



TRABAJO FIN DE GRADO
Grado en Ingeniería Informática
ARENA: GESTIÓN DE ACTIVIDADES DEPORTIVAS
ORGANIZADAS POR USUARIOS DEPORTISTAS

Autor:

Víctor Monserrat Villatoro.

Directores:

Dr. Pedro Antonio Gutiérrez Peña, Dpto. de Informática y Análisis Numérico.

D. Sergio Gómez Bachiller, Aula de Software Libre.



UNIVERSIDAD DE CÓRDOBA

Certificado

Dr. Pedro Antonio Gutiérrez Peña, Profesor Titular del Departamento de Informática y Análisis Numérico en la Escuela Politécnica Superior de la Universidad de Córdoba y D. Sergio Gómez Bachiller, Coordinador del Aula de Software Libre de la Universidad de Córdoba

INFORMAN

Que han dirigido el *Trabajo Fin de Grado* de la titulación *Grado en Ingeniería Informática* denominado "*Arena: Gestión de actividades deportivas organizadas por usuarios deportistas*", realizado por D. Víctor Monserrat Villatoro en la Escuela Politécnica Superior de la Universidad de Córdoba, reuniendo, a su juicio, las condiciones establecidas en este tipo de trabajos.

Y para que conste, firmamos el presente informe en Córdoba a 1 de septiembre de 2017.

Dr. Pedro Antonio Gutiérrez Peña.

D. Sergio Gómez Bachiller.

Esta página se ha dejado vacía a propósito.

Licencias

Licencia de documentación

Los manuales de este *Trabajo Fin de Grado* están distribuidos bajo la licencia *GNU Free Documentation License* versión 1.3. Puede encontrar una copia de la licencia en la web de la *Free Software Fundation*: <http://www.gnu.org/copyleft/fdl.html>.

Licencia de código

El código de este proyecto está distribuido bajo la *GNU General Public License* versión 3.0. Puede encontrar una copia de la licencia en la web de la *Free Software Fundation*: <http://www.gnu.org/licenses/gpl-3.0.html>.

Esta página se ha dejado vacía a propósito.

"Nunca dejes que nadie te diga que no puedes hacer algo. Ni siquiera yo, ¿vale? Si tienes un sueño, tienes que protegerlo. Las personas que no son capaces de hacer algo te dirán que tú tampoco puedes. Si quieres algo ve por ello y punto."

- En busca de la felicidad.

Esta página se ha dejado vacía a propósito.

Agradecimientos

Parece que fuera ayer cuando cogí por primera vez el cercanías con destino Rabanales y son ya cuatro años los que han pasado desde que di comienzo a la que sería una de las mejores etapas de mi vida, la universitaria. Decidí estudiar el *Grado en Ingeniería Informática* con ilusión por aprender a crear páginas web, videojuegos o máquinas inteligentes. Mi mayor descubrimiento fue que todo aquello que yo quería aprender se hacía a través de algo que entonces no conocía, la programación, y que esta se transformaría en mi principal afición. Sabía que el camino no sería fácil pero las ganas por aprender superaron mis miedos por enfrentarme a lo desconocido y tras duros años de esfuerzo y trabajo parece que he conseguido el objetivo. Todo esto no habría sido posible sin todos vosotros y por ello os estaré siempre agradecido.

A los que dejasteis de ser compañeros para convertirse en amigos, por todos esos "ratitos" que me habéis regalado y todos los que aún nos quedan. Porque "esos ratitos son pa' mí", porque "unas veces se gana, y otras se aprende".

A los que amáis la informática, por enseñarme a enamorarme de ella. Por vuestra paciencia, por vuestra docencia.

A los que no tenéis miedo a la altura, porque más que "hormiguitas" somos un hormiguero. Porque "la UCO es el equipo de todos".

A los que también veis ese banco invisible, por ser únicos. Por todas nuestras aventuras, que algún día quedarán escritas.

A los que compartís mis apellidos y no necesariamente en vuestro nombre, porque sin vosotros nada es posible. A los que estáis conmigo y a los que, por desgracia, ya no. Siempre os llevaré conmigo familia. Ojalá fueseis eternos.

A ti crack, por haber dado todo lo que llevas dentro para que yo pudiera recoger tan solo una pequeña parte, por tus pomodoros. Espero que la vida te dé todo lo que mereces porque yo nunca seré capaz de devolverte todo lo que has hecho por mí.

A ti hermano, por mostrarme el camino a seguir. Por compartir todo conmigo, por cuidar de mí.

A ti mamá, por ser la mujer más maravillosa. Por traerme al mundo, por quererme así.

A ti papá, por no rendirte. Por ser un ejemplo para mí.

A ti Laura, por existir.

Esta página se ha dejado vacía a propósito.

Índice de contenidos

Introducción	25
Capítulo 1 Introducción.....	27
Capítulo 2 Definición del problema.....	29
2.1 Problema real	29
2.2 Problema técnico	30
Capítulo 3 Objetivos	35
Capítulo 4 Antecedentes.....	37
4.1 Hat-trick.....	37
4.2 Deporticket.....	37
Capítulo 5 Restricciones.....	39
5.1 Factores dato	39
5.2 Factores estratégicos.....	39
Capítulo 6 Recursos.....	45
6.1 Recursos humanos.....	45
6.2 Recursos hardware	45
6.3 Recursos software.....	46
Análisis.....	49
Capítulo 7 Especificación de requisitos	51
7.1 Requisitos de información	51
7.2 Requisitos funcionales.....	54
7.3 Casos de uso.....	57
7.4 Requisitos no funcionales	76
Capítulo 8 Análisis de la información.....	77
8.1 Descripción del modelo conceptual de datos	77
8.2 Descripción de las clases.....	78
Capítulo 9 Análisis funcional	79
9.1 Descripción arquitectónica.....	79
9.2 Descripción del comportamiento.....	80
Capítulo 10 Especificación de requisitos de la interfaz.....	95

10.1	Características generales de la interfaz	95
10.2	Controles de la interfaz	96
Diseño		113
Capítulo 11 Diseño del sistema		115
11.1	Organización de los componentes.....	115
11.2	Diagrama de despliegue: <i>backend</i>	115
11.3	Diagrama de despliegue: <i>frontend</i>	116
11.4	Esquema de seguridad del sistema	117
11.5	Identificación de la concurrencia	118
Capítulo 12 Diseño de datos		119
12.1	Diseño de la clase <i>User</i>	121
12.2	Diseño de la clase <i>Sport</i>	122
12.3	Diseño de la clase <i>Activity</i>	123
12.4	Diseño de la clase <i>Registration</i>	125
12.5	Diseño de la clase <i>Notification</i>	126
12.6	Diseño de la clase <i>Communication</i>	127
12.7	Modelo Entidad-Relación.....	128
Capítulo 13 Diseño de la interfaz		129
13.1	Interfaz de administración	130
13.2	Interfaz de usuario	143
Capítulo 14 Pruebas		151
14.1	Paquete <i>Users</i>	151
14.2	Paquete <i>Sports</i>	155
14.3	Paquete <i>Activities</i>	157
14.4	Paquete <i>Registrations</i>	164
14.5	Paquete <i>Applications</i>	166
14.6	Paquete <i>Invitations</i>	172
Capítulo 15 Conclusiones y futuras mejoras.....		179
15.1	Análisis de objetivos	179
15.2	Problemas encontrados y soluciones	180
15.3	Conclusiones.....	181
15.4	Futuras mejoras	181

Apéndices	185
Apéndice A1 Competencias	187
Apéndice A2 Manual del administrador	193
A2.1 Instalación del <i>backend</i>	193
A2.2 Instalación del <i>frontend</i>	199
A2.3 Desinstalación	202
Apéndice A3 Manual de usuario	203
A3.1 Administración.....	203
A3.2 Aplicación	211
Apéndice A4 Manual de desarrollador.....	221
A4.1 Guía de desarrollo	221
A4.2 <i>Endpoints</i>	223
Apéndice A5 Manual de código.....	227
A5.1 Introducción	227
A5.2 Requisitos previos	227
A5.3 Descripción modular.....	228

Esta página se ha dejado vacía a propósito.

Lista de figuras

Figura 7.1 Casos de uso	57
Figura 8.1 Análisis del modelo de clases	77
Figura 9.1 Diagrama de paquetes	80
Figura 9.2 Diagrama de creación de deportes	81
Figura 9.3 Diagrama de listado de deportes	81
Figura 9.4 Diagrama de actualización de deportes	82
Figura 9.5 Diagrama de borrado de deportes	82
Figura 9.6 Diagrama de creación de actividades	83
Figura 9.7 Diagrama de listado de actividades	83
Figura 9.8 Diagrama de actualización de actividades	84
Figura 9.9 Diagrama de borrado de actividades	84
Figura 9.10 Diagrama de creación de usuarios	85
Figura 9.11 Diagrama de listado usuarios	85
Figura 9.12 Diagrama de actualización de usuarios	86
Figura 9.13 Diagrama de borrado de usuarios	86
Figura 9.14 Diagrama de creación de registros	87
Figura 9.15 Diagrama de listado de registros	87
Figura 9.16 Diagrama de actualización de registros	88
Figura 9.17 Diagrama de borrado de registros	88
Figura 9.18 Diagrama de creación de notificaciones	89
Figura 9.19 Diagrama de listado de notificaciones	89
Figura 9.20 Diagrama de actualización de notificaciones	90
Figura 9.21 Diagrama de borrado de notificaciones	90
Figura 9.22 Diagrama de creación de comunicaciones	91
Figura 9.23 Diagrama de listado de comunicaciones	91
Figura 9.24 Diagrama de actualización de comunicaciones	92
Figura 9.25 Diagrama de borrado de comunicaciones	92

Figura 9.26 Diagrama de autenticación	93
Figura 10.1 Wireframe principal	96
Figura 10.2 Wireframe de listado de actividades	97
Figura 10.3 Wireframe de listado de participaciones	97
Figura 10.4 Wireframe de listado de solicitudes	98
Figura 10.5 Wireframe de listado de invitaciones	98
Figura 10.6 Wireframe de perfil de usuario	99
Figura 10.7 Wireframe de creación de actividades	99
Figura 10.8 Wireframe de modificación de actividades	100
Figura 10.9 Wireframe de actividad detallada	100
Figura 10.10 Wireframe de autenticación	101
Figura 10.11 Wireframe de listado de usuarios	102
Figura 10.12 Wireframe de modificación de usuarios	102
Figura 10.13 Wireframe de borrado de usuarios	103
Figura 10.14 Wireframe de creación de deportes	103
Figura 10.15 Wireframe de listado de deportes	104
Figura 10.16 Wireframe de modificación de deportes	104
Figura 10.17 Wireframe de borrado de deportes	105
Figura 10.18 Wireframe de creación de actividades	105
Figura 10.19 Wireframe de listado de actividades	106
Figura 10.20 Wireframe de modificación de actividades	106
Figura 10.21 Wireframe de borrado de actividades	107
Figura 10.22 Wireframe de creación de registros	107
Figura 10.23 Wireframe de listado de registros	108
Figura 10.24 Wireframe de modificación de registros	108
Figura 10.25 Wireframe de borrado de registros	109
Figura 10.26 Wireframe de listado de notificaciones	109
Figura 10.27 Wireframe de modificación de notificaciones	110
Figura 10.28 Wireframe de borrado de notificaciones	110

Figura 10.29 Wireframe de creación de comunicaciones	111
Figura 10.30 Wireframe de listado de comunicaciones	111
Figura 10.31 Wireframe de modificación de comunicaciones	112
Figura 10.32 Wireframe de borrado de comunicaciones	112
Figura 11.1 Diagrama de despliegue del backend	116
Figura 11.2 Diagrama de despliegue del frontend	116
Figura 12.1 Modelo de clases completo	120
Figura 12.2 Modelo Entidad-Relación	128
Figura 13.1 Elementos genéricos de la interfaz de administración	130
Figura 13.2 Interfaz de listado de usuarios	131
Figura 13.3 Interfaz de edición de usuarios	131
Figura 13.4 Interfaz de borrado de usuarios	132
Figura 13.5 Interfaz de listado de comunicaciones	133
Figura 13.6 Interfaz de creación de comunicaciones	133
Figura 13.7 Interfaz de listado de deportes	134
Figura 13.8 Interfaz de creación de deportes	135
Figura 13.9 Interfaz de edición de deportes	135
Figura 13.10 Interfaz de borrado de deportes	136
Figura 13.11 Interfaz de listado de actividades	137
Figura 13.12 Interfaz de creación de actividades	137
Figura 13.13 Interfaz de edición de actividades	138
Figura 13.14 Interfaz de borrado de actividades	139
Figura 13.15 Interfaz de listado de registros	139
Figura 13.16 Interfaz de creación de registros	140
Figura 13.17 Interfaz de edición de registros	141
Figura 13.18 Interfaz de borrado de registros	141
Figura 13.19 Interfaz de listado de notificaciones	142
Figura 13.20 Elementos genéricos de la interfaz de usuario	143
Figura 13.21 Interfaz principal	144

Figura 13.22 Interfaz de actividades	144
Figura 13.23 Interfaz de participaciones	145
Figura 13.24 Interfaz de solicitudes	146
Figura 13.25 Interfaz de invitaciones	146
Figura 13.26 Interfaz de perfil de usuario	147
Figura 13.27 Interfaz de formulario de actividad	148
Figura 13.28 Interfaz de actividad	149
Figura 13.29 Interfaz de no encontrado	149
Figura A2.1 Comprobación de configuración	199
Figura A3.1 Panel de administración	204
Figura A3.2 Listado de usuarios	205
Figura A3.3 Modificación de elementos	205
Figura A3.4 Borrado de elementos	206
Figura A3.5 Creación de elementos	206
Figura A3.6 Formulario de usuarios	207
Figura A3.7 Formulario de comunicaciones	207
Figura A3.8 Formulario de deportes	208
Figura A3.9 Formulario de actividades	209
Figura A3.10 Formulario de registros	210
Figura A3.11 Navegación de la aplicación	211
Figura A3.12 Desplegable del avatar	211
Figura A3.13 Inicio de sesión	212
Figura A3.14 Inicio de sesión con Google	212
Figura A3.15 Página principal	212
Figura A3.16 Página de actividades	213
Figura A3.17 Página de participaciones	213
Figura A3.18 Página de solicitudes	214
Figura A3.19 Página de invitaciones	214
Figura A3.20 Página de perfil de usuario	215

Figura A3.21 Snackbar	215
Figura A3.22 Formulario de actividad: paso 1	216
Figura A3.23 Formulario de actividad: paso 2	216
Figura A3.24 Selector de fechas	217
Figura A3.25 Selector de horas	217
Figura A3.26 Formulario de actividad: paso 3	217
Figura A3.27 Formulario de actividad: paso 4	218
Figura A3.28 Página de actividad detallada	219

Esta página se ha dejado vacía a propósito.

Lista de tablas

Tabla 11.1 Esquema de seguridad del sistema	117
Tabla 12.1 Atributos de la clase User	121
Tabla 12.2 Métodos de la clase User	121
Tabla 12.3 Atributos de la clase Sport	122
Tabla 12.4 Métodos de la clase Sport	122
Tabla 12.5 Atributos de la clase Activity	123
Tabla 12.6 Métodos de la clase Activity	124
Tabla 12.7 Atributos de la clase Registration	125
Tabla 12.8 Métodos de la clase Registration	125
Tabla 12.9 Atributos de la clase Notification	126
Tabla 12.10 Métodos de la clase Notification	126
Tabla 12.11 Atributos de la clase Communication	127
Tabla 12.12 Métodos de la clase Communication	127
Tabla A1.1 Tabla de competencias	188
Tabla A2.1 Tabla de requisitos para la instalación del backend	193
Tabla A2.2 Tabla de requisitos para la instalación del frontend	199
Tabla A4.1 Métodos HTTP de la API	221
Tabla A4.2 Códigos de estado de la API	222
Tabla A4.3 Endpoints de la API	223
Tabla A5.1 Tabla de requisitos para la instalación del backend	227
Tabla A5.2 Tabla de requisitos para la instalación del frontend	228
Tabla A5.3 Directorios principales de la API	228
Tabla A5.4 Directorios app/	229
Tabla A5.5 Directorios app/config/	229
Tabla A5.6 Ficheros app/config/bundles/	229
Tabla A5.7 Ficheros app/config/services/	230
Tabla A5.8 Ficheros app/config/	230

Tabla A5.9 Directorios app/Resources/	231
Tabla A5.10 Directorios src/ de la API	231
Tabla A5.11 Ficheros src/AppBundle/Action/	232
Tabla A5.12 Directorios src/AppBundle/Behat/	233
Tabla A5.13 Ficheros src/AppBundle/Behat/Composition/	233
Tabla A5.14 Directorios src/AppBundle/Behat/Context/	233
Tabla A5.15 Ficheros src/AppBundle/Command/	234
Tabla A5.16 Ficheros src/AppBundle/Doctrine/	234
Tabla A5.17 Ficheros src/AppBundle/Entity/	234
Tabla A5.18 Ficheros src/AppBundle/Event/	235
Tabla A5.19 Ficheros src/AppBundle/EventDispatcher/	236
Tabla A5.20 Ficheros src/AppBundle/EventSubscriber/	236
Tabla A5.21 Ficheros src/AppBundle/Factory/	237
Tabla A5.22 Ficheros src/AppBundle/Repository/	237
Tabla A5.23 Directorios src/AppBundle/Resources/	238
Tabla A5.24 Ficheros src/AppBundle/Security/	238
Tabla A5.25 Ficheros src/AppBundle/Serializer/	238
Tabla A5.26 Ficheros src/AppBundle/Services/	238
Tabla A5.27 Ficheros src/	239
Tabla A5.28 Directorios var/	239
Tabla A5.29 Ficheros raíz de la API	239
Tabla A5.30 Dependencias de la API para el entorno de producción	240
Tabla A5.31 Dependencias de la API para el entorno de desarrollo	240
Tabla A5.32 Directorios principales de la aplicación	241
Tabla A5.33 Directorios src/ de la aplicación	241
Tabla A5.34 Directorios src/actions/	242
Tabla A5.35 Ficheros src/components/	242
Tabla A5.36 Ficheros src/containers/	243
Tabla A5.37 Ficheros src/reducers/	244

Tabla A5.38 Ficheros raíz de la aplicación	244
Tabla A5.39 Dependencias de la aplicación para el entorno de producción	245
Tabla A5.40 Dependencias de la aplicación para el entorno de desarrollo	245

Esta página se ha dejado vacía a propósito.

Introducción

Esta página se ha dejado vacía a propósito.

Capítulo 1

Introducción

El deporte es una actividad física reglamentada, normalmente con carácter competitivo, que permite mejorar la condición física y mental de quien lo practica. Según el Comité Olímpico Internacional, la práctica del deporte es un derecho humano, y uno de los principios fundamentales del Olimpismo es que "toda persona debe tener la posibilidad de practicar deporte sin discriminación de ningún tipo y dentro del espíritu olímpico, que exige comprensión mutua, solidaridad y espíritu de amistad y de juego limpio" (*COI, 2004*).

En la actualidad, muchas personas hacen ejercicio con el fin de mejorar su salud y modo de vida; el deporte se considera una actividad saludable que ayuda a mantenerse en forma, psicológica y físicamente. El deporte contribuye a establecer relaciones sociales entre diferentes personas y diferentes culturas y así contribuye a inculcar la noción de respeto hacia los otros, enseñando cómo competir constructivamente, sin hacer del antagonismo un fin en sí (*Gómez, 2001*).

Uno de los motivos por el que muchas personas no hacen ejercicio es que no son capaces de practicar deporte solos o simplemente les supone mayor esfuerzo que hacerlo acompañando. Otras personas con gustos minoritarios dejan de practicar deporte por no conocer otras personas con sus mismas aficiones. Este caso se agudiza con personas que cambian de lugar de residencia, como por ejemplo algunos estudiantes que llegan nuevos a una ciudad por estudios. Además, para la práctica de la mayoría de los deportes, tanto individuales como colectivos, se necesitan rivales, por su carácter competitivo.

En muchas ocasiones, ocurre que se necesita una persona más para jugar un partido, alguien no puede asistir en el último momento o simplemente no se encuentra a nadie que pueda jugar y finalmente no se realiza la actividad. Esto repercute económicamente de forma negativa a las entidades arrendadoras de pistas deportivas. En el caso de la Univer-

sidad de Córdoba son muchas pistas las que quedan vacías por este problema.

La búsqueda de compañeros de equipo y rivales para la organización de actividades deportivas puede llegar a ser una tarea tediosa. Este es el motivo principal por el que surge la idea de realizar este proyecto que soluciona dicho problema. Así, el sistema permitirá que personas deportistas puedan conectar y organizar actividades para practicar deporte cuando y donde mejor les parezca, o incluso permitir que empresas arrendadoras de pistas deportivas puedan organizar este tipo de actividades.

Capítulo 2

Definición del problema

2.1 Problema real

La OMS^[1] recomienda la actividad física en todos los grupos de edades y ha elaborado las *Recomendaciones mundiales sobre la actividad física para la salud (OMS, 2010)* con el objetivo de proporcionar orientación sobre frecuencia, duración, intensidad, tipo y cantidad total de actividad física según el grupo.

La práctica de ejercicio físico en solitario siempre tiene mayor dificultad que de forma colectiva. El problema por el que muchas personas dejan de realizar la actividad física recomendable reside en la búsqueda de personas con mismos intereses deportivos y misma disponibilidad temporal para practicar deporte colectivamente.

El proceso de organización de actividades deportivas requiere que un organizador dedique demasiado tiempo sin garantías de que finalmente llegue a celebrarse la misma. El organizador debe contactar con todos sus conocidos que puedan estar interesados en participar e intentar que estos, a su vez, contacten con los suyos con el mismo fin.

Una vez son todos contactados, los interesados deben concretar la fecha, hora y lugar de la actividad para conseguir un número aceptable de participantes en ella. Este es el punto de inflexión donde la mayoría de actividades de mayor participación no suelen tener éxito, como por ejemplo un partido de fútbol siete donde se necesitan un total de catorce personas para celebrarse correctamente. Además, suponiendo que se consiga organizar finalmente, siempre existe la posibilidad de que alguien, finalmente, cause baja.

[1] Organización Mundial de la Salud.

Otro proceso que se lleva a cabo para actividades que necesitan un número de deportistas más elevado, como por ejemplo la celebración de una carrera, es la pegada de carteles en la que aparecen otros problemas como la creación y distribución de estos, que pueden ser bastante costosos y al final repercute en el participante, que tiene que abonar una cantidad para su participación.

Además, existe un problema para aquellos deportistas que viajan a otro lugar o que cambian de residencia, les es difícil practicar deporte porque no tienen suficientes contactos en su nueva residencia o lugar de vacaciones.

Todos estos problemas que se plantean pueden hacer que no llegue a realizarse la actividad física o que su realización suponga un coste más elevado de lo normal y que entidades de alquiler de pistas deportivas, como *Ucodeporte*, pierdan ingresos ya que sus pistas no están en uso. El motivo por el que estas entidades organizan actividades y hacen una inversión en propaganda es llenar sus pistas deportivas.

2.2 Problema técnico

2.2.1 Funcionamiento

Desde el punto de vista del administrador del sistema:

- Instalación y configuración de la aplicación.
- Configuración de la identificación de usuarios.
- Administración de los usuarios.

Desde el punto de vista del administrador de la aplicación:

- Control y gestión de los usuarios.
- Gestión de los deportes. Se podrá crear, modificar o eliminar un deporte.
- Notificación de usuarios. El administrador podrá notificar a los usuarios.

Desde el punto de vista del usuario:

- Identificación. El usuario podrá registrarse e identificarse en la aplicación, no siendo necesario para la visualización y búsqueda de actividades.
- Gestión de actividades. El usuario podrá crear actividades para un deporte, decidirá la fecha, hora, lugar y duración de las mismas. Podrá modificar o eliminar las actividades que ha creado anteriormente.
- Invitación a actividades. El creador de una actividad podrá invitar a otros usuarios de la aplicación a participar en su actividad.

- Gestión de invitaciones. El usuario podrá eliminar las invitaciones que ha realizado.
- Búsqueda de actividades. El usuario podrá buscar actividades que le interesen.
- Solicitud de participación. Cualquier usuario podrá solicitar su participación en una de las actividades que ha creado otro usuario de la aplicación.
- Gestión de solicitudes. El usuario podrá eliminar las solicitudes que ha realizado.
- Aceptación de solicitudes. El creador de una actividad podrá aceptar solicitudes de participación. Una vez que un usuario es aceptado, podrá ver los usuarios participantes en la actividad.
- Denegación de solicitudes. El creador de una actividad podrá rechazar solicitudes de participación.
- Gestión de la participación. El creador de una actividad podrá ver y eliminar a los participantes.

2.2.2 Entorno

El entorno de ejecución del *backend* de la aplicación será en servidores con soporte de lenguajes de programación web y acceso a Internet. El *frontend* será de acceso remoto desde cualquier equipo conectado a internet, ya sea ordenador personal, portátil, *tablet*, *smartphone* o similar. En general el *frontend* de la aplicación funcionará en cualquier navegador web compatible con las últimas tecnologías web (*HTML5*, *CSS3* y *JavaScript*).

2.2.3 Vida esperada

En principio no se ha determinado una vida concreta útil para el proyecto, puesto que el proceso de organización de actividades deportivas no debe sufrir grandes cambios y la aplicación se prevee que sea lo suficientemente versátil para poder asumir nuevas necesidades y nuevos criterios. Por lo que se determina que la vida del proyecto será amplia y debe seguir activa mientras sea de utilidad para su propósito.

2.2.4 Ciclo de mantenimiento

Las revisiones que pueda sufrir la plataforma estarán supeditadas a revisiones de seguridad y mantenimiento de los entornos de desarrollo sobre los que esté construida la herramienta. Si se dictamina que el *frontend* de clientes estará basado en *React*, está previsto que siga actualizándose, con lo que habrá que comprobar que sigue siendo compatible con las nuevas versiones. Lo mismo con respecto al *backend* y el *frontend* de administración. También puede darse el caso de que haya que revisar el código de la aplicación, si el lenguaje utilizado sufre algún cambio o si alguna actualización del servidor obliga a ello. Como las actualizaciones de los servidores o de los lenguajes no son una circunstancia que se sepa a priori, no se puede determinar si obligará a hacer alguna operación de mantenimiento.

En cualquier caso, como mínimo cada seis meses, puede establecerse un chequeo del entorno para comprobar que la herramienta va a seguir siendo compatible con el entorno descrito anteriormente, independientemente de que alguna notificación de seguridad urgente obligue a hacer una operación de mantenimiento mucho antes.

2.2.5 Competencia

Este *Trabajo Fin de Grado* es una solución a un problema que ya se ha intentado solucionar de diferentes formas. Aunque hay multitud de webs que permiten organizar actividades deportivas, ninguna es gratuita y está tan orientada al usuario deportista. Además, por cuestiones éticas, este proyecto será distribuido como *Software Libre*, con lo que no se puede decir que vaya a competir con otros productos, puesto que ese no es el espíritu de este tipo de proyectos.

2.2.6 Aspecto externo

Tanto para los *frontend* de administración como para los del usuario, la aplicación se construirá en *HTML5* cumpliendo con los estándares de accesibilidad web, de tal manera que sea accesible para todos los usuarios.

Respecto a la distribución del producto, este estará disponible en alguna forja de software para su descarga libre, y contará con un manual de instalación y otro de usuario para que pueda ser usado por otras entidades.

2.2.7 Estandarización

La aplicación se desarrollará en *PHP*, ya que, entre otros motivos, se contará muy probablemente con servidores webs con soporte para este lenguaje, lo que acabará convertido en un factor dato. Por ello seguiremos las indicaciones del *PHP Framework Interop Group*^[2] para la estandarización del código fuente.

El *backend* estará construido a través de *REST*^[3], un estilo de arquitectura software para sistemas hipermédia distribuidos como la *World Wide Web*, por lo que se seguirán su estándar y recomendaciones. Para los *frontend* se usará *HTML5*, *CSS3* y *JavaScript*, además de las recomendaciones de la *W3C*^[4] para accesibilidad.

2.2.8 Calidad y fiabilidad

Los principales puntos de fallo a tener en cuenta a la hora de que la aplicación sea fiable y de calidad son:

- El sistema debe asegurarse de que las actividades se creen correctamente, de tal

[2] **PHP-FIG:** <http://www.php-fig.org/>

[3] La Transferencia de Estado Representacional (Representational State Transfer) o *REST* es una técnica de arquitectura software para sistemas web.

[4] *World Wide Web Consortium.*

forma que no exista la posibilidad de que surjan errores de índole temporal.

- Los *frontend* deben cumplir los estándares web para que sean accesibles desde el mayor número de plataformas posible. No es aceptable que solo pueda usarse un tipo de navegador para usar el software.

2.2.9 Programa de tareas

Para el desarrollo del *Trabajo Fin de Grado* no se utilizará el desarrollo clásico en cascada, sino que se seguirá un paradigma de desarrollo ágil denominado SCRUM. Las características principales de este tipo de metodología y que suponen una ventaja respecto al modelo anterior son:

- Se asumen que puede haber cambios en el contexto del proyecto, por lo tanto, se controlarán los resultados obtenidos y, en función de estos, se realizarán las adaptaciones adecuadas.
- Las fases se plantean en función de los objetivos, las cuales suelen ser de períodos cortos de tiempo (mínimo dos semanas, máximo diez) y en las que al final de las mismas se hacen demostraciones al cliente. De esta forma es fácil realizar cambios y el proceso no precisa de tanto control.
- El cliente es parte del proyecto durante toda la vida del mismo.
- Se realizan retrospectivas durante todo el proyecto, que permiten ver el trabajo realizado, las mejoras necesarias y los problemas que pueden aparecer.

Este paradigma es un ciclo iterativo o cíclico donde las fases son:

1. **Concepto:** se definen las características del producto.
2. **Especulación:** esta fase se repite en cada iteración y consiste en:
 - Desarrollar y revisar los requisitos generales.
 - Mantener la lista de las funcionalidades esperadas.
 - Plan de entrega. Se establecen las fechas de las versiones, hitos e iteraciones.
3. **Exploración:** se incrementa el producto en el que se añaden las funcionalidades de la fase anterior.
4. **Revisión:** se revisa lo construido y se contrasta con el objetivo deseado.
5. **Cierre:** se entregará en la fecha acordada una versión del producto. Se trata de versiones, no del producto final, que no estará listo hasta la última iteración.

2.2.10 Pruebas

Para el desarrollo del software usaremos una metodología denominada *Desarrollo Orientado al Comportamiento* del inglés *Behaviour-driven Development (BDD)*. Esta metodología está basada en el desarrollo orientado a pruebas (*Test-driven Development o TDD*). Las ventajas de este paradigma de desarrollo son:

- Las pruebas se definen antes de desarrollar el software. Al contrario de la metodología clásica donde se hacen pruebas después de la fase de desarrollo, aquí las pruebas se construyen antes, con lo que se evita la posibilidad de escribir *ad-hoc* para que el sistema cumpla. Es evidente que las pruebas fallarán al comienzo, puesto que aún no ha sido programado el software, pero es precisamente lo que se persigue: conocer de antemano cual es el contrato que debe cumplir el software para que cumpla con las pruebas.
- *BDD* cuenta con un lenguaje que permite escribir las pruebas en historias que el cliente puede entender.

Debido a esos dos puntos, el cliente puede saber, a priori, que es lo que el software va a realizar y cómo.

2.2.11 Seguridad

Los puntos más críticos de seguridad son los siguientes:

- El sistema de autenticación debe ser seguro y proveer de perfiles de seguridad, ya que no todos los usuarios van a tener las mismas responsabilidades.
- La base de datos debe ser protegida contra accesos no autorizados, de tal manera que se eviten los problemas inherentes a los permisos de lectura/escritura.
- Los *frontend* de usuario deben evitar la posibilidad de que sean vulnerables a los ataques típicos a las plataformas web como *XSS*^[5], inyecciones *SQL* o ejecución de código en el servidor.
- Deben seguirse los mantenimientos de seguridad escrupulosamente, ya que son la principal causa de ataques a páginas web.

[5] Cross Site Scripting.

Capítulo 3

Objetivos

El objetivo principal de este proyecto es el diseño, desarrollo e implementación de un sistema que permita conectar deportistas para la organización de actividades deportivas.

Dicho propósito se desea alcanzar mediante la consecución de los siguientes objetivos específicos:

- Identificación de los usuarios mediante un sistema de identificación federada como *Google, Facebook o RedIRIS (SIR)*.
- Gestión de usuarios. El administrador podrá gestionar los usuarios registrados y ser capaz de visualizar, modificar o borrar cualquiera de ellos.
- Gestión de deportes. El administrador podrá inscribir nuevos deportes, modificar existentes o eliminar cualquiera de ellos.
- Gestión de actividades. Los usuarios podrán crear nuevas actividades deportivas, así como modificar o eliminar aquellas en las que posean permisos para ello.
- Gestión de notificaciones. El administrador podrá enviar notificaciones a los usuarios registrados.
- Personalización de usuarios. Los usuarios podrán definir su propio perfil deportivo.
- Acceso a las actividades. Los usuarios podrán buscar actividades en las que tengan interés, inscribirse en ellas y contactar con los participantes de estas.

Esta página se ha dejado vacía a propósito.

Capítulo 4

Antecedentes

La organización de actividades deportivas por usuarios deportistas es un problema que ya se ha intentado solucionar con diferentes criterios utilizando tecnologías informáticas. Existen en la red numerosas aplicaciones webs que permiten la organización de estas actividades deportivas por usuarios, pero ninguna de ellas satisface del todo las necesidades de estos usuarios.

La mayoría de estas aplicaciones como *Hat-trick*^[1] o *Deporticket*^[2] están enfocadas a la organización de grandes eventos como carreras populares o torneos pequeños, es decir, los eventos son organizados por alguna entidad o club. Este proyecto está destinado tanto a este tipo de eventos como cualquier otro tipo de actividad, ya sea un partido de tenis para dos personas o un torneo de fútbol en el que participan numerosos equipos con muchos jugadores.

4.1 Hat-trick

La aplicación de *Hat-trick* consiste en un blog que facilita la publicación de eventos a las entidades organizadoras, pero no considera en ningún momento la inscripción de estos.

4.2 Deporticket

La aplicación de *Deporticket* sí que permite la inscripción del usuario con su Número de Identificación Fiscal o NIF. Sin embargo, aún queda muy lejos de todas las posibilidades que ofrece el dominio del problema.

Otro problema frecuente en este tipo de aplicaciones es que basan su éxito en revelar algún método de contacto entre usuarios que buscan actividades similares o simplemente han

[1] *Hat-trick*: <http://hat-trick.es/>

[2] *Deporticket*: <http://www.deporticket.com/>

usado un filtro por edad y zona, es decir, son más bien webs de contacto. Este proyecto evitirá que los usuarios contacten de forma directa, sino que lo hagan a través de la propia aplicación.

Es importante poner de manifiesto que tanto el autor del proyecto como los directores del mismo apuestan firmemente por el software libre debido a su filosofía en el proceso de desarrollo y a los beneficios sociales que reporta. En este sentido, no ha sido posible encontrar ninguna aplicación cuyo proyecto sea libre.

Capítulo 5

Restricciones

5.1 Factores dato

Por imposición del proyecto y de las circunstancias y características, el proyecto debe cumplir con las siguientes restricciones:

- Será liberado con una licencia de *Software Libre* para que cualquiera de estas entidades pueda utilizarlo, ya que el producto es una idea original del *Aula de Software Libre de la Universidad de Córdoba*. Dentro de las opciones disponibles entre la multitud de licencias existentes (*Free Software Foundation, 2012*) hemos escogido la licencia GPLv3 debido a que es la más extendida y la que garantiza las cuatro libertades que definen al software libre (*Free Software Foundation, 1996*).
- La aplicación dispondrá de una *API*^[1] cuyo objetivo es que otras entidades puedan realizar aplicaciones en distintas plataformas (web, móvil o escritorio) y que estas se puedan comunicar con la nuestra para completar la solución que daremos con este proyecto.

5.2 Factores estratégicos

La mayoría de las entidades de alquileres de pistas deportivas ya cuentan con un *hosting* con soporte de *PHP* y *MySQL* donde albergan su portal web con un gestor de contenidos basado en *PHP*, *WordPress*^[2]. Según *W3techs*, un 59.2% del total de webs actuales que gestionan contenido utilizan *WordPress*^[3]. Para evitar gastos suplementarios, aprovecharemos el servicio con el que cuentan y la plataforma se construirá bajo ese lenguaje de programación y con ese motor relacional.

[1] Interfaz de Programación de Aplicaciones (*Application Program Interface*).

[2] *WordPress*: <http://www.wordpress.org/>

[3] *W3techs*: <https://w3techs.com/technologies/details/cm-wordpress/all/all/>

Generalmente los usuarios son reacios a crear nuevas cuentas para utilizar sitios webs, aproximadamente un 86% de los usuarios reportan ser molestados al tener que crear nuevas cuentas (*Soni, 2017*). Por ello, la identificación de los usuarios en nuestra aplicación se realizará a través de un sistema de identificación federada como *Google, Facebook o RedIRIS (SIR)*. finalmente, usaremos *Google* por ser el de mayor número de usuarios.

5.2.1 Interfaz de usuario

Si bien hemos hablado de que el *frontend* del cliente va a ser una aplicación web, no queda tan claro que eso obligue al *backend* y al *frontend* de administración a que también sean aplicaciones web. Podrían ser aplicaciones nativas de escritorio y un servicio centralizado con otro tipo de interfaz que no sea web, como *RPC* o *SOAP*. La cuestión es que, como ya hemos citado, se contará con un hosting que ofrece servicios de *PHP* y *MySQL*, con lo que, a priori, la solución web es la menos costosa y más fácil de implementar. Además, crear aplicaciones nativas provoca que la aplicación solo funcione bajo determinados sistemas, salvo que se realice un cliente para cada plataforma existente (*Windows, GNU/Linux, macOS, Android, iOS, BlackBerry, Windows Phone*, etc.) mientras que una aplicación web estaría disponible para cualquier sistema que cuente con un navegador web. De esa manera, además, nos ahorraremos la distribución del cliente nativo entre todos los usuarios. Esto también tiene otra serie de ventajas, como que cualquier actualización en el *frontend* estaría automáticamente disponible mientras que si el cliente fuese nativo habría que volver a distribuir la aplicación entre todos los usuarios.

Por otra parte, para garantizar que la aplicación es accesible, usaremos algún entorno de desarrollo para web que garantice que la web será sensible al contexto. Una web sensible al contexto, lo que se denomina como *Responsive Design*, es una página que se ajusta al tamaño del dispositivo donde se muestra. No es lo mismo si la web se muestra en un monitor de 24 pulgadas que en una *tablet* de 8 pulgadas. La web se tiene que adaptar al espacio que tiene para mostrarse. Estos entornos no son más que hojas de estilo web (*CSS*) predefinidas con una serie de clases que proporcionan un interfaz para la construcción de páginas. Los más conocidos son: *Bootstrap* y *Gumby Framework*, que además de ofrecer un sistema sensible al contexto, ofrecen un *toolkit* completo para crear botones, formularios, etc. No hay una razón de peso, a priori, para elegir una opción u otra, excepto la estética que cada uno ofrece. Ambos son productos con licencia libre y ambos tienen buena documentación. Sin embargo, si es cierto que *Bootstrap* tiene una comunidad treinta veces mayor que la del otro *framework*.

Como el desarrollo del proyecto es iterativo, lo decidimos durante la primera iteración, de forma que usaremos *Twig*^[4] para el *frontend* de administración y el conjunto de componentes de *Material-UI*^[5] para el *frontend* dedicado al usuario por la gran familiaridad que

[4] *Twig*: <https://twig.symfony.com/>

[5] *Material-UI*: <http://www.material-ui.com/>

tiene el usuario web actual con *Material Design*^[6].

5.2.2 Entorno de desarrollo

La cuestión fundamental para el desarrollo del proyecto que tenemos que plantearnos es en qué entorno de desarrollo (*framework*) vamos a implementar la plataforma.

Backend

Aunque *WordPress* no es un entorno de desarrollo completo, sí que dispone de una *API* propia que permite el desarrollo de aplicaciones a través de la ampliación de funcionalidades del gestor de contenidos. La ventaja que implicaría usar *WordPress* es que a toda la plataforma se accedería a través del *backend* de *WordPress*. La desventaja, por otra parte, es que *WordPress* no implementa un sistema de seguridad muy completo, ofreciendo solo una serie de roles y permisos-roles que además se mezclan con los permisos del gestor de contenidos, que para ese objetivo es suficiente, pero para los requisitos de este proyecto podrían no serlo. *WordPress*, tampoco usa una base de datos relacional normalizada, debido a que su objetivo es guardar información heterogénea. Si bien eso no es un problema, puesto que gracias al uso de índices y de cachés, el rendimiento está más que asegurado (no en vano es el gestor de contenido más usado actualmente en Internet con un 42.46% del parque de sitios web^[7]). En cualquier caso, si implementamos el *backend* en *WordPress* cerraremos la posibilidad de que se pudieran hacer otros *frontend* de cliente para otros gestores de contenido, como *Joomla* o *Drupal*, o al menos haríamos muy complicada su implementación.

Por otra parte, si elaboramos el *backend* en otro entorno de desarrollo, conseguiremos que sea posible implementar *frontends* para otros gestores de contenido, simplemente ofreciendo una *API* abierta para hacer las consultas (tipo *REST* por ejemplo). De igual manera, tanto la gestión de los datos, como la seguridad, se verá incrementada, puesto que la mayoría de los *frameworks* de desarrollo ofrecen características que facilitan estas tareas. De entre los *frameworks* más avanzados de PHP con licencias libres encontramos los siguientes:

Symfony.

Es un completo *framework* diseñado para optimizar el desarrollo de las aplicaciones web basado en el patrón *Modelo-Vista-Controlador*. Para empezar, separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. El resultado de todas estas ventajas es que no se debe reinventar la rueda cada vez que se crea una nue-

[6] *Material Design*: <https://material.io/>

[7] *CMS Usage Statistics*: <http://trends.builtwith.com/cms/>

va aplicación web (*Wikipedia, 2007*). *Symfony* está basado en componentes desacoplables, que además han servido de base para otros proyectos y otros entornos de desarrollo. Tiene soporte comercial a través de una empresa francesa, que es la desarrolladora (*SensioLabs*). Se distribuye con una licencia libre.

CodeIgniter.

Es un *framework* para aplicaciones web de código abierto para crear sitios web dinámicos con *PHP*. La empresa que lo desarrolla, *EllisLab*, decidió ponerlo a la venta y su futuro está en el aire.

CakePHP.

Es un *framework* para el desarrollo aplicaciones web escrito en *PHP*, creado sobre los conceptos de *Ruby on Rails*. Su principal debilidad con respecto a los otros ya citados es que su documentación no es tan completa y carece de gestor de depuración y de una capa separada de servicios.

Laravel.

Es otro *framework* desarrollado en *PHP* que comparte con *Symfony* varios componentes. Se creó en 2011 y es relativamente el más moderno de todos, ya que *Symfony* sufrió una reescritura desde cero en 2009 que terminó en 2011. Actualmente se encuentra en auge, ya que permite reutilizar componentes y conocimientos de *Symfony*, pero siendo más sencillo de configurar. Si bien *Symfony* es un *framework* muy completo, es igualmente complejo y para proyectos pequeños puede que no compense su utilización, si no se van a usar todas sus características. *Laravel* se ajusta mejor a requisitos de proyectos de complejidad baja-alta, y debido a eso es por lo que está aumentando su popularidad entre la comunidad de desarrolladores de *PHP*.

De entre todas estas soluciones, finalmente hemos decidido que se desarrollará en *Symfony* por las siguientes razones:

- Si se hace finalmente en *WordPress* podríamos cerrar la puerta a que otras entidades puedan usar la herramienta si usan un gestor de contenidos diferente.
- De entre todos los entornos de desarrollo mostrados, es el que mejor documentación tiene y el que tiene la mejor comunidad de desarrolladores, con listas de correo y blogs muy activos, que pueden ayudar a resolver dudas y problemas, en el caso de que se den.
- *Symfony* cuenta, además, con multitud de extensiones o *plugins*, conocidos como *Bundles*, que permiten de una forma fácil añadir funcionalidades a las aplicaciones que se desarrollan en este *framework*. Por ejemplo, con un par de *Bundles* podemos

ofrecer soporte para interfaces *REST* y para identificación *OAuth*, si fuera necesario para conectar los *frontends* de cliente con el *backend*.

- El *Aula de Software Libre* ya cuenta con experiencia en el desarrollo de aplicaciones con *Symfony*, lo que evitara retrasos en la entrega del trabajo. Todo *framework* tiene una curva de aprendizaje que inevitablemente consume mucho tiempo. eligiendo *Symfony* podemos asegurarnos de que nos ajustaremos a la planificación inicial.

Frontend

En el *frontend*, debemos decidir que librería utilizar para construir el cliente web. Las más representativas son:

ReactJS.

Es una librería de *Javascript* para construir interfaces de usuario. Es declarativo, se diseñan las vistas para cada estado y la propia librería se encarga de actualizar y renderizar los componentes necesarios cuando los datos cambian. Y otra característica importante es que está basado en componentes, cada uno de ellos con su propio estado y se componen para crear interfaces complejas.

VueJS.

Consiste en una librería para crear interfaces web que cuenta con un sistema flexible de transiciones, es rápido y está orientado a componentes.

Angular.

Framework para el desarrollo de aplicaciones web, móviles y de escritorio. Cuenta con gran cantidad de herramientas para el desarrollo.

Ember.

Se trata de un *framework* libre y de código abierto para el desarrollo de aplicaciones web. Cuenta con el sistema de plantillas *Handlebars* integrado, que se actualiza automáticamente cada vez que cambian los datos y con herramientas que facilitan el desarrollo.

De todas las librerías indicadas solamente *Angular* y *Ember* son *frameworks* completos y no sólo librerías para el desarrollo de interfaces. En el caso de *ReactJS* o *VueJS*, para poder realizar el cliente web completo debemos acompañarlo de una librería para controlar el estado de nuestra aplicación. Librerías como pueden ser *Redux*^[8] o *MobX*^[9].

[8] **Redux:** <http://redux.js.org/>

[9] **MobX:** <https://mobx.js.org/>

Finalmente, se utilizará *ReactJS* acompañado de *Redux* como contenedor de estado porque el autor de este *Trabajo Fin de Grado* está más familiarizado con el uso de estas tecnologías, lo que ayudará a ajustarnos a la planificación del proyecto y ambas tecnologías tienen mayor comunidad respecto a las demás soluciones.

Capítulo 6

Recursos

6.1 Recursos humanos

La dirección, coordinación y supervisión del proyecto ha correspondido a:

- Dr. Pedro Antonio Gutiérrez Peña, Profesor Titular del Departamento de Informática y Análisis Numérico de la Escuela Politécnica Superior de la Universidad de Córdoba y miembro del grupo de investigación AYRNA.
- D. Sergio Gómez Bachiller, Postgraduado en Ingeniería Informática y Coordinador del Aula de Software Libre.

El estudio del problema, análisis de requisitos, diseño, codificación, generación de pruebas, documentación y desarrollo del proyecto ha correspondido a D. Víctor Monserrat Vilalatoro, alumno del Grado en Ingeniería Informática, miembro del Aula de Software Libre y becario del Servicio de Informática de la Universidad de Córdoba.

6.2 Recursos hardware

6.2.1 Entorno de desarrollo

El equipo con el que se ha efectuado el desarrollo del proyecto ha sido un *MacBook Pro* personal con pantalla retina de 13 pulgadas (principios de 2015) compuesto por un procesador *Intel Core i5* de doble núcleo a 2.7 GHz (*Turbo Boost* de hasta 3,1 GHz) con 3 MB de caché de nivel 3 compartida, una memoria *RAM LPDDR3* de 16 GB a 1.866 MHz y una unidad de almacenamiento *SSD PCIe* de 128 GB.

6.2.2 Entorno de preproducción

El entorno de preproducción será un *droplet* de *DigitalOcean*^[1] compuesto por una *CPU* de

512 MB, una unidad de almacenamiento *SSD* de 20 GB y un límite de transferencia mensual de 1000 GB, cuyo sistema operativo será *Ubuntu 17.04 x64* y cuya región del centro de datos será Londres.

6.3 Recursos software

Los principales recursos software utilizados para el desarrollo del proyecto han sido:

- Sistema operativo: *macOS*.
- Plataforma de contenedores: *Docker*.
- Sistema de control de versiones: *Git (GitLab)*.
- Editor de texto: *Atom*.
- Documentación del proyecto: *Easybook*.
- Diagramas *UML*: *Draw.io*.
- Diseño *GUI*: *Draw.io*.

Para el desarrollo del *backend* del proyecto se han utilizado:

- Entorno de desarrollo: *PHPStorm*.
- Entorno de desarrollo de base de datos: *DataGrip*.
- Framework de desarrollo: *Symfony*.
- Framework de desarrollo *API*: *API Platform*.
- Framework de desarrollo orientado a comportamiento: *Behat*.

Para el desarrollo del *frontend* del proyecto se han utilizado:

- Motor de plantillas: *Twig*.
- Librería de desarrollo: *React*.
- Contenedor del estado: *Redux*.
- Librería de componentes: *Material-UI*.
- Librería de autenticación: *Hello*.
- Librería de horarios: *Moments*.
- Librería de enrutamiento: *Router*.
- Librería de sensibilidad al contexto: *Grid System*.

[1] *DigitalOcean*: <https://www.digitalocean.com/>

Los requisitos de la plataforma de desarrollo son:

- *PHP* versión 7.1 o superior. Los siguientes módulos de *PHP* deben estar presentes: *curl, intl, json, mysqli, xml*.
- Servidor de base de datos relacional compatible con *Doctrine 2*. Se ha usado *Mysql 5.5*.
- Servidor web compatible con *PHP*. Se han usado, indistintamente *Apache 2.4, nginx 1.4* y el motor interno de *HTTP* que trae *PHP* a partir de la versión 5.4.

Esta página se ha dejado vacía a propósito.

Análisis

Esta página se ha dejado vacía a propósito.

Capítulo 7

Especificación de requisitos

Este *Trabajo Fin de Grado* abordará la creación de una aplicación web que permita la gestión de actividades deportivas y su organización por usuarios deportistas y entidades arrendadoras de pistas deportivas. Además, abordará la creación de una aplicación web de administración que permita a usuarios administradores gestionar usuarios, deportes y actividades de la aplicación web principal.

7.1 Requisitos de información

Este apartado recoge toda la información relevante que la aplicación debe almacenar y gestionar. Los requisitos de información son los siguientes.

Requisito 1: Información sobre el usuario.

El sistema almacenará información relevante sobre el usuario.

Datos específicos:

- Identificador.
- Nombre de usuario.
- Contraseña.
- Nombre completo.
- Correo electrónico.
- Habilitación del usuario.

- Semilla de cifrado.
- Avatar.
- Registros.
- Actividades.
- Roles.

Requisito 2: Información sobre los deportes disponibles.

El sistema almacenará información relevante sobre los deportes de los que podrán organizar actividades los usuarios de la aplicación.

Datos específicos:

- Identificador.
- Etiqueta.
- Nombre.
- Descripción.
- Actividades.

Requisito 3: Información sobre las actividades organizadas por los usuarios.

El sistema almacenará información relevante sobre las actividades gestionadas y organizadas por los usuarios de la aplicación.

Datos específicos:

- Identificador.
- Título.
- Fecha y hora de comienzo.
- Duración (en minutos).
- Lugar.
- Descripción.
- Plazas.
- Registros.
- Dueño.

- Deporte.

Requisito 4: Información sobre los registros de los usuarios en las actividades.

El sistema almacenará información relevante sobre los tipos de registros que harán los usuarios de la aplicación en las actividades.

Datos específicos:

- Identificador.
- Tipo:
 - Solicitud.
 - Invitación.
- Estado:
 - Aceptada.
 - Rechazada.
 - En espera.
- Fecha y hora de creación.
- Usuario.
- Actividad.

Requisito 5: Información sobre las notificaciones del sistema a los usuarios.

El sistema almacenará información relevante sobre las notificaciones que el sistema envíe a los usuarios de la aplicación.

Datos específicos:

- Identificador.
- Tipo:
 - Solicitud.
 - Invitación.
 - Aceptación.
 - Rechazo.

- Fecha y hora de envío.
- Usuario.
- Actividad.

Requisito 6: Información sobre las comunicaciones del administrador a los usuarios.

El sistema almacenará información relevante sobre las comunicaciones que el administrador envíe a los usuarios de la aplicación.

Datos específicos:

- Identificador.
- Asunto.
- Mensaje.
- Fecha y hora de envío.
- Usuario.

7.2 Requisitos funcionales

Los requisitos funcionales determinarán lo que el sistema debe hacer. Se usará la siguiente nomenclatura: *RFnn*, donde *nn* indica un número consecutivo que sirve para identificar al requisito.

RF01 El sistema permitirá el acceso a administradores y usuarios.

RF02 El *backend* de administración permitirá el acceso solo a usuarios administradores.

RF03 Un usuario administrador debe poder ver la lista de todos los usuarios.

RF04 Un usuario administrador debe poder editar los datos de los usuarios.

RF05 Un usuario administrador debe poder deshabilitar a cualquier usuario.

RF06 Un usuario administrador debe poder asignar roles a cualquier usuario.

RF07 Un usuario administrador debe poder eliminar a cualquier usuario.

RF08 Un usuario administrador debe poder administrar (*CRUD*)^[1] los deportes.

RF09 Un usuario administrador debe poder ver la lista de todas las actividades organizadas por los usuarios.

RF10 Un usuario administrador debe poder editar las actividades organizadas por los

[1] *CRUD (Create, Read, Update y Delete)*: <http://es.wikipedia.org/wiki/CRUD/>

usuarios.

RF11 Un usuario administrador debe poder eliminar las actividades organizadas por los usuarios.

RF12 Un usuario administrador debe poder ver la lista de todos los registros de los usuarios en las actividades.

RF13 Un usuario administrador debe poder editar los registros de los usuarios en las actividades.

RF14 Un usuario administrador debe poder eliminar los registros de los usuarios en las actividades.

RF15 Un usuario administrador debe poder notificar a cualquier usuario.

RF16 El sistema debe poder notificar automáticamente a cualquier usuario.

RF17 Un usuario debe poder registrarse mediante un sistema de identificación federada.

RF18 Un usuario debe poder identificarse mediante un sistema de identificación federada.

RF19 Un usuario identificado debe poder cerrar sesión en cualquier momento.

RF20 Un usuario debe poder ver la lista de actividades sin necesidad de identificarse.

RF21 Un usuario identificado debe poder crear actividades.

RF22 Un usuario identificado debe poder modificar los datos de sus actividades.

RF23 Un usuario identificado debe poder eliminar sus actividades.

RF24 Un usuario identificado debe poder invitar a otros usuarios a sus actividades.

RF25 Un usuario identificado debe poder ver sus invitaciones a sus actividades.

RF26 Un usuario identificado debe poder cancelar sus invitaciones a sus actividades.

RF27 Un usuario identificado debe poder aceptar invitaciones a actividades.

