

**SportNet**

**DISEÑO E IMPLEMENTACIÓN DE UNA APLICACIÓN WEB  
PARA UNA RED SOCIAL DEPORTIVA**

**ID: MIA-19-008**

**Madrid, julio 2019**

**Versión 1.0**

**Estatus: público**

# Índice de Contenidos

<b>1 Objetivo .....</b>	<b>1</b>
<b>2 Ciclo de vida de desarrollo.....</b>	<b>2</b>
<b>3 Análisis.....</b>	<b>4</b>
3.1 Funcionalidades principales.....	4
3.2 Desglose en subsistemas.....	4
3.3 Requisitos.....	5
<b>4 Diseño.....</b>	<b>8</b>
4.1 Arquitectura general .....	8
4.2 Interfaz de usuario .....	9
4.3 Base de datos .....	9
4.4 API RPC .....	13
<b>5 Implementación .....</b>	<b>14</b>
5.1 Estructura del sistema .....	14
5.1.1. Controladores de la parte de la aplicación .....	14
5.1.2. Conexión con la base de datos .....	15
5.1.3. Controladores de la API RPC .....	17
5.1.4. Capa de negocio .....	18
5.2 Inicio de la aplicación .....	19
5.2.1. Interfaz de deportes .....	20
5.2.2. Interfaz de edición de perfiles .....	21
5.2.3. Interfaz de eventos y publicaciones .....	22
5.2.4. Interfaz de fichas personales y deportivas .....	22
5.2.5. Interfaz de amistades.....	23

5.2.6.	Interfaz del perfil de usuario .....	23
5.2.7.	Interfaz del timeline .....	24
5.2.8.	Otras interfaces .....	24
5.3	Login de administrador .....	25
5.3.1	Dashboard .....	25
5.3.2	Interfaces de control .....	26
<b>6</b>	<b>Pruebas .....</b>	<b>28</b>
6.1	Verificación .....	28
6.1.1.	Estrategia de pruebas .....	28
6.1.2.	Desarrollo de las pruebas .....	29
6.2.	Validación.....	30
6.2.1.	Estrategia y desarrollo de pruebas de validación.....	30
6.2.2.	Pruebas de aceptación .....	31

# Índice de Figuras

FIGURA 2.1: CICLO DE VIDA DE LA APLICACIÓN .....	2
FIGURA 4.1: DIAGRAMA DE LA ARQUITECTURA GENERAL DE LA APLICACIÓN.....	9
FIGURA 4.2: DIAGRAMA E/R DE LA BASE DE DATOS.....	11
FIGURA 5.1: DIAGRAMA DE CLASES DE LOS CONTROLADORES DE LA PARTE DE LA APLICACIÓN	14
FIGURA 5.2: DIAGRAMA DE CLASES DE LOS <i>MAPPERS</i> .....	16
FIGURA 5.3: DIAGRAMA DE CLASES DE LOS <i>DBTABLES</i> .....	16
FIGURA 5.4: DIAGRAMA DE CLASES DE LOS <i>OBJECTS</i> .....	17
FIGURA 5.5: DIAGRAMA DE CLASES DE LOS CONTROLADORES DE LA API RPC.....	18
FIGURA 5.6: DIAGRAMA DE CLASES DE LA CAPA DE NEGOCIO .....	18
FIGURA 5.7: INTERFAZ DE BIENVENIDA .....	20
FIGURA 5.8: INTERFAZ DE DEPORTES.....	20
FIGURA 5.9: INTERFAZ DE EDICIÓN DE PERFIL PERSONAL.....	21
FIGURA 5.10: INTERFAZ DE EDICIÓN DE PERFIL DEPORTIVO .....	22
FIGURA 5.11: INTERFAZ DE EVENTOS.....	22
FIGURA 5.12: INTERFAZ DE PUBLICACIONES.....	22
FIGURA 5.13: INTERFAZ DE FICHA PERSONAL Y DEPORTIVA.....	23
FIGURA 5.14: INTERFAZ DE AMISTADES.....	23
FIGURA 5.15: INTERFAZ DE PERFIL DE USUARIO .....	24
FIGURA 5.16: INTERFAZ DEL TIMELINE .....	24
FIGURA 5.17: INTERFAZ DE VALORACIÓN .....	25
FIGURA 5.18: INTERFAZ DE LOGIN DE ADMINISTRADOR .....	25

FIGURA 5.19: INTERFAZ DE DASHBOARD .....	26
FIGURA 5.20: INTERFAZ DE CONTROL DE USUARIOS .....	26
FIGURA 5.21: INTERFACES DE CONTROL DE EVENTOS Y PUBLICACIONES.....	27

# Glosario

**PHP:** Lenguaje de programación del lado del servidor diseñado para el desarrollo web.

**Javascript:** Lenguaje de programación interpretado utilizado, principalmente, en el lado del cliente para el desarrollo web dinámico.

**Jquery:** Biblioteca de Javascript.

**HTML:** *HyperText Markup Language*. Lenguaje de marcado para elaboración de páginas web.

**SQL:** *Structured Query Language*. Lenguaje declarativo de acceso a base de datos relacionales.

**HTTP:** *Hypertext Transfer Protocol*. Protocolo usado para acceder a la web.

**AJAX:** *Asynchronous JavaScript And XML*. Técnica de desarrollo web que implementan aplicaciones interactivas.

**JSON:** *JavaScript Object Notation*. Estándar basado en texto plano para el intercambio de información.

**Framework:** Estructura de soporte definida utilizada para organizar y desarrollar un proyecto software.

**API:** *Application Programming Interface*. Interfaz de programación entre componentes software.

**RPC:** *Remote Procedure Call*. Protocolo utilizado en aplicaciones distribuidas cliente-servidor en el que no hay que preocuparse por la comunicación entre ambos.

**MVC:** *Model - View - Controller*. Patrón de diseño software que propone separar el código de los programas por responsabilidades.

**LAMP:** *Linux – Apache – MySQL – PHP*. Sistema creado por el conjunto de aplicaciones libres Linux, Apache, MySQL y PHP.

**Front-end:** Parte del software que interactúa con los usuarios.

**Back-end:** Parte del software que procesa la entrada desde el front-end.

## **1. Objetivo**

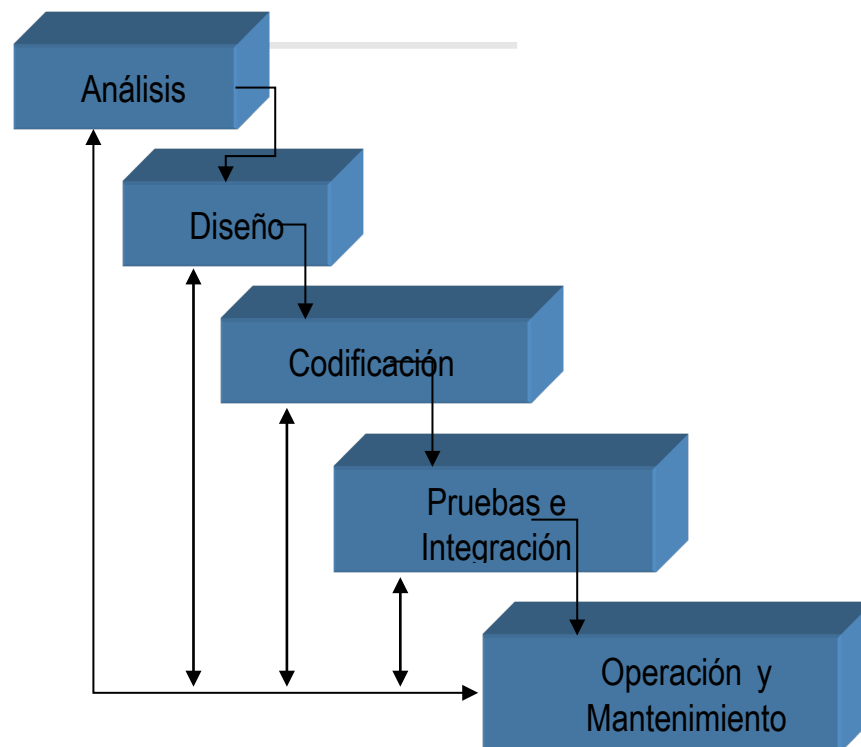
El objetivo de este proyecto es el diseño e implementación de una aplicación web para crear y gestionar una red social deportiva en Internet. El propósito principal de la aplicación es facilitar el encuentro entre usuarios con perfiles deportivos en común a través de organización de eventos y creación de publicaciones.

Por tanto, este proyecto conlleva la definición, implementación y validación de un sistema informático web cuyo propósito es gestionar una red social deportiva en Internet para facilitar el encuentro entre todo tipo de aficionados del deporte.

Esta herramienta, resultado de este proyecto, se visualizará a partir de una interfaz gráfica compatible con todos los navegadores modernos y se estructurará en dos módulos independientes con el fin de crear un sistema cuyos servicios sean compatibles con otro tipo de soportes como, por ejemplo, dispositivos móviles. Además, se pretende acercar la aplicación al mayor número de clientes posibles ya que un aficionado al deporte no tiene por qué tener conocimientos medio-avanzados sobre navegación web y, por tanto, se pretende que la interacción del usuario con el sistema sea lo más simplificada posible. Por otro lado, se quiere dar a conocer a la sociedad una serie de deportes menos extendidos para ampliar el rango de deportes practicados entre la población.

## 2. Ciclo de vida de desarrollo

Para el desarrollo de este proyecto se ha decidido seguir un ciclo de vida en cascada con refinamiento. Se ha elegido este ciclo de vida ya que, el tamaño de la aplicación no es grande, con lo que se puede controlar con cierta facilidad, pero los requisitos pueden cambiar y, no sabemos seguro, el resto de las fases puede sufrir alguna modificación con el fin de mejorar funcionalidades. Por otra parte, no requiere de entregas frecuentes ni de sistemas parciales que estén operativos antes de la finalización del desarrollo. Tampoco es adecuada una metodología ágil porque es una herramienta para usuarios potenciales que no se conocen. Así, se elige el ciclo de vida de cascada con refinamiento por pasos que ofrece la posibilidad de realizar diferentes cambios de forma controlada en el conjunto de fases durante todo el ciclo de vida software, como se muestra en la figura 2.1.



