLAS

Tabla 1 Especificación de requisitos funcionales 12

1. INTRODUCCIÓN

1.1 Motivación y contexto tecnológico
1.2 Objetivos
1.3 Estructura del documento

La introducción consta de tres apartados en los cuales se explican los objetivos y la motivación por la cual se realiza este proyecto. En el primer apartado se explica el contexto tecnológico del mundo de las apuestas deportivas online y la motivación para solucionar la problemática de las apuestas colaborativas online. El segundo apartado describe los objetivos que se deben alcanzar durante el desarrollo del proyecto. Por último, el tercer apartado muestra la estructura del resto de la memoria.

1.1 Motivación y contexto tecnológico

El sector de las apuestas deportivas on-line es un modelo de negocio en auge. En los últimos tiempos es muy común que grupos de apostantes compartan una cuenta en alguna casa de apuestas o, simplemente, que cada usuario use su propia cuenta para realizar apuestas colaborativas con otras personas. En este contexto se presenta el reto de facilitar mecanismos para debatir las cantidades y los pronósticos por parte de todos los miembros de un grupo de apostantes. Además, debido a que las cuotas varían continuamente en función de la actualidad que precede al evento deportivo, es necesario decidir con rapidez y llegar a un acuerdo con la mayor brevedad para realizar la apuesta en el momento en el que la cuota sea la más beneficiosa para el grupo.

Un ejemplo paradigmático de las peculiaridades de las apuestas colaborativas es el caso del grupo de amigos que elaboran apuestas deportivas online de forma común y uno de ellos observa que hay una interesante cuota para una determinada apuesta. Las apuestas deportivas pueden cambiar en un instante, en el fútbol, por ejemplo, una lesión de un jugador importante los días previos al evento haría variar las cuotas del evento, y, por lo tanto, hay que apostar prácticamente al minuto. Es entonces cuando nos encontramos ante el problema de poner en comunicación a los miembros del grupo, explicar de qué se trata y empezar a debatir sobre las cuotas y sobre la conveniencia o no de realizar la apuesta.

Este tipo de situaciones motivan un proyecto que desarrolle una aplicación para facilitar la colaboración entre sus usuarios a la hora de realizar apuestas y proporcione los mecanismos de comunicación y coordinación entre grupos de personas que apuestan colaborativamente. Por lo tanto, la meta del proyecto es desarrollar una aplicación para agilizar todo el proceso para realizar apuestas colaborativas. Para ello, se aprovechará el actual contexto tecnológico en el que vivimos, caracterizado por el fácil y rápido acceso a internet a través de los Smartphone. Así el proyecto contempla el desarrollo de una aplicación móvil para Smartphone que facilite a los usuarios la realización de apuestas y un sistema web que facilite la gestión de usuario y eventos deportivos.

1.2 Objetivos

El proyecto tiene un objetivo principal que es desarrollar una aplicación móvil para facilitar y agilizar las apuestas grupales, donde se puedan formar grupos de personas, apostar conjuntamente, recibir pronósticos de otros compañeros de grupo, votar positiva o negativamente los pronósticos propuestos en el grupo y chatear en grupo para opinar sobre los pronósticos. Además, un sistema web deberá facilitar la administración de los usuarios, eventos, sus cuotas y resultados. En consecuencia se pueden establecer los siguientes sub-objetivos:

1. Diseñar un sistema web que permita:
   1.1 Darse de alta en el sistema como administrador.
   1.2 Administrar competiciones y eventos deportivos.
   1.3 Administrar cuotas y resultados de los eventos deportivos
2. Diseñar una aplicación para dispositivos móviles inteligentes que permita:
   2.1 Darse de alta en el sistema como usuario.
   2.2 Ver los eventos deportivos, sus cuotas y filtrar los eventos por competición deportiva.
   2.3 Crear grupos de apuestas y enviar invitaciones a otros usuarios
   2.4 Ver los grupos en los que somos participante y la descripción del mismo, así como los otros participantes del grupo.
   2.5 Enviar cupones a los grupos con pronósticos y cantidades para proceder a la votación al instante.
   2.6 Chatear con el resto de integrantes del grupo a través de un tablón de mensajes y valorar los comentarios.
   2.7 Ver un histórico de los cupones de apuestas de cada grupo.

1.3 Estructura del documento

Esta memoria incluye cinco capítulos adicionales. El contenido de cada uno de estos capítulos es el siguiente:

- El **Capítulo 2: Material y métodos** donde se describe las distintas herramientas, tecnologías y metodologías utilizadas para la realización del proyecto.
- El **Capítulo 3: Análisis de requisitos** donde se definen los requisitos funcionales y los correspondientes casos de uso. Además este capítulo expone los requisitos no funcionales de la aplicación.
- El **Capítulo 4: Diseño e implementación** donde se describe el proceso de diseño y construcción de la aplicación.
- El **Capítulo 5: Evaluación y pruebas** donde se muestra el contenido y el resultado de las pruebas realizadas para verificar y validar la aplicación.
- El **Capítulo 6: Conclusiones y trabajos futuros** donde se exponen las diferentes conclusiones derivadas de la realización del proyecto y de la herramienta como tal. Además, se discuten las líneas de trabajo futuro que abre este proyecto.

2. MATERIAL Y MÉTODOS

2.1 Herramientas y tecnologías
2.2 Metodología

El segundo capítulo del documento contiene dos apartados, en el primero se describe detalladamente las herramientas y tecnologías utilizadas para la realización del proyecto. En el segundo apartado se describe la metodología de desarrollo software empleada para la realización del proyecto.

2.1 Herramientas y tecnologías

En este apartado se van a describir las diferentes herramientas y tecnologías utilizadas en el proyecto, en qué consisten los framework CodeIgniter y Android Studio y las tecnologías utilizadas para la realización del proyecto.

Axure RP PRO 7.0

Axure RP Pro 7.0 es una herramienta para crear prototipos de la interfaz de usuario de páginas webs y de aplicaciones sin necesidad de escribir código fuente. Tiene la posibilidad de generar el prototipo en código HTML o JavaScript para poder ser visionado en el navegador, de esta forma clientes, desarrolladores y stakeholders pueden visionar los prototipos creados con Axure sin necesidad de tenerlo instalado.

Android Studio

Android Studio [1] es el IDE (Entorno de Desarrollo Integrado) oficial de Android. Es de uso gratuito y está disponible para Windows, Mac OS X y Linux. Una de las características importantes de Android Studio es que cuenta con *AVD Manager (Android Virtual Devices Manager)* para emular terminales virtuales a los que se les puede modificar sus especificaciones y probar el funcionamiento de las aplicaciones.

Por otro lado, *Android Studio* cuenta con el *SDK Manager (Software Development Kit)*, donde el usuario puede seleccionar y descargar las herramientas, componentes y APIs (*Application Programming Interface*) de la versión Android que crea conveniente.

De este *SDK Manager,* podemos descargar *Google USB Driver* el cual instala los drivers USB necesarios para poder probar las aplicaciones directamente en el teléfono, conectando el teléfono al PC por USB y poniendo el smartphone en modo desarrollado, en vez de utilizar el simulador virtual explicado en anteriormente.

MySQL

MySQL [2] es un sistema de gestión de bases de datos relacional, bajo licencia dual: Licencia pública general GNU / Licencia comercial por Oracle Corporation. Está considerada como la base de datos de código abierto más popular del mundo. MySQL utiliza lenguaje de consultas a bases de datos SQL (*Structured Query Language*), que nos permitirá crear, modificar, consultar y eliminar tanto bases de datos, como sus tablas y registros. Se puede combinar con PHP para realizar la interacción de la aplicación web con la base de datos.

MySQL WorkBench 6.3 CE

MySQL WorkBench 6.3 CE [3] es una herramienta que permite modelar diagramas Entidad-Relación para bases de datos MySQL. Se puede utilizar para diseñar el esquema de una base de datos nueva y generar el script necesario para la creación de la base de datos que se ha modelado en el esquema. Además proporciona herramientas visuales para crear, ejecutar y optimizar consultas SQL así como un editor SQL con color resaltado de sintaxis e historial de ejecución.

HTML

HTML (HyperText Markup Language) [4] es el lenguaje básico de la World Wide Web, estandarizado por la W3C (WorldWideWeb Consortium). HTML sirve para crear webs, darlas estructura y contenido, está formado por una serie de etiquetas que el navegador interpreta y forma en la pantalla. La versión que se utiliza en el proyecto es su versión actual, HTML 5 que contiene mejoras respecto a sus predecesores en el diseño de formularios, nuevos elementos multimedia y nuevas etiquetas semánticas.

CSS

CSS (Cascading Style Sheets) [5] o lo que es lo mismo hojas de estilo en cascada, es un lenguaje que sirve para definir la presentación de los documentos HTML. Permite controlar la presentación de muchos documentos HMTL desde una sola hoja de estilo, abarcando cuestiones relativas al tamaño, posicionamiento, tipos de fuente, colores, márgenes, imágenes de fondo entre muchas otras funcionalidades. La versión actual es CSS3 y es soportada por todos los navegadores actuales.

PHP

PHP (*Hypertext Preprocessor)* [6] es un lenguaje muy popular y especialmente adecuado para el desarrollo web ya que puede ser incrustado en HTML. Suele estar asociado a bases de datos MySQL. En este proyecto se utilizan scripts PHP incrustados en las vistas de la web de administración para realizar peticiones de datos a la base de datos, así como en todos los controladores y modelos de Codeigniter.

CodeIgniter

CodeIgniter [7] es un framework para el desarrollo de aplicaciones en lenguaje PHP, utiliza la arquitectura modelo-vista-controlador y contiene una serie de helpers y librerias que son de mucha ayuda para realizar acciones sencillas, además permite crear las tuyas propias para llevar a cabo acciones más concretas y complejas. CodeIgniter es distribuido bajo licencia de código abierto.

JavaScript

JavaScript [8] es un lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas. Una web dinámica es aquella que incorpora efectos y acciones por ejemplo al pulsar un botón. Una de las ventajas de este lenguaje es que no es necesario compilar los programas para ejecutarlos, al ser un lenguaje interpretado los programas se pueden probar directamente en el navegador sin necesidad de intermediarios.

Ajax

Ajax (Asynchronous JavaScript and XML) [9] es una técnica para desarrollar páginas web que implementan aplicaciones interactivas, permitiendo mediante programas escritos en JavaScript, intercambiar información con el servidor de forma asíncrona. La gran ventaja de Ajax es que al realizar la petición y al realizar cambios en la página no es necesario recargarla.

JSON

JSON (*JavaScript Object Notation*) [10] es un formato ligero para el intercambio de datos. Una de las mayores ventajas que tiene el uso de JSON es que puede ser leído por cualquier lenguaje de programación y ser usado para el intercambio de información entre distintas tecnologías. En concreto en este proyecto podemos realizar el intercambio de datos entre el Servicio en JavaScript y la base de datos a través de los controladores y modelos de Codeigniter en PHP.

Bootstrap

Bootstrap [11] es un framework CSS y JS desarrollado originalmente por Mark Otto y Jacob Thornton de Twitter. Este framework permite la creación de aplicaciones y sitios webs adaptables a cualquier dispositivo, ofreciendo un amplio abanico de herramientas y funciones, así como plantillas de diseño con tipografía, formularios, botones, cuadros y otros elementos.

jQuery

jQuery [12] es una librería de JavaScript de código abierto que permite agilizar la gestión de eventos y animaciones de una página HTML, su principal funcionalidad es que sirve interactuar con ciertos componentes de la web sin tener que recargar todo el documento HTML de nuevo. Por ejemplo en este proyecto se utiliza jQuery para insertar contenido en el cupón de apuestas y mostrar la línea de apuesta con la información de la cuota seleccionada por el usuario, de la misma manera utilizando jQuery se puede eliminar dicho contenido del cupón sin tener que recargar la vista al completo.

PhoneGap

PhoneGap [13]es un framework para el desarrollo de aplicaciones móviles apoyándose en tecnologías HTML, CSS y JavaScript. Las aplicaciones que se realizan son híbridas y no nativas por tanto pueden ser utilizadas en múltiples plataformas (Android, iOS, Blackberry OS) siendo esta una de las grandes ventajas de Phonegap, además contiene una serie de APIs para manejar la cámara, acelerómetro, GPS, notificaciones, ficheros y demás utilidades de los smartphones.

PhpUnit

PhpUnit [14] es un framework para PHP desarrollado por Sebastian Bergmann el cual facilita la creación de clases de prueba para realizar test sobre aplicaciones basadas en PHP.

2.2 Metodología

Para realizar el proyecto se sigue una metodología iterativa incremental. Esta metodología propone dividir el trabajo a realizar en sucesivas iteraciones, en las que se va suministrando al cliente funcionalidad adicional, esto ayuda al cumplimiento de los requisitos los cuales en ocasiones son cambiantes. Como se muestra en la Figura 1, cada requisito deberá ser terminado, documentado y se le realizarán las pruebas en una única iteración, verificando así que cumplen todos los objetivos para ser entregado al cliente, minimizando de este modo el riesgo de fallo en la entrega final del producto.

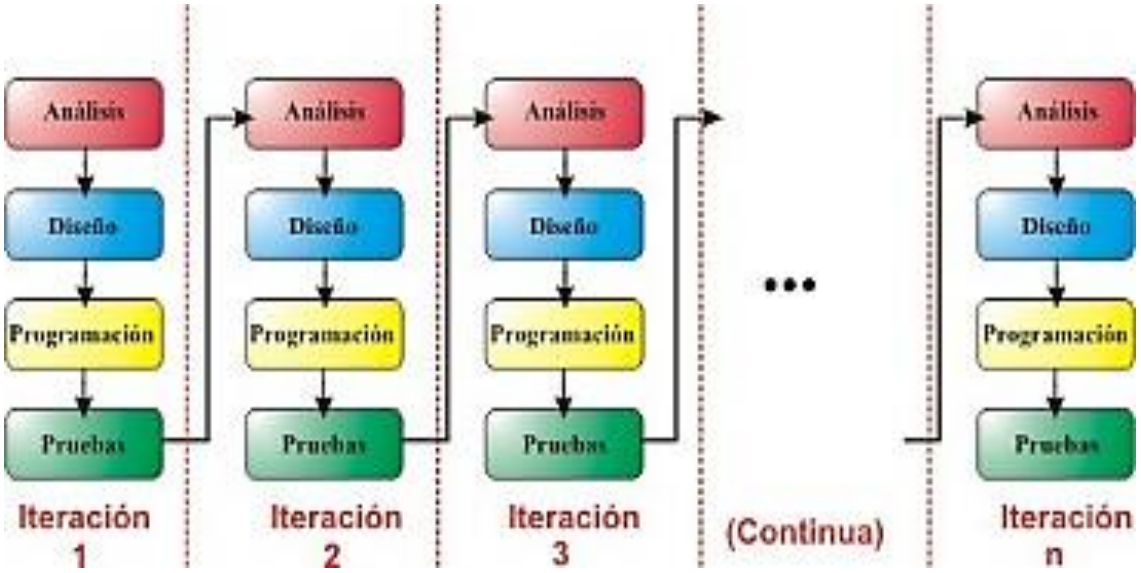


Figura 1 Ciclo de vida del desarrollo iterativo incremental [15]

3. ANÁLISIS DE REQUISITOS

3.1 Requisitos funcionales
3.2 Requisitos no funcionales

En este capítulo se analizan los requisitos a tener en cuenta para la realización de la aplicación. Los requisitos se detallan en dos apartados diferentes, uno para los requisitos funcionales y otro para los no funcionales.

3.1 Requisitos funcionales

En este apartado se describen los requisitos funcionales de la aplicación. "*Los requisitos funcionales de un sistema describen lo que el sistema debe hacer. Estos requisitos dependen del tipo de software que se desarrolle, de los posibles usuarios del software y del enfoque general tomado por la organización al redactar requisitos"* [16].

La aplicación será principalmente utilizada por tres actores principales, invitados, usuarios y administradores. El actor invitado tendrá la opción de registrarse en la aplicación y loguearse una vez registrado. El usuario, una vez logueado puede ver los eventos deportivos, filtrarlos por competición, crear y participar en grupos, proponer apuestas a sus grupos, ver su historial y sus pronósticos de eventos futuros. Además la aplicación tendrá un administrador que a través de la web de administración podrá añadir nuevas competiciones, nuevos eventos a las competiciones anteriormente creadas, crear y modificar las cuotas de los eventos, y establecer los resultados de los eventos que se hayan disputado para automatizar la comprobación de los cupones de apuestas ganadores.