RF28 Un usuario identificado debe poder rechazar invitaciones a actividades.

RF29 Un usuario identificado debe poder solicitar su participación en actividades.

RF30 Un usuario identificado debe poder ver sus solicitudes de participación a actividades.

RF31 Un usuario identificado debe poder cancelar sus solicitudes de participación en actividades.

RF32 Un usuario identificado debe poder aceptar solicitudes de participación en sus

actividades.

RF33 Un usuario identificado debe poder rechazar solicitudes de participación en sus actividades.

RF34 Un usuario identificado debe poder eliminar participaciones de sus actividades.

RF35 Un usuario identificado debe poder cancelar su participación en actividades.

RF36 Un usuario debe poder ver los deportes disponibles sin necesidad de identificarse.

RF37 Un usuario identificado debe poder administrar los datos de su perfil.

RF38 Un usuario identificado debe poder leer sus notificaciones.

RF39 Un usuario identificado debe poder eliminar su perfil de la aplicación.

RF40 El sistema debe ofrecer una *API* que permita la gestión y consulta de la base de datos desde cualquier aplicación móvil o web que la consuma.

RF41 La *API* debe poder realizar cualquier funcionalidad anteriormente descrita.

RF42 El sistema debe ofrecer un cliente web que consuma la *API* para el uso de los usuarios.

7.3 Casos de uso

Para el modelado de los casos de uso seguiremos las recomendaciones de IBM (*IBM, 2008*). En la "Figura 7.1" puede verse el esquema general de los casos de uso.

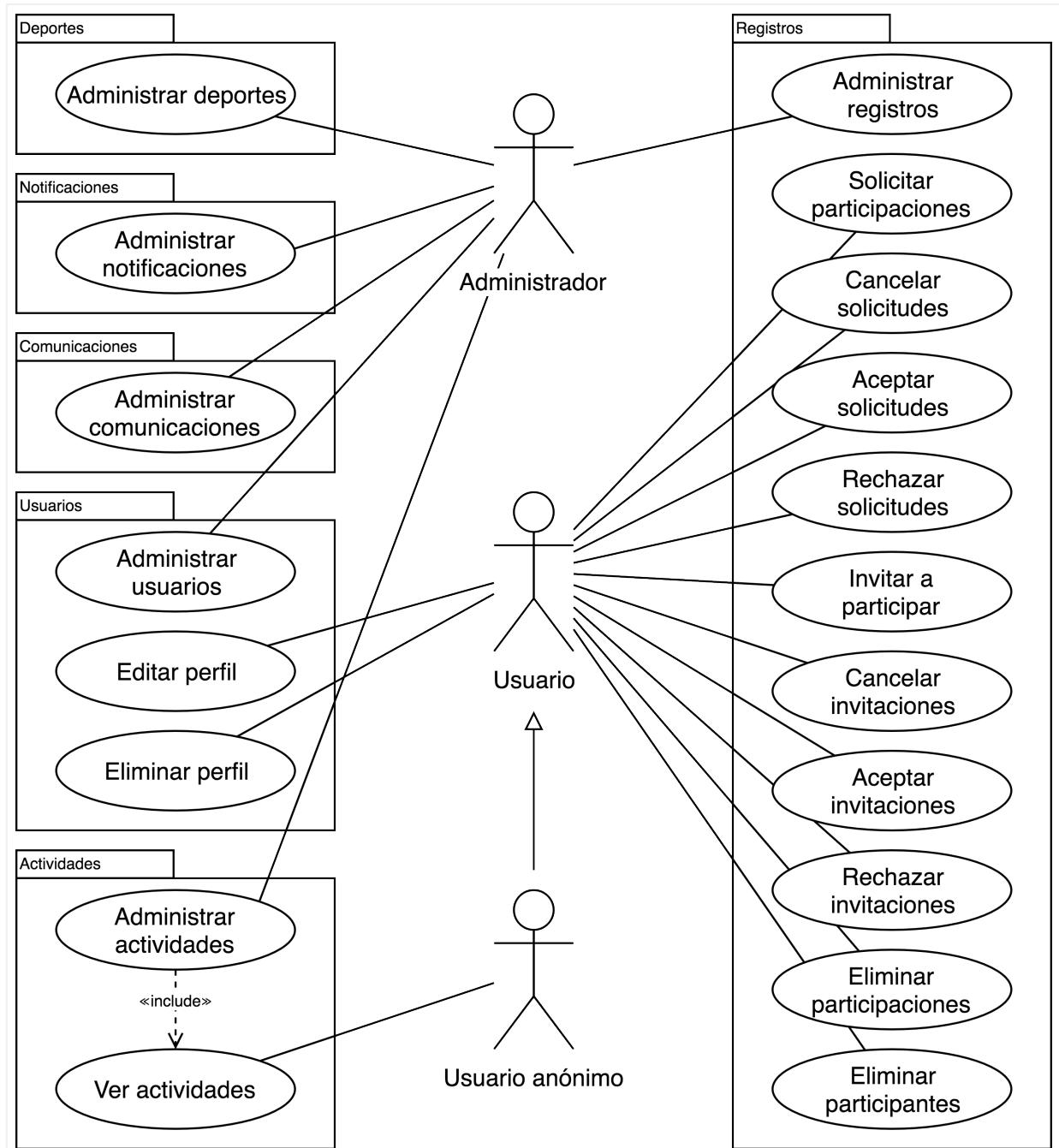


Figura 7.1 Casos de uso

7.3.1 Actores

Antes de comenzar vamos a indicar como referencia los distintos tipos de actores:

Administrador.

Los usuarios administradores son responsables de la gestión de la aplicación, sus datos y usuarios.

Usuario.

Es una generalización del tipo de actor **usuario anónimo**. Son aquellos usuarios con acceso a la aplicación.

Usuario anónimo.

Son aquellos usuarios con acceso a la aplicación que esta no identifica.

7.3.2 Caso de uso #1: Administrar deportes

Descripción.

Este caso de uso permite a los administradores gestionar los deportes de la aplicación.

Actores.

Actor principal: administrador.

Precondiciones.

El administrador debe estar identificado para tener acceso al panel de administración.

Postcondiciones.

Las modificaciones de los deportes efectuadas por el administrador persisten en la base de datos.

Flujo de eventos.**1. Flujo principal.****1.1. AÑADIR UN NUEVO DEPORTE.**

El sistema muestra la lista de acciones de las que dispone el usuario con privilegios de administración del sistema. El administrador elige '*Añadir un nuevo deporte*'.

1.2. COMPLETAR LOS DATOS DEL DEPORTE.

El administrador proporciona y envía los datos del nuevo deporte. Deberá indicar imprescindiblemente el nombre y la descripción. Además, el nombre deberá de ser único.

1.3. GUARDAR EL DEPORTE.

El sistema persiste el deporte en la base de datos, mostrando una confirmación de que el proceso se ha realizado con éxito.

2. Flujos alternativos.

2.1. MODIFICAR UN DEPORTE EXISTENTE.

Una vez realizado el paso de AÑADIR UN NUEVO DEPORTE, el sistema permite mostrar la lista de todos los deportes creados y editar cada uno de ellos para modificar sus datos, seleccionando '*Modificar el deporte*'. El flujo continúa en el paso de GUARDAR EL DEPORTE.

2.2. ELIMINAR UN DEPORTE EXISTENTE.

Una vez realizado el paso de AÑADIR UN NUEVO DEPORTE, el sistema permite mostrar la lista de todos los deportes creados y eliminar cada uno de ellos, seleccionando '*Eliminar el deporte*'. El sistema reclama una confirmación por parte del administrador y una vez confirmado se eliminan los datos del deporte y todos los datos asociados a este. El flujo termina.

2.3. FILTRAR DEPORTES.

Una vez realizado el paso de AÑADIR UN NUEVO DEPORTE, el sistema permite filtrar la lista de todos los deportes creados, de forma que se visualicen solo aquellos que cumplan los criterios de búsqueda. El flujo continúa por el mismo paso.

2.4. CERRAR SESIÓN.

En cualquier momento el usuario administrador podrá cerrar la sesión. El flujo termina.

7.3.3 Caso de uso #2: Administrar actividades

Descripción.

Este caso de uso permite a los administradores gestionar las actividades de la aplicación.

Actores.

Actor principal: administrador.

Precondiciones.

El administrador debe estar identificado para tener acceso al panel de administración.

Postcondiciones.

Las modificaciones de las actividades efectuadas por el administrador persisten en la base de datos.

Flujo de eventos.

1. Flujo principal.

1.1. AÑADIR UNA NUEVA ACTIVIDAD.

El sistema muestra la lista de acciones de las que dispone el usuario con privilegios de administración del sistema. El administrador elige '*Añadir una nueva actividad*'.

1.2. COMPLETAR LOS DATOS DE LA ACTIVIDAD.

El administrador proporciona y envía los datos de la nueva actividad. Deberá indicar imprescindiblemente el título, el inicio, la duración y el creador. Además, podrá añadir una descripción, el lugar y las plazas disponibles. Si no se especifican plazas, el sistema entenderá que son ilimitadas.

1.3. GUARDAR LA ACTIVIDAD.

El sistema persiste la actividad en la base de datos, mostrando una confirmación de que el proceso se ha realizado con éxito.

2. Flujos alternativos.

2.1. MODIFICAR UNA ACTIVIDAD EXISTENTE.

Una vez realizado el paso de AÑADIR UNA NUEVA ACTIVIDAD, el sistema permite mostrar la lista de todas las actividades creadas y editar cada una de ellas para modificar sus datos seleccionando '*Modificar la actividad*'. El flujo continúa en el paso de GUARDAR LA ACTIVIDAD.

2.2. ELIMINAR UNA ACTIVIDAD EXISTENTE.

Una vez realizado el paso de AÑADIR UNA NUEVA ACTIVIDAD, el sistema permite mostrar la lista de todas las actividades creadas y eliminar cada una de ellas seleccionando '*Eliminar la actividad*'. El sistema reclama una confirmación por parte del administrador y una vez confirmado se eliminan los datos de la actividad y todos los datos asociados a esta. El flujo termina.

2.3. FILTRAR ACTIVIDADES.

Una vez realizado el paso de AÑADIR UNA NUEVA ACTIVIDAD, el sistema permite filtrar la lista de todas las actividades creadas, de forma que se visualicen solo aquellas que cumplan los criterios de búsqueda. El flujo continúa por el mismo paso.

2.4. CERRAR SESIÓN.

En cualquier momento el usuario administrador podrá cerrar la sesión. El flujo termina.

7.3.4 Caso de uso #3: Administrar registros

Descripción.

Este caso de uso permite a los administradores gestionar los registros de los usuarios en las actividades.

Actores.

Actor principal: administrador.

Precondiciones.

El administrador debe estar identificado para tener acceso al panel de administración.

Postcondiciones.

Las modificaciones de los registros efectuadas por el administrador persisten en la base de datos.

Flujo de eventos.**1. Flujo principal.****1.1. AÑADIR UN NUEVO REGISTRO.**

El sistema muestra la lista de acciones de las que dispone el usuario con privilegios de administración del sistema. El administrador elige '*Añadir un nuevo registro*'.

1.2. COMPLETAR LOS DATOS DEL REGISTRO.

El administrador proporciona y envía los datos del nuevo registro. Deberá indicar imprescindiblemente el tipo, el estado, el usuario y la actividad. Además, se guardará la fecha y hora de creación del registro automáticamente.

1.3. GUARDAR EL REGISTRO.

El sistema persiste el registro en la base de datos, mostrando una confirmación de que el proceso se ha realizado con éxito.

2. Flujos alternativos.**2.1. MODIFICAR UN REGISTRO EXISTENTE.**

Una vez realizado el paso de AÑADIR UN NUEVO REGISTRO, el sistema permite mostrar la lista de todos los registros creados y editar cada uno de ellos para modificar sus datos, seleccionando '*Modificar el registro*'. El flujo continúa en el paso de GUARDAR EL REGISTRO.

2.2. ELIMINAR UN REGISTRO EXISTENTE.

Una vez realizado el paso de AÑADIR UN NUEVO REGISTRO, el sistema permite mostrar la lista de todos los registros creados y eliminar cada uno de ellos, seleccionando '*Eliminar el registro*'. El sistema reclama una confirmación por parte del administrador y una vez confirmado se eliminan los datos del registro y todos los datos asociados a este. El flujo termina.

2.3. FILTRAR REGISTROS.

Una vez realizado el paso de AÑADIR UN NUEVO REGISTRO, el sistema permite filtrar la lista de todos los registros creados, de forma que se visualicen solo aquellos que cumplan los criterios de búsqueda. El flujo continúa por el mismo paso.

2.4. CERRAR SESIÓN.

En cualquier momento el usuario administrador podrá cerrar la sesión. El flujo termina.

7.3.5 Caso de uso #4: Administrar usuarios

Descripción.

Este caso de uso permite a los administradores gestionar los usuarios de la aplicación.

Actores.

Actor principal: administrador.

Actor secundario: usuario.

Precondiciones.

El usuario debe estar identificado si no es el caso de registro.

Postcondiciones.

Las modificaciones de los usuarios efectuadas por el administrador persisten en la base de datos.

Flujo de eventos.

1. Flujo principal.

1.1. AÑADIR UN NUEVO USUARIO.

El sistema muestra la lista de acciones de las que dispone el usuario con privilegios de administración del sistema. El administrador elige '*Añadir un nuevo usuario*'.

1.2. COMPLETAR LOS DATOS DEL USUARIO.

El administrador proporciona y envía los datos del nuevo usuario. Deberá indicar imprescindiblemente el correo electrónico, el nombre de usuario, la contraseña y el rol. Por defecto, los usuarios registrados estarán habilitados.

1.3. GUARDAR EL USUARIO.

El sistema persiste el usuario en la base de datos, mostrando una confirmación de que el proceso se ha realizado con éxito.

2. Flujos alternativos.

2.1. MODIFICAR UN USUARIO EXISTENTE.

Una vez realizado el paso de AÑADIR UN NUEVO USUARIO, el sistema permite mostrar la lista de todos los usuarios creados y editar cada uno de ellos para modificar sus datos, seleccionando '*Modificar el usuario*'. El flujo continúa en el paso de GUARDAR EL USUARIO.

2.2. ELIMINAR UN USUARIO EXISTENTE.

Una vez realizado el paso de AÑADIR UN NUEVO USUARIO, el sistema permite mostrar la lista de todos los usuarios creados y eliminar cada uno de ellos, seleccionando '*Eliminar el usuario*'. El sistema reclama una confirmación por parte del administrador y una vez confirmado se eliminan los datos del usuario y todos los datos asociados a este. El flujo termina.

2.3. FILTRAR USUARIOS.

Una vez realizado el paso de AÑADIR UN NUEVO USUARIO, el sistema permite filtrar la lista de todos los usuarios creados, de forma que se visualicen solo aquellos que cumplan los criterios de búsqueda. El flujo continúa por el mismo paso.

2.4. CERRAR SESIÓN.

En cualquier momento el usuario administrador podrá cerrar la sesión. El flujo termina.

7.3.6 Caso de uso #5: Invitar a participar

Descripción.

Este caso de uso permite a los usuarios invitar a participar a otros en sus actividades.

Actores.

Actor principal: usuario.

Precondiciones.

El usuario debe estar identificado.

Postcondiciones.

Las invitaciones de los usuarios y las notificaciones pertinentes persisten en la base de datos.

Flujo de eventos.

1. Flujo principal.

1.1. AÑADIR UNA NUEVA INVITACIÓN.

El sistema muestra la lista de acciones de las que dispone el usuario del sistema. El usuario elige '*Añadir una nueva invitación*'.

1.2. ELEGIR USUARIO.

El sistema proporciona y envía la lista de usuarios a los que invitar. El usuario deberá indicar el usuario a invitar.

1.3. GUARDAR INVITACIÓN.

El sistema persiste la invitación en la base de datos, mostrando una confirmación de que el proceso se ha realizado con éxito.

1.4. NOTIFICAR INVITACIÓN.

El sistema notifica de la invitación al usuario implicado y la persiste en la base de datos.

2. Flujos alternativos.**2.1. CERRAR SESIÓN.**

En cualquier momento el usuario podrá cerrar la sesión. El flujo termina.

7.3.7 Caso de uso #6: Cancelar invitaciones

Descripción.

Este caso de uso permite a los usuarios cancelar invitaciones a sus actividades.

Actores.

Actor principal: usuario.

Precondiciones.

El usuario debe estar identificado.

Postcondiciones.

Las cancelaciones de invitación de los usuarios y las notificaciones pertinentes persisten en la base de datos.

Flujo de eventos.**1. Flujo principal.****1.1. CANCELAR UNA INVITACIÓN.**

El sistema muestra la lista de acciones de las que dispone el usuario del sistema. El usuario elige '*'Cancelar una invitación'*'.

1.2. ELEGIR INVITACIÓN.

El sistema proporciona y envía la lista de invitaciones. El usuario deberá indicar la invitación a cancelar.

1.3. GUARDAR CANCELACIÓN.

El sistema persiste la cancelación en la base de datos, mostrando una confirmación de que el proceso se ha realizado con éxito.

1.4. NOTIFICAR CANCELACIÓN.

El sistema notifica de la cancelación al usuario implicado y la persiste en la base de datos.

2. Flujos alternativos.**2.1. CERRAR SESIÓN.**

En cualquier momento el usuario podrá cerrar la sesión. El flujo termina.

7.3.8 Caso de uso #7: Aceptar invitaciones

Descripción.

Este caso de uso permite a los usuarios aceptar invitaciones de otros a sus actividades.

Actores.

Actor principal: usuario.

Precondiciones.

El usuario debe estar identificado.

Postcondiciones.

Las aceptaciones y las notificaciones pertinentes persisten en la base de datos.

Flujo de eventos.**1. Flujo principal.****1.1. VER INVITACIONES PENDIENTES.**

El sistema muestra la lista de invitaciones pendientes de las que dispone el usuario del sistema.

1.2. ELEGIR INVITACIÓN.

El sistema proporciona y envía la lista de invitaciones. El usuario deberá indicar la invitación que desea aceptar e indicar que acepta.

1.3. GUARDAR ACEPTACIÓN.

El sistema persiste la aceptación en la base de datos, mostrando una confirmación de que el proceso se ha realizado con éxito.

1.4. NOTIFICAR ACEPTACIÓN.

El sistema notifica de la aceptación al usuario implicado y la persiste en la base de datos.

2. Flujos alternativos.**2.1. CERRAR SESIÓN.**

En cualquier momento el usuario podrá cerrar la sesión. El flujo termina.

7.3.9 Caso de uso #8: Rechazar invitaciones

Descripción.

Este caso de uso permite a los usuarios rechazar invitaciones de otros a sus actividades.

Actores.

Actor principal: usuario.

Precondiciones.

El usuario debe estar identificado.

Postcondiciones.

Los rechazos y las notificaciones pertinentes persisten en la base de datos.

Flujo de eventos.

1. Flujo principal.

1.1. VER INVITACIONES PENDIENTES.

El sistema muestra la lista de invitaciones pendientes de las que dispone el usuario del sistema.

1.2. ELEGIR INVITACIÓN.

El sistema proporciona y envía la lista de invitaciones. El usuario deberá indicar la invitación que desea rechazar e indicar que rechaza.

1.3. GUARDAR RECHAZO.

El sistema persiste el rechazo en la base de datos, mostrando una confirmación de que el proceso se ha realizado con éxito.

1.4. NOTIFICAR RECHAZO.

El sistema notifica del rechazo al usuario implicado y lo persiste en la base de datos.

2. Flujos alternativos.

2.1. CERRAR SESIÓN.

En cualquier momento el usuario podrá cerrar la sesión. El flujo termina.

7.3.10 Caso de uso #9: Solicitar participaciones

Descripción.

Este caso de uso permite a los usuarios solicitar su participación en actividades.

Actores.

Actor principal: usuario.

Precondiciones.

El usuario debe estar identificado.

Postcondiciones.

Las solicitudes de los usuarios y las notificaciones pertinentes persisten en la base de datos.

Flujo de eventos.**1. Flujo principal.****1.1. AÑADIR UNA NUEVA SOLICITUD.**

El sistema muestra la lista de acciones de las que dispone el usuario del sistema.

El usuario elige '*Añadir una nueva solicitud*'.

1.2. ELEGIR ACTIVIDAD.

El sistema proporciona y envía la lista de actividades que solicitar. El usuario deberá indicar la actividad a solicitar.

1.3. GUARDAR SOLICITUD.

El sistema persiste la solicitud en la base de datos, mostrando una confirmación de que el proceso se ha realizado con éxito.

1.4. NOTIFICAR SOLICITUD.

El sistema notifica de la solicitud al usuario implicado y la persiste en la base de datos.

1. Flujos alternativos.**1.1. CERRAR SESIÓN.**

En cualquier momento el usuario podrá cerrar la sesión. El flujo termina.

7.3.11 Caso de uso #10: Cancelar solicitudes

Descripción.

Este caso de uso permite a los usuarios cancelar solicitudes de participación a actividades.

Actores.

Actor principal: usuario.

Precondiciones.

El usuario debe estar identificado.

Postcondiciones.

Las cancelaciones de solicitud de los usuarios y las notificaciones pertinentes persisten

en la base de datos.

Flujo de eventos.

1. Flujo principal.

1.1. CANCELAR UNA SOLICITUD.

El sistema muestra la lista de acciones de las que dispone el usuario del sistema.

El usuario elige '*Cancelar una solicitud*'.

1.2. ELEGIR SOLICITUD.

El sistema proporciona y envía la lista de solicitudes. El usuario deberá indicar la solicitud a cancelar.

1.3. GUARDAR CANCELACIÓN.

El sistema persiste la cancelación en la base de datos, mostrando una confirmación de que el proceso se ha realizado con éxito.

1.4. NOTIFICAR CANCELACIÓN.

El sistema notifica de la cancelación al usuario implicado y la persiste en la base de datos.

2. Flujos alternativos.

2.1. CERRAR SESIÓN.

En cualquier momento el usuario podrá cerrar la sesión. El flujo termina.

7.3.12 Caso de uso #11: Aceptar solicitudes

Descripción.

Este caso de uso permite a los usuarios aceptar solicitudes de otros a sus actividades.

Actores.

Actor principal: usuario.

Precondiciones.

El usuario debe estar identificado.

Postcondiciones.

Las aceptaciones y las notificaciones pertinentes persisten en la base de datos.

Flujo de eventos.

1. Flujo principal.

1.1. VER SOLICITUDES PENDIENTES.

El sistema muestra la lista de solicitudes pendientes de las que dispone el usuario del sistema.

1.2. ELEGIR SOLICITUD.

El sistema proporciona y envía la lista de solicitudes. El usuario deberá indicar la solicitud que desea aceptar e indicar que acepta.

1.3. GUARDAR ACEPTACIÓN.

El sistema persiste la aceptación en la base de datos, mostrando una confirmación de que el proceso se ha realizado con éxito.

1.4. NOTIFICAR ACEPTACIÓN.

El sistema notifica de la aceptación al usuario implicado y la persiste en la base de datos.

2. Flujos alternativos.**2.1. CERRAR SESIÓN.**

En cualquier momento el usuario podrá cerrar la sesión. El flujo termina.

7.3.13 Caso de uso #12: Rechazar solicitudes

Descripción.

Este caso de uso permite a los usuarios rechazar solicitudes de otros a sus actividades.

Actores.

Actor principal: usuario.

Precondiciones.

El usuario debe estar identificado.

Postcondiciones.

Los rechazos y las notificaciones pertinentes persisten en la base de datos.

Flujo de eventos.**1. Flujo principal.****1.1. VER SOLICITUDES PENDIENTES.**

El sistema muestra la lista de solicitudes pendientes de las que dispone el usuario del sistema.

1.2. ELEGIR SOLICITUD.

El sistema proporciona y envía la lista de solicitudes. El usuario deberá indicar la solicitud que desea rechazar e indicar que rechaza.

1.3. GUARDAR RECHAZO.

El sistema persiste el rechazo en la base de datos, mostrando una confirmación de que el proceso se ha realizado con éxito.

1.4. NOTIFICAR RECHAZO.

El sistema notifica del rechazo al usuario implicado y lo persiste en la base de datos.

2. Flujos alternativos.**2.1. CERRAR SESIÓN.**

En cualquier momento el usuario podrá cerrar la sesión. El flujo termina.

7.3.14 Caso de uso #13: Eliminar participaciones**Descripción.**

Este caso de uso permite a los usuarios eliminar su participación en actividades.

Actores.

Actor principal: usuario.

Precondiciones.

El usuario debe estar identificado.

Postcondiciones.

La eliminación y las notificaciones pertinentes persisten en la base de datos.

Flujo de eventos.**1. Flujo principal.****1.1. VER PARTICIPACIONES.**

El sistema muestra la lista de participaciones del usuario del sistema.

1.2. ELEGIR PARTICIPACIÓN.

El sistema proporciona y envía la lista de participaciones. El usuario deberá indicar la participación que desea eliminar.

1.3. ELIMINAR PARTICIPACIÓN.

El sistema persiste la eliminación en la base de datos, mostrando una confirmación de que el proceso se ha realizado con éxito.

1.4. NOTIFICAR ELIMINACIÓN.

El sistema notifica de la eliminación al usuario implicado y la persiste en la base de datos.

2. Flujos alternativos.

2.1. CERRAR SESIÓN.

En cualquier momento el usuario podrá cerrar la sesión. El flujo termina.

7.3.15 Caso de uso #14: Eliminar participantes

Descripción.

Este caso de uso permite a los usuarios eliminar participantes de sus actividades.

Actores.

Actor principal: usuario.

Precondiciones.

El usuario debe estar identificado.

Postcondiciones.

La eliminación y las notificaciones pertinentes persisten en la base de datos.

Flujo de eventos.

1. Flujo principal.

1.1. VER PARTICIPANTES.

El sistema muestra la lista de participantes en la actividad.

1.2. ELEGIR PARTICIPANTE.

El sistema proporciona y envía la lista de participantes. El usuario deberá indicar el participante que desea eliminar.

1.3. ELIMINAR PARTICIPANTE.

El sistema persiste la eliminación en la base de datos, mostrando una confirmación de que el proceso se ha realizado con éxito.

1.4. NOTIFICAR ELIMINACIÓN.

El sistema notifica de la eliminación al usuario implicado y la persiste en la base de datos.

2. Flujos alternativos.

2.1. CERRAR SESIÓN.

En cualquier momento el usuario podrá cerrar la sesión. El flujo termina.

7.3.16 Caso de uso #15: Administrar notificaciones

Descripción.

Este caso de uso permite a los usuarios administradores gestionar las notificaciones de la aplicación.

Actores.

Actor principal: administrador.

Precondiciones.

El administrador debe estar identificado para tener acceso al panel de administración.

Postcondiciones.

Las modificaciones de las notificaciones efectuadas por el administrador persisten en la base de datos.

Flujo de eventos.**1. Flujo principal.****1.1. AÑADIR UNA NUEVA NOTIFICACIÓN.**

El sistema muestra la lista de acciones de las que dispone el usuario con privilegios de administración del sistema. El administrador elige '*Añadir una nueva notificación*'.

1.2. COMPLETAR LOS DATOS DE LA NOTIFICACIÓN.

El administrador proporciona y envía los datos de la nueva notificación. Deberá indicar imprescindiblemente el tipo (solicitud, invitación, aceptación o rechazo), el destinatario y la actividad.

1.3. ENVIAR LA NOTIFICACIÓN.

El sistema persiste la notificación en la base de datos y la envía, mostrando una confirmación de que el proceso se ha realizado con éxito.

2. Flujos alternativos.**2.1. MODIFICAR UNA NOTIFICACIÓN EXISTENTE.**

Una vez realizado el paso de AÑADIR UNA NUEVA NOTIFICACIÓN, el sistema permite mostrar la lista de todas las notificaciones creadas y editar cada una de ellas para modificar sus datos, seleccionando '*Modificar la notificación*'. El flujo continúa en el paso de GUARDAR LA NOTIFICACIÓN.

2.2. ELIMINAR UNA NOTIFICACIÓN EXISTENTE.

Una vez realizado el paso de AÑADIR UNA NUEVA NOTIFICACIÓN, el sistema permite mostrar la lista de todas las notificaciones creadas y eliminar cada una de ellas, seleccionando '*Eliminar la notificación*'. El sistema reclama una confirmación por parte del administrador y una vez confirmado se eliminan los datos de la notificación y todos los datos asociados a esta. El flujo termina.

2.3. FILTRAR NOTIFICACIONES.

Una vez realizado el paso de AÑADIR UNA NUEVA NOTIFICACIÓN, el sistema permite filtrar la lista de todas las notificaciones creadas, de forma que se visualicen solo aquellas que cumplan los criterios de búsqueda. El flujo continúa por el mismo paso.

2.4. CERRAR SESIÓN.

En cualquier momento el usuario administrador podrá cerrar la sesión. El flujo termina.

7.3.17 Caso de uso #16: Administrar comunicaciones

Descripción.

Este caso de uso permite a los usuarios administradores gestionar las comunicaciones de la aplicación.

Actores.

Actor principal: administrador.

Precondiciones.

El administrador debe estar identificado para tener acceso al panel de administración.

Postcondiciones.

Las modificaciones de las comunicaciones efectuadas por el administrador persisten en la base de datos.

Flujo de eventos.

1. Flujo principal.

1.1. AÑADIR UNA NUEVA COMUNICACIÓN.

El sistema muestra la lista de acciones de las que dispone el usuario con privilegios de administración del sistema. El administrador elige '*Añadir una nueva comunicación*'.

1.2. COMPLETAR LOS DATOS DE LA COMUNICACIÓN.

El administrador proporciona y envía los datos de la nueva comunicación. Deberá indicar imprescindiblemente el asunto, el mensaje y el destinatario.

1.3. GUARDAR LA COMUNICACIÓN.

El sistema persiste la comunicación en la base de datos y la envía, mostrando una confirmación de que el proceso se ha realizado con éxito.

2. Flujos alternativos.

2.1. MODIFICAR UNA COMUNICACIÓN EXISTENTE.

Una vez realizado el paso de AÑADIR UNA NUEVA COMUNICACIÓN, el sistema permite mostrar la lista de todas las comunicaciones creadas y editar cada una de ellas para modificar sus datos, seleccionando '*Modificar la comunicación*'. El flujo continúa en el paso de GUARDAR LA NOTIFICACIÓN.

2.2. ELIMINAR UNA COMUNICACIÓN EXISTENTE.

Una vez realizado el paso de AÑADIR UNA NUEVA COMUNICACIÓN, el sistema permite mostrar la lista de todas las comunicaciones creadas y eliminar cada una de ellas, seleccionando '*Eliminar la comunicación*'. El sistema reclama una confirmación por parte del usuario y una vez confirmado se eliminan los datos de la comunicación y todos los datos asociados a esta. El flujo termina.

2.3. FILTRAR COMUNICACIONES.

Una vez realizado el paso de AÑADIR UNA NUEVA COMUNICACIÓN, el sistema permite filtrar la lista de todas las comunicaciones creadas, de forma que se visualicen solo aquellas que cumplan los criterios de búsqueda. El flujo continúa por el mismo paso.

2.4. CERRAR SESIÓN.

En cualquier momento el usuario administrador podrá cerrar la sesión. El flujo termina.

7.3.18 Caso de uso #17: Editar perfil

Descripción.

Este caso de uso permite a los usuarios editar su perfil de la aplicación.

Actores.

Actor principal: usuario.

Precondiciones.

El usuario debe estar identificado.

Postcondiciones.

Las modificaciones de los usuarios de su perfil persisten en la base de datos.

Flujo de eventos.

1. Flujo principal.

1.1. EDITAR PERFIL.

El sistema muestra la lista de acciones de las que dispone el usuario del sistema. El usuario elige '*Editar perfil*'.

1.2. MODIFICAR PERFIL.

El sistema proporciona y envía los datos actuales del usuario. El usuario deberá indicar los nuevos datos.

1.3. GUARDAR PERFIL.

El sistema persiste las modificaciones del usuario en la base de datos, mostrando una confirmación de que el proceso se ha realizado con éxito.

2. Flujos alternativos.**2.1. CERRAR SESIÓN.**

En cualquier momento el usuario podrá cerrar la sesión. El flujo termina.

7.3.19 Caso de uso #18: Eliminar perfil

Descripción.

Este caso de uso permite a los usuarios eliminar su perfil de la aplicación.

Actores.

Actor principal: usuario.

Precondiciones.

El usuario debe estar identificado.

Postcondiciones.

El perfil del usuario ya no persiste en la base de datos.

Flujo de eventos.**1. Flujo principal.****1.1. ELIMINAR PERFIL.**

El sistema muestra la lista de acciones de las que dispone el usuario del sistema. El usuario elige '*Eliminar perfil*'.

1.2. CONFIRMA ELIMINACIÓN.

El sistema envía un mensaje de confirmación al usuario. El usuario deberá confirmar que desea borrar su perfil de la aplicación.

1.3. BORRAR PERFIL.

El sistema deja de persistir el perfil del usuario en la base de datos, mostrando una confirmación de que el proceso se ha realizado con éxito.

2. Flujos alternativos.**2.1. CERRAR SESIÓN.**

En cualquier momento el usuario podrá cerrar la sesión. El flujo termina.

7.3.20 Caso de uso #19: Ver actividades

Descripción.

Este caso de uso permite a los usuarios anónimos ver la lista de actividades de la aplicación.

Actores.

Actor principal: usuario anónimo.

Flujo de eventos.

1. Flujo principal.

1.1. VER ACTIVIDADES.

El sistema muestra la lista de acciones de las que dispone el usuario no identificado por el sistema. El usuario elige '*Ver actividades*'.

1.2. LISTAR ACTIVIDADES.

El sistema envía la lista de actividades al usuario.

7.4 Requisitos no funcionales

A continuación se presentan requisitos que no son visibles para los usuarios y que no incluyen una relación directa con el comportamiento funcional del sistema. Se usará la siguiente nomenclatura: *RNFnn*, donde *nn* indica un número consecutivo que sirve para identificar al requisito.

RNF01 La respuesta del sistema debe ser la menor posible para ofrecer una mejor experiencia de usuario.

RNF02 La interfaz debe ser clara y ofrecer guías o explicaciones, de tal manera que su uso no requiera de entrenamiento previo.

RNF03 El sistema debe presentar mensajes de error que permitan al usuario identificar el problema.

RNF04 El sistema debe ofrecer facilidad para comprobar posibles errores durante la fase de pruebas de la interfaz.

RNF05 El sistema debe proporcionar mecanismos para facilitar la instalación y configuración del software.

RNF06 La seguridad del sistema debe estar garantizada, confirmando que usuarios de roles inferiores no usurpen actividades de roles superiores

RNF07 La integridad de los datos debe estar garantizada, de tal manera que se asegure que no hay datos incorrectos.

RNF08 La seguridad de la capa pública del sistema (*API*) debe estar garantizada.

Capítulo 8

Análisis de la información

Procederemos a describir la información que será manipulada por el sistema, la cual ha sido definida a partir del estudio del problema.

8.1 Descripción del modelo conceptual de datos

Para analizar el modelo de datos, usaremos la metodología UML para definir el modelo de clases. Primero indicaremos las clases que componen el sistema, una descripción de las mismas y las relaciones entre ellas. Más adelante, en el diseño del sistema mostraremos sus atributos y sus funcionalidades. La "Figura 8.1" muestra el modelo de clases general.

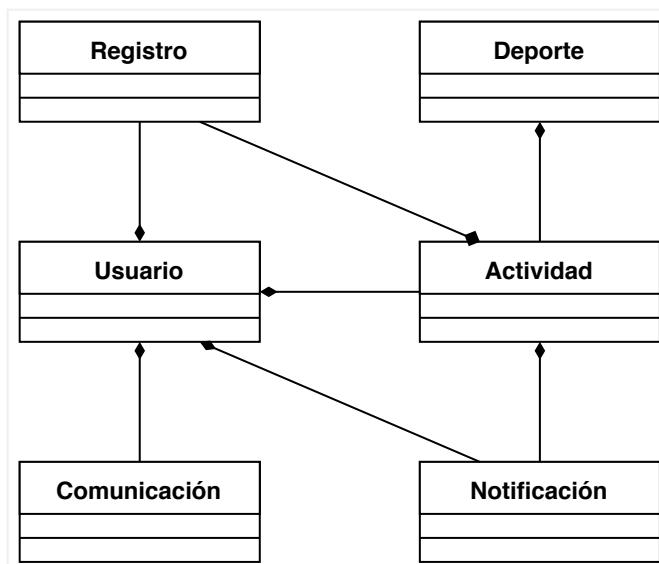


Figura 8.1 Análisis del modelo de clases

8.2 Descripción de las clases

8.2.1 Clase usuario

Representa la información correspondiente al usuario del sistema. Sus atributos son: identificador, nombre de usuario, contraseña, nombre completo, correo electrónico, avatar, registros, actividades y roles. Además, tendrá todos los atributos correspondientes a su autenticación y gestión comunes en todos los problemas que tienen este tipo de entidad.

8.2.2 Clase deporte

Representa la información correspondiente a un deporte del sistema. Sus atributos son: identificador, etiqueta, nombre, descripción y actividades.

8.2.3 Clase actividad

Representa la información correspondiente a una actividad organizada por un usuario del sistema. Sus atributos son: identificador, título, fecha y hora de comienzo, duración, lugar, descripción, plazas, registros y dueño.

8.2.4 Clase registro

Representa la información correspondiente a cualquier registro de un usuario en una actividad del sistema. Sus atributos son: identificador, tipo (solicitud o invitación), estado (aceptada, rechazada o en espera), fecha y hora de creación, usuario y actividad.

8.2.5 Clase notificación

Representa la información correspondiente a una notificación del sistema a un determinado usuario de la aplicación. Sus atributos son: identificador, tipo (solicitud, invitación, aceptación o rechazo), fecha y hora de envío, destinatario y actividad.

8.2.6 Clase comunicación

Representa la información correspondiente a una comunicación del administrador a un determinado usuario de la aplicación. Sus atributos son: identificador, asunto, mensaje, fecha y hora de envío y destinatario.

Capítulo 9

Análisis funcional

9.1 Descripción arquitectónica

El diseño del sistema está orientado a uno de tipo *Modelo-Vista-Controlador*, que es el modelo que implica el desarrollo con *Symfony*. Recordamos que este es el entorno de desarrollo elegido durante la fase de restricciones.

Eso implica que el sistema está dividido en tres capas:

1. Capa de datos o modelo.
2. Capa de interfaz de usuario o vista.
3. Capa lógica o controlador.

Symfony ofrece una abstracción del uso de estas tres capas a través los siguientes elementos:

1. De la capa de datos, a través de un **ORM** llamado *Doctrine*^[1].
2. De la interfaz de usuario a través del sistema de plantillas *Twig*^[2].
3. La capa de control es *Symfony* en sí mismo. Se recomienda leer la bibliografía recomendada para comprender el funcionamiento del entorno de desarrollo (*Eguiluz, 2012*).

Para nuestro sistema, vamos a aprovechar esta abstracción para enfocar la arquitectura desde otra perspectiva, que nos permita crear componentes fácilmente desacoplables. Es decir, que sea fácil sustituir los componentes del sistema por otros nuevos, sin que el resto del sistema se vea afectado.

[1] *Doctrine*: <http://doctrine-project.org/>

[2] *Twig*: <https://twig.symfony.com/>

Para ello vamos a dividir el sistema en componentes funcionales, aglutinando las clases en componentes afines como vemos en la "Figura 9.1".

9.1.1 Capa usuario

Compuesta por la clase *usuario*. Será responsable de la administración de usuarios y la gestión de los permisos.

9.1.2 Capa deporte

Compuesta por la clase *deporte*. Será responsable de la administración y gestión de los deportes.

9.1.3 Capa actividad

Compuesta por la clase *actividad*. Será responsable de la funcionalidad de la administración y gestión de las actividades.

9.1.4 Capa registro

Compuesta por la clase *registro*. Será responsable de la funcionalidad de las solicitudes, invitaciones y participaciones.

9.1.5 Capa comunicación

Compuesta por la clase *comunicación*. Será responsable de la comunicación entre el administrador y los usuarios.

9.1.6 Capa notificación

Compuesta por la clase *notificación*. Será responsable de la notificación de eventos a los usuarios.

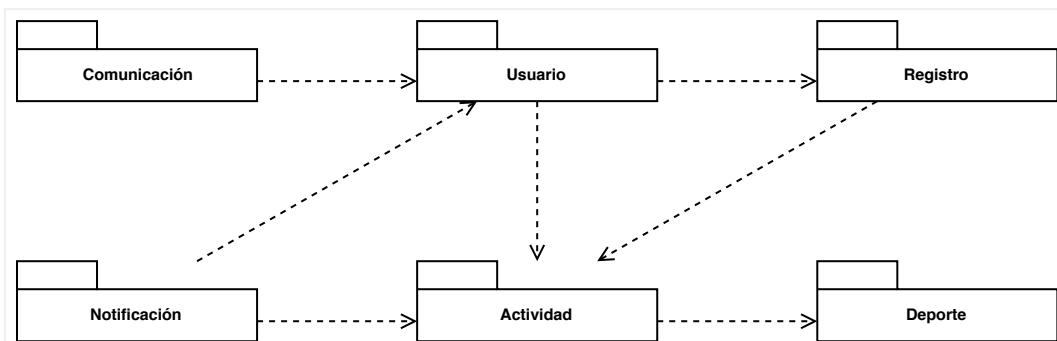


Figura 9.1 Diagrama de paquetes

9.2 Descripción del comportamiento

Los siguientes diagramas de secuencia describen la colaboración entre los distintos objetos del sistema:

9.2.1 Diagrama de secuencia: Crear deporte

La "Figura 9.2" muestra la secuencia de creación de un deporte. La acción (*CreateSportAction*) solicita una nueva instancia a la factoría (*SportFactory*) y establece el valor de sus atributos con los métodos *set* correspondientes. Finalmente, solicita al servicio (*SportManager*) que la guarde (*persist*) y ejecute la transacción (*flush*) usando el *ObjectManager*, y que lance el evento (*CreatedSportEvent*) a través del *EventDispatcher*.

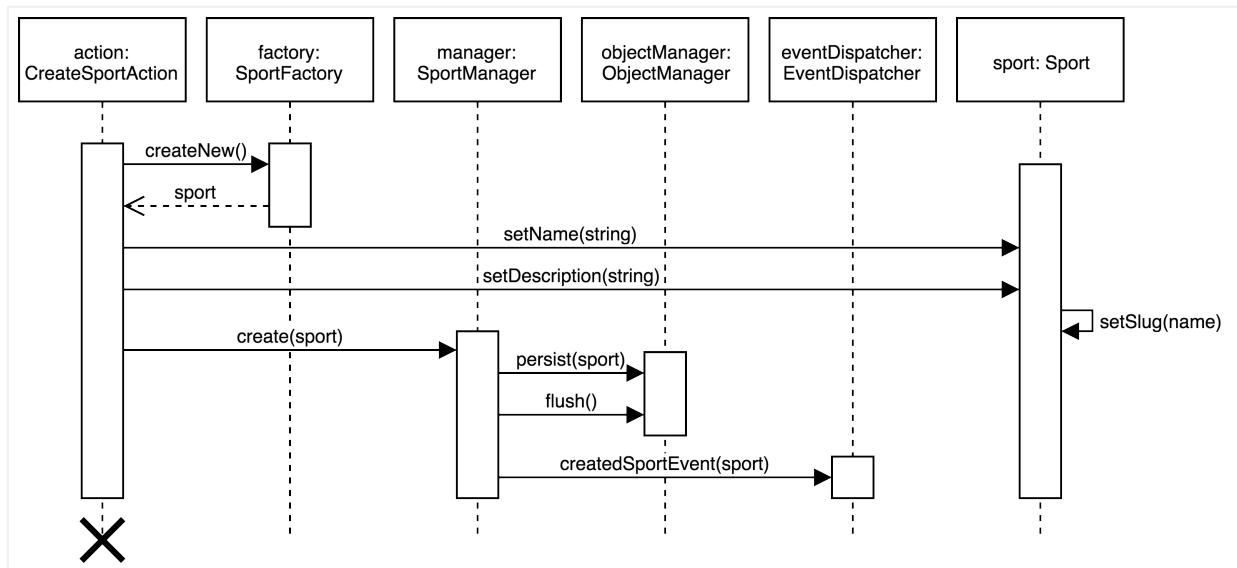


Figura 9.2 Diagrama de creación de deportes

9.2.2 Diagrama de secuencia: Listar deportes

La "Figura 9.3" muestra la secuencia para listar los deportes. La acción (*ReadSportsAction*) solicita la colección de datos paginados al repositorio (*SportRepository*).

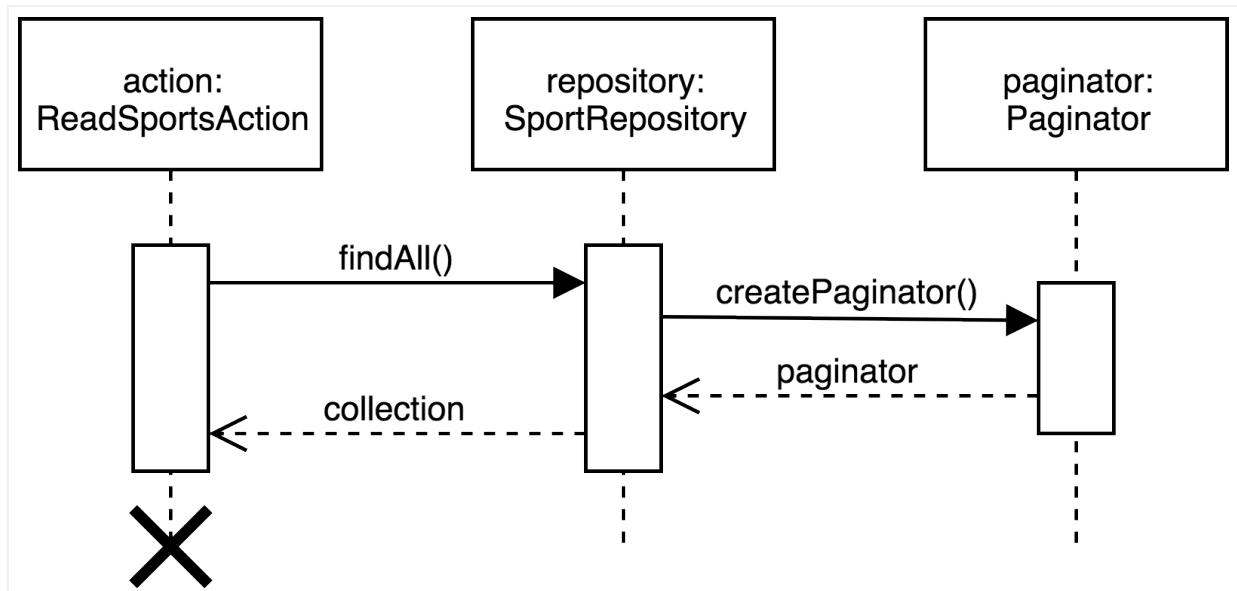


Figura 9.3 Diagrama de listado de deportes

9.2.3 Diagrama de secuencia: Actualizar deporte

La "Figura 9.4" muestra la secuencia de actualización de un deporte. La acción (*UpdateSportAction*) solicita la instancia al repositorio (*SportRepository*) y modifica el valor de sus atributos con los métodos *set* correspondientes. Finalmente, solicita al servicio (*SportManager*) que la guarde (*persist*) y ejecute la transacción (*flush*) usando el *ObjectManager*, y que lance el evento (*UpdatedSportEvent*) a través del *EventDispatcher*.

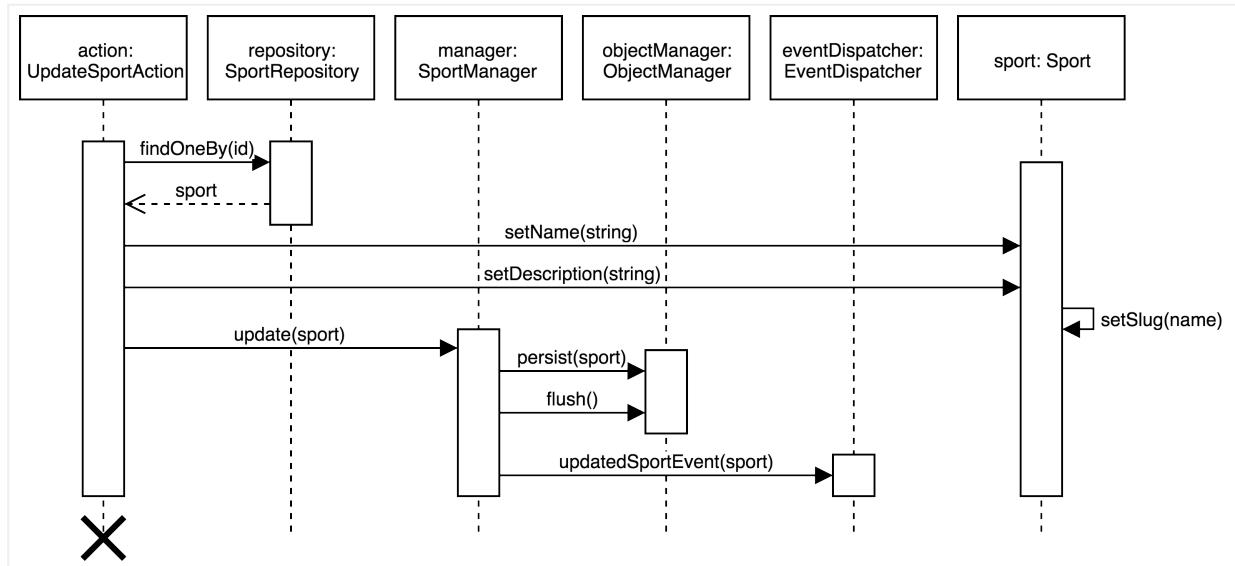


Figura 9.4 Diagrama de actualización de deportes

9.2.4 Diagrama de secuencia: Eliminar deporte

La "Figura 9.5" muestra la secuencia de borrado de un deporte. La acción (*DeleteSportAction*) solicita la instancia al repositorio (*SportRepository*). Finalmente, solicita al servicio (*SportManager*) que la elimine (*delete*) y ejecute la transacción (*flush*) usando el *ObjectManager*, y que lance el evento (*DeletedSportEvent*) a través del *EventDispatcher*.

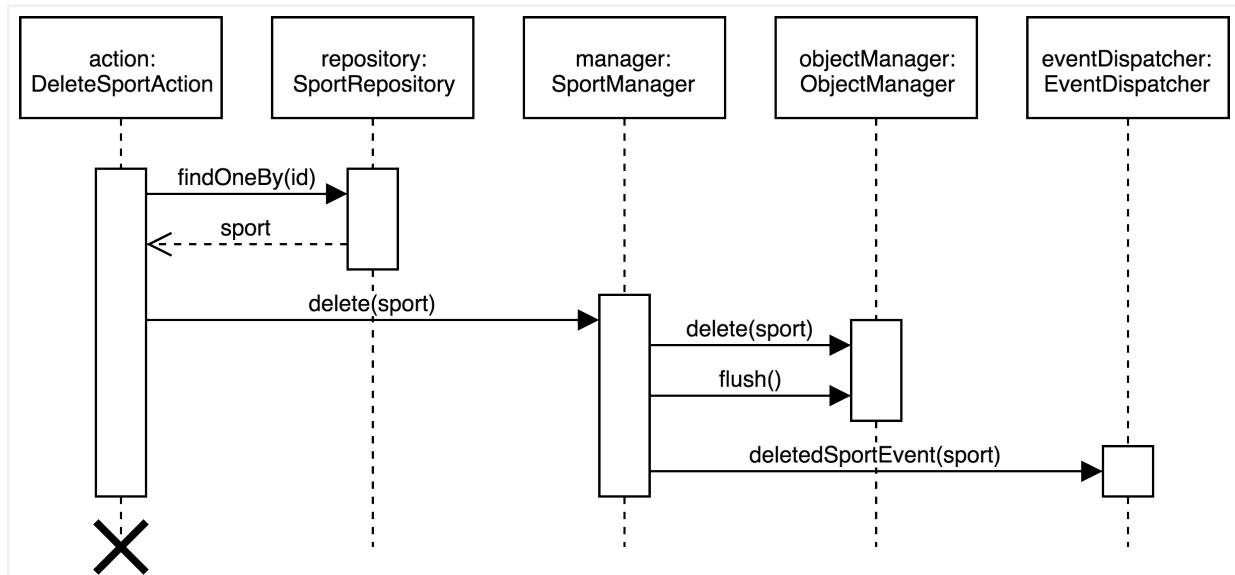


Figura 9.5 Diagrama de borrado de deportes

9.2.5 Diagrama de secuencia: Crear actividad

La "Figura 9.6" muestra la secuencia de creación de una actividad. La acción (*CreateActivityAction*) solicita una nueva instancia a la factoría (*ActivityFactory*) y establece el valor de sus atributos con los métodos *set* correspondientes. Finalmente, solicita al servicio (*ActivityManager*) que la guarde (*persist*) y ejecute la transacción (*flush*) usando el *ObjectManager*, y que lance el evento (*CreatedActivityEvent*) a través del *EventDispatcher*.

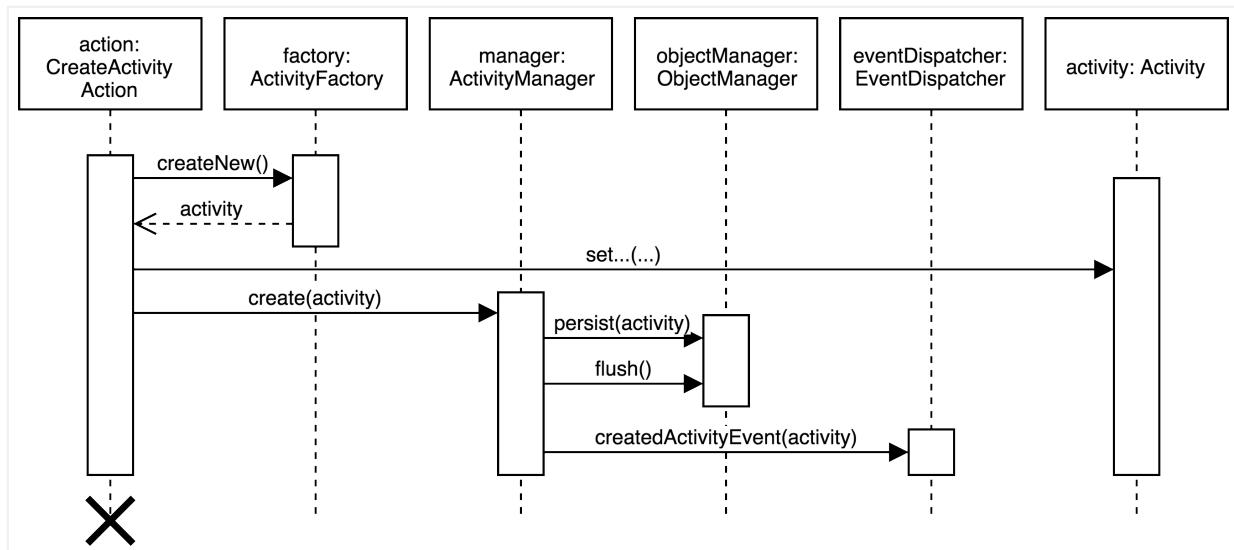


Figura 9.6 Diagrama de creación de actividades

9.2.6 Diagrama de secuencia: Listar actividades

La "Figura 9.7" muestra la secuencia para listar las actividades. La acción (*ReadActivitiesAction*) solicita la colección de datos paginados al repositorio (*ActivityRepository*).