**Figura 2.1: Ciclo de vida de la aplicación**

A continuación, se detallan las tareas realizadas en cada una de las fases del desarrollo:

### Análisis

- ❖ Definición de objetivos y funcionalidades.
- ❖ Especificación de requisitos funcionales y no funcionales.

### Diseño

- ❖ Diseño del diagrama Entidad-Relación (E-R) para la base de datos MySQL.
- ❖ Diseño de la interfaz de usuario.
- ❖ Diseño del diagrama de clases de la parte de la aplicación.
- ❖ Diseño del diagrama de clases de la API RPC.



### **Codificación**

- ❖ Implementación de la base de datos: Mysql
- ❖ Implementación del front-end del sistema: HTML + Javascript
- ❖ Implementación del back-end del sistema: PHP

### **Pruebas y validación**

- ❖ Diseño de los casos de prueba.
- ❖ Estrategia de pruebas.
- ❖ Evaluación del sistema tras el plan de pruebas.

### **3. Análisis**

#### **3.1. Funcionalidades principales**

Las funcionalidades más destacadas del proyecto son la organización de eventos deportivos entre usuarios, publicación de comentarios y noticias en muro personal, disponibilidad de una lista de actividades deportivas en las que los usuarios puedan inscribirse, creación de perfiles personales y deportivos por niveles de práctica, seguimiento entre usuarios para crear lazos en la comunidad y valoración de contenidos y deportes.

#### **3.2. Desglose en subsistemas**

Teniendo en cuenta las funcionalidades mencionadas, el sistema propuesto se puede desglosar en los siguientes subsistemas.

##### ***Organización de eventos deportivos***

Este sistema permite a los usuarios crear eventos sobre un determinado deporte con la finalidad de que el resto de usuarios se unan al mismo hasta completar todas las plazas disponibles.

El usuario creador del evento pasará a ser el organizador y, además, será incluido con el rol de participante. Dicho organizador deberá establecer unos parámetros informativos obligatorios como fecha del evento, lugar de encuentro, descripción, número máximo de participantes, etc. Dependiendo del número de participantes y la fecha actual, el evento puede encontrarse en 3 estados diferentes: abierto, completo o cerrado.

Los participantes adjuntos al evento pueden salirse en cualquier momento dejando su plaza libre para otros posibles usuarios. El organizador no puede abandonar el evento, únicamente tiene la potestad de cancelarlo, eliminándose de la web.

##### ***Publicaciones en tablón personal***

La aplicación permite la escritura de publicaciones, formadas por un título y una descripción, en el muro personal de cada usuario para ser vistas por el resto de la comunidad teniendo que ser indicado el deporte con el cual se relaciona la publicación.

A diferencia de los eventos, las publicaciones tienen únicamente carácter informativo para el resto de la red social.

##### ***Inscripción de deportes***

El sistema oferta más de 25 deportes para que los usuarios de la red puedan añadirlos a su perfil. Una vez inscrito un determinado deporte, el usuario tiene acceso a las dos funcionalidades anteriores, es decir, a todos los eventos y publicaciones relacionados con dicho deporte. En cualquier momento el cliente puede cancelar la inscripción a un deporte.

##### ***Perfiles personales y deportivos***

Al tratarse de una comunidad que tiene como fin conocer nuevos usuarios con deportes en común, el sistema necesita que los usuarios creen sus propios perfiles para dar a conocerse.

En primer lugar, es necesario establecer un perfil con los datos personales del usuario. Dichos datos se dividen en dos secciones:

- Datos obligatorios: necesarios para poder registrarse en el sistema y poder tener acceso al mismo.
- Datos opcionales: dirigidos a que otros usuarios puedan conocer mejor a un individuo.

Por otra parte, por cada deporte que un beneficiario inscribe, se tiene que crear un perfil deportivo individual en el que se informe de una serie de datos como nivel de juego, lugar de desarrollo, días y horas que se lleva a cabo su práctica, etc. De esta forma, se proporciona información muy detallada de cada actividad deportiva, simplificando la búsqueda de usuarios con gustos en común.

### ***Seguimiento de usuarios***

La web dispone de un sistema de amistades en el que un usuario puede 'seguir' a cualquier individuo de la comunidad y, a su vez, puede ser 'seguido' por otros.

De esta forma, si se tienen aficiones en común con un usuario y se quiere conocer que publica o que eventos organiza, en el timeline aparecerán las notificaciones del amigo en cuestión. La web dispone de una sección de amistades con un mayor número de opciones relacionadas con esta funcionalidad.

### ***Valoración de contenidos***

Se ha comprobado que un usuario se suele guiar por las valoraciones que realizan otros usuarios de la comunidad sobre diversos temas. Por consiguiente, el sistema permite realizar una valoración numérica de otros usuarios de la comunidad y de los deportes en los que se encuentre inscrito. Para ello, se utiliza una ventana con cinco estrellas.

Por otra parte, los eventos y publicaciones disponen de un sistema de 'MG' para ver que contenidos son más destacables. Un usuario solo puede insertar un 'MG' por cada contenido.

### ***Control de usuarios y contenidos***

El acceso al sistema se puede realizar como dos tipos de usuario: cliente y administrador. El administrador dispone de una serie de servicios para mantener la red social limpia de contenidos no apropiados, pudiendo incluso eliminar usuarios de la red.

## **3.3. Requisitos**

Una vez realizado el análisis inicial del sistema, su desglose en subsistemas y la definición del ciclo de vida, así como las necesidades de potenciales usuarios a los que se ha entrevistado, se van a enumerar los requisitos necesarios para dotar a la aplicación de todas las funcionalidades para que el sistema tenga un comportamiento correcto.

Los usuarios del sistema serán los administradores y el público en general, al que denominaremos comúnmente "usuarios".

**Requisitos Funcionales**

- RF. 1.** Un usuario podrá darse de alta en la red mediante un conjunto de datos personales obligatorios.
- RF. 2.** Los usuarios y administradores podrán desde puntos de entrada diferentes.
- RF. 3.** Cada usuario dispondrá de un perfil personal que podrá ser actualizado con una serie de datos personales opcionales.
- RF. 4.** El sistema ofertará una lista de deportes disponibles para los usuarios de la comunidad. La lista sólo podrá ser modificada por el administrador de la aplicación y no por otro rango de usuarios.
- RF. 5.** Cada usuario podrá añadir a su perfil cualquier deporte disponible ofertado por el sistema, pudiendo eliminarlo del mismo en todo momento. Será necesario indicar el nivel practicado para la inscripción de cada deporte.
- RF. 6.** Cada usuario dispondrá de un perfil deportivo por cada deporte en el que se encuentre inscrito. Este perfil podrá ser actualizado con un conjunto de datos opcionales.
- RF. 7.** Los usuarios podrán ver el perfil personal y perfil deportivo del resto de usuarios de la red.
- RF. 8.** Cada usuario podrá 'seguir' a cualquier otro usuario y, a su vez, podrá ser 'seguido' por otros usuarios. Cuando un usuario X sigue a otro usuario Y, este último se añade a la sección amigos-siguiendo del usuario X y, al mismo tiempo, se añade a la sección de amigos-seguidores del usuario Y.
- RF. 9.** Cada usuario puede 'dejar de seguir' a otro dejando de ver sus notificaciones en el timeline y quedando eliminado de la lista de amistades.
- RF. 10.** El sistema recomendará usuarios que no se encuentren enlazados para facilitar el encuentro entre individuos con características similares tanto personales y como deportivas.
- RF. 11.** El sistema dispondrá de una sección de amistades con un conjunto de filtros para ver a los usuarios de la red: siguiendo, seguidores y recomendaciones/público.
- RF. 12.** Cada usuario podrá crear un evento deportivo definiendo una serie de datos obligatorios (título, descripción, fecha y hora, lugar, número total de participantes y deporte al que pertenece) para que otros usuarios se unan al mismo. Estos datos podrán ser actualizados en cualquier momento por el organizador, es decir, el usuario creador del evento.
- RF. 13.** Cualquier usuario no organizador de un evento podrá unirse al mismo para participar siempre que no esté el cupo de personas completo o la fecha del evento sea menor que la fecha actual. Dicho usuario podrá salirse del evento en cualquier momento.
- RF. 14.** Cada usuario podrá crear publicaciones en su tablón personal indicando siempre al deporte a la que pertenece. Esta publicación podrá ser actualizada por el mismo usuario.
- RF. 15.** Todo usuario podrá borrar contenido de su tablón, tanto eventos como publicaciones.

- RF. 16.** El sistema permitirá hacer un filtrado de eventos y publicaciones en el timeline del usuario y en el tablón personal.
- RF. 17.** El sistema dispondrá de un método de evaluación por estrellas para que los usuarios puedan valorar deportes en los que se encuentren inscritos.
- RF. 18.** El sistema dispondrá de un método de evaluación MG para que los usuarios puedan valorar eventos y publicaciones de otros usuarios.
- RF. 19.** Cada usuario podrá darse de baja de la aplicación, perdiendo todo el contenido almacenado desde que se registró en la web.
- RF. 20.** Los administradores de la red podrán eliminar los siguientes contenidos de la red: cuentas de usuarios, eventos y publicaciones.