Identificador	Descripción	Iteración
RF001	**El invitado quiere poder registrarse en la aplicación.**	**Iteración 1**
RF002	**El invitado quiere loguearse en la aplicación para acceder a su cuenta de usuario.**	**Iteración 1**
RF003	**El administrador quiere cambiar su contraseña**	**Iteración 1**
RF004	**El administrador quiere añadir nuevas competiciones deportivas a la aplicación.**	**Iteración 2**
RF005	**El administrador quiere añadir nuevos eventos deportivos a la aplicación.**	**Iteración 2**
RF006	**El administrador quiere modificar cuotas de eventos deportivos.**	**Iteración 2**
RF007	**El administrador quiere borrar un evento deportivo**	**Iteración 2**
RF008	**El administrador quiere borrar una competición deportiva**	**Iteración 2**
RF009	**El usuario quiere ver todos los eventos futuros y sus cuotas.**	**Iteración 3**
RF0010	**El usuario quiere filtrar los eventos futuros por competición deportiva.**	**Iteración 3**

RF0011	**El usuario quiere crear nuevos grupos de apuestas deportivas**	**Iteración 4**
RF0012	**El usuario quiere ver los grupos de apuestas que es participante.**	**Iteración 4**
RF0013	**El usuario quiere buscar usuarios y enviarlos invitaciones a los grupos que es participante.**	**Iteración 4**
RF0014	**El usuario quiere ver las invitaciones a grupos y aceptar o rechazarlas.**	**Iteración 4**
RF0015	**El usuario quiere publicar comentarios en los grupos que es participante.**	**Iteración 4**
RF0016	**El usuario quiere valorar los comentarios publicados en los grupos que es participante.**	**Iteración 4**
RF0017	**El usuario desea enviar cupones de apuestas de un evento deportivo a los grupos que participa.**	**Iteración 5**
RF0018	**El usuario quiere ver los cupones de apuestas propuestos y votar si se realiza o no la apuesta.**	**Iteración 5**
RF0019	**El administrador quiere actualizar los resultados de los eventos deportivos**	**Iteración 6**
RF0020	**El usuario desea ver la lista de apuestas realizadas así como un histórico de los cupones del grupo.**	**Iteración 6**

Tabla 1 Especificación de requisitos funcionales

A partir de los requisitos funcionales de la Tabla 1 se obtiene el siguiente modelo de casos de uso de la Figura 2.

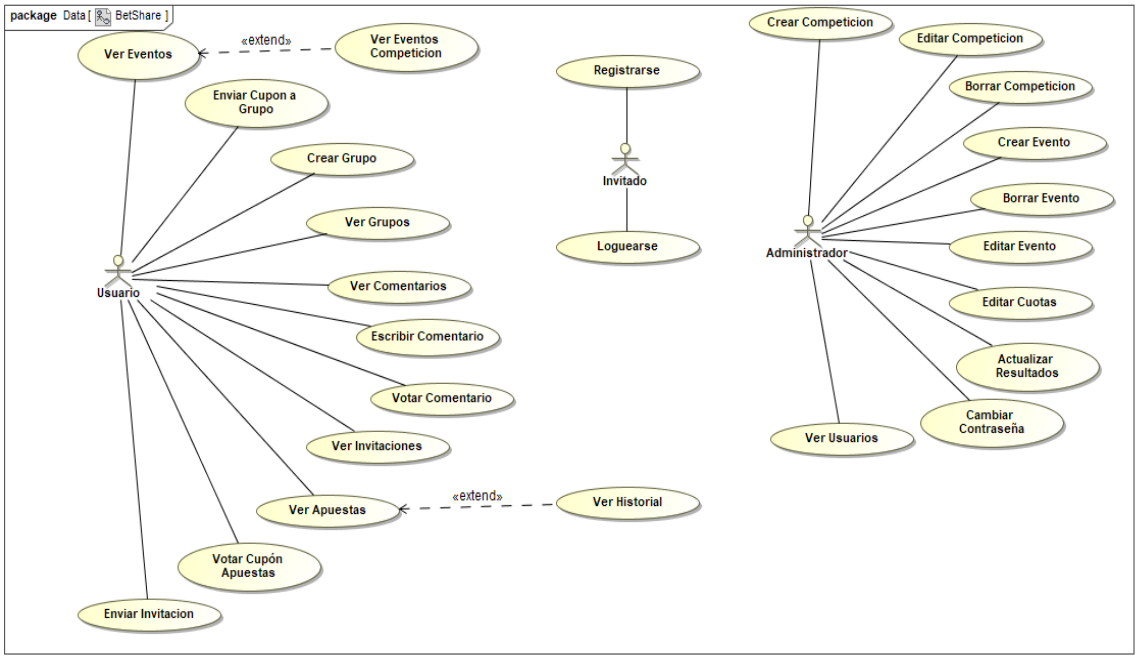


Figura 2 Diagrama de casos de uso

3.1.1 Requisitos funcionales para el invitado

Caso de uso: Registrarse

Actor:

Invitado: Desea registrarse en la aplicación para adquirir las capacidades de Usuario.

Precondiciones:

- La aplicación está instalada en el teléfono.
- Se tiene conexión a internet (3G, 4G, WIFI).

Postcondiciones:

La cuenta de Usuario creada será almacenada en la base de datos de la aplicación.

Escenario principal de éxito (o flujo básico):

1. El invitado pulsa el botón registrarse.
2. El usuario introduce el login, la contraseña, la fecha de nacimiento y el email en los apartados correspondientes.
3. El usuario pulsa el botón Aceptar.

Extensiones (flujos alternativos):

Punto 2 del escenario principal:

- El login está utilizado.
- La aplicación lo notifica y da la opción al usuario de cambiar el login.
- El usuario introduce otro login y se continúa con el punto 2 del escenario principal.

Punto 2 del escenario principal:

- La contraseña no cumple con condiciones de formato y seguridad de la aplicación.
- La aplicación notifica el formato requerido de la contraseña y le da la opción de volver a intentarlo.
- El usuario introduce otra contraseña y se vuelve al punto 2 del escenario principal.

Punto 2 del escenario principal:

- La fecha de nacimiento corresponde a una persona menor de 18 años.
- La aplicación notifica que no está permitido registrarse siendo menor de edad.
- El caso de uso termina.

Caso de uso: Loguearse

Actor:

Invitado: Desea acceder a su cuenta de usuario dentro de la aplicación.

Precondiciones:

- La aplicación está instalada en el teléfono.
- Se tiene conexión a internet (3G, 4G, WIFI).
- El usuario dispone de cuenta de usuario creada en el sistema.
- El usuario conoce su usuario y contraseña.

Postcondiciones:

El usuario está en el sistema logueado y puede acceder a los privilegios de su cuenta.

Escenario principal de éxito (o flujo básico):

1. El usuario inicia la aplicación.
2. El usuario introduce su usuario y password y pulsa el botón Loguearse.
3. La aplicación se conecta a la base de datos y valida la operación.
4. Los datos son correctos y el usuario accede a la aplicación con su perfil.

Extensiones (flujos alternativos):

Punto 1 del escenario principal:

- No se dispone de conexión a internet.
- La aplicación lo notifica y da la opción al usuario de activar los datos móviles.
- El usuario los activa y se continúa con el escenario principal.

Punto 2 del escenario principal:

- El usuario deja algún campo en blanco.
- La aplicación le notifica cuál de los campos está en blanco y le da la opción de volver a intentarlo.
- El usuario acepta y se vuele al punto 2 del escenario principal.

Punto 4 del escenario principal:

- El login y/o contraseña son incorrectos.
- La aplicación notifica el error y vuelve a la pantalla de login.
- Se vuelve al punto 2 del escenario principal.

3.1.2 Requisitos funcionales para el usuario

Caso de uso: Ver eventos por competición
Actor: **Usuario:** Desea ver los eventos de una competición determinada.
Precondiciones: • La aplicación está instalada en el teléfono. • Se tiene conexión a internet (3G, 4G, WIFI). • El usuario está logueado con su cuenta.
Postcondiciones: La aplicación muestra los eventos por disputar y sus cuotas de la competición seleccionada.
Escenario principal de éxito (o flujo básico): 1. El usuario pulsa el botón de acceso directo a eventos. 2. La aplicación muestra todos los eventos por disputar y sus cuotas. 3. El usuario pulsa la competición deseada el scroll horizontal de competiciones. 4. La aplicación muestra los eventos de la competición seleccionada.

Caso de uso: Crear grupo
Actor: **Usuario:** Desea crear nuevo grupo para realizar apuestas colaborativas.
Precondiciones: • La aplicación está instalada en el teléfono. • Se tiene conexión a internet (3G, 4G, WIFI). • El usuario está logueado con su cuenta.
Postcondiciones: El grupo se crea en el sistema y el usuario es el administrador del grupo.
Escenario principal de éxito (o flujo básico): 1. El usuario pulsa el botón grupos en el menú desplegable o botón de acceso directo a grupos. 2. El usuario pulsa crear en el scroll horizontal en la pantalla de grupos. 3. El usuario introduce el nombre del grupo, la imagen, la descripción y el protocolo de votación entre uno de los protocolos disponibles en la aplicación. 4. El usuario pulsa el botón Crear Grupo.
Extensiones (flujos alternativos): **Punto 3 del escenario principal:** • El nombre del grupo está utilizado. • La aplicación notifica el error con el mensaje “NOMBRE DE GRUPO EN USO, ELIGE OTRO NOMBRE”. • El usuario introduce otro nombre y se continúa con el punto 3 del escenario principal.

Punto 3 del escenario principal:

- El campo nombre o el campo descripción están vacíos.
- La aplicación notifica el error con el mensaje “RELLENA EL CAMPO NOMBRE Y DESCRIPCION”.
- El usuario rellena los campos y se vuelve al punto 3 del escenario principal.

Caso de uso: Enviar cupón a grupo
Actor: **Usuario:** Desea enviar un cupón de apuestas a un grupo del que es participante.
Precondiciones: • La aplicación está instalada en el teléfono. • Se tiene conexión a internet (3G, 4G, WIFI). • El usuario dispone de cuenta de usuario creada en el sistema. • El usuario es participante de algún grupo.
Postcondiciones: El cupón aparece en el apartado de Apuestas-En Votación dentro del grupo al que ha sido enviado.
Escenario principal de éxito (o flujo básico): 1. El usuario pulsa en la cuota de un evento deportivo y se abre el menú lateral derecho con el cupón de apuestas. 2. El usuario introduce el importe a apostar y pulsa el botón apostar. 3. La aplicación muestra los grupos que el usuario es participante. 4. El usuario pulsa en uno de los grupos para elegir el grupo donde se enviará el cupón para someterse a votación. 5. La aplicación muestra la pantalla de grupos.
Extensiones (flujos alternativos): **Punto 1 del escenario principal:** • El usuario introduce un importe menor que 0 y pulsa Apostar**.** • La aplicación no realiza nada a la espera de introducir una cantidad correcta. • El usuario introduce un importe mayor a 0 y pulsa el botón Apostar. • Se vuelve al punto 3 del escenario principal.

Caso de uso: Invitar usuario a grupo
Actor: **Usuario:** Desea invitar a un usuario a uno de los grupos en los que participa.
Precondiciones: • La aplicación está instalada en el teléfono. • Se tiene conexión a internet (3G, 4G, WIFI). • El usuario es participante en un grupo. • El usuario tiene en su lista un amigo que no es participante del grupo.
Postcondiciones: El usuario ha sido invitado y tiene en la pantalla Grupos-Invitaciones la invitación para entrar en el grupo.
Escenario principal de éxito (o flujo básico): 1. El usuario da al botón Grupos en el menú desplegable o en el botón de acceso directo a grupos. 2. El usuario pulsa el grupo en el que quiere invitar a su amigo dentro de la pantalla de Grupos. 3. El usuario pulsa el botón Añadir dentro de la pantalla del grupo seleccionado. 4. La aplicación muestra usuarios de la aplicación que no son miembros del grupo y permite filtrar usuarios introduciéndolo en la barra Buscar Usuario. 5. El usuario pulsa el usuario en los checkbox de los usuarios que quiere invitar y pulsa el botón Invitar. 6. Se envía las invitaciones a los usuarios seleccionados y la aplicación muestra la pantalla del grupo.

Caso de uso: Votar cupón
Actor: **Usuario:** Desea votar un cupón que esta propuesto para realizar apuesta dentro de un grupo.
Precondiciones: • La aplicación está instalada en el teléfono. • Se tiene conexión a internet (3G, 4G, WIFI). • El usuario dispone de cuenta de usuario creada en el sistema. • El usuario es participante de algún grupo. • El cupón está en el estado En Votación dentro de algún grupo.
Postcondiciones: El voto se ha añadido al contador de votos del cupón.
Escenario principal de éxito (o flujo básico): 1. El usuario pulsa Apuestas en el scroll horizontal dentro de un grupo. 2. El usuario pulsa En Votación dentro del scroll horizontal de las apuestas de grupo. 3. La aplicación muestra los cupones que están en votación con toda su información. 4. El usuario pulsa el botón Apostar o No Apostar. 5. El voto se suma al contador de votos en función del botón que hayamos pulsado (Apostar-No Apostar).

Extensiones (flujos alternativos):

Punto 3 del escenario principal:

- El usuario pulsa el botón Apostar o No Apostar**.**
- La aplicación no suma el voto y muestra la ventana de alerta "YA HAS VOTADO EN ESE CUPON".
- Se vuelve al punto 3 del escenario principal.

Caso de uso: Escribir comentario
Actor: **Usuario:** Desea escribir un comentario en el tablón Comentarios de un grupo.
Precondiciones: • La aplicación está instalada en el teléfono. • Se tiene conexión a internet (3G, 4G, WIFI). • El usuario dispone de cuenta de usuario creada en el sistema. • El usuario es participante de algún grupo.
Postcondiciones: El comentario se ha añadido al tablón de comentarios del grupo.
Escenario principal de éxito (o flujo básico): 1. El usuario pulsa Comentarios en el scroll horizontal dentro de un grupo. 2. El usuario introduce el comentario en el textbox *Escribe un comentario* y pulsa el botón Enviar. 3. La aplicación muestra los comentarios ordenados por fecha de publicación siendo el primero el último publicado.
Extensiones (flujos alternativos): **Punto 2 del escenario principal:** • No hay texto introducido al pulsar el botón enviar. • La aplicación no realiza nada pues no se ha introducido ningún comentario cuando se ha pulsado el botón Enviar**.** • Se vuelve al punto 2 del escenario principal.

3.1.3 Requisitos funcionales para el administrador

Caso de uso: Crear competición deportiva
Actor: **Administrador:** Quiere añadir una nueva competición a la aplicación.
Precondiciones: • La aplicación está instalada en el teléfono. • Se tiene conexión a internet (3G, 4G, WIFI). • El administrador está logueado con permisos de administrador en la aplicación.
Postcondiciones: La competición añadida por el administrador aparece en la lista de competiciones.
Escenario principal de éxito (o flujo básico): 1. El administrador da al botón Crear Competición en la botonera lateral dentro de Competiciones. 2. El administrador rellena el formulario indicando el nombre, el deporte dentro de la lista de deportes disponibles en la aplicación y una descripción de la misma. 3. El administrador el botón Crear Competición.
Extensiones (flujos alternativos): **Punto 3 del escenario principal:** • Ya existe una competición con ese nombre. • La aplicación notifica con un mensaje en rojo "El nombre introducido está en uso, prueba con otro". • Se vuelve al punto 2 del escenario principal.

Caso de uso: Crear evento deportivo
Actor: **Administrador:** Quiere añadir nuevos eventos deportivos a la aplicación.
Precondiciones: • La aplicación está instalada en el teléfono. • Se tiene conexión a internet (3G, 4G, WIFI). • El administrador está logueado con permisos de administrador en la aplicación.
Postcondiciones: El evento deportivo creado por el administrador aparece en la competición correspondiente.
Escenario principal de éxito (o flujo básico): 1. El administrador da al botón Crear evento en la botonera lateral dentro de Eventos. 2. El administrador rellena el formulario indicando el local, visitante, la competición dentro de las competiciones disponibles, la fecha, la cuota de ganador local, empate y visitante, así como una descripción del evento. 3. El administrador el botón Crear Evento.

