Escuela Politécnica Superior de Jaén

Grado en ingeniería informática

UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior de Jaén

Trabajo Fin de Grado

ESTUDIO Y DESARROLLO DE UN PROTOTIPO DE APLICACIÓN WEB PARA EL ALQUILER DE EQUIPAMIENTO E INSTALACIONES EN UN CENTRO DEPORTIVO

Alumno: **Álvaro Aglio Sánchez**

Tutor: Prof. D. José Ramón Balsas Almagro
Depto.: Departamento de informática

Septiembre, 2016

Índice

Tablas

Ilustraciones

1.- Introducción

1.1.- Motivación

El presente trabajo fin de grado está dirigido a poner en práctica los conocimientos adquiridos en las asignaturas del grado en ingeniería informática. Se centra en un marco práctico, con el fin de ser algo "tangible" y relacionado con el futuro profesional.

El propósito del trabajo fin de grado es analizar, diseñar y desarrollar un prototipo de aplicación web para centros deportivos, que permita gestionar los usuarios registrados en la plataforma y las reservas de pistas y equipamiento deportivo.

La elección de la temática de la aplicación viene dada por el uso habitual de instalaciones deportivas, que me lleva a la necesidad de proponer el desarrollo de esta aplicación web. A día de hoy, para practicar deporte existen infinidad de centros deportivos. A la mayoría de ellos hay que desplazarse o hacer una llamada telefónica, para realizar la reserva de instalaciones y/o equipamiento.

Una aplicación web que gestione todo esto hace que sea mucho más fácil y cómodo para el usuario practicar deporte y conocer la disponibilidad de instalaciones y/o equipamiento de su centro deportivo habitual, además de facilitar el trabajo al personal del centro.

1.2.- Objetivos

Los objetivos que se han planteado para el desarrollo del proyecto son los siguientes:

- Realizar un estudio de las necesidades y soluciones para la gestión de recursos de un centro deportivo.

- Implementar un prototipo de aplicación web usando tecnologías JEE en el lado del servidor y una interfaz adaptativa basada en HTML5 en el lado del cliente.

- Uso del patrón de arquitectura Modelo-Vista-Controlador para el desarrollo de la aplicación, que permita separar las capas de los datos y la lógica de negocio de la interfaz del usuario y el módulo encargado de gestionar los eventos.

- Uso del framework SpringMVC para el desarrollo de la aplicación.

- Análisis, diseño, implementación y pruebas de la aplicación.

- Realización de una memoria con la documentación correspondiente al desarrollo de la aplicación.

1.3.- Metodología

Para el desarrollo de la aplicación web propuesta vamos a seguir una de las diversas metodologías ágiles, basadas en un desarrollo iterativo e incremental, en concreto la metodología *Scrum*.

Para ello haré un estudio previo del proyecto, identificando los requisitos principales, ya que necesitamos describir los *sprints* para seguir el desarrollo ágil. También vamos a realizar la estimación temporal y costes de desarrollo.

Scrum

Scrum [1] es un proceso ágil que nos permite centrarnos en ofrecer el más alto valor de negocio en el menor tiempo. Está especialmente indicado para proyectos en entornos complejos, donde se necesita obtener resultados pronto, los requisitos son cambiantes o están poco definidos, donde la innovación, la flexibilidad y la productividad son fundamentales [2].

En *Scrum* se realizan entregas parciales y regulares del producto final, a dichas entregas se les denomina *sprints* y tienen una duración de entre 2-4 semanas. Al finalizar cada *sprint*, se tiene software funcionando y se decide si liberarlo o seguir mejorándolo. El software es diseñado, codificado y testeado durante el *sprint*. En un

sprint no hay cambios, su duración se planea en función de cuánto tiempo podemos comprometernos a mantener los cambios fuera del *sprint*.

Los equipos de trabajo de Scrum están formados por:

- Product Owner: responsable del producto
- Scrum Master: responsable del funcionamiento de Scrum
- Team: equipo responsable del desarrollo

Ciclo de trabajo de *Scrum*:

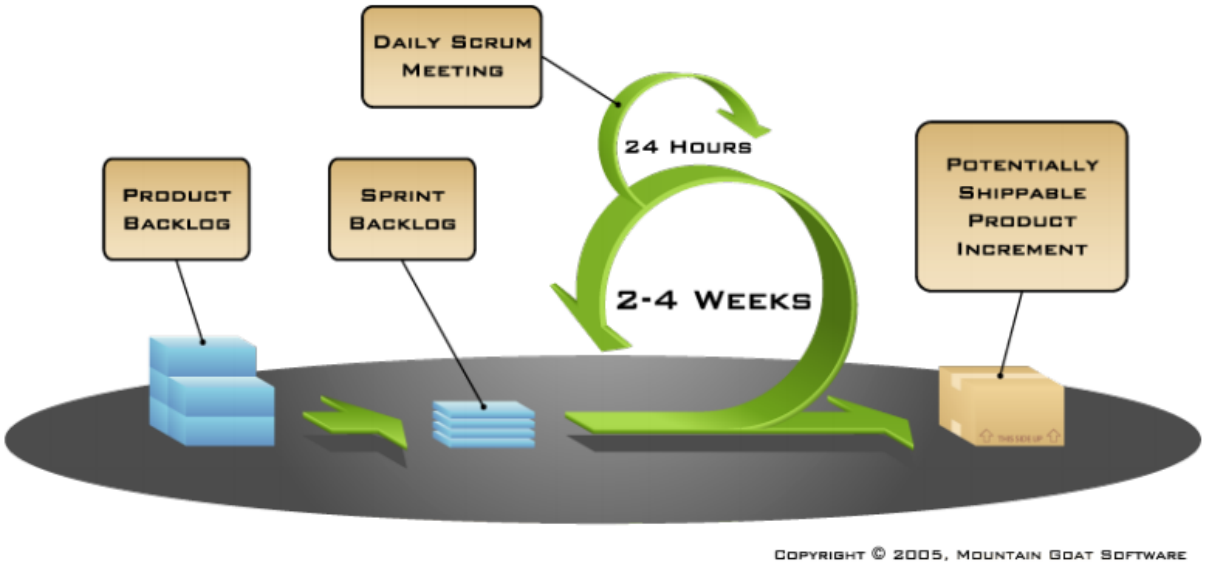

Ilustración 1- Ciclo de trabajo de Scrum

1. Toma de requisitos al cliente. Para cada requisito principal se crea un bloque de trabajo, llamado historia.

2. El cliente ordena los bloques de trabajo (historias) en una pila de producto, según su prioridad de entrega.

3. El equipo de trabajo toma un grupo de historias, con el que trabajan durante una iteración o *sprint*.

4. Una vez finalizado un *sprint* entregan al cliente el resultado del trabajo. Se vuelve al punto 2 hasta terminar la pila de producto.

1.4.- Estructura del documento

La documentación referente a este proyecto se divide en 6 bloques generales:

En el primer bloque hablamos de la motivación que nos lleva al desarrollo del proyecto y sus objetivos principales, así como la metodología que se usará para su realización.

En el segundo bloque realizamos un análisis de los centros deportivos existentes y de aplicaciones web similares. Además, se indica la propuesta de solución de forma justificada y dando una visión general de las metodologías y tecnologías que se van a utilizar. Se muestran los requisitos del sistema y las historias de usuario, la planificación de desarrollo, el estudio de viabilidad de nuestra aplicación y el modelo de dominio correspondiente al sistema que queremos desarrollar.

En el tercer bloque mostramos el diseño que tendrá nuestro sistema, en lo relativo al diseño de clases, el diseño de la capa de persistencia y diseño de la interfaz. Además se muestran los principales diagramas de secuencia de nuestra aplicación.

En el cuarto bloque hablamos de la implementación de nuestra aplicación. Indicando la arquitectura en la que se desarrolla el sistema y mostrando los principales detalles de la implementación.

En el quinto bloque realizamos una conclusión tras el desarrollo del proyecto, analizando todos los aspectos.

En el sexto y último bloque indicamos la bibliografía que hemos utilizado para el desarrollo de la documentación del proyecto.

Además, como información complementaria se incluyen: anexo de manual de usuario y anexo con la descripción de contenidos suministrados.

2.- Análisis

2.1.- Gestión de instalaciones deportivas

Hoy en día, el uso de instalaciones deportivas para practicar deporte viene siendo muy habitual. El cambio de mentalidad respecto a la práctica de ejercicio físico ha cambiado, ahora deportes como el fútbol, el pádel o el tenis son muy habituales a la hora de hacer deporte.

Los centros deportivos que ofrecen este tipo de deportes están en auge y, gracias a ellos, nos es más fácil practicar dichos deportes.

Además de ofrecernos pistas deportivas acondicionadas para la práctica de un deporte en particular, nos dan la posibilidad, en algunos casos, de usar material deportivo correspondiente a cada deporte, por lo que no hace falta que un usuario posea o compre el material necesario para practicar el deporte que quiera. Ahora bien, para poder realizar lo anteriormente explicado, el usuario tiene que reservar la pista y material necesario.

Es muy habitual ver a los usuarios acudir a su centro deportivo habitual o más cercano a consultar horarios y días disponibles para realizar las reservas pertinentes. Una aplicación que permita al usuario realizar todo esto de manera más cómoda usando su ordenador, tablet o smartphone, y desde cualquier lugar, facilita dicha tarea.

Debido a esto, una aplicación que ofrezca la posibilidad de reservar de manera adicional material deportivo para realizar el deporte que el usuario requiera, hace que el usuario tenga más predisposición a practicar deporte, aun no teniendo el material necesario, lo que aportaría algo nuevo y siendo un reclamo por parte de los centros deportivos hacia sus usuarios.

Dejando a un lado las necesidades del usuario, hay centros que incluso no tienen un sistema de gestión para llevar el control de reservas. Simplemente usan unos cuadrantes en papel donde apuntan las distintas pistas que se encuentran reservadas, por lo que para los trabajadores del centro deportivo, la aplicación sería

una ventaja, ya que les ayudaría a facilitar su trabajo en el día a día. Además, no todos los deportes tienen la misma duración, que normalmente es de 1 hora. Deportes como el pádel tienen una duración de 1 hora y media, por lo que el control de la reserva varía en función de su duración.

El personal del centro encargado de llevar el control necesita información de las reservas de pistas y materiales para la preparación de las instalaciones y la colocación o recogida del material. Por lo que mostrarle al encargado dicha información mediante la aplicación facilitaría su trabajo.

2.1.1 Aplicaciones similares

A continuación vamos a analizar algunos de los servicios web que nos proporcionan los distintos centros deportivos de la ciudad de Linares, en los que practico deporte habitualmente.

Pádel 5 Linares

Ilustración 2- Logo pádel5Linares (http://www.padel5linares.es/)

El centro deportivo Pádel 5 Linares, nos ofrece en su web la posibilidad de registrarnos como usuarios en su sistema de reservas.

Una vez registrados, el sistema no nos da de alta, el usuario recibe un correo en el que se le informa que debe acudir personalmente a las instalaciones y dejar una fotocopia del DNI. Entregado el documento requerido, el usuario recibe un número de socio y se le da de alta en el sistema.

Dado de alta en el sistema, el usuario puede seleccionar el deporte y la fecha requeridos y a continuación se le muestra la disponibilidad de las pistas, pudiendo éste realizar la reserva. El pago de la correspondiente reserva se realiza en las oficinas del centro. En caso de querer cancelar la reserva, el usuario tendrá que llamar por teléfono al centro con 24 horas de antelación, si se anula después de ese tiempo, se tendrá que abonar la cantidad correspondiente a la pista.

Bajo mi punto de vista, la opción de registrarse en el sistema desde la web del centro es ventajosa, pero el hecho de tener que desplazarse hasta el centro para aportar la documentación necesaria y así poder acceder al sistema, hace que el usuario pierda toda comodidad la primera vez que accede al sistema. Una vez dado de alta el usuario, la forma de realizar la reserva indicando fecha y deporte es clara e intuitiva. Un pequeño inconveniente aparece a la hora de realizar la cancelación de la reserva, donde el usuario no puede realizarla desde la web, sino que tiene que realizar una llamada al centro e indicarlo al encargado.

Ayuntamiento de Linares (Centros deportivos municipales)

Ilustración 3- Logo ayuntamiento de Linares (http://pistasdeportivas.ciudaddelinares.es/)

La página web del ayuntamiento de Linares nos ofrece información sobre la disponibilidad de las pistas de los centros deportivos municipales.
En este caso, tras seleccionar un deporte y una fecha, la web nos muestra la disponibilidad del deporte seleccionado en los distintos centros.

Dicha página no ofrece ningún servicio más, aparte del meramente informativo. Si el usuario quisiera realizar la reserva, tiene que dirigirse personalmente a las oficinas del centro deportivo.

Es una ventaja que ofrezca la información solicitada al usuario en distintos centros, sin embargo, no poder realizar ninguna operación más, como es el caso de realizar reservas, es un gran inconveniente.

Parque deportivo La garza

Ilustración 4- Parque deportivo "La garza" (http://www.lagarzaesdeporte.com/instalaciones/7)

La web del parque deportivo “La garza”, situada en Linares, nos ofrece un sistema de reservas.

Una vez registrado, el sistema le pide al usuario que seleccione el deporte que quiera practicar y la fecha. Realizada la consulta se muestran las pistas disponibles, que el usuario puede seleccionar para reservarlas y pagar mediante tarjeta bancaria. En caso de querer cancelar la reserva, el usuario tendrá que llamar al centro para comunicarlo, pudiendo hacer la cancelación directa o cambiar la fecha y hora de la reserva.

En mi opinión, el sistema de reservas del parque deportivo “La garza” es bastante completo, ya que nos permite realizar las operaciones de consulta y reserva de pistas, aunque no de material. Como pequeño inconveniente volvemos a tener la cancelación de la reserva, la cual hay que realizar llamando por teléfono al centro y comunicarlo al encargado.

2.2.- Propuesta de solución

Tras realizar un análisis exhaustivo sobre el tema del proyecto, la gestión de centros deportivos, y vistas algunas de las herramientas y aplicaciones que actualmente nos ofrecen diversas posibilidades sobre el tema, voy a proponer una

solución para resolver las diferentes necesidades por parte de los centros y los usuarios.

Al ser usuario asiduo de este tipo de centros deportivos y practicar deporte habitualmente, veo que existen pequeños inconvenientes para realizar las reservas de instalaciones, ya que casi siempre el usuario acaba yendo a las oficinas del centro a consultar horarios y realizar las pertinentes reservas.

Por lo tanto, se me plantea el objetivo de realizar un prototipo de aplicación web que pueda llevar el control de la gestión de un centro deportivo.

La idea de usar una plataforma web se basa en que además, cualquier usuario que desee practicar deporte pueda consultar horarios y realizar las reservas desde cualquier lugar, usando simplemente su ordenador, smartphone o tablet.

Para el desarrollo de la aplicación he decidido aplicar el patrón de arquitectura software Modelo-Vista-Controlador, con el fin de tener separados los datos y la lógica de negocio de la interfaz de usuario y el controlador de los eventos, generando así una aplicación robusta.

Además, esto nos permite que en un futuro tengamos ventajas a la hora de realizar cambios o incluir mejoras en la aplicación, gracias a su diseño modulable.

Para este patrón escogido vamos a usar el *framework SpringMVC* [3], que nos proporciona, entre otras, las siguientes ventajas:

- Proporciona código mucho más limpio, generando una ventaja a la hora de entender más rápidamente el código.
- Capacidad para integrarse con otras herramientas.
- Fácil configuración en archivos xml.
- Uso de anotaciones que nos ahorran código

Por estas razones, y por tener ya experiencia de uso en este *framework* al haberlo usado en distintas asignaturas de la carrera, he decidido incorporarlo al proyecto. Lo que contribuye a reducir los tiempos de desarrollo dada la complejidad de aprender a utilizar *frameworks* de estas características.

Para el diseño de la interfaz, voy a usar el *framework* libre *Bootstrap* [4], ya que es uno de los más populares que se usan hoy en día, aparte de su gran éxito gracias a su buen funcionamiento. Nos permite crear interfaces de usuario con CSS y JavaScript que hacen que la web se adapte al dispositivo donde el usuario la esté visualizando. Además tenemos infinidad de plantillas gratuitas que proporcionan interfaces adaptadas a los usuarios de hoy en día, lo que es una ventaja para el desarrollo del proyecto.

2.3.- Historias de usuario

En este apartado vamos a mostrar las historias de usuario [6] para el desarrollo de nuestro sistema.

Como historias de usuario entendemos una herramienta que usan las metodologías ágiles para obtener los requisitos de un proyecto a desarrollar, a partir de la información obtenida de los usuarios [7].

Son una manera simple y corta de describir una tarea concisa, evolucionan a lo largo de la vida del proyecto y se adaptan a los cambios de los requisitos [8].

Las historias de usuario se describen mediante:

- **Id:** identificador de la historia de usuario.
- **Título:** título descriptivo de la historia de usuario.
- **Descripción:** descripción sintetizada de la historia de usuario.
- **Estimación:** estimación del coste de implementación en unidades de desarrollo (estas unidades representarán el tiempo teórico de desarrollo que se estipule al comienzo del proyecto).

- **Prioridad:** prioridad en la implementación de la historia de usuario respecto al resto de las historias de usuario.
- **Dependencias:** Una historia de usuario no debería ser dependiente de otra historia, pero a veces es inevitable.

Las historias de usuario que forman nuestro proyecto son:

Id	Título	Descripción	Estimación	Prioridad
1	Configurar entorno de desarrollo	Como desarrollador, quiero configurar el entorno para poder empezar el desarrollo del proyecto	15	Alta
2	Registro en la aplicación	Como usuario, quiero registrarme en la aplicación para poder usarla	3	Alta
3	Iniciar sesión en la aplicación	Como usuario, quiero iniciar sesión en la aplicación para poder usarla	4	Alta
4	Consultar pistas	Como usuario, quiero consultar las pistas disponibles	6	Alta
5	Modificar datos	Como usuario, quiero modificar mis datos en la aplicación	3	Media
6	Visualizar reservas	Como usuario, quiero ver mis reservas realizadas en la aplicación	3	Baja
7	Realizar reservas	Como usuario, quiero realizar reservas de pistas y material	3	Alta
8	Cerrar sesión	Como usuario, quiero cerrar sesión para salir de la aplicación	3	Alta
9	Crear tipo de pista	Como administrador, quiero crear nuevos tipos de pista en la aplicación	3	Alta

10	Editar tipo de pista	Como administrador, quiero editar tipos de pista existentes en la aplicación	3	Media
11	Eliminar tipo de pista	Como administrador, quiero eliminar los tipos de pista existentes en la aplicación	3	Baja
12	Crear pista	Como administrador, quiero crear nuevas pistas en la aplicación	3	Alta
13	Editar pista	Como administrador, quiero editar pistas existentes en la aplicación	3	Media
14	Eliminar pista	Como administrador, quiero eliminar pistas existentes en la aplicación	3	Baja
15	Crear tipo de material	Como administrador, quiero crear nuevos tipos de materiales en la aplicación	3	Alta
16	Editar tipo de material	Como administrador, quiero editar los tipos de material existentes en la aplicación	3	Media
17	Eliminar tipo de material	Como administrador, quiero eliminar tipos de material existentes en la aplicación	3	Baja
18	Crear material	Como administrador, quiero crear nuevos materiales en la aplicación	3	Alta
19	Editar material	Como administrador, quiero editar material existente en la aplicación	3	Media
20	Eliminar material	Como administrador, quiero eliminar material existente en la aplicación	3	Baja
21	Crear usuario	Como administrador, quiero crear nuevos usuarios en la aplicación	3	Alta

22	Editar usuario	Como administrador, quiero editar usuarios existentes en la aplicación	3	Media
23	Eliminar usuario	Como administrador, quiero eliminar usuarios existentes en la aplicación	3	Baja
24	Visualizar usuario	Como administrador, quiero visualizar los usuarios existentes en la aplicación	3	Media
25	Crear reservas	Como administrador, quiero crear nuevas reservas en la aplicación	3	Alta
26	Eliminar reservas	Como usuario o administrador, quiero eliminar las reservas existentes en la aplicación	3	Alta
27	Visualizar reservas	Como administrador, quiero visualizar las reservas existentes en la aplicación	3	Media

Tabla 1- Historias de usuario

2.4.- Requisitos del sistema

En este apartado vamos a mostrar los requisitos que debe cumplir nuestro sistema, funcionales y no funcionales.