### ***Requisitos no funcionales***

A continuación, se describen los requisitos no funcionales del sistema:





- RNF. 1.** La aplicación dispondrá de una interfaz gráfica sencilla e intuitiva para que la interacción de los usuarios sea lo más fácil posible.
- RNF. 2.** Todas las páginas que conforman la web contienen una cabecera con diferentes opciones y un pie de página informativo.
- RNF. 3.** Los datos mostrados a los administradores tendrán un diseño más complejo en forma de listas y tablas para estructurar la información de forma óptima.
- RNF. 4.** La interfaz mostrará mensajes de error explicativos si los contenidos o las respuestas del servidor son erróneas.
- RNF. 5.** El idioma disponible en el sistema será únicamente el castellano.
- RNF. 6.** Se podrá utilizar la aplicación en cualquier navegador moderno, exceptuando Internet Explorer, como Chrome, Mozilla u Opera.
- RNF. 7.** Se requiere identificación de usuario con contraseña para acceder al sistema.
- RNF. 8.** La sesión permanecerá abierta mientras no se cierre sesión de la misma.
- RNF. 9.** La base de datos tiene un espacio limitado (200 GB) y podrá procesar un máximo de peticiones simultáneas. El usuario máster del sistema puede ampliar el tamaño de ambos si fuese necesario.
- RNF. 10.** La aplicación guarda todos los cambios, realizando un backup nada más llevarlos a cabo, con el objetivo de tener los datos actualizados en caso de que otro usuario esté utilizando la plataforma.

## 4. Diseño

Tras establecer los requisitos del sistema en el apartado anterior, en el siguiente punto se va a definir la arquitectura general del sistema y, a continuación, se va a exponer en detalle el diseño de la base de datos, la interfaz gráfica y la API RPC.

### 4.1. *Arquitectura general*

En la figura 4.1 se muestra la estructura de la aplicación. A continuación, se exponen los componentes principales que la componen:

-  Una base de datos que cumple con el modelo relacional en la que se almacena toda la información necesaria para posteriores usos como, por ejemplo, cuentas de usuarios, perfiles deportivos de los clientes, datos relacionados con los deportes ofertados, eventos y publicaciones, etc.
-  Una API basada en el protocolo RPC (Remote Procedure Call) que posee un conjunto de operaciones que dota de funcionalidad a la aplicación. Este módulo dispone del único punto de acceso a la base de datos del sistema. Está formada por controladores, modelos y una capa de negocio intermedia entre ambos. La aplicación accede a la API por mediante llamadas AJAX y, dicha API, devuelve las respuestas en formato JSON.
-  Un módulo que contiene la parte de la aplicación. Esta parte es totalmente independiente de la API y no tiene conexión directa con la base de datos. Está formada por controladores y vistas. La funcionalidad de dichos controladores es mínima ya que únicamente proporcionan el código de las vistas y gestionan el paso de variables entre páginas. El acceso a este módulo se realiza mediante el protocolo HTTP. El módulo está formado por la aplicación cliente y la aplicación de administrador.
-  Por último, la interfaz de usuario permite tanto al cliente como a los administradores interactuar con el sistema para realizar búsquedas de usuarios con gustos comunes, añadir deportes a un perfil deportivo, organizar eventos, escribir publicaciones sobre actividades deportivas, controlar el contenido de la web, etc.

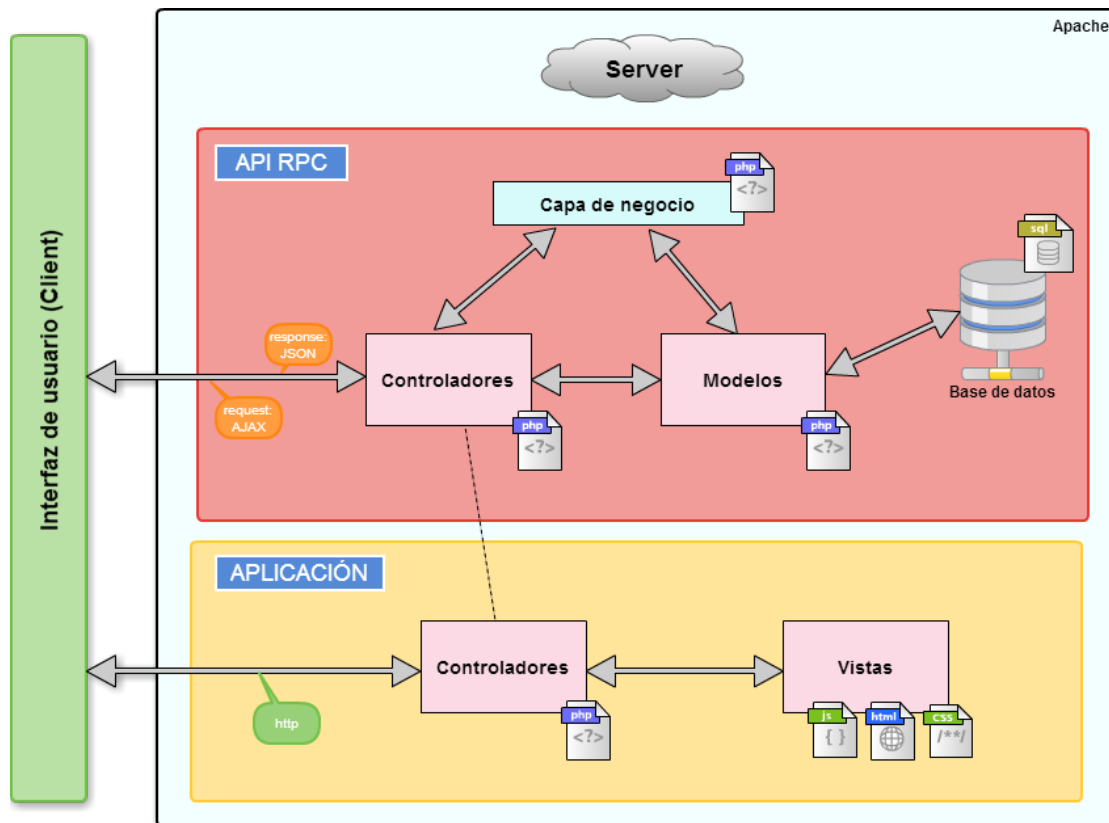


Figura 4.1: Diagrama de la arquitectura general de la aplicación

## 4.2. Interfaz de usuario

El diseño de la interfaz de usuario se ha realizado partiendo del objetivo de que la mayor parte de clientes tengan una interacción lo más intuitiva posible con la aplicación, es decir, se ha realizado un diseño muy orientado hacia el usuario. Los principios fundamentales por los que se ha guiado el diseño son los siguientes: familiaridad del usuario, consistencia, mínima sorpresa y recuperabilidad.

## 4.3. Base de datos

Para que persistan los datos en la aplicación se ha utilizado una base de datos relacional que almacena todos los datos relacionados con deportes, niveles de práctica, usuarios, eventos, publicaciones y valoraciones. Aunque se ha seguido un ciclo de vida en cascada que permite posibles modificaciones en fases anteriores, se ha realizado un diseño robusto de la base de datos debido a la importancia que adquiere el sistema de almacenamiento en el proyecto.

En la figura 4.2 se muestra el diseño del diagrama E/R correspondiente a la base de datos. A continuación, se describe brevemente cada una de las entidades y relaciones que componen la base de datos:

## Entidades

- ✚ **USUARIOS:** representa a los clientes que se han registrado en la aplicación. La contraseña de usuario está encriptada en md5.
- ✚ **DEPORTES:** representa los deportes que ofrece el sistema. Contiene la información básica del mismo.
- ✚ **NIVELES:** representa un determinado nivel de práctica que está asociado a un deporte único. Un deporte debe contener al menos un nivel para estar disponible en el sistema.
- ✚ **EVENTOS:** es la entidad encargada de guardar los eventos organizados por un usuario sobre un determinado deporte y ofrecidos para un conjunto de niveles [1-max]. Otros usuarios no organizadores pueden unirse al evento a través de otra relación.
- ✚ **PUBLICACIONES:** es la entidad encargada de guardar las publicaciones escritas por los usuarios sobre un determinado deporte.
- ✚ **PROVINCIAS:** es una tabla que contiene las 52 provincias del estado español distinguidas por un identificador único. Está disponible en las fuentes oficiales del estado, aunque dicha tabla es un volcado a sql ya que se ofrecen en formato csv.
- ✚ **MUNICIPIOS:** es la entidad que contiene todos los 1.552 municipios del estado español distinguidos por un identificador único y por el DC. También se encuentra disponible en las webs oficiales del estado. Es un volcado de dichas fuentes ya que se ofrecen en formato csv.
- ✚ **TUTORIALES:** es una tabla que almacena el contenido de una guía de ayuda que se ofrece al usuario para aprender a interactuar con el sistema.
- ✚ **ADMINISTRADORES:** es la entidad que contiene los administradores que tienen acceso a las operaciones de control y mantenimiento de la red. La contraseña de administrador está encriptada en md5.