Extensiones (flujos alternativos):

Punto 3 del escenario principal:

- Algunos de los campos están vacíos.
- La aplicación notifica con un mensaje en rojo que algún campo requerido está vacío indicando cuales.
- Se vuelve al punto 2 del escenario principal, el administrador introduce los datos requeridos que faltan y se pulsa Crear Evento.

Caso de uso: Editar evento deportivo
Actor: **Administrador:** Quiere editar un evento deportivo a la aplicación.
Precondiciones: • La aplicación está instalada en el teléfono. • Se tiene conexión a internet (3G, 4G, WIFI). • El administrador está logueado con permisos de administrador en la aplicación.
Postcondiciones: El evento deportivo editado por el administrador aparece con sus datos actualizados en los eventos de la aplicación.
Escenario principal de éxito (o flujo básico): 1. El administrador da al botón Ver Eventos en la botonera lateral dentro de Eventos. 2. Pulsa el botón de edición con icono de lápiz del evento que quiera editar. 3. La aplicación muestra el formulario de edición. 4. El administrador modifica en el formulario cualquiera de los campos local, visitante, la competición dentro de las competiciones disponibles, la fecha, la cuota de ganador local, empate y visitante, así como una descripción del evento. 5. El administrador pulsa el botón Editar Evento.
Extensiones (flujos alternativos): **Punto 3 del escenario principal:** • Algunos de los campos están vacíos. • La aplicación notifica con un mensaje en rojo que algún campo requerido está vacío indicando cuales. • Se vuelve al punto 2 del escenario principal, el administrador introduce los datos requeridos que faltan y se pulsa Crear Evento.

Caso de uso: Borrar evento deportivo
Actor: **Administrador:** Quiere borrar un evento deportivo a la aplicación.
Precondiciones: • La aplicación está instalada en el teléfono. • Se tiene conexión a internet (3G, 4G, WIFI). • El administrador está logueado con permisos de administrador en la aplicación.
Postcondiciones: El evento deportivo borrado por el administrador no aparece en los eventos de la aplicación.
Escenario principal de éxito (o flujo básico): 1. El administrador da al botón Ver Eventos en la botonera lateral dentro de Eventos. 2. Pulsa el botón de borrado con icono de papelera del evento que quiera borrar. 3. La aplicación muestra recarga la lista de eventos sin el evento borrado.
Extensiones (flujos alternativos): **Punto 2 del escenario principal:** • El evento que queremos borrar y sus cuotas tienen cupones de apuestas en la aplicación. • La aplicación notifica con un mensaje que no es posible borrar ese evento. • Se vuelve al punto 1 del escenario principal, el administrador introduce los datos requeridos que faltan y se pulsa Crear Evento.

Caso de uso: Modificar cuotas de evento deportivo
Actor: **Administrador:** Quiere modificar cuotas de eventos deportivos
Precondiciones: • La aplicación está instalada en el teléfono. • Se tiene conexión a internet (3G, 4G, WIFI). • El administrador está logueado con permisos de administrador en la aplicación.
Postcondiciones: El evento deportivo modificado por el administrador aparece en la sección del deporte correspondiente con sus nuevas cuotas.
Escenario principal de éxito (o flujo básico): 1. El administrador busca el evento que quiere modificar en la sección Ver Eventos. 2. La aplicación muestra todos los eventos. 3. El administrador pulsa en el botón Cuotas del evento que quiere modificar las cuotas. 4. La aplicación muestra las cuotas y la descripción de las mismas. 5. El administrador pulsa el botón de edición en la cuota que quiera editar. 6. La aplicación muestra el formulario de edición. 7. El administrador introduce la nueva Cuota y pulsa el botón Editar Cuota. 8. La aplicación muestra las cuotas descripción de las mismas con los cambios realizados.

Extensiones (flujos alternativos):

- El administrador busca el evento que quiere modificar en la sección Ver Eventos de la competición correspondiente.
- La aplicación muestra los eventos de la competición seleccionada.
- El administrador pulsa en el botón Cuotas del evento que quiere modificar las cuotas.
- La aplicación muestra las cuotas y la descripción de las mismas.
- El administrador pulsa el botón de edición en la cuota que quiera editar.
- La aplicación muestra el formulario de edición.
- El administrador introduce la nueva Cuota y pulsa el botón Editar Cuota.
- La aplicación muestra las cuotas descripción de las mismas con los cambios realizados.

Caso de uso: Actualizar resultado de evento
Actor: **Administrador:** Quiere actualizar el resultado de un evento deportivo que se ha disputado.
Precondiciones: • La aplicación está instalada en el teléfono. • Se tiene conexión a internet (3G, 4G, WIFI). • El administrador está logueado con permisos de administrador en la aplicación.
Postcondiciones: El evento pasa a ser un evento pasado, se fija el resultado, las cuotas ganadoras y perdedoras.
Escenario principal de éxito (o flujo básico): 1. El administrador pulsa Actualizar Resultados en el apartado Resultados de la botonera lateral. 2. La aplicación muestra los eventos que están pendientes de disputarse. 3. El administrador pulsa en el botón 1 X 2 para dictaminar el resultado del evento que quiera actualizar el resultado. 4. La aplicación muestra los eventos que todavía no se han disputado.

3.2 Requisitos no funcionales

Los requisitos no funcionales son propiedades o cualidades que el sistema debe tener y que se diferencian de comportamientos específicos del sistema los cuales son detallados en los requisitos funcionales.

Estos requisitos no funcionales imponen restricciones o necesidades de la aplicación en términos de apariencia, de usabilidad, seguridad y obligaciones legales dado el sector de la aplicación, así como requisitos hardware para poder ejecutar la aplicación.

3.2.1 Usabilidad y apariencia

La Organización Internacional para la Estandarización [17] define la usabilidad como "*la medida en la que un producto se puede usar por determinados usuarios para conseguir objetivos específicos con efectividad, eficiencia y satisfacción en un contexto de uso especificado".*

Es decir estos requisitos establecen cuando una interfaz es suficientemente buena y si es fácil su aprendizaje para el usuario. Se puede medir tanto objetivamente, midiendo el tiempo que tarda el usuario en realizar las operaciones a través de la aplicación, como de forma subjetiva, es decir mediante encuestas y opiniones de los usuarios. Por todo ello, será necesario realizar pronósticos de forma eficiente y eficaz, sin confusiones dentro de la aplicación.

Por último en términos de apariencia el requisito principal es que el color predominante sea el azul celeste, puesto que es el color característico de nuestra casa de apuestas y que nos diferencia de la competencia.

3.2.2 Requisitos Hardware

Nuestra aplicación para dispositivos móviles requerirá de un Smartphone y como se realizan apuestas online es necesario conexión a internet, ya sea por 3G, 4G o WIFI para acceder a la aplicación puesto que es 100% online.

3.2.3 Requisitos legales

Nuestro sistema almacena datos personales de los usuarios como emails, fechas de nacimiento y contraseñas, por tanto, el sistema debe cumplir con el Reglamento Europeo de Protección de Datos de 27 de abril de 2016 y que entró en vigor en España el 25 de mayo de 2018 [18].

3.2.4 Requisitos de seguridad

El sistema cumple con los controles de acceso al sistema y con los tres principios de la seguridad, la confidencialidad, la información manejada por el sistema está protegida de acceso no autorizado y cifrada para el caso de las contraseñas. El segundo de los principios con los que cumple la aplicación es la integridad, no permitiendo al usuario introducir datos corruptos o inconsistentes mediante mecanismos de chequeo de datos y formularios de CodeIgniter.

El último de los principios es la disponibilidad, los usuarios autorizados tendrán acceso a la información y los mecanismos de seguridad no ocultarán dicha información cuando el usuario la solicite.

4. DISEÑO E IMPLEMENTACIÓN

4.1 Diseño arquitectónico
4.2 Diseño de la app
4.3 Diseño de la web

En este capítulo se explican el diseño y la implementación del sistema, consta de tres partes, el diseño arquitectónico de la aplicación, el diseño de la app para smartphones y el diseño de la página web. La implementación aplica las técnicas necesarias para llevar a cabo el desarrollo físico del sistema. A continuación se explican las técnicas de diseño arquitectónico aplicadas a nuestro sistema para posteriormente, explicar detalladamente el diseño e implementación de la aplicación para smartphones y de la página web.

4.1 Diseño arquitectónico

El sistema está diseñado bajo una arquitectura de tres capas, las tres capas diferencian la lógica del sistema en capa de presentación, capa de negocio y capa de persistencia. Cada capa tiene una función muy diferenciada del resto, se explica a continuación en qué consisten:

- **Capa de presentación**: es la capa que permite al sistema interactuar con el usuario, se comunica con él aportando datos y solicitándoselos cuando sea necesario. Esta capa se comunica únicamente con la capa de negocio.

- **Capa de negocio**: es la capa donde residen las funciones que se ejecutan dentro del sistema, hace de intermediaria entre la capa de presentación y la de datos. Se encarga de recibir las peticiones del usuario y mostrar los resultados a través de la capa de presentación. Si es necesario almacena/obtiene los datos en la capa de persistencia.

- **Capa de persistencia**: es donde se almacenan los datos y es la capa encargada de acceder a los mismos. Está formada por uno o más gestores de bases de datos localizados en uno o varios servidores, que realizan el almacenamiento de datos, reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio.

4.2 Diseño e implementación la app

En este apartado se explica el diseño y la implementación de la aplicación para Smartphones, en concreto para el sistema operativo Android. Consta de tres apartados acorde con las tres capas del diseño arquitectónico, por tanto se explica por separado el diseño e implementación de la capa de persistencia, negocio y presentación.

4.2.1 Diseño e implementación de la capa de persistencia

En este apartado se va a explicar el diseño de la capa de persistencia, para ello partimos del diagrama Entidad-Relación que se muestra en la Figura 3, donde se puede observar el modelo conceptual de la capa de persistencia.

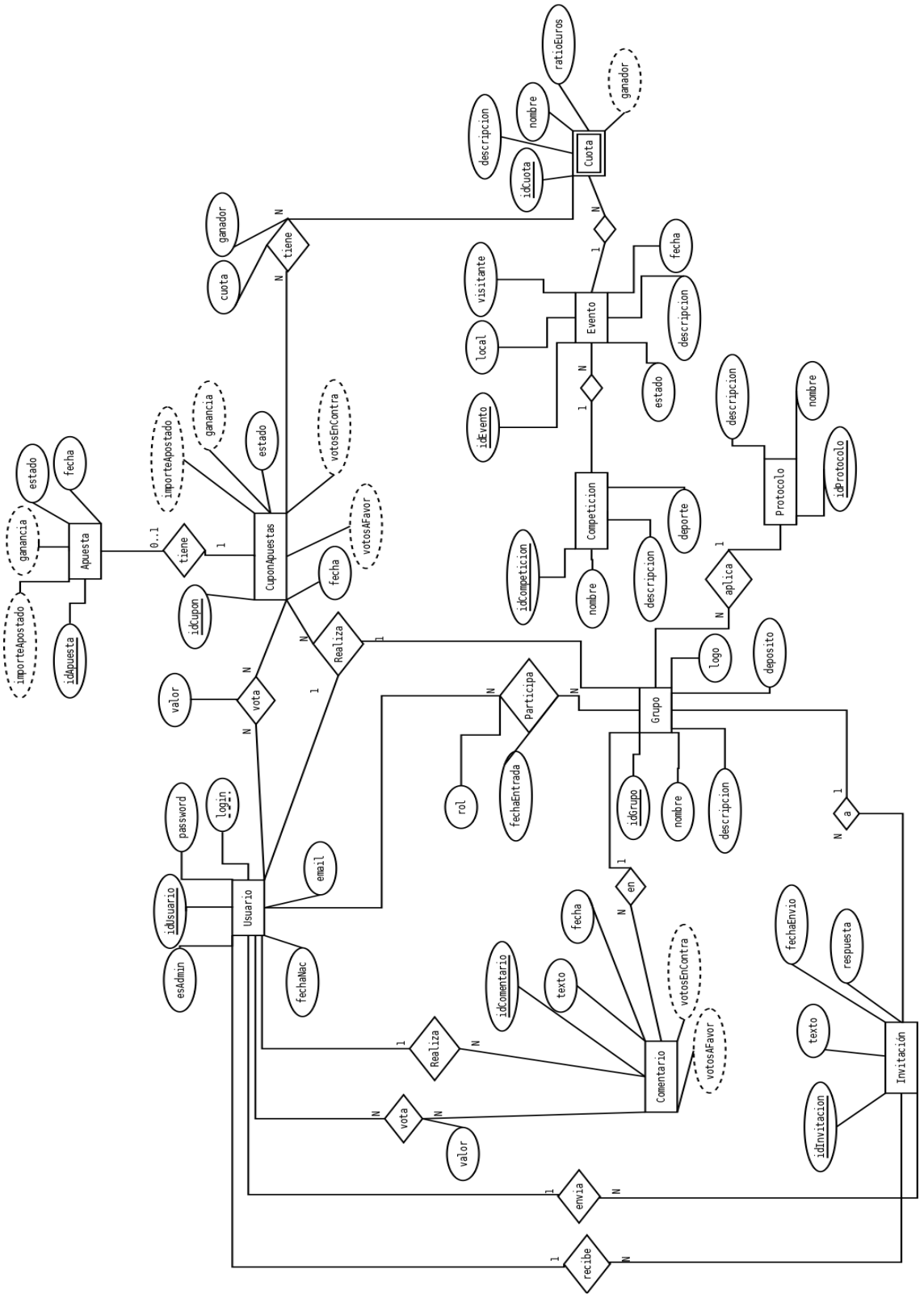


Figura 3 Modelo Entidad-Relación

A partir de este diagrama Entidad-Relación construimos las tablas de la base datos, como se aprecia en la Figura 4 un USUARIO tiene un id, login, password, email, fecha de nacimiento y un booleano para saber si es Administrador de la aplicación. El USUARIO se relaciona con GRUPO a través de la tabla *participa*, permitiendo una relación N-N entre ellos, teniendo también una marca de tiempo para saber cuando el USUARIO empezó a formar parte del GRUPO y su rol (ADMIN, USER) dentro del mismo.

La *USUARIO* puede enviar y recibir INVITACIONES a grupo, es decir una INVITACION contiene el ID del usuario que envía la invitación, el ID del que la recibe, así como el GRUPO al que pertenece la INVITACION, además como una marca de tiempo, un mensaje y la respuesta. Cada GRUPO, tiene un id, nombre, descripción, un logotipo, un depósito de dinero con el que realizar apuestas y un PROTOCOLO que será el que dictaminará la política de votaciones y apuestas. El PROTOCOLO tiene el id, el nombre y la descripción. La tabla COMENTARIO recoge el id del usuario y el id del grupo, además del texto del comentario, la marca de tiempo, los votos a favor y en contra. Para tener un control sobre los votos, tenemos la tabla VOTOS_COMENTARIO que recoge el usuario, el comentario y el valor del voto (POSITIVO, NEGATIVO) en nuestro caso.

Nuestra aplicación utiliza la tabla COMPETICION para almacenar las diferentes competiciones deportivas con su id, nombre, descripción y el deporte. La tabla EVENTO contiene el id, el equipo local, el visitante, la fecha, una descripción, la competición y un estado que indica si el evento es FUTURO o PASADO. Por último cada evento tiene diversas cuotas a las que los usuarios podrán apostar, cada CUOTA consta de un id, un nombre (1,X,2), una descripción, el ratioEuros es decir la cuota en sí, lo que se paga sobre euro apostado, además del id del evento y un booleano *ganador* que indicará si la cuota es ganadora en función del resultado del EVENTO.

Cuando un usuario pulsa en una CUOTA y la añade al CUPON, se crea una entrada en la tabla CUPONCUOTA permitiendo una relación N-N entre CUOTA y CUPON, esta tabla además almacena si es ganador y el valor cuota es decir el ratioEuros de la tabla CUOTA. Esto se explica pues una CUOTA puede variar sus cantidades con el paso del tiempo, de este modo almacenamos el valor que tenía la CUOTA en el momento de realizar el CUPON. Una vez el usuario introduce el *importe* a apostar, se calcula la *ganancia* potencial y el usuario lo envía a algún grupo, se crea el CUPON que tiene un id, el id del grupo y del usuario que lo envía, el importe apostado, la ganancia, la fecha, los votos a favor, en contra y el estado (En Votación, En Curso, Ganado, Perdido) en que se encuentra el CUPON.