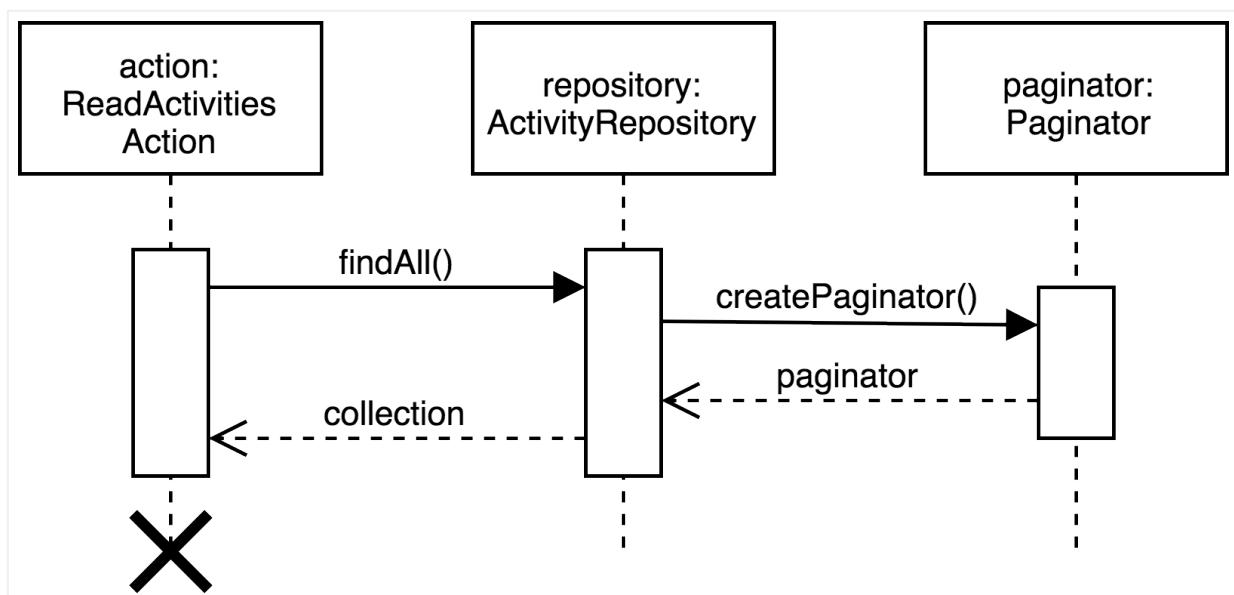


Figura 9.7 Diagrama de listado de actividades

9.2.7 Diagrama de secuencia: Actualizar actividad

La "Figura 9.8" muestra la secuencia de actualización de una actividad. La acción (*UpdateActivityAction*) solicita la instancia al repositorio (*ActivityRepository*) y modifica el valor de sus atributos con los métodos *set* correspondientes. Solicita al servicio (*ActivityManager*) que la guarde (*persist*) y ejecute la transacción (*flush*) usando el *ObjectManager*, y que lance el evento (*UpdatedActivityEvent*) a través del *EventDispatcher*.

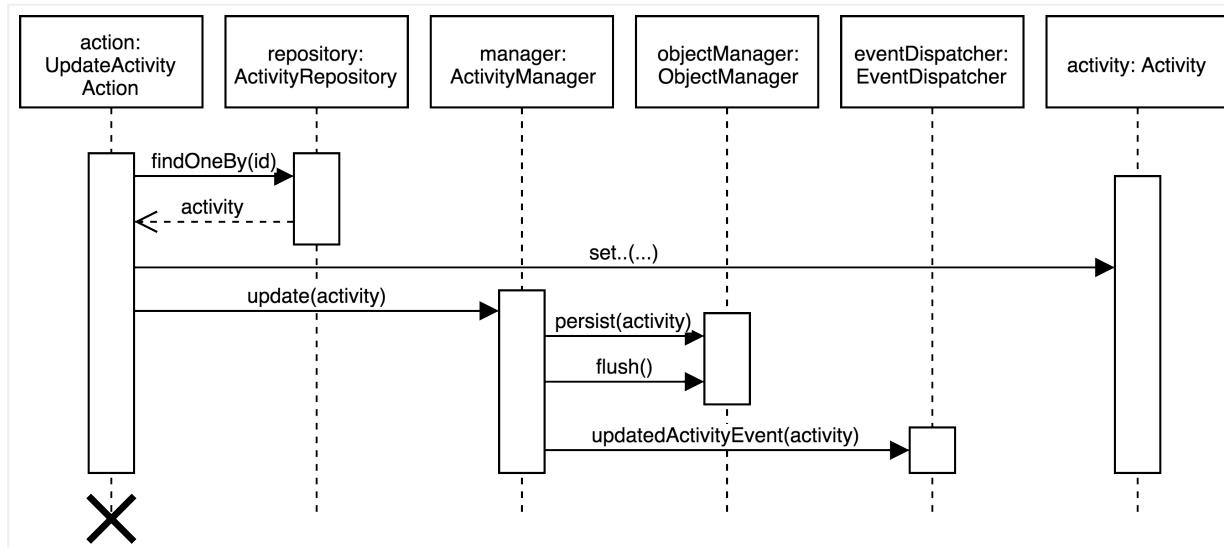


Figura 9.8 Diagrama de actualización de actividades

9.2.8 Diagrama de secuencia: Eliminar actividad

La "Figura 9.9" muestra la secuencia de borrado de una actividad. La acción (*DeleteActivityAction*) solicita la instancia al repositorio (*ActivityRepository*). Solicita al servicio (*ActivityManager*) que la elimine (*delete*) y ejecute la transacción (*flush*) usando el *ObjectManager*, y que lance el evento (*DeletedActivityEvent*) a través del *EventDispatcher*.

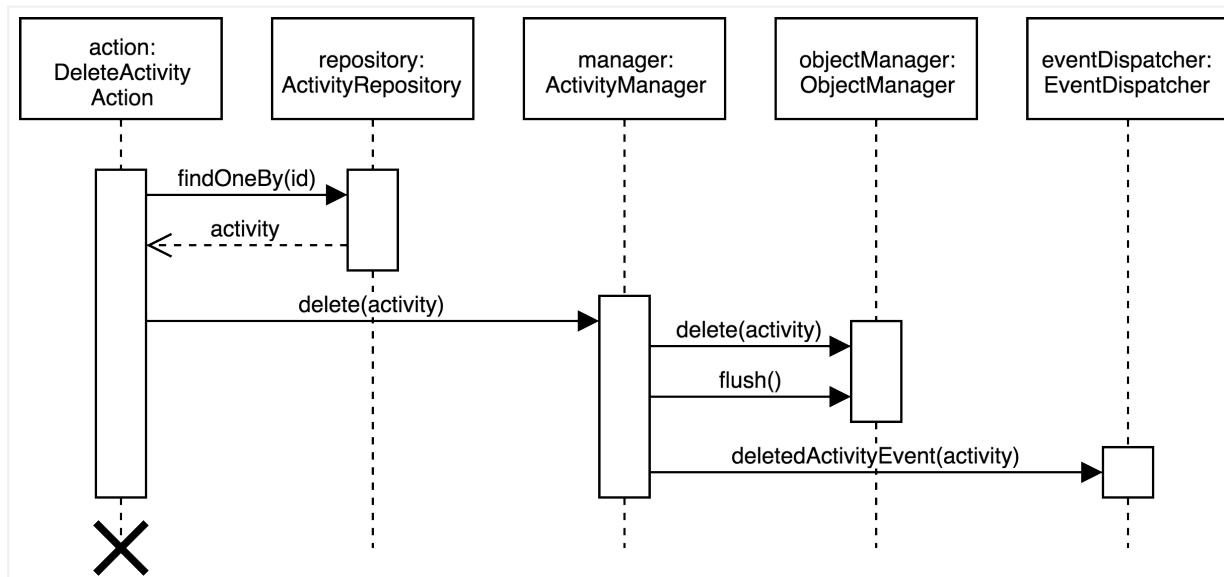


Figura 9.9 Diagrama de borrado de actividades

9.2.9 Diagrama de secuencia: Crear usuario

La "Figura 9.10" muestra la secuencia de creación de un usuario. La acción (*CreateUserAction*) solicita una nueva instancia a la factoría (*UserFactory*) y establece el valor de sus atributos con los métodos *set* correspondientes. Finalmente, solicita al servicio (*UserManager*) que la guarde (*persist*) y ejecute la transacción (*flush*) usando el *ObjectManager*, y que lance el evento (*CreatedUserEvent*) a través del *EventDispatcher*.

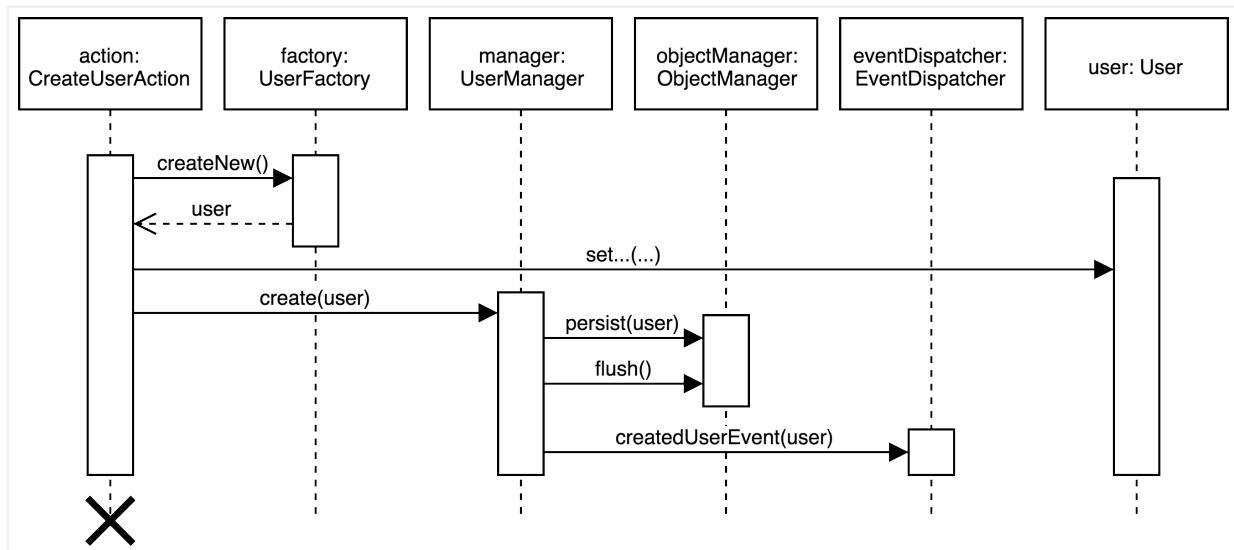


Figura 9.10 Diagrama de creación de usuarios

9.2.10 Diagrama de secuencia: Listar usuarios

La "Figura 9.11" muestra la secuencia para listar los usuarios. La acción (*ReadUsersAction*) solicita la colección de datos paginados al repositorio (*UserRepository*).

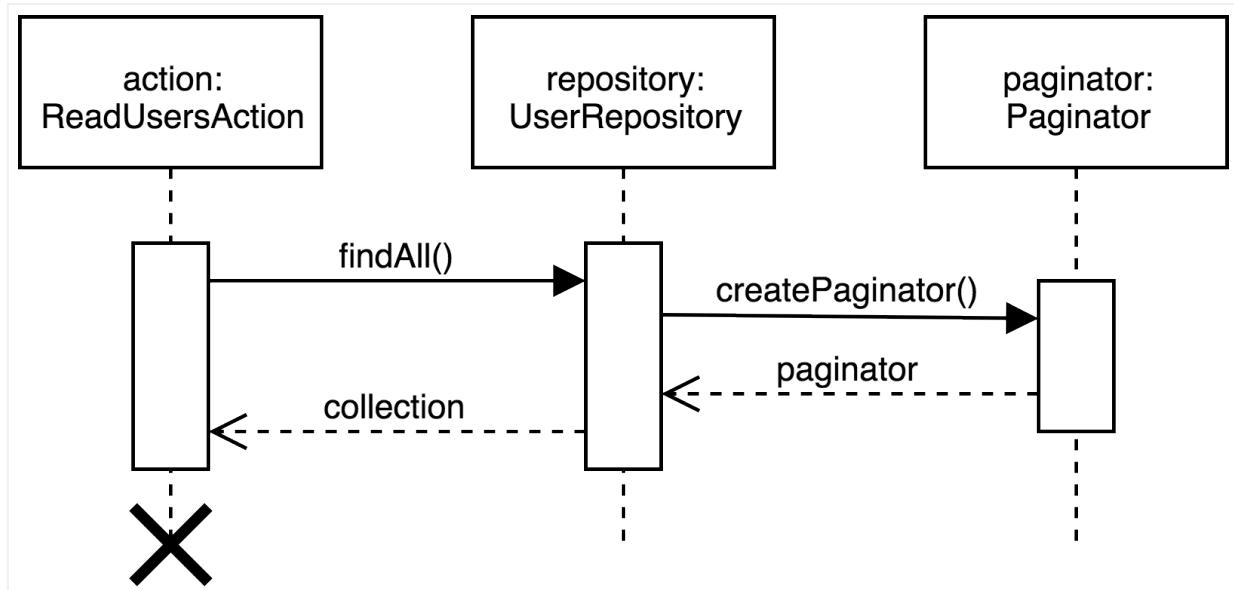


Figura 9.11 Diagrama de listado usuarios

9.2.11 Diagrama de secuencia: Actualizar usuario

La "Figura 9.12" muestra la secuencia de actualización de un usuario. La acción (*UpdateUserAction*) solicita la instancia al repositorio (*UserRepository*) y modifica el valor de sus atributos con los métodos *set* correspondientes. Finalmente, solicita al servicio (*UserManager*) que la guarde (*persist*) y ejecute la transacción (*flush*) usando el *ObjectManager*, y que lance el evento (*UpdatedUserEvent*) a través del *EventDispatcher*.

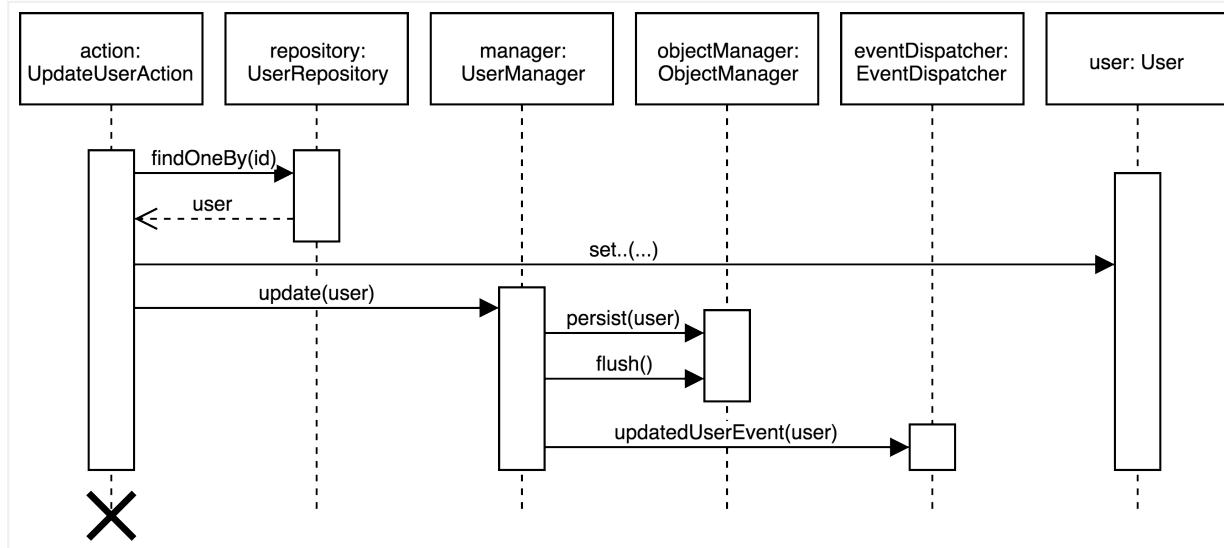


Figura 9.12 Diagrama de actualización de usuarios

9.2.12 Diagrama de secuencia: Eliminar usuario

La "Figura 9.13" muestra la secuencia de borrado de un usuario. La acción (*DeleteUserAction*) solicita la instancia al repositorio (*UserRepository*). Finalmente, solicita al servicio (*UserManager*) que la elimine (*delete*) y ejecute la transacción (*flush*) usando el *ObjectManager*, y que lance el evento (*DeletedUserEvent*) a través del *EventDispatcher*.

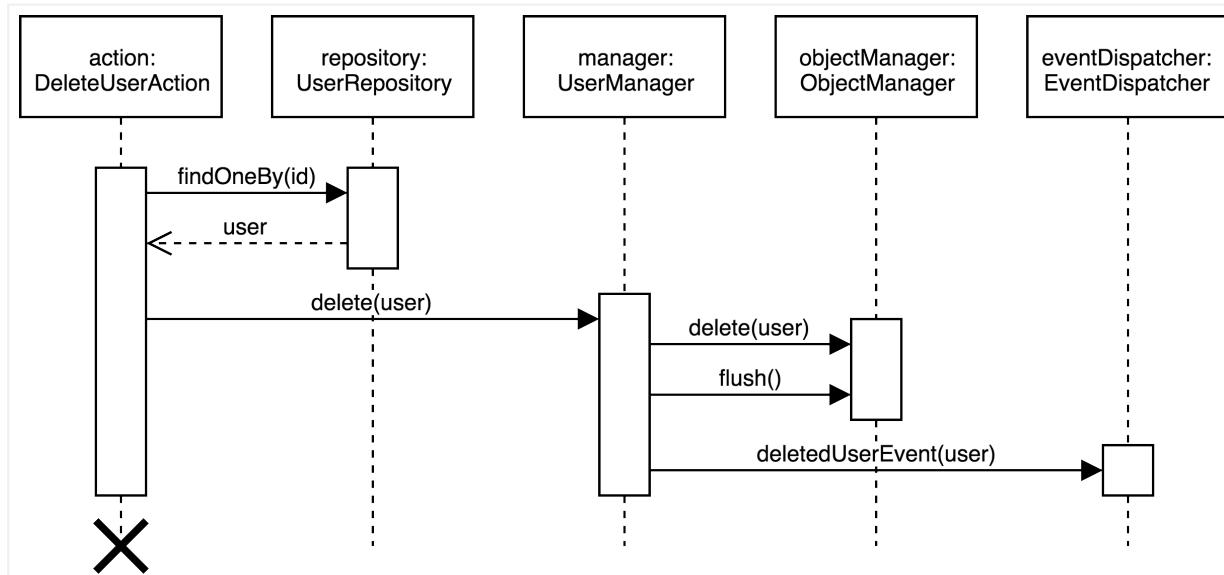


Figura 9.13 Diagrama de borrado de usuarios

9.2.13 Diagrama de secuencia: Crear registro

La "Figura 9.14" muestra la secuencia de creación de un registro. La acción (*CreateRegistrationAction*) solicita una nueva instancia a la factoría (*RegistrationFactory*) y establece el valor de sus atributos con los métodos *set* correspondientes. Finalmente, solicita al servicio (*RegistrationManager*) que la guarde (*persist*) y ejecute la transacción (*flush*) usando el *ObjectManager*, y que lance el evento (*CreatedRegistrationEvent*) a través del *EventDispatcher*.

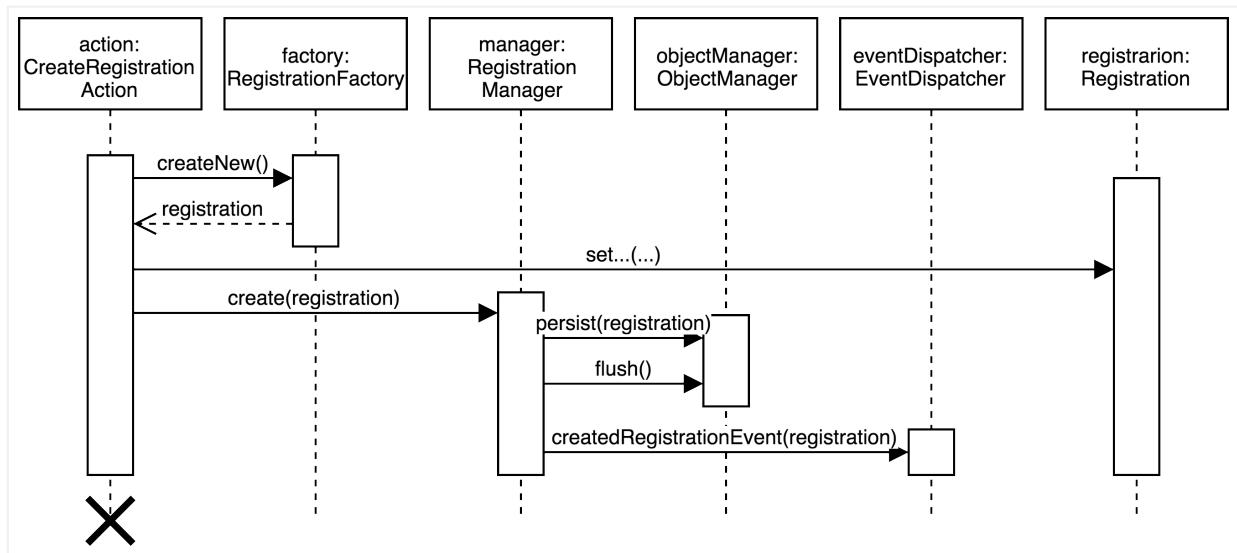


Figura 9.14 Diagrama de creación de registros

9.2.14 Diagrama de secuencia: Listar registros

La "Figura 9.15" muestra la secuencia para listar los registros. La acción (*ReadRegistrationsAction*) solicita la colección de datos paginados al repositorio (*RegistrationRepository*).

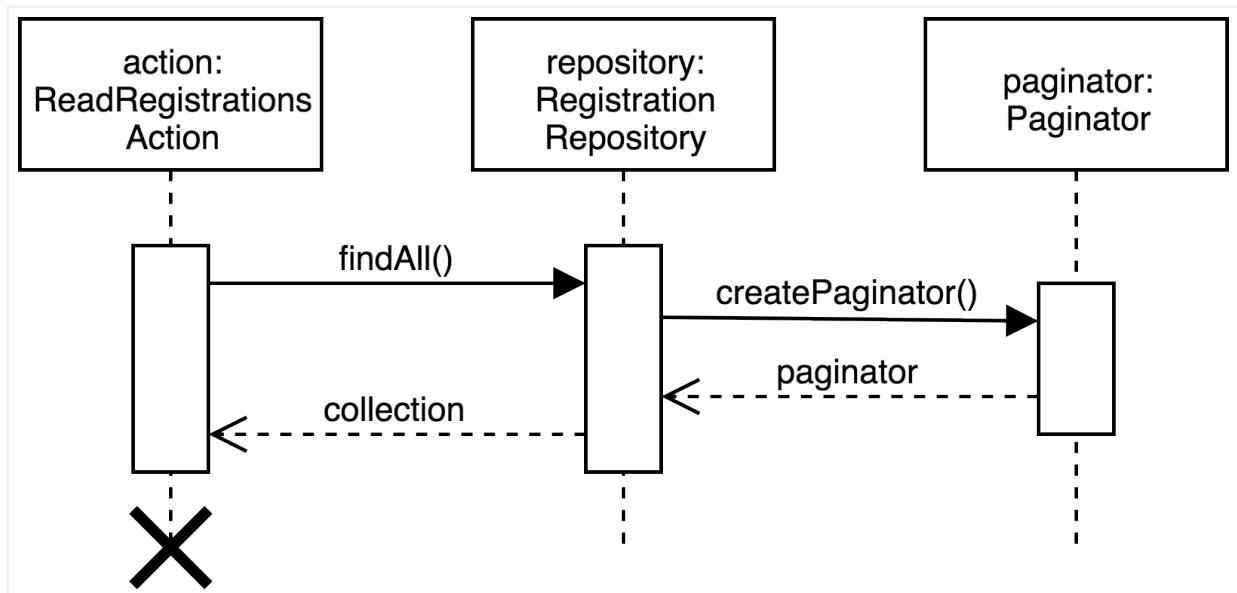


Figura 9.15 Diagrama de listado de registros

9.2.15 Diagrama de secuencia: Actualizar registro

La "Figura 9.16" muestra la secuencia de actualización de un registro. La acción (*UpdateRegistrationAction*) solicita la instancia al repositorio (*RegistrationRepository*) y modifica sus valores con los métodos *set*. Sigue la solicitud al servicio (*RegistrationManager*) que la guarde (*persist*) y ejecute la transacción (*flush*) usando el *ObjectManager*, y que lance el evento (*UpdatedRegistrationEvent*) a través del *EventDispatcher*.

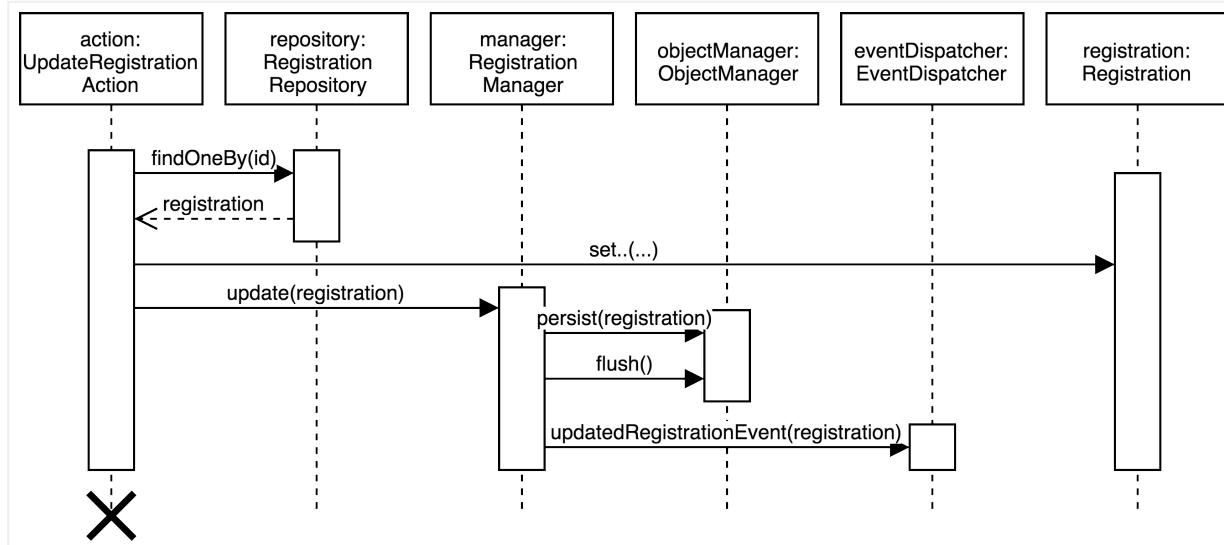


Figura 9.16 Diagrama de actualización de registros

9.2.16 Diagrama de secuencia: Eliminar registro

La "Figura 9.17" muestra la secuencia de borrado de un registro. La acción (*DeleteRegistrationAction*) solicita la instancia al repositorio (*RegistrationRepository*). El servicio (*RegistrationManager*) elimina (*delete*) y ejecuta la transacción (*flush*) usando el *ObjectManager*, y lanza el evento (*DeletedRegistrationEvent*) a través del *EventDispatcher*.

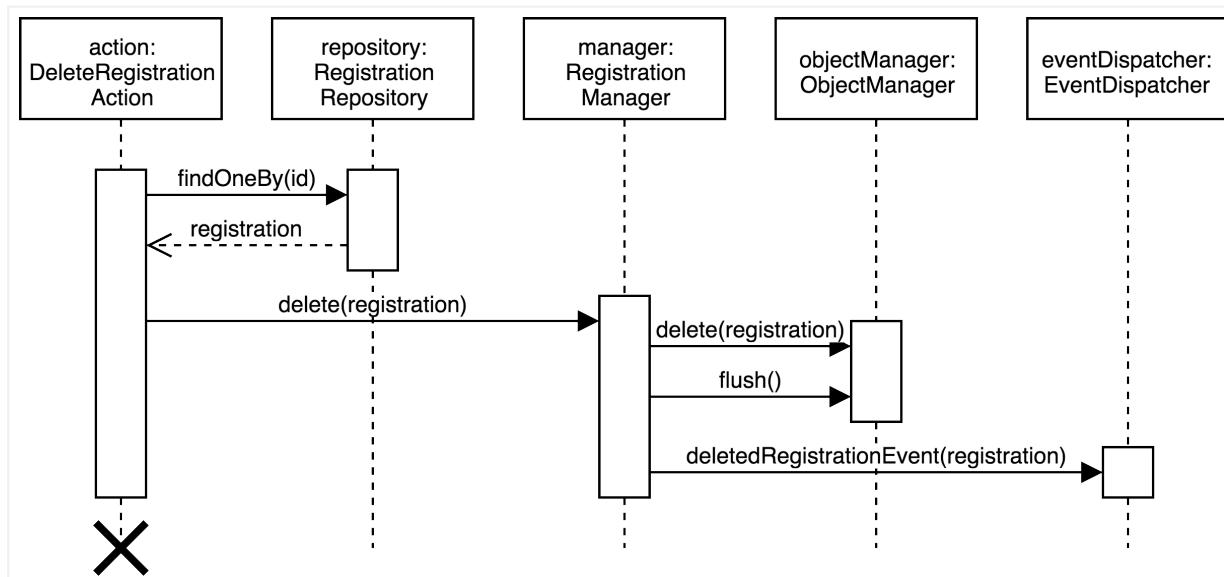


Figura 9.17 Diagrama de borrado de registros

9.2.17 Diagrama de secuencia: Crear notificación

La "Figura 9.18" muestra la secuencia de creación de una notificación. La acción (*CreateNotificationAction*) solicita una nueva instancia a la factoría (*NotificationFactory*) y establece el valor de sus atributos con los métodos *set* correspondientes. Finalmente, solicita al servicio (*NotificationManager*) que la guarde (*persist*) y ejecute la transacción (*flush*) usando el *ObjectManager*, y que lance el evento (*CreatedNotificationEvent*) a través del *EventDispatcher*.

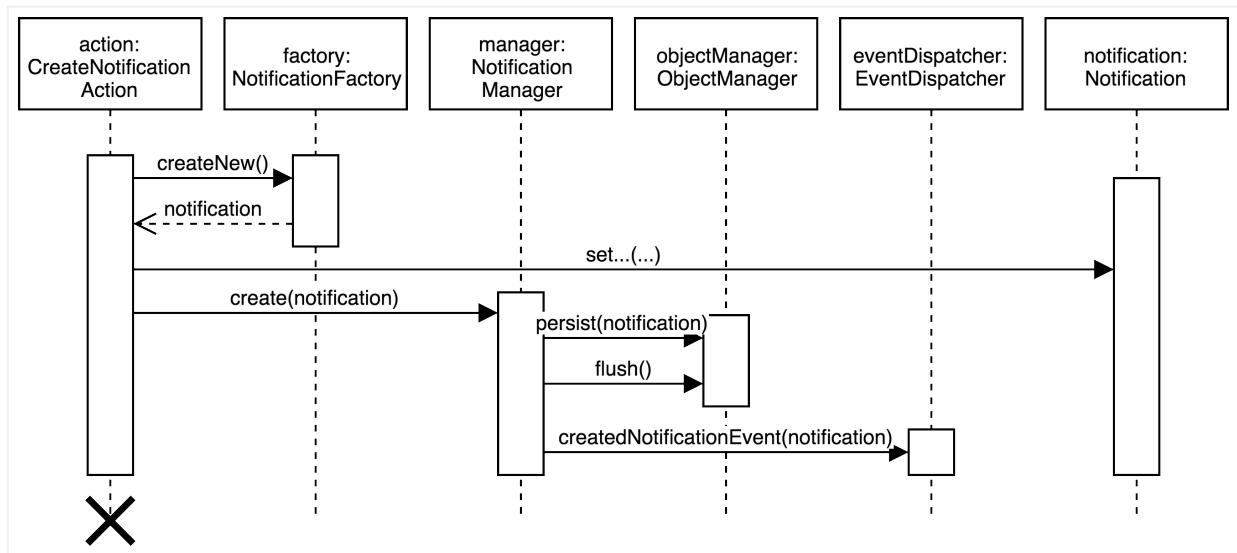


Figura 9.18 Diagrama de creación de notificaciones

9.2.18 Diagrama de secuencia: Listar notificaciones

La "Figura 9.19" muestra la secuencia para listar las notificaciones. La acción (*ReadNotificationsAction*) solicita la colección de datos paginados al repositorio (*NotificationRepository*).

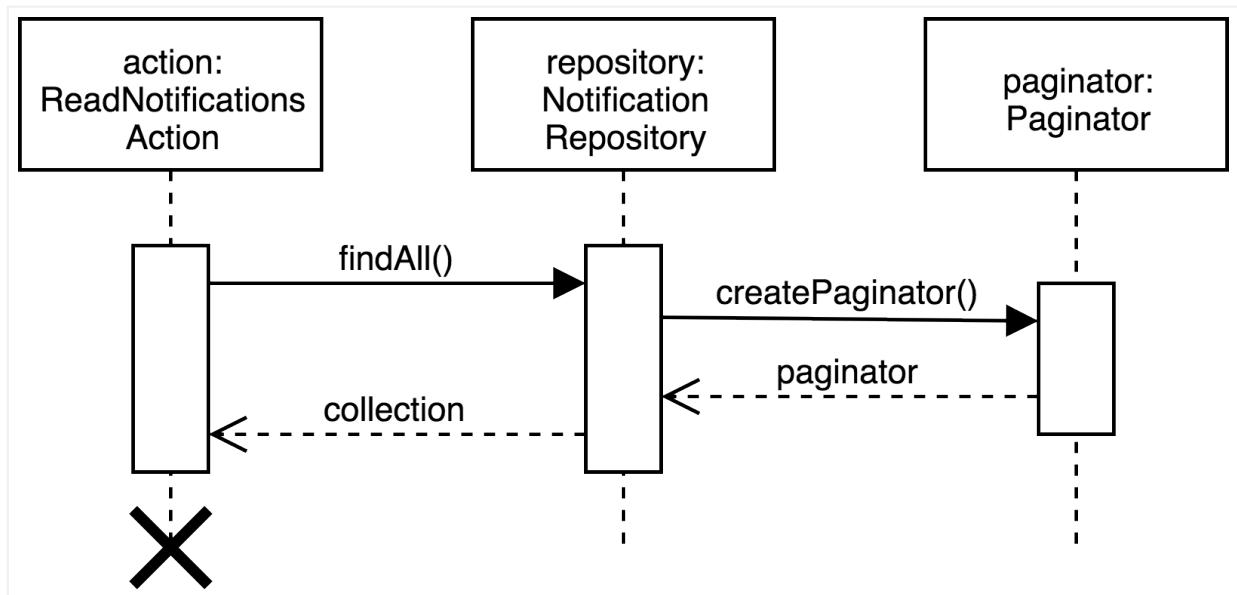


Figura 9.19 Diagrama de listado de notificaciones

9.2.19 Diagrama de secuencia: Actualizar notificación

La "Figura 9.20" muestra la secuencia de actualización de una notificación. La acción (*UpdateNotificationAction*) solicita la instancia al repositorio (*NotificationRepository*) y modifica sus valores con los métodos *set*. Sigue la solicitud al servicio (*NotificationManager*) que la guarde (*persist*) y ejecute la transacción (*flush*) usando el *ObjectManager*, y que lance el evento (*UpdatedNotificationEvent*) a través del *EventDispatcher*.

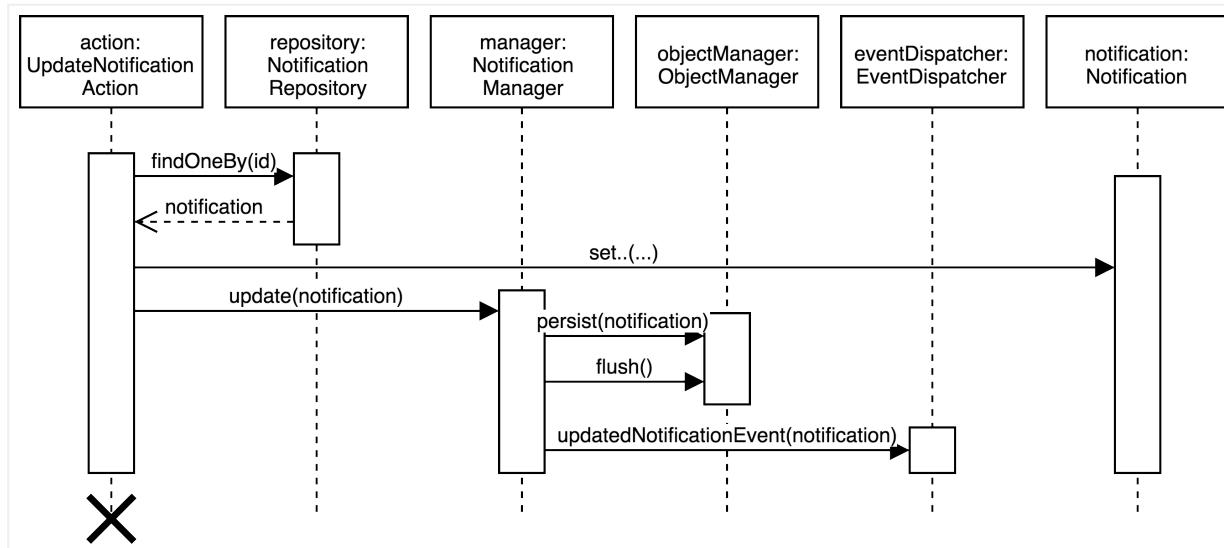


Figura 9.20 Diagrama de actualización de notificaciones

9.2.20 Diagrama de secuencia: Eliminar notificación

La "Figura 9.21" muestra la secuencia de borrado de una notificación. La acción (*DeleteNotificationAction*) solicita la instancia al repositorio (*NotificationRepository*). El servicio (*NotificationManager*) elimina (*delete*) y ejecuta la transacción (*flush*) con el *ObjectManager*, y lanza el evento (*DeletedNotificationEvent*) con el *EventDispatcher*.

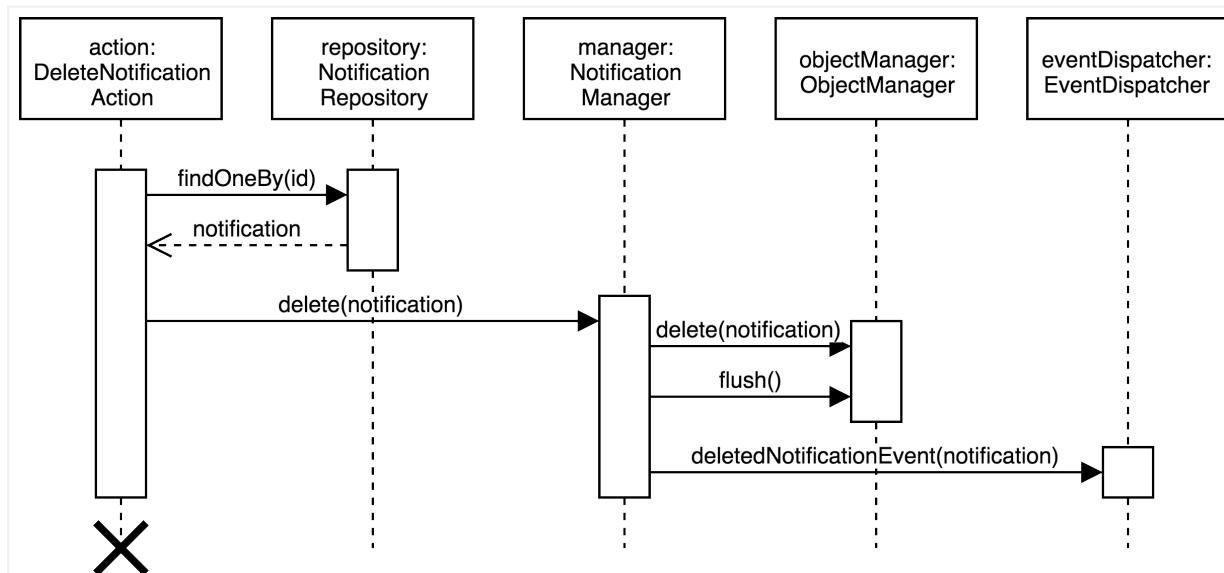


Figura 9.21 Diagrama de borrado de notificaciones

9.2.21 Diagrama de secuencia: Crear comunicación

La "Figura 9.22" muestra la secuencia de creación de una comunicación. La acción (*CreateCommunicationAction*) solicita una nueva instancia a la factoría (*CommunicationFactory*) y establece el valor de sus atributos con los métodos *set* correspondientes. Finalmente, solicita al servicio (*CommunicationManager*) que la guarde (*persist*) y ejecute la transacción (*flush*) usando el *ObjectManager*, y que lance el evento (*CreatedCommunicationEvent*) a través del *EventDispatcher*.

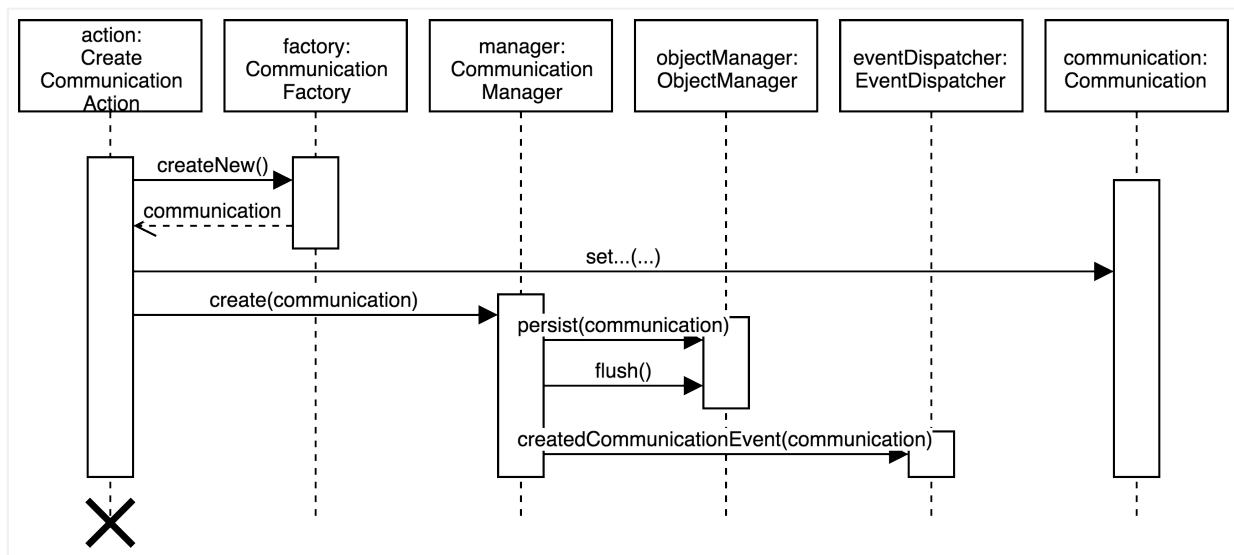


Figura 9.22 Diagrama de creación de comunicaciones

9.2.22 Diagrama de secuencia: Listar comunicaciones

La "Figura 9.23" muestra la secuencia para listar las comunicaciones. La acción (*ReadCommunicationsAction*) solicita la colección de datos paginados al repositorio (*CommunicationRepository*).

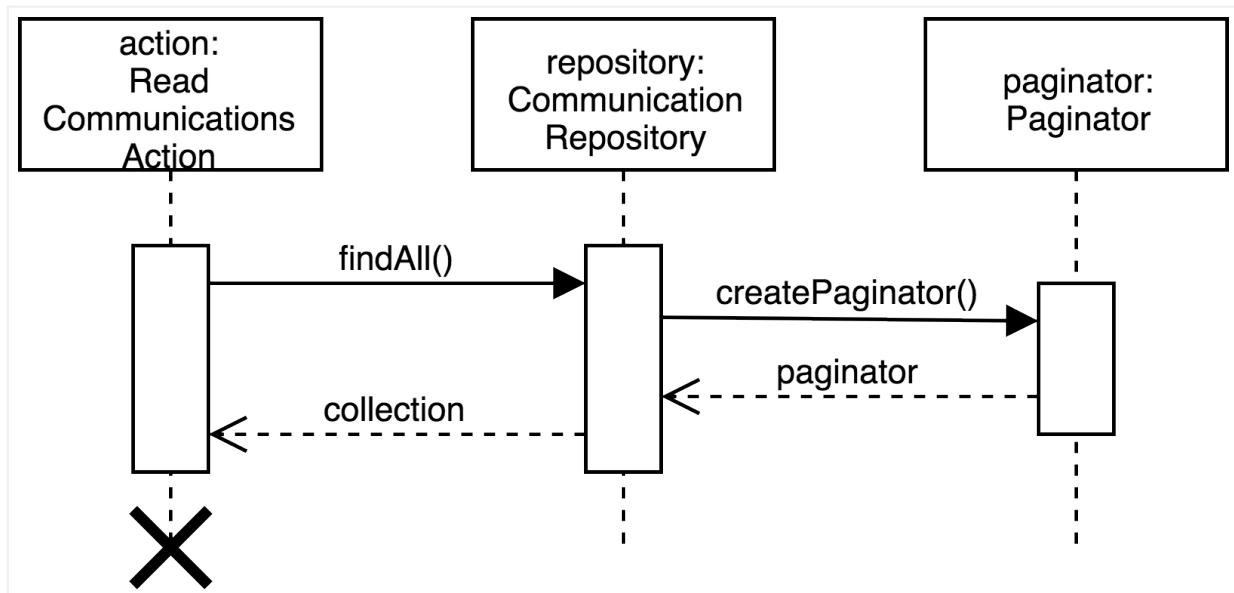


Figura 9.23 Diagrama de listado de comunicaciones

9.2.23 Diagrama de secuencia: Actualizar comunicación

La "Figura 9.24" muestra la secuencia de actualización de una comunicación. La acción (*UpdateCommunicationAction*) solicita la instancia al repositorio (*CommunicationRepository*) y modifica el valor de sus atributos con los métodos *set*. El servicio (*CommunicationManager*) la guarda (*persist*) y ejecuta la transacción (*flush*) usando el *ObjectManager*, y lanza el evento (*UpdatedCommunicationEvent*) con el *EventDispatcher*.

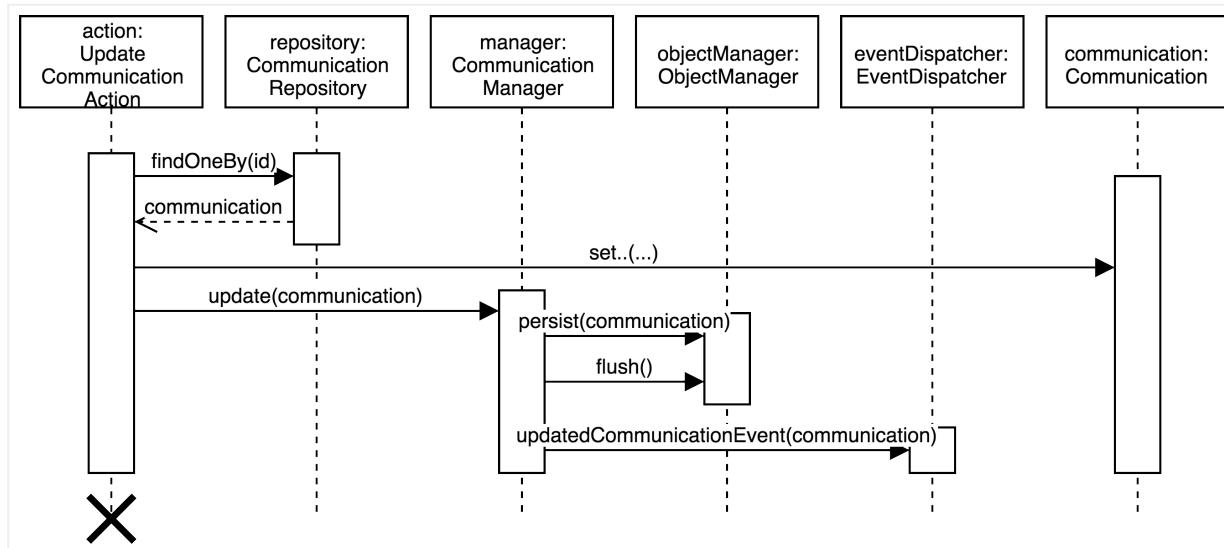


Figura 9.24 Diagrama de actualización de comunicaciones

9.2.24 Diagrama de secuencia: Eliminar comunicación

La "Figura 9.25" muestra la secuencia de borrado de una comunicación. La acción (*DeleteCommunicationAction*) pide la instancia al repositorio (*CommunicationRepository*). El servicio (*CommunicationManager*) elimina y ejecuta la transacción con el *ObjectManager*, y lanza el evento (*DeletedCommunicationEvent*) con el *EventDispatcher*.

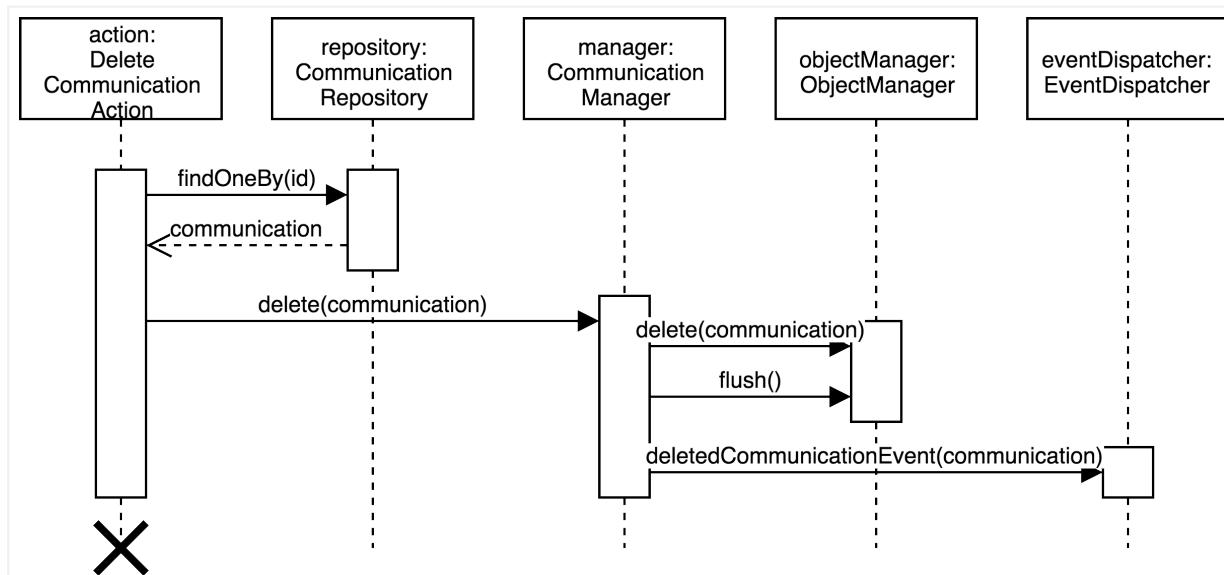


Figura 9.25 Diagrama de borrado de comunicaciones

9.2.25 Diagrama de secuencia: Autenticación

La "Figura 9.26" muestra la secuencia de autenticación de un usuario. La acción (*AuthAction*) solicita al servidor de *Google OAuth2* la autenticación del usuario. El usuario envía sus credenciales para iniciar sesión en *Google* y el servidor devuelve el código de autenticación. Después, solicita su información de usuario al servicio de *Google* a través del código, el servicio (*Google Userinfo Service*) devuelve la información de usuario y se solicita al repositorio (*UserRepository*) que busque un usuario con el correo electrónico recibido. Si el repositorio encuentra el usuario, lo devuelve, se actualiza con el servicio *UserManager* a través del *ObjectManager* y se lanza el evento *UpdatedUserEvent* usando el *EventDispatcher*. Si no se encuentra, solicita a la factoría (*UserFactory*) que cree uno nuevo con ese correo electrónico, se crea con el servicio *UserManager* a través del *ObjectManager* y se lanza el evento *CreatedUserEvent* usando el *EventDispatcher*. Finalmente, solicita al servicio de *JWT*^[3] (*JWTManager*) un nuevo token para el usuario y este es devuelto.

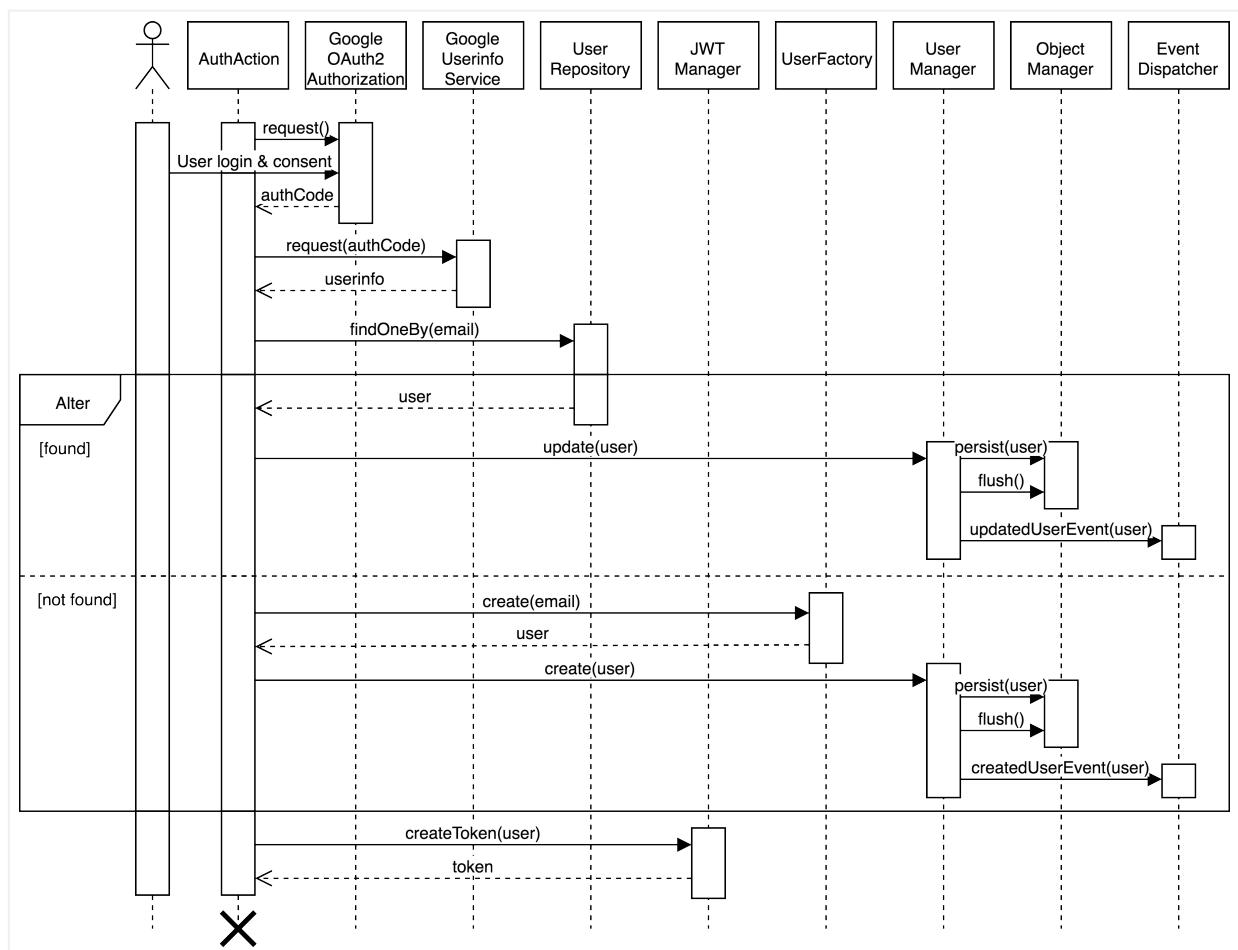


Figura 9.26 Diagrama de autenticación

9.2.26 Otros componentes del sistema

Además de todos los objetos ya expuestos, se han elaborado otros que también influyen en el comportamiento de la aplicación, pero no intervienen de forma secuencial en las se-

[3] Json Web Tokens: <https://jwt.io/>

cuencias descritas.