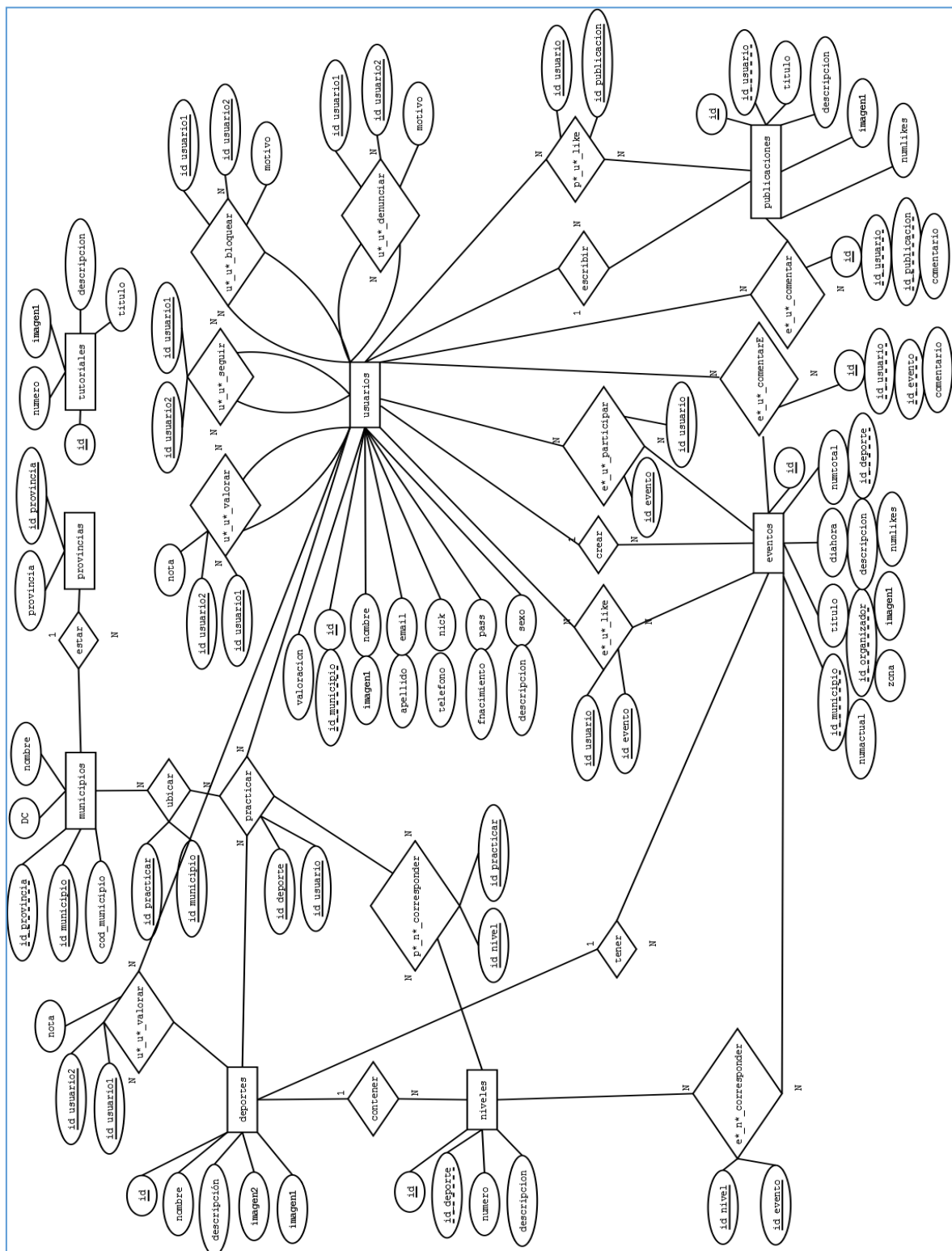














Figura 4.2: Diagrama E/R de la base de datos

## Relaciones N-N

-  **USUARIOS\_DEPORTES\_PRACTICAR:** es la relación encargada de almacenar los deportes en los que se inscribe cada usuario del sistema. Dicha relación es la más destacada de la base de datos ya que de la misma extienden otras dos relaciones.
-  **PRACTICAR\_NIVELES\_CORRESPONDER:** representa una relación que vincula la relación anterior con la entidad que almacena los niveles. De esta forma, cada usuario que se inscribe en la práctica de un deporte, puede indicar los niveles que le corresponden de dicho deporte.
-  **PRACTICAR\_MUNICIPIOS\_UBICAR:** vincula la relación *practicar* con la entidad que contiene todos los municipios del estado. Por consiguiente, cada usuario inscrito en un deporte, puede indicar los municipios en los que suele realizar la práctica deportiva pudiendo indicar diferentes municipios para cada deporte.
-  **EVENTOS\_USUARIOS\_PARTICIPAR:** representa la relación de los usuarios que se han unido a un evento. El organizador del evento también se encuentra en esta tabla.
-  **EVENTOS\_NIVELES\_CORRESPONDER:** es una relación de los niveles de práctica que se han elegido para un evento.
-  **USUARIOS\_USUARIOS\_SEGUIR:** representa la relación entre usuarios que se siguen en la red. Un usuario puede seguir a otro sin que éste le siga.
-  **DEPORTES\_USUARIOS\_VALORAR:** es una asociación de un usuario que realiza una valoración sobre un deporte en el que se encuentra inscrito.
-  **EVENTOS\_USUARIOS\_VALORAR:** relación de un usuario con un evento que indica si a dicho usuario le gusta el evento.
-  **PUBLICACIONES\_USUARIOS\_VALORAR:** es una relación de un usuario con una publicación que indica si el usuario está interesado en la misma.
-  **USUARIOS\_USUARIOS\_BLOQUEAR:** representa una asociación de un usuario que bloquea a otro indicando un determinado motivo.
-  **EVENTOS\_USUARIOS\_COMENTAR:** es una relación de un usuario con el evento sobre el que ha realizado un comentario.

 **PUBLICACIONES\_USUARIOS\_COMENTAR:** es una relación de un usuario con la publicación sobre la que ha realizado un comentario.

#### **4.4. API RPC**

Un principio muy importante de la aplicación es su comportamiento fuertemente interactivo. Para ello, se usa una técnica de desarrollo web llamada AJAX (*Asynchronous JavaScript And XML*) en la que se realizan llamadas al servidor mientras el usuario sigue navegando por la misma página web sin necesidad de recargar la misma. Las respuestas de estas llamadas por parte del servidor se realizan, entre otros formatos soportados, en JSON.

El servidor debe estar preparado para realizar estas funcionalidades. Por este motivo, se ha diseñado una API llamada RPC totalmente independiente de la parte de la aplicación, es decir, ambos módulos podrían estar en máquinas diferentes sin compartir recursos y el sistema seguiría funcionando correctamente. RPC (*Remote Procedure Call*) es un protocolo que permite ejecutar código en una máquina remota sin tener que preocuparse por las comunicaciones entre ambos. La gran ventaja de este diseño es que dicho módulo puede ser utilizado, no solo por la aplicación web desarrollada, sino por otro tipo de dispositivos con diferente soporte. El diseño interno del módulo sigue el patrón MC (modelo controlador) incorporando una capa de negocio intermedia entre ambos.

## 5. Implementación

A continuación, se explica cómo se han implementado los diferentes componentes diseñados en el capítulo anterior. Se detalla el desarrollo de los módulos de mayor importancia indicando las herramientas utilizadas para su implementación.

### 5.1. Estructura del sistema

El sistema, como toda aplicación web, dispone de una parte cliente llamada front-end y de una parte servidora denominada back-end. Además, en cuanto a modularización, se dispone de una API RPC que dispone de la funcionalidad destacada del sistema y otro módulo que contiene la parte de la aplicación cómo vimos en la figura 4.1.

El front-end o interfaz es la parte que interactúa con los usuarios y su código corresponde a las vistas de la parte de la aplicación. Dichas vistas se han implementado en los lenguajes HTML, JS y CSS con la ayuda de Bootstrap y Backbone, frameworks de diseño web y de desarrollo JavaScript respectivamente. Por otra parte, el back-end o motor es la parte que procesa la entrada desde la interfaz y está formada por los controladores y modelos de la API y por los controladores del módulo de la aplicación. Se ha decidido implementar dicho motor en el lenguaje PHP ya que permite representar entidades como clases, está pensado para construir páginas web dinámicas y, además, está ampliamente extendido en desarrollo web.

Por último, destacar que la implementación del conjunto del sistema se ha realizado mediante el framework de desarrollo web PHP Zend. Esta herramienta cuenta con un gran número de utilidades como soporte para la conexión y manejo de la base de datos, diseño patrones de arquitectura MVC, interpretación de datos en formato JSON, solución de problema de seguridad como inyección SQL, etc. A continuación, se ha realizado un diagrama de clases UML que describe la estructura utilizada en la implementación del sistema.

#### 5.1.1. Controladores de la parte de la aplicación

En la figura 5.1 se muestra la clase que contiene los controladores de la interfaz de la aplicación de usuarios y administradores. Dichos controladores, solicitados mediante peticiones HTTP, se pueden denominar ‘vagos’ ya que, únicamente, proporcionan las vistas a la parte del cliente y realizan la lectura y escritura de los parámetros de entrada/salida. Por requerimiento de Zend, el nombre del controlador corresponde a la página web que se quiere obtener.

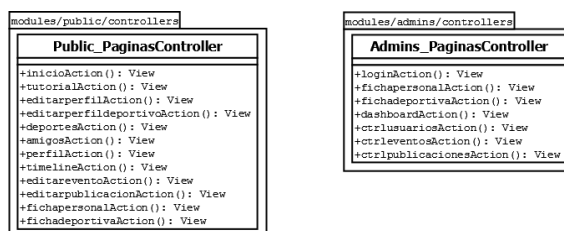


Figura 5.3: Diagrama de clases de los controladores de la parte de la aplicación

### 5.1.2. Conexión con la base de datos

El modelo es el encargado de realizar la conexión con la DDBB de la aplicación. Como se muestra en las figuras 5.2, 5.3, 5.4, el modelo está dividido en tres directorios:

- **Objects:** Este conjunto de clases son los contenedores de los objetos. Representan las entidades de la base de datos en la aplicación. Se podría prescindir de ellos, pero sirven para saber de qué tipo de elemento son los resultados de los *select* y qué propiedades tienen los mismos. La clase padre *Mapper* contiene un conjunto de funciones comunes a todos los objetos.
- **DbTables:** Es un grupo de clases que hay que crear obligatoriamente para seguir la estructura propuesta por Zend para implementar el modelo. Contiene un único atributo con el nombre de la base de datos y heredan de la clase primitiva de Zend llamada *Zend\_Db\_Table*.
- **Mappers:** Son las clases que contienen las funcionalidades del modelo. Necesitan las clases *DbTable* para poder ejecutar operaciones en la DDBB como *insert*, *delete*, *select*, etc. y las clases *Objects* para poder devolver los resultados con el tipo correspondiente. Dichas clases mapean las acciones de la DDBB en la aplicación. Se ha creado una clase padre llamada *Mapper* que proporciona un conjunto de operaciones que, entre otras funcionalidades, permite realizar *selects* muy robustos de una manera óptima. De esta forma, se consigue realizar diferentes *selects* con múltiples condiciones a través de una única función denominada *listAll*.