La mecánica de votación es idéntica a la de los comentarios, nos apoyamos en la tabla VOTOS_CUPON. Por último, si se cumplen las condiciones de voto en función del protocolo del grupo para la realización de apuestas, se crea la APUESTA, que almacena el id, el importe apostado, la ganancia, la marca de tiempo, el id del cupón y del grupo, además del estado (Ganada, Perdida) para realizar el pago de la ganancia en caso afirmativo.

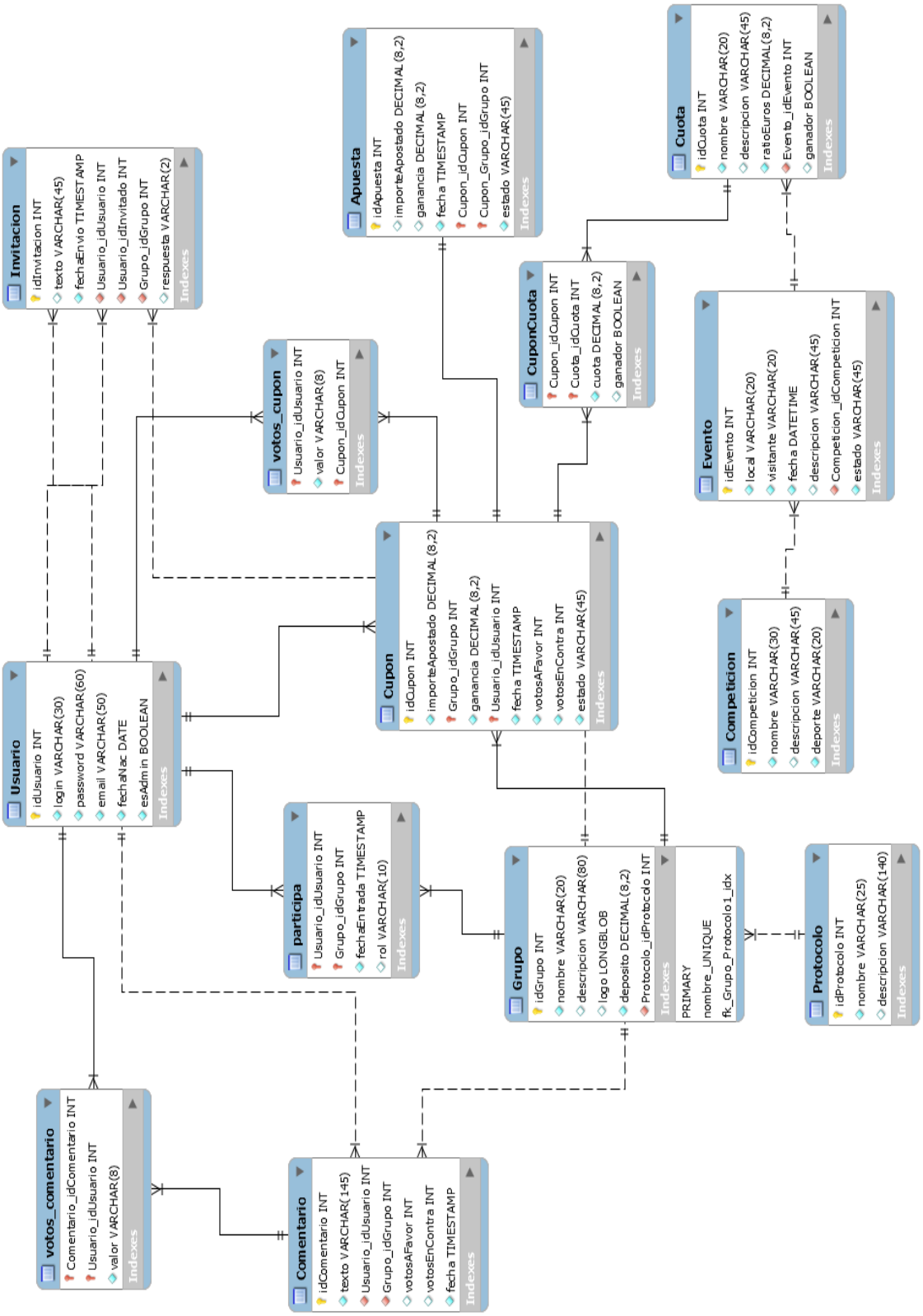


Figura 4 Diseño de la base de datos

4.2.2 Diseño e implementación de la capa de negocio

En este apartado se tratará el diseño e implementación de la capa de negocio de la aplicación móvil. Como apreciamos en la Figura 5, la aplicación contiene tres hojas de estilos CSS. La primera *style.css* contiene estilos comunes a toda la aplicación, el segundo *logueado.css* contiene estilos para las vistas una vez estamos logueados en la aplicación, como pueden ser los dos menús laterales o la presentación de los eventos. Por último *grupos.css* contiene estilos para el apartado de grupos de la aplicación, el tablón de comentarios y los cupones de apuestas con sus correspondientes votaciones. A parte de los mencionados en el párrafo anterior, la aplicación contiene las hojas de estilo *bootstrap.css* y *font-awesome.css* correspondientes a ambos frameworks de estilos CSS.

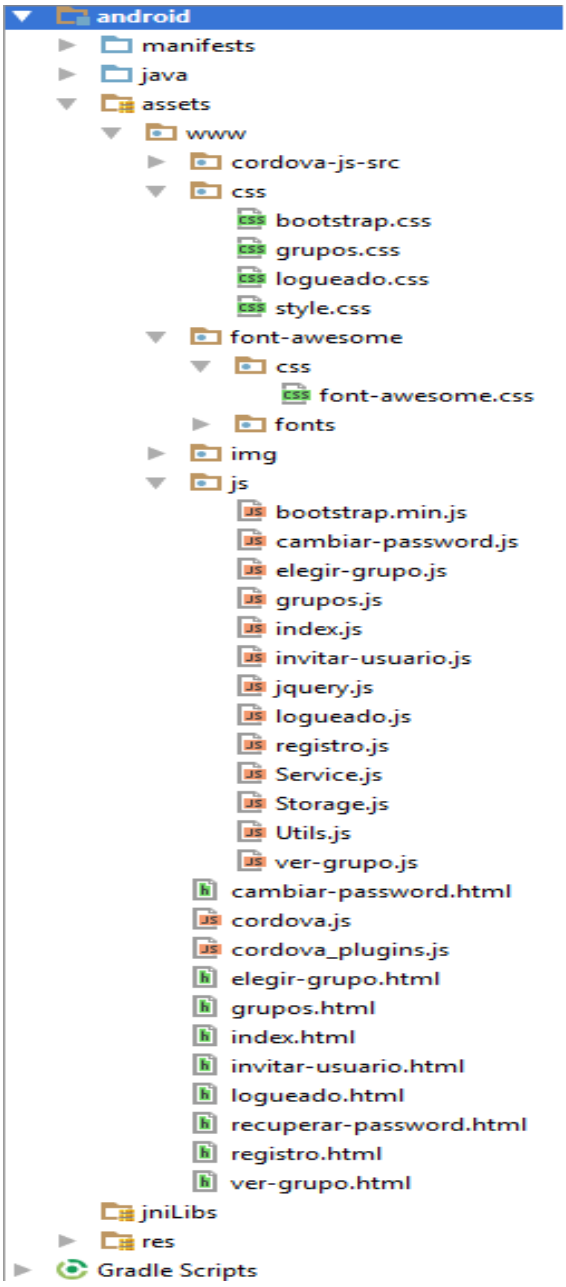


Figura 5 Estructura de ficheros de la aplicación Android

La carpeta js contiene todos los archivos JavaScript de la aplicación. Cada vista HMTL tiene un JavaScript asociado con las funciones que se ejecutarán en cada una de ellas. Además de tres clases JajaScript adicionales, *Utils* con funciones auxiliares, *Storage* que contiene funciones para almacenar, obtener y borrar tanto usuarios como datos en el almacenamiento local del teléfono y *Service.* La clase *Service* es la encargada de realizar el intercambio de datos entre las vistas y los controladores. Las vistas ejecutan funciones Javascript y estas pasan datos a la clase *Service* para ejecutar alguna de sus funciones. Las funciones del servicio envían datos a una URL, estas URLs son interpretadas por el archivo *routes.php* de Codeigniter que realizar la traducción de URL a función de un controlador en Codeigniter. Vamos a explicarlo a continuación mediante un ejemplo para el caso del registro de nuevo usuario, el usuario pulsa el botón Registrarse y se ejecuta la función doRegistrarse() cuyo código se puede apreciar en la Figura 6.

```
<button onclick="doRegistrarse()" class="btn btn-theme btn-block" name="registrarse">
    Registrarse
</button>
```

Figura 6 Vista registro de nuevo usuario

```
function doRegistrarse(){
    //Cogemos los valores del formulario
    var login = $("#login").val();
    var password = $("#password").val();
    var repetirpassword = $("#repetirpassword").val();
    var email = $("#email").val();
    var fechaNac = $("#fechaNac").val();

    if (isEmpty(login) || isEmpty(password) || isEmpty(repetirpassword)|| isEmpty(email)|| isEmpty(fechaNac)){
        $("#error").remove();
        $("#formulario-login").prepend("<h3 id='error' align='left' style='color:red;font-size:12px;'>Hay campos vacios</h3>");
        return;
    }

    if (password != repetirpassword){
        $("#error").remove();
        $("#formulario-login").prepend("<h3 id='error' align='left' style='color:red;font-size:12px;'>error: Password no coinciden</h3>");
        return;
    }
    //Llamamos al servicio
    var result = Service.doRegistrarse(login,password, repetirpassword, email, fechaNac);

    if (result.data.length == 0){
        $("#error").remove();
        $("#formulario-login").prepend("<h3 id='error' align='left' style='color:red;font-size:12px;'>Usuario no registrado</h3>");
        return;
    }

    window.location = "index.html";
}
```

Figura 7 Función doRegistrarse() de la clase registro.js

Como vemos en la Figura 7, hay una llamada al servicio en la que se pasan los datos necesarios para el registro a la función de registro de nuevo usuario del servicio. Para explicar la clase *Service* primero vamos a explicar su función sendData, en la que enviamos utilizando AJAX un objeto JSON a una URL pasados ambos como parámetros como se puede apreciar en la Figura 8.

```
sendData: function sendData(data,url,sync){
    var resultado;
    $.ajax({
        type:"POST",
        contentType:"application/json",
        url:url,
        data:data,
        async: sync,
        cache: false,
        beforeSend: function(){},
        success:function(data){
            console.log(data);
            resultado = data;
        },
        error:function(XMLHttpRequest, textStatus, errorThrown){
            alert("Error:"+errorThrown);
        }
    });

    return resultado;
},
```

Figura 8 Función sendData de la clase Service.js

Esta función sendData la utilizan el resto de funciones del servicio como se ve en la Figura 9 para el caso del registro de usuario.

```
doRegistrarse : function doRegistrarse (login,password,repetirpassword,email,fechaNac){
    var data = JSON.stringify({"login":login,"password":password,"repetirpassword":repetirpassword,
                               "email":email,"fechaNac":fechaNac});
    return this.sendData(data,this.url+"/api/registrarse",false);
},
```

Figura 9 Función doRegistrarse de la clase Service.js

Con esto ya se ha enviado el objeto JSON con los datos introducidos por el usuario en el formulario, a la dirección introducida como parámetro. Esta dirección es la del servidor de aplicación donde tenemos las clases de negocio de la aplicación bajo el framework CodeIgniter. En concreto, la dirección corresponde con el registro de usuario, ahora vamos a ver en la Figura 10 como Codeigniter traduce de forma muy sencilla esa dirección a través del archivo de configuración *routes.php.*

```
$route['api/registrarse'] = "api/user_controller/doRegistrarse";
```

Figura 10 Traducción de URLs en routes.php de CodeIgniter

Todas las rutas conducen a la carpeta controllers, en CodeIgniter la parte más importante de la parte de negocio son los controladores, estos contienen funciones, realizan operaciones y hacen peticiones a los modelos para obtener datos de la base de datos. En este proyecto, dentro de controladores tenemos por separado los propios de la web de administración con las operaciones de administración y los controladores de la aplicación móvil con las funciones que puede realizar el usuario, se puede observar en la Figura 11.

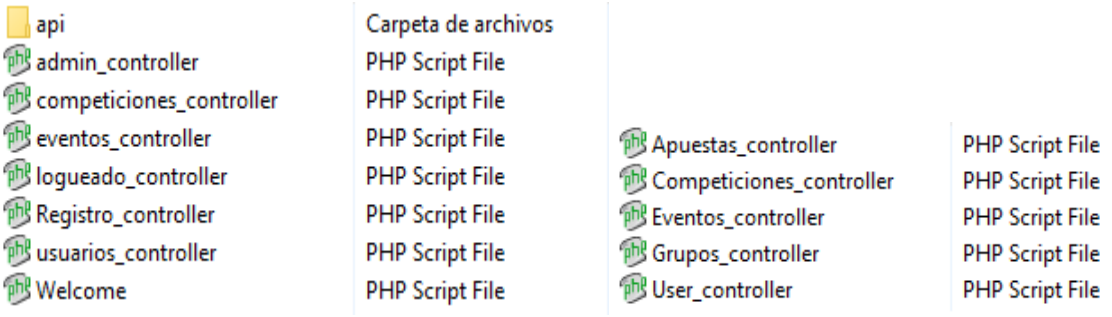


Figura 11 Carpeta controllers (pantalla izquierda) - Carpeta api (pantalla derecha)

Como se ha visto en la Figura 10, la ruta apunta a la carpeta api donde están los controladores de la aplicación, en concreto a la función *doRegistrarse()* del controlador *user_controller*. El código de la función es la que vemos en la Figura 12.

```php
public function doRegistrarse(){
    $data = $this->read_json();

    $login = $data['login'];
    $password = $data['password'];
    $repetirpassword = $data['repetirpassword'];
    $email = $data['email'];
    $fechaNac =$data['fechaNac'];

    if (empty($login) || empty($password)|| empty($repetirpassword)|| empty($email)|| empty($fechaNac)){
        $this->response_error("001");
    }

    //Funciones de comprobación de datos
    if ($password != $repetirpassword){
        $this->response_error("004");
    }
    $result = $this->checkLogin($login);
    if ($result === FALSE){
        $this->response_error("005");
    }
    $result = $this->checkEmail($email);
    if ($result === FALSE){
        $this->response_error("006");
    }
    $result = $this->checkFechaNac($fechaNac);
    if ($result === FALSE){
        $this->response_error("No es mayor de edad, no puede registrarse");
    }
    //Cargamos el modelo y ejecutamos la función de registro de usuario
    $this->load->model('Usuario_model','',TRUE);
    if($this->Usuario_model->registro_usuario($login,$password,$repetirpassword,$email,$fechaNac)== TRUE){
        //Construimos la respuesta con los datos
        $response['login'] = $login;
        $response['email'] = $email;
        $response['fechaNac'] = $fechaNac;

        $this->response_ok($response);
    }
    else{
        $this->response_error("20");    //Error al persistir en la base de datos
    }

}
```

Figura 12 Código de la función de registro de usuario del controlador User_controller

Como se ve en la Figura 12, se realiza la lectura del objeto JSON, se comprueba que los datos cumplen con las reglas de negocio para el registro, es decir que el login y el email no estén en uso, además de que el usuario sea mayor de edad como hemos visto en el capítulo de análisis de requisitos, en el apartado [3.1.1]. Posteriormente se procede a cargar el modelo, ejecutar la función del modelo y retornar la respuesta correcta o el mensaje de error si se ha producido algún error.