Como requisito funcional entendemos requisitos que especifican la funcionalidad que el sistema debe proporcionar a los usuarios. Son declaraciones de servicios que el sistema tiene que ofrecer, la forma en que el sistema debe responder a determinadas entradas y cómo el sistema debería comportarse en situaciones particulares [5].

Los requisitos funcionales que forman nuestro sistema son los siguientes:

Respecto al **usuario** de la aplicación:

- **Identificar usuario:** el usuario debe poder acceder al sistema introduciendo el nombre de usuario y la contraseña.
- **Registrar usuario:** el usuario debe poder registrarse en el sistema introduciendo los datos que le sean demandados.
- **Consultar disponibilidad pistas:** el usuario de la aplicación debe poder consultar la disponibilidad de las pistas estando y sin estar registrado en el sistema.
- **Modificar datos de usuario:** el usuario, una vez identificado en el sistema, debe poder modificar sus datos en la aplicación.
- **Visualizar reservas de usuario:** el usuario, una vez identificado en el sistema, debe poder visualizar las reservas realizadas.
- **Cerrar sesión:** el usuario debe poder cerrar sesión y salir del sistema.
- **Realizar reservas:** el usuario, una vez registrado, debe poder realizar reservas de pistas y materiales.
- **Eliminar reservas:** el usuario debe poder cancelar sus reservas.

Respecto al **administrador** de la aplicación:

- **Crear tipo de pista:** el administrador debe poder crear un nuevo tipo de pista en el sistema.
- **Editar tipo de pista:** el administrador debe poder editar un tipo de pista existente en el sistema.
- **Eliminar tipo de pista:** el administrador debe poder eliminar un tipo de pista existente en el sistema.
- **Crear pista:** el administrador debe poder crear una nueva pista en el sistema.
- **Editar pista:** el administrador debe poder editar una pista existente en el sistema.
- **Eliminar pista:** el administrador debe poder eliminar una pista existente en el sistema.

- **Visualizar usuarios:** el administrador debe poder ver los usuarios registrados en el sistema.
- **Crear usuarios:** el administrador debe poder crear usuarios en el sistema.
- **Editar usuarios:** el administrador debe poder editar usuarios existentes en el sistema.
- **Eliminar usuarios:** el administrador debe poder eliminar usuarios existentes en el sistema.
- **Visualizar reservas:** el administrador debe poder ver las reservas existentes en el sistema.
- **Crear reservas:** el administrador debe poder crear reservas en el sistema.
- **Eliminar reservas:** el administrador debe poder eliminar reservas del sistema.
- **Crear tipo de material:** el administrador debe poder crear un nuevo tipo de material en el sistema.
- **Editar tipo de material:** el administrador debe poder editar un tipo de material existente en el sistema.
- **Eliminar tipo de material:** el administrador debe poder eliminar un tipo de material existente en el sistema.
- **Crear material:** el administrador debe poder crear un nuevo material en el sistema.
- **Editar material:** el administrador debe poder editar un material existente en el sistema.
- **Eliminar material:** el administrador debe poder eliminar un material existente en el sistema.

Como requisito no funcional entendemos requisitos relacionados con la calidad del sistema que se va a desarrollar.

Los requisitos no funcionales que forman nuestro sistema son los siguientes:

- **Disponibilidad:** el sistema debe estar un alto porcentaje de tiempo en funcionamiento y disponible para usarlo.
- **Integridad:** el sistema debe tener un grado aceptable de protección frente a accesos no autorizados, violación de datos o pérdida de información.

- **Usabilidad:** el sistema debe ser fácil de usar para cualquier usuario, sin someter al usuario a un alto grado de esfuerzo para preparar las entradas o interpretar las salidas del sistema.
- **Confiabilidad:** el sistema debe tener alta probabilidad de funcionar sin fallos durante un periodo específico de tiempo.

2.5.- Planificación inicial de tareas

En este apartado vamos a mostrar la planificación inicial de tareas para el desarrollo de nuestra aplicación web. Dichas tareas las vamos a extraer de las historias de usuario definidas anteriormente, que a su vez se van a dividir en subtareas y vamos a asignar una estimación mediante puntos de historia.

Los puntos de historia son una medida de la complejidad de un requisito, comparado con la duración necesaria para completar ese requisito. Son una unidad de medida para expresar el tamaño de un trabajo utilizando valores relativos [9].

1 punto de historia
=
1 jornada de trabajo de un miembro del equipo en dedicación exclusiva

Ilustración 5- Estimación en puntos de historia

Es decir, si una historia tiene 3 puntos significa que si un miembro del equipo (sólo uno) se dedica a trabajar para la historia, sin tener que dedicar tiempo a ninguna otra cuestión (ni reuniones, ni llamadas, ni distracciones en general), necesitará 3 jornadas de trabajo, donde la duración de la jornada de trabajo es de 6 horas.

Para estimar los puntos de historia vamos a utilizar la técnica de *planning poker*, método descrito por **James Grenning** en 2002, que combina el juicio experto, la estimación por analogía y la desagregación en un enfoque que consigue estimaciones rápidas pero fiables. Utiliza la serie de **Fibonacci: 1, 2, 3, 5, 8…** para estimar las distintas historias [10].

A continuación se muestran las tablas de las tareas con sus respectivas subtareas, estimadas en puntos de historia y quién se encarga de realizarlas.

Id	Tarea	Estimación	Rol
1	**Configurar entorno de desarrollo**	**15**	
1.1	Configurar SpringMVC	3	Programador
1.2	Análisis y diseño de la arquitectura del sistema	5	Analista
1.3	Crear controladores	1	Programador
1.4	Crear DAO´s	1	Programador
1.5	Crear clases	2	Programador
1.6	Crear base de datos	3	Programador

Tabla 2- Tarea configurar entorno de desarrollo

Id	Tarea	Estimación	Rol
2	**Registro en la aplicación**	**3**	
2.1	Configurar controlador	1	Programador
2.2	Configurar Dao	1	Programador
2.3	Configurar vista	1	Programador

Tabla 3- Tarea registro en la aplicación

Id	Tarea	Estimación	Rol
3	**Iniciar sesión en la aplicación**	**4**	
3.1	Configurar SpringSecurity	3	Programador
3.2	Configurar vista	1	Programador

Tabla 4- Tarea iniciar sesión en la aplicación

Id	Tarea	Estimación	Rol
4	**Consultar pistas**	**6**	
4.1	Configurar controlador	1	Programador
4.2	Configurar Dao	2	Programador
4.3	Configurar vista	3	Programador

Tabla 5- Tarea consultar pistas

Id	Tarea	Estimación	Rol
5	**Modificar datos**	**3**	
5.1	Configurar controlador	1	Programador
5.2	Configurar Dao	1	Programador
5.3	Configurar vista	1	Programador

Tabla 6- Tarea modificar datos

Id	Tarea	Estimación	Rol
6	**Visualizar reservas**	**3**	
6.1	Configurar controlador	1	Programador
6.2	Configurar Dao	1	Programador
6.3	Configurar vista	1	Programador

Tabla 7- Tarea visualizar reservas

Id	Tarea	Estimación	Rol
7	**Realizar reservas**	**3**	
7.1	Configurar controlador	1	Programador
7.2	Configurar Dao	1	Programador
7.3	Configurar vista	1	Programador

Tabla 8- Tarea realizar reservas

Id	Tarea	Estimación	Rol
8	**Cerrar sesión**	**3**	
8.1	Configurar controlador	1	Programador
8.2	Configurar Dao	1	Programador
8.3	Configurar vista	1	Programador

Tabla 9- Tarea cerrar sesión

Id	Tarea	Estimación	Rol
9	**Crear tipo de pista**	**3**	
9.1	Configurar controlador	1	Programador
9.2	Configurar Dao	1	Programador
9.3	Configurar vista	1	Programador

Tabla 10- tarea crear tipo de pista

Id	Tarea	Estimación	Rol
10	**Editar tipo de pista**	**3**	
10.1	Configurar controlador	1	Programador
10.2	Configurar Dao	1	Programador
10.3	Configurar vista	1	Programador

Tabla 11- Tarea editar tipo de pista

Id	Tarea	Estimación	Rol
11	**Eliminar tipo de pista**	**3**	
11.1	Configurar controlador	1	Programador
11.2	Configurar Dao	1	Programador
11.3	Configurar vista	1	Programador

Tabla 12- Tarea eliminar tipo de pista

Id	Tarea	Estimación	Rol
12	**Crear pista**	**3**	
12.1	Configurar controlador	1	Programador
12.2	Configurar Dao	1	Programador
12.3	Configurar vista	1	Programador

Tabla 13- Tarea crear pista

Id	Tarea	Estimación	Rol
13	**Editar pista**	**3**	
13.1	Configurar controlador	1	Programador
13.2	Configurar Dao	1	Programador
13.3	Configurar vista	1	Programador

Tabla 14- Tarea editar pista

Id	Tarea	Estimación	Rol
14	**Eliminar pista**	**3**	
14.1	Configurar controlador	1	Programador
14.2	Configurar Dao	1	Programador
14.3	Configurar vista	1	Programador

Tabla 15- Tarea eliminar pista

Id	Tarea	Estimación	Rol
15	**Crear tipo de material**	**3**	
15.1	Configurar controlador	1	Programador
15.2	Configurar Dao	1	Programador
15.3	Configurar vista	1	Programador

Tabla 16- Tarea crear tipo de material

Id	Tarea	Estimación	Rol
16	**Editar tipo de material**	**3**	
16.1	Configurar controlador	1	Programador
16.2	Configurar Dao	1	Programador
16.3	Configurar vista	1	Programador

Tabla 17- Tarea editar tipo de material

Id	Tarea	Estimación	Rol
17	**Eliminar tipo de material**	**3**	
17.1	Configurar controlador	1	Programador
17.2	Configurar Dao	1	Programador
17.3	Configurar vista	1	Programador

Tabla 18- Tarea eliminar tipo de material

Id	Tarea	Estimación	Rol
18	**Crear material**	**3**	
18.1	Configurar controlador	1	Programador
18.2	Configurar Dao	1	Programador
18.3	Configurar vista	1	Programador

Tabla 19- Tarea crear material

Id	Tarea	Estimación	Rol
19	**Editar material**	**3**	
19.1	Configurar controlador	1	Programador
19.2	Configurar Dao	1	Programador
19.3	Configurar vista	1	Programador

Tabla 20- Tarea editar material

Id	Tarea	Estimación	Rol
20	**Eliminar material**	**3**	
20.1	Configurar controlador	1	Programador
20.1	Configurar Dao	1	Programador
20.3	Configurar vista	1	Programador

Tabla 21- Tarea eliminar material

Id	Tarea	Estimación	Rol
21	**Crear usuario**	**3**	
21.1	Configurar controlador	1	Programador
21.2	Configurar Dao	1	Programador
21.3	Configurar vista	1	Programador

Tabla 22- Tarea crear usuario

Id	Tarea	Estimación	Rol
22	**Editar usuario**	**3**	
22.1	Configurar controlador	1	Programador
22.2	Configurar Dao	1	Programador
22.3	Configurar vista	1	Programador

Tabla 23- Tarea editar usuario

Id	Tarea	Estimación	Rol
23	**Eliminar usuario**	**3**	
23.1	Configurar controlador	1	Programador
23.2	Configurar Dao	1	Programador
23.3	Configurar vista	1	Programador

Tabla 24- Tarea eliminar usuario

Id	Tarea	Estimación	Rol
24	**Visualizar usuario**	**3**	
24.1	Configurar controlador	1	Programador
24.2	Configurar Dao	1	Programador
24.3	Configurar vista	1	Programador

Tabla 25- Tarea visualizar usuario

Id	Tarea	Estimación	Rol
25	**Crear reservas**	**3**	
25.1	Configurar controlador	1	Programador
25.2	Configurar Dao	1	Programador
25.3	Configurar vista	1	Programador

Tabla 26- Tarea crear reservas

Id	Tarea	Estimación	Rol
26	**Eliminar reservas**	**3**	
26.1	Configurar controlador	1	Programador
26.2	Configurar Dao	1	Programador
26.3	Configurar vista	1	Programador

Tabla 27- Tarea eliminar reservas

Id	Tarea	Estimación	Rol
27	**Visualizar reservas**	**3**	
27.1	Configurar controlador	1	Programador
27.2	Configurar Dao	1	Programador
27.3	Configurar vista	1	Programador

Tabla 28- Tarea visualizar reservas

Después de dividir todas las tareas en sus respectivas subtareas y realizar una estimación por puntos de historia, el total de puntos de historia del proyecto es de 97.

2.5.1.- Planificación de iteraciones

Tras la estimación realizada en el apartado anterior obtenemos 97 puntos de historia. El equipo de desarrollo del proyecto está formado por una persona.

Scrum estima que un sprint se debe realizar entre 2 y 4 semanas, por lo que vamos a estimar que nuestros *sprints* serán de 3 semanas, que corresponden a 15 jornadas de trabajo efectivo por *sprint*.

Al estar formado por una persona el equipo de desarrollo y no tener tiempo exclusivo para el desarrollo del proyecto, vamos a estimar en 10 puntos de historia el

desarrollo de cada sprint, para así tener un pequeño “colchón” y destinarlo a acciones como desplazamientos a la universidad o consultar dudas sobre el proyecto.

Sprint	Inicio	Fin	Puntos de historia	Velocidad
1	11/Enero/2016	29/Enero/2016	10	66%
2	1/Febrero/2016	19/Febrero/2016	10	66%
3	22/Febrero/2016	11/Marzo/2016	10	66%
4	14/Marzo/2016	1/Abril/2016	10	66%
5	4/Abril/2016	22/Abril/2016	10	66%
6	25/Abril/2016	13/Mayo/2016	10	66%
7	16/Mayo/2016	3/Junio/2016	10	66%
8	6/Junio/2016	24/Junio/2016	10	66%
9	27/Junio/2016	15/Julio/2016	10	66%
10	18/Julio/2016	5/Agosto/2016	7	46%

Tabla 29- Iteraciones

Tras realizar las estimaciones necesarias, calcular los *sprints* y asignar los puntos de historia, el proyecto tiene una fecha de comienzo del 11 de Enero de 2016 y una fecha de finalización del 5 de Agosto de 2016. La velocidad del proyecto es del 66% (10 puntos de historia completados por *sprint* / 15 jornadas de trabajo), excepto en el último, donde es del 46% al completar los 7 puntos de historia restantes.

A continuación vamos a mostrar el diagrama *BurnDown* [11], que nos indica el tiempo que falta para completar el trabajo. En el eje de la “X” se encuentran el número de *sprints* y en el eje de la “Y” el número de puntos de historia.

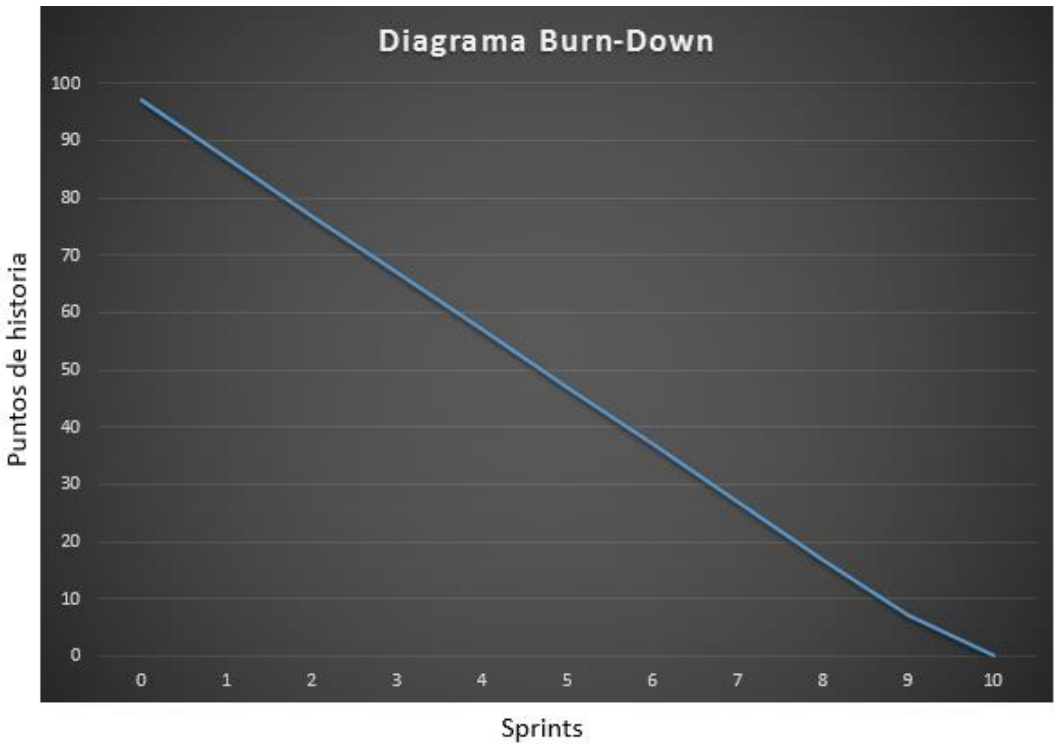

Ilustración 6- Diagrama Burn-Down

La planificación se ha visto cumplida en su totalidad. Pese a que hubo ciertos problemas a la hora de realizar la configuración del motor de persistencia, las estimaciones de las tareas correspondientes a realizar dicha configuración fueron suficientes para que la planificación no se viera afectada.

2.6.- Estudio de viabilidad

En este apartado vamos a mostrar un estudio de viabilidad del desarrollo del proyecto, es decir, los gastos que conlleva hacer realidad nuestro proyecto.

Software	Precio	Desarrollo del proyecto	Precio total
NetBeans 8.1	Gratis	7 meses	0€
Visual Paradigm International 12.2	Gratis	7 meses	0€
Axure RP 8	Gratis	1 mes	0€
Apache Tomcat 8.0.29	Gratis	7 meses	0€
Google Chrome 51.0.2704.103	Gratis	7 meses	0€
Microsoft Office 365 Hogar	10€ / mes	7 meses	70€
Total gastos			**70€**

Tabla 30- Gastos software

La tabla anterior nos muestra los gastos respecto al software que conlleva el desarrollo del proyecto, la mayoría de las herramientas software que vamos a utilizar son de código libre por lo que los gastos ascienden a **70€**.

Hardware	Precio	Vida útil	Desarrollo del proyecto	Precio total
Asus F555LJ-XX1010T 15.6" Año 2015	599€	6 años	7 meses	58,24€
Total gastos				**58,24€**

Tabla 31- Gastos hardware

La tabla anterior nos muestra el gasto en hardware para el desarrollo del proyecto, que en este caso sólo será un ordenador portátil, estimando una vida útil de 6 años y siendo el desarrollo del proyecto 7 meses, el gasto asciende a **58,24€**.