NotExpiredExtension consiste en una extensión del **ORM** para que, en las consultas cuyo recurso sea la clase *Actividad*, se filtren solo aquellas cuyo comienzo es posterior a la fecha y hora de la consulta.

ActivityVoter es un sistema de seguridad basado en votaciones que a través del método *isInvolved()* decide si un determinado usuario del sistema está involucrado en una actividad (creador o participante).

ActivityContextBuilder es un serializador que añade el grupo de serialización *involved* en una actividad si *ActivityVoter* determina que el usuario está involucrado.

Por último, se ha generado también un *EventSubscriber* por cada evento. Estos se encargan de crear las notificaciones y enviar los correos electrónicos correspondientes dependiendo del evento que ha sido lanzado.

Capítulo 10

Especificación de requisitos de la interfaz

El sistema presenta dos tipos de interfaz y ambas serán una aplicación web. Una, que denominaremos *backend*, es la parte de administración del sistema y disponible solo para usuarios autorizados. La otra, que denominaremos *frontend*, es la parte visible y pública de nuestra aplicación.

10.1 Características generales de la interfaz

La característica principal que debemos tener en cuenta es la usabilidad. Para ello tendremos en cuenta el diseño de las interfaces más usuales. No tiene sentido pretender ser innovador en algo que los usuarios ya están acostumbrados a usar habitualmente, las pantallas de inserción de datos.

Para poder realizar una interfaz usable debemos tener en cuenta los siguientes puntos:

1. Los nombres de los controles deben ser auto descriptivos. Para los términos más confusos se añadirá un mensaje de ayuda.
2. Los formularios tendrán un título que describa claramente cuál es su cometido.
3. Los formularios deben informar con claridad al usuario en caso de que se produzca un error en la inserción de datos, indicando cuál fue el motivo que causó el error y cómo pueden resolverlo.
4. Los formularios deben indicar qué campos son de inserción obligatoria.
5. Los controles o campos que, dependiendo del contexto, no son necesarios deben ser desactivados o, directamente, no deben aparecer.

10.2 Controles de la interfaz

Como indicamos hay dos interfaces, la que usarán los usuarios para gestionar eventos deportivos (*frontend*) y la que usará el administrador para gestionar la aplicación completa (*backend*). Además, ambas estarán integradas, por lo que las interfaces deben ajustarse al mismo formato.

10.2.1 Interfaz de usuario

Esta interfaz será usada por los usuarios de la aplicación para la gestión y organización de eventos deportivos.

Para el acceso a la vista principal de interfaz de usuario no será necesario identificarse en la aplicación, pero sí que será necesario para acceder al resto de vistas. De la interfaz para la identificación del usuario se hará cargo *Google* por lo que no decidiremos el diseño de esta. Si la identificación no es válida la interfaz mostrará el error y la identificación del usuario no será posible. Una vez identificado, el usuario tendrá acceso a un botón situado en la parte superior derecha para cerrar sesión en cualquier momento.

Como se ha mencionado anteriormente, para el acceso a la vista principal no será necesaria la identificación del usuario y este podrá ver una lista de todas las actividades de la aplicación, como muestra la "Figura 10.1".

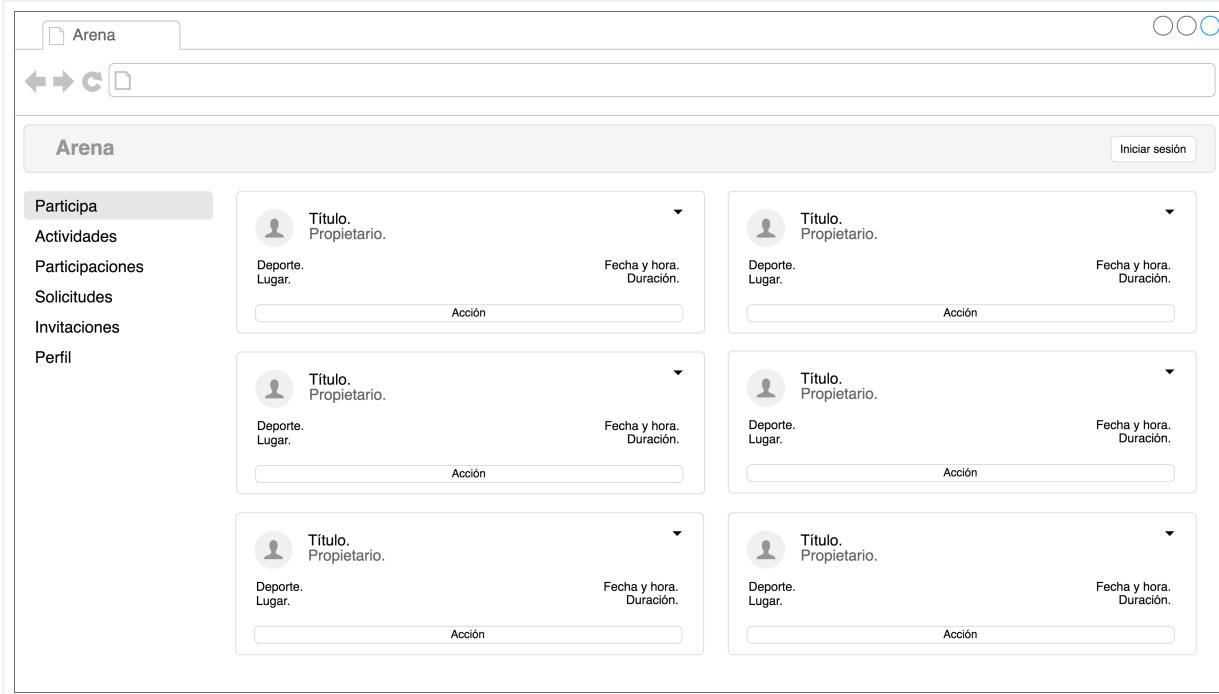


Figura 10.1 Wireframe principal

Esta se verá afectada en función de si el usuario está identificado o no. En el caso en que el usuario ha sido identificado se mostrarán las acciones disponibles. En el otro caso, se mostrará un mensaje para que el usuario inicie sesión si quiere realizar alguna acción.

Como se puede apreciar en la "Figura 10.2", un usuario identificado podrá ver la lista de sus actividades para gestionarlas, editarlas o eliminarlas. Además, podrá crear nuevas.

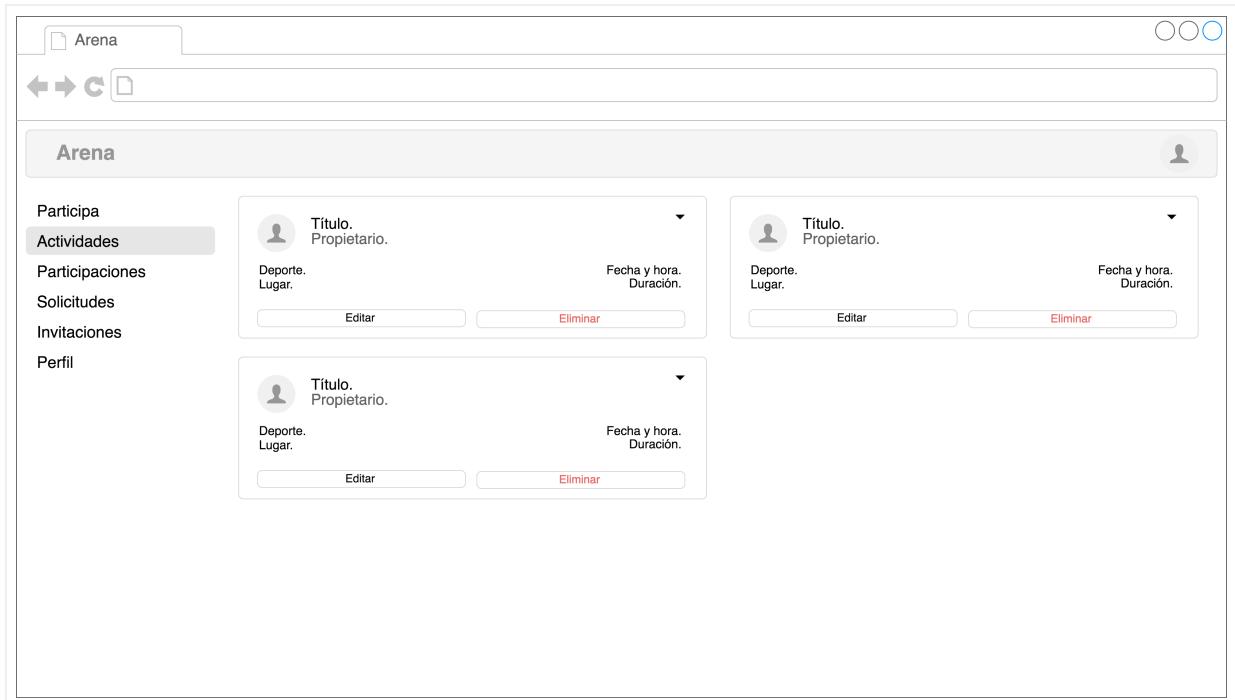


Figura 10.2 Wireframe de listado de actividades

Un usuario identificado podrá ver la lista de todas las actividades donde es participante y eliminar su participación, como se aprecia en la "Figura 10.3".

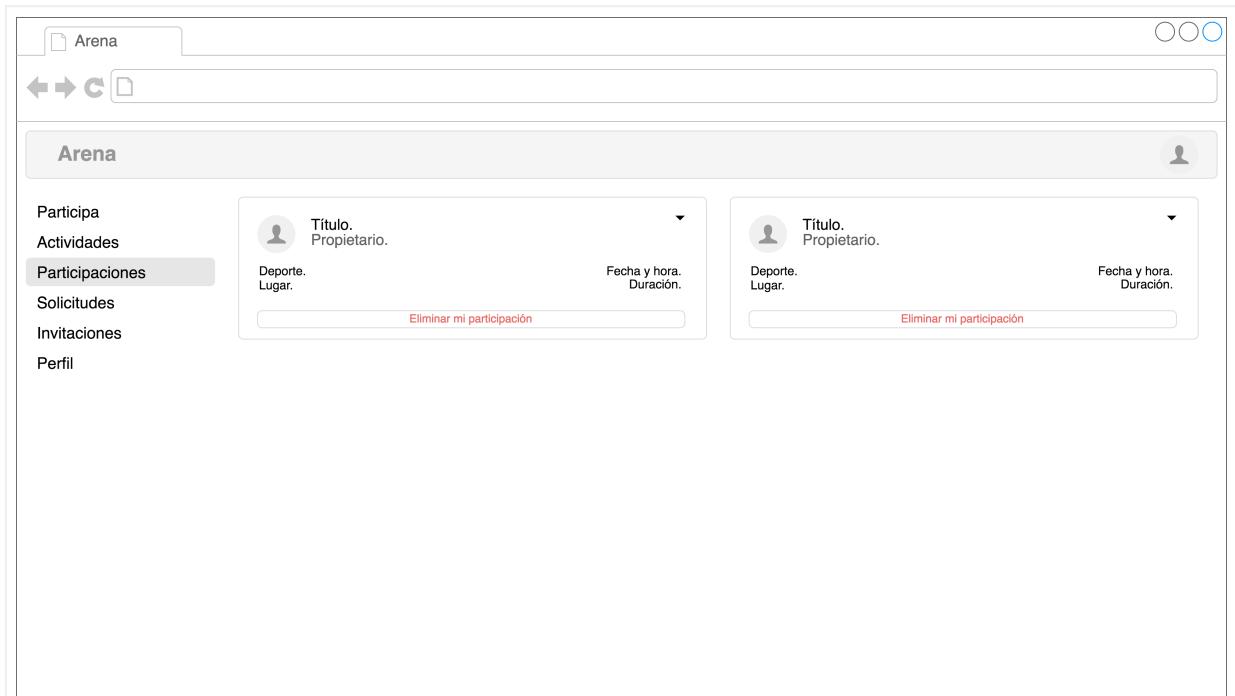


Figura 10.3 Wireframe de listado de participaciones

Un usuario identificado podrá ver la lista de todas sus solicitudes en actividades y cancelar su solicitud, como se aprecia en la "Figura 10.4".

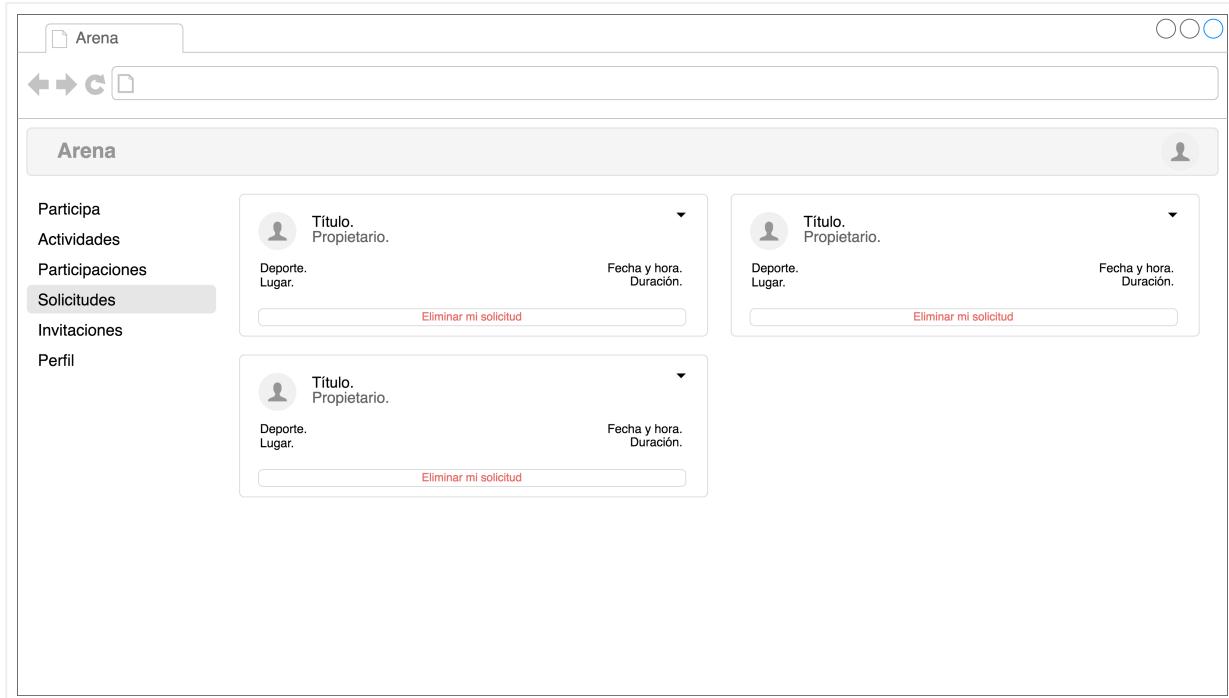


Figura 10.4 Wireframe de listado de solicitudes

Un usuario identificado podrá ver la lista de todas sus invitaciones de participación en actividades para aceptarlas o rechazarlas, como se aprecia en la "Figura 10.5".

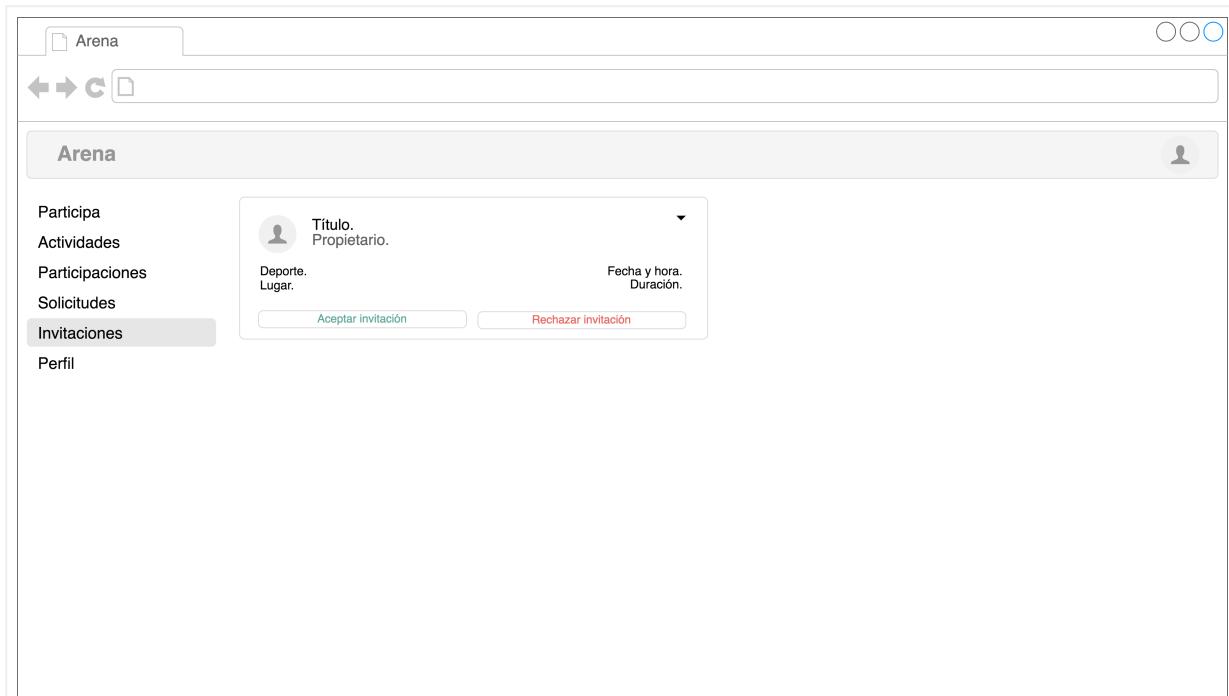


Figura 10.5 Wireframe de listado de invitaciones

Un usuario identificado podrá ver sus datos de perfil para editarlos o eliminar su cuenta, como se aprecia en la "Figura 10.6".

Este wireframe muestra la interfaz de usuario para el perfil de un usuario. En la parte superior, hay un menú horizontal con iconos para 'Arena' (que está resaltado), 'C' (que es un botón para cambiar la vista), y tres óvalos azules. Abajo de eso, hay un cuadro de diálogo titulado 'Arena' que contiene un icono de persona y el nombre del usuario. A la izquierda de este cuadro, hay una lista vertical con enlaces: 'Participa', 'Actividades', 'Participaciones', 'Solicitudes', 'Invitaciones' y 'Perfil', donde 'Perfil' está resaltado. A la derecha del cuadro principal, hay tres campos de texto para 'Correo electrónico', 'Nombre' y 'Apellidos'. En la parte inferior del cuadro, hay dos botones: 'Guardar cambios' y 'Eliminar cuenta'.

Figura 10.6 Wireframe de perfil de usuario

Un usuario identificado podrá nuevas actividades a través de un formulario, como se aprecia en la "Figura 10.7".

Este wireframe muestra la interfaz de usuario para crear una nueva actividad. La estructura es similar al wireframe de perfil, con un menú superior y un cuadro de diálogo 'Arena' en el centro. Dentro del cuadro, se requieren campos para 'Título' y 'Deporte'. Debajo de estos, hay un grupo de campos para 'Fecha y hora de inicio' (con un icono de calendario), 'Duración' y 'Plazas'. A la derecha de estos campos, hay un campo para 'Lugar'. Abajo de los campos de fecha y hora, hay un campo para 'Descripción'. En la parte inferior del cuadro, hay un botón 'Crear actividad'.

Figura 10.7 Wireframe de creación de actividades

Un usuario identificado podrá modificar los datos de sus actividades a través de un formulario, como se aprecia en la "Figura 10.8".

Este wireframe muestra una interfaz de usuario para la modificación de actividades. En la parte superior, hay un menú con iconos para 'Arena' y otras opciones. Abajo de eso, un botón para 'Arena'. La sección principal titulada 'Arena' contiene campos para 'Título' (con placeholder 'Título'), 'Deporte' (con placeholder 'Deporte'), y 'Duración', 'Plazas', 'Lugar' (cada uno con placeholder correspondiente). Hay un campo 'Descripción' con placeholder 'Descripción'. A continuación, un botón 'Guardar cambios'.

Figura 10.8 Wireframe de modificación de actividades

Un usuario podrá ver los datos de una actividad de forma detallada para realizar una acción sobre ella, como se aprecia en la "Figura 10.9".

Este wireframe muestra una interfaz de usuario para ver los datos de una actividad detallada. En la parte superior, hay un menú con iconos para 'Arena' y otras opciones. Abajo de eso, un botón para 'Arena'. La sección principal titulada 'Arena' muestra información detallada: 'Título. Propietario.', 'Deporte. Lugar.', 'Descripción.', 'Plazas disponibles.', 'Fecha y hora. Duración.', y un campo 'Acción'. Debajo de esto, sección 'Participaciones' con un campo para 'Nombre completo' y un botón 'x'. Sección 'Solicitudes' y 'Invitaciones', ambas con campos para 'Nombre completo' y botones '✓' y 'x'. Hay un botón 'Añadir invitaciones'.

Figura 10.9 Wireframe de actividad detallada

Esta vista se verá afectada en función de si el usuario está involucrado o no. Los usuarios involucrados verán la sección de participantes. El propietario de la actividad, además, verá las secciones de solicitudes e invitaciones y podrá gestionar las participaciones, solicitudes e invitaciones.

10.2.2 Interfaz del administrador

Esta interfaz será usada por el administrador de la aplicación para la gestión de los datos de la aplicación y la administración de los usuarios.

Para el acceso a la interfaz de administrador es necesario identificarse como usuario administrador de la aplicación, por lo que la primera interfaz que verá el usuario será la de autenticación. Si la identificación no es válida la interfaz de la "Figura 10.10" mostrará el error y no se permitirá el acceso a la interfaz de administración.

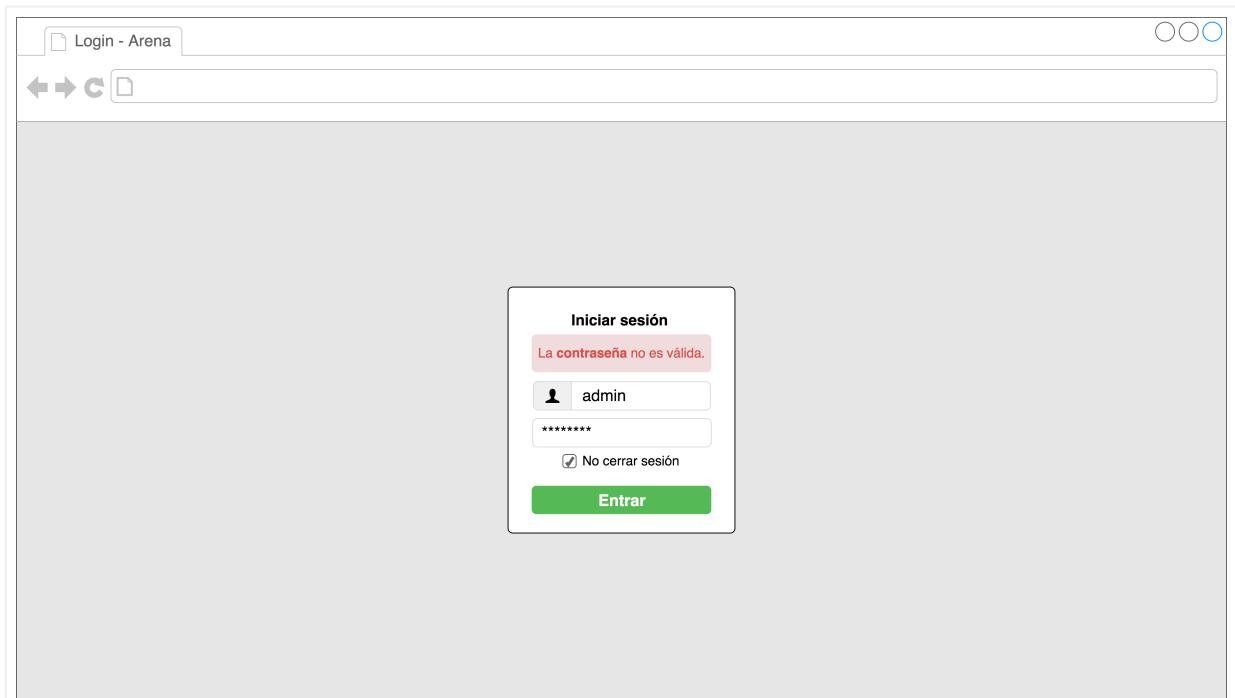
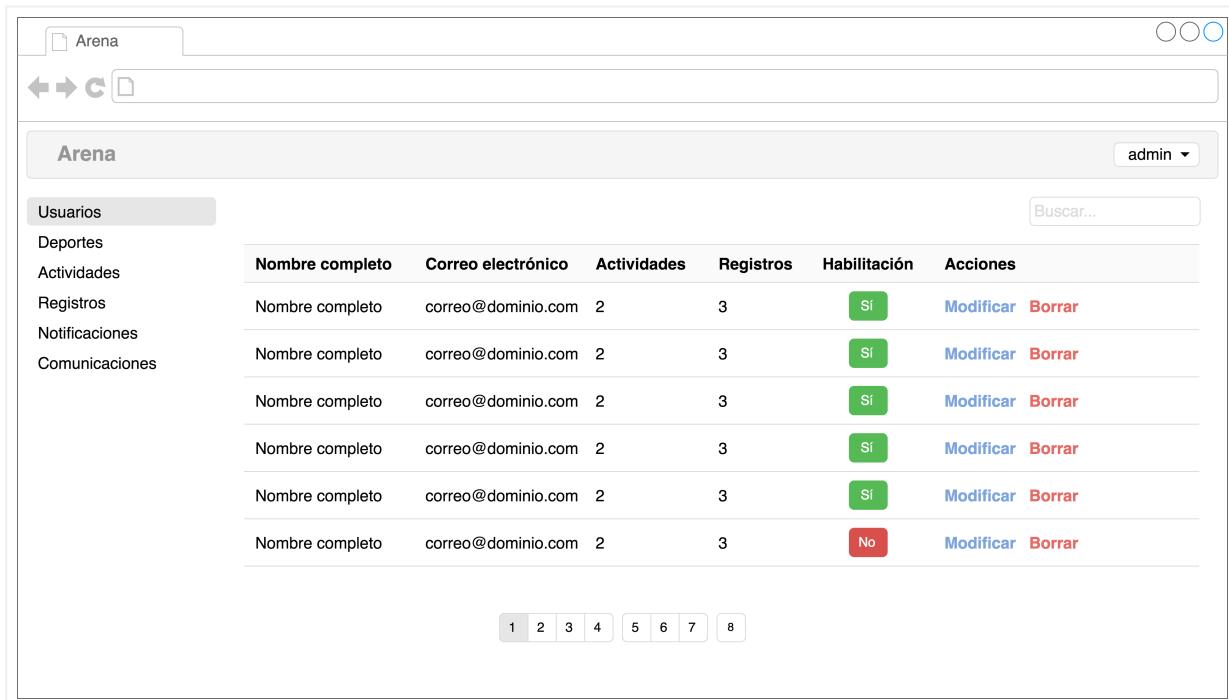


Figura 10.10 Wireframe de autenticación

En cualquier momento, el administrador tendrá acceso a un botón situado en la parte superior derecha para cerrar sesión, y si lo hace, volverá a la página principal de la aplicación.

El usuario administrador podrá seleccionar en el menú de la "Figura 10.11" la opción para administrar usuarios. De esta forma, se mostrará la lista de todos los usuarios.

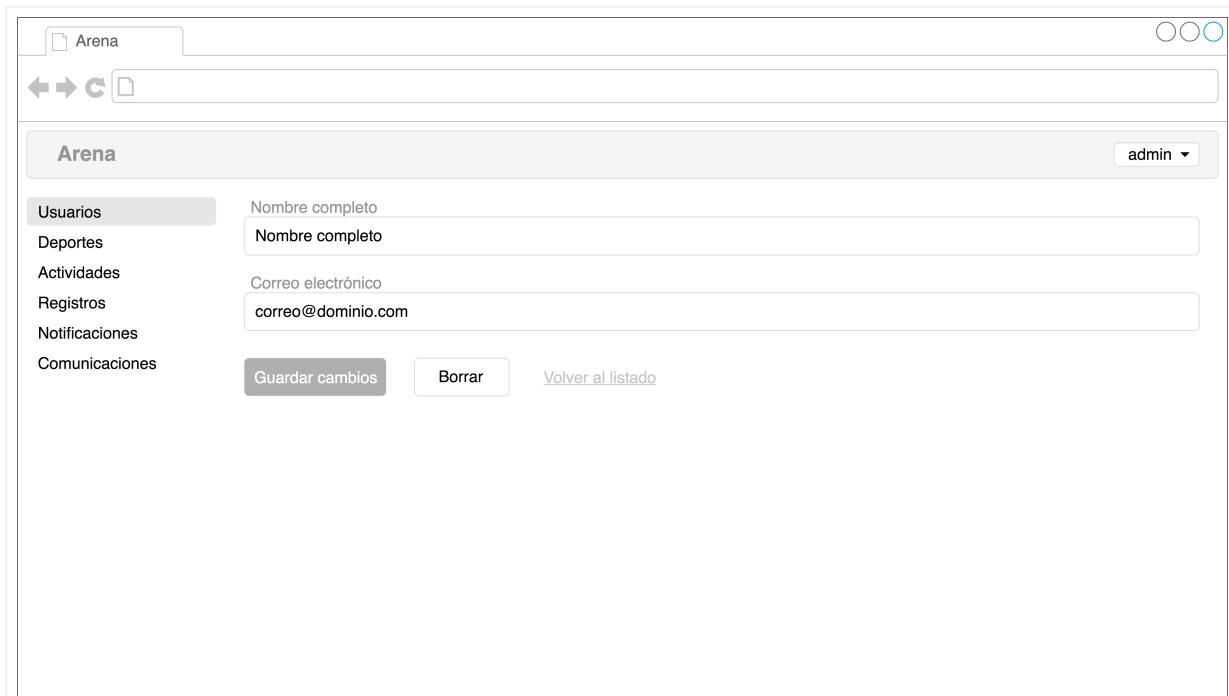


Este wireframe muestra una lista de usuarios en una interfaz web. La barra superior incluye un logo "Arena", iconos para retroceder, avanzar, recargar y cerrar, y un menú suscrito a "admin". El panel lateral izquierdo tiene enlaces para "Usuarios", "Deportes", "Actividades", "Registros", "Notificaciones" y "Comunicaciones". El panel central es una tabla con las siguientes columnas: Nombre completo, Correo electrónico, Actividades, Registros, Habilitación y Acciones. Los datos muestran seis registros de usuarios con nombres completos generados aleatoriamente y correos en dominio.com. Cada fila incluye botones "Modificar" y "Borrar" al lado de un cuadro que indica si el usuario está habilitado ("Sí" o "No"). Una barra de navegación en la parte inferior muestra páginas 1 a 8.

	Nombre completo	Correo electrónico	Actividades	Registros	Habilitación	Acciones
	Nombre completo	correo@dominio.com	2	3	Si	Modificar Borrar
	Nombre completo	correo@dominio.com	2	3	Si	Modificar Borrar
	Nombre completo	correo@dominio.com	2	3	Si	Modificar Borrar
	Nombre completo	correo@dominio.com	2	3	Si	Modificar Borrar
	Nombre completo	correo@dominio.com	2	3	Si	Modificar Borrar
	Nombre completo	correo@dominio.com	2	3	No	Modificar Borrar

Figura 10.11 Wireframe de listado de usuarios

El usuario administrador podrá seleccionar la opción para editar un determinado usuario como en la "Figura 10.12".



Este wireframe muestra un formulario para editar un usuario. La barra superior y el menú lateral izquierdo son los mismos que en la Figura 10.11. El panel central contiene campos para "Nombre completo" (con valor "Nombre completo") y "Correo electrónico" (con valor "correo@dominio.com"). Abajo del formulario hay botones "Guardar cambios", "Borrar" y "Volver al listado".

Figura 10.12 Wireframe de modificación de usuarios

El usuario administrador podrá seleccionar la opción para eliminar un determinado usuario como en la "Figura 10.13".

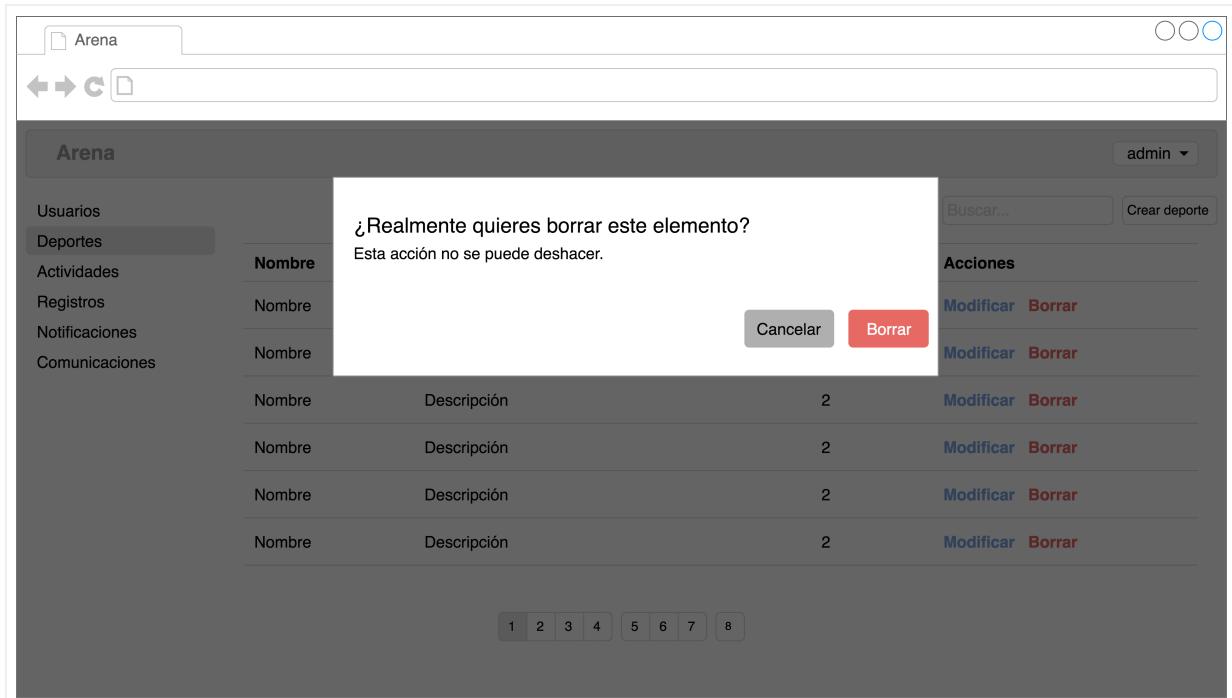


Figura 10.13 Wireframe de borrado de usuarios

El usuario administrador podrá seleccionar la opción para crear un nuevo deporte como en la "Figura 10.14".

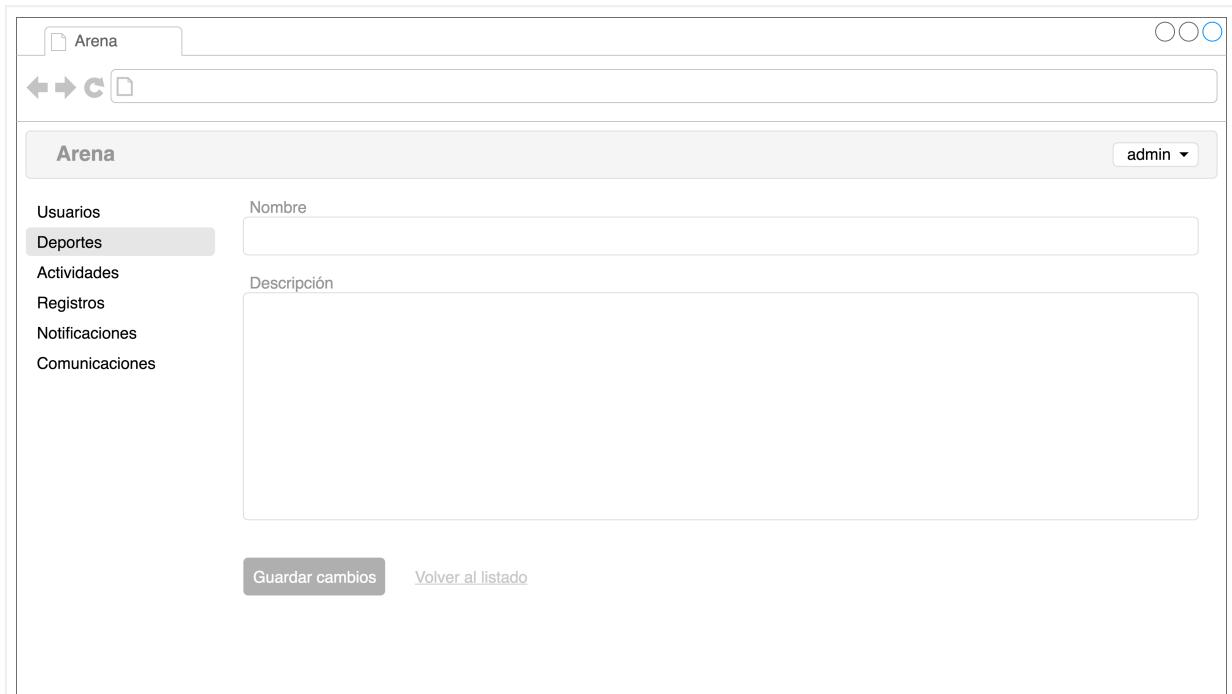


Figura 10.14 Wireframe de creación de deportes

El usuario administrador podrá seleccionar en el menú la opción para administrar deportes como en la "Figura 10.15".

The wireframe shows a user interface for managing sports. At the top left is a navigation bar with icons for back, forward, search, and refresh. On the right are three circular status indicators. The main header is 'Arena' with a dropdown menu showing 'admin'. A sidebar on the left lists 'Users', 'Sports' (which is selected), 'Activities', 'Registers', 'Notifications', and 'Communications'. The main content area is titled 'Arena' and contains a table for listing sports. The table has columns: 'Nombre' (Name), 'Descripción' (Description), 'Actividades' (Activities), and 'Acciones' (Actions). There are six rows of data, each with a 'Modificar' (Edit) and 'Borrar' (Delete) button. Below the table is a navigation bar with pages 1 through 8.

Nombre	Descripción	Actividades	Acciones
Nombre	Descripción	2	Modificar Borrar
Nombre	Descripción	2	Modificar Borrar
Nombre	Descripción	2	Modificar Borrar
Nombre	Descripción	2	Modificar Borrar
Nombre	Descripción	2	Modificar Borrar

Figura 10.15 Wireframe de listado de deportes

El usuario administrador podrá seleccionar la opción para editar un determinado deporte como en la "Figura 10.16".

The wireframe shows a user interface for editing a sport. It has a similar top navigation and sidebar to Figura 10.15. The main content area is titled 'Arena' and contains two input fields: 'Nombre' (Name) and 'Descripción' (Description). At the bottom are three buttons: 'Guardar cambios' (Save changes) in a grey box, 'Borrar' (Delete), and 'Volver al listado' (Return to list).

Figura 10.16 Wireframe de modificación de deportes

El usuario administrador podrá seleccionar la opción para eliminar un determinado deporte como en la "Figura 10.17".

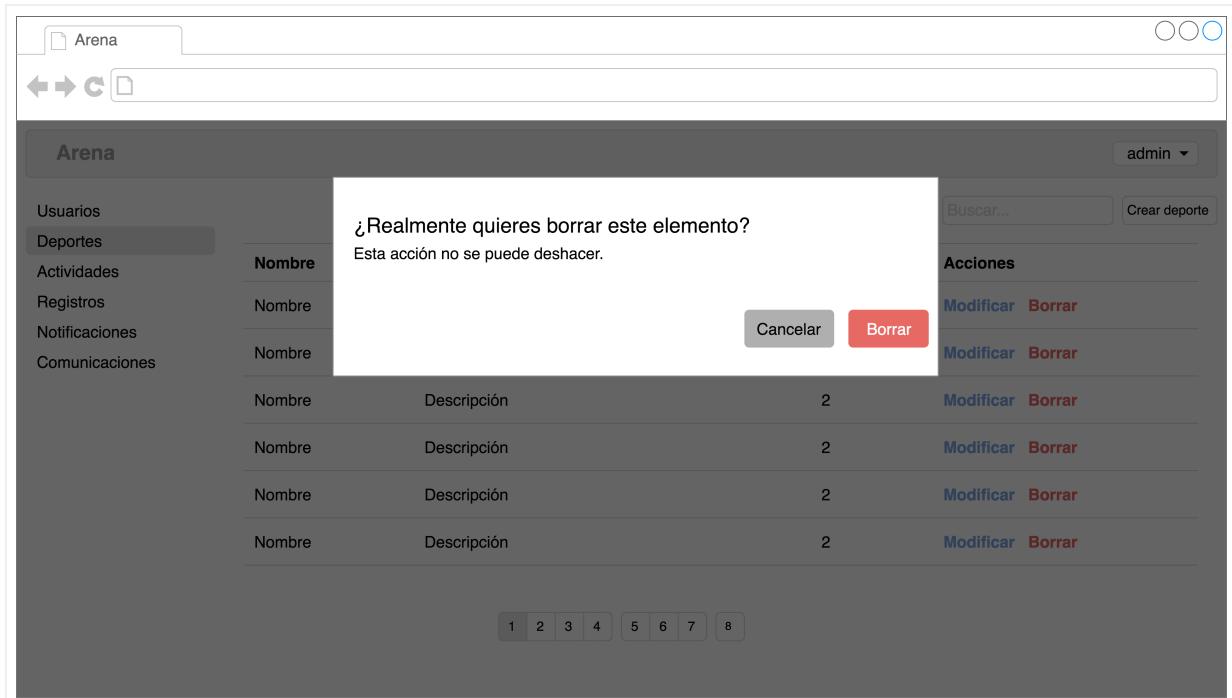


Figura 10.17 Wireframe de borrado de deportes

El usuario administrador podrá seleccionar la opción para crear una nueva actividad como en la "Figura 10.18".

 A wireframe of a web application interface titled 'Arena'. On the left, a sidebar menu includes 'Usuarios', 'Deportes', 'Actividades' (which is selected), 'Registros', 'Notificaciones', and 'Comunicaciones'. The main content area contains several input fields: 'Título' (Title) with a placeholder 'Deporte', 'Deporte' (Sport) with a placeholder 'Fútbol', 'Creador' (Creator) with a placeholder 'Administrador', 'Fecha y hora de comienzo' (Start Date and Time) with a date picker icon, 'Duración' (Duration), 'Plazas' (Places), and 'Lugar' (Place). Below these fields is a large 'Descripción' (Description) text area. At the bottom are two buttons: 'Guardar cambios' (Save changes) and 'Volver al listado' (Return to list).

Figura 10.18 Wireframe de creación de actividades

El usuario administrador podrá seleccionar en el menú la opción para administrar actividades como en la "Figura 10.19".

Este wireframe muestra una lista de actividades en una interfaz web. La barra lateral izquierda tiene opciones: Usuarios, Deportes, Actividades (selecciónada), Registros, Notificaciones y Comunicaciones. La barra superior tiene iconos para Área, Navegación y Usuario (admin). El encabezado de la tabla es: Título, Deporte, Creador, Comienzo, Duración, Plazas, Registros, Acciones. Los datos de la tabla son:

Título	Deporte	Creador	Comienzo	Duración	Plazas	Registros	Acciones
Título	Deporte	Creador	dd/mm/aaaa hh:mm	27	5	3	Modificar Borrar
Título	Deporte	Creador	dd/mm/aaaa hh:mm	27	5	3	Modificar Borrar
Título	Deporte	Creador	dd/mm/aaaa hh:mm	27	5	3	Modificar Borrar
Título	Deporte	Creador	dd/mm/aaaa hh:mm	27	5	3	Modificar Borrar
Título	Deporte	Creador	dd/mm/aaaa hh:mm	27	5	3	Modificar Borrar
Título	Deporte	Creador	dd/mm/aaaa hh:mm	27	5	3	Modificar Borrar

En la parte inferior hay una barra de navegación con números de página (1, 2, 3, 4, 5, 6, 7, 8).

Figura 10.19 Wireframe de listado de actividades

El usuario administrador podrá seleccionar la opción para editar una determinada actividad como en la "Figura 10.20".

Este wireframe muestra un formulario para editar una actividad. La barra lateral izquierda tiene las mismas opciones que en la Figura 10.19. Los campos para llenar son: Título, Deporte, Creador, Comienzo (con calendario), Duración (60), Plazas (10), Lugar y Descripción. A continuación hay botones para Guardar cambios, Borrar y Volver al listado.

Figura 10.20 Wireframe de modificación de actividades

El usuario administrador podrá seleccionar la opción para eliminar una determinada actividad como en la "Figura 10.21".

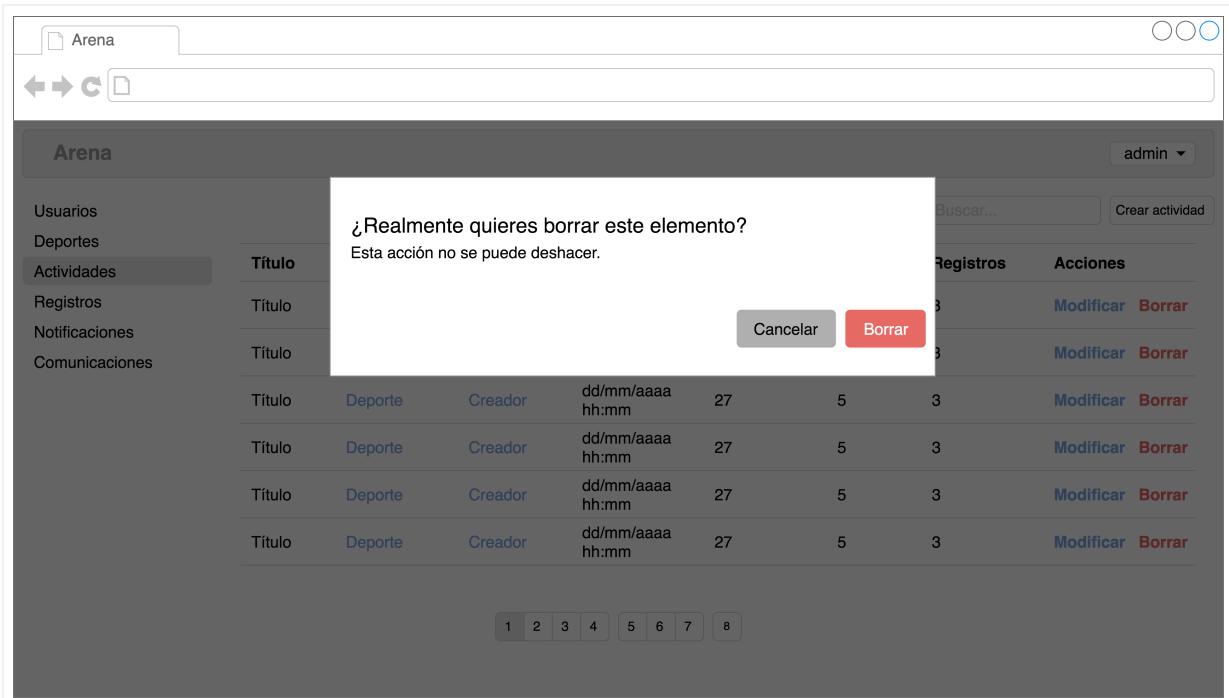


Figura 10.21 Wireframe de borrado de actividades

El usuario administrador podrá seleccionar la opción para crear un nuevo registro como en la "Figura 10.23".

 A wireframe of a web application interface titled 'Arena'. On the left, a sidebar menu includes 'Usuarios', 'Deportes', 'Actividades', 'Registros' (which is selected), 'Notificaciones', and 'Comunicaciones'. The main content area contains four input fields: 'Usuario' (with placeholder 'User'), 'Actividad' (with placeholder 'Activity'), 'Tipo' (with placeholder 'Type'), and 'Estado' (with placeholder 'State'). Below these fields are two buttons: 'Guardar cambios' (Save changes) and 'Volver al listado' (Return to list).

Figura 10.22 Wireframe de creación de registros

El usuario administrador podrá seleccionar en el menú la opción para administrar registros como en la "Figura 10.24".

Usuario	Actividad	Tipo	Estado	Acciones
Usuario	Actividad	Tipo	Estado	Modificar Borrar
Usuario	Actividad	Tipo	Estado	Modificar Borrar
Usuario	Actividad	Tipo	Estado	Modificar Borrar
Usuario	Actividad	Tipo	Estado	Modificar Borrar
Usuario	Actividad	Tipo	Estado	Modificar Borrar
Usuario	Actividad	Tipo	Estado	Modificar Borrar
Usuario	Actividad	Tipo	Estado	Modificar Borrar

Figura 10.23 Wireframe de listado de registros

El usuario administrador podrá seleccionar la opción para editar un determinado registro como en la "Figura 10.25".

Figura 10.24 Wireframe de modificación de registros

El usuario administrador podrá seleccionar la opción para eliminar un determinado registro como en la "Figura 10.26".

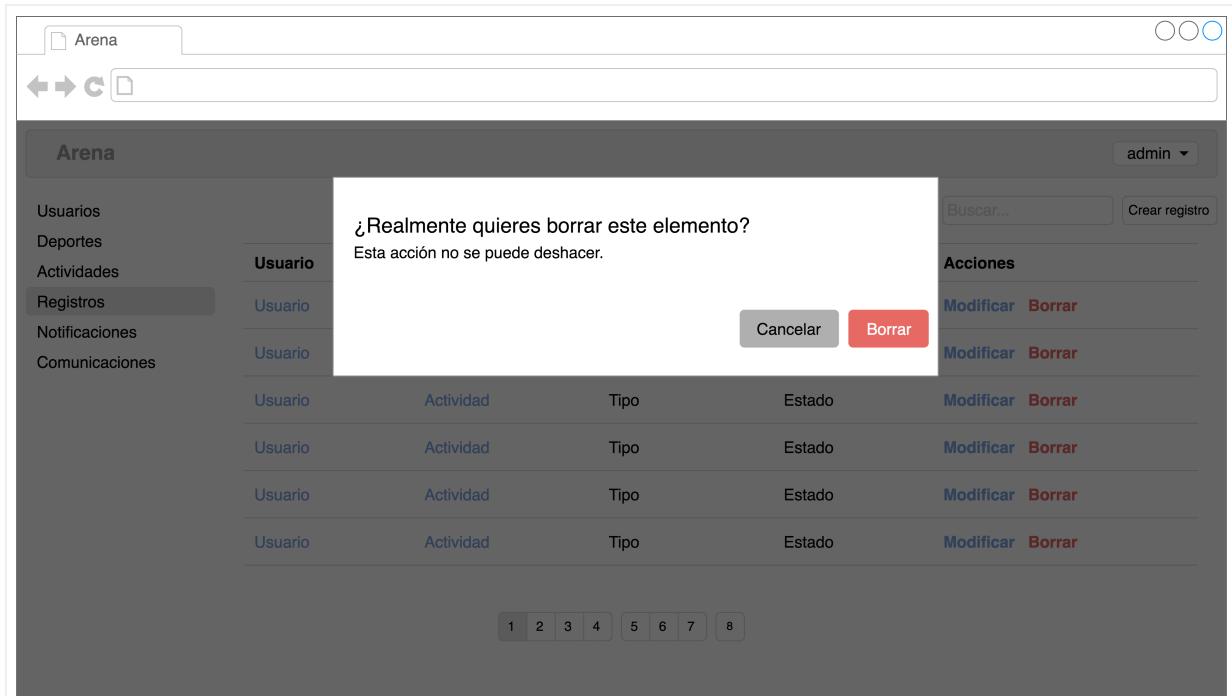


Figura 10.25 Wireframe de borrado de registros

El usuario administrador podrá seleccionar en el menú la opción para administrar notificaciones como en la "Figura 10.27".

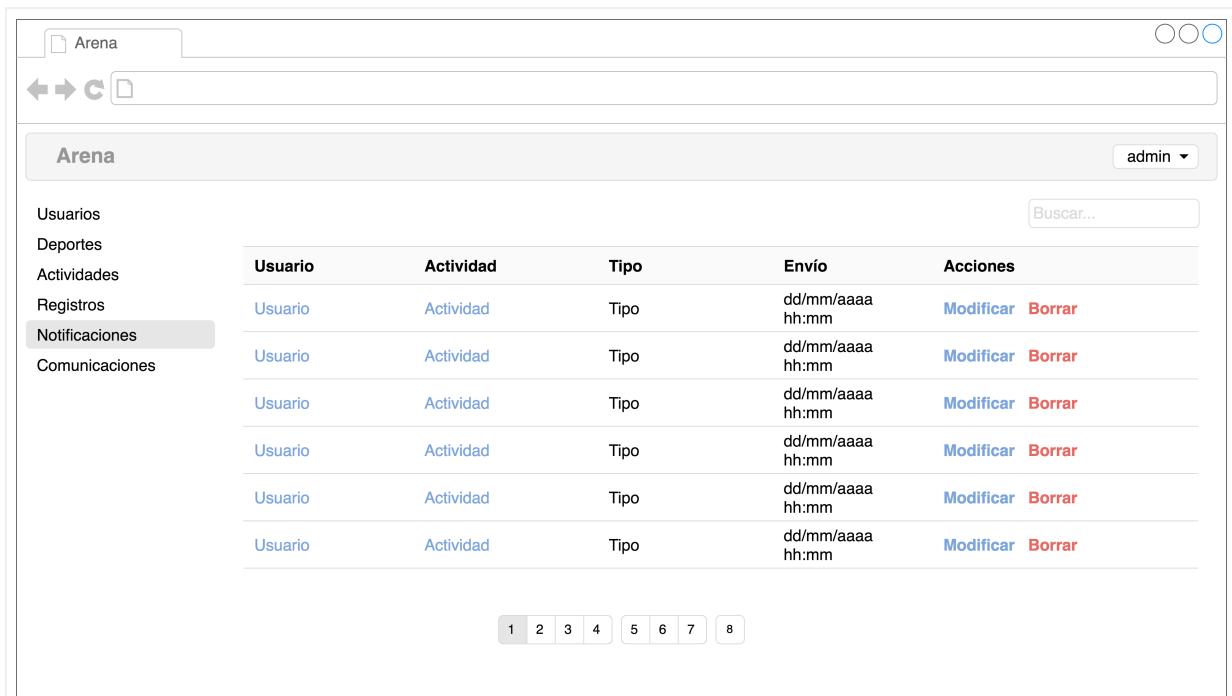


Figura 10.26 Wireframe de listado de notificaciones

El usuario administrador podrá seleccionar la opción para editar una determinada notificación como en la "Figura 10.28".