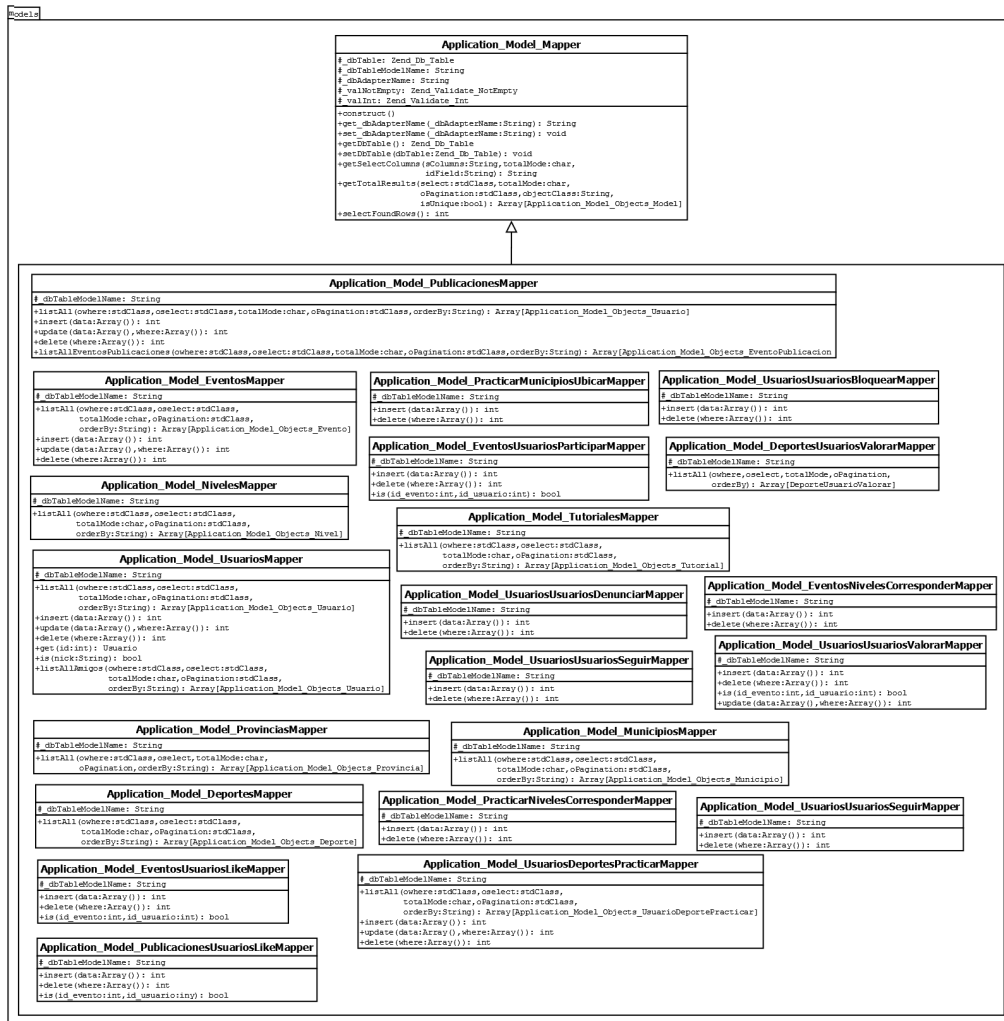


Figura 5.4: Diagrama de clases de los mappers

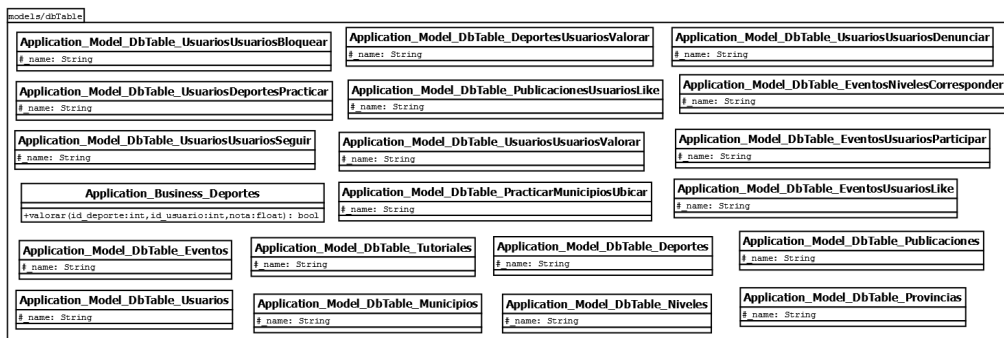
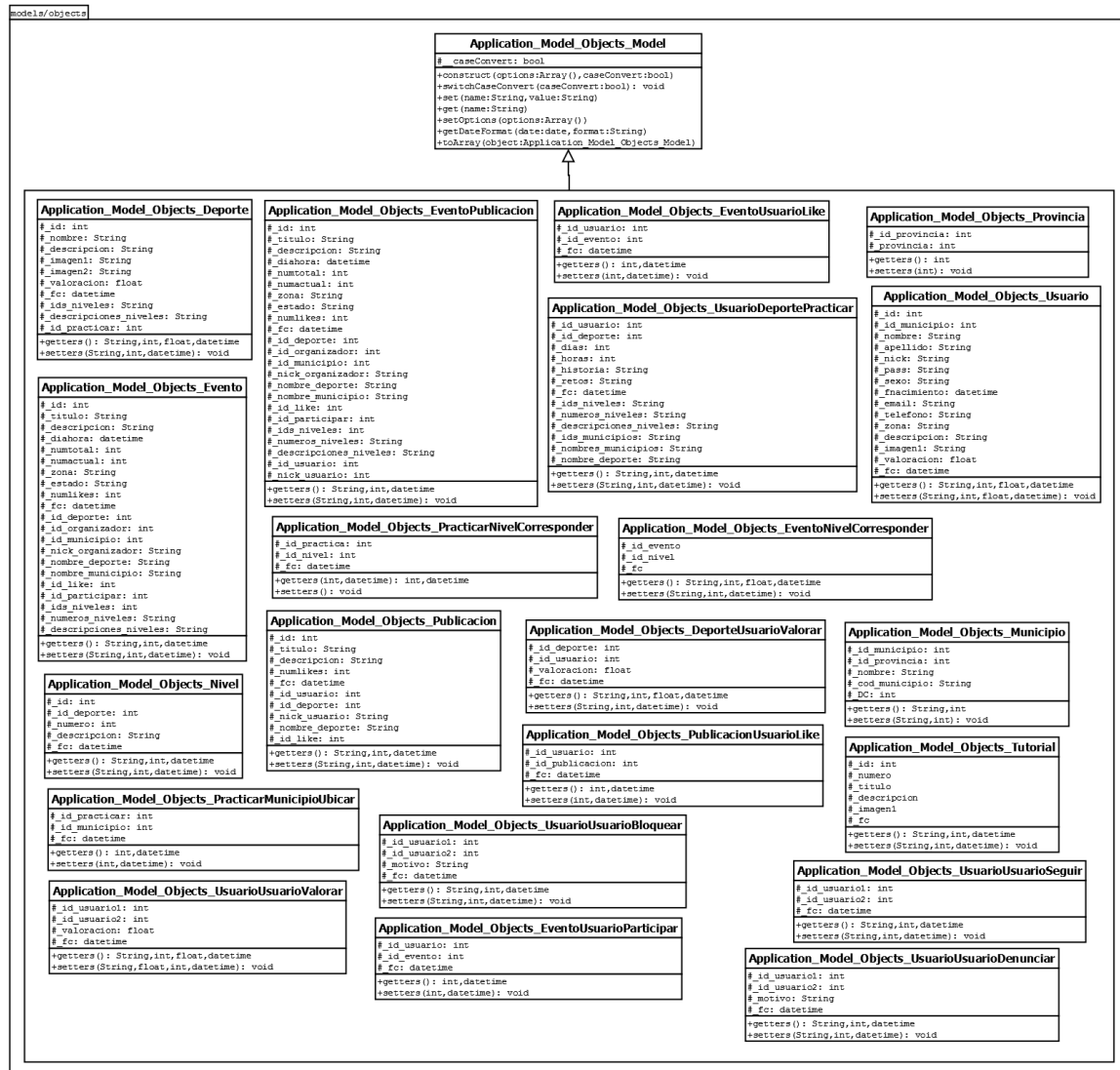


Figura 5.5: Diagrama de clases de los dbTables

Figura 5.6: Diagrama de clases de los *objects*

### 5.1.3. Controladores de la API RPC

En la figura 5.5 se puede observar el diagrama de clases correspondiente a los controladores de la API RPC. Los controladores proporcionan la respuesta en formato JSON y, aunque no son restrictivos a otros protocolos como HTTP, nuestra aplicación cliente solicita los controladores mediante llamadas AJAX. Cada uno de sus métodos corresponde a acciones a realizar en el modelo o en la capa de negocio dotando de funcionalidad a la aplicación. El conjunto de clases hereda de una clase padre llamada *Services\_RpcController* que es la encargada de realizar el envío de datos al cliente a través de la función *sendResponse*.

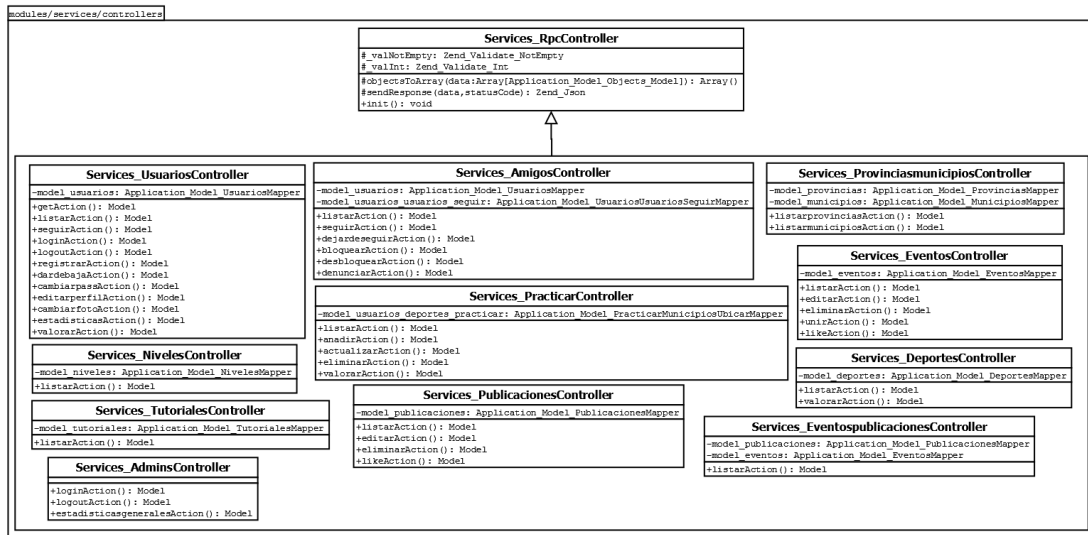


Figura 5.7: Diagrama de clases de los controladores de la API RPC

#### 5.1.4. Capa de negocio

En la figura 5.6 se muestran las clases correspondientes a la capa de negocio de la aplicación utilizada para separar la lógica de negocio de la de diseño. Se utilizan dichas clases para programación de algoritmos y para operaciones que requieren más de un acceso a la base de datos. El nombre de la clase se corresponde con el controlador con el que está relacionado. La capa de negocio es una capa intermedia entre los controladores y el modelo de la API pero no es necesario que siempre se pase por ella para el intercambio de datos.