Tipo de profesional	Salario	Puntos de historia asignados	Duración	Salario total
Programador	1418,18€ / mes	92	7 meses	9927,26€
Analista	1559,26€ / mes	5	8 días	542,35€
		Total		**10469,61€**

Tabla 32- Gasto desarrolladores

La tabla anterior nos muestra el gasto en los dos tipos de profesionales que necesitamos para el desarrollo, analista y programador. En este caso, el analista no cobrará lo mismo que el programador, puesto que no realizan el mismo trabajo, al analista se le han asignado menos puntos de historia. El gasto el personal asciende a **10469,61€**. El salario al mes que perciben en este caso el programador y el analista han sido obtenidos del BOE en el siguiente enlace: https://www.boe.es/boe/dias/2016/03/15/pdfs/BOE-A-2016-2624.pdf

A continuación se muestra una tabla resumen de todos los gastos que conlleva el desarrollo del proyecto.

Gastos	Precio
Software	70€
Hardware	58,24€
Personal de desarrollo	10469,61€
Total	**10597,85€**

Tabla 33- Gastos totales

El total de gastos para el desarrollo completo del proyecto es de **10597,85€**.

2.7.- Modelo de dominio

En este apartado vamos a mostrar el diagrama que proponemos como solución al problema. Se trata de un diagrama de clases conceptuales [12] desde el punto de vista del análisis, por lo que más adelante pueden ser modificados elementos como relaciones o entidades.

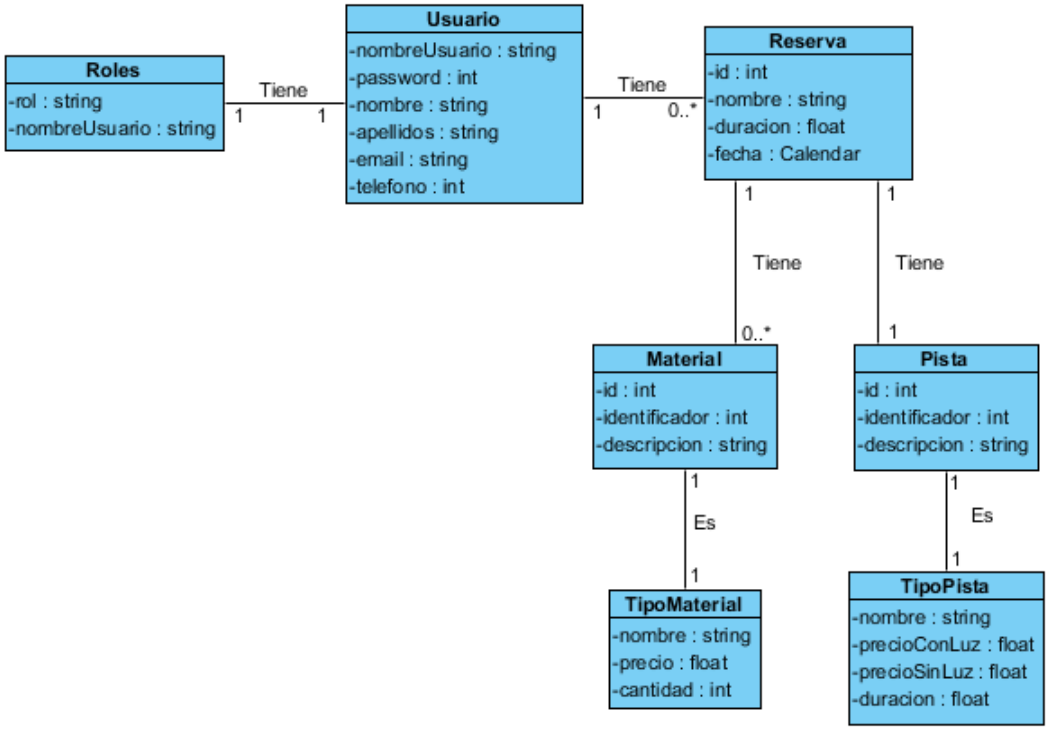

Ilustración 7- Modelo de dominio

Como podemos ver, las entidades que forman nuestro sistema son las siguientes:

- **Usuario**: entidad que almacena toda la información relativa a un usuario del sistema, formada por el nombre de usuario, password, nombre, apellidos, email y teléfono.
- **Rol**: entidad que almacena el rol del usuario en el sistema, formada por el nombre de usuario y rol.
- **Reserva**: entidad que almacena toda la información correspondiente a una reserva, formada por el id, nombre de usuario, duración y fecha.
- **Pista**: entidad que almacena información de la pista, formada por el id, el identificador y descripción.

- **Material**: entidad que almacena información del material, formada por el identificador y descripción.
- **TipoPista**: entidad que almacena la información referente a un tipo de pista, nombre del tipo de pista, precio con luz, precio sin luz y duración.
- **TipoMaterial**: entidad que almacena la información referente a un tipo de material, formada por el nombre del tipo de material, cantidad y precio.

3.- Diseño

En este apartado vamos a mostrar la parte de diseño de nuestro sistema; diagrama Entidad-Relación, diagrama de clases, diagramas de secuencia y diseño de la interfaz, siempre considerando los aspectos más relevantes.

3.1.- Diagrama de clases

En este apartado vamos mostrar los diagramas de clases correspondientes a nuestro sistema, incluyendo las relaciones existentes y aplicando el patrón MVC [13]. Dicho patrón divide la aplicación en modelo, vista y controlador, 3 partes que están débilmente acopladas entre sí, de manera que cualquier cambio que se pueda producir en alguna de ellas, no afecte prácticamente en nada a las demás.

A continuación se mostrarán los diagramas en forma de paquetes, para luego mostrar una visión de conjunto.

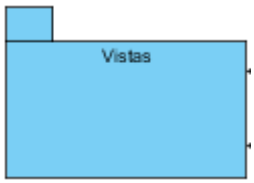

Ilustración 8- Paquete "vistas"

El paquete "Vistas" contiene todas las vistas necesarias para mostrar la información que se solicite a nuestro sistema. No se han reflejado las vistas, ya que en las aplicaciones web como es nuestro caso se tratan de ficheros plantilla que son procesados por los controladores para generar respuesta al usuario y se considera que no aportan realmente valor al diagrama en este apartado.

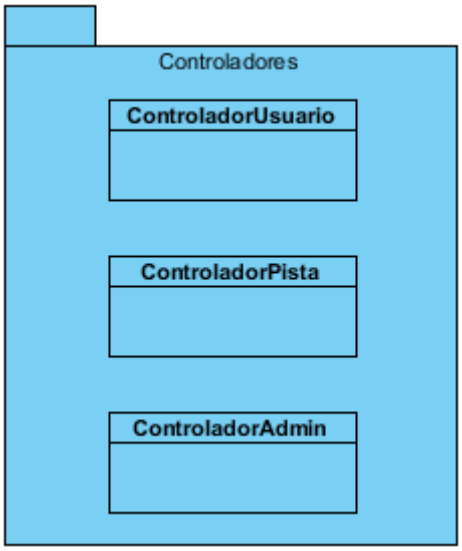

Ilustración 9- Paquete "controladores"

El paquete "controladores" contiene los controladores necesarios para el manejo del nuestro sistema, que en este caso son tres, "ControladorUsuario", "ControladorAdmin" y "ControladorPista". Los métodos no se han reflejado, ya que se considera que no aportan valor al diagrama en este apartado.

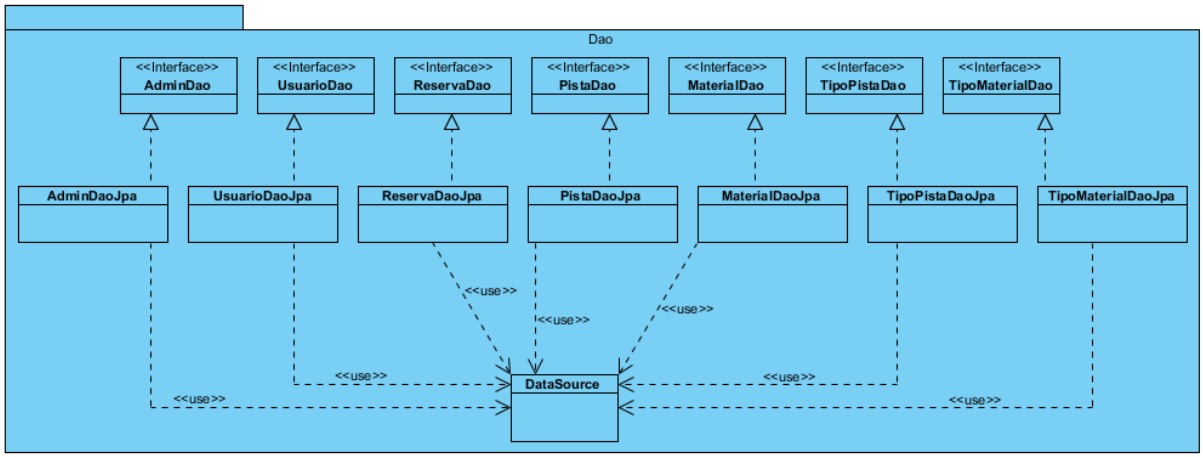

Ilustración 10- Paquete "Dao"

El paquete "Dao" contiene las interfaces y clases correspondientes para poder manejar la base de datos y por tanto hacer las gestiones CRUD en nuestro sistema. Por cada tabla de la base de datos relacional existirá un DAO [14]. En este caso, las interfaces "AdminDao", "UsuarioDao", "ReservaDao", "PistaDao", "MaterialDao", "TipoPistaDao", "TipoMaterialDao" y las clases "AdminDaoJpa", "UsuarioDaoJpa", "ReservaDaoJpa", "PistaDaoJpa", "MaterialDaoJpa", "TipoPistaDaoJpa" y "TipoMaterialDaoJpa" son las encargadas de interaccionar con la base de datos.

También podemos observar las relaciones que componen este paquete. No se han especificado métodos, ya que se considera que no aportan realmente valor al diagrama en este apartado.

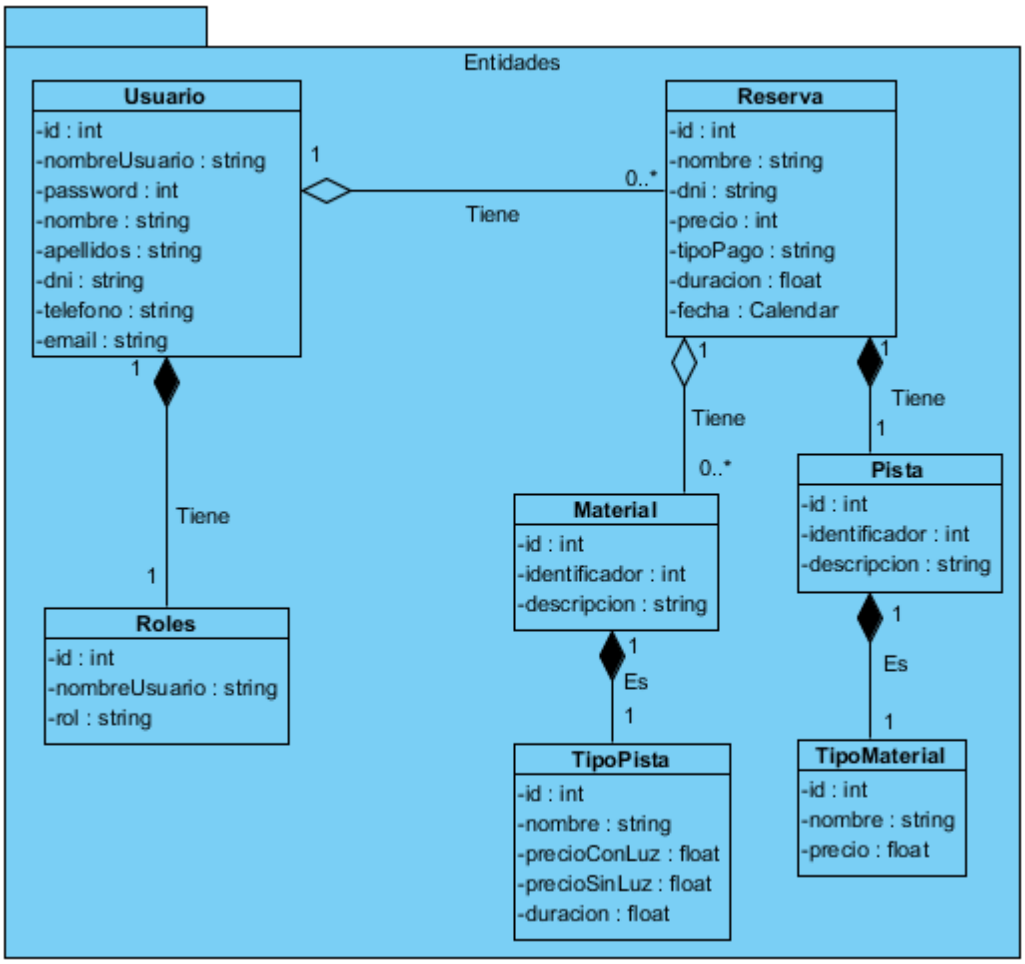

Ilustración 11- Paquete "entidades"

El paquete "Entidades" contiene las entidades que forman nuestro sistema. Respecto al modelo del dominio planteado en el apartado 2.7 de esta documentación, las entidades no han sufrido cambios, pero sí alguno de sus atributos, vistas las necesidades de la aplicación. También podemos observar las relaciones que existen entre las clases. Aunque no se muestra, por considerar que no aporta valor al diagrama en este apartado, todas las clases del diagrama implementan sus respectivos constructores parametrizados y no parametrizados, además de los métodos getters y setters de cada atributo que forman las clases.

Como entidades más relevantes, considero las correspondientes a “TipoPista” y “TipoMaterial”, la información relativa a una pista va relacionada con el tipo de deporte que se practique en ella, lo mismo ocurre para el material. Por ello he decidido crear dichas entidades que almacenarán la información correspondiente y se le asignan a las entidades “Pista” y “Material” respectivamente.

Las relaciones serán utilizadas posteriormente por el motor de persistencia que vamos a usar en el desarrollo del proyecto, en este caso JPA [15], para crear de forma prácticamente automática las tablas de la base de datos.

Tanto el paquete “Dao” como el paquete “Entidades” están a su vez en un paquete “Modelo”, completando así el patrón de arquitectura MVC que hemos aplicado al sistema. A continuación vamos a mostrar un ejemplo de cómo se relacionan los DAOs con sus entidades.

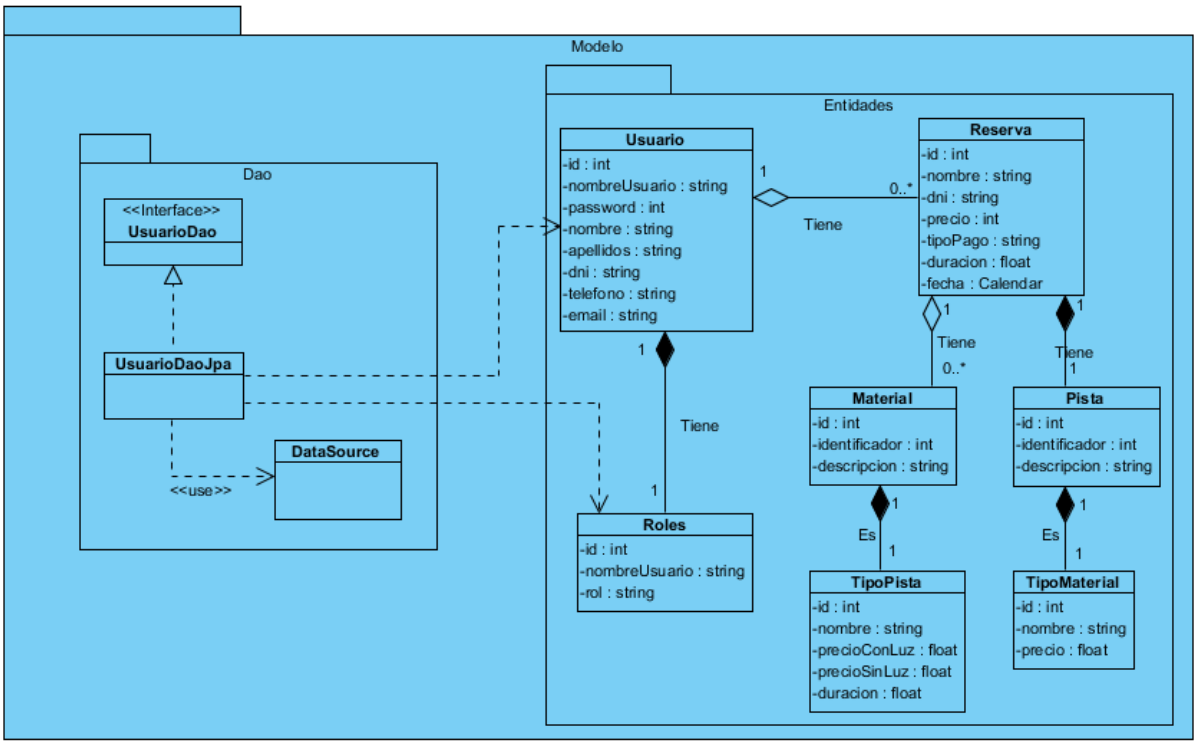

Ilustración 12- Paquete "modelo"

En la imagen 12 podemos ver cómo se relaciona, en este caso, el DAO correspondiente al usuario con las entidades “Usuario” y “Roles”. Por problemas de visión para poner todas las relaciones de todos los DAOs con sus entidades, la imagen es sólo un ejemplo de cómo se relacionan entre ellos, el diagrama completo se

encuentra en el CD adjunto, en el que se pueden ver todas las relaciones. Además, en el diagrama completo que se mostrará a continuación también se pueden observar dichas relaciones.

Una vez que hemos estudiado todos los bloques constituyentes, mostramos en el siguiente diagrama cómo se relacionan entre ellos.