Este wireframe muestra la interfaz de usuario para la modificación de notificaciones. En la parte superior, se encuentra el menú principal con las opciones: Usuarios, Deportes, Actividades, Registros, Notificaciones (destacada) y Comunicaciones. A la derecha del menú, se encuentran los campos para búsqueda: Usuario, Actividad, Tipo y un campo para actividad. En la parte superior derecha, hay un botón para administración y tres iconos de control (minimizar, maximizar, cerrar). El fondo es blanco y no contiene información detallada.

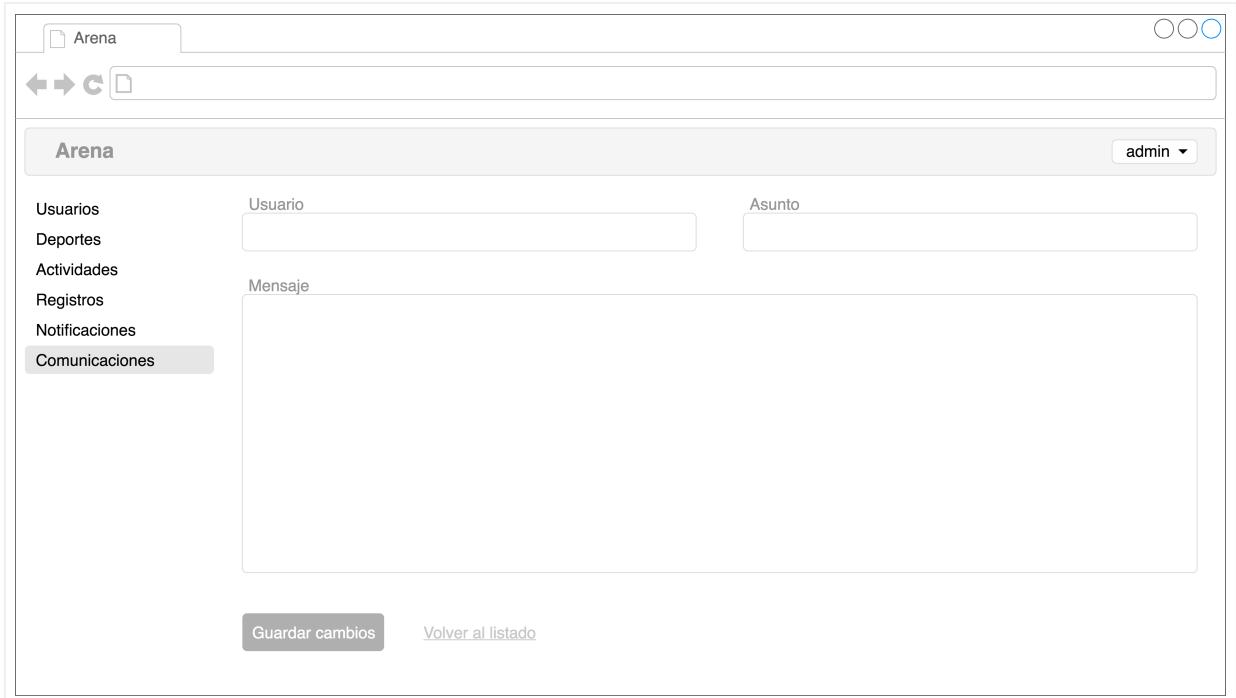
Figura 10.27 Wireframe de modificación de notificaciones

El usuario administrador podrá seleccionar la opción para eliminar una determinada notificación como en la "Figura 10.29".

Este wireframe muestra la interfaz de usuario para el borrado de notificaciones. La estructura es similar al anterior, con el menú principal y los campos de búsqueda. Sin embargo, un cuadro de diálogo central pregunta: "¿Realmente quieres borrar este elemento? Esta acción no se puede deshacer." Dentro del cuadro, hay dos botones: "Cancelar" y "Borrar". A la derecha del cuadro, se muestra una lista de notificaciones con columnas para Usuario, Actividad, Tipo y fecha/hora. Cada fila tiene botones para "Modificar" y "Borrar". Abajo de la lista, se encuentran los números de página 1 a 8.

Figura 10.28 Wireframe de borrado de notificaciones

El usuario administrador podrá seleccionar la opción para crear una nueva comunicación como en la "Figura 10.30".

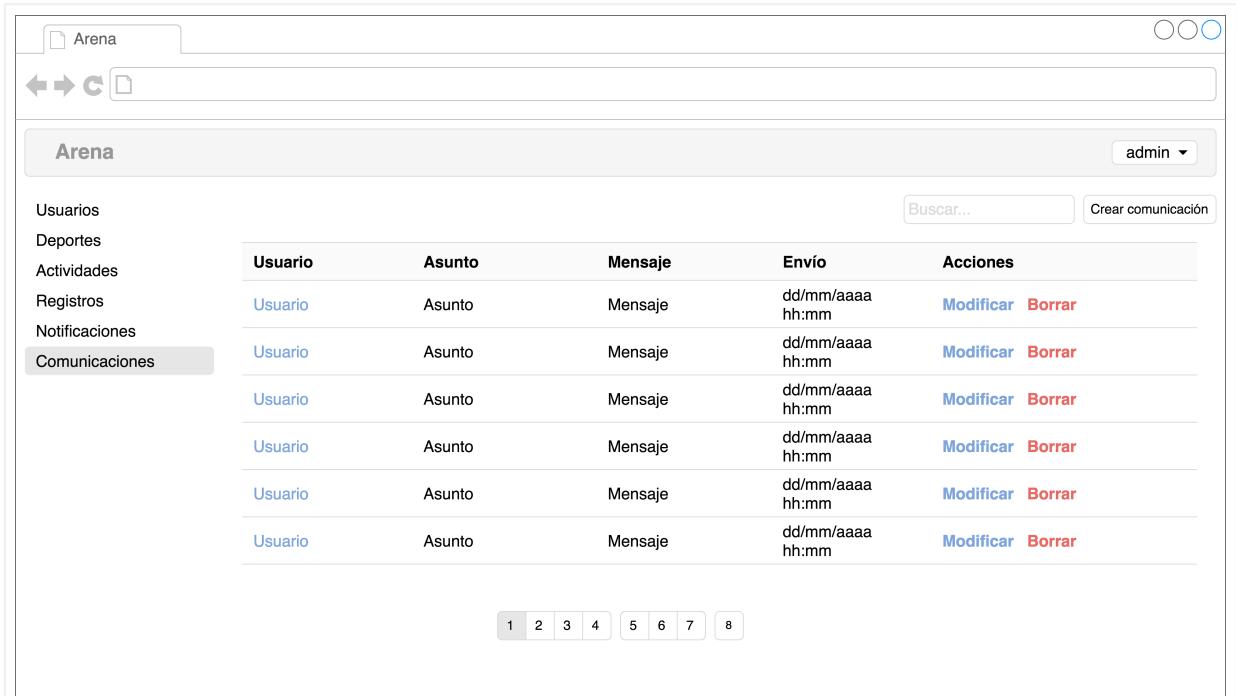


Wireframe de creación de comunicaciones:

- Barra superior:** Botones para navegar entre aplicaciones, un botón para cerrar la ventana y el nombre del espacio de trabajo "Arena".
- Menú:** Una lista desplegable que incluye "admin" y otros roles.
- Tabs:** Una barra horizontal con los siguientes tabs: "Usuarios", "Deportes", "Actividades", "Registros", "Notificaciones" y "Comunicaciones". El tab "Comunicaciones" está resaltado.
- Formulario:** Un cuadro grande que contiene campos para "Usuario" (que se ha escrito "User") y "Asunto", así como un área grande para el "Mensaje".
- Botones de acción:** "Guardar cambios" y "Volver al listado".

Figura 10.29 Wireframe de creación de comunicaciones

El usuario administrador podrá seleccionar en el menú la opción para administrar comunicaciones como en la "Figura 10.31".



Wireframe de listado de comunicaciones:

Arena					
	Usuario	Asunto	Mensaje	Envío	Acciones
Usuarios	Usuario	Asunto	Mensaje	dd/mm/aaaa hh:mm	Modificar Borrar
Deportes	Usuario	Asunto	Mensaje	dd/mm/aaaa hh:mm	Modificar Borrar
Actividades	Usuario	Asunto	Mensaje	dd/mm/aaaa hh:mm	Modificar Borrar
Registros	Usuario	Asunto	Mensaje	dd/mm/aaaa hh:mm	Modificar Borrar
Notificaciones	Usuario	Asunto	Mensaje	dd/mm/aaaa hh:mm	Modificar Borrar
Comunicaciones	Usuario	Asunto	Mensaje	dd/mm/aaaa hh:mm	Modificar Borrar

Botones de búsqueda y creación: "Buscar..." y "Crear comunicación".

Página: 1 2 3 4 5 6 7 8

Figura 10.30 Wireframe de listado de comunicaciones

El usuario administrador podrá seleccionar la opción para editar una determinada comunicación como en la "Figura 10.32".

Este wireframe muestra la interfaz de usuario para la edición de una comunicación. En la parte superior izquierda, hay un menú lateral con las siguientes opciones: Usuarios, Deportes, Actividades, Registros, Notificaciones y Comunicaciones. La opción 'Comunicaciones' está resaltada. En el centro, se visualiza un formulario para editar una comunicación, que incluye campos para 'Usuario' (con el valor 'Usuario'), 'Asunto' (con el valor 'Asunto') y 'Mensaje' (con el valor 'Mensaje'). A la derecha del formulario, hay un botón 'admin' con un menú desplegable. En la parte inferior, se encuentran los botones 'Guardar cambios', 'Borrar' y 'Volver al listado'.

Figura 10.31 Wireframe de modificación de comunicaciones

El usuario administrador podrá seleccionar la opción para eliminar una comunicación como en la "Figura 10.33".

Este wireframe muestra la interfaz de usuario para la eliminación de una comunicación. En la parte superior izquierda, hay un menú lateral con las siguientes opciones: Usuarios, Deportes, Actividades, Registros, Notificaciones y Comunicaciones. La opción 'Comunicaciones' está resaltada. En el centro, se visualiza una lista de comunicaciones con columnas para 'Usuario', 'Asunto', 'Mensaje', 'dd/mm/aaaa hh:mm' y 'Acciones'. Una气泡对话框弹出，显示询问：“¿Realmente quieres borrar este elemento? Esta acción no se puede deshacer.” con botones 'Cancelar' y 'Borrar'. A la derecha de la lista, hay un botón 'Crear comunicación' y una sección titulada 'Acciones' con los botones 'Modificar' y 'Borrar' repetidos para cada fila. En la parte inferior, se encuentran los botones '1', '2', '3', '4', '5', '6', '7' y '8' para navegar entre páginas.

Figura 10.32 Wireframe de borrado de comunicaciones

Diseño

Esta página se ha dejado vacía a propósito.

Capítulo 11

Diseño del sistema

Una vez finalizada la fase de análisis, especificaremos las decisiones tomadas para resolver el problema y nos aproximaremos a la solución básica del sistema antes de proceder al diseño de los datos.

11.1 Organización de los componentes

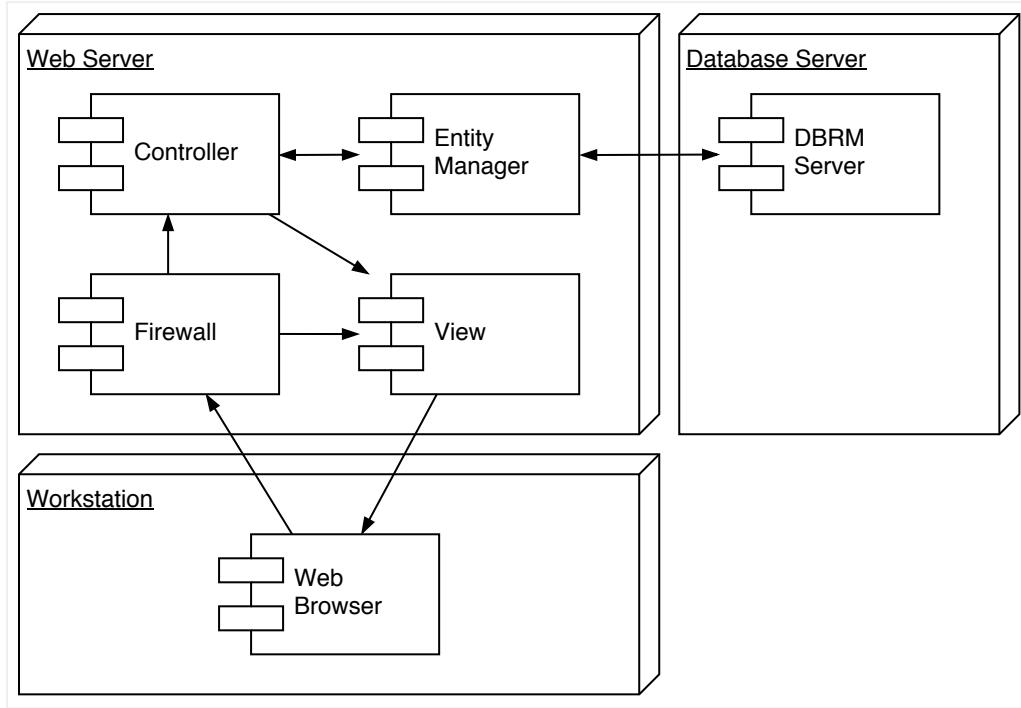
A través de los diagramas de despliegue mostraremos la composición arquitectónica de nuestro sistema. Separaremos lo que es la arquitectura del *backend* de la del *frontend*.

11.2 Diagrama de despliegue: *backend*

En este caso, el diagrama consta de tres nodos como se ve en la "Figura 11.1":

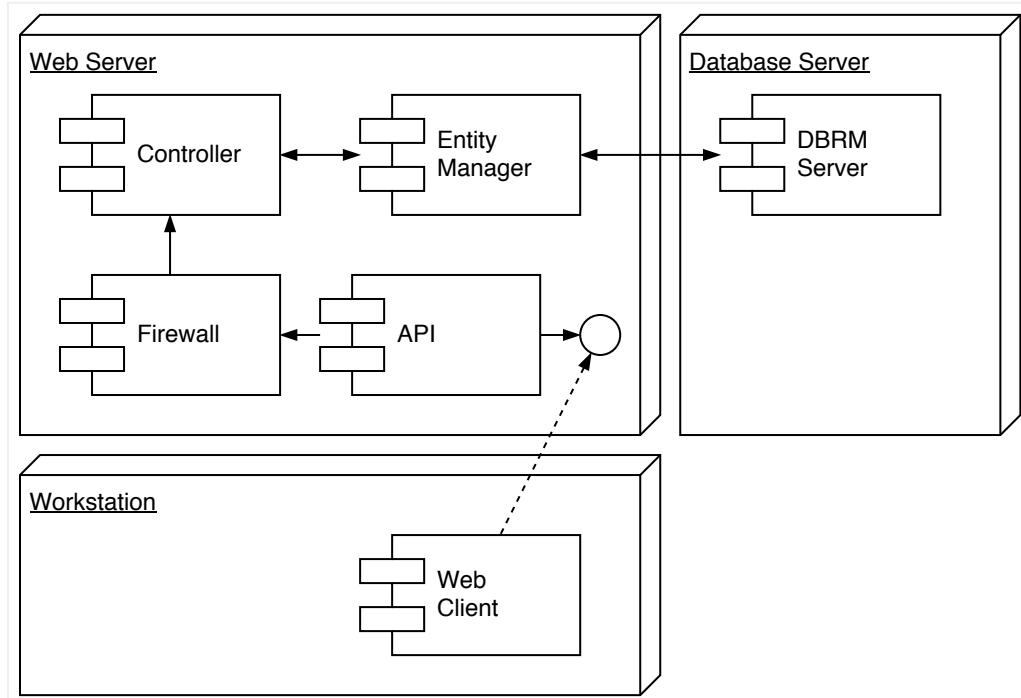
- Nodo *cliente*. Es el nodo del usuario que accede al sistema a través de un navegador.
- Nodo *base de datos*. Es el nodo que contiene la base de datos del sistema, manejada por un *motor relacional*. Solo interacciona con el nodo principal a través del controlador de entidad (*Entity Manager* o *EM*). El *EM* es un software, en este caso un **ORM**^[1] llamado Doctrine2, que se encarga de transformar el modelo de datos en tablas y viceversa, de forma transparente. De esta manera, lo que para el sistema es un objeto, para la base de datos es una tabla, y para lo que el sistema es una instancia de un objeto, para la base de datos es una tupla de dicha tabla.
- Nodo *controlador* o nodo *servidor*. Es el que contiene toda la lógica del sistema. El navegador accede a este nodo a través del *cortafuegos*. Este tiene una doble función: autenticación y autorización. Es el responsable de dar acceso y controlar que las acciones que haga el usuario estén habilitadas para su rol. El resto de componentes son los clásicos de un paradigma *Modelo-Vista-Controlador*.

[1] *Object Relational Model*: https://es.wikipedia.org/wiki/Mapeo_objeto-relacional/

**Figura 11.1** Diagrama de despliegue del backend

11.3 Diagrama de despliegue: *frontend*

El esquema de la "Figura 11.2" es similar al anterior, excepto por una diferencia, en este caso la vista es generada por el cliente y no el servidor porque la *API* devuelve los datos de forma serializada. Así, el cliente puede obtener los datos como objetos y tratarlos como mejor le convenga.

**Figura 11.2** Diagrama de despliegue del frontend

11.4 Esquema de seguridad del sistema

Debemos establecer cuáles serán las políticas de seguridad del sistema. El sistema diferencia entre tres diferentes roles: administrador, usuario y usuario anónimo. En la "Tabla 11.1" se pueden diferenciar los permisos que se implementarán en el sistema por rol.

Tarea	Administrador	Usuario	Usuario anónimo
Listar usuarios	✓	✗	✗
Editar usuarios	✓	✗	✗
Eliminar usuarios	✓	✗	✗
Cerrar sesión	✓	✓	✗
Editar mi perfil	✓	✓	✗
Borrar mi perfil	✓	✓	✗
Crear deportes	✓	✗	✗
Listar deportes	✓	✓	✓
Editar deportes	✓	✗	✗
Eliminar deportes	✓	✗	✗
Crear actividades	✓	✓	✗
Listar actividades	✓	✓	✓
Editar actividades	✓	✗	✗
Eliminar actividades	✓	✗	✗
Crear registros	✓	✗	✗
Listar registros	✓	✗	✗
Editar registros	✓	✗	✗
Eliminar registros	✓	✗	✗
Listar notificaciones	✓	✗	✗
Editar notificaciones	✓	✗	✗
Eliminar notificaciones	✓	✗	✗
Leer notificaciones	✓	✓	✗
Crear comunicaciones	✓	✗	✗
Listar comunicaciones	✓	✗	✗
Editar comunicaciones	✓	✗	✗
Eliminar comunicaciones	✓	✗	✗

Tarea	Administrador	Usuario	Usuario anónimo
Enviar comunicaciones	✓	✗	✗
Invitar a participar	✓	✓	✗
Ver mis invitaciones	✓	✓	✗
Cancelar mis invitaciones	✓	✓	✗
Aceptar invitaciones	✓	✓	✗
Rechazar invitaciones	✓	✓	✗
Ver mis solicitudes	✓	✓	✗
Cancelar mis solicitudes	✓	✓	✗
Aceptar solicitudes	✓	✓	✗
Rechazar solicitudes	✓	✓	✗
Eliminar participantes	✓	✓	✗
Eliminar mis participaciones	✓	✓	✗

Tabla 11.1 Esquema de seguridad del sistema

11.5 Identificación de la concurrencia

Este sistema es concurrente, puesto que más de un usuario podría estar editando el mismo objeto^[2]. Sin embargo, este sistema se espera que tenga un pequeño número de usuarios con respecto al número de actividades de la aplicación así que las posibilidades de problemas de concurrencia son muy bajas. Solo podrían presentarse en dos ocasiones:

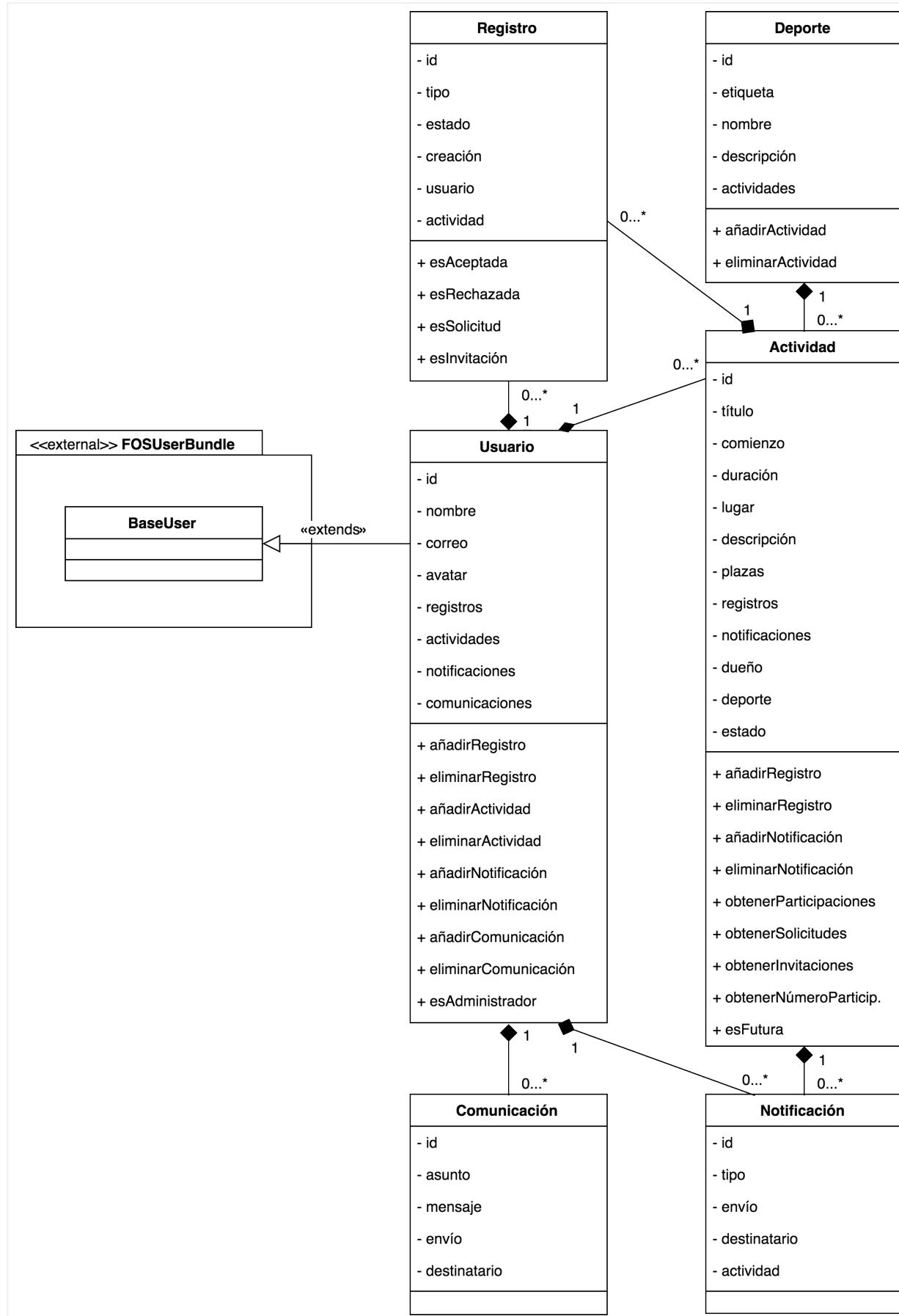
1. Edición simultánea de un mismo elemento. Para solventar este problema, se comprobará la consistencia de la base de datos, y que los datos que se enviaron a guardar coinciden con los existentes en el formulario que se envió. En caso de error se haría un *rollback* y se mostraría un mensaje de error. Además, podemos controlar la concurrencia con *Doctrine* a través del bloqueo optimista, bloqueando un recurso si la versión que tiene este no coincide con la que debería tener.
2. Edición de un elemento ya borrado. En este caso, se lanzaría un mensaje de error de que el elemento ya no existe.

[2] Concurrencia: <http://docs.doctrine-project.org/projects/doctrine-orm/en/latest/reference/transactions-and-concurrency.html>

Capítulo 12

Diseño de datos

Una vez realizado el análisis completo del sistema, casos de uso, diagrama de paquetes y diagramas de secuencias, podemos concluir el diseño del sistema al completo. La "Figura 12.1" muestra la composición de las clases del modelo de datos y sus dependencias externas. Para simplificar, se omiten todos los métodos *get*, *set*, *construct* y *toString*, el tipo devuelto y los tipos de datos ya que se especifican posteriormente. Salvo que se indique lo contrario, todos los atributos son de lectura y escritura, excepto *id* que representa la clave primaria y no puede cambiar.

**Figura 12.1** Modelo de clases completo

12.1 Diseño de la clase *User*

La clase *User* hereda de la clase externa *BaseUser*, perteneciente al paquete *FOSUserBundle*.

Atributos en la "Tabla 12.1".

Nombre	Tipo	Descripción
id	string	Clave primaria
fullName	string	Nombre completo
email	string	Correo electrónico
avatar	string	Ruta de la imagen de perfil
activities	Collection	Actividades creadas
registrations	Collection	Registros donde está implicado
notifications	Collection	Notificaciones recibidas
communications	Collection	Comunicaciones recibidas

Tabla 12.1 Atributos de la clase User

Métodos en la "Tabla 12.2".

Nombre	Retorno	Descripción
__construct()	self	Construye la instancia
_toString()	string	Obtiene el nombre de la instancia
getId()	string	Obtiene la clave primaria
getFullName()	string	Obtiene el nombre completo
setFullName(string: fullName)	self	Establece el nombre completo
getEmail()	string	Obtiene el correo electrónico
setEmail(string: email)	self	Establece el correo electrónico
getAvatar()	string	Obtiene la ruta del avatar
setAvatar(string: avatar)	self	Establece la ruta del avatar
getActivities()	Collection	Obtiene sus actividades
addActivity(Activity: activity)	self	Añade su actividad
removeActivity(Activity: activity)	self	Elimina su actividad
getRegistrations()	Collection	Obtiene sus registros

Nombre	Retorno	Descripción
addRegistration(Registration: registration)	self	Añade su registro
removeRegistration(Registration: registration)	self	Elimina su registro
getNotifications()	Collection	Obtiene sus notificaciones
addNotification(Notification: notification)	self	Añade su notificación
removeNotification(Notification: notification)	self	Elimina su notificación
getCommunications()	Collection	Obtiene sus comunicaciones
addCommunication(Communication: communication)	self	Añade su comunicación
removeCommunication(Communication: communication)	self	Elimina su comunicación
isAdmin()	bool	Devuelve si es administrador

Tabla 12.2 Métodos de la clase User

12.2 Diseño de la clase Sport

Atributos en la "Tabla 12.3".

Nombre	Tipo	Descripción
id	string	Clave primaria
slug	string	Etiqueta
name	string	Nombre
description	string	Descripción
activities	Collection	Lista de actividades

Tabla 12.3 Atributos de la clase Sport

Métodos en la "Tabla 12.4".

Nombre	Retorno	Descripción
__construct()	self	Construye la instancia
__toString()	string	Obtiene el nombre de la instancia
getId()	string	Obtiene la clave primaria

Nombre	Retorno	Descripción
getSlug()	string	Obtiene la etiqueta
setSlug(string: slug)	self	Establece la etiqueta
getName()	string	Obtiene el nombre
setName(string: name)	self	Establece el nombre
getDescription()	string	Obtiene la descripción
setDescription(string: description)	self	Establece la descripción
getActivities()	Collection	Obtiene la lista de actividades
addActivities(Activity: activity)	self	Añade una actividad a la lista
removeActivities(Activity: activity)	self	Elimina una actividad de la lista

Tabla 12.4 Métodos de la clase Sport

12.3 Diseño de la clase *Activity*

Atributos en la "Tabla 12.5".

Nombre	Tipo	Descripción
id	string	Clave primaria
title	string	Nombre
startsAt	datetime	Fecha y hora de comienzo
duration	integer	Duración
location	string	Lugar
description	string	Descripción
seats	integer	Plazas
registrations	Collection	Registros donde está implicada
notifications	Collection	Notificaciones donde está implicada
owner	User	Propietario
sport	Sport	Deporte
status	string	Estado

Tabla 12.5 Atributos de la clase Activity

Métodos en la "Tabla 12.6".

Nombre	Retorno	Descripción
__construct()	self	Construye la instancia
_toString()	string	Obtiene el nombre de la instancia
getId()	string	Obtiene la clave primaria
getTitle()	string	Obtiene el nombre
setTitle(string: title)	self	Establece el nombre
getStartsAt()	datetime	Obtiene la fecha y hora de comienzo
setStartsAt(datetime: startsAt)	self	Establece la fecha y hora de comienzo
getDuration()	integer	Obtiene la duración
setDuration(integer: duration)	self	Establece la duración
getLocation()	string	Obtiene el lugar
setLocation(string: location)	self	Establece el lugar
getDescription()	string	Obtiene la descripción
setDescription()	string	Establece la descripción
getSeats()	integer	Obtiene el número de plazas
setSeats(integer: seats)	self	Establece el número de plazas
getRegistrations()	Collection	Obtiene sus registros
addRegistration(Registration: registration)	self	Añade su registro
removeRegistration(Registration: registration)	self	Elimina su registro
getNotifications()	Collection	Obtiene sus notificaciones
addNotification(Notification: notification)	self	Añade su notificación
removeNotification(Notification: notification)	self	Elimina su notificación
getOwner()	User	Obtiene el propietario
setOwner(User: owner)	self	Establece el propietario
getSport()	Sport	Obtiene el deporte
getSport(Sport: sport)	self	Establece el deporte
getStatus()	string	Obtiene el estado respecto al usuario
setStatus(string: string)	self	Establece el estado respecto al usuario
setStatusWithUser(User: user)	self	Establece el estado respecto al usuario

Nombre	Retorno	Descripción
getAcceptedRegistrations()	Collection	Obtiene sus registros aceptados
getApplications()	Collection	Obtiene sus solicitudes
getInvitations()	Collection	Obtiene sus invitaciones
getAcceptedRegistrationsCount()	integer	Obtiene el número de sus registros aceptados

Tabla 12.6 Métodos de la clase Activity

12.4 Diseño de la clase *Registration*

Atributos en la "Tabla 12.7".

Nombre	Tipo	Descripción
id	string	Clave primaria
type	string	Tipo
status	string	Estado
createdAt	datetime	Fecha y hora de creación
user	User	Usuario
activity	Activity	Actividad

Tabla 12.7 Atributos de la clase Registration

Métodos en la "Tabla 12.8".

Nombre	Retorno	Descripción
__construct()	self	Construye la instancia
_toString()	string	Obtiene el nombre de la instancia
getId()	string	Obtiene la clave primaria
getType()	string	Obtiene el tipo
setType(string: type)	self	Establece el tipo
getStatus()	string	Obtiene el estado
setStatus(string: status)	self	Establece el estado
getCreatedAt()	datetime	Obtiene la fecha y hora de creación
setCreatedAt(datetime: createdAt)	self	Establece la fecha y hora de creación
getUser()	User	Obtiene su usuario

Nombre	Retorno	Descripción
setUser(User: user)	self	Establece su usuario
getActivity()	Activity	Obtiene su actividad
setActivity(Activity: activity)	self	Establece su actividad
isAccepted	bool	Devuelve si está aceptada
isRefused	bool	Devuelve si está rechazada
isApplication	bool	Devuelve si es solicitud
isInvitation	bool	Devuelve si es invitación

Tabla 12.8 Métodos de la clase Registration

12.5 Diseño de la clase *Notification*

Atributos en la "Tabla 12.9".

Nombre	Tipo	Descripción
id	string	Clave primaria
type	string	Tipo
sentAt	datetime	Fecha y hora de envío
user	User	Usuario
activity	Activity	Actividad

Tabla 12.9 Atributos de la clase Notification

Métodos en la "Tabla 12.10".

Nombre	Retorno	Descripción
__construct()	self	Construye la instancia
__toString()	string	Obtiene el nombre de la instancia
getId()	string	Obtiene la clave primaria
getType()	string	Obtiene el tipo
setType(string: type)	self	Establece el tipo
getSentAt()	datetime	Obtiene la fecha y hora de envío
setSentAt(datetime: sentAt)	self	Establece la fecha y hora de envío
getUser()	User	Obtiene su usuario

Nombre	Retorno	Descripción
setUser(User: user)	self	Establece su usuario
getActivity()	Activity	Obtiene su actividad
setActivity(Activity: activity)	self	Establece su actividad

Tabla 12.10 Métodos de la clase Notification

12.6 Diseño de la clase *Communication*

Atributos en la "Tabla 12.11".

Nombre	Tipo	Descripción
id	string	Clave primaria
subject	string	Asunto
body	string	Mensaje
sentAt	datetime	Fecha y hora de envío
user	User	Usuario

Tabla 12.11 Atributos de la clase Communication

Métodos en la "Tabla 12.12".

Nombre	Retorno	Descripción
__construct()	self	Construye la instancia
_toString()	string	Obtiene el nombre de la instancia
getId()	string	Obtiene la clave primaria
getSubject()	string	Obtiene el asunto
setSubject(string: type)	self	Establece el asunto
getBody()	string	Obtiene el mensaje
setBody(string: body)	self	Establece el mensaje
getSentAt()	datetime	Obtiene la fecha y hora de envío
setSentAt(datetime: sentAt)	self	Establece la fecha y hora de envío
getUser()	User	Obtiene el usuario
setUser(User: user)	self	Establece el usuario

Tabla 12.12 Métodos de la clase Communication

12.7 Modelo Entidad-Relación

Aunque para el desarrollo del sistema, se usa el modelo de datos, este tiene una representación en la base de datos que almacena la información. Si bien, *Doctrine2* nos abstrae del sistema de almacenamiento, como referencia indicamos en la "Figura 12.2" el modelo entidad-relación que se ha generado a partir de nuestro modelo de datos. Para la notación se ha seguido el esquema *patas de gallo* (*Stewart, 2008*).

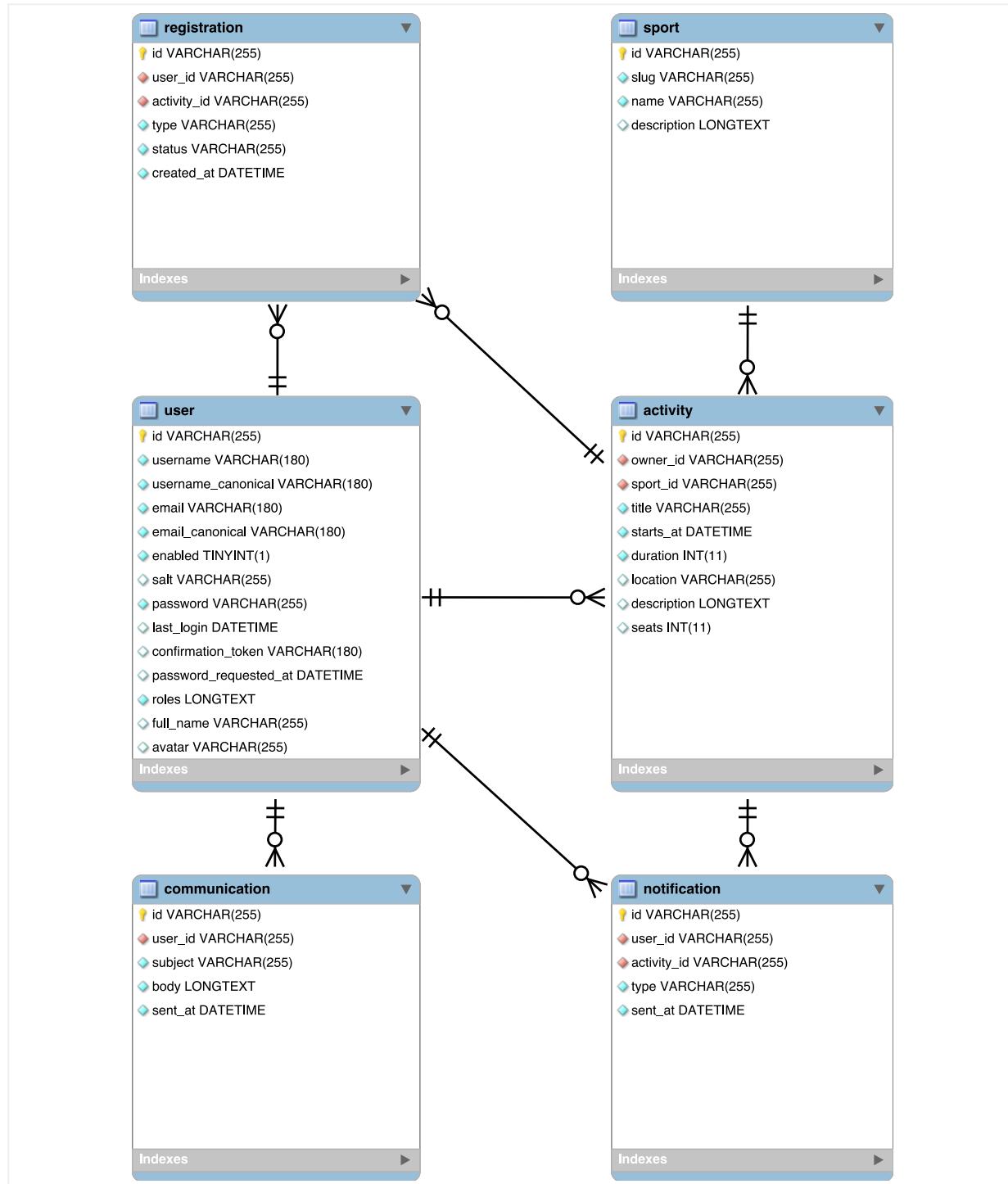


Figura 12.2 Modelo Entidad-Relación

Capítulo 13

Diseño de la interfaz

A continuación, presentamos, a partir de los requisitos impuestos en la fase de análisis, los diseños de las vistas de nuestra aplicación. Como hemos dividido la aplicación en dos partes, las visualizaremos por separado.

13.1 Interfaz de administración

13.1.1 Elementos genéricos

La interfaz de administración está compuesta por una serie de elementos genéricos mostrados en la "Figura 13.1":

1. *Barra lateral.* Permite acceder rápidamente a la sección de listado de cualquier elemento desde cualquier lugar.
2. *Botón de despliegue.* Este botón permite mostrar y ocultar la barra lateral para permitir más o menos espacio a la sección principal. Está pensando para monitores con poca resolución.
3. *Botón de salida.* Este botón permite al usuario cerrar su sesión y salir de la aplicación.



Figura 13.1 Elementos genéricos de la interfaz de administración

13.1.2 Interfaz de listado de usuarios

La interfaz de listado de usuarios está compuesta por una serie de elementos mostrados en la "Figura 13.2":

1. *Tabla de usuarios.* Cada fila muestra la información de un usuario y la tabla posibilita la ordenación de las filas por algún atributo.
2. *Acciones.* Las acciones que el administrador puede realizar con cada usuario.
3. *Búsqueda.* El sistema permite buscar resultados para encontrar elementos concretos.

Usuarios		3				Buscar
	Nombre completo	Correo electrónico	Actividades	Registros	Habilidades	Acciones
1	Elena Diaz Crespo	elenadiaz@gmail.com	1	1	Si	Modificar Borrar
2	Laura González Moreno-Vaquerizo	lauragonzalez@gmail.com	1	1	Si	Modificar Borrar
3	Jesús Rojas Ramos	jesusrojas@gmail.com	1	0	Si	Modificar Borrar
4	José Manuel Ruano Ruiz	josemanuelruano@gmail.com	1	0	Si	Modificar Borrar
5	Francisco José Rodríguez Ramírez	franciscorodriguez@gmail.com	1	0	Si	Modificar Borrar
6	Rubén Medina Zamorano	rubenmedina@gmail.com	2	0	Si	Modificar Borrar
7	José Pérez-Parras Toledo	joseperezparras@gmail.com	1	0	Si	Modificar Borrar
8	Omar Sotillo Franco	omarsotillo@gmail.com	1	0	Si	Modificar Borrar
9	Iván Portillo Leal	ivanportillo@gmail.com	1	1	Si	Modificar Borrar
10	Víctor Monserrat Villatoro	victor1995mv@gmail.com	2	8	Si	Modificar Borrar
NULL	admin@admin.com	admin@admin.com	0	0	Si	Modificar Borrar

Figura 13.2 Interfaz de listado de usuarios

13.1.3 Interfaz de edición de usuarios

La interfaz de edición de usuarios se compone de los elementos de la "Figura 13.3":

1. *Formulario.* Se podrán modificar el nombre completo y el correo electrónico.
2. *Botón de guardado.* Permite guardar los cambios.
3. *Botón de borrado y 4. Botón para volver.*

Nombre completo
 Víctor Monserrat Villatoro

Correo electrónico *
 victor1995mv@gmail.com

Guardar cambios Borrar Volver al listado

Figura 13.3 Interfaz de edición de usuarios

13.1.4 Interfaz de borrado de usuarios

La interfaz de borrado de usuarios está compuesta por una serie de elementos mostrados en la "Figura 13.4":

1. *Mensaje de confirmación.*
2. *Botón de cancelación.* Permite cancelar el borrado.
3. *Botón de confirmación.* Permite confirmar el borrado.

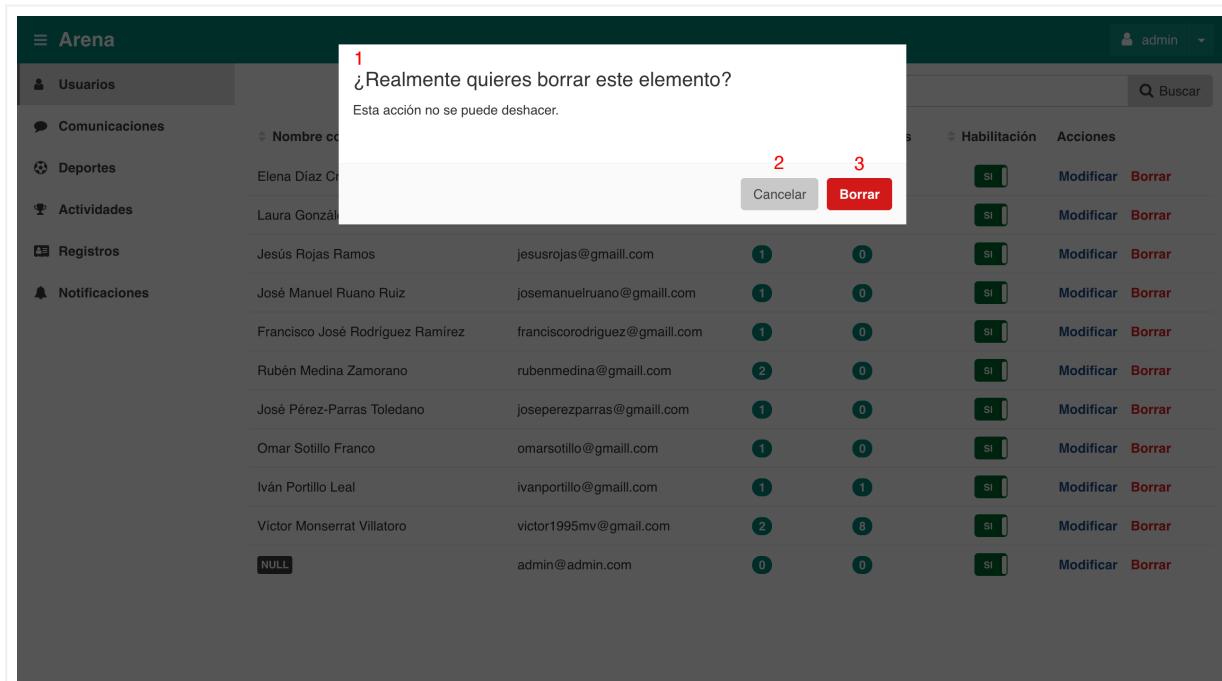


Figura 13.4 Interfaz de borrado de usuarios

13.1.5 Interfaz de listado de comunicaciones

La interfaz de listado de comunicaciones está compuesta por una serie de elementos mostrados en la "Figura 13.5":

1. *Tabla de usuarios.* Cada fila muestra la información de una comunicación y la tabla posibilita la ordenación de las filas por algún atributo.
2. *Búsqueda.* El sistema permite buscar resultados para encontrar elementos concretos.
3. *Botón de creación.* Permite crear una nueva comunicación.

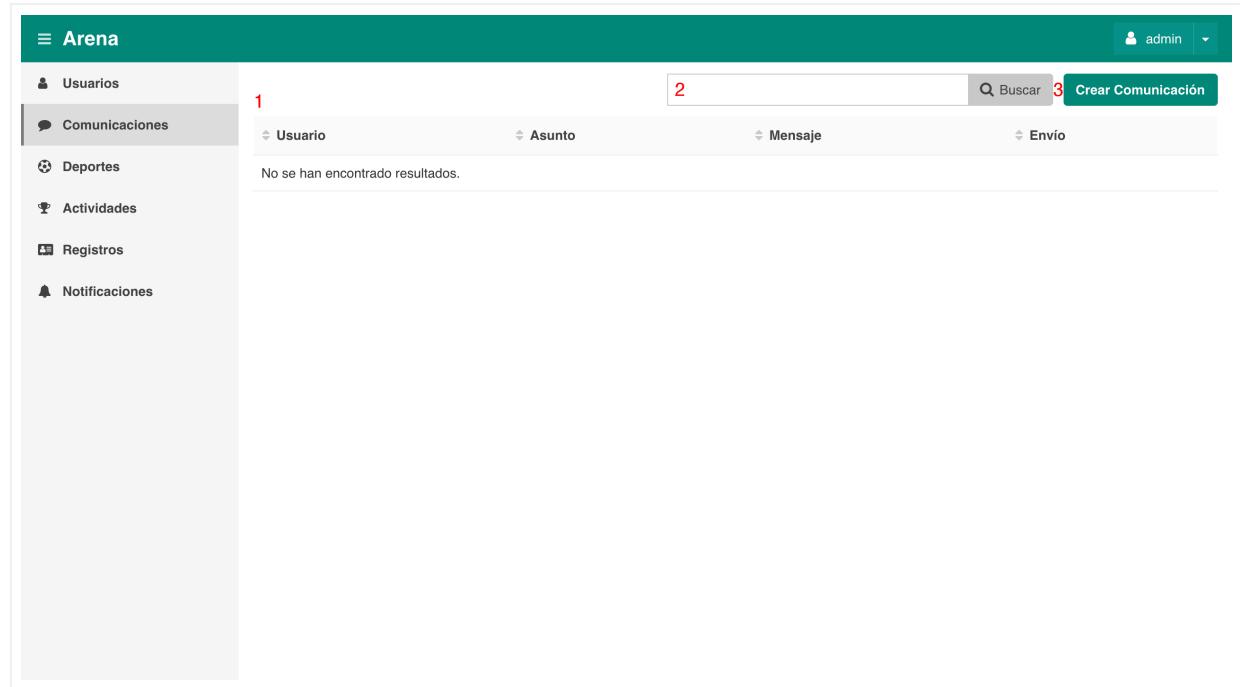


Figura 13.5 Interfaz de listado de comunicaciones

13.1.6 Interfaz de creación de comunicaciones

La interfaz de creación de comunicaciones se compone de los elementos de la "Figura 13.6":

1. *Formulario.* Se podrán establecer el nombre completo y el correo electrónico.
2. *Botón de guardado.* Permite guardar los cambios.
3. *Botón para volver.* Permite volver al listado de usuarios.

1 Usuario * <input type="text" value="victor1995mv@gmail.com"/>	2 Asunto * <input type="text"/>	3 Mensaje * <div style="border: 1px solid #ccc; height: 100px; margin-bottom: 10px;"></div>
Guardar cambios Volver al listado		

Figura 13.6 Interfaz de creación de comunicaciones

13.1.7 Interfaz de listado de deportes

La interfaz de listado de deportes está compuesta por una serie de elementos mostrados en la "Figura 13.7":

1. *Tabla de usuarios.* Cada fila muestra la información de un deporte y la tabla posibilita la ordenación de las filas por algún atributo.
2. *Acciones.* Las acciones que el administrador puede realizar con cada deporte.
3. *Búsqueda.* El sistema permite buscar resultados para encontrar elementos concretos.
4. *Botón de creación.* Permite crear un nuevo deporte.

Nombre	Descripción	Actividades	Acciones
Running	Perspiciatis tempora aut alias quidem. Atque nulla necessitatibus...	2	Modificar Borrar
Natación	Veniam non perspiciatis totam et. Corrupti eius optio nihil volu...	1	Modificar Borrar
Balonmano	Cupiditate vitae dolores et rem est ipsam voluptatem. Ut quibusd...	1	Modificar Borrar
Ciclismo	Accusamus ex voluptatem et eveniet nemo voluptibus necessitatibus...	1	Modificar Borrar
Pádel	Non enim enim cumque quasi corporis impedit porro. Adipisci vita...	1	Modificar Borrar
Tenis	Aut ipsum possimus consequuntur aliquam fugiat sit ratione. Aut...	1	Modificar Borrar
Baloncesto	Occaecati nihil quam quaerat ipsum earum vel. Itaque voluptatem...	3	Modificar Borrar
Fútbol	Et optio rerum quaerat. Fuga non quibusdam itaque. Quia at quia...	2	Modificar Borrar

Figura 13.7 Interfaz de listado de deportes

13.1.8 Interfaz de creación de deportes

La interfaz de creación de deportes está compuesta por una serie de elementos mostrados en la "Figura 13.8":

1. *Formulario.* Se podrán establecer el nombre y la descripción.
2. *Botón de guardado.* Permite guardar los cambios.
3. *Botón para volver.* Permite volver al listado de deportes.

Figura 13.8 Interfaz de creación de deportes

13.1.9 Interfaz de edición de deportes

La interfaz de edición de deportes se compone de los elementos de la "Figura 13.9":

1. *Formulario.* Se podrán modificar el nombre y la descripción.
2. *Botón de guardado.* Permite guardar los cambios.
3. *Botón de borrado y 4. Botón para volver.*

Figura 13.9 Interfaz de edición de deportes

13.1.10 Interfaz de borrado de deportes

La interfaz de borrado de deportes está compuesta por una serie de elementos mostrados en la "Figura 13.10":

1. *Mensaje de confirmación.*
2. *Botón de cancelación.* Permite cancelar el borrado.
3. *Botón de confirmación.* Permite confirmar el borrado.

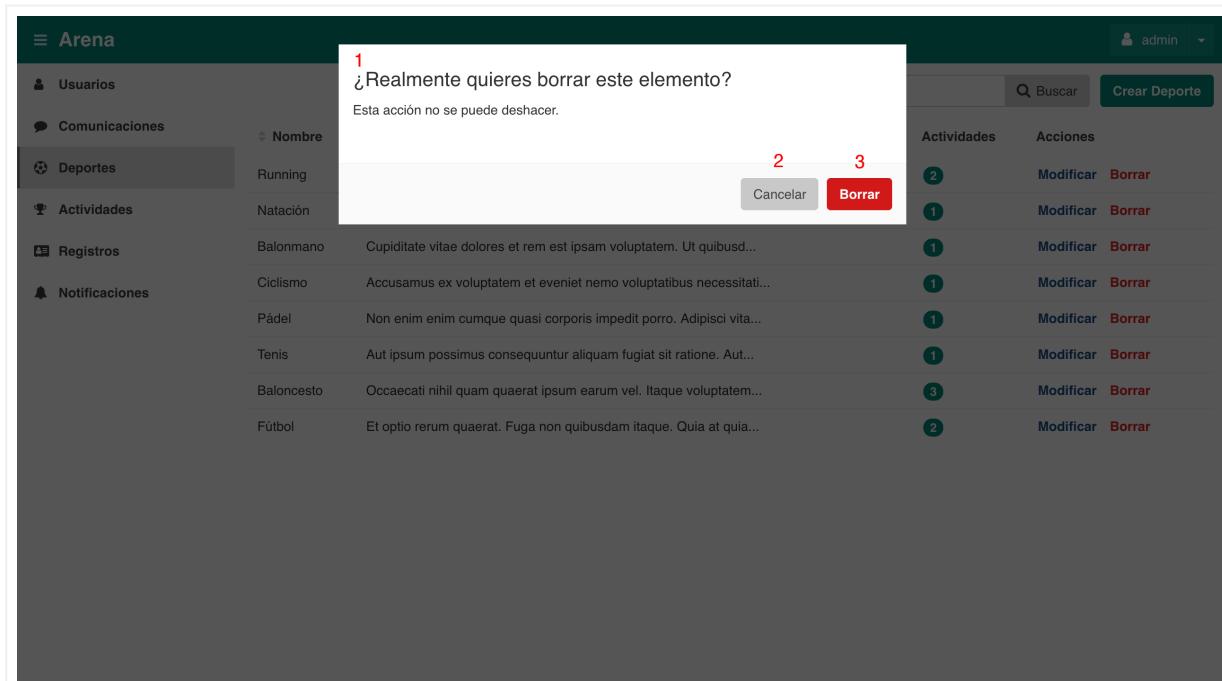


Figura 13.10 Interfaz de borrado de deportes

13.1.11 Interfaz de listado de actividades

La interfaz de listado de actividades está compuesta por una serie de elementos mostrados en la "Figura 13.11":

1. *Tabla de usuarios.* Cada fila muestra la información de una actividad y la tabla posibilita la ordenación de las filas por algún atributo.
2. *Acciones.* Las acciones que el administrador puede realizar con cada actividad.
3. *Búsqueda.* El sistema permite buscar resultados para encontrar elementos concretos.
4. *Botón de creación.* Permite crear una nueva actividad.