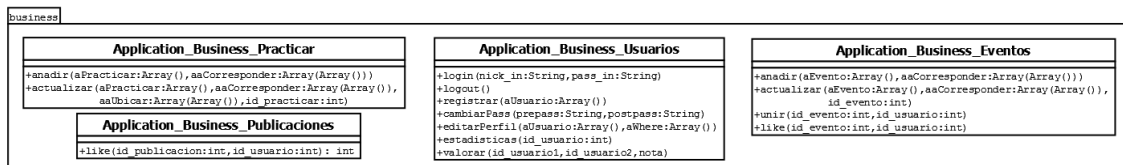


Figura 5.8: Diagrama de clases de la capa de negocio

### Base de datos

La base de datos, mostrada en un diagrama entidad-relación en la figura 4.2 del capítulo anterior, se ha implementado mediante el sistema de gestión de bases de datos relacional MySQL. Para poder implementar dicha base de datos se ha realizado un proceso de normalización sobre la misma para transformar el modelo entidad-relación previamente diseñado en un modelo relacional.

### Interfaz de usuario

La interfaz de esta aplicación está implementada para navegadores web y las vistas por las que está formada son ficheros HTML, JS y CSS: HTML permite mostrar el contenido de la página; JS

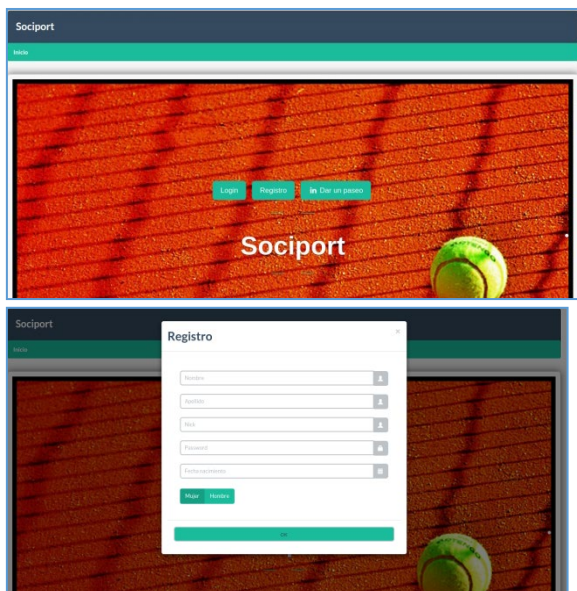


permite implementar la interfaz de forma dinámica y, además, es la encargada de hacer la carga de datos obtenidos de la base de datos y de imágenes; CSS permite crear la presentación de la página. La interfaz se ha implementado con la ayuda del framework de diseño *Bootstrap 3* y del framework Javascript *Backbone*.

A continuación, se muestra y se realiza una breve descripción de las interfaces que componen el sistema en el rol de cliente:

## 5.2. Inicio de la aplicación

La página de inicio, mostrada en la figura 5.7, es la interfaz que se encuentra el usuario al acceder a la aplicación y contiene un breve resumen de los servicios del sistema. Los botones de la primera imagen permiten al usuario realizar el login o el registro en la app mediante un modal que se abre en la propia ventana. El tercer botón permite navegar a otra página que contiene una guía sobre la aplicación. Todas las páginas que conforman la web contienen una cabecera con diferentes opciones y un pie de página informativo.



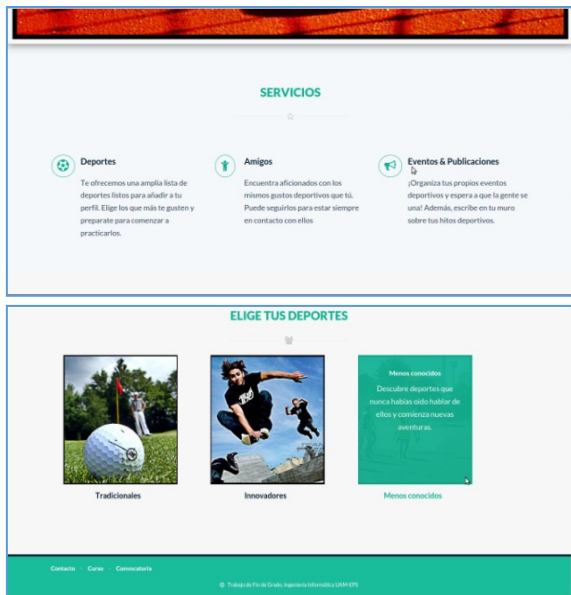


Figura 5.9: Interfaz de bienvenida

### 5.2.1. Interfaz de deportes

En la figura 5.8 se muestra la interfaz de deportes que contiene una lista con todos los deportes disponibles en la web. Se puede seleccionar cada deporte de la lista izquierda y se cargarán los datos del mismo en la vista de la mitad derecha de la página. Dichos datos están formados por una imagen, un cuadro con la descripción del deporte y una lista de checkboxes para elegir los niveles de práctica. En la parte inferior se encuentra un botón para añadir cada deporte al perfil del usuario.

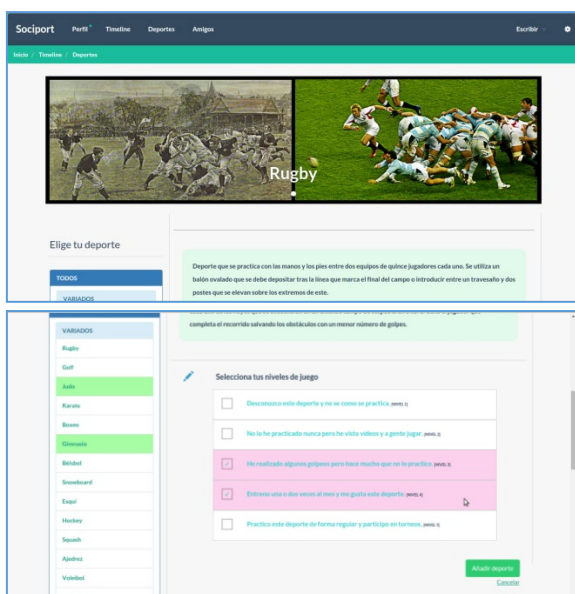


Figura 5.10: Interfaz de deportes

## 5.2.2. Interfaz de edición de perfiles

### Edición de perfil personal

Al registrarse en la aplicación el usuario tiene que proporcionar una serie de datos personales obligatorios. Una vez registrado y logueado, el cliente tiene la opción de modificar dichos datos y, además, puede proporcionar unos datos adicionales para completar su perfil personal. Para ello, se emplea un conjunto de inputs, selects, textboxes y un cuadro para cambiar la imagen de perfil. En la figura 5.9 se muestra dicha página:

Figura 5.11: Interfaz de edición de perfil personal

### Edición de perfil deportivo

Por cada deporte en el que se inscribe un usuario se pueden proporcionar una serie de datos adicionales como municipios, horas y días de practica, etc. además de modificar los niveles de practica elegidos al añadir el deporte. Para ello, como se aprecia en la figura 5.10, se dispone de una vista a la izquierda de la imagen con la lista de deportes añadidos y a la derecha un conjunto de checkboxes, selects y textboxes para rellenar dicha información.

Figura 5.12: Interfaz de edición de perfil deportivo

### 5.2.3. Interfaz de eventos y publicaciones

#### Eventos

Para crear nuevos eventos deportivos o para modificar uno ya creado se dispone de la interfaz mostrada en la figura 5.11. Dicha interfaz está formada por elementos de entrada de datos similares a los explicados en las anteriores interfaces.