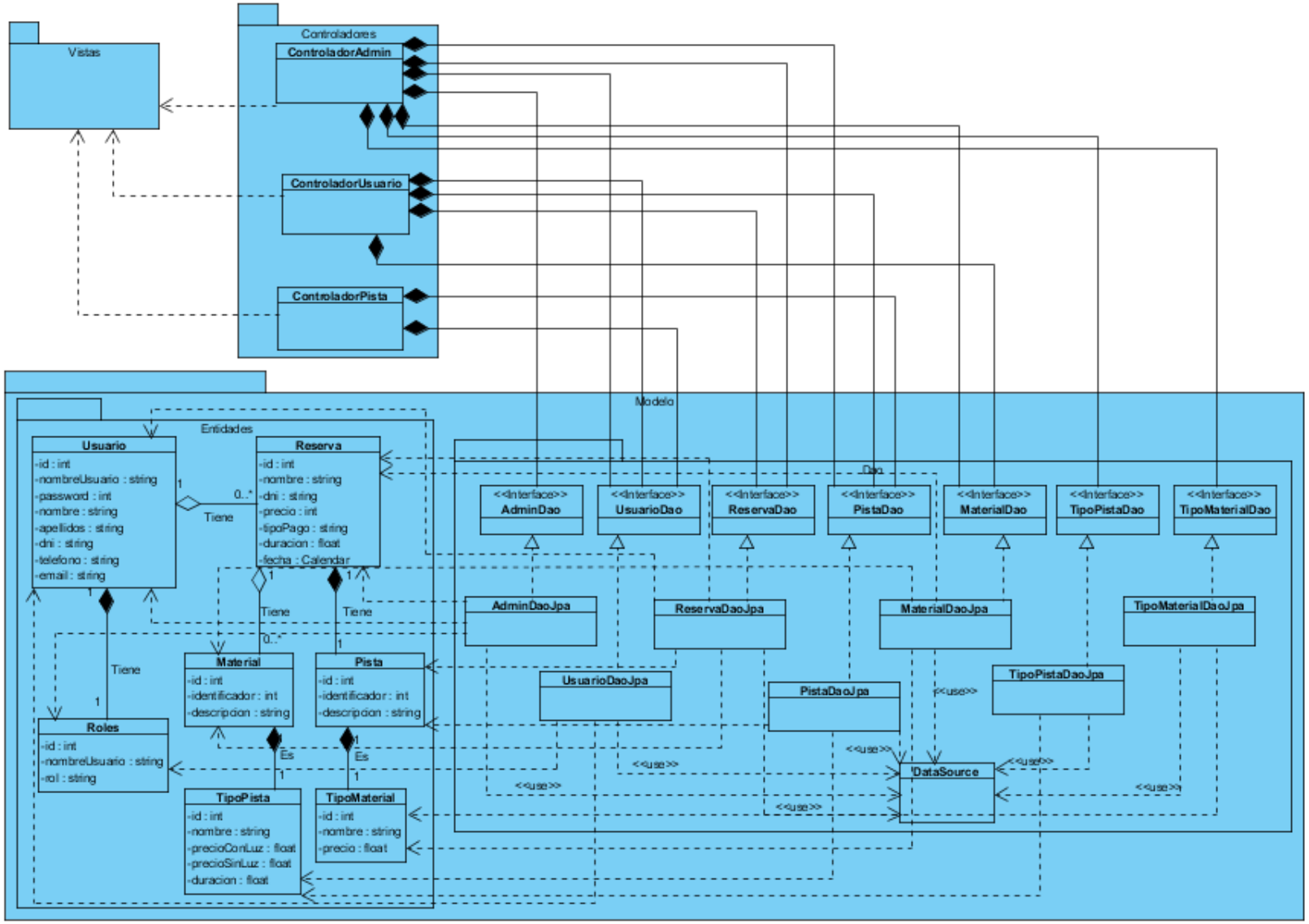

Ilustración 13- Diagrama de clases

3.2.- Diagrama Entidad-Relación

A continuación mostraremos el diseño que tendrá nuestro sistema en la base de datos, tanto las tablas como sus respectivas relaciones.

Las entidades que vamos a usar para el diseño del diagrama Entidad-Relación [16], serán las mismas descritas anteriormente en el apartado 3.1 (paquete "entidades", ilustración 11) de esta documentación con sus correspondientes atributos.

Las tablas que obtenemos para la BBDD son las siguientes:

Tabla Usuario:

Atributo	Tipo	Clave	Descripción
id	Integer(10)	Primaria	Identificador único del usuario
nombreUsuario	Varchar(50)	-	Nombre de usuario único para acceder al sistema
password	Varchar(50)	-	Password para acceder al sistema
nombre	Varchar(50)	-	Nombre de pila del usuario
apellidos	Varchar(50)	-	Apellidos del usuario
dni	Varchar(9)	-	Dni del usuario
teléfono	Varchar(9)	-	Teléfono de contacto del usuario
email	Varchar(50)	-	Email de contacto del usuario
rol_id	Integer(10)	Foránea	Identificador único del rol del usuario

Tabla 34- Tabla usuario BBDD

Tabla Roles:

Atributo	Tipo	Clave	Descripción
id	Integer(10)	Primaria	Identificador único del rol
nombreUsuario	Varchar(50)	-	Nombre de usuario único para acceder al sistema
rol	Varchar(50)	-	Rol del usuario en el sistema

Tabla 35- Tabla roles BBDD

Tabla Reserva:

Atributo	Tipo	Clave	Descripción
id	Integer(10)	Primaria	Identificador único de la reserva
nombre	Varchar(50)	-	Nombre del propietario de la reserva
dni	Varchar(9)	-	Dni del propietario de la reserva
precio	Real(10)	-	Precio de la reserva
tipoPago	Varchar(50)	-	Tipo de pago con el que se realiza la reserva
duración	Integer(10)	-	Duración de la reserva
fecha	TimeStamp	-	Fecha de la reserva
pista_id	Integer(10)	Foránea	Identificador único de la pista de la reserva

Tabla 36- Tabla reserva BBDD

Tabla Pista:

Atributo	Tipo	Clave	Descripción
id	Integer(10)	Primaria	Identificador único de la pista para la BBDD
identificador	integer(10)	-	Identificador de la pista
descripción	Varchar(255)	-	Descripción de la pista
tipoPista_id	Varchar(50)	Foránea	Identificador único del tipo de pista

Tabla 37- Tabla pista BBDD

Tabla Material:

Atributo	Tipo	Clave	Descripción
id	Integer(10)	Primaria	Identificador único del material para la BBDD
identificador	integer(10)	-	Identificador de la pista
descripción	Varchar(255)	-	Descripción de la pista
tipoMaterial_id	Varchar(50)	Foránea	Identificador único del tipo de material

Tabla 38- Tabla material BBDD

Tabla TipoPista:

Atributo	Tipo	Clave	Descripción
id	Integer(10)	Primaria	Identificador único del tipo de pista para la BBDD
nombre	Varchar(50)	-	Nombre único del tipo de pista
duración	Real(10)	-	Duración en tiempo del tipo de pista
precioConLuz	Real(10)	-	Precio con luz del tipo de pista
precioSinLuz	Real(10)	-	Precio sin luz del tipo de la pista

Tabla 39- Tabla tipoPista BBDD

Tabla TipoMaterial:

Atributo	Tipo	Clave	Descripción
id	Integer(10)	Primaria	Identificador único del tipo de material para la BBDD
nombre	Varchar(50)	-	Nombre único del tipo de material
precio	Real(10)	-	Precio del tipo de material

Tabla 40- Tabla tipoMaterial BBDD

Tras explicar las entidades que forman nuestro sistema y ver cómo quedan las tablas en nuestra base de datos con las claves primarias y foráneas, a continuación vamos a mostrar cómo están relacionadas todas las tablas, para tener un total acceso a todas las entidades.

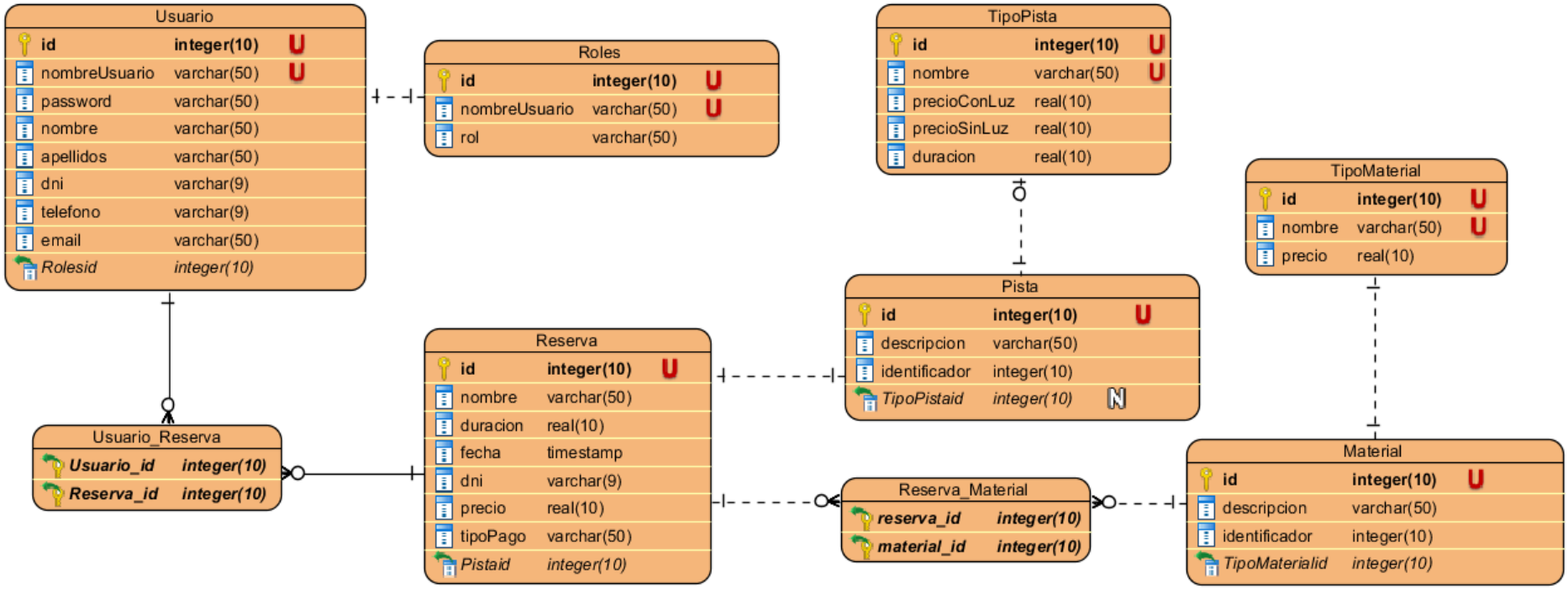

Ilustración 14- Diagrama Entidad-Relación

La tabla “Usuario” tiene una relación de uno a uno con la tabla “Rol”, ya que un usuario sólo tendrá un rol en el sistema, cuya clave foránea es el “id” de la tabla “Rol”. También, la tabla “Usuario” tiene una relación de uno a muchos con la tabla “Reserva”, ya que un usuario del sistema podrá tener varias reservas. En este caso, al ser una relación de uno a muchos, se genera una tabla intermedia que almacena las claves primarias de ambas tablas y así poder acceder desde un usuario a sus reservas.

La tabla “Reserva” tiene una relación de uno a uno con la tabla “Pista”, ya que una reserva tendrá una pista relacionada, cuya clave foránea es el “id” de la pista. También, la tabla “Reserva” tiene una relación de uno a muchos con la tabla “Material”, ya que una reserva podrá tener de uno a varios materiales. En este caso, también se genera una tabla intermedia que almacena las claves primarias de ambas tablas y así poder acceder desde una reserva a sus materiales.

La tabla “Pista” tiene una relación de uno a uno con la tabla “TipoPista”, ya que una pista sólo tendrá asociado un tipo de pista, cuya clave foránea es el “id” del tipo de pista.

La tabla “Material” tiene una relación de uno a uno con la tabla “TipoMaterial”, ya que un material sólo tendrá asociado un tipo de material, cuya clave foránea es el “id” del tipo de material.

3.2.1.- Normalización

La normalización [17] en bases de datos es un proceso que consiste en aplicar una serie de reglas a las tablas para conseguir evitar la redundancia de datos, disminuir los problemas de actualización de los datos en las tablas y proteger la integridad de los datos.

Existen 3 niveles de normalización que deben respetarse para poder decir que nuestra base de datos se encuentra normalizada. Dichos niveles son:

- **Primera forma normal (FN1):** una tabla está en primera forma normal si todos los atributos no clave, dependen funcionalmente de la clave. Es decir, no existen grupos repetitivos para un valor de clave.

- **Segunda forma normal (FN2):** una tabla está en segunda forma normal si está en primera forma normal y además todos los atributos que no pertenecen a la clave dependen funcionalmente de forma completa de ella.

- **Tercera forma normal (FN3):** una tabla está en tercera forma normal si está en segunda forma normal y además no existen atributos que no pertenecen a la clave que dependen transitivamente de la clave. Es decir, la tabla se encuentra en segunda forma normal y no existen atributos no clave que tengan dependencias funcionales entre sí.

Como podemos observar en el apartado anterior y tras explicar el proceso de normalización, nuestras tablas están normalizadas, ya que cumplen todas las características. No tenemos grupos repetitivos para un valor clave, los atributos no clave dependen funcionalmente de forma completa de la clave y no tenemos atributos no clave que sean dependientes entre sí.

3.3.- Diagramas de secuencia

En este apartado se muestran los diagramas más relevantes para comprender el funcionamiento del diseño planteado. El resto de diagramas tendrían una estructura similar y su comportamiento sería muy parecido a los propuestos.

Diagrama de secuencia: Registro

En este diagrama podemos ver como se registra un usuario en nuestro sistema. El controlador carga la vista que contiene el formulario de registro, tras introducir los datos el usuario, el controlador recoge los datos y los envía al DAO, que contiene el método encargado de crear el usuario en la base de datos. Una vez creado el usuario, el controlador muestra la página de login al usuario.

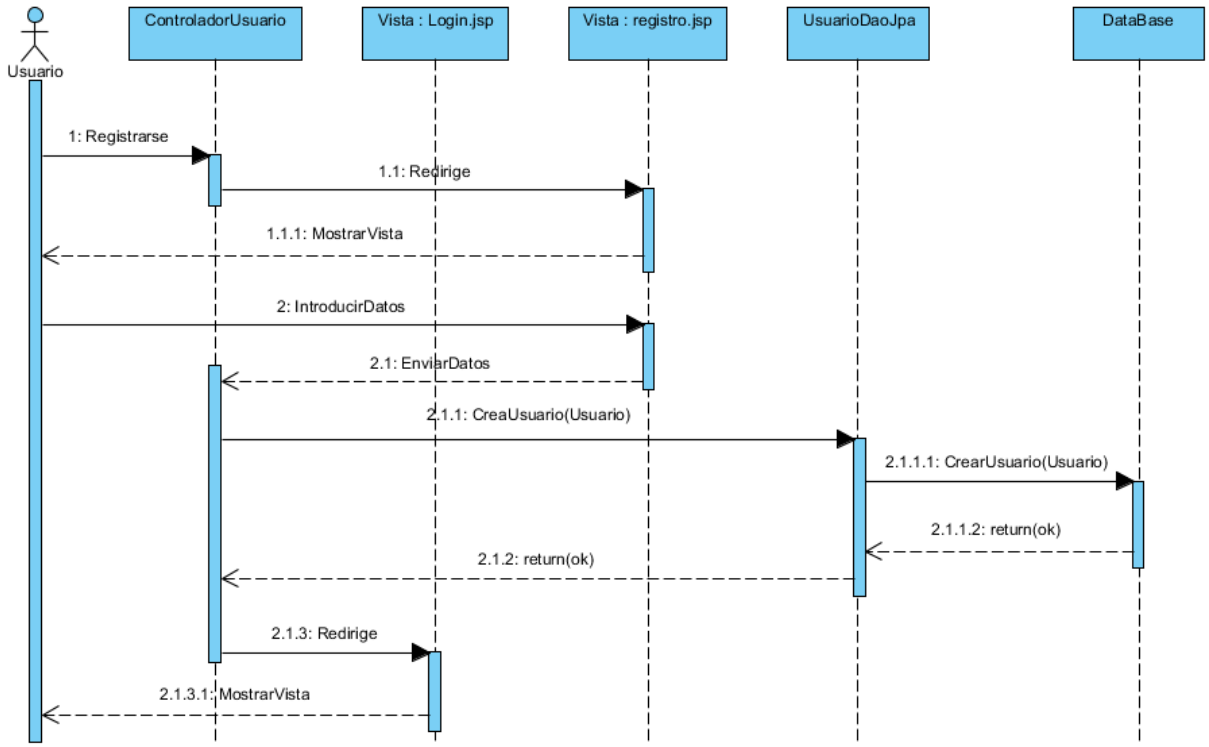

Ilustración 15- Diagrama de secuencia de registro

Diagrama de secuencia: Login

En este diagrama podemos ver como un usuario accede a nuestro sistema. El módulo de SpringSecurity es el encargado de mostrar la vista de login al usuario. Tras introducir los datos, se comprueba en la base de datos que el usuario y la contraseña son correctos. A continuación, se redirige al usuario a la página de su perfil.

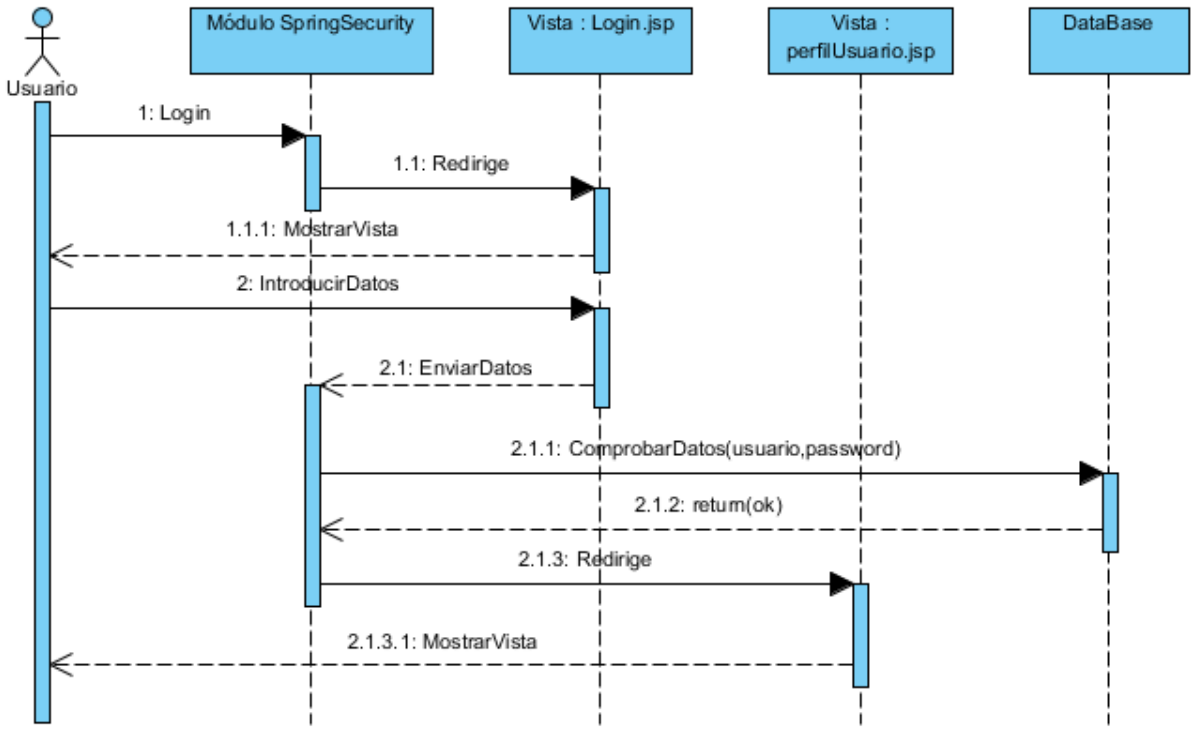

Ilustración 16- Diagrama de secuencia de login

Diagrama de secuencia: Consultar pista

En este diagrama podemos ver como un usuario consulta la disponibilidad de una pista. El controlador muestra la vista para que el usuario seleccione la consulta deseada. El controlador recoge los datos de la vista y los envía al DAO, cuyo método se encarga de comprobar la disponibilidad en la base de datos. Tras esto, el DAO envía los datos de la disponibilidad al controlador y éste redirige al usuario a la vista para mostrarle la información.