Los modelos en CodeIgniter realizan la interacción con la base de datos mediante sentencias SQL y querys, como vemos en Figura 13.

```php
function registro_usuario($user = '',$password='',$repassword='',$email='',$fechaNac=''){
    //Comprobamos que no haya campos vacios
    if ($user == '' && $password == '' && $repassword == '' && $email == '' && $fechaNac == ''){
        return FALSE;
    }
    //Comprobado en el controlador, volvemos a comprobar para mayor seguridad
    if($this->comprobar_edad($fechaNac) == FALSE){
        //Es menor de edad, no se puede registrar
        return FALSE;
    }
    //Comprobado en el controlador, volvemos a comprobar para mayor seguridad
    if(trim($password) != trim($repassword)){
        return FALSE;
    }

    //ESCAPAMOS PARA EL SQL INJECTION
    $password = sha1($password); //CODIFICAMOS ShA1
    $user = $this->db->escape($user);
    $password = $this->db->escape($password);
    $email = $this->db->escape($email);
    $fechaNac= $this->db->escape(date("Y-m-d",strtotime($fechaNac))); //MYSQL format Y-m-d

    $sql = "SELECT * FROM Usuario WHERE login=$user OR email = $email LIMIT 1";
    $query = $this->db->query($sql);
    if ($query->num_rows() > 0){
        //El usuario y/o el email ya están en la base de datos,no se puede registrar con ese login, email
        return FALSE;
    }
    //Se insertan los datos a la base de datos y el usuario ya fue registrado con exito.
    $sql = "INSERT INTO Usuario (login,password,email,fechaNac,esAdmin) VALUES ($user,$password,$email,$fechaNac, 0)";
    $query = $this->db->query($sql);

    //Se ha registrado el usuario
    return TRUE;
}
```

Figura 13 Función registro_usuario del modelo usuario_model.php

En este caso en concreto, se vuelve a realizar una comprobación de los parámetros recibidos, se codifica la contraseña y se escapan los parámetros para su posterior inserción en la base de datos. La configuración de la conexión a la base de datos, CodeIgniter la realiza en el archivo de configuración *database.php* que se muestra en la Figura 14.

```php
$db['default'] = array(
        'dsn'   => '',
        'hostname' => 'h1128.dinaserver.com',
        'username' => 'proyecto_mv',
        'password' => 'jrQxAV19',
        'database' => 'proyecto_mv',
        'dbdriver' => 'mysqli',
        'dbprefix' => '',
        'pconnect' => FALSE,
        'db_debug' => (ENVIRONMENT !== 'production'),
        'cache_on' => FALSE,
        'cachedir' => '',
        'char_set' => 'utf8',
        'dbcollat' => 'utf8_general_ci',
        'swap_pre' => '',
        'encrypt' => FALSE,
        'compress' => FALSE,
        'stricton' => FALSE,
        'failover' => array(),
        'save_queries' => TRUE
);
```

Figura 14 Configuración de conexión a la base de datos en el fichero database.php

4.2.3 Diseño e implementación de la capa de presentación

En este apartado explicará el diseño e implementación de la capa de presentación detallando cada una de las vistas de la aplicación mediante diagramas y capturas. Se explica la navegabilidad por las vistas de la aplicación en la Figura 15.

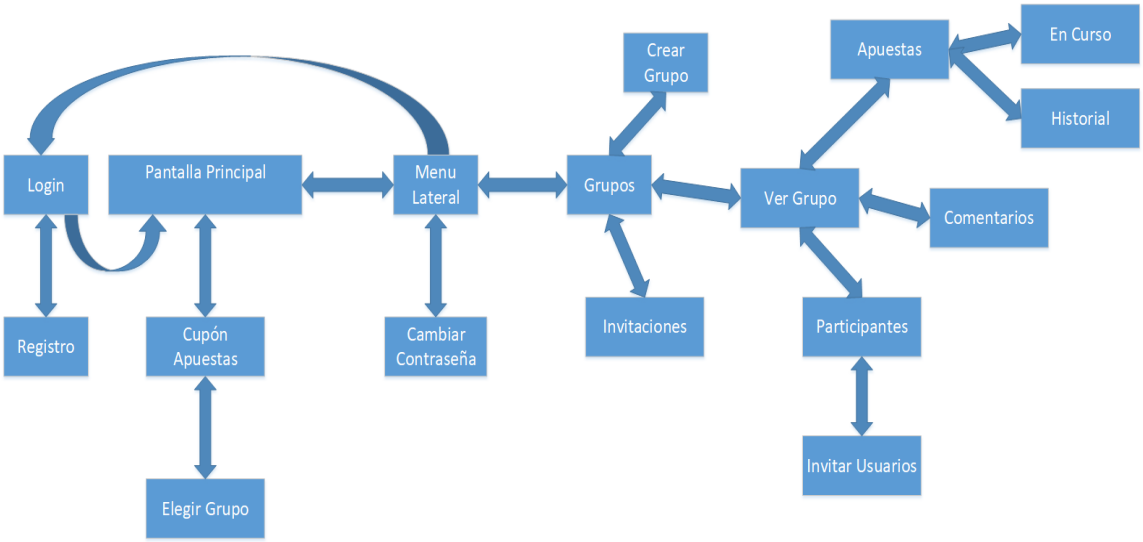


Figura 15 Vistas de la aplicación móvil

Ahora se va a detallar las vistas del diagrama anterior mediante capturas de pantalla y una breve explicación de su funcionalidad, así como la implementación de algunas de ellas.

(1) – Pantalla Login y Registro

La aplicación parte de una pantalla de Login donde el usuario deberá introducir su login y su password y presionar el botón loguearse. Este formulario comprueba que el login y el password son correctos y se carga la vista logueado, que corresponde a la pantalla principal. Si pulsamos en el enlace “Crear nueva cuenta de Usuario” nos muestra el formulario de registro donde el invitado deberá introducir el login, el password, repetir el password, un email y su fecha de nacimiento y pulsar el botón registrarse. La apariencia de ambas vistas se pueden ver en la Figura 16.

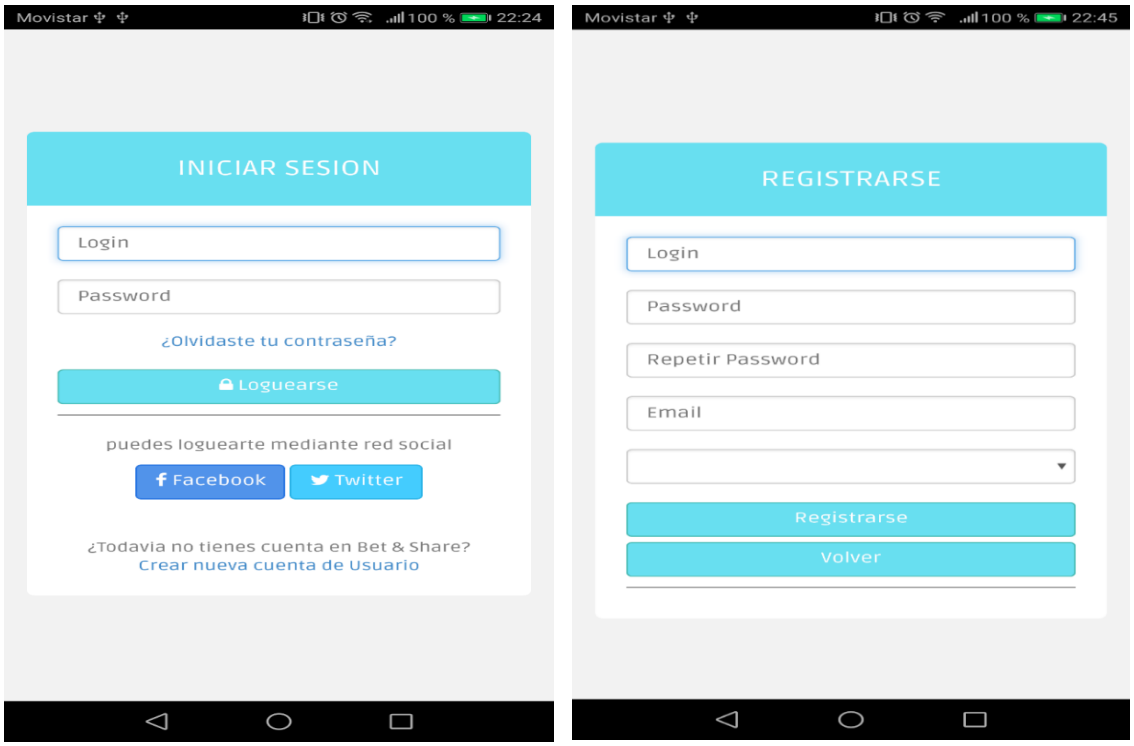


Figura 16 Pantalla de login (izquierda) - Pantalla de registro (derecha)

(2) – Pantalla principal- Logueado

La pantalla principal una vez logueado muestra todos los eventos por disputar, como se ve en la Figura 17 contiene información de los eventos y sus cuotas. Esta interfaz de usuario contiene un scroll horizontal donde se muestran todas las competiciones de la aplicación. Pulsando en una competición se realiza el filtrado para solo mostrar los eventos de la misma.

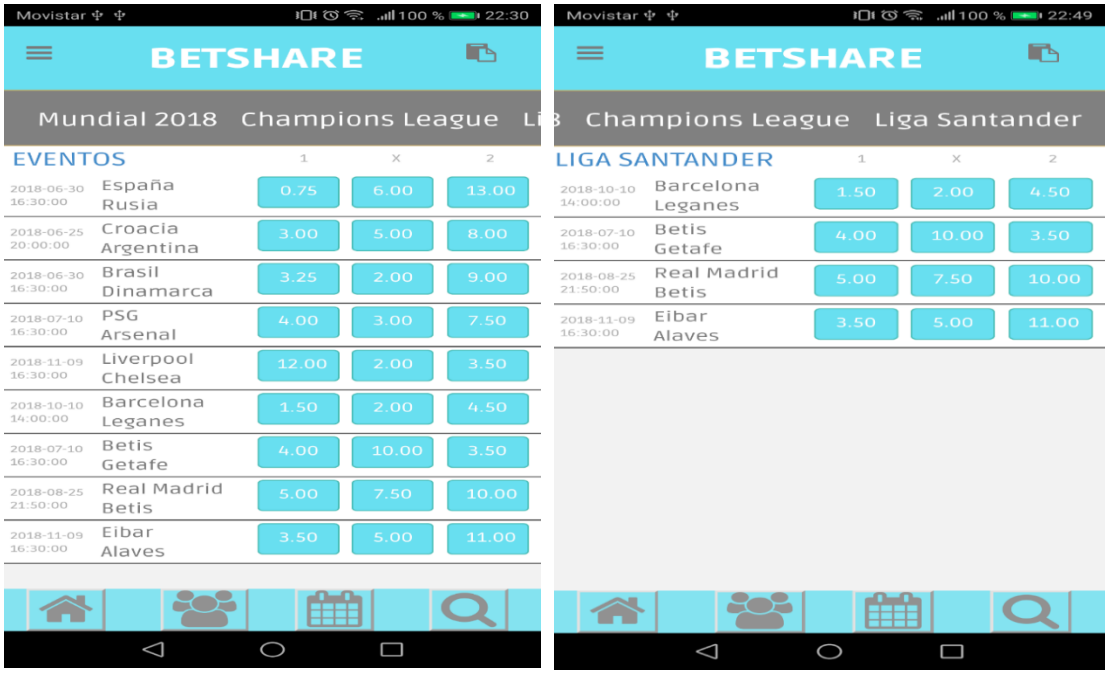


Figura 17 Logueado donde vemos los eventos (izquierda) - Filtrado de eventos por competición (derecha)

A continuación se va a explicar la implementación del filtrado de eventos, la pantalla logueado muestra por defecto todos los eventos por disputar de BETSHARE. Se utiliza una tabla donde la cabecera es el nombre de la competición que variará dinámicamente según se pulse en las diferentes competiciones. Y la tabla estática 1-X-2. En el cuerpo de la tabla mostraremos los eventos de forma dinámica, vemos el código HTML de la vista logueado de la Figura 18.

```
<!-- Tabla con los partidos - Cabecera Estatica-->
<div class="competicion-partidos" >
    <table class="tabla-partidos" >
        <thead>
        <tr>
            <td colspan="2">
                <div class="competicion-cabecera" id="nombre-competicion">
                    <!-- Nombre competicion - Javascript-->
                </div>
            </td>
            <td class="tabla-1X2">
                <div>
                    <table>
                        <tbody>
                        <tr>
                            <td>
                                <div class="cabecera-1X2">
                                    1
                                </div>
                            </td>
                            <td>
                                <div class="cabecera-1X2">
                                    X
                                </div>
                            </td>
                            <td>
                                <div class="cabecera-1X2">
                                    2
                                </div>
                            </td>
                        </tr>
                        </tbody>
                    </table>
                </div>
            </td>
        </tr>
        </thead>
        <tbody id="fila-evento">
        <!-- Tabla de eventos dinamica - Rellenamos con Javascript-->
        </tbody>
    </table>
</div>
```

Figura 18 Sección de código de la vista logueado.html para la parte de los eventos

La vista al cargarse ejecuta la función *getInitData()* de la clase JavaScript *logueado.js* importada al final del documento. En la Figura 19 y Figura 20 se muestra el código de la ejecución de esta función y de la carga del script dentro de la vista.

```
<body onload="getInitData();">
```

Figura 19 Ejecución de la función getInitData() en logueado.html

```
<script type="text/javascript" src="js/logueado.js"></script>
```

Figura 20 Carga del script logueado.js en la vista logueado.html

Esta función entre otras cosas llama al servicio para obtener los datos de los eventos futuros, los recorre para en cada iteración mostrar sus datos (fecha, equipo local, equipo visitante), obtener sus cuotas a través del servicio y presentarlas al usuario como se puede apreciar en la Figura 17.

(3) – Menú lateral y sus secciones. Desplegable del Cupón de apuestas

El menú lateral contiene las secciones de grupos que nos lleva a la pantalla de grupos del usuario, configuración que nos abre un formulario para el cambio de contraseña y salir para realizar logout y volver a la interfaz de usuario de login.

El cupón de apuestas es un desplegable en la parte derecha de la pantalla, cuando el usuario pulsa en alguna cuota, se abre el desplegable con toda la información referida al evento y a la cuota. El usuario introduce la cantidad a apostar, la aplicación muestra la ganancia potencial en función de dicha cantidad y pulsa el botón apostar como se muestra en la Figura 21. Es entonces cuando la aplicación muestra la pantalla elegir grupo donde se muestra la lista de grupos que el usuario es participante, pulsa en uno de ellos y se envía el cupón a dicho grupo para someterlo a votación.

(4) – Grupos

En la pantalla Grupos se puede observar los grupos en los que el usuario participa, pulsando en uno de ellos la app visualiza a la pantalla Ver-Grupo. Si se hace clic en Crear se muestra un formulario de creación de grupo donde el usuario introduce el nombre, selecciona una imagen como logo del grupo, elige un protocolo entre los protocolos de votación disponibles en la aplicación y añade una pequeña descripción al grupo para posteriormente pulsar el botón crear grupo.

Y por último si se pulsa en Invitaciones, podemos ver las invitaciones recibidas y aceptarlas o rechazarlas.

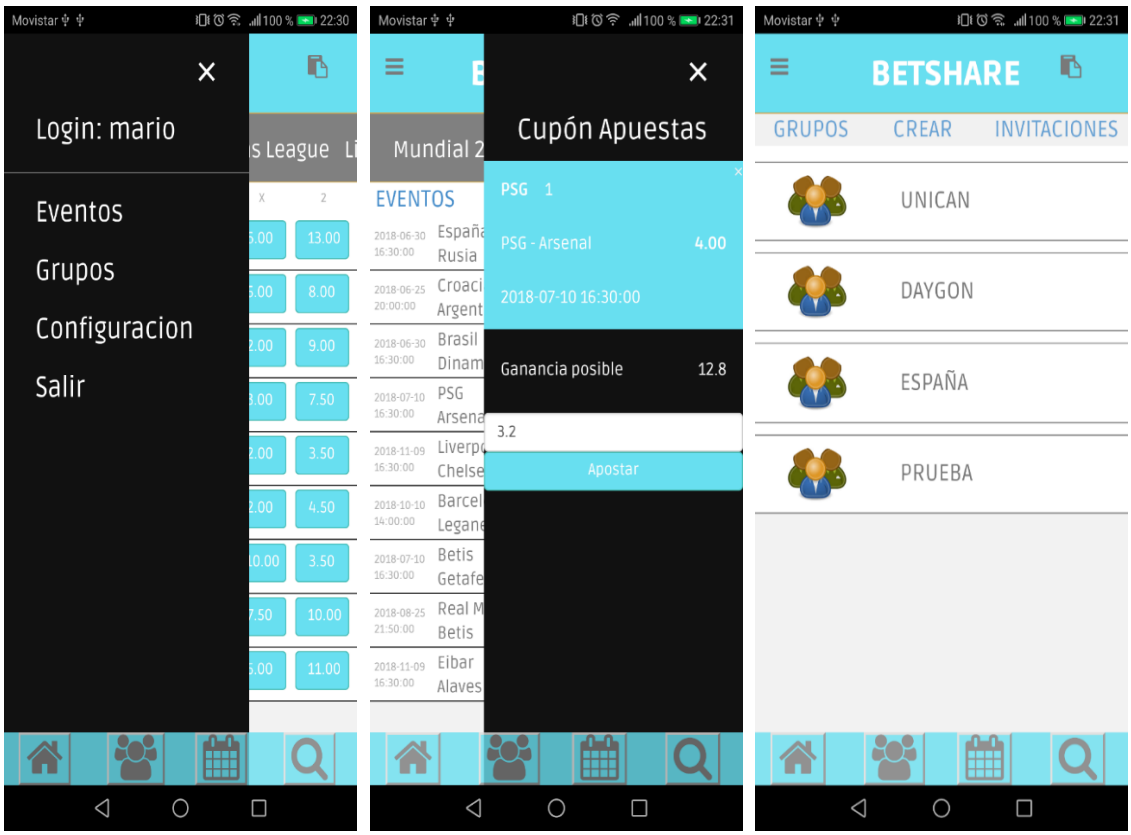


Figura 21 Menú lateral (izquierda) - Cupón Apuestas (centro) - Pantalla Grupos (derecha)

(5) – Ver Grupo y sección de Apuestas

Cuando pulsamos en uno de los grupos en la pantalla de la Figura 21 entramos dentro de la pantalla del grupo (Figura 22). En el apartado Info podemos ver la descripción del grupo, el depósito, el protocolo de votación elegido y ver los participantes del grupo con su rol, pudiendo enviar invitaciones pulsando el botón añadir. Al pulsar el botón añadir se carga la vista invitar usuario, donde se ve la lista de usuarios que no son participantes en el grupo y se pueden seleccionar para invitarlos al grupo pulsando el botón invitar.