Figura 5.13: Interfaz de eventos

#### Publicaciones

Las publicaciones, al igual que los eventos, utilizan la misma interfaz para crear una nueva publicación que para modificar una ya generada. En la figura 5.12 se muestra la implementación de dicha interfaz:

Figura 5.14: Interfaz de publicaciones

### 5.2.4. Interfaz de fichas personales y deportivas

Para poder ver de forma detallada la información personal y deportiva de otros usuarios se dispone de dos interfaces con contenido informativo. Están formadas únicamente por tablas y, en el caso de la ficha deportiva, la primera tabla contiene enlaces a tablas inferiores de cada deporte. En la figura 5.13 se muestran las dos interfaces:

**DATOS PERSONALES**

Foto	
Nombre	Vicente
Apellido	Torreal
Fecha de nacimiento	03-04-1991
Sexo	Hombre
Ubicación	Madrid
Email	vicente@gmail.com
Telefono	622209602
Fecha de alto	15-09-2023 10:38:03
Validación de otros usuarios (0-10)	4

**Datos deportivos**

Deportes practicados
Tenis
Padel
Running
Golf
Autos

**Tenis**

Victorias	45
Derrotas	10
Partidos ganados	100
Partidos perdidos	20
Partidos empatados	5
Partidos no jugados	10
Partidos en disputa	2

Figura 5.15: Interfaz de ficha personal y deportiva

### 5.2.5. Interfaz de amistades

La página de amistades está formada por un *nav* en el que se puede elegir el tipo de amigos a listar en la vista inferior. Cada amigo está formado por un cuadro con una imagen más botones de opciones. En la parte inferior se dispone de un paginador para ir mostrando los amigos en una cantidad definida previamente y así no generar una lista muy alta de datos. En la figura 5.14 se puede ver el resultado:

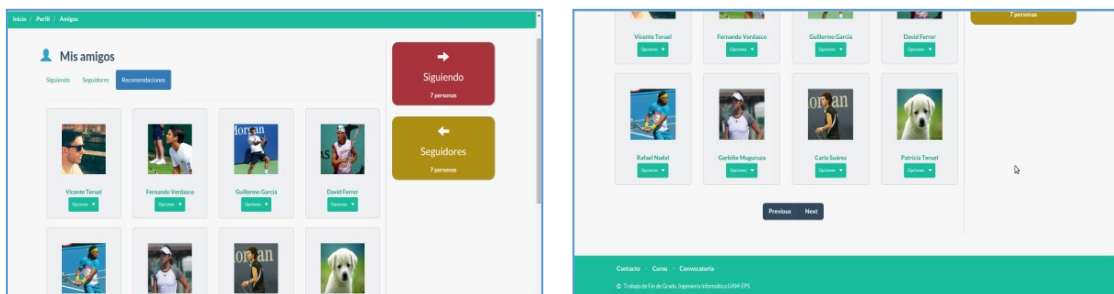


Figura 5.16: Interfaz de amistades

### 5.2.6. Interfaz del perfil de usuario

La interfaz perfil es la página principal de cada usuario. En ella se encuentran enlaces al resto de páginas de la web como editar perfiles, amigos, fichas, etc. Entre sus componentes destacamos la imagen del usuario, un *nav* acompañado de un *select* para filtrar el listado de los eventos y publicaciones del propio usuario mediante un paginador (la lista generada será muy extensa; en la figura 5.15 es un caso de prueba). Si el usuario que accede al perfil es el propio usuario se verán todas las opciones disponibles, pero si es el perfil de otro usuario no estarán disponibles algunas opciones como editar perfiles, borrar contenidos, etc.

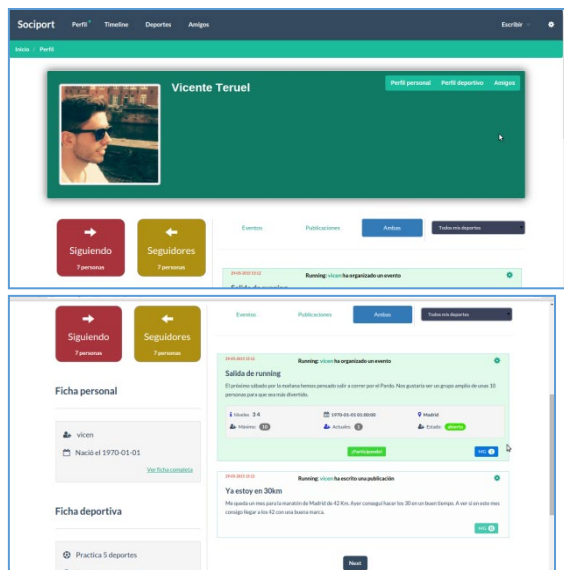


Figura 5.17: Interfaz de perfil de usuario

### 5.2.7. Interfaz del timeline

La interfaz timeline es muy similar al filtrado de eventos y publicaciones explicados en el punto anterior para el perfil del usuario. La diferencia es que en este listado se encuentran los eventos y publicaciones del resto de usuario de la comunidad: en primer lugar, los usuarios que se encuentre siguiendo el propio usuario y después el resto de usuarios. Está formado por un *nav* y un *select* para el filtrado y por los cuadros de eventos y publicaciones correspondientes.

Se puede comprobar también como están formados los cuadros de eventos y publicaciones. Ambos están compuestos por un título, una descripción, botón de *MG* y, en el caso de los eventos, de un cuadro interior con información detallada con múltiples despliegues y un botón para unirse al evento. En la figura 5.16 se puede ver dicha interfaz:

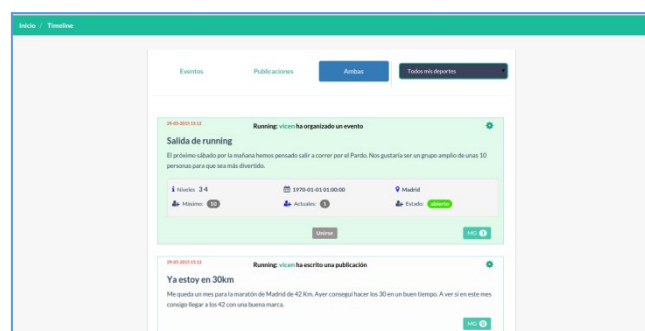
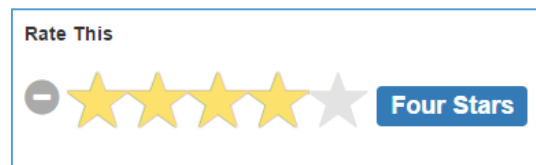


Figura 5.18: Interfaz del timeline

### 5.2.8. Otras interfaces

Hay una página que contiene una guía de aprendizaje de cómo utilizar la aplicación. Está formada por *carousel* que contiene capturas del sistema y un cuadro en el que se describe cada sección.

Además, como se ha visto en alguna de las anteriores interfaces, se dispone de un cuadro de estrellas para realizar valoraciones de deportes. En la figura 5.17 se puede ver la interfaz de valoraciones.

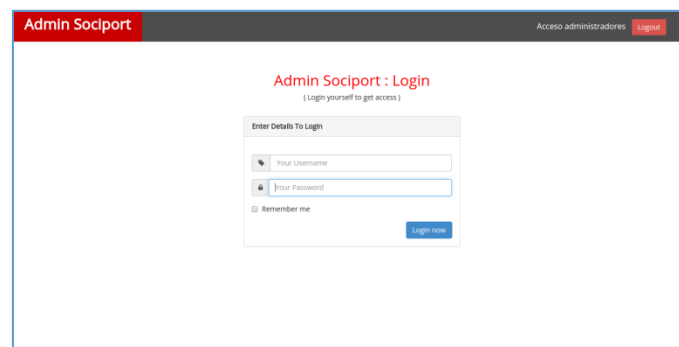


**Figura 5.19: Interfaz de valoración**

En los siguientes apartados se muestran las interfaces del rol administrador y se realiza una breve explicación de las mismas:

### **5.3. Login de administrador**

La página de login, mostrada en la figura 5.18, es la interfaz que se encuentra el encargado de mantener la red al acceder a la aplicación. Está formada por un formulario para autenticar a un administrador del sistema.



**Figura 5.20: Interfaz de login de administrador**

#### **5.3.1. Dashboard**

La interfaz dashboard es la página principal de la aplicación para administradores. Contiene un resumen de uso de la aplicación y una lista de opciones para acceder a los siguientes servicios: control de usuarios, control de eventos y control de publicaciones. En la figura 5.19 se puede ver la interfaz:

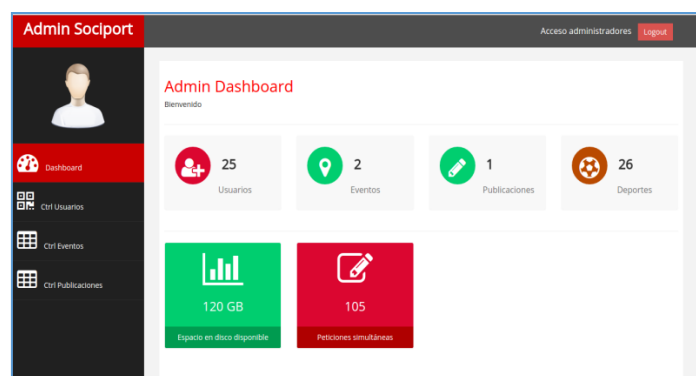


Figura 5.21: Interfaz de dashboard

### 5.3.2. Interfaces de control

#### Control de usuarios

En la interfaz de control de usuarios, mostrada en la figura 5.20, se listan los clientes registrados en el sistema. Se utiliza una tabla ordenada por fecha de actualización que contiene la id del usuario y dos links que acceden a las fichas personales y deportivas del usuario. Dichas fichas son similares a las mostradas en la figura 5.13. Se tiene la opción de eliminar el usuario desde la propia tabla.

# Admin Sociopt

Acceso administradores

Login

Dashboard

CHF Usuarios

CHF Eventos

CHF Publicaciones

## Tabla usuarios

Completa

Tabla avanzada admin

10 records per page

Search

B	U	Perfil personal	Perfil deportivo	Fecha
1	Go D.P.	Go D.D.	29-05-2015 15:30	
8	Go D.P.	Go D.D.	29-05-2015 15:17	
5	Go D.P.	Go D.D.	29-05-2015 14:53	
9	Go D.P.	Go D.D.	29-05-2015 14:52	
6	Go D.P.	Go D.D.	29-05-2015 14:51	
7	Go D.P.	Go D.D.	29-05-2015 14:51	
4	Go D.P.	Go D.D.	29-05-2015 14:50	
34	Go D.P.	Go D.D.	27-05-2015 17:53	
36	Go D.P.	Go D.D.	27-05-2015 16:50	

Figura 5.22: Interfaz de control de usuarios

#### Control de eventos y publicaciones

Las interfaces de control de eventos y publicaciones utilizan una tabla ordenada por fecha de actualización para mostrar el título y descripciones que el usuario introduce en cada uno de ellos. El administrador tiene la opción de eliminar las entradas de la tabla individualmente. En la siguiente figura se muestran las interfaces:



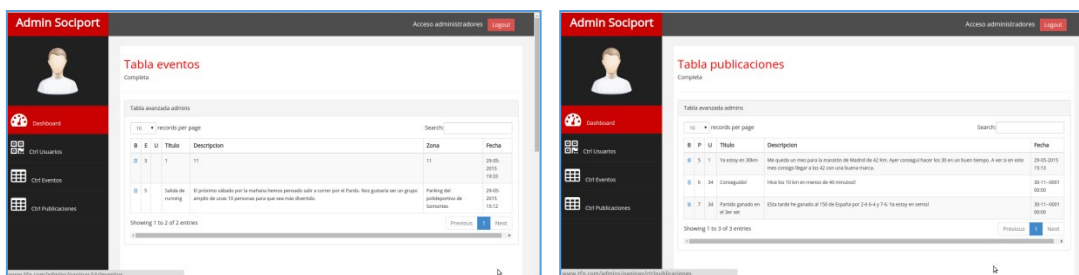


Figura 5.23: Interfaces de control de eventos y publicaciones

## 6. Pruebas

El objetivo de realizar una verificación del sistema es comprobar que todos los componentes que forman el sistema funcionan correctamente. Para llevar a cabo este proceso se ha creado un plan de pruebas formado por inspección de código, pruebas unitarias, de integración, de interfaz, de sistema y validación y de aceptación.