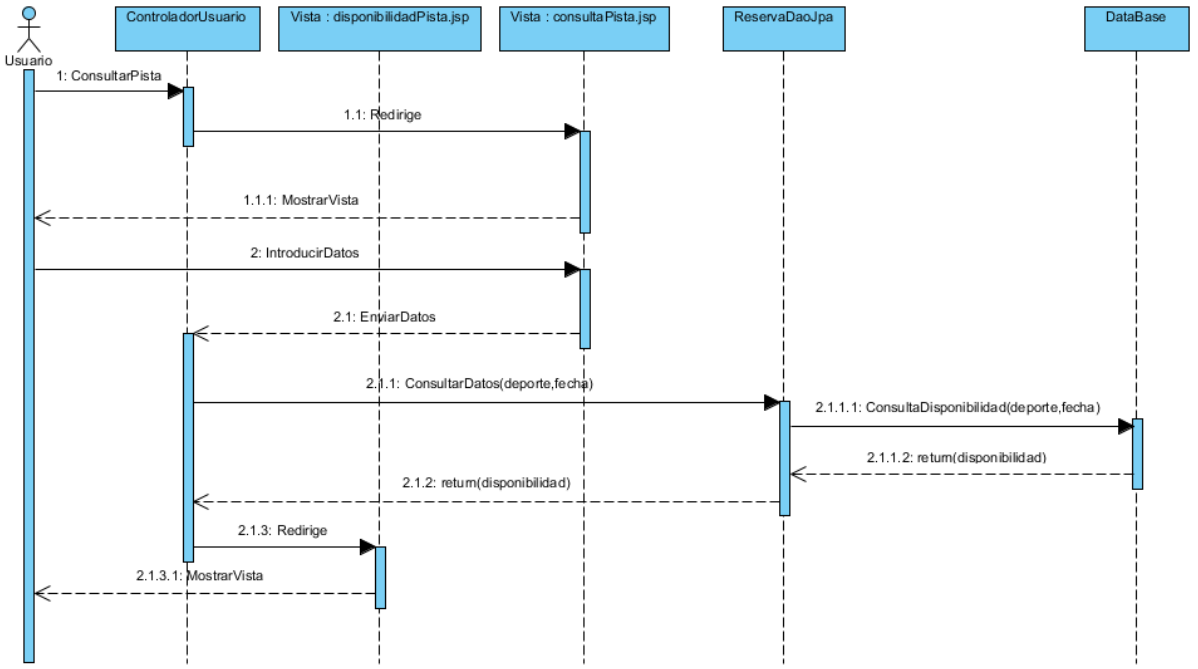

Ilustración 17- Diagrama de secuencia de consulta de pista

Diagrama de secuencia: Realizar reserva

En este diagrama podemos observar como un usuario realiza una reserva. Tras realizar la consulta de pista explicada anteriormente en la ilustración 17, el usuario selecciona la pista deseada. El controlador manda al DAO la fecha para comprobar el material disponible. El controlador muestra la vista con la pista seleccionada y el material disponible. El usuario selecciona el material y confirma la reserva. El controlador recoge los datos de la vista y se los pasa al DAO correspondiente para que se encargue de crear la reserva en la base de datos.

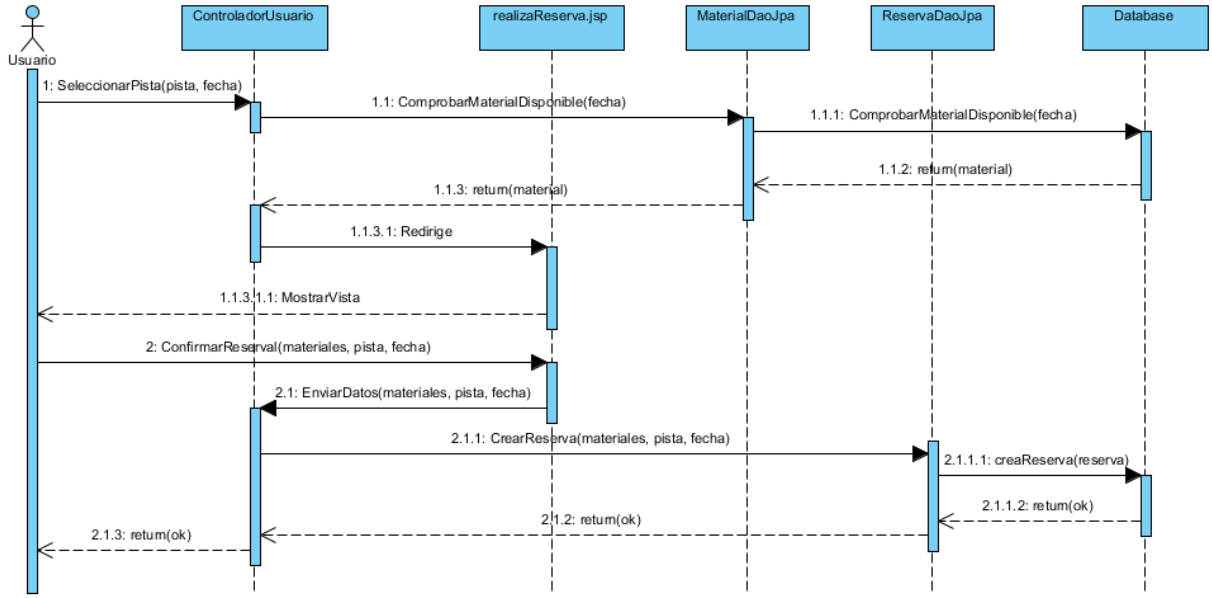

Ilustración 18- Diagrama de secuencia de realizar reserva

Diagrama de secuencia: Crear tipo de pista

En este diagrama podemos ver como el administrador del sistema crea un nuevo tipo de pista en la base de datos. El controlador redirige a la vista con el formulario para la creación del tipo de pista, tras recoger los datos, se envían al DAO, en el que existe un método que crea un nuevo tipo de pista en la base de datos.

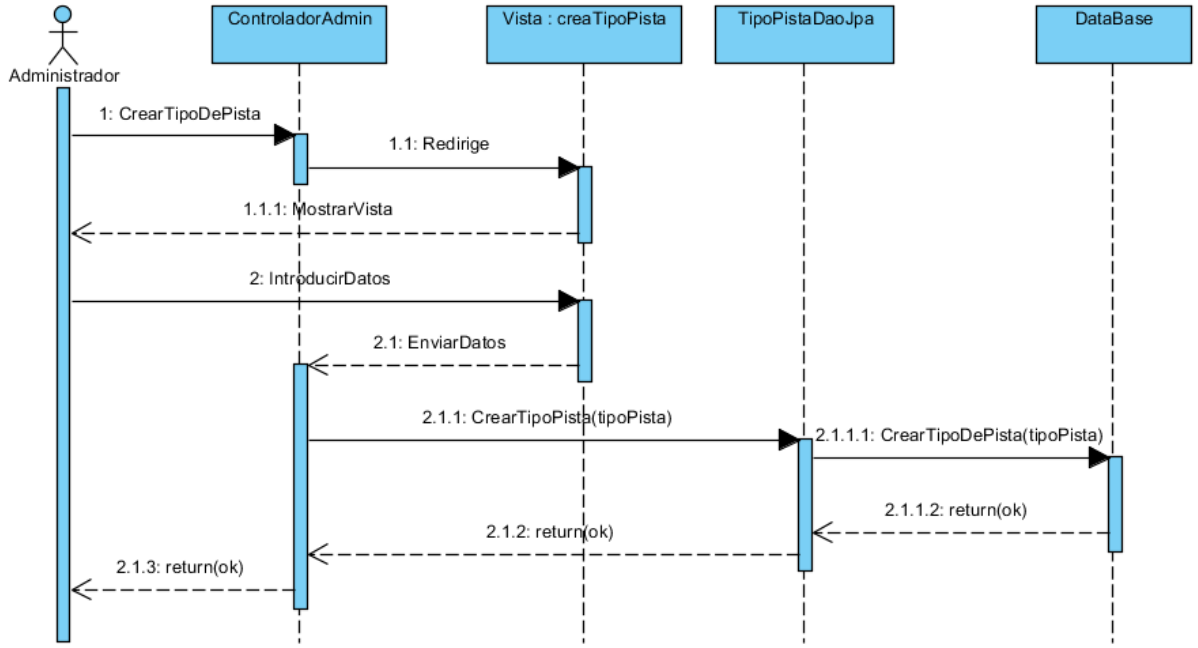

Ilustración 19- Diagrama de secuencia de crear tipo de pista

Diagrama de secuencia: Editar tipo de pista

En este diagrama podemos ver como el administrador del sistema edita un tipo de pista previamente creado en la base de datos. El controlador, mediante el DAO, obtiene todos los tipos de pista que hay creados en la base de datos y redirige al administrador a una vista para mostrarlos. El administrador selecciona el tipo de pista que desee editar y el controlador le redirige a un formulario de edición. Tras editar los datos, el controlador los recoge de la vista y los envía al DAO, cuyo método actualiza la información en la base de datos.

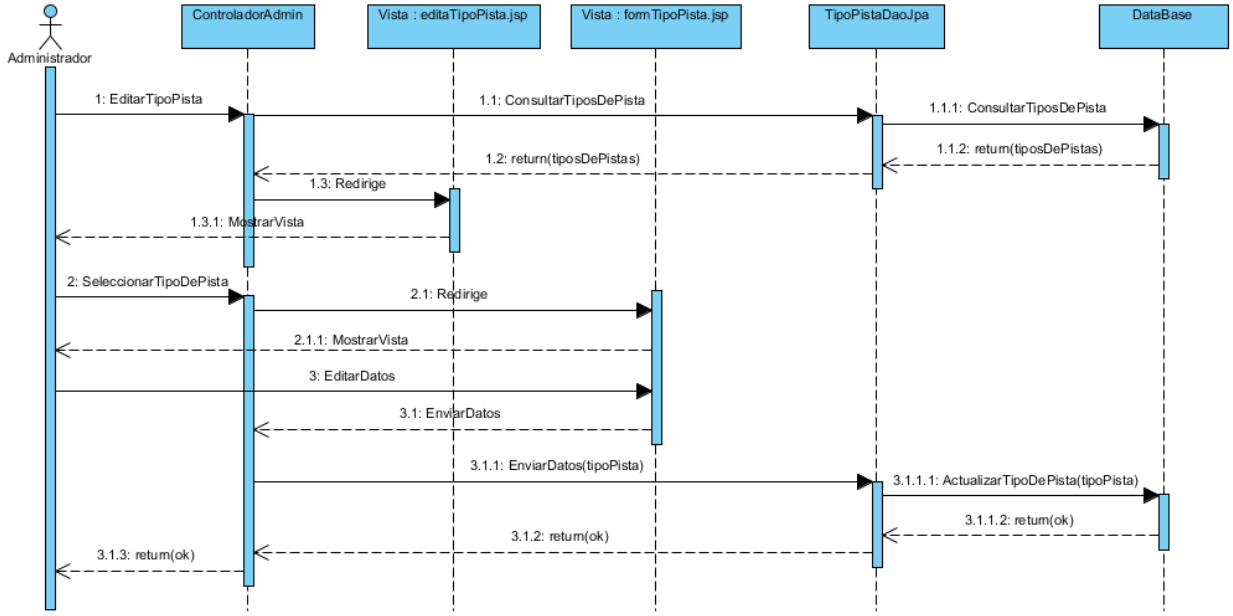

Ilustración 20- Diagrama de secuencia de editar tipo de pista

3.4.- Diseño de la interfaz

En este apartado vamos a mostrar el boceto o *wireframe* que será la interfaz gráfica de usuario que representa la aplicación web que queremos desarrollar.

Para realizar el boceto vamos a utilizar la herramienta software Axure RP [18]. Dicha herramienta es una aplicación que nos permite crear prototipos de lo que será nuestra aplicación. Axure RP está dirigido a aplicaciones web, aplicaciones de escritorio y aplicaciones móviles. Permite crear *wireframes* de forma sencilla e intuitiva. Aunque AxureRP es software de pago, nos permite un periodo de prueba de 30 días.

A continuación vamos a ver los prototipos que considero más importantes, ya que existen diferentes interfaces que realizan las mismas operaciones con distintos objetos de la aplicación, el resto de vistas son similares a las representadas en esta sección.

Interfaz home aplicación

Ilustración 21- Interfaz home aplicación

Boceto de la página principal de la aplicación web (ilustración 21), que está formada por:

- **Barra superior:** en la que se encuentra, en la parte derecha, el nombre de la aplicación, y a la izquierda tenemos las distintas opciones que se le ofrecen al usuario, consultar las pistas, iniciar sesión en la aplicación o registrarse en la aplicación.

- **Zona de anuncios:** que se encuentra en la parte central derecha, dedicada a anuncios de ofertas por parte del centro deportivo en cuestión.

Interfaz login - Usuario/administrador

Ilustración 22- Interfaz login

Boceto de la página de login (ilustración 22) que se muestra al usuario para iniciar sesión, que está formada por:

- **Barra superior:** en la que se encuentra, en la parte derecha, el nombre de la aplicación donde el usuario puede volver a la página principal, y a la izquierda tenemos la opción de acceder a registrarnos en la aplicación.

- **Formulario de login:** situado en la parte central, donde el usuario tendrá que rellenar usuario y contraseña para iniciar sesión.

Interfaz registro - Usuario

Ilustración 23- Interfaz registro

Boceto de la página de registro en la aplicación (ilustración 23), formado por:

- **Barra superior:** en la que se encuentra, en la parte derecha, el nombre de la aplicación donde el usuario puede volver a la página principal, y a la izquierda tenemos la opción de iniciar sesión.

- **Formulario de registro:** situado en la parte central, donde el usuario tendrá que rellenar los datos para darse de alta en la aplicación.

Interfaz perfil de usuario - Usuario

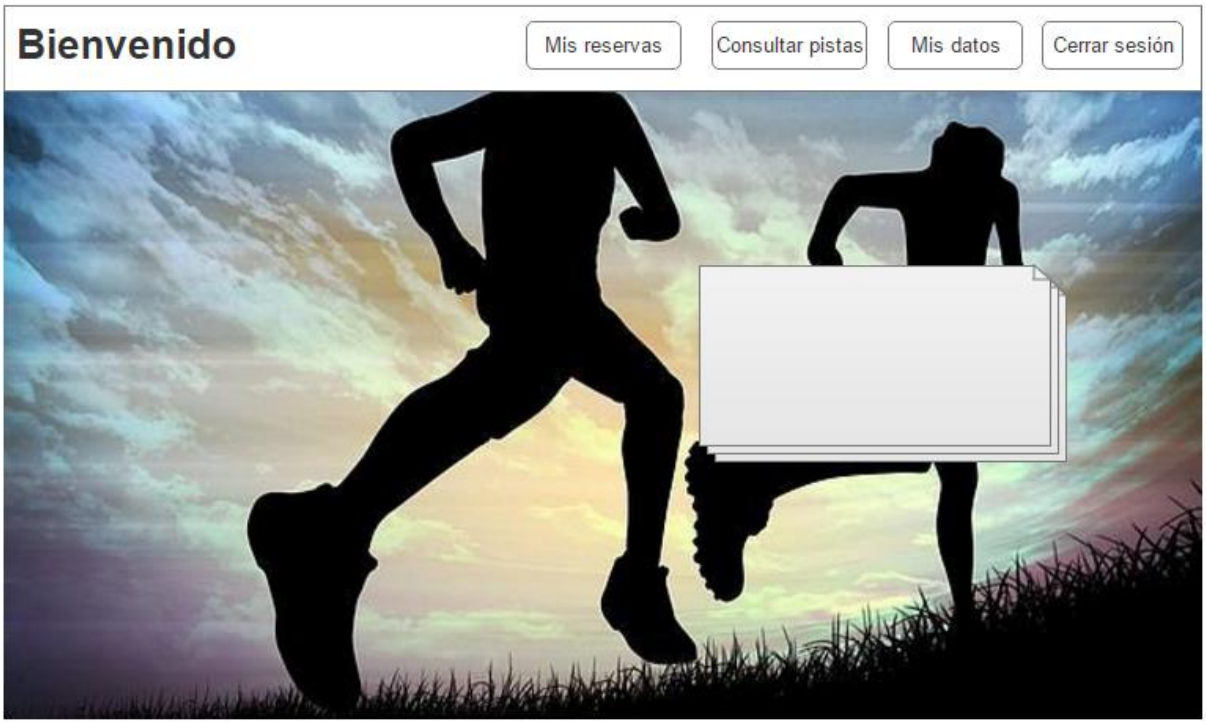

Ilustración 24- Interfaz perfil de usuario

Boceto de la página del perfil del usuario (ilustración 24), formada por:

- **Barra superior:** en la que se encuentra, en la parte derecha, un mensaje de bienvenida al usuario, y a la izquierda tenemos las distintas opciones que se le ofrecen al usuario, ver sus reservas, consultar las pistas, ver sus datos o cerrar sesión.

- **Zona de anuncios:** que se encuentra en la parte central derecha, dedicada a anuncios de ofertas por parte del centro deportivo en cuestión.

Interfaz datos de usuario - Usuario

Ilustración 25- Interfaz datos de usuario

Boceto de la página de datos del usuario (ilustración 25), formada por:

- **Barra superior:** en la que se encuentra, en la parte derecha, el nombre de la aplicación, y a la izquierda tenemos la opción de volver al perfil.

- **Formulario de datos de usuario:** situado en la parte central, donde el usuario modificará sus datos pertinentes para actualizarlos.

Interfaz mis reservas - Usuario

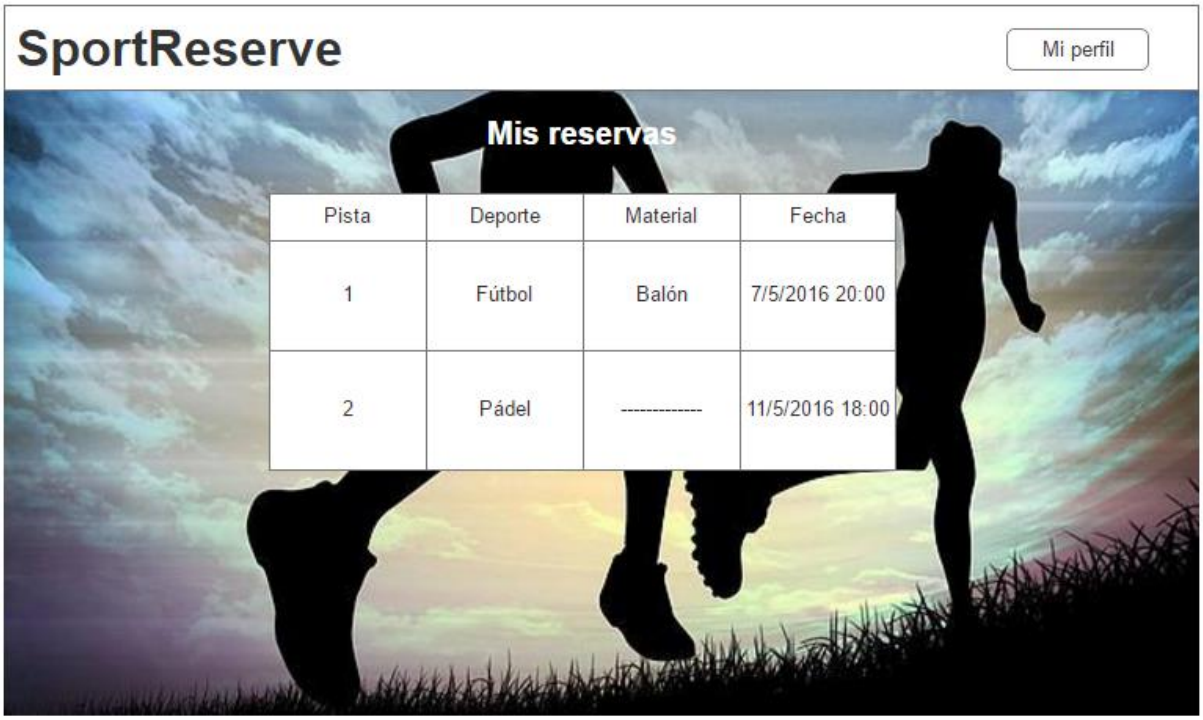

Ilustración 26- Interfaz reservas

Boceto de la página de reservas del usuario (ilustración 26), formada por:

- **Barra superior:** en la que se encuentra, en la parte derecha, el nombre de la aplicación, y a la izquierda tenemos la opción de volver al perfil.