En la sección Comentarios se pueden publicar comentarios y valorar los comentarios publicados anteriormente en el tablón.

La última sección Apuestas, está dividida en tres para tener las apuestas diferenciadas, Votación, En curso que son las apuestas realizadas a la espera del resultado del evento sobre el que se ha apostado y como vemos en la imagen de la derecha de la Figura 22, Historial con todos los cupones de apuestas, los rechazados, en votación, ganados y perdidos.

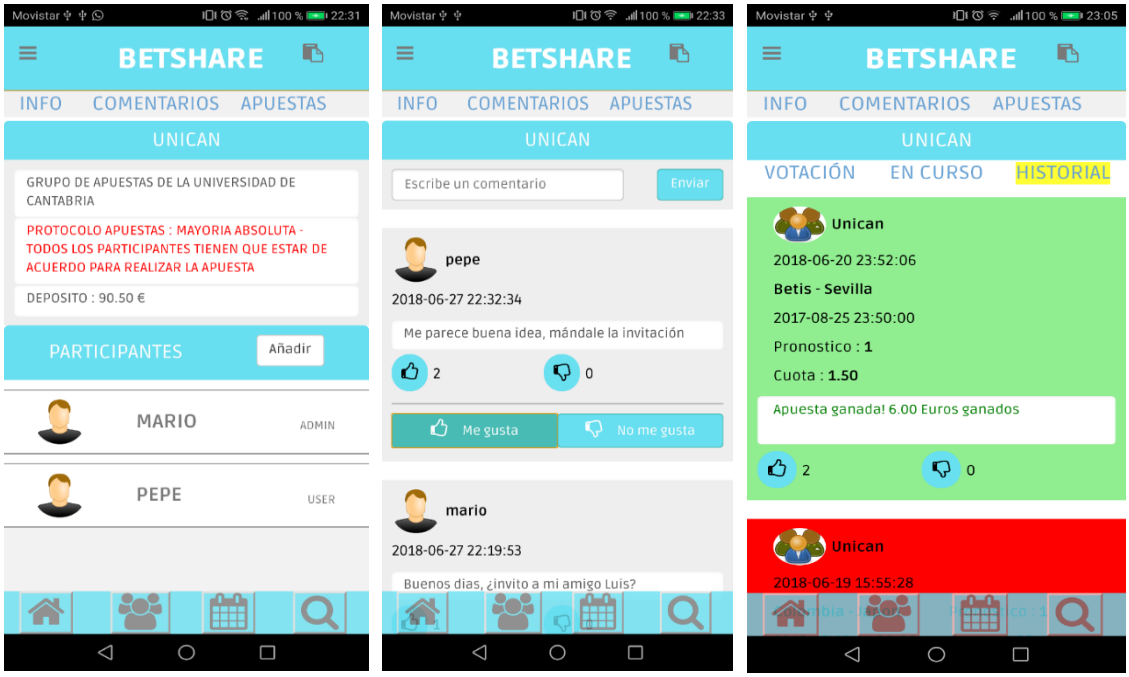


Figura 22 Información del grupo (izquierda) - Comentarios (centro) - Historial de apuestas (derecha)

A continuación se va a explicar la Figura 24, es decir el funcionamiento del cupón de apuestas desde el punto de vista de implementación. Cuando se pulsa en una cuota se ejecuta la función *addLineaApuesta(idCuota)*, se abre el cupón de apuestas y se hacen visibles el botón y los campos del cálculo de la ganancia. Obtenemos los datos de la cuota a través del servicio, pasando como parámetro el id que hemos recibido como parámetro en la función. Con esos datos nos muestra el nombre del equipo y el pronóstico “PSG 1”. Por último se muestra información del evento y de la cuota sobre la que vamos a apostar “4.00” en este caso como se aprecia en la Figura 21.

El cálculo de la ganancia se realiza de la siguiente manera, para ejecutar la función *calcularGanancia()* de la Figura 24, la vista HMTL utiliza la función *onkeyup* que ejecuta dicha función cuando el usuario suelta una tecla como se puede apreciar en la Figura 23.

```
<input type="number" min="0" pattern="^[0-9]+" class="form-control"
       name="importe" id="importe"  onkeyup="calcularGanancia(this)" placeholder="Importe" autofocus required >
```

Figura 23 Input para introducir la cantidad a apostar en la clase logueado.html

```
//Pulsas en una cuota y se añade al cupon de apuestas en el menu lateral DERECHO
function addLineaApuesta(idCuota){
    //Hacemos visibles el boton apostar, el campo para introducir el importe
    document.getElementById("cupon-form").style.visibility='visible';
    var datosCuota = Service.getDatosCuota(idCuota);   //pedimos al servicio los datos de la cuota(ratioEuros, ademas de datos del evento)
    if (datosCuota['resultado'] == "OK"){
        var datosCuota = datosCuota['data'];
        $("#lineas").empty();   //se ha pulsado otra cuota, eliminamos del menu lateral la anteriormente seleccionada
            $.each(datosCuota,function(index,val){
                $("#lineas").append("<a href='javascript:void(0)' class='eliminarbtn' onclick='eliminaLineaApuesta("+index+")'>&times;</a>");
                if(val.nombre == "1"){
                    $("#lineas").append("<p style='float:left;' id='ganador'><b>"+val.local+"</b></p>"+
                                        "<p>   1</p>");
                }
                if(val.nombre == "2"){
                    $("#lineas").append("<p style='float:left;' id='ganador'><b>"+val.visitante+"</b></p>"+
                                        "<p>   2</p>");
                }
                if(val.nombre == "X"){
                    $("#lineas").append("<p style='float:left;' id='ganador'><b>Empate</b></p>"+
                                        "<p>   X</p>");
                }
                $("#lineas").append("<p style='float:right;'  id='cuota'><b>"+val.ratioEuros+"</b></p>"+
                                    "<p id='evento-nombre'>"+val.local+" - "+val.visitante+"</p>"+
                                    "<p id='evento-fecha' style='padding-bottom:16px;'>"+val.fecha.toString()+"</p>"
                                    );

            });
    }

    document.getElementById("apostar").onclick=function(){doCupon(idCuota);};
    openCupon();
}

function eliminaLineaApuesta(lineaApuesta){
    $("#lineas").empty();
    document.getElementById("cupon-form").style.visibility='hidden';
}

 //Calcula la ganancia, con el importe introducido y el ratioEuros de la apuesta seleccionada
 function calcularGanancia(){

    var importe = document.getElementById("importe").value;
    var cuota = document.getElementById("cuota").innerText;
    ganancia = (importe) * (cuota)
    document.getElementById("ganancia").innerHTML = ganancia;

 }
```

Figura 24 Código con funciones del cupón de apuestas en la clase logueado.js

4.3 Diseño e implementación de la web de administración

En este apartado se va a explicar el diseño y la implementación de la web de administración. Se ha desarrollado con tecnologías HTML, CSS y PHP utilizando el framework CodeIgniter.

Tenemos por un lado las vistas en HTML y PHP, con el apoyo de hojas de estilo CSS para configurar los parámetros estéticos de las vistas los cuales se pueden observar en la Figura 28. Las vistas contienen los formularios de creación de usuario y logueo, además de los formularios y botones necesarios para la creación, edición y borrado de las competiciones, los eventos, las cuotas y los resultados. También nos permite tener un control de los usuarios registrados y cambiar nuestra contraseña.

Estos formularios ejecutan una función PHP, la ruta *comprobar-admin* en este caso, será traducida por el archivo de configuración *routes.php* de la misma manera que hemos visto anteriormente. Para mostrar los mensajes de error se llama a la función *validation_errors()* como se puede comprobar en la Figura 25.

```
<form class="form-login" method="POST" action="<?php echo base_url("crear-competicion")?>">
  <h3 align="left" style="color:red;font-size:12px;"> <?php echo  validation_errors(); ?></h3>
```

Figura 25 Formulario de la vista de creación de competición

Las funciones de los controladores de CodeIgniter de la web de administración, el funcionamiento es similar conceptualmente a lo explicado en el apartado Diseño e implementación de la capa de negocio4.2.2, de hecho los modelos son comunes tanto los de la aplicación móvil como los de la web de administración.

Los controladores sin embargo están organizados por separado como se pudo ver en la Figura 11, pues hay unas pequeñas diferencias en su funcionamiento debido al uso de formularios en las vistas. Vamos a ver la forma que se trabaja con ellos en CodeIgniter para el caso de uso en que el administrador quiere crear una nueva competición.

```
public function crear_competicion(){
    //Comprobamos que el nombre no esté en uso para crear y editar
    $this->form_validation->set_rules('nombre', 'Nombre', 'trim|required|callback_check_nombre');
    //Comprobamos que el deporte sea uno de los deportes que damos soporte en la casa de apuesta
    $this->form_validation->set_rules('deporte', 'Deporte', 'trim|required|callback_check_deporte');
     if ($this->form_validation->run() == FALSE)
    {
        //ERROR NO CUMPLE NUESTAS VALIDACIONES(Nombre en uso o deporte no soportado)
        $id_competicion = $this->input->post('id-competicion');
        if (empty($id_competicion)){
            //Si no existe id, es un error de validacion de formulario
            $this->index();
        }else{
            //Si tiene id es una edición
            $this->editar_competicion(urlencode(base64_encode($id_competicion)));
        }

    }
    else
    {
        //Si pasa el formulario
        //COGEMOS EL VALOR ID de competicion si esta vacio es crear
        $id_competicion = $this->input->post('id-competicion');
        //CARGAMOS LOS DATOS EN VARIABLES
        $nombre = $this->input->post('nombre');
        $deporte = $this->input->post('deporte');
        $descripcion = $this->input->post('descripcion');
        $this->load->model('competiciones_model','',TRUE);
        //Llamamos al modelo
        $resultado = $this->competiciones_model->crear_competicion($id_competicion,$nombre,$deporte,$descripcion);
            if ($resultado === FALSE){
                // SE HA PRODUCIDO UN ERROR....
                $this->index();
            }
            else{
                redirect(base_url("ver-competiciones"));
            }
    }
}
```

Figura 26 Sección de código de la función comprobar_admin()

A diferencia de la aplicación donde se envía al controlador un objeto JSON, aquí cargamos los datos en las variables utilizando el método *post* como se puede apreciar en la Figura 26. Una vez estén los datos en variables se procede a cargar el modelo y llamar a la función de creación de competición. Previamente se establecen reglas en sus parámetros, para al ejecutar la validación del formulario realizar las comprobaciones sobre los datos introducidos. Algunas de estas reglas pueden ser llamadas a funciones de comprobación como en el ejemplo de la Figura 27, donde se comprueba si el nombre de la competición está en uso y si el deporte es uno de los permitidos en la aplicación.

```php
//Funcion para la validación de formulario, comprueba si el nombre de competicion está en uso
function check_nombre($nombre){
    $id_competicion = $this->input->post('id-competicion');
    $this->load->model('competiciones_model','',TRUE);
    if ($this->competiciones_model->comprobar_nombre($id_competicion,$nombre) === FALSE){
                // El nombre está en uso
                $this->form_validation->set_message('check_nombre', 'El nombre introducido está en uso, prueba con otro');
                return FALSE;
        }
        return TRUE;
}
//Funcion para la validación de formulario, comprueba si el deporte introducido está permitido en nuestra aplicación.
function check_deporte($deporte){
    $this->load->model('competiciones_model','',TRUE);
    if ($this->competiciones_model->comprobar_deporte($deporte) === FALSE){
                // El deporte no es uno de los permitidos
                $this->form_validation->set_message('check_deporte', 'El deporte introducido no es un deporte permitido por BETSHARE');
                return FALSE;
        }
        return TRUE;
}
```

Figura 27 Funciones check_nombre y check_deporte dentro del controlador Competiciones_controller

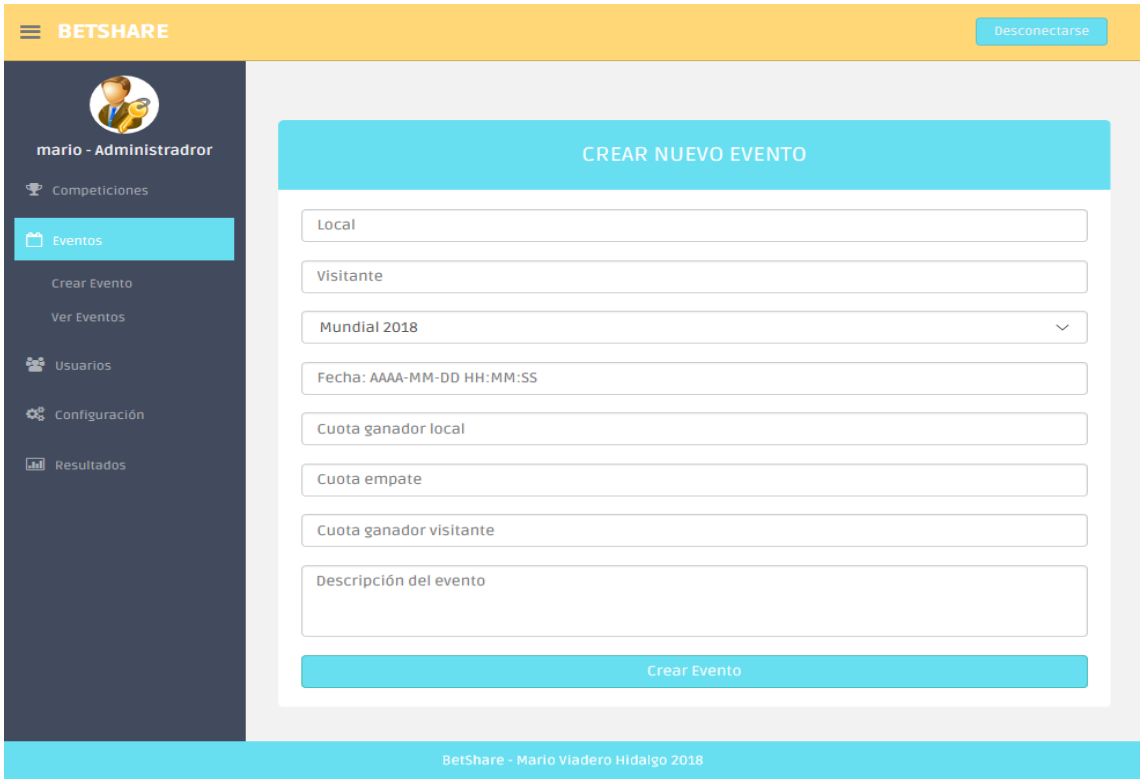


Figura 28 Web de administración. Formulario de creación de nuevo evento

5. Pruebas

5.1 Pruebas a la web de administración.
5.2 Pruebas a la aplicación móvil.

En esté capítulo se va a desarrollar el conjunto de pruebas realizadas a nuestro sistema. Una de las mejores formas para tener la certeza que el software desarrollado se comporta como se espera y de detectar errores, es probar su funcionamiento bajo ciertas circunstancias. Las pruebas se pueden separar en diferentes tipos [19]:

- **Pruebas unitarias:** Estas pruebas sirven para comprobar unidades del software, normalmente los métodos, aquí nos interesa como funciona la unidad y no la interacción entre componentes.
- **Pruebas de integración:** Este tipo de pruebas verifican que los componentes de la aplicación funcionan bien en conjunto por tanto son dependientes del entorno, pueden servir para ver como interactúan varias unidades, como por ejemplo las vistas con los controladores, o los modelos con la base de datos dentro de la aplicación.
- **Pruebas funcionales:** En este caso se quiere comprobar que el software que se ha creado no solo funciona correctamente como se puede comprobar en pruebas anteriores, sino que realiza la función para que se había programado. Serían lo que llamamos pruebas de caja negra, solo nos importan las entradas y las salidas de datos.
- **Pruebas de carga:** Con este tipo de pruebas lo que queremos es comprobar la respuesta de la aplicación ante un determinado número de peticiones.
- **Pruebas de estrés:** Es otro tipo de prueba de rendimiento del sistema, sometiéndole a situaciones extremas, sobrecargándolo de tal manera que intentemos que el sistema caiga y analizar en esa situación como se comporta, si es capaz de recuperarse o si se producen errores graves.
- **Pruebas de aceptación:** Son pruebas que se realizan con el cliente para comprobar si el software cumple con sus expectativas.