### 6.1. Verificación

#### 6.1.1. Estrategia de pruebas

En el sistema hay tres componentes claramente diferenciados: la base de datos, la parte de aplicación y la API RPC. Por este motivo, se han creado planes específicos para cada una de estas partes. Se ha considerado necesario realizar una etapa específica de prueba de la interfaz de usuario dada la importancia de este elemento en la aplicación. A continuación, se describe brevemente el plan para cada tipo de prueba para, posteriormente comentar la ejecución de la misma.

##### Inspección de código

El método de reconocimiento de código consiste en una lectura del mismo una vez finalizada la programación para detectar posibles errores a simple vista. Dicho método se ha utilizado para revisar cada una de las clases que forman los controladores de la parte de la aplicación y los controladores y modelos de la API RPC. Los métodos que han requerido de una revisión más minuciosa han sido los *listAll* de las clases del modelo ya que contienen un número muy alto de filtros para hacerlos reutilizables y la única *query* que se genera alcanza un tamaño considerable.

##### Pruebas unitarias

Las pruebas unitarias son las encargadas de comprobar cada módulo individualmente. Este tipo de prueba se ha realizado sobre la DDBB y el módulo RPC para determinar que los modelos y controladores funcionan correctamente y no tienen comportamientos erróneos. Los planes de pruebas unitarias de caja blanca y caja negra se han realizado con la herramienta DevTools disponible para Google Chrome.

##### Pruebas de integración

Tras realizar las pruebas unitarias se ha iniciado el plan de pruebas de integración. Estas pruebas tienen el objetivo de probar el conjunto de módulos que forman el sistema una vez unidos en bloque. Se ha realizado un plan escalable: en primer lugar, se ha comprobado el correcto funcionamiento de la parte cliente con el módulo de la aplicación, de la parte cliente con la API RPC, de la base de datos con los modelos de la API y de la parte de aplicación con la propia API; finalmente, se han realizado tests integrando todos los módulos.

##### Pruebas de interfaz gráfica

El objetivo de las pruebas de interfaz gráfica es comprobar que la visualización de cada una de las páginas web que forman el sistema se realiza de forma correcta y que todos los botones, elementos de entrada de datos, etc. funcionan de la forma esperada. Al tratarse de una página web con un alto contenido dinámico también se ha comprobado si la visualización y cambio de contenidos a partir de acciones de usuarios se realiza de forma óptima.

### 6.1.2. Desarrollo de las pruebas

En este apartado se describe cómo y cuándo se han realizado las pruebas detallando el procedimiento seguido y los resultados obtenidos para cada una de las estrategias anteriores.

#### Inspección de código

La inspección de código se ha realizado después de programar cada una de las clases PHP del servidor. Al ser un lenguaje interpretado y no disponer de una fase de compilación, esta prueba adquiere mayor relevancia. Un desarrollador y un programador ajeno al equipo de trabajo de esta aplicación han realizado una lectura minuciosa del código en la que se han detectado los siguientes fallos:

- Errores en la sintaxis de llamadas entre los métodos de los controladores listar y de los modelos listAll. Las funciones que hacen selects a la DDBB trabajan con los siguientes parámetros: oselect, owhere, opagination, totalMode y orderBy. Se detectaron fallos en la escritura y posterior lectura de los parámetros oselect y owhere de tipo stdClass en cuatro de estos métodos.
- Errores sintácticos en las queries realizadas en los métodos listAll del modelo. Se han detectado errores en la programación por las condiciones que impone Zend para realizar el modelaje de los selects en los mappers de eventos, publicaciones y usuarios\_practicar\_niveles.

#### Pruebas unitarias

Las pruebas de caja blanca se han realizado sobre los métodos listar de los controladores de la API RPC y de sus correspondientes mappers. El único error detectado en la prueba de los caminos independientes ha sido el siguiente:

- Se realizaban más de un join a la base de datos con el mismo alias para las tablas usuarios, eventos\_usuarios\_like y publicaciones\_usuarios\_like.

Las pruebas de caja negra se han elaborado sobre los controladores de la API RPC y de la base de datos. Se han introducido valores de entrada coherentes y otros límites para comprobar la salida de los mismos. A continuación, se describen los resultados:

- Las variables de tipo *datetime* no se almacenaban correctamente en la base de datos debido al formato establecido. Se ha optado por realizar el cambio de formato en los mappers.
- Se desbordaban *strings* en la base de datos ya que no se controlaba en el servidor el tamaño máximo disponible. Se ha comprobado el tamaño máximo en los controladores de la API.

## Pruebas de integración

Una vez acabado la programación de los módulos y haber pasado cada uno de ellos las pruebas unitarias, se han realizado las pruebas de integración. La integración de la parte del cliente con la parte de aplicación y de la API RPC con la DDBB no ha generado errores destacables.

Por otra parte, la integración de la parte del cliente con la API RPC sí que ha detectado fallos relevantes en el paso de información entre cliente-servidor ya que la estructura utilizada en los controladores amigos, usuarios, eventos y publicaciones no era la misma que la estructura seguida en los modelos y colecciones de Backbone de la parte del cliente.

Finalmente, se ha detectado fallos en la comunicación realizada con `Zend_Http_Client` entre los dos módulos servidores para la realización del login en la parte de la aplicación ya que la llamada no estaba bien definida.

## Pruebas de interfaz gráfica

Las últimas pruebas que se han realizado han sido las de interfaz gráfica. En primer lugar, se ha llevado a cabo un plan para intentar probar todas las posibles acciones de la interfaz. Se detectaron fallos en las cargas dinámicas de las páginas *editarperfildeportivo* y *editarperfilpersonal* y en las dimensiones de la imagen de *perfil* de usuarios.

## 6.2. Validación

El objetivo de validar el sistema es comprobar que los requisitos funcionales y no funcionales se cumplen satisfactoriamente.

### 6.2.1. Estrategia y desarrollo de pruebas de validación

Para validar el sistema se han comprobado si los requisitos previamente definidos se cumplían satisfactoriamente desde el punto de vista de un usuario final. La estrategia seguida ha sido la de colaborar con usuarios ajenos al desarrollo de la aplicación para que naveguen por la web como usuarios finales. Dichos usuarios disponían de conocimientos diferentes sobre navegación web, desde estudiantes de ingeniería informática hasta personas adultas que no suelen navegar por la red.

En primer lugar, se introdujeron usuarios en la red ficticios y se crearon una serie de eventos y publicaciones para que los usuarios que iban a probar la aplicación tuvieran otros usuarios anteriores con los que encontrarse. A cada usuario se le dio un tiempo de 20-25 minutos para interactuar con la aplicación, independientemente de sus conocimientos. Durante el conjunto de pruebas se disponía de dos tablas: en la primera se iban tachando los requisitos que se cumplían correctamente y, en la segunda, se anotaban los fallos encontrados por los usuarios y el requisito con el que estaba relacionado dicho error.

Los resultados obtenidos en esta primera vuelta han sido los siguientes:

- Requisitos funcionales: 18 de 20 cumplidos.
- Requisitos no funcionales: 9 de 10 cumplidos.

Tras realizar la corrección de los tres requisitos que no cumplían se volvió a realizar la misma prueba con los mismos usuarios, pero reduciendo el tiempo de la prueba a 10-15 minutos ya que los usuarios ya habían adquirido conocimientos sobre el funcionamiento de la web.

Los resultados finales de la segunda vuelta fueron satisfactorios y todos los requisitos pasaron a la primera tabla:

- Requisitos funcionales: 20 de 20 cumplidos.
- Requisitos no funcionales: 10 de 10 cumplidos.

### **6.2.2. Pruebas de aceptación**

Debido a que esta aplicación no está creada por encargo de ningún cliente, sino como herramienta para sacarla al mercado, las pruebas de aceptación recogen los comentarios realizados por un conjunto de usuarios finales que utilizaron la aplicación una vez concluido todo el desarrollo y pruebas.

Por ello, distintos usuarios con diferente nivel de navegación web han probado el sistema tras su validación. Todos ellos han considerado muy útil la usabilidad que aporta el sistema en comparación con otras herramientas con objetivos similares. Han señalado la facilidad con la que se pueden crear eventos y publicaciones y modificar los mismos sin necesidad de tener experiencia de uso con la aplicación. El cuadro de evento y el despliegue de su información ha sido el punto más destacado en cuanto a utilidad.

La interfaz gráfica ha sido evaluada favorablemente ya que todos han recalcado la facilidad de navegación que ofrece la web. En pocos minutos los usuarios fueron capaces de utilizar todos los servicios ofrecidos de forma intuitiva.

El sistema de recomendación por estrellas de los diferentes deportes ha sido señalado por los usuarios como una forma fácil y directa de conocer si el deporte ofertado puede ser de utilidad para ellos. Además, las descripciones de los deportes, las definiciones de sus niveles y las imágenes aportadas han sido destacadas por los usuarios probadores.

Finalmente, han aportado la posibilidad de crear un módulo de mensajería privada o de comentarios para facilitar la comunicación directa entre los usuarios del sistema. Dicha ampliación queda como posible línea futura de trabajo para próximas versiones.

Se considera, por tanto, que el sistema está listo para pasar a operación.