- **Tabla:** situada en la parte central, donde se muestra al usuario las reservas que ha realizado con distinta información de cada una de ellas.

Interfaz consultar pistas - Usuario

Ilustración 27- Interfaz consultar pistas

Boceto de la página de consulta de pistas del usuario (ilustración 27), formada por:

- **Barra superior:** en la que se encuentra, en la parte derecha, el nombre de la aplicación, y a la izquierda tenemos la opción de volver al perfil.

- **Formulario de consulta:** situado en la parte central, donde se pide al usuario que especifique el deporte y la fecha, para realizar la consulta.

Interfaz disponibilidad de pistas - Usuario

Ilustración 28- Interfaz disponibilidad de pistas

Boceto de la página de disponibilidad de las pistas (ilustración 28), formada por:

- **Barra superior:** en la que se encuentra, en la parte derecha, el nombre de la aplicación, y a la izquierda tenemos la opción de volver al perfil.

- **Tabla:** situada en la parte central, donde se muestra al usuario la información de la disponibilidad de las pistas, con opción de pulsar sobre las casillas marcadas como “libre”.

Interfaz realizar reserva - Usuario

Ilustración 29- Interfaz realizar reserva

Boceto de la página de realizar la reserva (ilustración 29), formada por:

- **Barra superior:** en la que se encuentra, en la parte derecha, el nombre de la aplicación, y a la izquierda tenemos la opción de volver al perfil.

- **Tabla superior:** situada en la parte central superior, donde se muestra al usuario la información de la disponibilidad de materiales, con opción de pulsar sobre los *checkbox* para realizar dicha reserva.

- **Tabla inferior:** situada en la parte central inferior, donde se muestra al usuario la información de la reserva.

- **Botón:** situado en la parte central inferior, donde el usuario puede pulsar para confirmar la reserva.

Interfaz menú administrador - Administrador

Ilustración 30- Interfaz menú administrador

Boceto de la página principal de administrador (ilustración 30), formada por:

- **Barra superior:** en la que se encuentra, en la parte derecha, el nombre de la aplicación, y a la izquierda tenemos la opción de cerrar sesión.

- **Botones centrales:** en los que se encuentran las distintas opciones que se ofrecen al administrador de la aplicación.

Interfaz gestión pistas - Administrador

Ilustración 31- Interfaz gestión de pistas

Boceto de la página de gestión de pistas (ilustración 31), formada por:

- **Barra superior:** en la que se encuentra, en la parte derecha, el nombre de la aplicación, y a la izquierda tenemos la opción de volver al menú principal del administrador.

- **Botones centrales:** en los que se encuentran las distintas opciones que se ofrecen al administrador de la aplicación para gestionar las pistas.

Interfaz crear tipo de pista - Administrador

Ilustración 32- Interfaz crear tipo de pista

Boceto de la página de crear un tipo de pista (ilustración 32), formada por:

- **Barra superior:** en la que se encuentra, en la parte derecha, el nombre de la aplicación, y a la izquierda tenemos la opción de volver al menú principal del administrador o volver a la página de gestión de pistas.

- **Formulario de creación de un tipo de pista:** situado en la parte central, donde el administrador tendrá que rellenar los datos pertinentes para crear un tipo de pista.

Interfaz gestión de usuarios - Administrador

Ilustración 33- Interfaz gestión de usuarios

Boceto de la página de gestión de usuarios (ilustración 33), formada por:

- **Barra superior:** en la que se encuentra, en la parte derecha, el nombre de la aplicación, y a la izquierda tenemos la opción de volver menú principal del administrador.

- **Tabla:** situada en la parte central, donde se muestra distinta información sobre los usuarios de la aplicación y una columna específica con las distintas opciones que se le ofrecen al administrador, modificar el usuario, ver sus reservas o eliminarlo de la aplicación.

- **Botón de creación de usuario:** situado en la parte superior de la tabla.

Interfaz borrar material - Administrador

Ilustración 34- Interfaz borrar material

Boceto de la página de borrar un material (ilustración 34), formada por:

- **Barra superior:** en la que se encuentra, en la parte derecha, el nombre de la aplicación, y a la izquierda tenemos la opción de volver al menú principal del administrador o volver a la página de gestión de material.

- **Visualización de materiales:** situado en la parte central, donde se mostrarán al administrador los distintos materiales para eliminarlos de la aplicación.

Interfaz gestión material - Administrador

Ilustración 35- Interfaz gestión de material

Boceto de la página de gestión de material (ilustración 35), formada por:

- **Barra superior:** en la que se encuentra, en la parte derecha, el nombre de la aplicación, y a la izquierda tenemos la opción de volver al menú principal del administrador.

- **Botones centrales:** en los que se encuentran las distintas opciones que se ofrecen al administrador de la aplicación para gestionar el material.

4.- Implementación

En este apartado vamos a mostrar la arquitectura de la aplicación, entrando en detalle en los aspectos más importantes y explicando su funcionamiento.

4.1.- Arquitectura

A continuación vamos a mostrar los dos tipos de arquitecturas que aplicamos a nuestro sistema.

- **Arquitectura cliente-servidor:** desde el punto de vista funcional, se puede definir la computación Cliente/Servidor como una arquitectura distribuida que permite a los usuarios obtener acceso a la información de forma transparente, aun en entornos multiplataforma. Es la arquitectura tradicional que se aplica a las aplicaciones web [19].

  Dicha arquitectura la forman el cliente y el servidor:

  - **Cliente:** es el proceso que permite al usuario formular los requisitos y pasarlos al servidor, se le conoce con el término *front-end.*
    Normalmente maneja todas las funciones relacionadas con la manipulación y despliegue de datos, por lo que están desarrollados sobre plataformas que permiten construir interfaces gráficas de usuario (GUI).

  - **Servidor:** es el proceso encargado de atender a múltiples clientes que hacen peticiones de algún recurso administrado por él. Al proceso del servidor se le conoce con el término *back-end*.

El servidor maneja normalmente las funciones relacionadas con las reglas de negocio y los recursos de datos.

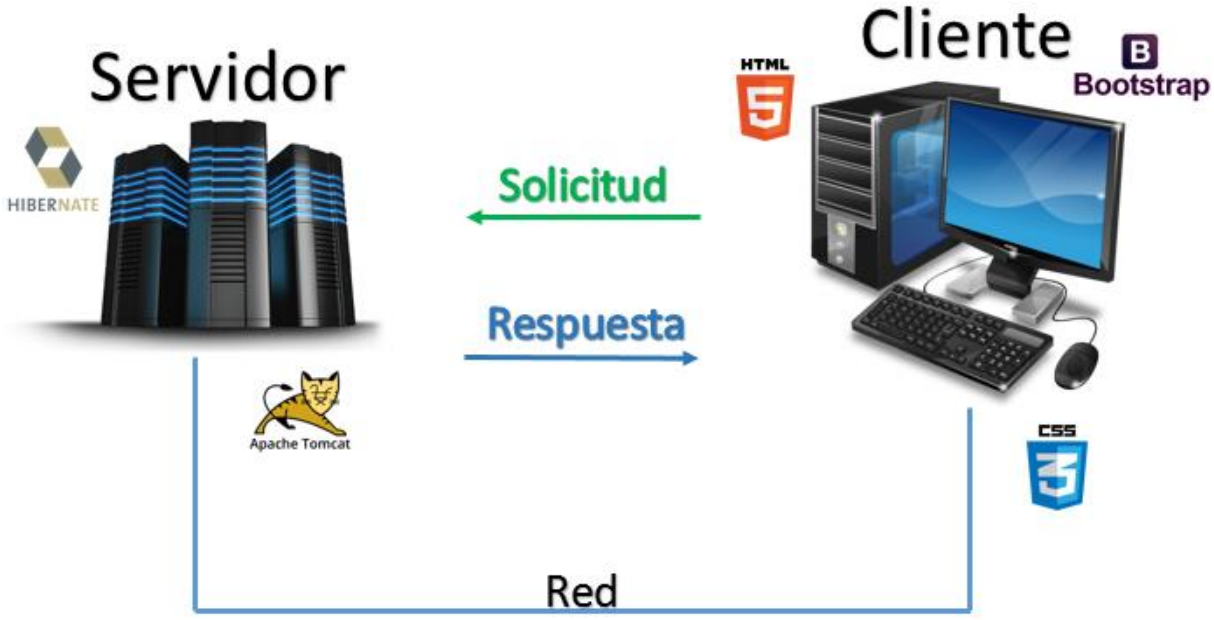

Ilustración 36- Arquitectura cliente-servidor

Como podemos ver en la imagen 36, en el modelo cliente-servidor, el cliente envía un mensaje solicitando un determinado servicio a un servidor (hace una petición), y este envía uno o varios mensajes con la respuesta (provee el servicio).

- **Modelo-Vista-Controlador:** es un patrón que define la organización independiente del **Modelo** (Objetos de Negocio), la **Vista** (interfaz de usuario) y el **Controlador** (controlador del *workflow* de la aplicación) [20].

  De esta forma, dividimos el sistema en tres capas, en el que tenemos la encapsulación de los datos, la interfaz o vista por otro y por último la lógica interna o controlador.

  - **Modelo:** es la capa donde se trabaja con los datos, por tanto contendrá mecanismos para acceder a la información y también para actualizar su estado. Los datos estarán habitualmente en una base de datos, por lo que en el modelo tendremos todas las funciones que accederán a las tablas y harán los correspondientes *selects, updates, inserts*, etc.

  - **Vista:** contiene el código de nuestra aplicación, que va a producir la visualización de las interfaces de usuario.

- **Controlador:** contiene el código necesario para responder a las acciones que se solicitan en la aplicación, como visualizar un elemento, realizar una reserva, una búsqueda de información, etc. Es una capa que sirve de enlace entre la vista y el modelo, respondiendo a los mecanismos que puedan requerirse en nuestra aplicación.

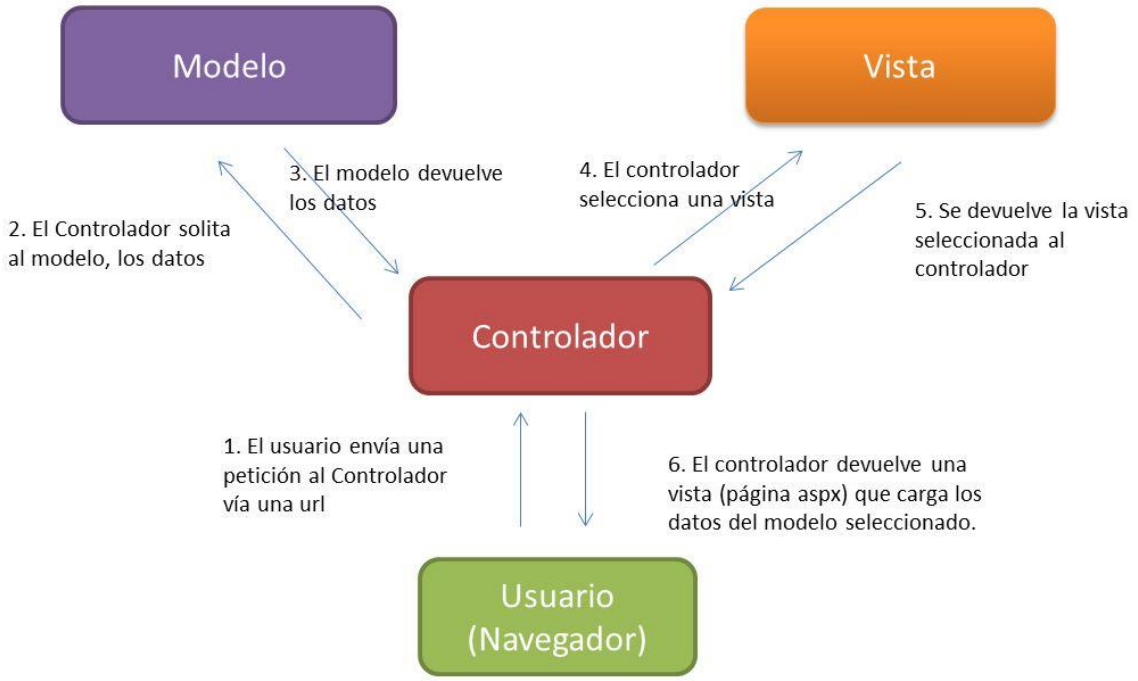

Ilustración 37- Modelo-Vista-Controlador (MVC)

En la imagen 37 podemos ver de forma clara la forma en que trabaja este patrón de arquitectura software.

En nuestro caso, para desarrollar la aplicación vamos a utilizar el *framework* Spring MVC, que como su propio nombre indica, utiliza este patrón.

El modelo de nuestra aplicación son clases Java que representan la lógica de negocio. Además, le vamos a aplicar el patrón DAO, que nos proporciona una interfaz entre el modelo y la base de datos, es decir, encapsula la forma de acceder a la fuente de datos.

Las vistas de nuestra aplicación están basadas en la tecnología JSP [21], implementando también JSTL [22] para realizar el procesamiento de los datos.

Los controladores de nuestra aplicación son clases Java que implementan los métodos para realizar las distintas acciones de la aplicación.

4.2.- Detalles sobre implementación

A continuación vamos a mostrar algunos detalles de implementación interesantes para explicar el desarrollo de nuestro proyecto.

SpringMVC

Spring MVC [23] es un framework de aplicaciones Java/J2EE desarrollado usando licencia OpenSource. Como su propio nombre indica, aplica el patrón MVC. Implementa el principio de programación *inversion of control* (IoC), su característica principal es la inyección de dependencias, que tiene como objetivo lograr un bajo acoplamiento entre los objetos de nuestra aplicación. Además nos ofrece una fácil configuración mediante ficheros .xml haciendo que nuestro código sea más limpio.

El funcionamiento de Spring es el siguiente:

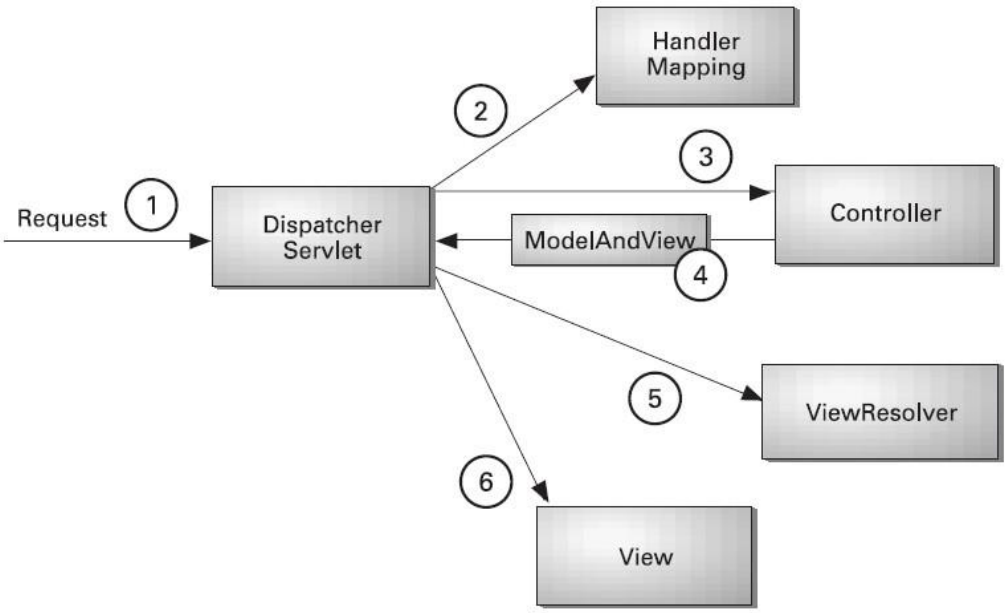

Ilustración 38- Funcionamiento de Spring (Fuente: https://hop2croft.wordpress.com/2011/09/10/ejemplo-basico-de-spring-mvc-con-maven/)

1. La petición llega al *DispatcherServlet* (1).

2. El *DispatcherServlet* tiene que encontrar el controlador que vaya a tratar la petición. Para ello, busca el manejador asociado a la url de la petición. Fase de *HandlerMapping* (2).

3. Cuando se encuentra el controlador (*Controller*), el *DispatcherServlet* deja gestionar la petición al controlador indicado (3). En el controlador se llama a la capa de servicios. El controlador devuelve al *DispatcherServlet* un objeto de tipo *ModelAndView* (4), dónde *Model* son los valores que obtenemos en la capa de servicio y *View* el nombre de la vista dónde mostramos la información.

4. Una vez que el *DispatcherServlet* tiene el objeto *ModelAndView,* asocia el nombre de la vista a una vista concreta (página jsp, jsf,…). Proceso *ViewResolver* (5).

5. Una vez resuelta la vista, el *DispatcherServlet* pasa el *Model* (valores obtenidos por el controlador a través de la capa de servicios) a la vista concreta *View* (6).

Una vez visto cómo funciona SpringMVC, vamos a mostrar las distintas configuraciones que hemos realizado para usar correctamente el *framework*.

En el fichero /WEB-INF/web.xml declaramos que nuestro *DispatcherServlet* es *SpringDispatcher* y a continuación indicamos la ruta donde tiene que buscar los controladores que formarán nuestra aplicación, que en este caso será la ruta /main/*.

```
<servlet>
    <servlet-name>SpringDispatcher</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
</servlet>

<servlet-mapping>
    <servlet-name>SpringDispatcher</servlet-name>
    <url-pattern>/main/*</url-pattern>
</servlet-mapping>
```

Ilustración 39- Fichero web.xml

Ahora vamos a mostrar la configuración de nuestro *SpringDispatcher*, que se encuentra en la ruta /WEB-INF/SpringDispatcher-servlet.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:mvc="http://www.springframework.org/schema/mvc"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:beans="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:p="http://www.springframework.org/schema/p"
       xsi:schemaLocation="
        http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans.xsd
        http://www.springframework.org/schema/mvc
        http://www.springframework.org/schema/mvc/spring-mvc.xsd
        http://www.springframework.org/schema/context
        http://www.springframework.org/schema/context/spring-context.xsd">

    <mvc:annotation-driven />
    <context:component-scan base-package="proyectoTFG.*" />

    <bean id="viewResolver"
          class="org.springframework.web.servlet.view.InternalResourceViewResolver"
          p:prefix="/WEB-INF/jsp/"
          p:suffix=".jsp" />
</beans>
```

Ilustración 40- Fichero SpringDispatcher-servlet.xml

- Mediante la anotación <mvc:annotation-driven/> indicamos que vamos a utilizar anotaciones mvc.

- Con la anotación <context:component-scan base-package="proyectoTFG.*"/> indicamos la jerarquía de paquetes donde Spring encontrará las clases de nuestra aplicación.

- Por último declaramos nuestro bean *ViewResolver* indicando la ruta en la que puede encontrar las vistas que forman nuestra aplicación y que serán .jsp.

Aunque anteriormente en la ilustración 39 hemos indicado la ruta donde nuestro *SpringDispatcher* buscará los controladores, tenemos que anotar las clases que harán esta función. Primero creamos el paquete *proyectoTFG.controladores* y en él añadiremos las clases.