5.1 Pruebas a la web de administración

Para realizar las pruebas a la web de administración usaremos Selenium, que es un plugin para el navegador Firefox que consta de un conjunto de utilidades que nos permite, grabar, editar y depurar casos de prueba que se podrán ejecutar de forma automática e iterativa. Se va a automatizar una prueba para que el administrador después de loguearse, cree una nueva competición rellenando el formulario, posteriormente la borre y por último cierre sesión. Se puede ver el resultado del test en la Figura 29.

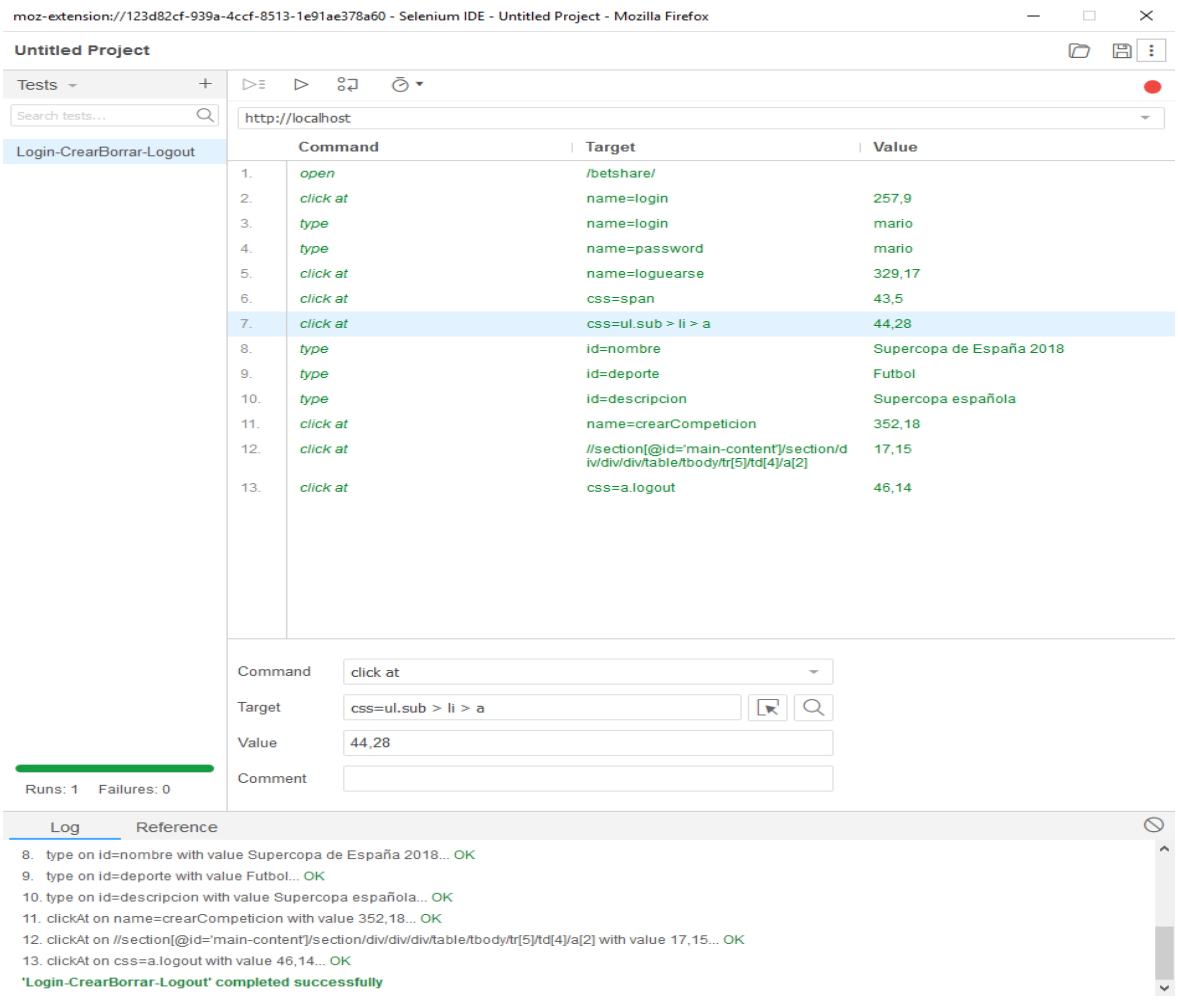


Figura 29 Prueba de la web de administración con Selenium

5.2 Pruebas a la aplicación móvil

Para la realización de las pruebas de la aplicación móvil se utilizan clases TestCase de PhpUnit en las que se probarán las funciones de los modelos y controladores de CodeIgniter de la aplicación. En primer lugar se probarán los modelos, las clases test tienen métodos para inicializar y finalizar el test, estos métodos son setUp(), donde por ejemplo se inicializa el modelo que se va a probar, para poder ejecutar sus métodos y ser probados y el método tearDown() en el que se termina el test.

Vamos a tomar como ejemplo las pruebas sobre el modelo encargado de gestionar las apuestas, se puede observar en la Figura 30 el aspecto que tienen los métodos de prueba test_doEnviarCupon y test_getCupones, donde se envía un cupón de apuesta a un grupo y donde se obtiene los cupones de apuestas de un grupo para el segundo caso.

```php
public function test_doEnviarCupon(){
    //Enviamos el cupon, con el id de la cuota, el importe apostado, la ganancia, el idgrupo y el idusuario
    $this->assertTrue($this->_apuestas_model->doEnviarCupon(80,2,4,1,1));
    $this->assertTrue($this->_apuestas_model->doEnviarCupon(80,2,4,1,1));
    $this->assertTrue($this->_apuestas_model->doEnviarCupon(85,2,8,1,2));
    $this->assertFalse($this->_apuestas_model->doEnviarCupon('','','','',''));
}

public function test_getCupones(){
    //Obtenemos los cupones en el grupo 1
    $cupones=$this->_apuestas_model->getCupones(1);
    $this->assertArrayHasKey(0,$cupones);
    $this->assertSame('1',$cupones[0]['Usuario_idUsuario']);
    $this->assertSame('2.00',$cupones[0]['importeApostado']);
    $this->assertSame('4.00',$cupones[0]['ganancia']);
    $this->assertSame('Liverpool',$cupones[0]['local']);
    $this->assertArrayHasKey(1,$cupones);
    $this->assertSame('2',$cupones[1]['Usuario_idUsuario']);
    $this->assertSame('2.00',$cupones[1]['importeApostado']);
    $this->assertSame('8.00',$cupones[1]['ganancia']);
    $this->assertSame('Madrid',$cupones[1]['local']);
}
```

Figura 30 Métodos de prueba test_doEnviarCupon() y test_getCupones()

Se puede observar el funcionamiento de los mismos, donde se ejecutan métodos a través de *_apuestas_model,* que es la variable donde se ha inicializado el modelo anteriormente en el método setUp() y se realizan las comprobaciones mediante asertos. Para el primer método se comprueba si los cupones se envían o no correctamente, para el segundo se comprueba si los datos devueltos por el modelo se corresponden con los datos correctos en la base de datos.

La ejecución de los test con PhpUnit se realiza por consola y una vez realizados los test, ya sea de todos los modelos, de un modelo o de un método en concreto, PhpUnit genera unos informes en nuestro caso HTML como se aprecia en la Figura 31. Estos informes detallan el grado en el que la clase a sido testeada y el porcentaje de líneas de código que se han cubierto en los test, en este caso se puede apreciar como en algunos pocos métodos faltan un par de líneas por cubrir, esto es debido a que existen ciertos métodos que inician transacciones para añadir datos a la base datos y se contempla el caso de que la transacción sea errónea. Por tanto para cubrir ese segmento de código se debería producir un error en la transacción dentro del gestor de bases de datos, algo que no es posible mediante este tipo de test.

Para concluir con la fase de pruebas referida a los modelos, se van a mostrar un ejemplo de clase de prueba con cobertura total (Figura 32) y el ejemplo detallado línea a línea de un método, donde se puede apreciar lo comentado anteriormente acerca de las transacciones erróneas (Figura 33).

C:\xampp\htdocs\betshare\application / models / apuestas_model.php

	Code Coverage								
	Classes and Traits			Functions and Methods			Lines		
Total		0.00%	0 / 1		80.00%	12 / 15	CRAP	96.08%	147 / 153
Apuestas_model		0.00%	0 / 1		80.00%	12 / 15	40	96.69%	146 / 151
__construct					100.00%	1 / 1	1	100.00%	2 / 2
getDatosCuota					100.00%	1 / 1	1	100.00%	4 / 4
doEnviarCupon					0.00%	0 / 1	7.06	89.47%	17 / 19
getCupones					100.00%	1 / 1	1	100.00%	4 / 4
getCuponesEnCurso					100.00%	1 / 1	1	100.00%	4 / 4
getCuponesHistorico					100.00%	1 / 1	1	100.00%	4 / 4
getVotosFavor					100.00%	1 / 1	2	100.00%	7 / 7
getVotoscontra					100.00%	1 / 1	2	100.00%	7 / 7
doVotarCupon					100.00%	1 / 1	6	100.00%	23 / 23
getDatosCupon					100.00%	1 / 1	1	100.00%	4 / 4
getDatosApuesta					100.00%	1 / 1	1	100.00%	4 / 4
getImporte					100.00%	1 / 1	2	100.00%	7 / 7
getGanancia					100.00%	1 / 1	2	100.00%	7 / 7
doApostar					0.00%	0 / 1	6.07	87.50%	14 / 16
doCambiarEstado					0.00%	0 / 1	6	97.44%	38 / 39

Figura 31 Cobertura de las pruebas en la clase apuestas_model.php

C:\xampp\htdocs\betshare\application / models / usuario_model.php

	Code Coverage								
	Classes and Traits			Functions and Methods			Lines		
Total		100.00%	1 / 1		100.00%	14 / 14	CRAP	98.96%	95 / 96
Usuario_model		100.00%	1 / 1		100.00%	14 / 14	33	100.00%	94 / 94
__construct					100.00%	1 / 1	1	100.00%	2 / 2
login_usuario					100.00%	1 / 1	4	100.00%	11 / 11
registro_usuario					100.00%	1 / 1	9	100.00%	18 / 18
comprobar_edad					100.00%	1 / 1	2	100.00%	4 / 4
comprobar_email					100.00%	1 / 1	2	100.00%	6 / 6
comprobar_login					100.00%	1 / 1	2	100.00%	6 / 6
getAllUsuarios					100.00%	1 / 1	1	100.00%	3 / 3
getUsuariosInvitar					100.00%	1 / 1	1	100.00%	4 / 4
getUsuarios					100.00%	1 / 1	1	100.00%	5 / 5
getUserByEmail					100.00%	1 / 1	3	100.00%	9 / 9
getUserByLogin					100.00%	1 / 1	3	100.00%	9 / 9
cambiar_contraseña					100.00%	1 / 1	1	100.00%	6 / 6
eliminar_usuario					100.00%	1 / 1	1	100.00%	4 / 4
getDatosUsuario					100.00%	1 / 1	2	100.00%	7 / 7

Figura 32 Cobertura de las pruebas en la clase usuario_model.php

```
36          $query = $this->db->query($sql);
37          $idCupon = $this->db->insert_id();
38          $cuota =$ganancia/$importe;
39          $sql = "INSERT into CuponCuota(Cupon_idCupon,Cuota_idCuota,cuota) values($idCupon,$idCuota,$cuota)";
40          $query = $this->db->query($sql);
41          if ($this->db->trans_status() === FALSE){
42              //Hubo errores en la consulta, se cancela la transacción.
43              $this->db->trans_rollback();
44              return false;
45          }else{
46              //No hubo errores
47              $this->db->trans_commit();
48              return true;
49          }
```

Figura 33 Fragmento del método doEnviarCupon()

La siguiente fase de las pruebas sobre la aplicación móvil son las pruebas sobre el servicio, que es el encargado de enviar peticiones a los controladores e intercambiar información mediante archivos JSON. Para ello también se utiliza PhpUnit, por tanto el funcionamiento, formato, los informes y la forma de ejecución de los test es similar a lo explicado para los modelos. Lo que se diferencia en este caso es el manejo de archivos JSON y las peticiones a los controladores, estas particularidades se van a explicar a continuación en la Figura 34.

```
public function test_doEnviarCupon(){
    $data = json_encode(array("idCuota" => "85",
        'importe' => "10.00",
        'ganancia' => "40.00",
        'idGrupo' => "2",
        'user' => "1"));
    $cupon = $this->request('POST', 'api/enviar-cupon' , $data);
    $this->assertTrue($cupon);
}
```

Figura 34 Código del método test_doEnviarCupon()

Siguiendo con el ejemplo del método en el cual un usuario envía un cupón de apuestas a un grupo, se puede observar que lo primero que realiza el método es codificar el archivo JSON con los datos que necesita el controlador para atender a la petición, posteriormente se realiza la petición donde se envía el archivo JSON anteriormente codificado a la dirección *api/enviar-cupon.* Esta dirección como se ha explicado en capítulos anteriores, el archivo *routes.php* se encargará de mapear la dirección en una función de alguno de los controladores de la aplicación, por último, guardamos la respuesta en una variable para comprobar que la operación se ha realizado correctamente.

6. Conclusiones

6.1 Consecución de objetivos.
6.2 Trabajos futuros.

En este último apartado se va a analizar si los objetivos planteados al inicio del proyecto en el apartado 1.2, se han cumplido durante la realización del mismo. Además se plantearan las posibles funcionalidades que se pueden añadir al sistema en futuros trabajos, partiendo del estado en el que se encuentra el proyecto en la actualidad.

6.1 Consecución de objetivos

Al inicio del proyecto se marcaron unos objetivos que se debían cumplir durante el desarrollo del sistema para ser considerado satisfactorio, estos objetivos se plantearon en forma de lista como se muestra a continuación:

1. Diseñar un sistema web que permita:
- **1.1 Darse de alta en el sistema como administrador.**
- **1.2 Administrar competiciones y eventos deportivos.**
- **1.3 Administrar cuotas y resultados de los eventos deportivos.**

En primer lugar se diseñó la web de administración encargada de gestionar las competiciones y eventos deportivos con sus cuotas, además de los resultados de los eventos, puesto que sin ella en funcionamiento no se podrían realizar apuestas con la aplicación para dispositivos móviles. Las dos primeras decisiones que se tomaron para el desarrollo de la web de administración fueron utilizar lenguaje MySQL para la base de datos y utilizar el framework PHP Codeigniter.