Arena								
Usuarios	1	3	Buscar	Crear Actividad				
Comunicaciones								
Deportes								
Actividades								
Registros								
Notificaciones								
1			3			4		
Título			Deporte			Creador		
Quema grasas			Running			lauragonzalez@gmail.com		
28/11/2017 08:30			60			NULL		
Balonmano en Córdoba			Balonmano			victor1995mv@gmail.com		
02/12/2017 10:15			60			6		
Solteros contra casados			Fútbol			jesusrojas@gmail.com		
28/11/2017 12:30			90			12		
Aguas abiertas			Natación			josemanuelruano@gmail.com		
28/11/2017 12:30			45			4		
Triples y tiros libres			Baloncesto			franciscorodriguez@gmail.com		
28/11/2017 12:30			60			NULL		
Tarde de raquetas			Tenis			victor1995mv@gmail.com		
26/11/2017 19:30			120			3		
II Memorial Antonio Ruiz			Fútbol			rubenmedina@gmail.com		
12/10/2017 22:00			90			14		
Bicis, montaña y birras			Ciclismo			joseperezparras@gmail.com		
05/10/2017 09:00			180			NULL		
Partido benéfico			Baloncesto			elenadiaz@gmail.com		
25/09/2017 20:30			60			NULL		
Juego, set y partido			Pádel			omarsotillo@gmail.com		
20/09/2017 20:30			60			2		
Juego, set y partido			Pádel			omarsotillo@gmail.com		
20/09/2017 20:30			60			1		

Figura 13.11 Interfaz de listado de actividades

13.1.12 Interfaz de creación de actividades

La interfaz de creación de actividades se compone de los elementos de la "Figura 13.12":

1. *Formulario.* Se podrán establecer el título, el deporte, el propietario, la fecha y hora de comienzo, la duración, el lugar, las plazas y la descripción.
2. *Botón de guardado y 3. Botón para volver.*

Arena	
Usuarios	1
Comunicaciones	
Deportes	
Actividades	
Registros	
Notificaciones	
1	
Información básica	
Título *	
Deporte *	Fútbol
Creador *	admin@admin.com
Comienzo *	1 ene 2012 00:00
Duración *	En minutos
2	
3	
<input type="button" value="Guardar cambios"/> <input type="button" value="Volver al listado"/>	

Figura 13.12 Interfaz de creación de actividades

13.1.13 Interfaz de edición de actividades

La interfaz de edición de actividades está compuesta por una serie de elementos mostrados en la "Figura 13.13":

1. *Formulario.* Se podrán modificar el título, el deporte, el propietario, la fecha y hora de comienzo, la duración, el lugar, las plazas y la descripción.
2. *Botón de guardado.* Permite guardar los cambios.
3. *Botón de borrado.* Permite borrar la actividad.
4. *Botón para volver.* Permite volver al listado de actividades.

Figura 13.13 Interfaz de edición de actividades

13.1.14 Interfaz de borrado de actividades

La interfaz de borrado de actividades está compuesta por una serie de elementos mostrados en la "Figura 13.14":

1. *Mensaje de confirmación.*
2. *Botón de cancelación.* Permite cancelar el borrado.
3. *Botón de confirmación.* Permite confirmar el borrado.

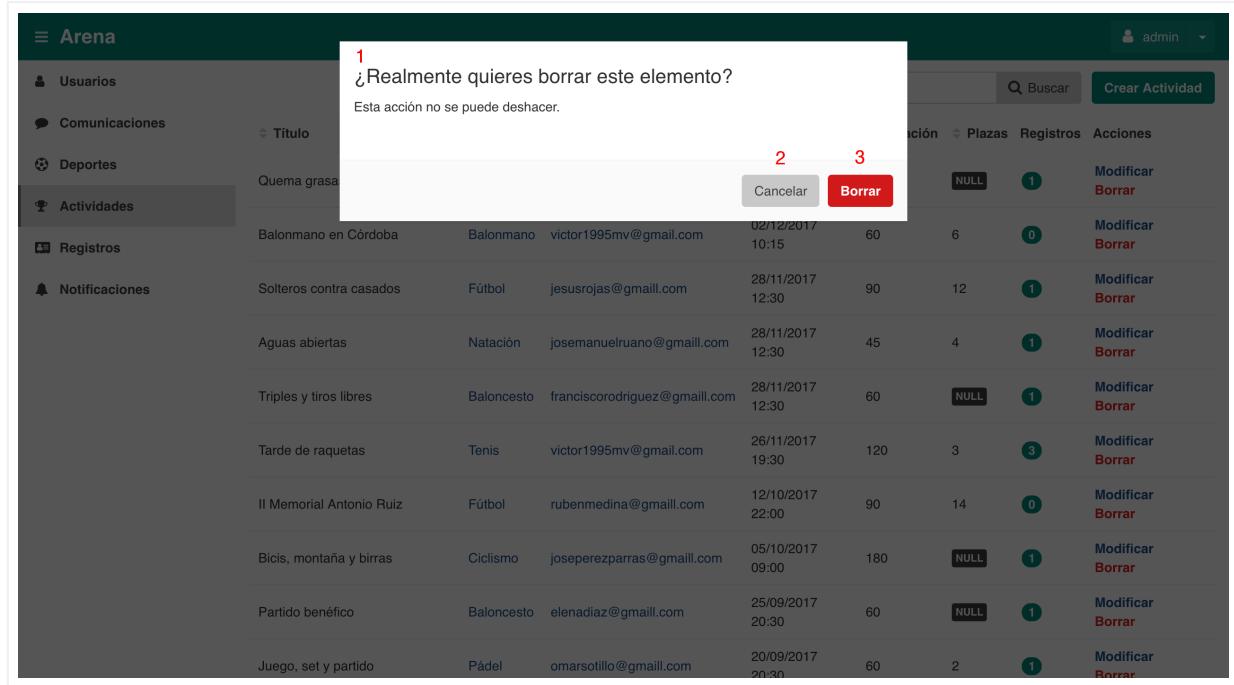


Figura 13.14 Interfaz de borrado de actividades

13.1.15 Interfaz de listado de registros

La interfaz de listado de registros se compone de los elementos de la "Figura 13.15":

1. *Tabla de usuarios.* Cada fila muestra la información de un registro y la tabla posibilita la ordenación de las filas por algún atributo.
2. *Acciones, 3. Búsqueda y 4. Botón de creación.*

La captura de pantalla muestra la interfaz de usuario de la aplicación 'Arena'. En la parte superior, hay un menú con iconos para Usuarios, Comunicaciones, Deportes, Actividades, Registros y Notificaciones. El usuario actual es 'admin'. En la barra superior derecha, hay un botón 'Crear Registro' y un buscador.

En la lista principal, se muestran los siguientes registros:

Usuario	Actividad	Tipo	Estado	Acciones
elenadiaz@gmail.com	Tarde de raquetas	invitation	waiting	Modificar Borrar
ivanportillo@gmail.com	Tarde de raquetas	application	waiting	Modificar Borrar
lauragonzalez@gmail.com	Tarde de raquetas	application	accepted	Modificar Borrar
victor1995mv@gmail.com	Partido benéfico	application	accepted	Modificar Borrar
victor1995mv@gmail.com	Quema grasas	invitation	waiting	Modificar Borrar
victor1995mv@gmail.com	Solteros contra casados	application	waiting	Modificar Borrar
victor1995mv@gmail.com	Aguas abiertas	invitation	waiting	Modificar Borrar
victor1995mv@gmail.com	Triples y tiros libres	application	accepted	Modificar Borrar
victor1995mv@gmail.com	Carrera solidaria	application	waiting	Modificar Borrar
victor1995mv@gmail.com	Bicis, montaña y birras	invitation	accepted	Modificar Borrar
victor1995mv@gmail.com	Juego, set y partido	invitation	waiting	Modificar Borrar

Un diálogo emergente pregunta: '¿Realmente quieres crear este registro?' (Número 1). En el diálogo, hay un botón 'Cancelar' (Número 2) y un botón 'Crear' (Número 3). En la parte superior derecha, hay un botón 'Crear Registro' (Número 4).

Figura 13.15 Interfaz de listado de registros

13.1.16 Interfaz de creación de registros

La interfaz de creación de registros está compuesta por una serie de elementos mostrados en la "Figura 13.16":

1. *Formulario.* Se podrán establecer el usuario, la actividad, el tipo, y el estado.
2. *Botón de guardado.* Permite guardar los cambios.
3. *Botón para volver.* Permite volver al listado de registros.

Figura 13.16 Interfaz de creación de registros

13.1.17 Interfaz de edición de registros

La interfaz de edición de registros está compuesta por una serie de elementos mostrados en la "Figura 13.17":

1. *Formulario.* Se podrán modificar el usuario, la actividad, el tipo, y el estado.
2. *Botón de guardado.* Permite guardar los cambios.
3. *Botón de borrado.* Permite borrar el registro.
4. *Botón para volver.* Permite volver al listado de registros.

The screenshot shows the 'Arena' application interface. On the left is a sidebar with navigation links: Usuarios, Comunicaciones, Deportes, Actividades, Registros (which is selected and highlighted in grey), and Notificaciones. The main content area has the following fields:

- 1 Usuario ***: A dropdown menu containing "elenadiaz@gmail.com".
- Actividad ***: A dropdown menu containing "Tarde de raquetas".
- Tipo ***: A dropdown menu containing "Invitación".
- Estado ***: A dropdown menu containing "En espera".

At the bottom of the form are three buttons: **Guardar cambios** (green), **Borrar** (grey), and **Volver al listado** (blue). Below these buttons are red numbers 2, 3, and 4.

Figura 13.17 Interfaz de edición de registros

13.1.18 Interfaz de borrado de registros

La interfaz de borrado de registros se compone de los elementos de la "Figura 13.18":

1. *Mensaje de confirmación*.
2. *Botón de cancelación*. Permite cancelar el borrado.
3. *Botón de confirmación*. Permite confirmar el borrado.

The screenshot shows the 'Arena' application interface. On the left is a sidebar with navigation links: Usuarios, Comunicaciones, Deportes, Actividades, Registros (selected), and Notificaciones. The main content area displays a table of registered activities. A modal dialog box is open in the center:

- 1**: The title of the dialog is "¿Realmente quieres borrar este elemento?". Below it is the message "Esta acción no se puede deshacer."
- 2**: A grey button labeled "Cancelar".
- 3**: A red button labeled "Borrar".

Below the dialog, the table lists registered activities with columns: Estado, Acciones, Usuario, Actividad, Tipo, and Estado. The "Acciones" column contains "Modificar" and "Borrar" buttons for each row.

Figura 13.18 Interfaz de borrado de registros

13.1.19 Interfaz de listado de notificaciones

La interfaz de listado de notificaciones está compuesta por una serie de elementos mostrados en la "Figura 13.19":

1. *Tabla de usuarios.* Cada fila muestra la información de una notificación y la tabla posibilita la ordenación de las filas por algún atributo.
2. *Búsqueda.* El sistema permite buscar resultados para encontrar elementos concretos.

The screenshot shows the 'Arena' application interface. On the left is a sidebar with navigation links: Usuarios, Comunicaciones, Deportes, Actividades, Registros, and Notificaciones (which is highlighted). The main content area has a header with a user icon labeled 'admin'. Below the header is a search bar with a magnifying glass icon and the word 'Buscar'. A red box highlights the search bar. The main table has the following data:

Usuario	Actividad	Tipo	Envío
victor1995mv@gmail.com	Partido benéfico	application	31/07/2017 16:03
victor1995mv@gmail.com	Quema grasas	invitation	13/08/2017 20:21
victor1995mv@gmail.com	Solteros contra casados	application	03/08/2017 05:13
victor1995mv@gmail.com	Aguas abiertas	invitation	21/07/2017 16:32
victor1995mv@gmail.com	Triples y tiros libres	acceptance	16/08/2017 17:10
victor1995mv@gmail.com	Triples y tiros libres	application	08/08/2017 12:23
victor1995mv@gmail.com	Carrera solidaria	application	30/07/2017 18:04
victor1995mv@gmail.com	Bicis, montaña y birras	acceptance	31/07/2017 10:21
victor1995mv@gmail.com	Bicis, montaña y birras	invitation	27/07/2017 19:58
victor1995mv@gmail.com	Juego, set y partido	invitation	22/07/2017 03:56

Figura 13.19 Interfaz de listado de notificaciones

13.2 Interfaz de usuario

13.2.1 Elementos genéricos

La interfaz de usuario está compuesta por una serie de elementos genéricos mostrados en la "Figura 13.20":

1. *Barra de navegación.* Permite mostrar y ocultar la barra lateral, iniciar y cerrar sesión o volver a la página principal.
2. *Barra lateral.* Permite acceder rápidamente a la mayoría de interfaces de la aplicación. Sólo estará disponible una vez el usuario esté identificado.
3. *Avatar.* Permite cerrar sesión. Sólo estará disponible una vez el usuario esté identificado.
4. *Pie de página.*

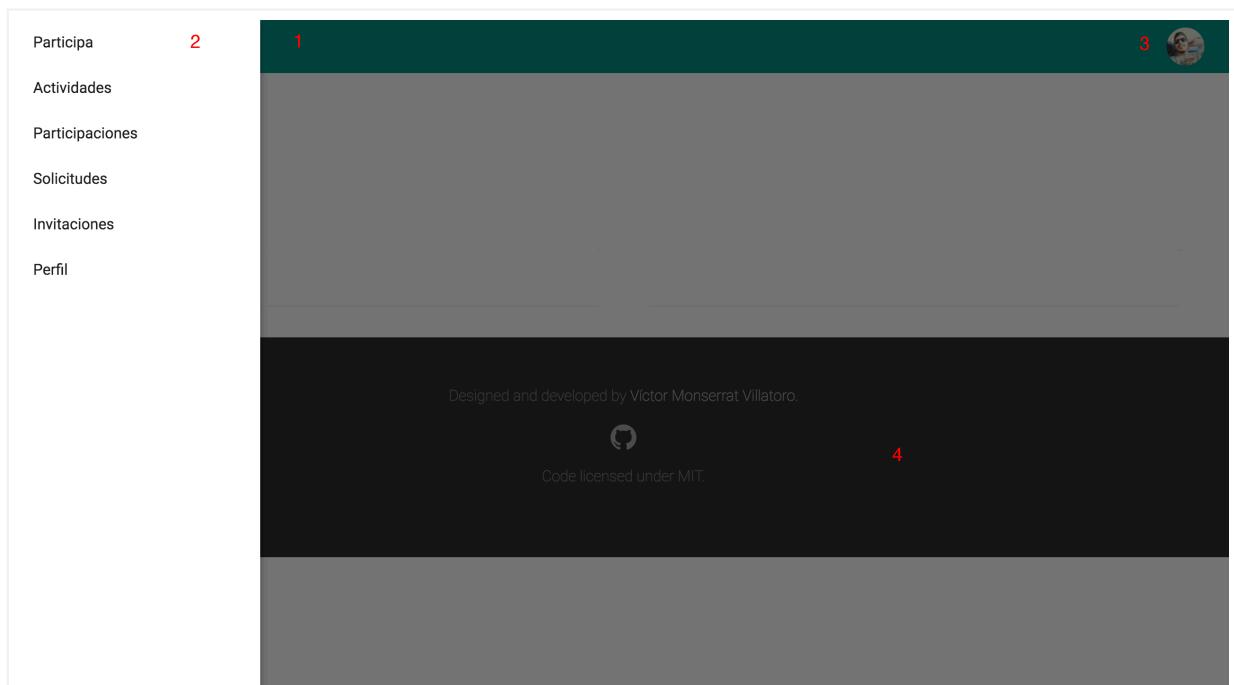


Figura 13.20 Elementos genéricos de la interfaz de usuario

13.2.2 Interfaz principal

La interfaz principal está compuesta por una serie de elementos mostrados en la "Figura 13.21":

1. *Lista de actividades.*
2. *Actividad.*

1

2

3

INICIAR SESIÓN

Actividad	Descripción	Fecha/Hora	Plazas	Opciones
II Memorial Antonio Ruiz.	Rubén Medina Zamorano.	12/10/2017 a las 22:00 90 min.	14 plazas	INICIAR SESIÓN PARA PODER PARTICIPAR
Fútbol.	Colegio La Salle, Córdoba.			
Un poco de basket antes del viaje.	Iván Portillo Leal.	16/09/2017 a las 21:00 40 min.	5 plazas	INICIAR SESIÓN PARA PODER PARTICIPAR
Baloncesto.	Rabanales, Córdoba.			
Partido benéfico.	Elena Díaz Crespo.	25/09/2017 a las 20:30 60 min.	12 plazas	INICIAR SESIÓN PARA PODER PARTICIPAR
Partido benéfico.	Elena Díaz Crespo.	25/09/2017 a las 20:30 60 min.	12 plazas	INICIAR SESIÓN PARA PODER PARTICIPAR
Partido benéfico.	Elena Díaz Crespo.	25/09/2017 a las 20:30 60 min.	12 plazas	INICIAR SESIÓN PARA PODER PARTICIPAR
Triples y tiros libres.	Francisco José Rodríguez Ramírez.	28/11/2017 a las 12:30 60 min.	2 plazas	INICIAR SESIÓN PARA PODER PARTICIPAR
Tarde de raquetas.	Víctor Monserrat Villatoro.	26/11/2017 a las 19:30 120 min.	2 plazas	INICIAR SESIÓN PARA PODER PARTICIPAR
Tenis.	Club Figueroa, Córdoba.			

Figura 13.21 Interfaz principal

13.2.3 Interfaz de actividades

La interfaz de actividades se compone de los elementos de la "Figura 13.22":

1. *Botón de creación.* Permite crear una nueva actividad.
2. *Lista de actividades.*
3. *Actividad.*

1 CREAR UNA NUEVA ACTIVIDAD

2

3

EDITAR

ELIMINAR

Designed and developed by Víctor Monserrat Villatoro.

Code licensed under MIT.

Figura 13.22 Interfaz de actividades

13.2.4 Interfaz de participaciones

La interfaz de participaciones está compuesta por una serie de elementos mostrados en la "Figura 13.23":

1. *Lista de actividades.*
2. *Actividad.*

The screenshot shows a mobile application interface titled 'Arena'. At the top, there is a navigation bar with a menu icon and a user profile picture. Below the navigation bar, there are three cards representing different activities:

- Card 1 (Top Left):** 'Bicis, montaña y birras.' by José Pérez-Parras Toledano. It includes a small profile picture, the activity name, the organizer's name, and a date/time detail '05/10/2017 a las 09:00 180 min.'. A red number '1' is positioned above the card. At the bottom, there is a red button labeled 'ELIMINAR MI PARTICIPACIÓN'.
- Card 2 (Top Right):** 'Triples y tiros libres.' by Francisco José Rodríguez Ramírez. It includes a small profile picture, the activity name, the organizer's name, and a date/time detail '28/11/2017 a las 12:30 60 min.'. A red number '2' is positioned above the card. At the bottom, there is a red button labeled 'ELIMINAR MI PARTICIPACIÓN'.
- Card 3 (Bottom Left):** 'Partido benéfico.' by Elena Díaz Crespo. It includes a small profile picture, the activity name, the organizer's name, and a date/time detail '25/09/2017 a las 20:30 60 min.'. At the bottom, there is a red button labeled 'ELIMINAR MI PARTICIPACIÓN'.

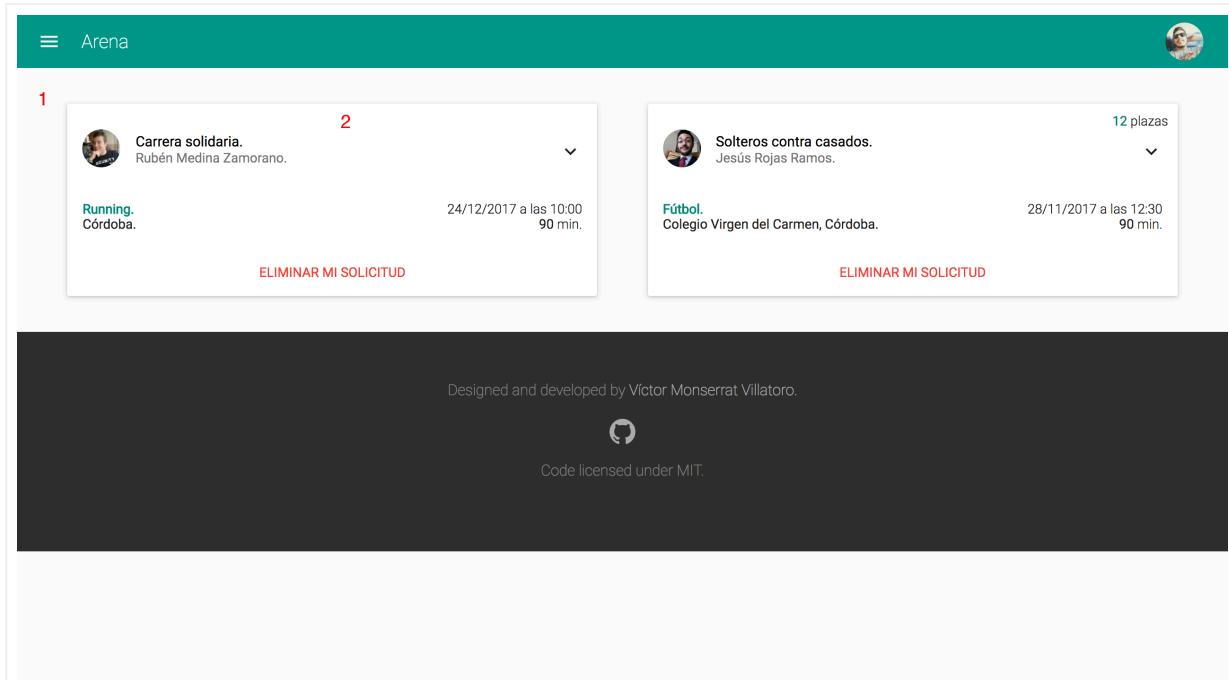
At the bottom of the screen, there is a dark footer bar with white text that reads 'Designed and developed by Víctor Monserrat Villatoro.' and a small circular logo with the letter 'G'.

Figura 13.23 Interfaz de participaciones

13.2.5 Interfaz de solicitudes

La interfaz de solicitudes está compuesta por una serie de elementos mostrados en la "Figura 13.24":

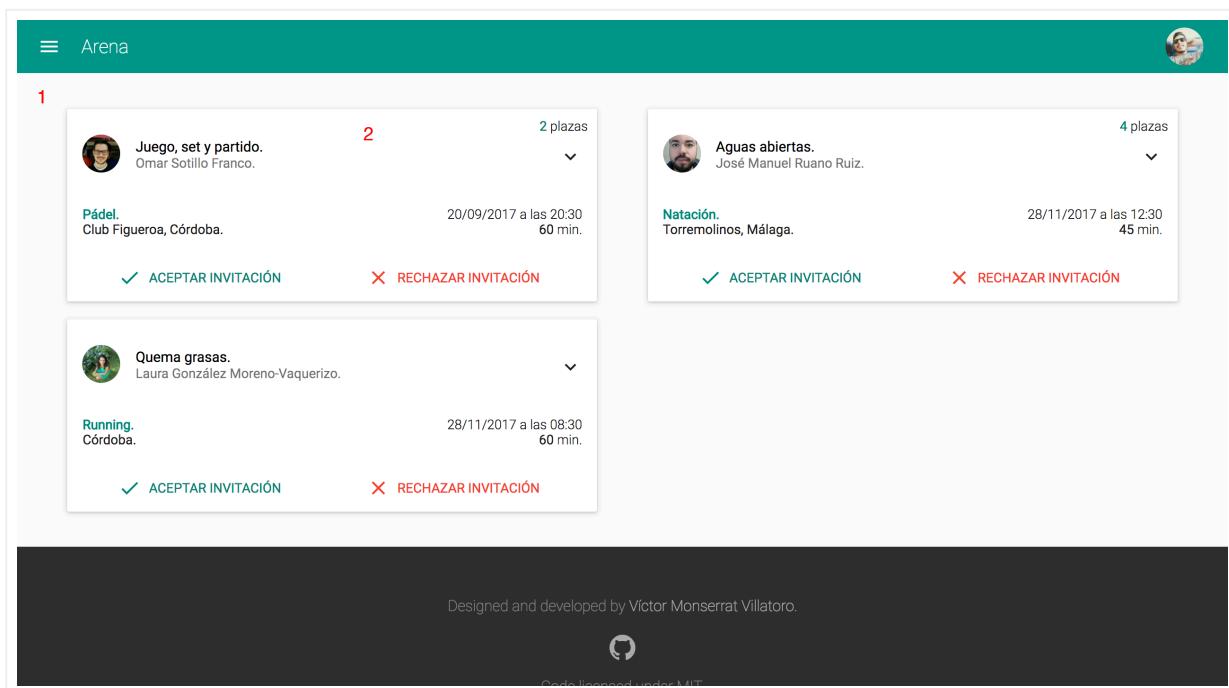
1. *Lista de actividades.*
2. *Actividad.*

**Figura 13.24** Interfaz de solicitudes

13.2.6 Interfaz de invitaciones

La interfaz de invitaciones está compuesta por una serie de elementos mostrados en la "Figura 13.25":

1. *Lista de actividades.*
2. *Actividad.*

**Figura 13.25** Interfaz de invitaciones

13.2.7 Interfaz de perfil de usuario

La interfaz de perfil de usuario está compuesta por una serie de elementos mostrados en la "Figura 13.26":

1. *Avatar.*
2. *Formulario.* Se podrán modificar el nombre y los apellidos.
3. *Botón de guardado.* Permite guardar los cambios.
4. *Botón de borrado.* Permite borrar el perfil de usuario.

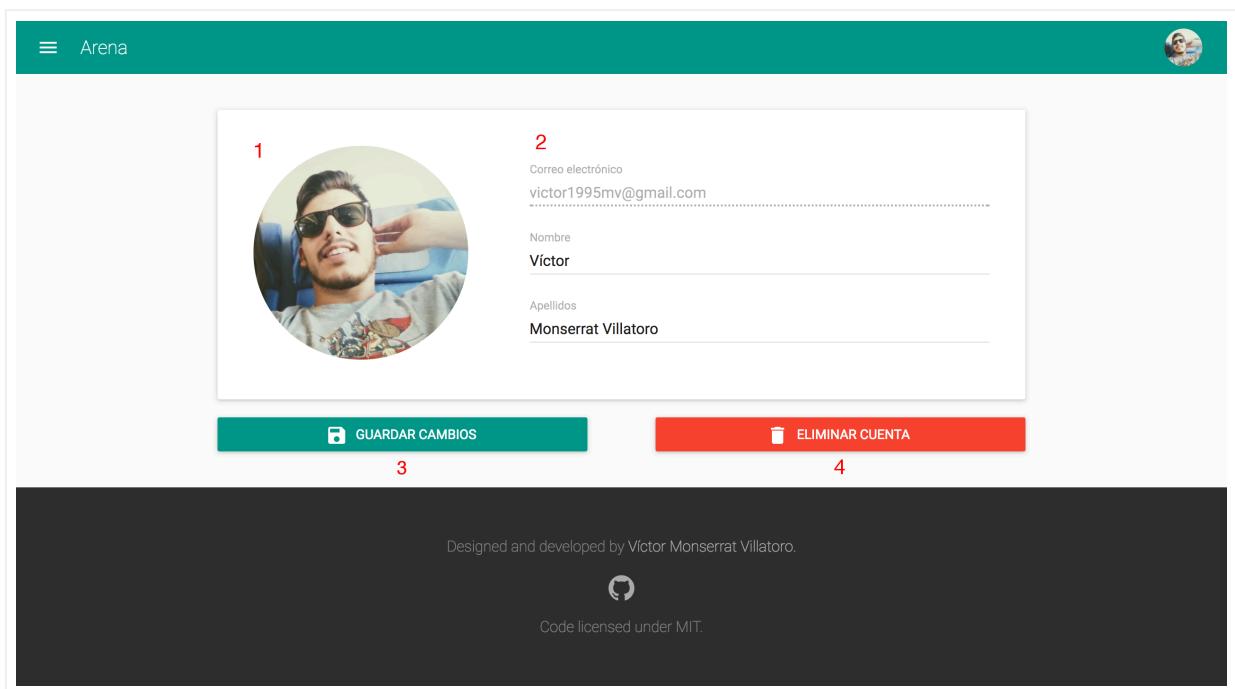


Figura 13.26 Interfaz de perfil de usuario

13.2.8 Interfaz de formulario de actividad

La interfaz de formulario de actividad está compuesta por una serie de elementos mostrados en la "Figura 13.27":

1. *Formulario.* Se podrán establecer o modificar el título, el deporte, el propietario, la fecha y hora de comienzo, la duración, el lugar, las plazas y la descripción.
2. *Botón para continuar.* Habrá uno por cada sección. Se activará cuando los datos sean válidos y el último guardará la actividad o los cambios realizados.
3. *Botón para volver.* Permite volver a la sección anterior.

Figura 13.27 Interfaz de formulario de actividad

13.2.9 Interfaz de actividad

La interfaz de actividad se compone de los elementos mostrados en la "Figura 13.28":

1. *Avatar del propietario.*
2. *Título.*
3. *Nombre del propietario.*
4. *Número de plazas.* Aparecerá cuando no sea ilimitado.
5. *Deporte.*
6. *Lugar.*
7. *Fecha y hora de comienzo.*
8. *Duración.*
9. *Descripción.*
10. *Acciones.* Las acciones que el usuario puede realizar con la actividad.
11. *Lista de participaciones.* Aparecerá si el usuario está involucrado en la actividad.
12. *Lista de solicitudes.* Aparecerá si el usuario es el propietario de la actividad.
13. *Lista de invitaciones.* Aparecerá si el usuario es el propietario de la actividad.
14. *Registro.* Puede ser una participación, solicitud o invitación y podrán realizarse acciones si el usuario es el propietario de la actividad.

15. *Formulario de invitaciones.* Aparecerá si el usuario es el propietario de la actividad.

The screenshot shows a web-based application interface for managing activities. At the top, there's a navigation bar with a menu icon and the word 'Arena'. On the right side of the header is a user profile picture. Below the header, the main content area is divided into several sections:

- Top Activity Details:** Shows a small profile picture (1), the activity name "Tarde de raquetas." (2), the organizer's name "Victor Monserrat Villatoro" (3), a red number '4' indicating 2 plazas available, and a date/time entry "26/11/2017 a las 19:30" (7) with a duration of "120 min." (8).
- Description:** A short description of the activity: "Partido de tenis (dobles) nivel principiante. Iremos cambiando de pareja cada set y luego podemos cenar algo en el bar del club." followed by a red number '9'.
- Buttons:** Below the description are three buttons: a pencil icon labeled "EDITAR" (10), a red trash can icon labeled "ELIMINAR" (11), and a small red number '10'.
- Participaciones:** A list showing one participant, "Laura González Moreno-Vaquerizo" (14), with a small profile picture.
- Solicitudes:** A list showing one request, "Iván Portillo Leal" (12), with a small profile picture.
- Invitaciones:** A list showing one invitation, "Elena Díaz Crespo" (13), with a small profile picture. Below this is a button labeled "Añadir invitaciones" (15).

Figura 13.28 Interfaz de actividad

13.2.10 Interfaz de no encontrado

La interfaz de no encontrado se muestra en la "Figura 13.29":

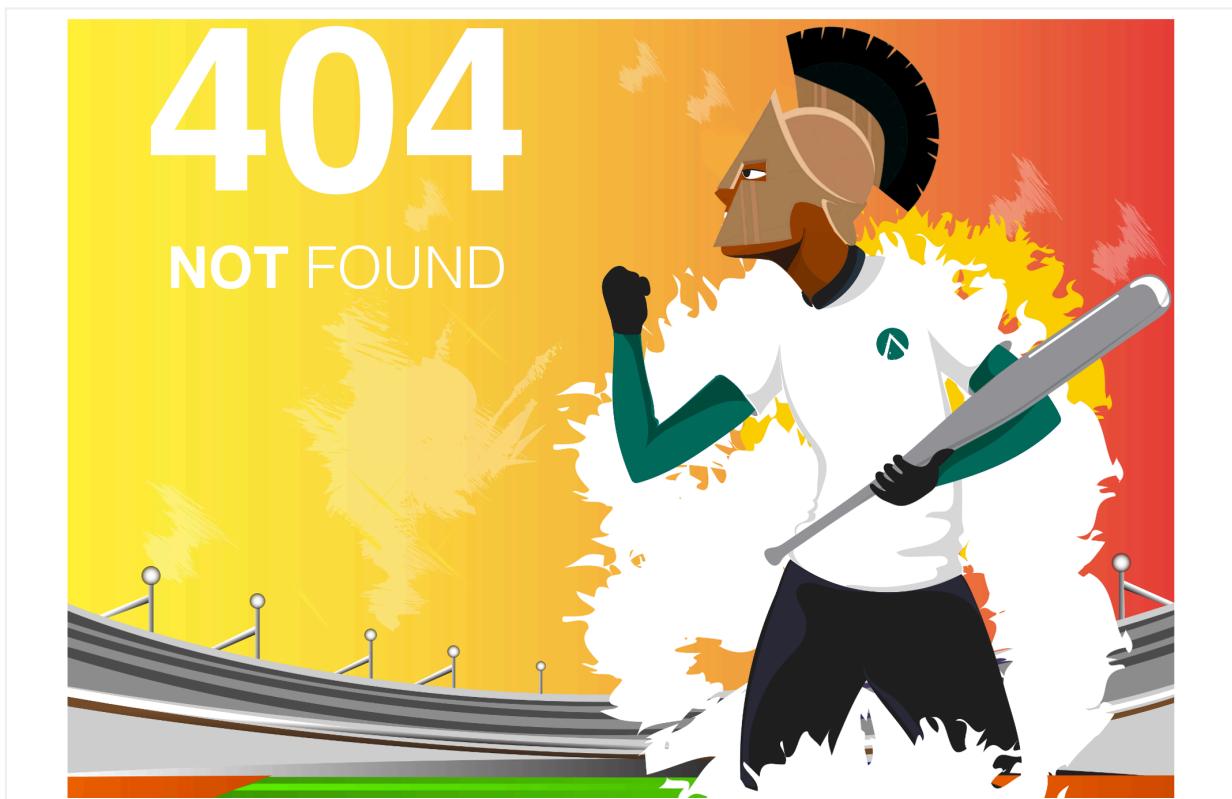


Figura 13.29 Interfaz de no encontrado

Esta página se ha dejado vacía a propósito.

Capítulo 14

Pruebas

En este capítulo se tratarán las pruebas que se han realizado durante la realización del sistema. Destacamos el hecho de que la metodología usada en el proyecto ha sido la denominada como "Desarrollo orientado a comportamiento" o *BDD*^[1]. Eso significa que las pruebas que se presentarán fueron escritas antes de la implementación del código.

Las pruebas se presentarán por el paquete del que forman parte. Se indicará, conforme al formato del lenguaje *Gherkin*^[2], la característica de la prueba, los antecedentes y los escenarios que se deben cumplir.

14.1 Paquete *Users*

14.1.1 Característica *Listar usuarios*

Para ver los usuarios de la aplicación

Como persona interesada en practicar deporte

Quiero poder listarlos

Antecedentes.

Dados los usuarios:

Nombre de usuario
usuario
usuario1
usuario2

[1] *Behavior-Driven Development*: https://es.wikipedia.org/wiki/Desarrollo_guiado_por_comportamiento/

[2] *Gherkin*: <https://github.com/cucumber/cucumber/wiki/Gherkin/>

Escenarios.

Escenario: Listar los usuarios existentes autenticado.

Dado que **usuario** está autenticado

Cuando quiere listar los usuarios de la aplicación

Entonces ve **2** usuarios

Escenario: Buscar un usuario existente autenticado.

Dado que **usuario** está autenticado

Cuando busca el usuario **usuario1**

Entonces ve **1** usuario

Escenario: Buscar un usuario no existente autenticado.

Dado que **usuario** está autenticado

Cuando busca el usuario **usuario3**

Entonces ve **0** usuarios

Escenario: Listar los usuarios existentes sin autenticar.

Cuando alguien quiere listar los usuarios disponibles

Entonces ve **0** usuarios

Escenario: Buscar un usuario existente sin autenticar.

Cuando alguien busca el usuario **usuario1**

Entonces ve **0** usuarios

Escenario: Buscar un usuario no existente sin autenticar.

Cuando alguien busca el usuario **usuario3**

Entonces ve **0** usuarios

14.1.2 Resultados

Tras ejecutar las pruebas estos son los resultados:

```
$ behat features/users/list_users.feature
...
6 scenarios (6 passed)
...
```

14.1.3 Característica *Mostrar usuario*

Para ver un usuario con detalle

Como persona interesada en practicar deporte

Quiero poder mostrarlo

Antecedentes.

Datos los usuarios:

Nombre de usuario
usuario
usuario1

Escenarios.

Escenario: Mostrar un usuario existente autenticado.

Dado que *usuario* está autenticado

Cuando quiere mostrar el usuario *usuario1*

Entonces ve el usuario

Escenario: Mostrar un usuario no existente autenticado.

Dado que *usuario* está autenticado

Cuando quiere mostrar el usuario *usuario2*

Entonces ve que el usuario no ha sido encontrado

Escenario: Mostrar un usuario existente sin autenticar.

Cuando alguien quiere mostrar el usuario *usuario1*

Entonces ve que el usuario no ha sido encontrado

Escenario: Mostrar un usuario no existente sin autenticar.

Cuando alguien quiere mostrar el usuario *usuario2*

Entonces ve que el usuario no ha sido encontrado

14.1.4 Resultados

Tras ejecutar las pruebas estos son los resultados:

```
$ behat features/users/show_user.feature
...
4 scenarios (4 passed)
...
```

14.1.5 Característica *Editar perfil*

Para cambiar los datos de mi perfil

Como usuario de la aplicación

Quiero poder editarlos

Antecedentes.

Dado el usuario *usuario*

Escenarios.

Escenario: Editar mi perfil autenticado.

Dado que *usuario* está autenticado

Cuando edita su perfil de usuario

Entonces su perfil se actualiza con éxito

Escenario: Editar mi perfil sin autenticar.

Cuando alguien quiere editar su perfil

Entonces ve que el usuario debe autenticarse

14.1.6 Resultados

Tras ejecutar las pruebas estos son los resultados:

```
$ behat features/users/update_profile.feature
...
2 scenarios (2 passed)
..
```

14.1.7 Característica *Eliminar perfil*

Para borrar los datos de mi perfil

Como usuario de la aplicación

Quiero poder eliminar mi perfil

Antecedentes.

Dado el usuario *usuario*

Escenarios.

Escenario: Eliminar mi perfil autenticado.

Dado que *usuario* está autenticado

Cuando elimina su perfil de usuario

Entonces su perfil deja de existir

Escenario: Eliminar mi perfil sin autenticar.

Cuando alguien quiere editar su perfil

Entonces ve que el usuario debe autenticarse

14.1.8 Resultados

Tras ejecutar las pruebas estos son los resultados:

```
$ behat features/users/delete_profile.feature
```

```
...
2 scenarios (2 passed)
...
```

14.2 Paquete Sports

14.2.1 Característica *Listar deportes*

Para ver los deportes disponibles

Como persona interesada en practicar deporte

Quiero poder listarlos

Antecedentes.

Dados los deportes:

	Nombre	
	Tenis	
	Golf	

Y el usuario *usuario*

Escenarios.

Escenario: Listar los deportes disponibles autenticado.

Dado que *usuario* está autenticado

Cuando quiere listar los deportes disponibles

Entonces ve *2* deportes

Escenario: Buscar un deporte disponible autenticado.

Dado que *usuario* está autenticado

Cuando busca el deporte *Tenis*

Entonces ve *1* deporte

Escenario: Buscar un deporte no disponible autenticado.

Dado que *usuario* está autenticado

Cuando busca el deporte *Baloncesto*

Entonces ve *0* deportes

Escenario: Listar los deportes disponibles sin autenticar.

Cuando alguien quiere listar los deportes disponibles

Entonces ve *2* deportes

Escenario: Buscar un deporte disponible sin autenticar.

Cuando alguien busca el deporte *Tenis*

Entonces ve *1* deporte

Escenario: Buscar un deporte no disponible sin autenticar.

Cuando alguien busca el deporte *Baloncesto*

Entonces ve *0* deportes

14.2.2 Resultados

Tras ejecutar las pruebas estos son los resultados:

```
$ behat features/sports/list_sports.feature
...
6 scenarios (6 passed)
...
```

14.2.3 Característica *Mostrar deporte*

Para ver un deporte con detalle

Como persona interesada en practicar deporte

Quiero poder mostrarlo

Antecedentes.

Dado el deporte *Tenis*

Y el usuario *usuario*

Escenarios.

Escenario: Mostrar un deporte disponible autenticado.

Dado que *usuario* está autenticado

Cuando quiere mostrar el deporte *Tenis*

Entonces ve el deporte

Escenario: Mostrar un deporte no disponible autenticado.

Dado que *usuario* está autenticado

Cuando quiere mostrar el deporte *Baloncesto*

Entonces ve que el deporte no ha sido encontrado

Escenario: Mostrar un deporte disponible sin autenticar.

Cuando alguien quiere mostrar el deporte *Tenis*

Entonces ve el deporte

Escenario: Mostrar un deporte no disponible sin autenticar.

Cuando alguien quiere mostrar el deporte *Baloncesto*

Entonces ve que el deporte no ha sido encontrado

14.2.4 Resultados

Tras ejecutar las pruebas estos son los resultados:

```
$ behat features/sports/show_sport.feature
...
4 scenarios (4 passed)
...
```

14.3 Paquete *Activities*

14.3.1 Característica *Listar actividades*

Para ver las actividades disponibles

Como persona interesada en practicar deporte

Quiero poder listarlas

Antecedentes.

Datos los deportes:

Nombre
Tenis
Golf

Y los usuarios:

Nombre
usuario
usuario1
usuario2

Y las actividades:

Nombre	Deporte	Propietario	Comienzo
actividad1	Tenis	usuario1	mañana
actividad2	Golf	usuario2	ayer

Escenarios.

Escenario: Listar las actividades disponibles autenticado.

Dado que **usuario** está autenticado

Cuando quiere listar las actividades disponibles

Entonces ve **1** actividad

Escenario: Buscar una actividad disponible autenticado.

Dado que **usuario** está autenticado

Cuando busca la actividad **actividad1**

Entonces ve **1** actividad

Escenario: Buscar una actividad no disponible autenticado.

Dado que **usuario** está autenticado

Cuando busca la actividad **actividad2**

Entonces ve **0** actividades

Escenario: Listar las actividades disponibles sin autenticar.

Cuando alguien quiere listar las actividades disponibles

Entonces ve **1** actividad

Escenario: Buscar una actividad disponible sin autenticar.

Cuando alguien busca la actividad **actividad1**

Entonces ve **1** actividad

Escenario: Buscar una actividad no disponible sin autenticar.

Cuando alguien busca el deporte **actividad2**

Entonces ve **0** actividades

14.3.2 Resultados

Tras ejecutar las pruebas estos son los resultados:

```
$ behat features/activities/list_activities.feature
...
6 scenarios (6 passed)
...
```

14.3.3 Característica *Mostrar actividad*

Para ver una actividad con detalle

Como persona interesada en practicar deporte

Quiero poder mostrarlo

Antecedentes.

Datos los deportes:

Nombre
Tenis
Golf

Y los usuarios:

	Nombre	
	usuario	
	usuario1	
	usuario2	

Y las actividades:

Nombre	Deporte	Propietario	Comienzo
actividad1	Tenis	usuario1	mañana
actividad2	Golf	usuario2	ayer

Escenarios.

Escenario: Mostrar una actividad disponible autenticado.

Dado que *usuario* está autenticado

Cuando quiere mostrar la actividad *actividad1*

Entonces ve la actividad

Escenario: Mostrar una actividad no disponible autenticado.

Dado que *usuario* está autenticado

Cuando quiere mostrar la actividad *actividad2*

Entonces ve que la actividad no ha sido encontrada

Escenario: Mostrar una actividad disponible sin autenticar.

Cuando alguien quiere mostrar la actividad *actividad1*

Entonces ve la actividad

Escenario: Mostrar una actividad no disponible sin autenticar.

Cuando alguien quiere mostrar la actividad *actividad2*

Entonces ve que la actividad no ha sido encontrada

14.3.4 Resultados

Tras ejecutar las pruebas estos son los resultados:

```
$ behat features/activities/show_activity.feature
...
4 scenarios (4 passed)
...
```

14.3.5 Característica *Listar mis actividades*

Para ver mis actividades

Como usuario de la aplicación

Quiero poder listarlas

Antecedentes.

Datos los deportes:

Nombre
Tenis
Golf

Y el usuario **usuario**

Y las actividades:

Nombre	Deporte	Propietario	Comienzo
actividad1	Tenis	usuario	mañana
actividad2	Golf	usuario	ayer

Escenarios.

Escenario: Listar mis actividades autenticado.

Dado que **usuario** está autenticado

Cuando quiere listar sus actividades

Entonces ve **1** actividad

Escenario: Listar mis actividades sin autenticar.

Cuando alguien quiere listar sus actividades

Entonces ve que el usuario debe autenticarse

14.3.6 Resultados

Tras ejecutar las pruebas estos son los resultados:

```
$ behat features/activities/list_my_activities.feature
...
2 scenarios (2 passed)
...
```

14.3.7 Característica *Mostrar mi actividad*

Para ver mi actividad con detalle

Como usuario de la aplicación

Quiero poder mostrarla

Antecedentes.

Datos los deportes:

	Nombre	
	Tenis	
	Golf	

Y el usuario **usuario**

Y las actividades:

Nombre	Deporte	Propietario	Comienzo	
actividad1	Tenis	usuario	mañana	
actividad2	Golf	usuario	ayer	

Escenarios.

Escenario: Mostrar mi actividad disponible autenticado.

Dado que **usuario** está autenticado

Cuando quiere mostrar la actividad **actividad1**

Entonces ve la actividad

Escenario: Mostrar mi actividad no disponible autenticado.

Dado que **usuario** está autenticado

Cuando quiere mostrar la actividad **actividad2**

Entonces ve que la actividad no ha sido encontrada

Escenario: Mostrar mi actividad sin autenticar.

Cuando alguien quiere mostrar su actividad

Entonces ve que el usuario debe autenticarse

14.3.8 Resultados

Tras ejecutar las pruebas estos son los resultados:

```
$ behat features/activities/show_my_activity.feature
...
3 scenarios (3 passed)
...
```

14.3.9 Característica *Editar actividad*

Para cambiar los datos de mi actividad

Como usuario de la aplicación

Quiero poder editarla

Antecedentes.

Dados los deportes:

Nombre
Tenis
Golf

Y los usuarios:

Nombre
usuario
usuario1

Y las actividades:

Nombre	Deporte	Propietario	Comienzo
actividad1	Tenis	usuario	mañana
actividad2	Golf	usuario	ayer
actividad3	Fútbol	usuario1	mañana
actividad4	Surf	usuario1	ayer

Escenarios.

Escenario: Editar mi actividad disponible autenticado.

Dado que *usuario* está autenticado

Cuando edita la actividad *actividad1*

Entonces su actividad se actualiza con éxito

Escenario: Editar mi actividad no disponible autenticado.

Dado que *usuario* está autenticado

Cuando quiere editar la actividad *actividad2*

Entonces ve que la actividad no ha sido encontrada

Escenario: Editar otra actividad disponible autenticado.

Dado que *usuario* está autenticado

Cuando quiere editar la actividad *actividad3*

Entonces ve que no es el propietario de la actividad

Escenario: Editar otra actividad no disponible autenticado.

Dado que *usuario* está autenticado

Cuando quiere editar la actividad *actividad4*

Entonces ve que la actividad no ha sido encontrada

Escenario: Editar actividad sin autenticar.

Cuando alguien quiere editar su actividad

Entonces ve que el usuario debe autenticarse

14.3.10 Resultados

Tras ejecutar las pruebas estos son los resultados:

```
$ behat features/activities/update_activities.feature
...
5 scenarios (5 passed)
...
```

14.3.11 Característica *Eliminar actividad*

Para borrar los datos de mi actividad

Como usuario de la aplicación

Quiero poder eliminarla

Antecedentes.

Datos los deportes:

Nombre
Tenis
Golf

Y los usuarios:

Nombre
usuario
usuario1

Y las actividades:

Nombre	Deporte	Propietario	Comienzo
actividad1	Tenis	usuario	mañana
actividad2	Golf	usuario	ayer
actividad3	Fútbol	usuario1	mañana
actividad4	Surf	usuario1	ayer

Escenarios.

Escenario: Eliminar mi actividad disponible autenticado.

Dado que *usuario* está autenticado

Cuando elimina la actividad *actividad1*

Entonces su actividad deja de existir

Escenario: Eliminar mi actividad no disponible autenticado.

Dado que *usuario* está autenticado

Cuando quiere eliminar la actividad *actividad2*
Entonces ve que la actividad no ha sido encontrada

Escenario: Eliminar otra actividad disponible autenticado.

Dado que *usuario* está autenticado
Cuando quiere eliminar la actividad *actividad3*
Entonces ve que no es el propietario de la actividad

Escenario: Eliminar otra actividad no disponible autenticado.

Dado que *usuario* está autenticado
Cuando quiere eliminar la actividad *actividad4*
Entonces ve que la actividad no ha sido encontrada

Escenario: Eliminar actividad sin autenticar.

Cuando alguien quiere eliminar su actividad
Entonces ve que el usuario debe autenticarse

14.3.12 Resultados

Tras ejecutar las pruebas estos son los resultados:

```
$ behat features/activities/delete_activities.feature
...
5 scenarios (5 passed)
...
```

14.4 Paquete *Registrations*

14.4.1 Característica *Eliminar participante*

Para borrar participantes de una actividad
Como propietario de la actividad
Quiero poder eliminarlos

Antecedentes.

Dado el deporte *Tenis*

Y los usuarios:

Nombre
usuario
usuario1

Y la actividad:

Nombre	Deporte	Propietario	Comienzo	
actividad	Tenis	usuario	mañana	

Y la participación:

Usuario	Actividad	
usuario1	actividad	

Escenarios.

Escenario: Eliminar participante autenticado.

Dado que **usuario** está autenticado

Cuando elimina al participante **usuario1** de la actividad **actividad**

Entonces el **usuario1** deja de participar en la actividad **actividad**

Escenario: Eliminar participante sin autenticar.

Cuando alguien quiere eliminar un participante

Entonces ve que el usuario debe autenticarse

14.4.2 Resultados

Tras ejecutar las pruebas estos son los resultados:

```
$ behat features/registrations/delete_registrations.feature
...
2 scenarios (2 passed)
...
```

14.4.3 Característica *Eliminar mi participación*

Para borrar mi participación de una actividad

Como participante de la actividad

Quiero poder eliminarla

Antecedentes.

Dado el deporte Tenis

Y los usuarios:

Nombre	
usuario	
usuario1	

Y la actividad:

Nombre	Deporte	Propietario	Comienzo	
actividad	Tenis	usuario1	mañana	

Y la participación:

Usuario	Actividad	
usuario	actividad	

Escenarios.

Escenario: Eliminar mi participación autenticado.

Dado que *usuario* está autenticado

Cuando elimina su participación de la actividad *actividad*

Entonces deja de participar en la actividad *actividad*

Escenario: Eliminar mi participación sin autenticar.

Cuando alguien quiere eliminar su participación

Entonces ve que el usuario debe autenticarse

14.4.4 Resultados

Tras ejecutar las pruebas estos son los resultados:

```
$ behat features/registrations/delete_my_registrations.feature
...
2 scenarios (2 passed)
...
```

14.5 Paquete *Applications*

14.5.1 Característica *Listar solicitudes*

Para ver las solicitudes de participación en mis actividades

Como propietario de la actividad

Quiero poder mostrarlas

Antecedentes.

Dado el deporte *Tenis*

Y los usuarios:

Nombre	
usuario	
usuario1	

Y la actividad:

Nombre	Deporte	Propietario	Comienzo	
actividad	Tenis	usuario	mañana	

Y la solicitud:

Usuario	Actividad	
usuario1	actividad	

Escenarios.

Escenario: Listar solicitudes autenticado.

Dado que *usuario* está autenticado

Cuando lista las solicitudes de la actividad *actividad*

Entonces ve *1* solicitud

Escenario: Listar solicitudes sin autenticar.

Cuando alguien quiere listar las solicitudes

Entonces ve que el usuario debe autenticarse

14.5.2 Resultados

Tras ejecutar las pruebas estos son los resultados:

```
$ behat features/applications/list_applications.feature
...
2 scenarios (2 passed)
...
```

14.5.3 Característica *Aceptar solicitudes*

Para tener participantes en mis actividades

Como propietario de la actividad

Quiero poder aceptar solicitudes

Antecedentes.

Dado el deporte *Tenis*

Y los usuarios:

Nombre	
usuario	
usuario1	

Y la actividad:

Nombre	Deporte	Propietario	Comienzo
actividad	Tenis	usuario	mañana

Y la solicitud:

Usuario	Actividad
usuario1	actividad

Escenarios.

Escenario: Aceptar solicitud autenticado.

Dado que *usuario* está autenticado

Cuando acepta la solicitud del usuario *usuario1* en la actividad *actividad*

Entonces el usuario *usuario1* ya participa en la actividad *actividad*

Escenario: Aceptar solicitud sin autenticar.

Cuando alguien quiere aceptar solicitudes

Entonces ve que el usuario debe autenticarse

14.5.4 Resultados

Tras ejecutar las pruebas estos son los resultados:

```
$ behat features/applications/accept_applications.feature
...
2 scenarios (2 passed)
...
```

14.5.5 Característica *Rechazar solicitudes*

Para evitar participantes en mis actividades

Como propietario de la actividad

Quiero poder rechazar solicitudes

Antecedentes.

Dado el deporte *Tenis*

Y los usuarios:

Nombre
usuario
usuario1

Y la actividad:

Nombre Deporte Propietario Comienzo
actividad Tenis usuario mañana

Y la solicitud:

Usuario Actividad
usuario1 actividad

Escenarios.

Escenario: Rechazar solicitud autenticado.

Dado que *usuario* está autenticado

Cuando rechaza la solicitud del usuario *usuario1* en la actividad *actividad*

Entonces el usuario *usuario* no participará en la actividad *actividad*

Escenario: Rechazar solicitud sin autenticar.

Cuando alguien quiere rechazar solicitudes

Entonces ve que el usuario debe autenticarse

14.5.6 Resultados

Tras ejecutar las pruebas estos son los resultados:

```
$ behat features/applications/refuse_applications.feature
...
2 scenarios (2 passed)
...
```

14.5.7 Característica *Listar mis solicitudes*

Para ver mis solicitudes en actividades

Como solicitante

Quiero poder listar mis solicitudes

Antecedentes.

Dado el deporte *Tenis*

Y los usuarios:

Nombre
usuario
usuario1

Y la actividad:

Nombre	Deporte	Propietario	Comienzo
actividad	Tenis	usuario1	mañana

Y la solicitud:

Usuario	Actividad
usuario	actividad

Escenarios.

Escenario: Listar mis solicitudes autenticado.

Dado que *usuario* está autenticado

Cuando lista sus solicitudes

Entonces ve [1](#)

Escenario: Listar mis solicitudes sin autenticar.