```
@Controller
@RequestMapping("/usuarios")
public class ControladorUsuario {
```

Ilustración 41- Controlador usuario

Este es el caso del "ControladorUsuario", las anotaciones para que las clases se reconozcan como tales son *@Controller* y *@RequestMapping*.

Para terminar de explicar cómo funciona un controlador, vamos a ver dos métodos, que en este caso se encargan de mostrar un formulario para registrar un usuario; y de recibir los datos correspondientes al usuario, creándo el usuario si los datos introducidos son correctos.

```
@RequestMapping(value = "/registro", method = RequestMethod.GET)
public String creaUsuario(ModelMap model) {
    model.addAttribute("usuario", new Usuario());
    return "usuarios/registro";
}

@RequestMapping(value = "/registro", method = RequestMethod.POST)
public String creaUsuario(@ModelAttribute("usuario") @Valid Usuario usuario, BindingResult result, ModelMap model) {
    String view = "redirect:login";
    if (!result.hasErrors()) {
        if (usuarioDao.existeUsuario(usuario.getNombreUsuario())) {
            view = "redirect:registro";
        } else {
            usuarioDao.creaUsuario(usuario);
            model.clear();
        }
    } else {
        view = "usuarios/registro";
    }
    return view;
}
```

Ilustración 42- Métodos del controlador usuario

Los métodos se anotan con *@RequestMapping,* indicándoles la ruta de la función y el tipo de petición HTTP que tiene que responder.

La primera función realiza un petición de tipo GET, la cual prepara el modelo y nos redirige a la vista correspondiente.

La segunda función realiza una petición de tipo POST, la cual recoge los datos introducidos en el formulario por parte del usuario. Mediante la anotación *@Valid* indicamos que los datos deben ser comprobados, si son correctos, "usuarioDao" se

encarga de almacenar los datos en la BBDD y por último se redirige a la página de *login*, en caso contrario se redirige al formulario de registro y se indica el error.

Conexión y gestión de la base de datos

El gestor de bases de datos que vamos a utilizar será Apache Derby, que puede ser empotrado en aplicaciones Java y utilizado para procesos de transacciones online.

Vamos a utilizar Mapeo objeto-relacional [24] para la base de datos, que nos hace simplificar las operaciones CRUD habituales sobre una base de datos relacional para almacenar y recuperar los objetos de negocio, esto lo haremos con la herramienta *Hibernate* [25] apoyándonos en la especificación de JPA (*Java Persistence API*), con cuyas especificaciones y reglas vamos a construir la capa de persistencia de nuestra aplicación.

El mapeo objeto-relacional no es 100% automático, JPA requiere una mínima información para generar el mapeado y nos ofrece anotaciones en el código Java para indicar la información de mapeado. Una vez hechas, nuestro esquema de BBDD se genera de forma automática.

A continuación vamos a mostrar el mapeo de una clase de nuestro sistema.

```
@Entity
@Table(name = "Pista")
public class Pista implements Serializable{
    @Id
    @GeneratedValue(strategy=GenerationType.AUTO)
    private int id;
    private int identificador;
    private String descripcion;
    @OneToOne(fetch = FetchType.EAGER)
    private TipoPista tipoPista;
```

Ilustración 43- Entidad pista

Como podemos ver en la imagen 43, se trata de nuestra entidad “Pista”. Para marcar la entidad como persistente, lo hacemos mediante la anotación *@Entity,*

además, tenemos la opción de poner un nombre a la tabla con la anotación *@Table*, en otro caso el nombre de la tabla generada será el nombre que tenga la entidad.

Con la anotación *@Id* marcamos el atributo que hace de clave primaria de la tabla, en este caso, será autogenerada con la anotación *@GeneratedValue*.

Nuestra entidad tiene una relación de "uno a uno" con otra entidad "TipoPista", por lo que marcaremos esa relación con la anotación *@OneToOne*.

Una vez anotada la información mínima necesaria en cada entidad persistente, necesitamos iniciar JPA, para ello en una aplicación Java SE la forma habitual de realizarlo es creando un *EntityManagerFactory,* posteriormente el *EntityManagerFactory* nos proporcionará un *EntityManager* con el que interactuar con JPA. En nuestro caso, Spring será el encargado configurar, crear y gestionar la EMF (*EntityManagerFactory*).

Para realizar toda la configuración, tenemos que declarar una serie de beans en el fichero de configuración /WEB-INF/applicationContext.xml.

```
<bean id="dataSource" class="org.springframework.jdbc.datasource.DriverManagerDataSource">
    <property name="driverClassName" value="org.apache.derby.jdbc.ClientDriver" />
    <property name="url" value="jdbc:derby://localhost:1527/Red_Social_BBDD" />
    <property name="username" value="root" />
    <property name="password" value="root" />
</bean>

<bean id="jpaVendorAdapter" class="org.springframework.orm.jpa.vendor.HibernateJpaVendorAdapter">
    <property name="database" value="DERBY" />
    <property name="databasePlatform" value="org.hibernate.dialect.DerbyTenSevenDialect" />
    <property name="showSql" value="false" />
    <property name="generateDdl" value="true" />
</bean>

<bean id="emf" class="org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean">
    <property name="packagesToScan" value="red_social.beans" />
    <property name="dataSource" ref="dataSource" />
    <property name="jpaVendorAdapter" ref="jpaVendorAdapter" />
</bean>

<bean id="transactionManager" class="org.springframework.orm.jpa.JpaTransactionManager">
    <property name="entityManagerFactory" ref="emf" />
</bean>

<tx:annotation-driven transaction-manager="transactionManager" />
```

Ilustración 44- Fichero applicationContext.xml

Como podemos ver en la ilustración 44, la configuración es la siguiente:

- Bean “dataSource”: en el que definimos el *driver* de conexión, la url, nombre y usuario de la base de datos.
- Bean “jpaVendorAdapter”: en el que definimos la configuración de JPA.
- Bean “emf”: en el que definimos la configuración del *EntityManagerFactory.*
- Bean “transactionManager”: en el que definimos las transacciones.
- Anotación <tx:annotation-driven transaction-manager="transactionManager"/> para definir que vamos a utilizar anotaciones de transacciones.

A continuación vamos a crear el fichero *persistence.xml* situado en la ruta /META-INF/persistence.xml. Dicho fichero nos lo genera automáticamente nuestra herramienta software de desarrollo *NetBeans*.

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.1" xmlns="http://xmlns.jcp.org/xml/ns/persistence"
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
                http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd">
  <persistence-unit name="PersistenceUnitTFG" transaction-type="RESOURCE_LOCAL">
    <provider>org.hibernate.ejb.HibernatePersistence</provider>
    <exclude-unlisted-classes>false</exclude-unlisted-classes>
    <properties>
      <property name="javax.persistence.jdbc.url" value="jdbc:derby://localhost:1527/BDTFG"/>
      <property name="javax.persistence.jdbc.user" value="root"/>
      <property name="javax.persistence.jdbc.driver" value="org.apache.derby.jdbc.ClientDriver"/>
      <property name="javax.persistence.jdbc.password" value="root"/>
      <property name="hibernate.cache.provider_class" value="org.hibernate.cache.NoCacheProvider"/>
    </properties>
  </persistence-unit>
</persistence>
```

Ilustración 45- Fichero persistence.xml

Realizadas todas las configuraciones, vamos a mostrar cómo sería la gestión de nuestra base de datos desde nuestros DAOs.

```java
@Repository(value = "pistadaojpa")
public class PistaDaoJpa implements PistaDao{

    EntityManagerFactory emf = null;
    EntityManager em = null;

    public PistaDaoJpa() {
        emf = Persistence.createEntityManagerFactory("PersistenceUnitTFG");
        em = emf.createEntityManager();
    }
```

Ilustración 46- Clase PistaDaoJpa

Siguiendo con el ejemplo anterior de nuestra entidad pista, la imagen 46 nos muestra el DAO correspondiente a dicha entidad. Marcamos la clase con *@Repository* y añadimos un identificador, ya que utilizaremos varios DAOs en nuestra aplicación y así no habrá ambigüedad del DAO a inyectar. Iniciamos la EMF correspondiente, indicando el nombre de la unidad de persistencia creada en el apartado anterior. A continuación definimos el DAO en nuestro controlador de la siguiente forma:

```java
@Autowired
@Qualifier("pistadaojpa")
private PistaDao pistaDao;
```

Ilustración 47- Definición del DAO en el controlador

Ahora vamos a mostrar un ejemplo del método para crear, en este caso, nuestro objeto “Pista” en la base de datos.

```java
@Override
@Transactional(propagation = Propagation.REQUIRED, readOnly = false, rollbackFor = Throwable.class)
public void creaPista(String id, String desc, String tipoPista) {
    try {
        TipoPista tp = em.createQuery("select t from TipoPista t where t.nombre = ?1", TipoPista.class)
                .setParameter(1, tipoPista)
                .getSingleResult();
        Pista p = new Pista(id, desc, tp);
        em.getTransaction().begin();
        em.persist(p);
        em.getTransaction().commit();
    } catch (Exception e) {
        throw new ExcepcionBBDD();
    }
}
```

Ilustración 48- Método creaPista

Marcamos nuestro método con la anotación *@Transactional* para indicar que se trata de una transacción. Tras crear el objeto, abrimos la transacción con “em.getTransaction().begin()”, creamos nuestro objeto con la operación *persist()* y cerramos la transacción con “em.getTransaction().commit()”.

Spring Security

Spring Security [26] es un *framework* que permitirá gestionar todo lo relativo a la seguridad de nuestra aplicación web, desde el protocolo de seguridad, hasta los roles que necesitan los usuarios para acceder a los diferentes recursos de la aplicación. Se podría decir que *Spring Security* es declarativo y apenas hace falta programar nada y todo se configura mediante un fichero de configuración.

Para poder usar dicho *framework*, primero vamos a añadir las dependencias necesarias, que en este caso son las siguientes:

```
<dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-config</artifactId>
    <version>3.1.3.RELEASE</version>
</dependency>
<dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-core</artifactId>
    <version>3.1.3.RELEASE</version>
</dependency>
<dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-taglibs</artifactId>
    <version>3.1.3.RELEASE</version>
</dependency>
<dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-web</artifactId>
    <version>3.1.3.RELEASE</version>
</dependency>
```

Ilustración 49- Dependencias SpringFramework

El siguiente paso es habilitar Spring Security añadiendo un filtro en el web.xml:

```
<filter>
    <filter-name>springSecurityFilterChain</filter-name>
    <filter-class>org.springframework.web.filter.DelegatingFilterProxy</filter-class>
    <init-param>
        <param-name>encoding</param-name>
        <param-value>UTF-8</param-value>
    </init-param>
    <init-param>
        <param-name>forceEncoding</param-name>
        <param-value>true</param-value>
    </init-param>
</filter>

<filter-mapping>
    <filter-name>springSecurityFilterChain</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
```

Ilustración 50- Filtro SpringSecurity

A continuación es necesario crear el archivo de configuración de *Spring Security*, que llamaremos spring-security.xml, situado en la ruta /WEB-INF/Spring-security.xml y que contiene lo siguiente:

```
<http use-expressions="true">
    <intercept-url pattern="/main/admin/**" access="hasRole('ROLE_ADMIN')"/>
    <intercept-url pattern="/main/usuarios/**" access="hasAnyRole('ROLE_USER','ROLE_ADMIN')"/>
    <form-login login-page="/main/usuarios/login" default-target-url="/main/usuarios/perfilUsuario" />
    <logout logout-url="/main/usuarios/logout" delete-cookies="JSESSIONID"/>
    <session-management invalid-session-url="/main/usuarios/login" ></session-management>
</http>
```

Ilustración 51- Fichero de configuración Spring-security.xml

Como se observa en la imagen 51, declaramos las *urls* que *Spring Security* tiene que interceptar para los distintos roles que manejaremos en el sistema. Indicamos la página de *login*, la página a la que se accederá tras hacer *login* y la página de *logout*.

```
<http pattern="/main/usuarios/login" security="none"/>
<http pattern="/main/usuarios/registro" security="none"/>
<http pattern="/main/usuarios/consultaPista" security="none"/>
<http pattern="/main/usuarios/disponibilidadPista" security="none"/>
```

Ilustración 52- Urls sin filtro de seguridad de SpringSecurity

Como observamos en la ilustración 52, indicamos a qué urls se podrá acceder sin el filtro de seguridad.

```
<beans:bean id="miUserServiceJDBC"
            class="org.springframework.security.core.userdetails.jdbc.JdbcDaoImpl">
    <beans:property name="dataSource" ref="dataSource"/>
    <beans:property name="usersByUsernameQuery"
                    value="SELECT nombreUsuario as username, password, true as enabled
                   FROM usuario WHERE nombreUsuario=?"/>
    <beans:property name="authoritiesByUsernameQuery"
                    value="SELECT nombreUsuario as username, rol as authority
                   FROM roles WHERE nombreUsuario=?"/>
</beans:bean>
```

Ilustración 53- Bean de configuración y consulta de la BBDD para SpringSecurity

Como vemos en la ilustración 53, definimos un *Bean* en el que indicamos la configuración y las consultas a la base de datos que debe realizar para autenticar y autorizar a un usuario en el sistema [27].

```
<authentication-manager alias="authenticationManager">
    <authentication-provider user-service-ref="miUserServiceJDBC" />
</authentication-manager>
```

Ilustración 54- Gestor de autenticación

Como podemos ver en la imagen 54, definimos un *authentication manager* que se encarga de gestionar la autentificación, que depende de un *authentication provider* que son los que de manera efectiva obtienen las credenciales de usuario, en este caso de nuestra base de datos que ha sido definida anteriormente.

```
<jee:jndi-lookup id="dataSource" jndi-name="jdbc/BDTFG" resource-ref="true"/>
```

Ilustración 55- Conexión a la BBDD mediante JNDI

Como podemos observar en la imagen 55, definimos la conexión a la base de datos mediante JNDI.

En el fichero /META-INF/context.xml debemos definir el siguiente *resource*:

```
<?xml version="1.0" encoding="UTF-8"?>
<Context path="/ProyectoTFG">

<Resource name="jdbc/BDTFG"
          auth="Container"
          type="javax.sql.DataSource"
          driverClassName="org.apache.derby.jdbc.ClientDriver"
          url="jdbc:derby://localhost:1527/BDTFG"
          username="root"
          password="root"
          maxActive="100"
          maxIdle="30"
          maxWait="10000"/>

</Context>
```

Ilustración 56- Fichero context.xml

De esta forma tenemos el *framework Spring Security* listo para usarlo.

5.- Conclusiones

Tras realizar el desarrollo del proyecto, podemos decir que el resultado ha sido satisfactorio. El avance de la tecnología dentro de la informática en los últimos años, nos ha ayudado a esto. La aparición de nuevas metodologías, tecnologías, nuevos *frameworks…* hacen que el desarrollo de software sea más liviano para el desarrollador. La motivación que me ha llevado a realizar el proyecto ha sido poder acercar a los centros deportivos este tipo de aplicaciones, para realizar la gestión de forma más sencilla y con la consecuente ventaja hacia el usuario, quien desde cualquier lugar puede acceder a la aplicación para realizar las operaciones oportunas.

Durante el desarrollo del proyecto se ha realizado un estudio de aplicaciones de reserva de pistas en centros deportivos existentes en la actualidad, se ha elaborado una planificación temporal del proyecto y se han calculado sus costes, se ha diseñado una aplicación para realizar consultas y reservas de instalaciones y material en centros deportivos y por último se ha implementado un prototipo de aplicación web para el alquiler de equipamiento e instalaciones en un centro deportivo. Por lo tanto, podemos concluir que se han alcanzado los objetivos inicialmente propuestos con un nivel satisfactorio.

Desarrollar el proyecto ha sido un gran desafío, ya que es la primera vez que desarrollo software de gran envergadura. Por supuesto, al realizar el desarrollo de forma autónoma y sin un equipo de trabajo ha supuesto más inconvenientes que ventajas, ya que como se suele decir “cuatro ojos ven más que dos”. Aunque gracias a lo aprendido en las asignaturas, al tutor del proyecto y compañeros de clase, las dificultades que han surgido a lo largo del desarrollo, se han resuelto sin grandes problemas. Además, la bibliografía aportada durante el grado, me ha hecho afianzar conocimientos que me han permitido mejorar aspectos de la programación, además de aprender cosas nuevas.

Las metodologías y tecnologías que he utilizado habían sido estudiadas en diferentes asignaturas del grado, aunque siempre aparecen los típicos “problemillas”. Esto no ha supuesto más que una motivación, profundizando y afianzando

conocimientos en dichos aspectos. Por lo que en general, las tecnologías y metodologías elegidas para el desarrollo han sido muy satisfactorias.

Para desarrollar el proyecto se ha utilizado el lenguaje de programación Java, usando la herramienta de desarrollo software NetBeans 8.1, dónde la experiencia es bastante amplia. Como *framework* de arquitectura se ha utilizado SpringMVC, dónde la experiencia no era tan amplia, pero he podido profundizar mucho en su utilización. Para la interfaz hemos usado una plantilla del *framework* Bootstrap, que nos proporciona una enorme facilidad a la hora de desarrollar la interfaz de nuestra aplicación.

Durante el desarrollo del proyecto he conseguido aplicar los conocimientos obtenidos en las distintas asignaturas del grado y profundizar en nuevos conocimientos en el área de las aplicaciones web y el *framework* SpringMVC.

Utilizar la metodología ágil Scrum ha sido beneficioso, ya que al ser la primera vez que desarrollo un proyecto de gran envergadura, he podido poner en práctica los conocimientos teóricos sobre esta metodología y ver sus ventajas e inconvenientes.

Finalmente tras hacer un breve resumen de todos los puntos que han intervenido en el proyecto, puedo decir que el resultado ha sido muy satisfactorio.

6.- Bibliografía

[1] P. González, Apuntes de la asignatura: Desarrollo Ágil, Jaén: Universidad de Jaén, 2014.

[2] "Qué es Scrum", proyectos ágiles [Online]. Disponible en: https://proyectosagiles.org/que-es-scrum/. Acceso: Recuperado en Abril. 16, 2016.

[3] "21. Web MVC framework", Docs.spring.io. [Online]. Disponible en: http://docs.spring.io/spring/docs/current/spring-framework-reference/html/mvc.html. Acceso: Recuperado en Mayo. 2, 2016.

[4] A. Mark Otto, "Bootstrap · The world's most popular mobile-first and responsive front-end framework.", *Getbootstrap.com*, 2016. [Online]. Disponible en: http://getbootstrap.com/. Acceso: Recuperado en Mayo. 17, 2016.

[5] M. García, Apuntes de la asignatura: Ingeniería de requisitos, Jaén: Universidad de Jaén, 2014.

[6] M. Cohn and K. Beck, *User stories applied: For agile software development*, 12th ed. Boston: Addison-Wesley Educational Publishers, 2004

[7] P. González, Apuntes de la asignatura: Desarrollo Ágil, Jaén: Universidad de Jaén, 2014.