El uso de MySQL se justifica en que la base de datos es relacional y además tiene licencia pública general, se crea la base de datos partiendo de un modelo entidad-relación que irá evolucionando a lo largo de las iteraciones del desarrollo. La base de datos desde un primer momento se alojó en un servidor. Se utilizó Codeigniter para el desarrollo de aplicaciones y sitios web utilizando PHP, debido a que actualmente es un framework muy utilizado y muy versátil ya que trabaja en la mayoría de entornos y servidores, además se adapta al diseño del sistema puesto que utiliza modelo-vista-controlador y agiliza mucho el desarrollo porque contiene un conjunto de librerias para tareas comúnmente necesarias permitiendo centrarse creativamente en el proyecto y minimizando el código necesario. Durante el desarrollo se utilizó el servidor web de XAMPP para alojar el proyecto Codeigniter, por tanto los modelos, los controladores de la web y de la aplicación, y las vistas de la web de administración solo estaban disponibles de manera local en este servidor.

Una vez creada la base de datos y configurado el proyecto Codeginiter donde entre otras cosas se realiza la conexión a la base de datos, en primer lugar se implementó la lógica del sistema de registro, login y logout del sistema mediante modelos y controladores de CodeIgniter, y paralelamente sus vistas y formularios. Ya estando logueado como administrador se diseña la interfaz visual de la web y su funcionalidad, para permitir listar, crear, borrar y editar competiciones deportivas y eventos con sus cuotas.

A partir de aquí el desarrollo se centra en la aplicación para móviles, para ello se utiliza Android Studio y consta de vistas HTML y CSS, clases JavaScript de cada vista entre las que se encuentra el servicio encargado de llamar a funciones de los controladores e intercambiar paquetes JSON. Los controladores de la aplicación son independientes a los de la web de administración, pues manejan estos paquetes JSON pero forman parte del mismo proyecto CodeIgniter ejecutado en el servidor web de XAMPP durante el desarrollo. Por último también se implementan modelos y funciones propios de la aplicación para móviles, con las peticiones a la base de datos necesarias. Siguiendo la lista de objetivos:

2. Diseñar una aplicación para dispositivos móviles inteligentes que permita:

- **2.1 Darse de alta en el sistema como usuario.**
- **2.2 Ver los eventos deportivos, sus cuotas y filtrar los eventos por competición deportiva.**
- **2.3 Crear grupos de apuestas y enviar invitaciones a otros usuarios**
- **2.4 Ver los grupos en los que somos participante y la descripción del mismo, así como los otros participantes del grupo.**
- **2.5 Enviar cupones a los grupos con pronósticos y cantidades para proceder a la votación al instante.**
- **2.6 Chatear con el resto de integrantes del grupo a través de un tablón de mensajes y valorar los comentarios.**
- **2.7 Ver un histórico de los cupones de apuestas de cada grupo.**

Para cada fase del desarrollo, lo que se va a a implementar son las vistas HTML y sus hojas de estilo CSS, además de las funciones del servicio encargado de llamar a las funciones de los controladores las cuales también habrá que implementar en el controlador correspondiente, y por último las clases JavaScript de cada vista, que son las encargadas ejecutar funciones del servicio y modificar la vista con la información obtenida en estas llamadas.

La primera fase del desarrollo es adaptar el sistema de registro y login ya implementados en la web, a la aplicación móvil. Después de que el usuario se pueda dar de alta, loguear en la aplicación y ver las competiciones, eventos y cuotas, la siguiente parte a desarrollar es la lógica en torno a los grupos, que es la parte principal de este proyecto, realizar apuestas deportivas colaborativamente.

Llegados a este punto, los usuarios pueden crear grupos de apuestas seleccionando el protocolo de votación y enviar invitaciones para participar en el grupo a otros usuarios que no formen parte del grupo. Además se implementa un tablón de comentarios con votos positivos o negativos.

La siguiente parte del desarrollo es la referida a los cupones de apuestas, en la que se implementa que el usuario pulse en una cuota de las disponibles en la aplicación y tras introducir una cantidad, mande el cupón a los grupos que desee. Este cupón se someterá a votación que estará regulada por el protocolo seleccionado anteriormente y en caso de ser favorable, se realizará la apuesta.

La última parte del desarrollo es la lógica de negocio para cuando el administrador asigna un resultado a un evento deportivo, se realice la comprobación de todas las apuestas realizadas a dicho evento y en función del resultado pronosticado, asignar a la apuesta la condición de ganada o perdida y realizar el pago de la cuota. Esto último además implica realizar

el diseño de la presentación del histórico de apuestas, diferenciando visualmente las apuestas ganadas, de las perdidas o rechazadas en votación.

Por último, para poder ejecutar la aplicación desde cualquier parte, se aloja el proyecto en el servidor donde también está alojada la base de datos. De esta manera se da por finalizado el desarrollo de la aplicación cumpliendo todos los objetivos marcados al inicio del proyecto.

6.2 Trabajos futuros

En primer lugar, me gustaría destacar que la realización de un proyecto de esta magnitud me ha servido de gran aprendizaje a la hora de utilizar herramientas y tecnologías que nunca antes había utilizado, especialmente las relacionadas con el mundo móvil.

Debido al desarrollo incremental del proyecto, a medida que se van finalizando etapas, surgen nuevas ideas para añadir nuevas funcionalidades al proyecto, algunas de las cuales excederían de los objetivos planteados en el inicio pero que en un futuro sería interesante añadir como por ejemplo:

- Añadir más tipos de apuestas y más deportes, con sus respectivas competiciones, eventos y cuotas.
- Permitir la personalización del avatar de usuario y del logo del grupo, en sustitución de imágenes genéricas actuales.
- Añadir mas protocolos de votación a la aplicación.
- Mensajería privada
- En lugar de utilizar dinero simulado, establecer un sistema para añadir fondos al grupo, con dinero real y paarelas de pago.
- Conexión con api de resultados de eventos deportivos para mejorar la gestión de las competiciones, eventos y resultados.
- Sistema de logueo con cuenta de una red social externa a BETSHARE (Facebook,Twitter).

Por último, se podría desarrollar la aplicación en otra plataforma distinta de Android como iOS para permitir el uso en más dispositivos.

Referencias

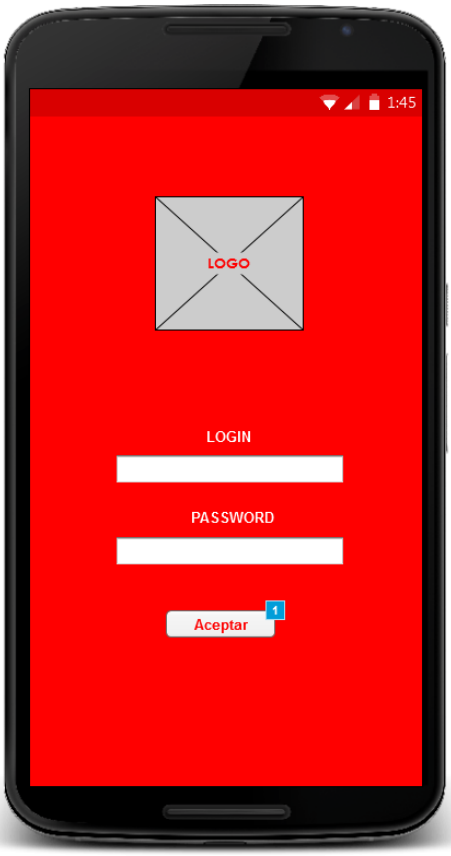
1. ANDROID STUDIO. Google LLC. [Última visita: 20 de agosto 2018]. Disponible en: https://developer.android.com/studio/
2. MYSQL. Oracle Corporation. [Última visita: 18 de marzo 2017]. Disponible en: https://www.mysql.com/
3. MYSQL WORKBENCH. Oracle Corporation. [Última visita: 18 de marzo 2017]. Disponible en: https://dev.mysql.com/downloads/workbench/?utm_source=tuicool
4. W3SCHOOLS HTML TUTORIAL. Refsnes Data. [Última visita: 9 de mayo 2018]. Disponible en: https://www.w3schools.com/html/
5. W3SCHOOLS CSS TUTORIAL. Refsnes Data. [Última visita: 14 de mayo 2018]. Disponible en: https://www.w3schools.com/css/
6. PHP. The PHP Group. [Última visita: 5 de junio 2018]. Disponible en: http://php.net/manual/es/intro-whatis.php
7. CODEIGNITER. EllisLab, Inc. [Última visita: 13 de agosto 2018]. Disponible en: https://codeigniter.com/
8. JAVASCRIPT. Pluralsight. [Última visita: 22 de junio 2018]. Disponible en: https://www.javascript.com/
9. W3SCHOOLS AJAX TUTORIAL. Refsnes Data. [Última visita: 3 de mayo 2018]. Disponible en: https://www.w3schools.com/js/js_ajax_intro.asp
10. JSON. [Última visita: 21 de junio 2018]. Disponible en: https://www.json.org/
11. BOOTSTRAP. Twitter, Inc. [Última visita: 12 de septiembre 2017]. Disponible en: https://getbootstrap.com/
12. JQUERY. The jQuery Foundation. [Última visita: 29 de agosto 2017]. Disponible en: https://jquery.com/
13. PHONEGAP. Adobe Systems Inc. [Última visita: 12 de septiembre 2017]. Disponible en: https://phonegap.com/
14. PHPUNIT. Sebastian Bergmann. [Última visita: 24 de agosto 2018]. Disponible en: https://phpunit.de/
15. MODELO ITERATIVO. [Última visita: 19 de septiembre 2017]. Disponible en: http://aurumsol.com/espanol/articulos/art1/images/iterativo.jpg
16. Sommerville, Ian. (2005). Ingeniería del software: Addison-Wesley
17. ISO 9241–11, Ergonomic Requirements for Office Work with Visual Display Terminals (VDTs) – Part 11: Guidance on Usability, 1998
18. Reglamento (UE) 2016/679 del Parlamento Europeo y del Consejo, de 27 de abril de 2016, relativo a la protección de las personas físicas en lo que respecta al tratamiento de datos personales y a la libre circulación de estos datos: Boletín Oficial del Estado, 15 de junio de 2018 p.94
19. TIPOS DE PRUEBAS. Javier Garzás. [Última visita: 13 de agosto 2018]. Disponible en: http://www.javiergarzas.com/2014/07/tipos-de-pruebas-10-min.html

Anexo I. Acrónimos

HTML - HyperText Markup Language
CSS - Cascading Style Sheets
MySQL – Structured Query Lenguage
AVD Manager - Android Virtual Devices Manager
SDK Manager – Software Development Kit
XML – eXtensible Markup Langueage
JSON – JavaScript Object Notation
PHP – Hypertext Preprocessor
IDE – Entorno de Desarollo Integrado
API – Application Programming Interface
W3C – WorldWideWeb Consortium
OS – Operating system
Ajax – Asynchronous JavaScript and XML

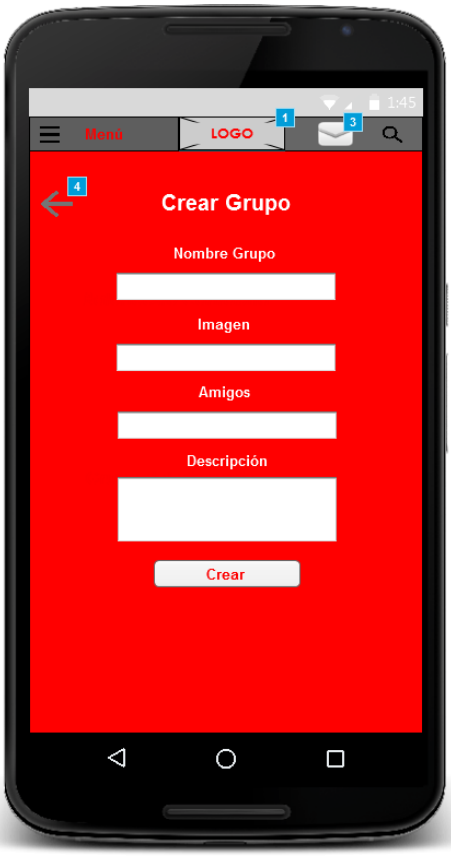
Anexo II. Prototipos Axure RP

En este anexo se muestran unas capturas de pantalla de Axure RP en las que se muestran los prototipos realizados para la aplicación móvil. Dichas imágenes pertenecen a la pantalla de login, la pantalla de home, el apartado de pronósticos de futbol, el tablón de comentarios, el apartado de grupos en los que participa el usuario y las diversas opciones y tablones de notificaciones dentro de un grupo.




1:45
Menú
LOGO
1
2
Futbol - Liga LFP - Primera División
sábado - 21/11/2015
Real Sociedad - Sevilla
1 2.85 X 2.00 2 1.95
Espanyol - Malaga
1 2.89 X 2.40 2 1.55
Valencia - U.D Las Palmas
1 2.85 X 2.00 2 1.95
Deportivo de La Coruña - Celta de Vigo
1 2.85 X 2.00 2 1.95
domingo - 22/11/2015
Sporting de Gijón - Levante
1 2.85 X 2.00 2 1.95
Villareal - Eibar
1 2.85 X 2.00 2 1.95
Real Madrid - Barcelona
1 2.85 X 2.00 2 1.95

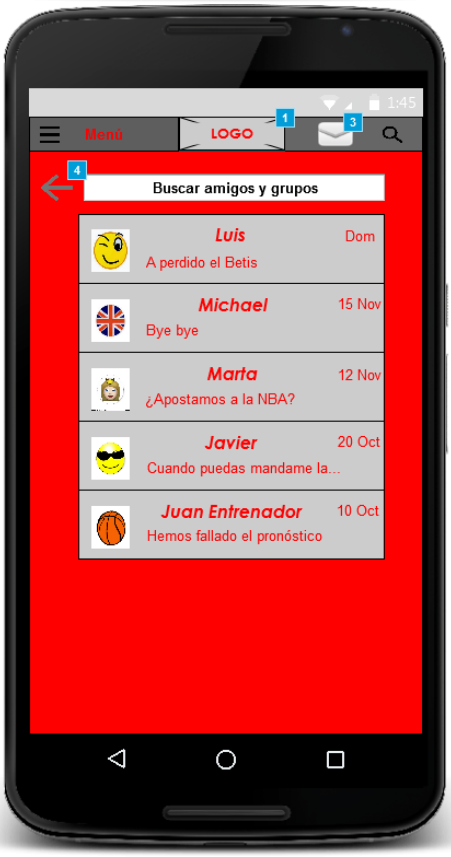

1:45
Menú
LOGO
1
3
4
Grupos en los que soy participante :
5
Baloncesto Cantabria
Amigos Universidad
Grupo del gimnasio
6
Crear nuevo grupo


1:45
Menú
LOGO
1
3
4
Crear Grupo
Nombre Grupo
Imagen
Amigos
Descripción
Crear


1:45
Menú
LOGO
1
3
4
Baloncesto Cantabria
5
Notificaciones
6
Pronósticos
Participantes
Luis
Sab
Ha recomendado apostar X en el Villareal - Eibar ; cuota 2,25
Apostar 5 Descartar 3
Mario
15/11
Mi amigo Francisco desea participar en el grupo
Aceptar 7 Rechazar 0
Michael
11/11
Estamos fallando pronósticos,deberiamos apostar menos cantidades de dinero
Me gusta 4 No me gusta 3
Inserta nuevo comentario en el tablón


1:45
Menú
LOGO
1
3
4
Baloncesto Cantabria
Notificaciones
5
Pronósticos
6
Participantes
Futbol
Sab
Has apostado 10 € a X en el Villareal - Eibar ; cuota 2,25
Baloncesto
1/12
Has apostado 5 € a la victoria de Lakers en el Lakers - Bulls ; cuota 3.30
Tenis
1/6/2016
Has apostado 20 € a que Rafael Nadal será el nº1 del mundo en Junio 2016 ; cuota 8,60
Ver historial de pronósticos
7


1:45
Menú
LOGO
1
3
4
Baloncesto Cantabria
Notificaciones
5
Pronósticos
6
Participantes
Futbol
Lun
Betis - Sevilla ; Pronóstico X ; Resultado 3-3 ; GANADA - 22,5 €
Baloncesto
10/09
Lakers - Celtics ; Pronostico Lakers +5 ; Resultado 90 - 100 ; PERDIDA - 5€
Tenis
12/6/2015
Nadal - Federer ; Pronóstico Nadal ; Resultado 6-3,6-2 ; GANADA 14,5 €
Ver pronósticos
7


1:45
Menú
LOGO
1
3
4
Buscar amigos y grupos
Luis
Dom
A perdido el Betis
Michael
15 Nov
Bye bye
Marta
12 Nov
¿Apostamos a la NBA?
Javier
20 Oct
Cuando puedas mandame la...
Juan Entrenador
10 Oct
Hemos fallado el pronóstico