Cuando alguien quiere listar sus solicitudes

Entonces ve que el usuario debe autenticarse

14.5.8 Resultados

Tras ejecutar las pruebas estos son los resultados:

```
$ behat features/sports/list_my_applications.feature
...
2 scenarios (2 passed)
...
```

14.5.9 Característica *Cancelar mis solicitudes*

Para eliminar mis solicitudes en actividades

Como solicitante

Quiero poder cancelar mis solicitudes

Antecedentes.

Dado el deporte *Tenis*

Y los usuarios:

Nombre
usuario
usuario1

Y la actividad:

Nombre	Deporte	Propietario	Comienzo
actividad	Tenis	usuario1	mañana

Y la solicitud:

Usuario	Actividad
usuario	actividad

Escenarios.

Escenario: Cancelar mis solicitudes autenticado.

Dado que *usuario* está autenticado

Cuando cancela su solicitud en la actividad *actividad*

Entonces deja de ser solicitante

Escenario: Cancelar mis solicitudes sin autentificar.

Cuando alguien quiere cancelar sus solicitudes

Entonces ve que el usuario debe autentificarse

14.5.10 Resultados

Tras ejecutar las pruebas estos son los resultados:

```
$ behat features/applications/delete_applications.feature
...
2 scenarios (2 passed)
...
```

14.5.11 Característica *Solicitar participación*

Para poder participar en actividades

Como usuario de la aplicación

Quiero poder solicitar mi participación

Antecedentes.

Dado el deporte *Tenis*

Y los usuarios:

Nombre
usuario
usuario1

Y la actividad:

Nombre	Deporte	Propietario	Comienzo
actividad	Tenis	usuario1	mañana

Escenarios.

Escenario: Solicitar participación autenticado.

Dado que **usuario** está autenticado

Cuando solicita su participación en la actividad **actividad**

Entonces es solicitante de la actividad **actividad**

Escenario: Solicitar participación sin autenticar.

Cuando alguien quiere solicitar su participación en una actividad

Entonces ve que el usuario debe autenticarse

14.5.12 Resultados

Tras ejecutar las pruebas estos son los resultados:

```
$ behat features/applications/apply.feature
...
2 scenarios (2 passed)
...
```

14.6 Paquete *Invitations*

14.6.1 Característica *Listar invitaciones*

Para ver las invitaciones de participación en mis actividades

Como propietario de la actividad

Quiero poder mostrarlas

Antecedentes.

Dado el deporte **Tenis**

Y los usuarios:

Nombre
usuario
usuario1

Y la actividad:

Nombre	Deporte	Propietario	Comienzo
actividad	Tenis	usuario1	mañana

actividad Tenis	usuario	mañana	
-------------------	---------	--------	--

Y la invitación:

Usuario Actividad
usuario1 actividad

Escenarios.

Escenario: Listar invitaciones autenticado.

Dado que *usuario* está autenticado

Cuando lista las invitaciones de la actividad *actividad*

Entonces ve *1* invitación

Escenario: Listar invitaciones sin autentificar.

Cuando alguien quiere listar las invitaciones

Entonces ve que el usuario debe autentificarse

14.6.2 Resultados

Tras ejecutar las pruebas estos son los resultados:

```
$ behat features/invitations/list_invitations.feature
...
2 scenarios (2 passed)
...
```

14.6.3 Característica *Aceptar invitaciones*

Para participar en actividades

Como usuario de la aplicación

Quiero poder aceptar invitaciones

Antecedentes.

Dado el deporte Tenis

Y los usuarios:

Nombre
usuario
usuario1

Y la actividad:

Nombre Deporte Propietario Comienzo

actividad Tenis	usuario1	mañana
-------------------	----------	--------

Y la invitación:

Usuario Actividad
usuario actividad

Escenarios.

Escenario: Aceptar solicitud autenticado.

Dado que **usuario** está autenticado

Cuando acepta la invitación en la actividad **actividad**

Entonces ya participa en la actividad **actividad**

Escenario: Aceptar solicitud sin autenticar.

Cuando alguien quiere aceptar invitaciones

Entonces ve que el usuario debe autenticarse

14.6.4 Resultados

Tras ejecutar las pruebas estos son los resultados:

```
$ behat features/invitations/accept_invitations.feature
...
2 scenarios (2 passed)
...
```

14.6.5 Característica *Rechazar invitaciones*

Para evitar participar en actividades

Como usuario de la aplicación

Quiero poder rechazar invitaciones

Antecedentes.

Dado el deporte **Tenis**

Y los usuarios:

Nombre
usuario
usuario1

Y la actividad:

Nombre	Deporte Propietario Comienzo	
--------	----------------------------------	--

actividad Tenis	usuario1	mañana
-------------------	----------	--------

Y la invitación:

Usuario Actividad
usuario actividad

Escenarios.

Escenario: Rechazar invitación autenticado.

Dado que *usuario* está autenticado

Cuando rechaza la invitación en la actividad *actividad*

Entonces no participará en la actividad *actividad*

Escenario: Rechazar invitación sin autentificar.

Cuando alguien quiere rechazar invitaciones

Entonces ve que el usuario debe autentificarse

14.6.6 Resultados

Tras ejecutar las pruebas estos son los resultados:

```
$ behat features/invitations/refuse_invitations.feature
...
2 scenarios (2 passed)
...
```

14.6.7 Característica *Listar mis invitaciones*

Para ver mis invitaciones en actividades

Como invitado

Quiero poder listar mis invitaciones

Antecedentes.

Dado el deporte Tenis

Y los usuarios:

Nombre
usuario
usuario1

Y la actividad:

Nombre Deporte Propietario Comienzo

actividad Tenis	usuario1	mañana	
-------------------	----------	--------	--

Y la invitación:

Usuario Actividad
usuario actividad

Escenarios.

Escenario: Listar mis invitaciones autenticado.

Dado que **usuario** está autenticado

Cuando lista sus invitaciones

Entonces ve [1](#)

Escenario: Listar mis invitaciones sin autenticar.

Cuando alguien quiere listar sus invitaciones

Entonces ve que el usuario debe autenticarse

14.6.8 Resultados

Tras ejecutar las pruebas estos son los resultados:

```
$ behat features/invitations/list_my_invitations.feature
...
2 scenarios (2 passed)
...
```

14.6.9 Característica *Cancelar mis invitaciones*

Para eliminar mis invitaciones en actividades

Como propietario de la actividad

Quiero poder cancelar mis invitaciones

Antecedentes.

Dado el deporte **Tenis**

Y los usuarios:

Nombre
usuario
usuario1

Y la actividad:

Nombre	Deporte Propietario Comienzo	
--------	----------------------------------	--

actividad	Tenis	usuario	mañana	
-----------	-------	---------	--------	--

Y la solicitud:

Usuario	Actividad
usuario1	actividad

Escenarios.

Escenario: Cancelar mis invitaciones autenticado.

Dado que *usuario* está autenticado

Cuando cancela la invitación del usuario *usuario1* en la actividad *actividad*

Entonces deja de estar invitado

Escenario: Cancelar mis invitaciones sin autentificar.

Cuando alguien quiere cancelar sus invitaciones

Entonces ve que el usuario debe autenticarse

14.6.10 Resultados

Tras ejecutar las pruebas estos son los resultados:

```
$ behat features/invitations/delete_invitations.feature
...
2 scenarios (2 passed)
...
```

14.6.11 Característica *Invitar a participar*

Para dejar que participen en mis actividades

Como propietario de la actividad

Quiero poder invitar a participar

Antecedentes.

Dado el deporte Tenis

Y los usuarios:

Nombre
usuario
usuario1

Y la actividad:

Nombre	Deporte	Propietario	Comienzo
--------	---------	-------------	----------

actividad Tenis	usuario	mañana	
-------------------	---------	--------	--

Escenarios.

Escenario: Invitar a participar autenticado.

Dado que *usuario* está autenticado

Cuando invita a participar al usuario *usuario1* en la actividad *actividad*

Entonces el usuario *usuario1* está invitado en la actividad *actividad*

Escenario: Invitar a participar sin autenticar.

Cuando alguien quiere invitar a participar en una actividad

Entonces ve que el usuario debe autenticarse

14.6.12 Resultados

Tras ejecutar las pruebas estos son los resultados:

```
$ behat features/invitations/invite.feature
...
2 scenarios (2 passed)
...
```

Capítulo 15

Conclusiones y futuras mejoras

Una vez finalizadas las etapas de este *Trabajo Fin de Grado* podemos hacer un análisis final del trabajo realizado y comprobar si hemos cumplido con todos los objetivos.

15.1 Análisis de objetivos

15.1.1 Objetivo 1: identificación de usuarios

El sistema final permite la identificación de los usuarios mediante un sistema de identificación federada, en este caso *Google*.

15.1.2 Objetivo 2: gestión de usuarios

El sistema final permite la gestión de usuarios al administrador. Este será capaz de visualizar, modificar o eliminar cualquiera de ellos.

15.1.3 Objetivo 3: gestión de deportes

El sistema final permite la gestión de deportes al administrador. Este podrá visualizar los deportes, inscribir nuevos, modificar los existentes o eliminar cualquiera de ellos.

15.1.4 Objetivo 4: gestión de actividades

El sistema final permite la gestión de actividades a los usuarios. Estos podrán crear nuevas actividades deportivas, así como modificar o eliminar aquellas que han sido creadas por ellos.

15.1.5 Objetivo 5: gestión de notificaciones

El sistema final permite la gestión de notificaciones al administrador. Este podrá enviar

notificaciones a los usuarios de la aplicación.

15.1.6 Objetivo 6: personalización de usuarios

El sistema final permite la personalización de usuarios a estos. Los usuarios podrán definir su propio perfil deportivo en la aplicación.

15.1.7 Objetivo 7: acceso a las actividades

El sistema final permite el acceso a las actividades a los usuarios. Estos podrán buscar actividades en las que tengan interés, inscribirse en ellas y contactar con los participantes de estas.

15.2 Problemas encontrados y soluciones

15.2.1 Almacenamiento del *JWT* en el cliente

El problema era que al recargar la página o abrir una nueva pestaña se pierde la autenticación. Esta se hace a través de un secreto compartido entre *frontend* y *backend*, que al reiniciar, el *frontend* no conserva. Finalmente, la solución ha sido utilizar el *Session Storage*^[1] de los navegadores para poder almacenar el secreto compartido.

15.2.2 Cola de correo electrónico

El envío inmediato de los correos electrónicos desde la aplicación provoca que el usuario espere a que la siguiente página se cargue mientras se envía el correo electrónico. Para evitar este problema, se ha configurado una cola de correo electrónico y es el administrador quien deberá ejecutar la emisión del correo electrónico o configurar un *cron* para que este la ejecute por él cada cierto tiempo.

15.2.3 Depuración de la *API*

Al tratarse de una *API*, la depuración con *Symfony* es bastante compleja. Para facilitar la depuración se ha usado el *Profiler* de *Symfony*.

15.2.4 Documentación de la *API*

Por defecto, *Api-Platform* cuenta con una documentación, generada con *Swagger UI*, aunque esta no da la posibilidad de enviar cabecera en las peticiones. Para permitir la identificación, con el envío del token en la cabecera, finalmente se ha utilizado *APIDoc*.

15.2.5 Configuración de servicios

En *Symfony* para inyectar servicios es necesario un archivo de configuración. Para evitar la creación de una cantidad excesiva de estos archivos, se ha habilitado el *autowiring*.

[1] *Session Storage*: https://www.w3schools.com/html/html5_webstorage.asp

15.2.6 Control de acceso HTTP (CORS)

Para el desarrollo de la aplicación se ejecuta en un servidor local, el *backend* en un puerto y el *frontend* en otro. Esto provoca un problema de seguridad que evitamos deshabilitando este control en el entorno de desarrollo.

15.3 Conclusiones

Podemos comprobar que se han cumplido todos los objetivos que nos propusimos al principio del proyecto y que hemos creado un sistema que, además, ha pasado todas las pruebas del sistema.

15.4 Futuras mejoras

Algunas de las cuestiones que pueden mejorar el sistema son:

- Realizar aplicaciones para teléfonos móviles, aprovechando el uso de la *API*.
- Añadir la identificación de los usuarios mediante otros sistemas de identificación federada, como *Facebook* o *RedIRIS (SIR)*.
- Añadir filtros para la búsqueda de actividades según sus campos principales.
- Añadir la posibilidad de agregar a amigos en la aplicación.
- Añadir la posibilidad de banear a otros usuarios para que no puedan ver las actividades creadas por el usuario.
- Mejorar la personalización del perfil deportivo de los usuarios.
- Realizar un sistema de elaboración de equipos para la organización de actividades.
- Añadir un sistema de pago para la organización de actividades.
- Realizar un sistema de puntuación para los usuarios deportivos y sus actividades.
- Añadir la posibilidad de comentarios en la organización de actividades.
- Añadir un chat para la comunicación entre usuarios de la aplicación.

Esta página se ha dejado vacía a propósito.

Bibliografía

- COI. 2004. *Carta Olímpica*. Disponible en: [http://www.coe.es/web/COEHOME.nsf/b8c1dabf8b650783c1256d560051ba4f/48781e452fd3070cc1256e23005a4454/\\$FILE/Charter_SPA_-_2004.pdf](http://www.coe.es/web/COEHOME.nsf/b8c1dabf8b650783c1256d560051ba4f/48781e452fd3070cc1256e23005a4454/$FILE/Charter_SPA_-_2004.pdf)
- EGUILUZ, J. 2012. *Desarrollo web ágil con Symfony2* [en línea]. Disponible en: <http://symfony.es/libro/>
- FREE SOFTWARE FOUNTATION. 1996. *¿Qué es el software libre?* [en línea]. Free Software Foundation, actualizado el 6 de abril de 2017. [Consulta: 22 de abril de 2017]. Disponible en: <http://www.gnu.org/philosophy/free-sw.html>
- FREE SOFTWARE FOUNDATION. 2012. *¿Varias licencias y comentarios acerca de las mismas?* [en línea]. Free Software Foundation, actualizado el 6 de abril de 2017. [Consulta: 20 de abril de 2017]. Disponible en: <http://www.gnu.org/licenses/license-list.html>
- GÓMEZ, A. 2001. Deporte y moral: los valores educativos del deporte escolar. *Lecturas: educación física y deportes* [en línea], 6 (31), 1-2. [Consulta: 10 enero 2017]. Disponible en: <http://www.efdeportes.com/efd31/valores1.htm>
- IBM. 2008. *Tips for writing good use cases* [en línea]. Disponible en: <ftp://ftp.software.ibm.com/software/rational/web/whitepapers/RW14023-USEN-00.pdf>
- JĘDRZEJEWSKI, P. 2011. *Sylius Documentation* [en línea]. Paweł Jędrzejewski, actualizado en 2017. [Consulta: 1 de mayo de 2017]. Disponible en: <http://sylius.readthedocs.org/en/latest/>
- OMS. 2010. *Recomendaciones mundiales sobre actividad física para la salud*. Suiza: OMS. ISBN 978 92 4 359997 7.
- SONI, R. 2017. *9 things you should know about social login & CRO* [en línea]. Rakish Soni, actualizado el 11 de agosto de 2017. [Consulta: 17 de agosto de 2017]. Disponible en: <https://conversionxl.com/blog/social-login/>

- STEWART, J. 2008. Crow's feer are best. *The Data Administration Newsletter* [en línea], 1-1. [Consulta: 29 de julio de 2017]. Disponible en: <http://www.tdan.com/view-articles/7474/>.
- WIKIPEDIA. 2007. *Symfony*. Wikipedia, actualizado el 10 de marzo de 2017. [Consulta: 21 de abril de 2017]. Disponible en: <http://es.wikipedia.org/wiki/Symfony/>

Apéndices

Esta página se ha dejado vacía a propósito.

Apéndice A1

Competencias

En la "Tabla A1.1" se muestran muchas de las competencias desarrolladas y/o aplicadas en este *Trabajo Fin de Grado*. Muchas de estas competencias fueron adquiridas en las siguientes asignaturas del *Grado de Ingeniería Informática*:

- (101380) *Introducción a la Programación*.
- (101381) *Metodología de la Programación*.
- (101385) *Matemática Discreta*.
- (101390) *Programación y Administración de Sistemas*.
- (101391) *Bases de datos*.
- (101392) *Estructuras de Datos*.
- (101393) *Ingeniería del Software*.
- (101394) *Sistemas de Información*.
- (101396) *Sistemas Operativos*.
- (101399) *Proyectos*.
- (101400) *Programación Web*.
- (101401) *Redes*.
- (101402) *Programación Orientada a Objetos*.
- (101403) *Configuración y Evaluación de Sistemas Informáticos*.
- (101404) *Legislación y Estandarización*.
- (101422) *Algorítmica*.

- (101424) *Procesadores de Lenguajes.*
- (101426) *Sistemas Interactivos.*

Competencia	Descripción
CB1	Que los estudiantes hayan demostrado poseer y comprender conocimientos procedentes de la vanguardia del campo de la Ingeniería Informática.
CB2	Que los estudiantes sepan aplicar sus conocimientos a su trabajo o vocación de una forma profesional y posean las competencias que suelen demostrarse por medio de la elaboración y defensa de argumentos y la resolución de problemas en el campo de la Ingeniería Informática.
CB3	Que los estudiantes tengan la capacidad de reunir e interpretar datos relevantes en el campo de la Ingeniería Informática para emitir juicios que incluyan una reflexión sobre temas relevantes de índole social, científica o ética.
CB4	Que los estudiantes puedan transmitir información, ideas, problemas y soluciones a un público tanto especializado como no especializado.
CB5	Que los estudiantes hayan desarrollado las habilidades de aprendizaje necesarias para emprender estudios posteriores con un alto grado de autonomía.
CU1	Acreditar el uso y dominio de una lengua extranjera.
CU2	Conocer y perfeccionar el nivel de usuario en el ámbito de las TIC.
CEB3	Capacidad para comprender y dominar los conceptos básicos de matemática discreta, lógica, algorítmica y complejidad computacional, y su aplicación para la resolución de problemas propios de la ingeniería.
CEB4	Conocimientos básicos sobre el uso y programación de los ordenadores, sistemas operativos, bases de datos y programas informáticos con aplicación en ingeniería.
CEB5	Conocimiento de la estructura, organización, funcionamiento e interconexión de los sistemas informáticos, los fundamentos de su programación, y su aplicación para la resolución de problemas propios de la ingeniería.
CEB6	Conocimiento adecuado del concepto de empresa, marco institucional y jurídico de la empresa. Organización y gestión de empresas.
CEC1	Capacidad para diseñar, desarrollar, seleccionar y evaluar aplicaciones y sistemas informáticos, asegurando su fiabilidad, seguridad y calidad, conforme a principios éticos y a la legislación y normativa vigente.

Competencia	Descripción
CEC2	Capacidad para planificar, concebir, desplegar y dirigir proyectos, servicios y sistemas informáticos en todos los ámbitos, liderando su puesta en marcha y su mejora continua y valorando su impacto económico y social.
CEC3	Capacidad para comprender la importancia de la negociación, los hábitos de trabajo efectivos, el liderazgo y las habilidades de comunicación en todos los entornos de desarrollo de software.
CEC4	Capacidad para elaborar el pliego de condiciones técnicas de una instalación informática que cumpla los estándares y normativas vigentes.
CEC5	Conocimiento, administración y mantenimiento sistemas, servicios y aplicaciones informáticas.
CEC6	Conocimiento y aplicación de los procedimientos algorítmicos básicos de las tecnologías informáticas para diseñar soluciones a problemas, analizando la idoneidad y complejidad de los algoritmos propuestos.
CEC7	Conocimiento, diseño y utilización de forma eficiente los tipos y estructuras de datos más adecuados a la resolución de un problema.
CEC8	Capacidad para analizar, diseñar, construir y mantener aplicaciones de forma robusta, segura y eficiente, eligiendo el paradigma y los lenguajes de programación más adecuados.
CEC10	Conocimiento de las características, funcionalidades y estructura de los Sistemas Operativos y diseñar e implementar aplicaciones basadas en sus servicios.
CEC11	Conocimiento y aplicación de las características, funcionalidades y estructura de los Sistemas Distribuidos, las Redes de Computadores e Internet y diseñar e implementar aplicaciones basadas en ellas.
CEC12	Conocimiento y aplicación de las características, funcionalidades y estructura de las bases de datos, que permitan su adecuado uso, y el diseño y el análisis e implementación de aplicaciones basadas en ellos.
CEC13	Conocimiento y aplicación de las herramientas necesarias para el almacenamiento, procesamiento y acceso a los Sistemas de información, incluidos los basados en web.
CEC14	Conocimiento y aplicación de los principios fundamentales y técnicas básicas de la programación paralela, concurrente, distribuida y de tiempo real.
CEC16	Conocimiento y aplicación de los principios, metodologías y ciclos de vida de la ingeniería de software.
CEC17	Capacidad para diseñar y evaluar interfaces persona

Competencia	Descripción
	computador que garantice la accesibilidad y usabilidad a los sistemas, servicios y aplicaciones informáticas.
CEC18	Conocimiento de la normativa y la regulación de la informática en los ámbitos nacional, europeo e internacional.
CTEIS1	Capacidad para desarrollar, mantener y evaluar servicios y sistemas software que satisfagan todos los requisitos del usuario y se comporten de forma fiable y eficiente, sean asequibles de desarrollar y mantener y cumplan normas de calidad, aplicando las teorías, principios, métodos y prácticas de la Ingeniería del Software.
CTEIS2	Capacidad para valorar las necesidades del cliente y especificar los requisitos software para satisfacer estas necesidades, reconciliando objetivos en conflicto mediante la búsqueda de compromisos aceptables dentro de las limitaciones derivadas del coste, del tiempo, de la existencia de sistemas ya desarrollados y de las propias organizaciones.
CTEIS3	Capacidad de dar solución a problemas de integración en función de las estrategias, estándares y tecnologías disponibles.
CTEIS4	Capacidad de identificar y analizar problemas y diseñar, desarrollar, implementar, verificar y documentar soluciones software sobre la base de un conocimiento adecuado de las teorías, modelos y técnicas actuales.
CTEIC3	Capacidad de analizar y evaluar arquitecturas de computadores, incluyendo plataformas paralelas y distribuidas, así como desarrollar y optimizar software de para las mismas.
CTEIC4	Capacidad de diseñar e implementar software de sistema y de comunicaciones. Capacidad de diseñar e implementar software de sistema y de comunicaciones.
CTEIC6	Capacidad para comprender, aplicar y gestionar la garantía y seguridad de los sistemas informáticos.
CTEIC7	Capacidad para analizar, evaluar, seleccionar y configurar plataformas hardware para el desarrollo y ejecución de aplicaciones y servicios informáticos.
CTEIC8	Capacidad para diseñar, desplegar, administrar y gestionar redes de computadores.
CTEC1	Capacidad para tener un conocimiento profundo de los principios fundamentales y modelos de la computación y saberlos aplicar para interpretar, seleccionar, valorar, modelar, y crear nuevos conceptos, teorías, usos y desarrollos tecnológicos relacionados con la informática.

Competencia	Descripción
CTEC2	Capacidad para conocer los fundamentos teóricos de los lenguajes de programación y las técnicas de procesamiento léxico, sintáctico y semántico asociadas, y saber aplicarlas para la creación, diseño y procesamiento de lenguajes.
CTEC3	Capacidad para evaluar la complejidad computacional de un problema, conocer estrategias algorítmicas que puedan conducir a su resolución y recomendar, desarrollar e implementar aquella que garantice el mejor rendimiento de acuerdo con los requisitos establecidos.
CTEC5	Capacidad para adquirir, obtener, formalizar y representar el conocimiento humano en una forma computable para la resolución de problemas mediante un sistema informático en cualquier ámbito de aplicación, particularmente los relacionados con aspectos de computación, percepción y actuación en ambientes o entornos inteligentes.
CTEC6	Capacidad para desarrollar y evaluar sistemas interactivos y de presentación de información compleja y su aplicación a la resolución de problemas de diseño de interacción persona computadora.
CTEC7	Capacidad para conocer y desarrollar técnicas de aprendizaje computacional y diseñar e implementar aplicaciones y sistemas que las utilicen, incluyendo las dedicadas a extracción automática de información y conocimiento a partir de grandes volúmenes de datos.
CETFG1	Ejercicio original a realizar individualmente y presentar y defender ante un tribunal universitario, consistente en un proyecto en el ámbito de las tecnologías específicas de la Ingeniería en Informática de naturaleza profesional en el que se sinteticen e integren las competencias adquiridas en las enseñanzas.

Tabla A1.1 Tabla de competencias

Esta página se ha dejado vacía a propósito.

Apéndice A2

Manual del administrador

A2.1 Instalación del *backend*

A2.1.1 Requisitos previos

El *backend* de la aplicación está programado con *Symfony3*, un *framework* desarrollado con *PHP* versión 5.5.9. Concretamente los requisitos de instalación se muestran en la "Tabla A2.1".

Software	Versión mínima	Versión recomendada
PHP (1)(2)	7.1	≥ 7.1
Apache (3)	2.2	2.4
Nginx (3)	1.4	1.4
MySQL (4)	5.0	5.5
Git	1.8	1.9

Tabla A2.1 Tabla de requisitos para la instalación del backend

Notas:

1. *PHP* debe ser instalado con los siguientes módulos extra: *curl*, *intl*, *json*, *mysqli* y *xml*.
2. *PHP* debe tener configurado el uso horario del sistema. Para ello añadir al *php.ini* del sistema lo siguiente: *date.timezone = "Europe/Madrid"*.

3. Solo es necesario uno de los dos servidores web.
4. El sistema soporta otros gestores como *MariaDB*, *Postgresql* u *Oracle*. Para más información, leer la documentación oficial de *Symfony* y *Doctrine*.

A2.1.2 Preparación del sistema

Antes de proceder a instalar el sistema debemos crear la base de datos. Los pasos pueden variar entre sistemas, incluso alguno puede tener herramientas de configuración visuales. Vamos a describir los pasos que se seguirían desde consola por parte del administrador de sistemas, que es la solución más general:

```
$ mysql -u root -p  
Enter password:  
mysql> CREATE DATABASE arena;
```

Además, para la configuración del proyecto necesitaremos el nombre del servidor de la base de datos, el nombre, el usuario y la contraseña. Además, necesitaremos tres claves para la seguridad del sistema, una para el cifrado de datos, otra la pública para *JWT* y la última la privada para *JWT*. Para el envío de correo necesitaremos también un servidor de correo electrónico, el usuario y su contraseña.

A2.1.3 Instalación del software

Lo primero será descomprimir el paquete o descargar la última versión del software del repositorio oficial. Suponiendo el primer caso:

```
$ cd /var/www #directorio donde se instalará el software  
$ tar zxf arena-backend-1.0.0.tar.gz  
$ cd arena-backend
```

Este proyecto tiene una serie de dependencias llamadas *bundles* que se instalan automáticamente a través de un software llamado *Composer*^[1], que es un gestor de paquetes para *PHP*. No viene con la distribución y si no lo está ya previamente, hay que instalarlo. En este caso lo instalaremos en el mismo directorio:

```
$ php -r "readfile('https://getcomposer.org/installer/');" | php  
#!/usr/bin/env php  
All settings correct for using Composer  
Downloading...  
  
Composer successfully installed to: /var/www/arena/composer.phar  
Use it: php composer.phar
```

[1] **Composer:** <https://getcomposer.org/>

Ahora podemos proceder a instalar las dependencias de nuestro proyecto. *Composer*, además, nos hará una serie de preguntas al terminar, con los datos de configuración que obtuvimos en el paso anterior.

```
$ SYMFONY_ENV=prod php composer.phar install --no-dev --optimize-autoloader
Loading composer repositories with package information
Installing dependencies (including require-dev) from lock file
- Installing twig/twig (v1.16.0)
  Loading from cache
# [...]
Loading from cache
Generating optimized autoload files
Creating the "app/config/parameters.yml" file
Some parameters are missing. Please provide them.
database_driver (pdo_mysql):
database_host (127.0.0.1):
database_port (null):
database_name (symfony): arena
database_user (root):
database_password (null): password
mailer_transport (smtp):
mailer_host (127.0.0.1):
mailer_user (null): noreply@arena.com
mailer_password (null): password
locale (en): es
secret (ThisTokenIsNotSoSecretChangeIt): 19454df76be75026a447ab1935ef1385
6ce2c7e4f6a78801ba3ff77e95f05f83c5c2928f67fea81625a080f825a7be87
Clearing the cache for the prod environment with debug false
Installing assets as hard copies
# [...]
```

Los únicos datos que debemos completar son los de nombre de la base de datos, usuario, contraseña, *locale* (idioma) y *secret* (cadena aleatoria), usuario del correo electrónico y contraseña.

Una vez instalado el proyecto, podremos generar las claves pública y privada para *JWT* con un comando que hemos creado en el proyecto con este fin:

```
$ console arena:security:keys password
Public and private keys created successfully.
Remember to configure "jwt_key_pass_phrase" option in your parameters.yml file to: password
```

Tendremos que pasar como argumento la contraseña (*password*) para generar las claves. Una vez generadas introducimos la siguiente configuración en el archivo de configuración:

```
jwt_private_key_path: '%kernel.root_dir%/../var/jwt/private.pem'
jwt_public_key_path: '%kernel.root_dir%/../var/jwt/public.pem'
```

El último paso es la instalación de las plantillas:

```
$ php bin/console assets:install web --env=prod
Installing assets as hard copies.
```

Bundle	Method / Error
✓ ApiPlatformBundle	copy
✓ NelmioApiDocBundle	copy
✓ EasyAdminBundle	copy

```
! [NOTE] Some assets were installed via copy. If you make changes to these assets you have to run this command again.
```

```
[OK] All assets were successfully installed.
```

A2.1.4 Provisión de la base de datos

A continuación, vamos a crear el esquema de la base de datos, crear los datos iniciales y un usuario de administración.

Para crear el esquema:

```
$ php bin/console doctrine:schema:create --env=prod
Creating database schema...
Database schema created successfully!
```

Para crear la configuración inicial:

```
$ php bin/console rad:fixtures:load --env=prod -l es_ES
#Bundle > AppBundle\AppBundle
```

Y finalmente creamos un usuario:

```
$ php bin/console fos:user:create admin admin@arena.com contraseña --super-admin --env=prod
Created user admin
```

A2.1.5 Configuración del sistema de archivos

Lo primero es dar permisos al servidor web para poder escribir en los directorios temporales. Vamos a dar la configuración básica y la avanzada:

Básica

Para un sistema basado en Debian:

```
$ rm -rf app/cache/*
$ rm -rf app/logs/*
$ mkdir web/uploads
$ chown www-data. app/cache app/logs web/uploads
```

Avanzada

Para cualquier sistema GNU/Linux con soporte de ACL en el sistema de archivos:

```
$ rm -rf app/cache/*
$ rm -rf app/logs/*
$ mkdir web/uploads
$ HTTPDUSER=`ps aux | grep -E '[a]pache|[h]ttpd|[_]www|[w]ww-data|[n]ginx` | grep -v root | head -1 | cut -d\` -f1`
$ sudo setfacl -R -m u:"$HTTPDUSER":rwX -m u:`whoami`:rwX app/cache app/logs web/uploads
$ sudo setfacl -dR -m u:"$HTTPDUSER":rwX -m u:`whoami`:rwX app/cache app/logs web/uploads
```

A2.1.6 Configuración del servidor web

La configuración dependerá del servidor elegido y del directorio de instalación. Lo normal es que tenga que consultar la documentación de su servidor, pero como referencia indicamos la configuración de la instalación clásica, que es el software instalado en la raíz de un dominio:

Configuración para un servidor *apache2* con *mod_php*

Configuración para el host virtual donde se aloje la web:

```
<VirtualHost *:80>
    ServerName www.api.arena.es
    ServerAlias api.arena.es

    DocumentRoot /var/www/apiarena/web
    <Directory /var/www/apiarena/web>
        # enable the .htaccess rewrites
        AllowOverride All
        Order allow,deny
        Allow from All
```

```

</Directory>

ErrorLog /var/log/apache2/arena_error.log
CustomLog /var/log/apache2/arena_access.log combined
</VirtualHost>

```

Si se usa Apache2.4, la sección *Directory* debe cambiarse por lo siguiente:

```

<Directory /var/www/apiarena/web>
    # enable the .htaccess rewrites
    AllowOverride All
    Require all granted
</Directory>

```

Configuración para un servidor *nginx*

De nuevo, para configuraciones avanzadas remitimos a la documentación oficial. La configuración mínima del sistema es la siguiente:

```

server {
    server_name api.arena.es www.api.arena.es;
    root /var/www/apiarena/web;

    location / {
        # try to serve file directly, fallback to app.php
        try_files $uri /app.php$is_args$args;
    }

    location ~ ^/(app|app_dev|config)\.php(/|$) {
        fastcgi_pass unix:/var/run/php5-fpm.sock;
        fastcgi_split_path_info ^(.+\.php)(/.*)$;
        include fastcgi_params;
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
        fastcgi_param HTTPS off;
    }

    error_log /var/log/nginx/apiarena_error.log;
    access_log /var/log/nginx/apiarena_access.log;
}

```

A2.1.7 Comprobación de la instalación

Para comprobar que la instalación se ha realizado correctamente accederemos a la siguiente página desde el servidor donde está instalada la aplicación:

<http://localhost/config.php/>

Nos mostrará una página web indicando si el sistema está correctamente configurado y, en caso contrario, las indicaciones necesarias para arreglarlo como en la "Figura A2.1".

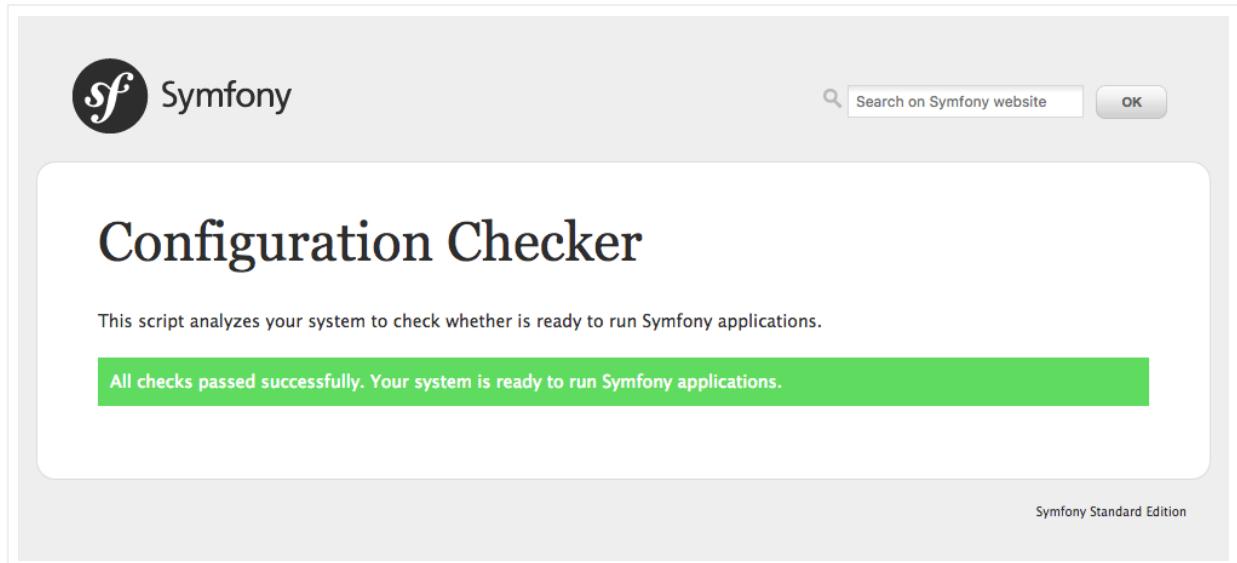


Figura A2.1 Comprobación de configuración

A2.1.8 Administración de correo electrónico

El envío inmediato de los correos electrónicos desde la aplicación podría provocar que el usuario espere a que la siguiente página se cargue mientras se envía el correo electrónico. Esto se puede evitar eligiendo "enviar" los correos electrónicos en lugar de enviarlos directamente. Esto significa que no intentaremos enviar el correo electrónico, sino que guardaremos el mensaje como un archivo. Otro proceso puede leer la cola de correos electrónicos y enviarlos. Para evitar este tipo de problema, se ha configurado una cola de correo electrónico y es el administrador quien deberá ejecutar la emisión del correo electrónico o configurar un *cron* para que este la ejecute por él cada cierto tiempo a través del siguiente comando:

```
$ php bin/console swiftmailer:spool:send
```

A2.2 Instalación del *frontend*

A2.2.1 Requisitos previos

El *frontend* de la aplicación está programado con *React*, una librería desarrollada con *Javascript*. Concretamente los requisitos previos de instalación se mostrarán en la "Tabla A2.2".

Software	Versión mínima	Versión recomendada
NodeJS	6.0.0	$\geq 6.5.0$
Nginx	1.4	1.4

Software	Versión mínima	Versión recomendada
Git	1.8	1.9

Tabla A2.2 Tabla de requisitos para la instalación del frontend

A2.2.2 Instalación del software

Lo primero será descomprimir el paquete o descargar la última versión del software del repositorio oficial. Suponiendo el primer caso:

```
$ cd /var/www #directorio donde se instalará el software
$ tar zxf arena-frontend-1.0.0.tar.gz
$ cd arena-frontend
```

Este proyecto tiene una serie de dependencias que se instalan automáticamente a través de un software llamado *npm*^[2] o *yarn*^[3], que son gestores de paquetes para *NodeJS*. No vienen con la distribución y si no lo están ya previamente, hay que instalarlos.

- npm:

```
npm install npm@latest -g
```

- yarn:

```
curl -sS https://dl.yarnpkg.com/debian/pubkey.gpg | sudo apt-key add -
echo "deb https://dl.yarnpkg.com/debian/ stable main" | sudo tee /etc/apt/sources.list.d/yarn.list
sudo apt-get update && sudo apt-get install yarn
```

Ahora podemos proceder a instalar las dependencias de nuestro proyecto. *Composer*, además, nos hará una serie de preguntas al terminar, con los datos de configuración que obtuvimos en el paso anterior.

- npm:

```
npm install
```

- yarn:

```
yarn install
```

[2] **npm**: <https://www.npmjs.com/>

[3] **yarn**: <https://yarnpkg.com/>

Configuración para un servidor *apache2 con mod_php*

Configuración para el host virtual donde se aloje la web:

```
<VirtualHost *:80>
    ServerName www.arena.es
    ServerAlias arena.es

    DocumentRoot /var/www/arena/web
    <Directory /var/www/arena/web>
        # enable the .htaccess rewrites
        AllowOverride All
        Order allow,deny
        Allow from All
    </Directory>

    ErrorLog /var/log/apache2/arena_error.log
    CustomLog /var/log/apache2/arena_access.log combined
</VirtualHost>
```

Si se usa Apache2.4, la sección *Directory* debe cambiarse por lo siguiente:

```
<Directory /var/www/arena/web>
    # enable the .htaccess rewrites
    AllowOverride All
    Require all granted
</Directory>
```

Configuración para un servidor *nginx*

De nuevo, para configuraciones avanzadas remitimos a la documentación oficial. La configuración mínima del sistema es la siguiente:

```
server {
    server_name arena.es www.arena.es;
    root /var/www/arena/web;

    error_log /var/log/nginx/arena_error.log;
    access_log /var/log/nginx/arena_access.log;
}
```

A2.2.3 Comprobación de la instalación

Para comprobar que la instalación se ha realizado correctamente accederemos a la siguiente página desde el servidor donde está instalada la aplicación:

<http://localhost/>

Nos mostrará la aplicación web si el sistema está correctamente configurado y, en caso contrario, las indicaciones necesarias para arreglarlo.

A2.3 Desinstalación

La desinstalación es un proceso muy sencillo, que básicamente se reduce a borrar los archivos creados y las configuraciones que indicamos en la sección anterior.

A2.3.1 Desinstalación del *backend*

Los pasos son los siguientes:

1. Borrar la configuración del servidor web.
2. Borrar la base de datos.
3. Borrar el directorio de instalación.

A2.3.2 Desinstalación del *frontend*

Los pasos son los siguientes:

1. Borrar la configuración del servidor web.
2. Borrar el directorio de instalación.

Apéndice A3

Manual de usuario

A3.1 Administración

Para el acceso a la interfaz de administración es necesaria la autenticación del usuario. Si el usuario que intenta acceder, a través de "/admin", no está identificado o no es usuario administrador, el sistema redirigirá al usuario a "/admin/login". Aquí, el usuario tendrá que identificarse con credenciales de usuario administrador válidas.

A3.1.1 Panel de administración

El panel de administración está compuesto por una serie de elementos mostrados en la "Figura A3.1":

1. *Barra lateral.* Permite acceder rápidamente a la sección de listado de cualquier elemento desde cualquier lugar.
2. *Botón de despliegue.* Este botón permite mostrar y ocultar la barra lateral para permitir más o menos espacio a la sección principal. Está pensando para monitores con poca resolución.
3. *Botón de salida.* Este botón permite al administrador cerrar su sesión y salir de la aplicación.



Figura A3.1 Panel de administración

A3.1.2 Listado de elementos

El listado está compuesto por una serie de elementos mostrados en la "Figura A3.2":

1. *Tabla de elementos.* El administrador puede ordenar las filas ascendente o descendente seg\xfan alg\xfqn atributo.
2. *Acciones.* El administrador puede modificar o borrar cualquier elemento de la lista.
3. *B\xfbsqueda.* El administrador podr\xe1 buscar resultados para encontrar elementos concretos.

Usuarios		Nombre completo	Correo electrónico	Actividades	Registros	Habilidades	Acciones
1	Elena Diaz Crespo	Elena Diaz Crespo	elenadiaz@gmail.com	1	1	Si	Modificar Borrar
2	Laura González Moreno-Vaquerizo	Laura González Moreno-Vaquerizo	lauragonzalez@gmail.com	1	1	Si	Modificar Borrar
3	Jesús Rojas Ramos	Jesús Rojas Ramos	jesusrojas@gmail.com	1	0	Si	Modificar Borrar
4	José Manuel Ruano Ruiz	José Manuel Ruano Ruiz	josemanuelruano@gmail.com	1	0	Si	Modificar Borrar
5	Francisco José Rodríguez Ramírez	Francisco José Rodríguez Ramírez	franciscorodriguez@gmail.com	1	0	Si	Modificar Borrar
6	Rubén Medina Zamorano	Rubén Medina Zamorano	rubenmedina@gmail.com	2	0	Si	Modificar Borrar
7	José Pérez-Parras Toledo	José Pérez-Parras Toledo	joseperezparras@gmail.com	1	0	Si	Modificar Borrar
8	Omar Sotillo Franco	Omar Sotillo Franco	omarsotillo@gmail.com	1	0	Si	Modificar Borrar
9	Iván Portillo Leal	Iván Portillo Leal	ivanportillo@gmail.com	1	1	Si	Modificar Borrar
10	Víctor Monserrat Villatoro	Víctor Monserrat Villatoro	victor1995mv@gmail.com	2	8	Si	Modificar Borrar
11	NULL	admin	admin@admin.com	0	0	Si	Modificar Borrar

Figura A3.2 Listado de usuarios

A3.1.3 Modificación de elementos

La modificación está compuesta por una serie de elementos mostrados en la "Figura A3.3":

1. *Formulario.*
2. *Botón de guardado.* Permite guardar los cambios.
3. *Botón de borrado y 4. Botón para volver.*

1 Nombre completo
Víctor Monserrat Villatoro

2 Correo electrónico *
victor1995mv@gmail.com

3 Guardar cambios
4 Volver al listado

Figura A3.3 Modificación de elementos

A3.1.4 Borrado de elementos

El borrado se compone por los elementos mostrados en la "Figura A3.4": 1. *Mensaje de confirmación*, 2. *Botón de cancelación* y 3. *Botón de confirmación*.

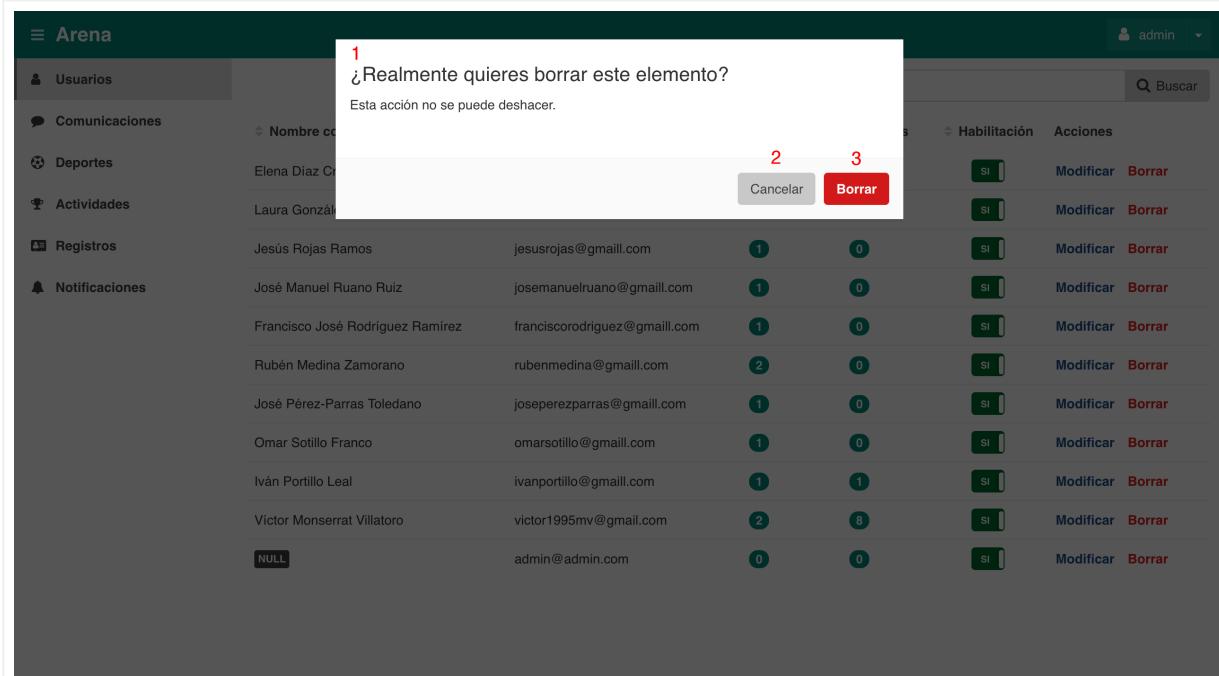


Figura A3.4 Borrado de elementos

A3.1.5 Creación de elementos

La creación se compone por los elementos mostrados en la "Figura A3.5": 1. *Formulario*, 2. *Botón de guardado* y 3. *Botón para volver*.

Figura A3.5 Creación de elementos

A3.1.6 Formularios

A3.1.7 Formulario de usuarios

El formulario de usuarios está compuesto por una serie de campos mostrados en la "Figura A3.6":

1. **Nombre completo.** El nombre completo del usuario es un campo opcional que no admite valores numéricos ni caracteres especiales.
2. **Correo electrónico.** El correo electrónico es un campo obligatorio y debe ser un correo electrónico válido.

Figura A3.6 Formulario de usuarios

A3.1.8 Formulario de comunicaciones

El formulario de comunicaciones está compuesto por una serie de campos mostrados en la "Figura A3.7":

1. **Usuario.** El usuario es un campo obligatorio y debe ser un usuario válido. Para ello, se utiliza un selector.
2. **Asunto.** El asunto es un campo obligatorio.
3. **Mensaje.** El mensaje es un campo obligatorio.

Figura A3.7 Formulario de comunicaciones

A3.1.9 Formulario de deportes

El formulario de deportes está compuesto por una serie de campos mostrados en la "Figura

A3.8":

1. *Nombre*. El nombre es un campo obligatorio de tipo textual.
2. *Descripción*. La descripción es un campo opcional de tipo textual.

Figura A3.8 Formulario de deportes

A3.1.10 Formulario de actividades

El formulario de actividades está compuesto por una serie de campos mostrados en la "Figura A3.9":

1. *Título*. El título es un campo obligatorio de tipo textual.
2. *Deporte*. El deporte es un campo obligatorio y debe ser un deporte válido. Para ello, se utiliza un selector.
3. *Propietario*. El propietario es un campo obligatorio y debe ser un usuario válido. Para ello, se utiliza un selector.
4. *Fecha y hora de comienzo*. La fecha y hora de comienzo es un campo obligatorio y debe ser válida. Para ello, se utiliza un selector de fechas y horas.
5. *Duración*. La duración es un campo obligatorio de tipo numérico. La duración está expresada en minutos.
6. *Lugar*. El lugar es un campo opcional de tipo textual.
7. *Plazas*. El número de plazas es un campo opcional de tipo numérico. Si no se expresa este será considerado infinito.
8. *Descripción*. La descripción es un campo opcional de tipo textual.

Figura A3.9 Formulario de actividades

A3.1.11 Formulario de registros

El formulario de registros está compuesto por una serie de campos mostrados en la "Figura A3.10":

1. *Usuario*. El usuario es un campo obligatorio y debe ser un usuario válido. Para ello, se utiliza un selector.
2. *Actividad*. La actividad es un campo obligatorio y debe ser una actividad válida. Para ello, se utiliza un selector.
3. *Tipo*. El tipo es un campo obligatorio y debe ser un tipo válido. Para ello, se utiliza un selector con las siguientes opciones:
 1. *Solicitud*.
 2. *Invitación*.
4. *Estado*. El estado es un campo obligatorio y debe ser un estado válido. Para ello, se

utiliza un selector con las siguientes opciones:

1. *Aceptada.*

2. *Rechazada.*

3. *En espera.*

The screenshot shows a user interface for a registration form. On the left, there is a sidebar with several menu items: 'Usuarios', 'Comunicaciones', 'Deportes', 'Actividades', 'Registros' (which is highlighted in grey), and 'Notificaciones'. The main area contains four input fields, each with a red asterisk indicating it is required. The first field is 'Usuario' with the value 'victor1995mv@gmail.com'. The second field is 'Actividad' with the value 'Un poco de basket antes del viaje'. The third field is 'Tipo' with the value 'Solicitud'. The fourth field is 'Estado' with the value 'Aceptada'. Each input field has a small dropdown arrow icon at its right end.

Figura A3.10 Formulario de registros

A3.2 Aplicación

A3.2.1 Navegación de la aplicación

Los elementos para la navegación en la aplicación se muestran en la "Figura A3.11":

1. *Barra de navegación.* Permite mostrar y ocultar la barra lateral, iniciar y cerrar sesión o volver a la página principal.
2. *Barra lateral.* Permite acceder rápidamente a la mayoría de interfaces de la aplicación. Sólo estará disponible una vez el usuario esté identificado.
3. *Avatar.* Permite cerrar sesión a través del desplegable de la "Figura A3.12". Sólo estará disponible una vez el usuario esté identificado.
4. *Pie de página.*

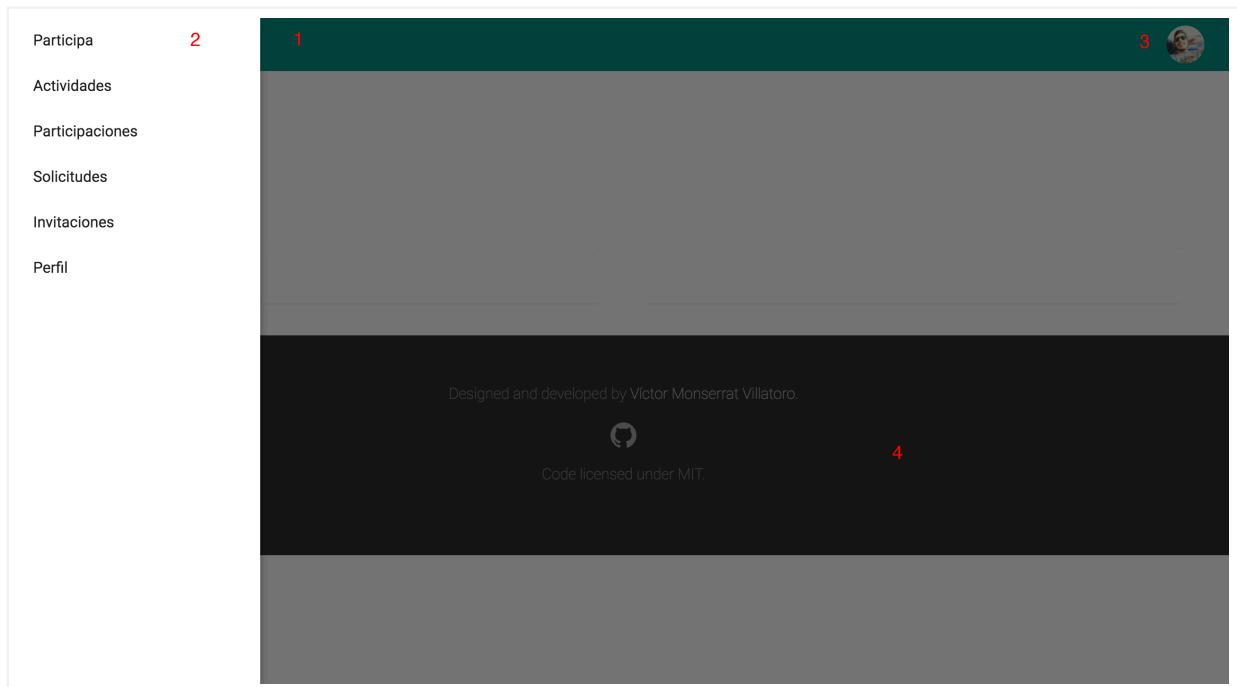


Figura A3.11 Navegación de la aplicación

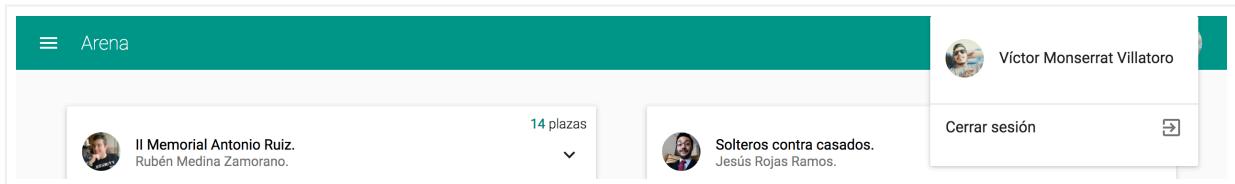


Figura A3.12 Desplegable del avatar

A3.2.2 Inicio de sesión

Para el inicio de sesión habrá que pulsar el botón destinado a ello de la "Figura A3.13" y pulsar en el botón de *Google* que se muestra en la "Figura A3.14".

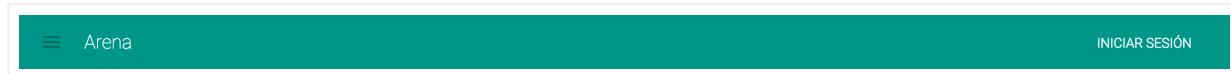


Figura A3.13 Inicio de sesión

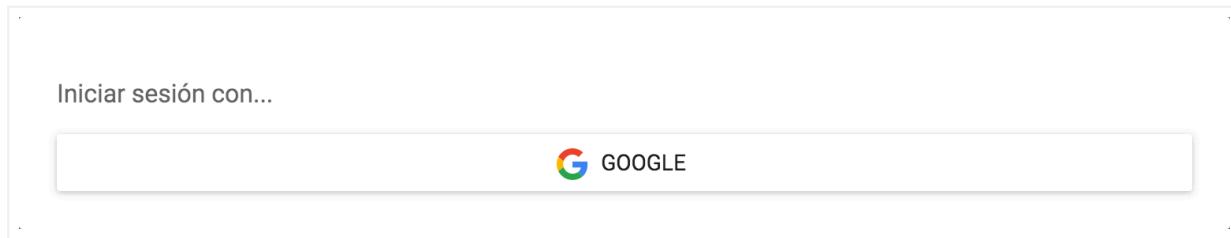


Figura A3.14 Inicio de sesión con Google

A3.2.3 Página principal

La página principal se compone de los elementos mostrados en la "Figura A3.15":

1. *Lista de actividades* y 2. *Actividad*. Al pulsar en el nombre de una actividad, el usuario será redirigido a la página de la actividad.

 A screenshot of the 'Arena' website's main page. The top navigation bar is green with the word 'Arena' and a 'INICIAR SESIÓN' button. The main content area contains a grid of activity cards. Each card includes a user profile picture, the activity name, the organizer's name, the date and time, the duration, and the number of available slots. Buttons for 'INICIA SESIÓN PARA PODER PARTICIPAR' are located at the bottom of each card.

Actividad	Organizador	Fecha y Hora	Duración	Plazas Disponibles
II Memorial Antonio Ruiz.	Rubén Medina Zamorano.	12/10/2017 a las 22:00	90 min.	14 plazas
Fútbol.	Colegio La Salle, Córdoba.			
Un poco de basket antes del viaje.	Iván Portillo Leal.	16/09/2017 a las 21:00	40 min.	5 plazas
Baloncesto.	Rabanales, Córdoba.			
Partido benéfico.	Elena Díaz Crespo.			
Partido benéfico.	Elena Díaz Crespo.	25/09/2017 a las 20:30	60 min.	2 plazas
Triples y tiros libres.	Francisco José Rodríguez Ramírez.	28/11/2017 a las 12:30	60 min.	
Tarde de raquetas.	Víctor Monserrat Villatoro.			
Tenis.	Club Figueroa, Córdoba.	26/11/2017 a las 19:30	120 min.	

Figura A3.15 Página principal

A3.2.4 Página de actividades

La página de actividades se compone de los elementos de la "Figura A3.16":

1. *Botón de creación*. Permite crear una nueva actividad. Al pulsar el botón, el usuario será redirigido al formulario de creación de actividades.
2. *Lista de actividades* y 2. *Actividad*. Al pulsar en el nombre de una actividad, el usuario será redirigido a la página de la actividad.

Designed and developed by Víctor Monserrat Villatoro.

Code licensed under MIT.

Figura A3.16 Página de actividades

A3.2.5 Página de participaciones

La página de participaciones se compone de los elementos mostrados en la "Figura A3.17":

1. *Lista de actividades* y 2. *Actividad*. Al pulsar en el nombre de una actividad, el usuario será redirigido a la página de la actividad.

Designed and developed by Víctor Monserrat Villatoro.

Code licensed under MIT.

Figura A3.17 Página de participaciones

A3.2.6 Página de solicitudes

La página de solicitudes se compone de los elementos mostrados en la "Figura A3.18":

1. *Lista de actividades* y 2. *Actividad*. Será redirigido a la página de la actividad.

The screenshot shows a web application interface titled 'Arena'. At the top, there's a navigation bar with three horizontal lines and the word 'Arena'. On the right side of the header is a user profile icon. Below the header, there are two activity requests listed in a grid-like format:

1	2	
Carrera solidaria. Rubén Medina Zamorano.	Solteros contra casados. Jesús Rojas Ramos.	12 plazas
Running. Córdoba.	Fútbol. Colegio Virgen del Carmen, Córdoba.	28/11/2017 a las 12:30 90 min.
ELIMINAR MI SOLICITUD		ELIMINAR MI SOLICITUD

At the bottom of the page, there is a dark footer section containing the text 'Designed and developed by Víctor Monserrat Villatoro.' and a small GitHub logo, followed by 'Code licensed under MIT.'

Figura A3.18 Página de solicitudes

A3.2.7 Página de invitaciones

La página de invitaciones se compone de los elementos mostrados en la "Figura A3.19":

1. *Lista de actividades* y 2. *Actividad*. Será redirigido a la página de la actividad.

The screenshot shows a web application interface titled 'Arena'. At the top, there's a navigation bar with three horizontal lines and the word 'Arena'. On the right side of the header is a user profile icon. Below the header, there are three invitation requests listed in a grid-like format:

1	2					
Juego, set y partido. Omar Sotillo Franco.	Aguas abiertas. José Manuel Ruano Ruiz.	4 plazas				
Pádel. Club Figueroa, Córdoba.	Natación. Torremolinos, Málaga.	28/11/2017 a las 12:30 45 min.				
✓ ACEPTAR INVITACIÓN ✗ RECHAZAR INVITACIÓN		✓ ACEPTAR INVITACIÓN ✗ RECHAZAR INVITACIÓN				
<table border="1"> <tr> <td> Quema grasas. Laura González Moreno-Vaquerizo. </td> <td style="text-align: right; vertical-align: bottom;"> Running. Córdoba. </td> </tr> <tr> <td colspan="2" style="text-align: center;"> ✓ ACEPTAR INVITACIÓN ✗ RECHAZAR INVITACIÓN </td> </tr> </table>			Quema grasas. Laura González Moreno-Vaquerizo.	Running. Córdoba.	✓ ACEPTAR INVITACIÓN ✗ RECHAZAR INVITACIÓN	
Quema grasas. Laura González Moreno-Vaquerizo.	Running. Córdoba.					
✓ ACEPTAR INVITACIÓN ✗ RECHAZAR INVITACIÓN						

At the bottom of the page, there is a dark footer section containing the text 'Designed and developed by Víctor Monserrat Villatoro.' and a small GitHub logo, followed by 'Code licensed under MIT.'

Figura A3.19 Página de invitaciones

A3.2.8 Página de perfil de usuario

La página de perfil de usuario se compone de los elementos mostrados en la "Figura A3.20":

1. *Avatar.*
2. *Formulario.*
 1. *Nombre.* El nombre es un campo opcional de tipo textual. No se permitirán valores numéricos.
 2. *Apellidos.* Los apellidos es un campo opcional de tipo textual. No se permitirán valores numéricos.
3. *Botón de guardado.* Permite guardar los cambios. Al guardar, aparecerá la *snackbar* de la "Figura A3.21" para poder deshacer los cambios.
4. *Botón de borrado.* Permite borrar el perfil de usuario.

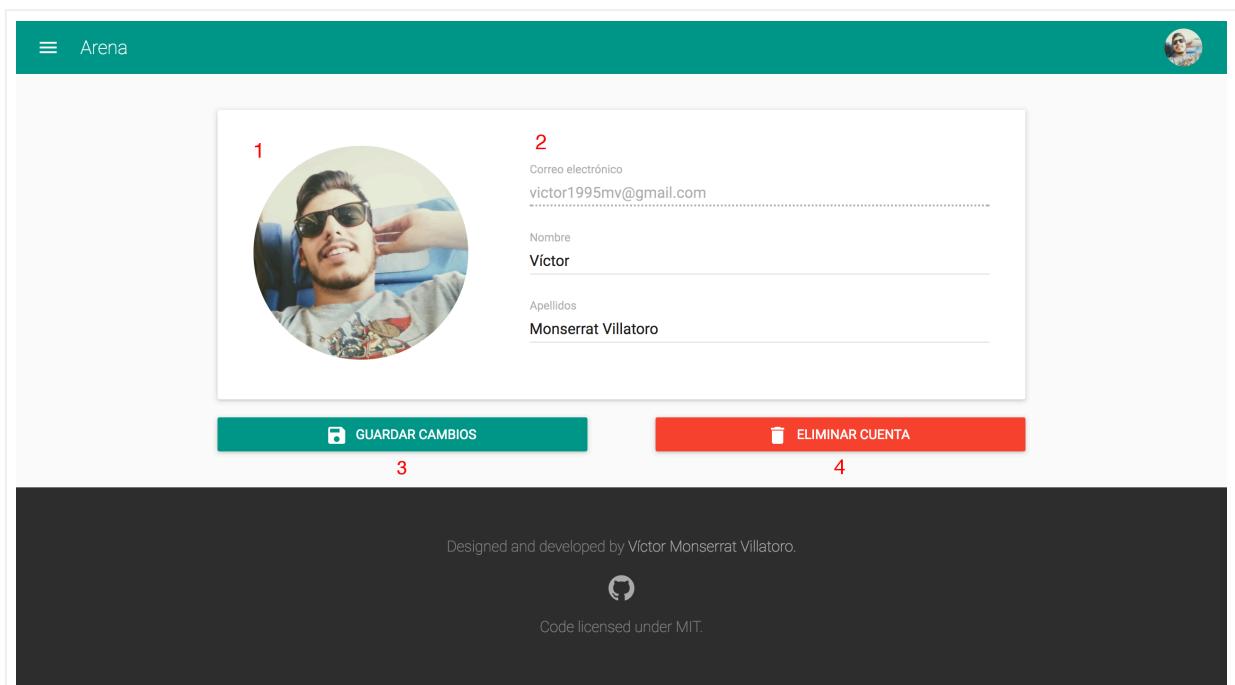


Figura A3.20 Página de perfil de usuario

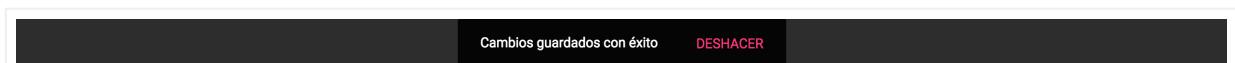


Figura A3.21 Snackbar

A3.2.9 Formulario de actividad

El formulario de actividad está compuesto por cuatro pasos:

Paso 1: deporte. Se puede ver el paso 1 en la "Figura A3.22".

1. **Deporte.** El deporte es un campo obligatorio y debe ser un deporte válido. Para ello, se utiliza un selector.
2. **Botón para continuar.** Se activará cuando los datos sean válidos.

1. ¿Qué deporte se practicará?

Deporte
* Este campo es obligatorio.

SIGUIENTE

2. ¿Cuándo comenzará?

3. Ponga título a la actividad.

4. Complete la información de la actividad.

Figura A3.22 Formulario de actividad: paso 1

Paso 2: comienzo y duración. Se puede ver el paso 2 en la "Figura A3.23".

✓ 1. ¿Qué deporte se practicará?

2. ¿Cuándo comenzará?

Fecha de inicio
* Este campo es obligatorio.

Hora de inicio
* Este campo es obligatorio.

Duración (en minutos)
* Este campo es obligatorio.

ATRÁS SIGUIENTE

3. Ponga título a la actividad.

4. Complete la información de la actividad.

Figura A3.23 Formulario de actividad: paso 2

1. **Fecha de inicio.** La fecha de inicio es un campo obligatorio y debe ser válida. Para ello, se utiliza el selector de fechas de la "Figura A3.24". Por defecto, será la actual.
2. **Hora de inicio.** La hora de inicio es un campo obligatorio y debe ser válida. Para ello, se utiliza el selector de horas de la "Figura A3.25". Por defecto, será la actual.
3. **Duración.** La duración es un campo obligatorio de tipo numérico. La duración está expresada en minutos.
4. **Botón para continuar.** Se activará cuando los datos sean válidos.
5. **Botón para volver.** Permite volver a la sección anterior.

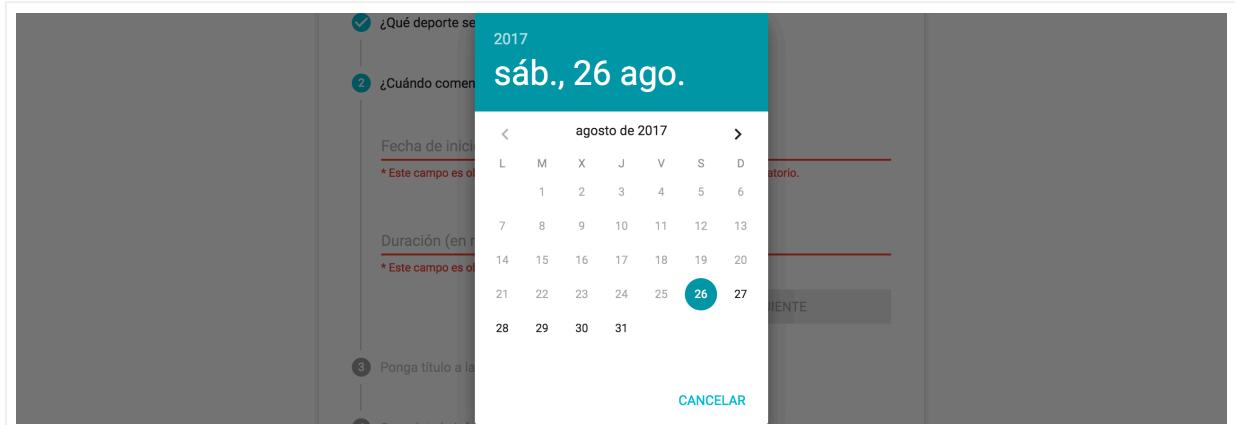


Figura A3.24 Selector de fechas

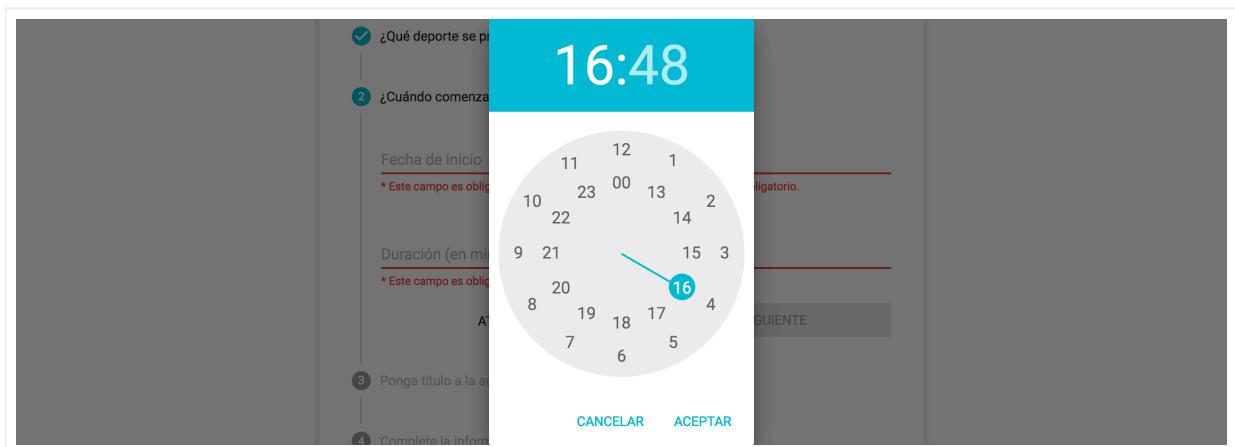


Figura A3.25 Selector de horas

Paso 3: título. Se puede ver el paso 2 en la "Figura A3.26".

1. **Título.** El título es un campo obligatorio de tipo textual.
2. **Botón para continuar.** Se activará cuando los datos sean válidos.
3. **Botón para volver.** Permite volver a la sección anterior.

Este formulario es la continuación del Paso 2. Se han completado los campos "¿Qué deporte se practicará?" y "¿Cuándo comenzará?". El campo "Título" es obligatorio, lo que se indica con un asterisco (*) y un color rojo. Los botones "ATRÁS" y "SIGUIENTE" están disponibles para navegar entre las secciones.

Figura A3.26 Formulario de actividad: paso 3

Paso 4: opcional. Se puede ver el paso 4 en la "Figura A3.27".

1. **Descripción.** La descripción es un campo opcional de tipo textual.
2. **Lugar.** El lugar es un campo opcional de tipo textual.
3. **Plazas.** El número de plazas es un campo opcional de tipo numérico. Si no se expresa este será considerado infinito.
4. **Botón para continuar.** Guardará la actividad o los cambios realizados.
5. **Botón para volver.** Permite volver a la sección anterior.

Formulario de actividad: paso 4

¿Qué deporte se practicará?
✓

¿Cuándo comenzará?
✓

Ponga título a la actividad.
✓

Complete la información de la actividad.
④

Descripción

Lugar

Plazas (infinitas por defecto)

ATRÁS CREAR ACTIVIDAD

A3.2.10 Página de actividad detallada

La página de actividad detallada está compuesta por una serie de elementos mostrados en la "Figura A3.28":

1. **Avatar del propietario.**
2. **Título.**
3. **Nombre del propietario.**
4. **Número de plazas.** Aparecerá cuando no sea ilimitado.
5. **Deporte.**
6. **Lugar.**
7. **Fecha y hora de comienzo.**
8. **Duración.**
9. **Descripción.**

10. **Acciones.** Las acciones que el usuario puede realizar con la actividad: editar, eliminar, eliminar participación, eliminar solicitud, aceptar invitación, rechazar invitación o solicitar.
11. **Lista de participaciones.** Aparecerá si el usuario está involucrado en la actividad.
12. **Lista de solicitudes.** Aparecerá si el usuario es el propietario de la actividad.
13. **Lista de invitaciones.** Aparecerá si el usuario es el propietario de la actividad.
14. **Registro.** Puede ser una participación, solicitud o invitación y podrán realizarse acciones si el usuario es el propietario de la actividad.
15. **Formulario de invitaciones.** Aparecerá si el usuario es el propietario de la actividad.

The screenshot shows a user interface for managing an activity named 'Tarde de raquetas'. The main section displays the activity details: 'Tarde de raquetas.' (1), 'Víctor Monserrat Villatoro' (2), 'Tenis.' (5), 'Club Figueroa, Córdoba.' (6), '4 2 plazas' (4), '26/11/2017 a las 19:30' (7), and '120 min.' (8). Below this, a note reads: 'Partido de tenis (dobles) nivel principiante. iremos cambiando de pareja cada set y luego podemos cenar algo en el bar del club.' (9). Action buttons for 'EDITAR' (10) and 'ELIMINAR' (11) are present. To the right, there are four smaller boxes: 'Participaciones' (12) with one entry for 'Laura González Moreno-Vaquerizo' (13), 'Solicitudes' (14) with one entry for 'Iván Portillo Leal' (15), 'Invitaciones' (16) with one entry for 'Elena Díaz Crespo' (17), and a button to 'Añadir invitaciones' (18).

Figura A3.28 Página de actividad detallada

Esta página se ha dejado vacía a propósito.

Apéndice A4

Manual de desarrollador

A4.1 Guía de desarrollo

A través de la *API* de *Arena*, otras aplicaciones pueden ver y editar contenido de *Arena*. Prácticamente, todas las peticiones a la *API* requieren autenticación y la cabecera de la petición enviada por la aplicación debe incluir un *token* de acceso válido. Para acceder a datos privados a través de la *API*, como perfiles de usuario, se debe de tener permiso por parte del usuario para obtener esos datos.

A4.1.1 Peticiones

La *API* de *Arena* está basada en los principios *REST*: se accede a los recursos a través de peticiones *HTTPS* en formato *UTF-8* a un *endpoint* de la *API*. Cuando sea posible, la *API* usará los métodos *HTTP* ("Tabla A4.1") apropiados para cada acción:

Método	Descripción
GET	Usado para obtener recursos.
POST	Usado para crear recursos.
PUT	Usado para modificar o remplazar recursos.
PATCH	Usado para modificar o remplazar parcialmente recursos.
DELETE	Usado para eliminar recursos.

Tabla A4.1 Métodos HTTP de la API

Limitación de peticiones

Por el momento, la *API* de *Arena* no tiene limitación de peticiones por aplicación. Para hacer la *API* más rápida para todos, en un futuro, se aplicará un límite de peticiones basado en la cantidad de usuarios que usen la aplicación. El código de estado **429**, indicará que se ha superado el máximo de peticiones en un determinado tiempo y se mostrará el tiempo necesario de espera para poder realizar la siguiente petición.

Autenticación

Todas las peticiones a la *API* requieren autenticación. Esto se logra enviando un *OAuth token* de acceso válido en la cabecera de la petición. Además, para acceder a la información personal de un usuario es necesario el permiso de este.

A4.1.2 Respuestas

Todos los datos y objetos de las respuestas se entregan como objetos *JSON-LD*.

Fechas y horas

Las fechas y horas son devueltas en formato *ISO 8601* en *UTC+0*: YYYY-MM-DDTHH:MM:SSZ.

Paginación

Algunos *endpoints* soportan la paginación del conjunto de datos, a través del envío del *offset* y el límite como parámetros de la consulta.

Códigos de estado

Todas las respuestas de la *API* usan los códigos de estado de la "Tabla A4.2", definidos en *RFC 2616* y *RFC 6585*.

Código de estado	Descripción
200	OK - The request has succeeded. The client can read the result of the request in the body and the headers of the response.
201	Created - The request has been fulfilled and resulted in a new resource being created.
204	No Content - The request has succeeded but returns no message body.
400	Bad Request - The request could not be understood by the server due to malformed syntax. The message body will contain more information.
401	Unauthorized - The request requires user authentication or, if the request included authorization credentials, authorization has been refused for those credentials.
403	Forbidden - The server understood the request, but is

Código de estado	Descripción
	refusing to fulfill it.
404	Not Found - The requested resource could not be found. This error can be due to a temporary or permanent condition.
410	Gone - The requested resource is no longer available at the server and no forwarding address is known. This condition is expected to be considered permanent.
500	Internal Server Error. You should never receive this error because our clever coders catch them all but if you are unlucky enough to get one, please report it to us through a comment at the bottom of this page.
502	Bad Gateway - The server was acting as a gateway or proxy and received an invalid response from the upstream server.
503	Service Unavailable - The server is currently unable to handle the request due to a temporary condition which will be alleviated after some delay. You can choose to resend the request again.

Tabla A4.2 Códigos de estado de la API

A4.2 Endpoints

Los *endpoints* de la *API* proporcionan acceso a los datos y a los usuarios de *Arena* a las aplicaciones externas. Estos se exponen en la "Tabla A4.3":

Endpoint	Método	Uso	Retorno	Autenticación
/activities	GET	Obtiene la lista de actividades	actividades*	
/activities	POST	Añade una actividad a mi lista	actividad	OAuth2
/activities/{id}	DELETE	Borra una actividad de mi lista		OAuth2
/activities/{id}	GET	Obtiene una actividad de la lista	actividad	
/activities/{id}	PUT	Modifica una actividad de mi lista	actividad	OAuth2
/activities/{id}/accept	PATCH	Acepta una invitación en una actividad	actividad	OAuth2
/activities/{id}/apply	POST	Solicita una participación en una actividad	actividad	OAuth2
/activities/{id}/cancel	DELETE	Cancela una		OAuth2

Endpoint	Método	Uso	Retorno	Autenticación
		participación o una solicitud en una actividad		
/activities/{id}/refuse	PATCH	Rechaza una invitación en una actividad	actividad	OAuth2
/me	DELETE	Borra mi perfil de usuario		OAuth2
/me	GET	Obtiene mi perfil de usuario	usuario	OAuth2
/me	PUT	Modifica mi perfil de usuario	usuario	OAuth2
/me/activities	GET	Obtiene mis actividades	actividades*	OAuth2
/me/activities/accepted	GET	Obtiene las actividades en las que participo	actividades*	OAuth2
/me/activities/applied	GET	Obtiene las actividades solicitadas	actividades*	OAuth2
/me/activities/invited	GET	Obtiene las actividades en las que estoy invitado	actividades*	OAuth2
/registrations	GET	Obtiene la lista de registros	registros*	OAuth2
/registrations	POST	Añade una invitación en una actividad	registro	OAuth2
/registrations/{id}	DELETE	Borra una participación o una invitación en una actividad		OAuth2
/registrations/{id}	GET	Obtiene un registro de la lista	registro	OAuth2
/registrations/{id}	PUT	Modifica un registro de mi lista	registro	OAuth2
/sports	GET	Obtiene la lista de deportes	deportes*	
/sports?name={name}	GET	Obtiene un deporte de la lista	deporte	
/sports	POST	Añade un deporte a la lista	deporte	OAuth2

Endpoint	Método	Uso	Retorno	Autenticación
/sports/{id}	DELETE	Borra un deporte de la lista		OAuth2
/sports/{id}	GET	Obtiene un deporte de la lista	deporte	
/sports/{id}	PUT	Modifica un deporte de la lista	deporte	OAuth2
/users	GET	Obtiene la lista de usuarios	usuarios*	OAuth2
/users	POST	Añade un usuario a la lista	usuario	OAuth2
/users/{id}	DELETE	Borra un deporte de la lista		OAuth2
/users/{id}	GET	Obtiene un usuario de la lista	usuario	OAuth2
/users/{id}	PUT	Modifica un usuario de la lista	usuario	OAuth2

Tabla A4.3 Endpoints de la API

Esta página se ha dejado vacía a propósito.

Apéndice A5

Manual de código

A5.1 Introducción

Este manual de código corresponde a la aplicación web *Arena*, orientada a usuarios deportistas para gestionar actividades deportivas. A continuación, se detallan los requisitos previos para la implementación del sistema y se presentan todos los archivos que forman parte de la aplicación, organizados según su funcionalidad y describiendo su contenido.

A5.2 Requisitos previos

A5.2.1 Backend

El *backend* de la aplicación está programado con *Symfony3*, un *framework* desarrollado con *PHP* versión 5.5.9. Concretamente los requisitos de instalación se muestran en la "Tabla A5.1".

Software	Versión mínima	Versión recomendada
PHP (1)(2)	7.1	≥ 7.1
Apache (3)	2.2	2.4
Nginx (3)	1.4	1.4
MySQL (4)	5.0	5.5
Git	1.8	1.9

Tabla A5.1 Tabla de requisitos para la instalación del backend

Notas:

1. *PHP* debe ser instalado con los siguientes módulos extra: *curl*, *intl*, *json*, *mysqli*, *xml*.
2. *PHP* debe tener configurado el uso horario del sistema. Para ello añadir al *php.ini* del sistema lo siguiente: *date.timezone = "Europe/Madrid"*.
3. Solo es necesario uno de los dos servidores web.
4. El sistema soporta otros gestores como *MariaDB*, *Postgresql* u *Oracle*. Para más información, leer la documentación oficial de *Symfony* y *Doctrine*.

A5.2.2 Frontend

El *frontend* de la aplicación está programado con *React*, una librería desarrollada con *Javascript*. Concretamente los requisitos de instalación se muestran en la "Tabla A5.2".

Software	Versión mínima	Versión recomendada
NodeJS	6.0.0	≥ 6.5.0
Nginx	1.4	1.4
Git	1.8	1.9

Tabla A5.2 Tabla de requisitos para la instalación del frontend

A5.3 Descripción modular

A5.3.1 Backend

Directorios principales

La estructura principal de la *API* es similar a la de cualquier proyecto *Symfony* con alguna modificación al usar *BDD*, como se aprecia en la "Tabla A5.3".

Directorio	Propósito
app/	Contiene los archivos de configuración y los recursos globales.
bin/	Contiene los archivos para ejecutar y depurar la aplicación.
features/	Contiene todos los archivos de pruebas basados en <i>BDD</i> .
src/	Guarda todo el código fuente propio de la aplicación.
test/	Contiene todos los archivos de pruebas de la aplicación. En este caso, las pruebas se encuentran en <i>features/</i> por usar <i>BDD</i> .
var/	Es el único directorio en el que <i>Symfony</i> debe tener permisos de escritura.
vendor/	Contiene todo el código fuente de <i>Symfony</i> y de todas las

Directorio	Propósito
	librerías externas.
web/	El único directorio público del proyecto. Contiene los archivos web (CSS, JavaScript e imágenes) y los controladores frontales de la aplicación.

Tabla A5.3 Directorios principales de la API

Directarios app/

Los directorios de [app/](#) se muestran en la "Tabla A5.4".

Directorio	Propósito
app/config/	Guarda todos los archivos de configuración de la aplicación.
app/Resources/	Almacena los recursos que se utilizan globalmente en el proyecto (como por ejemplo las plantillas).
app/spool/	Guarda los archivos de correo electrónico almacenados en cola.

Tabla A5.4 Directorios app/

Directarios app/config/

Los directorios de [app/config/](#) se muestran en la "Tabla A5.5".

Directorio	Propósito
app/config/bundles/	Guarda todos los archivos de configuración de los <i>bundles</i> utilizados.
app/config/services/	Contiene todos los archivos de configuración de servicios.

Tabla A5.5 Directorios app/config/

Ficheros app/config/bundles/

Los ficheros de [app/config/bundles/](#) se muestran en la "Tabla A5.6".

Fichero	Propósito
api_platform.yml	Contiene la configuración de <i>API-Platform</i> .
easyadmin.yml	Contiene la configuración de <i>EasyAdmin</i> .
fos_user.yml	Contiene la configuración de <i>FOSUser</i> .
lexic_jwt_authentication.yml	Contiene la configuración de <i>JWTAuthentication</i> .
nelmio_api_doc.yml	Contiene la configuración de <i>APIDoc</i> .

Fichero	Propósito
stofDoctrineExtensions.yml	Contiene la configuración de las extensiones de Doctrine (<i>Sluggable</i>).

Tabla A5.6 Ficheros app/config/bundles/**Ficheros app/config/services/**

Los ficheros de [app/config/services/](#) se muestran en la "Tabla A5.7".

Fichero	Propósito
alice_providers.yml	Contiene el alta de los servicios de proveedores de datos (deportes y avatares).
builders.yml	Contiene el alta del servicio del constructor del serializador para actividades.
dispatchers.yml	Contiene el alta de los servicios de disparadores de eventos.
factories.yml	Contiene el alta de los servicios de factorías.
filters.yml	Contiene el alta del servicio para filtrar deportes.
managers.yml	Contiene el alta de los servicios de gestión.
query_extensions.yml	Contiene el alta del servicio para extender las consultas a base de datos (actividades futuras).
repositories.yml	Contiene el alta de los servicios de repositorios.
voters.yml	Contiene el alta de los servicios de votación que controlan el acceso a los recursos (seguridad).

Tabla A5.7 Ficheros app/config/services/**Ficheros app/config/**

Los ficheros de [app/config/](#) se muestran en la "Tabla A5.8".

Fichero	Propósito
config.yml	Contiene la configuración general de la aplicación.
config_dev.yml	Contiene la configuración para el entorno de desarrollo de la aplicación.
config_prod.yml	Contiene la configuración para el entorno de producción de la aplicación.
config_test.yml	Contiene la configuración para el entorno de prueba de la aplicación.
parameters.yml	Contiene los parámetros de configuración variables en la aplicación.

Fichero	Propósito
routing.yml	Contiene la configuración de enrutamiento general de la aplicación.
routing_dev.yml	Contiene la configuración de enrutamiento para el entorno de desarrollo de la aplicación.
security.yml	Contiene la configuración de seguridad de la aplicación.

Tabla A5.8 Ficheros app/config/

Directrios app/Resources/

Los directorios de [app/Resources/](#) se muestran en la "Tabla A5.9".

Directorio	Propósito
app/Resources/translations/	Guarda todos los archivos de traducción de la aplicación.
app/Resources/views/	Almacena todos las vistas (plantillas) de la aplicación.

Tabla A5.9 Directorios app/Resources/

Directrios src/

Los directorios de [src/](#) se muestran en la "Tabla A5.10".

Directorio	Propósito
src/AppBundle/Action/	Contiene las acciones (controladores) de la <i>API</i> .
src/AppBundle/Behat/	Guarda los contextos y las composiciones necesarios para las pruebas con <i>Behat</i> .
src/AppBundle/Command/	Contiene los comandos propios de la aplicación que serán ejecutados por la consola de <i>Symfony</i> .
src/AppBundle/Doctrine/	Guarda los archivos para la extensión de <i>Doctrine</i> en la aplicación.
src/AppBundle/Entity/	Contiene las entidades (modelo) de la aplicación.
src/AppBundle/Event/	Contiene los eventos disparados en la aplicación.
src/AppBundle/EventDispatcher/	Contiene los disparadores de eventos de la aplicación.
src/AppBundle/EventSubscriber/	Contiene los suscriptores de eventos de la aplicación.
src/AppBundle/Exception/	Contiene las excepciones de la aplicación.
src/AppBundle/Factory/	Contiene las factorías de la aplicación.
src/AppBundle/Repository/	Contiene los repositorios de la aplicación.
src/AppBundle/Resources/	Almacena los recursos que utiliza la aplicación.

Directorio	Propósito
src/AppBundle/Security/	Guarda los archivos de seguridad de la aplicación.
src/AppBundle/Serializer/	Contiene los constructores de serializadores de la aplicación.
src/AppBundle/Services/	Contiene los servicios de la aplicación.

Tabla A5.10 Directorios src/ de la API

Ficheros src/AppBundle/Action/

Los ficheros de [src/AppBundle/Action/](#) se muestran en la "Tabla A5.11".

Fichero	Propósito
ActivitiesGetAction.php	Clase de la acción para obtener todas las actividades.
ActivityAcceptAction.php	Clase de la acción para aceptar una actividad.
ActivityApplyAction.php	Clase de la acción para solicitar la participación en una actividad.
ActivityCancelAction.php	Clase de la acción para cancelar la participación o la solicitud en una actividad.
ActivityDeleteAction.php	Clase de la acción para eliminar una actividad del usuario.
ActivityGetAction.php	Clase de la acción para obtener una actividad.
ActivityPostAction.php	Clase de la acción para crear una nueva actividad.
ActivityPutAction.php	Clase de la acción para modificar una actividad del usuario.
ActivityRefuseAction.php	Clase de la acción para rechazar una invitación al usuario en una actividad.
AuthAction.php	Clase de la acción para la autenticación de usuarios.
MeActivitiesAcceptedGetAction.php	Clase de la acción para obtener las actividades en las que participa el usuario.
MeActivitiesAppliedGetAction.php	Clase de la acción para obtener las actividades en las que ha solicitado participar el usuario.
MeActivitiesGetAction.php	Clase de la acción para obtener las actividades del usuario.
MeActivitiesInvitedGetAction.php	Clase de la acción para obtener las actividades en las que ha sido invitado el usuario.
MeDeleteAction.php	Clase de la acción para borrar el usuario.
MeGetAction.php	Clase de la acción para obtener el usuario.
MePutAction.php	Clase de la acción para modificar el usuario.
RegistrationAcceptAction.php	Clase de la acción para aceptar una solicitud en una actividad del usuario.

Fichero	Propósito
RegistrationDeleteAction.php	Clase de la acción para cancelar una participación o una invitación a una actividad del usuario.
RegistrationPostAction.php	Clase de la acción para hacer una invitación a una actividad del usuario.
RegistrationRefuseAction.php	Clase de la acción para rechazar una solicitud en una actividad del usuario.
UsersGetAction.php	Clase de la acción para obtener los usuarios de la aplicación.

Tabla A5.11 Ficheros src/AppBundle/Action/**Directorios src/AppBundle/Behat/**

Los directorios de [src/AppBundle/Behat/](#) se muestran en la "Tabla A5.12".

Directorio	Propósito
src/AppBundle/Behat/Composition/	Guarda las composiciones necesarios para las pruebas con <i>Behat</i> .
src/AppBundle/Behat/Context/	Guarda los contextos necesarios para las pruebas con <i>Behat</i> .

Tabla A5.12 Directorios src/AppBundle/Behat/**Ficheros src/AppBundle/Behat/Composition/**

Los ficheros de [src/AppBundle/Behat/Composition/](#) se muestran en la "Tabla A5.13".

Fichero	Propósito
KernelAwareTrait.php	Característica para injectar la obtención del <i>EntityManager</i> .
SharedStorageTrait.php	Característica para el uso de un almacenamiento compartido.

Tabla A5.13 Ficheros src/AppBundle/Behat/Composition/**Directorios src/AppBundle/Behat/Context/**

Los directorios de [src/AppBundle/Behat/Context/](#) se muestran en la "Tabla A5.14".

Directorio	Propósito
src/AppBundle/Behat/Context/Domain	Guarda todos los contextos del dominio del problema real.
src/AppBundle/Behat/Context/Hook	Almacena todos los contextos no incluidos en el dominio del problema real.
src/AppBundle/Behat/Context/	Guarda todos los contextos para la transformación de datos

Directorio	Propósito
Transform	en objetos.
DefaultContext.php	Clase de la que extenderán todos los contextos que añade una serie de características a la clase <i>Context</i>

Tabla A5.14 Directorios src/AppBundle/Behat/Context/**Ficheros src/AppBundle/Command/**

Los ficheros de [src/AppBundle/Command/](#) se muestran en la "Tabla A5.15".

Fichero	Propósito
GenerateKeysCommand.php	Comando de generación de las claves privada y pública para el uso de <i>JWT</i> .
GenerateTokenCommand.php	Comando para la creación de un token para un determinado usuario de la aplicación.

Tabla A5.15 Ficheros src/AppBundle/Command/**Ficheros src/AppBundle/Doctrine/**

Los ficheros de [src/AppBundle/Doctrine/](#) se muestran en la "Tabla A5.16".

Fichero	Propósito
ORM/Extension/ NotExpiredExtension.php	Extensión para filtrar las futuras actividades en las consultas a base de datos.

Tabla A5.16 Ficheros src/AppBundle/Doctrine/**Ficheros src/AppBundle/Entity/**

Los ficheros de [src/AppBundle/Entity/](#) se muestran en la "Tabla A5.17".

Fichero	Propósito
Activity.php	Clase que representa actividades del problema del mundo real.
Communication.php	Clase que representa comunicaciones entre administrador y usuarios.
Notification.php	Clase que representa notificaciones de la aplicación a usuarios de esta.
Registration.php	Clase que representa los diferentes tipos de registros de usuarios en actividades de la aplicación.
Sport.php	Clase que representa deportes del problema del mundo real.

Fichero	Propósito
User.php	Clase que representa usuarios de la aplicación.

Tabla A5.17 Ficheros src/AppBundle/Entity/**Ficheros src/AppBundle/Event/**

Los ficheros de [src/AppBundle/Event/](#) se muestran en la "Tabla A5.18".

Fichero	Propósito
Abstracts/AbstractActivityEvent.php	Clase de la que heredarán todos los eventos de actividades que contiene métodos comunes.
Abstracts/AbstractCommunicationEvent.php	Clase de la que heredarán todos los eventos de comunicaciones que contiene métodos comunes.
Abstracts/AbstractNotificationEvent.php	Clase de la que heredarán todos los eventos de notificaciones que contiene métodos comunes.
Abstracts/AbstractRegistrationEvent.php	Clase de la que heredarán todos los eventos de registros que contiene métodos comunes.
Abstracts/AbstractSportEvent.php	Clase de la que heredarán todos los eventos de deportes que contiene métodos comunes.
Abstracts/AbstractUserEvent.php	Clase de la que heredarán todos los eventos de usuarios que contiene métodos comunes.
AcceptedApplicationEvent.php	Evento de aceptación de una solicitud.
AcceptedInvitationEvent.php	Evento de aceptación de una invitación.
CreatedActivityEvent.php	Evento de creación de una actividad.
CreatedApplicationEvent.php	Evento de creación de una solicitud.
CreatedCommunicationEvent.php	Evento de creación de una comunicación.
CreatedInvitationEvent.php	Evento de creación de una invitación.
CreatedNotificationEvent.php	Evento de creación de una notificación.
CreatedSportEvent.php	Evento de creación de un deporte.
CreatedUserEvent.php	Evento de creación de un usuario.
DeletedActivityEvent.php	Evento de borrado de una actividad.
DeletedRegistrationEvent.php	Evento de borrado de un registro.
DeletedSportEvent.php	Evento de borrado de un deporte.
DeletedUserEvent.php	Evento de borrado de un usuario.
RefusedApplicationEvent.php	Evento de rechazo de una solicitud.
RefusedInvitationEvent.php	Evento de rechazo de una invitación.

Fichero	Propósito
UpdatedActivityEvent.php	Evento de modificación de una actividad.
UpdatedSportEvent.php	Evento de modificación de un deporte.
UpdatedUserEvent.php	Evento de modificación de un usuario.

Tabla A5.18 Ficheros src/AppBundle/Event/**Ficheros src/AppBundle/EventDispatcher/**

Los ficheros de [src/AppBundle/EventDispatcher/](#) se muestran en la "Tabla A5.19".

Fichero	Propósito
AbstractEventDispatcher.php	Clase de la que heredarán todos los disparadores de eventos.
ActivityEventDispatcher.php	Clase que disparará todos los eventos de actividades.
CommunicationEventDispatcher.php	Clase que disparará todos los eventos de comunicaciones.
NotificationEventDispatcher.php	Clase que disparará todos los eventos de notificaciones.
RegistrationEventDispatcher.php	Clase que disparará todos los eventos de registros.
SportEventDispatcher.php	Clase que disparará todos los eventos de deportes.
UserEventDispatcher.php	Clase que disparará todos los eventos de usuarios.

Tabla A5.19 Ficheros src/AppBundle/EventDispatcher/**Ficheros src/AppBundle/EventSubscriber/**

Los ficheros de [src/AppBundle/EventSubscriber/](#) se muestran en la "Tabla A5.20".

Fichero	Propósito
AcceptedApplicationSubscriber.php	Suscriptor del evento de aceptación de solicitudes.
AcceptedInvitationSubscriber.php	Suscriptor del evento de aceptación de invitaciones.
CreatedApplicationSubscriber.php	Suscriptor del evento de creación de solicitudes.
CreatedCommunicationSubscriber.php	Suscriptor del evento de creación de comunicaciones.
CreatedInvitationSubscriber.php	Suscriptor del evento de creación de invitaciones.
DeletedActivitySubscriber.php	Suscriptor del evento de borrado de actividades.
DeletedRegistrationSubscriber.php	Suscriptor del evento de borrado de registros.
DeletedSportSubscriber.php	Suscriptor del evento de borrado de deportes.
RefusedApplicationSubscriber.php	Suscriptor del evento de rechazo de solicitudes.

Fichero	Propósito
RefusedInvitationSubscriber.php	Suscriptor del evento de rechazo de invitaciones.
UpdatedActivitySubscriber.php	Suscriptor del evento de modificación de actividades.

Tabla A5.20 Ficheros src/AppBundle/EventSubscriber/**Ficheros src/AppBundle/Factory/**

Los ficheros de [src/AppBundle/Factory/](#) se muestran en la "Tabla A5.21".

Fichero	Propósito
ActivityFactory.php	Clase para generar actividades.
CommunicationFactory.php	Clase para generar comunicaciones.
Factory.php	Clase para generar factorías.
NotificationFactory.php	Clase para generar notificaciones.
RegistrationFactory.php	Clase para generar registros.
SportFactory.php	Clase para generar deportes.
UserFactory.php	Clase para generar usuarios.

Tabla A5.21 Ficheros src/AppBundle/Factory/**Ficheros src/AppBundle/Repository/**

Los ficheros de [src/AppBundle/Repository/](#) se muestran en la "Tabla A5.22".

Fichero	Propósito
ActivityRepository.php	Clase para hacer consultas sobre actividades.
CommunicationRepository.php	Clase para hacer consultas sobre comunicaciones.
NotificationRepository.php	Clase para hacer consultas sobre notificaciones.
RegistrationRepository.php	Clase para hacer consultas sobre registros.
SportRepository.php	Clase para hacer consultas sobre deportes.
UserRepository.php	Clase para hacer consultas sobre usuarios.

Tabla A5.22 Ficheros src/AppBundle/Repository/**Directorios src/AppBundle/Resources/**

Los directorios de [src/AppBundle/Resources/](#) se muestran en la "Tabla A5.23".

Directorio	Propósito
src/AppBundle/Resources/fixtures/orm/	Contiene los datos de carga para la aplicación.
src/AppBundle/Resources/fixtures/providers/	Contiene los proveedores de datos para la carga en la aplicación.

Tabla A5.23 Directorios src/AppBundle/Resources/

Ficheros src/AppBundle/Security/

Los ficheros de [src/AppBundle/Security/](#) se muestran en la "Tabla A5.24".

Fichero	Propósito
Voter/ActivityVoter.php	Clase para votar si un usuario está o no involucrado.

Tabla A5.24 Ficheros src/AppBundle/Security/

Ficheros src/AppBundle/Serializer/

Los ficheros de [src/AppBundle/Serializer/](#) se muestran en la "Tabla A5.25".

Fichero	Propósito
ActivityContextBuilder.php	Clase para modificar el serializador de actividades según el contexto.

Tabla A5.25 Ficheros src/AppBundle/Serializer/

Ficheros src/AppBundle/Services/

Los ficheros de [src/AppBundle/Services/](#) se muestran en la "Tabla A5.26".

Fichero	Propósito
ActivityManager.php	Clase para la realización de acciones sobre actividades.
CommunicationManager.php	Clase para la realización de acciones sobre comunicaciones.
NotificationManager.php	Clase para la realización de acciones sobre notificaciones.
RegistrationManager.php	Clase para la realización de acciones sobre registros.
SportManager.php	Clase para la realización de acciones sobre deportes.
UserManager.php	Clase para la realización de acciones sobre usuarios.

Tabla A5.26 Ficheros src/AppBundle/Services/

Ficheros src/

Los ficheros de [src/](#) se muestran en la "Tabla A5.27".

Fichero	Propósito
AppBundle/AppBundle.php	Clase del <i>bundle</i> de la aplicación.
AppBundle/ArenaEvents.php	Clase que registra los nombres de los eventos de la aplicación

Tabla A5.27 Ficheros src/

Directorios var/

Los directorios de [var/](#) se muestran en la "Tabla A5.28".

Directorio	Propósito
var/cache/	Contiene todos los archivos cacheados por <i>Symfony</i> (clases, enrutamiento, plantillas, entidades, validación, etc.).
var/jwt/	Contiene las claves privada y pública para el uso de <i>JWT</i> .
var/logs/	Contiene los archivos de <i>log</i> generados al ejecutar la aplicación.
var/sessions/	Contiene los archivos generados de las sesiones.

Tabla A5.28 Directorios var/

Ficheros raíz

Los ficheros del directorio [/](#) se muestran en la "Tabla A5.29".

Fichero	Propósito
behat.yml.dist	Contiene la configuración de <i>Behat</i> para el uso de <i>BDD</i> .
composer.json	Contiene la información y configuración del proyecto, así como sus dependencias.
phpspec.yml.dist	Contiene la configuración de <i>PHPSpec</i> para el uso de pruebas.
phpunit.yml.dist	Contiene la configuración de <i>PHPUnit</i> para el uso de pruebas.

Tabla A5.29 Ficheros raíz de la API

Dependencias para el entorno de producción

Las dependencias para el entorno de producción de la *API* se muestran en la "Tabla A5.30" junto a la versión mínima de esta.

Dependencia	Versión
php	7.0
symfony/symfony	3.2
api-platform/core	2.0
doctrine/orm	2.5
doctrine/doctrine-bundle	1.6
doctrine/doctrine-cache-bundle	1.2
symfony/swiftmailer-bundle	2.3
symfony/monolog-bundle	3.0
sensio/distribution-bundle	5.0
sensio/framework-extra-bundle	3.0.2
incenteev/composer-parameter-handler	2.0
dunglas/action-bundle	0.3
nelmio/cors-bundle	1.4
phpdocumentor/reflection-docblock	3.0
friendsofsymfony/user-bundle	2.0
stof/doctrine-extensions-bundle	1.2
lexik/jwt-authentication-bundle	2.2
webmozart/assert	1.2
nelmio/api-doc-bundle	2.13
fabpot/goutte	3.2
javierreguiluz/easyadmin-bundle	1.16

Tabla A5.30 Dependencias de la API para el entorno de producción

Dependencias para el entorno de desarrollo

Para el desarrollo de la *API*, además, tendremos más dependencias. Estas se muestran en la "Tabla A5.31", junto a la versión mínima de esta.

Dependencia	Versión
api-platform/schema-generator	1.2
sensio/generator-bundle	3.0

Dependencia	Versión
symfony/phpunit-bridge	3.0
behat/behat	3.1
behat/symfony2-extension	2.1
behat/mink	1.7
behat/mink-extension	2.2
behat/mink-browserkit-driver	1.3.1
behatch-contexts	2.5
knplabs/rad-fixtures-load	1.6
friends-of-behat/performance-extension	1.0
friends-of-behat/service-container-extension	0.3.0

Tabla A5.31 Dependencias de la API para el entorno de desarrollo

A5.3.2 Frontend

Directorios principales

La estructura principal de la aplicación es como la de cualquier proyecto *React*, podemos ver los directorios principales en la "Tabla A5.32".

Directorio	Propósito
node_modules/	Contiene todos los módulos de <i>NodeJS</i> y de todas las librerías externas.
public/	El único directorio público del proyecto. Contiene los archivos web (CSS, JavaScript e imágenes) y el índice del contenido web.
src/	Guarda todo el código fuente propio de la aplicación.

Tabla A5.32 Directorios principales de la aplicación

Directorios src/

Los directorios de [src/](#) son similares a cualquier proyecto con arquitectura *Redux* y se muestran en la "Tabla A5.33".

Directorio	Propósito
src/actions/	Contiene todas las acciones registradas en la aplicación. Las acciones modifican el estado del store.

Directorio	Propósito
src/components/	Contiene todos los componentes representados en la aplicación. Los componentes son independientes del estado de la aplicación.
src/containers/	Aplican el estado actual del <i>store</i> de la aplicación a los componentes usados.
src/reducers/	Contienen los parámetros variables del estado del <i>store</i> .
src/stories/	Es un directorio de <i>StoryBook</i> que facilita el desarrollo de componentes. Contiene historias o componentes con determinados estados para ver su representación fácilmente según estos.
src/index.js	Representa la aplicación completa y se incluye en el índice del contenido web.

Tabla A5.33 Directorios src/ de la aplicación

Directarios src/actions/

Los directarios de [src/actions/](#) se muestran en la "Tabla A5.34".

Directorio	Propósito
src/actions/activities/	Contiene las constantes y acciones relacionadas con la lista de actividades.
src/actions/activity/	Contiene las constantes y acciones relacionadas con la actividad detallada.
src/actions/activityForm/	Contiene las constantes y acciones relacionadas con el formulario de actividades.
src/actions/auth/	Contiene las constantes y acciones relacionadas con la autenticación.
src/actions/sports/	Contiene las constantes y acciones relacionadas con la lista de deportes.
src/actions/ui/	Contiene las constantes y acciones relacionadas con la interfaz de usuario.

Tabla A5.34 Directorios src/actions/

Ficheros src/components/

Los ficheros de [src/components/](#) se muestran en la "Tabla A5.35".

Fichero	Propósito
ActivitiesList.js	Componente con la lista de actividades.
Activity.js	Componente con los datos de una actividad.

Fichero	Propósito
ActivityEditForm.js	Componente con el formulario para modificar una actividad.
ActivityItem.js	Componente con un elemento de la lista de actividades.
App.js	Componente padre que contiene al resto de ellos y se incluye en el índice del contenido web.
AvatarComposition.js	Componente que contiene una composición del avatar de un usuario.
DatePickerComposition.js	Componente que contiene una composición de un selector de fecha.
DrawerComposition.js	Componente que contiene una composición de un <i>drawer</i> .
IconMenuComposition.js	Componente que contiene una composición de un ícono con menú.
LoginDialog.js	Componente con el diálogo para la identificación de un usuario.
NotFound.js	Componente para comunicar al usuario que el contenido no ha sido encontrado.
TimePickerComposition.js	Componente que contiene una composición de un selector de hora.
ToolbarComposition.js	Componente que contiene una composición de una barra de herramientas.
UserForm.js	Componente con el formulario para modificar el perfil de un usuario.

Tabla A5.35 Ficheros src/components/

Ficheros src/containers/

Los ficheros de [src/containers/](#) se muestran en la "Tabla A5.36".

Fichero	Propósito
ActivitiesList.js	Contenedor de la lista de actividades.
Activity.js	Contenedor de los datos de una actividad.
ActivityEditForm.js	Contenedor del formulario para modificar una actividad.
ActivityItem.js	Contenedor del elemento de la lista de actividades.
App.js	Contenedor de la aplicación.
AvatarComposition.js	Contenedor de la composición del avatar de un usuario.
DrawerComposition.js	Contenedor de la composición de un <i>drawer</i> .
InvitationsList.js	Contenedor de la lista de actividades con invitación.

Fichero	Propósito
LoginDialog.js	Contenedor del diálogo para la identificación de un usuario.
MyActivitiesList.js	Contenedor de la lista de las actividades de un usuario.
MyApplicationsList.js	Contenedor de la lista de las actividades solicitadas.
MyRegistrationsList.js	Contenedor de la lista de las actividades con registro.
ToolbarComposition.js	Contenedor de la composición de una barra de herramientas.
UserForm.js	Contenedor del formulario del perfil de un usuario.

Tabla A5.36 Ficheros src/containers/**Ficheros src/reducers/**

Los ficheros de [src/reducers/](#) se muestran en la "Tabla A5.37".

Fichero	Propósito
activities.js	Contienen los parámetros variables del estado del <i>store</i> sobre las actividades.
activity.js	Contienen los parámetros variables del estado del <i>store</i> sobre la actividad.
activityForm.js	Contienen los parámetros variables del estado del <i>store</i> sobre el formulario de actividad.
auth.js	Contienen los parámetros variables del estado del <i>store</i> sobre la autenticación.
index.js	Combina todos los reducers en uno para su inyección.
sports.js	Contienen los parámetros variables del estado del <i>store</i> sobre los deportes.
ui.js	Contienen los parámetros variables del estado del <i>store</i> sobre la interfaz de usuario.

Tabla A5.37 Ficheros src/reducers/**Ficheros raíz**

Los ficheros de [/](#) se muestran en la "Tabla A5.38".

Fichero	Propósito
.babelrc	Contiene la configuración del entorno de desarrollo.
package.json	Contiene la información y configuración del proyecto, así como sus dependencias.

Tabla A5.38 Ficheros raíz de la aplicación

Dependencias para el entorno de producción

Las dependencias para el entorno de producción de la aplicación se muestran en la "Tabla A5.39" junto a la versión mínima de esta.

Dependencia	Versión
axios	0.16.0
hellojs	1.14.1
intl	1.2.5
intl-locales-supported	1.0.0
jwt-decode	2.2.0
material-ui	0.17.1
moment	2.18.1
normalize.css	6.0.0
react	15.4.2
react-dom	15.4.2
react-grid-system	2.7.0
react-redux	5.0.3
react-router-dom	4.1.1
react-router-redux	4.0.8
react-swipeable-views	0.12.1
react-tap-event-plugin	2.0.1
redux	3.6.0

Tabla A5.39 Dependencias de la aplicación para el entorno de producción

Dependencias para el entorno de desarrollo

Para el desarrollo de la aplicación, además, tendremos más dependencias. Estas se muestran en la "Tabla A5.40", junto a la versión mínima de esta.

Dependencia	Versión
babel-preset-stage-1	6.22.0
react-scripts	0.9.5

Tabla A5.40 Dependencias de la aplicación para el entorno de desarrollo