[8] M. Cohn, "User Stories and User Story Examples by Mike Cohn", *Mountain Goat Software*, 2016. [Online]. Disponible en: http://www.mountaingoatsoftware.com/agile/user-stories. Acceso: Recuperado en Mayo. 23, 2016.

[9] D. Álvarez. Noviembre, 2013. *Be my Scrum master.* [Online]. Disponible en: http://bemyscrummaster.blogspot.com.es/2013/11/estimar-el-tamano-de-las-historias-con.html. Acceso: Recuperado en Mayo. 30, 2016.

[10] M. Cohn, *Agile estimating and planning.* Prentice-hall international edition, 2005

[11] A. Romeu, Dic, 2013*. ¿Qué es y cómo hacer un diagrama de Burndown?.* [Online]. Disponible en: http://albertoromeu.com/como-hacer-un-diagrama-de-burndown/. Acceso: Recuperado en Mayo. 17, 2016.

[12] I. Sommerville, *Software engineering (9th edition)*, 9th ed. Boston: Addison-Wesley Educational Publishers, 2010.

[13] J. García*. El patrón MVC en el desarrollo de aplicaciones web* [Online]. Disponible en: http://juandarodriguez.es/cursosf14/unidad2.html. Acceso: Recuperado en Junio. 12, 2016.

[14] S. Bennett, S. McRobb, and R. Farmer, *Object-oriented systems analysis and design using UML*, 3rd ed. London: McGraw Hill Higher Education, 2007.

[15] A. Ruiz, Apuntes de la asignatura: Desarrollo de aplicaciones empresariales, Jaén: Universidad de Jaén, 2015.

[16] A. Aguilar, Apuntes de la asignatura: Fundamentos de bases de datos, Jaén: Universidad de Jaén, 2013.

[17] Korth, H.F., Silberschatz, A. *Fundamentos de Bases de Datos*. McGraw-Hill, 1993.

[18] A. S. Solutions, "Prototypes, specifications, and diagrams in One tool," 2002. [Online]. Disponible en: http://www.axure.com/. Acceso: Recuperado en: Jun. 13, 2016.

[19] C. García*. Historia de la web, arquitectura cliente-servidor* [Online]. Disponible en: http://charliedaw2236.blogspot.com.es/p/arquitectura-cliente-servidor.html. Acceso: Recuperado en Junio. 22, 2016.

[20] M. Álvarez*. Qué es MVC* [Online]. Disponible en: http://www.desarrolloweb.com/articulos/que-es-mvc.html. Acceso: Recuperado en Junio. 23, 2016.

[21] *JavaServes Pages Technology.* Oracle [Online]. Disponible en: http://www.oracle.com/technetwork/java/javaee/jsp/index.html. Acceso: Recuperado en Junio. 28, 2016.

[22] *JavaServer Pages Estándar Tag Library*. Oracle [Online]. Disponible en: http://www.oracle.com/technetwork/java/jstl-137486.html. Acceso: Recuperado en Junio. 28, 2016.

[23] "21. Web MVC framework", *Docs.spring.io*, 2016. [Online]. Disponible en: http://docs.spring.io/spring/docs/current/spring-framework-reference/html/mvc.html. Acceso: Recuperado en Junio. 28, 2016.

[24] *¿Qué es un ORM?*. Entity Framework [Online]. Disponible en: http://www.tuprogramacion.com/glosario/que-es-un-orm/. Acceso: Recuperado en Junio. 28, 2016.

[25] *Hibernate. Everything data*. Hibernate [Online]. Disponible en: http://hibernate.org/. Acceso: Recuperado en Julio. 12, 2016.

[26] *Spring Security*. Spring [Online]. Disponible en: http://projects.spring.io/spring-security/. Acceso: Recuperado en Julio. 15, 2016.

[27] *Seguridad*. Universidad de alicante, Dpto. de Ciencia de la computación e inteligencia artificial. [Online]. Disponible en: http://www.jtech.ua.es/j2ee/publico/spring-2012-13/sesion07-apuntes.html. Acceso: Recuperado en Julio. 15, 2016.

Anexo I. Descripción de contenidos suministrados

El CD entregado contiene:

- **Memoria_ÁlvaroAglioSánchez**: memoria del TFG en formato PDF.
- **Código_ÁlvaroAglioSánchez**: código fuente del proyecto.
- **Diagramas_ÁlvaroAglioSánchez**: fichero de VisualParadigm con los diagramas descritos en la memoria.
- **Vídeo_ÁlvaroAglioSánchez**: Video de muestra del funcionamiento de la aplicación en formato mp4.

Anexo II. Manual de usuario

El siguiente manual de usuario explica de manera sencilla y visual las funcionalidades de la aplicación, para que cualquier usuario pueda hacer uso de ella. El manual está dividido en dos partes, una para el usuario de la aplicación y otra para el administrador de la aplicación.

- Usuario

1. Página principal de la aplicación

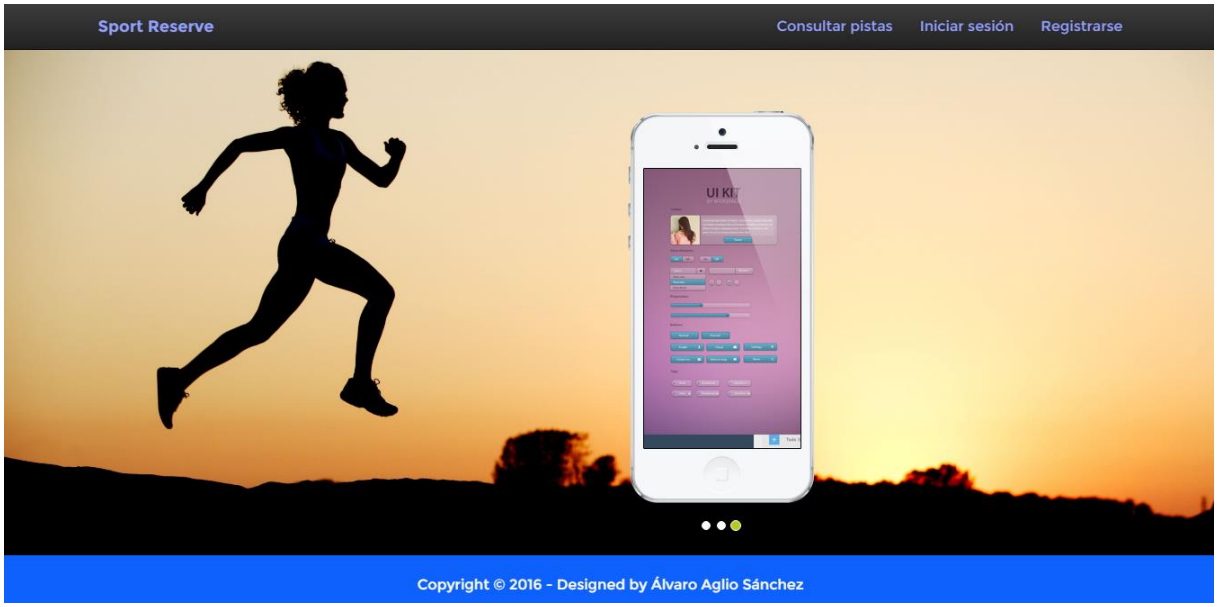

Ilustración 57- Página principal de la aplicación

En la ilustración 57 podemos ver la página principal de la aplicación. En ella se puede acceder a iniciar sesión, registrarse en la aplicación y consultar las pistas disponibles mediante la barra superior.

2. Registro en la aplicación

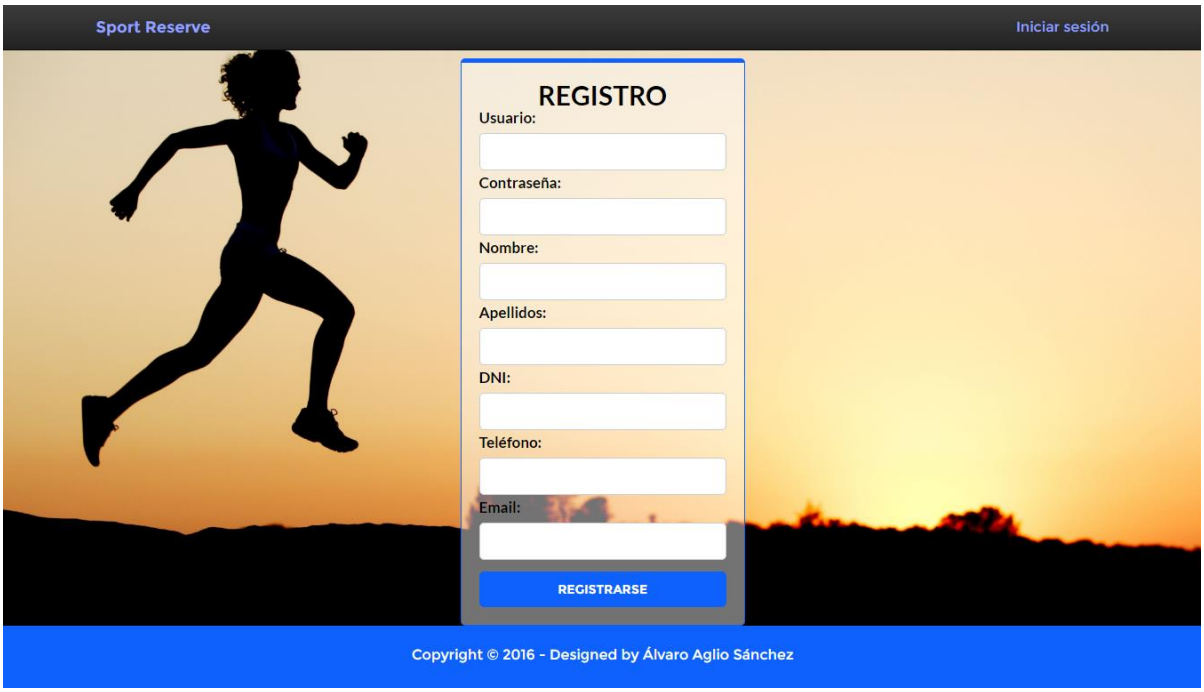

Ilustración 58- Registro en la aplicación

Para registrarse como usuario en la aplicación rellene el formulario y pulse el botón “Registrarse”. Si los datos son correctos se le redirigirá a la página para iniciar sesión, si no son correctos, se le indicará para corregirlos.

3. Iniciar sesión

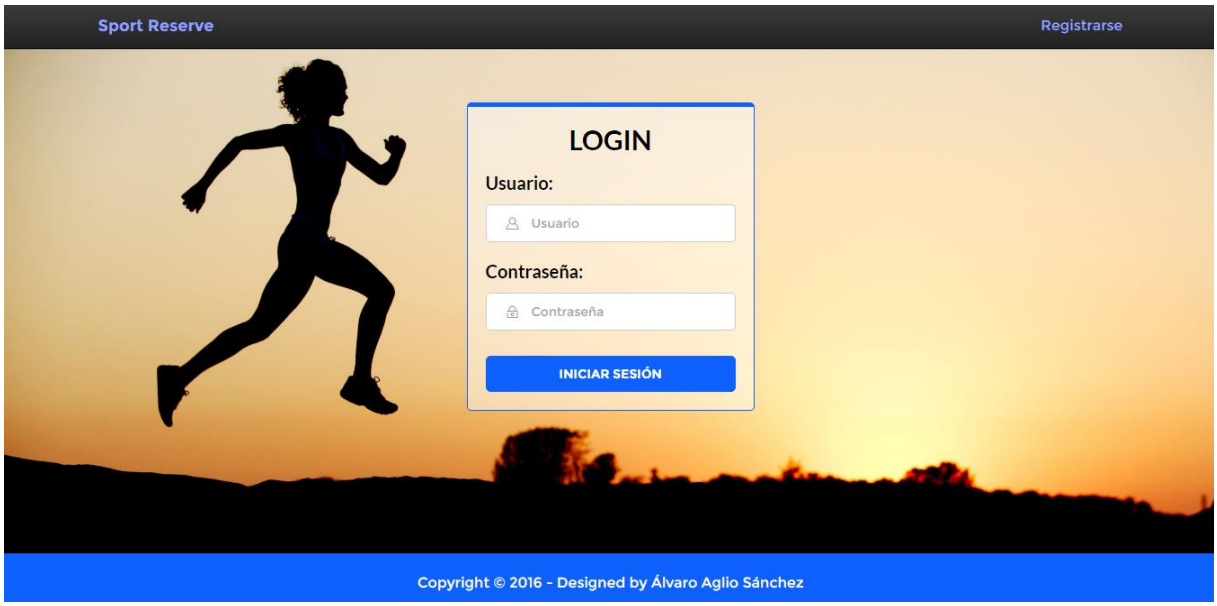

Ilustración 59- Iniciar sesión

Para iniciar sesión en la aplicación, rellene el nombre de usuario y la contraseña y pulse el botón “Iniciar sesión”, si los datos son correctos se le redirigirá al menú de usuario, si no son correctos, se le indicará para corregirlos.

4. Menú de usuario

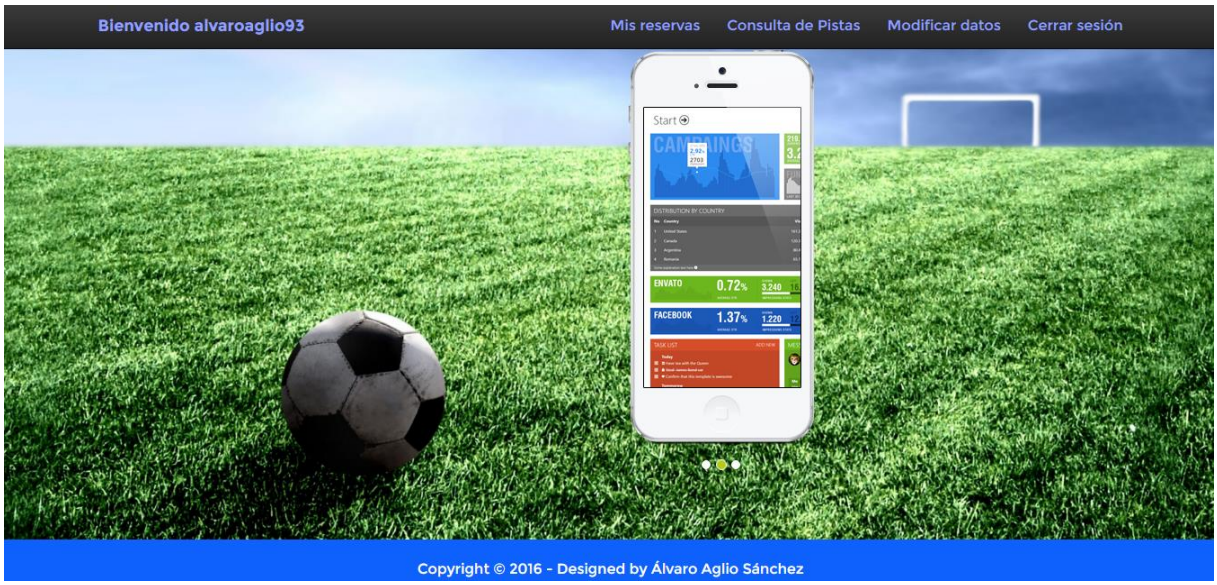

Ilustración 60- Menú de usuario

Una vez iniciada sesión, el menú del usuario permite, mediante la barra superior:

- Ver las reservas realizadas por el usuario
- Consultar la disponibilidad de las pistas
- Modificar los datos de usuario
- Cerrar sesión

Pulsando sobre los distintos botones accederemos a dichas acciones.

5. Modificar datos de usuario

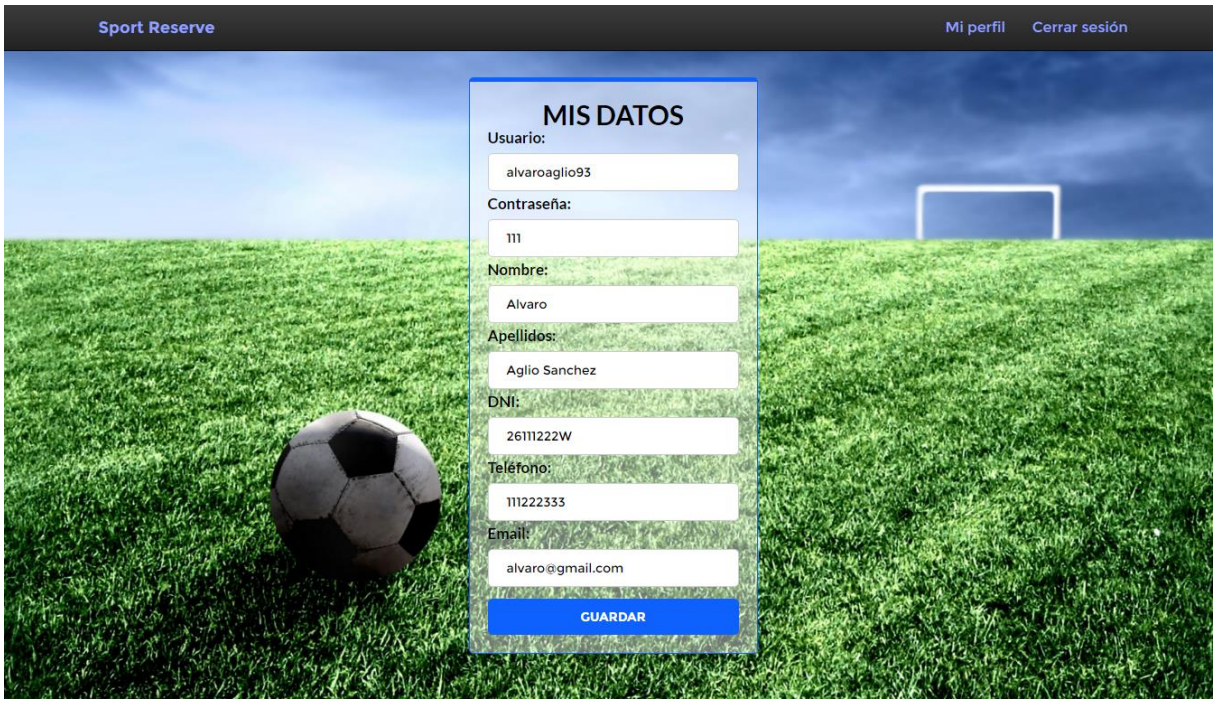

Ilustración 61- Modificar datos de usuario

Para modificar los datos de usuario, introduzca los cambios que se requieran en el formulario y pulse el botón “Guardar”, si los datos son correctos se actualizarán los datos de usuario, si no son correctos, se le indicará para corregirlos.

6. Visualizar reservas realizadas por el usuario

Ilustración 62- Visualizar reservas de usuario

Como podemos ver en la ilustración 62, se muestra la información de las reservas realizadas por el usuario, las cuales se pueden cancelar pulsando el botón "Cancelar reserva".

7. Consultar pistas disponibles

Ilustración 63- Consultar pistas disponibles

Para consultar las pistas disponibles seleccione el deporte requerido pulsando en el desplegable, a continuación introduzca la fecha pulsando sobre el botón del calendario y finalmente pulse el botón “Consultar”. A continuación se mostrará la ilustración 64.

8. Disponibilidad de las pistas

Ilustración 64- Información de la disponibilidad de las pistas

En la ilustración 64 podemos ver la tabla donde se muestra la información con las pistas disponibles. Para realizar una reserva, pulse sobre el botón “libre”. A continuación se mostrará la ilustración 65.

9. Realizar reserva

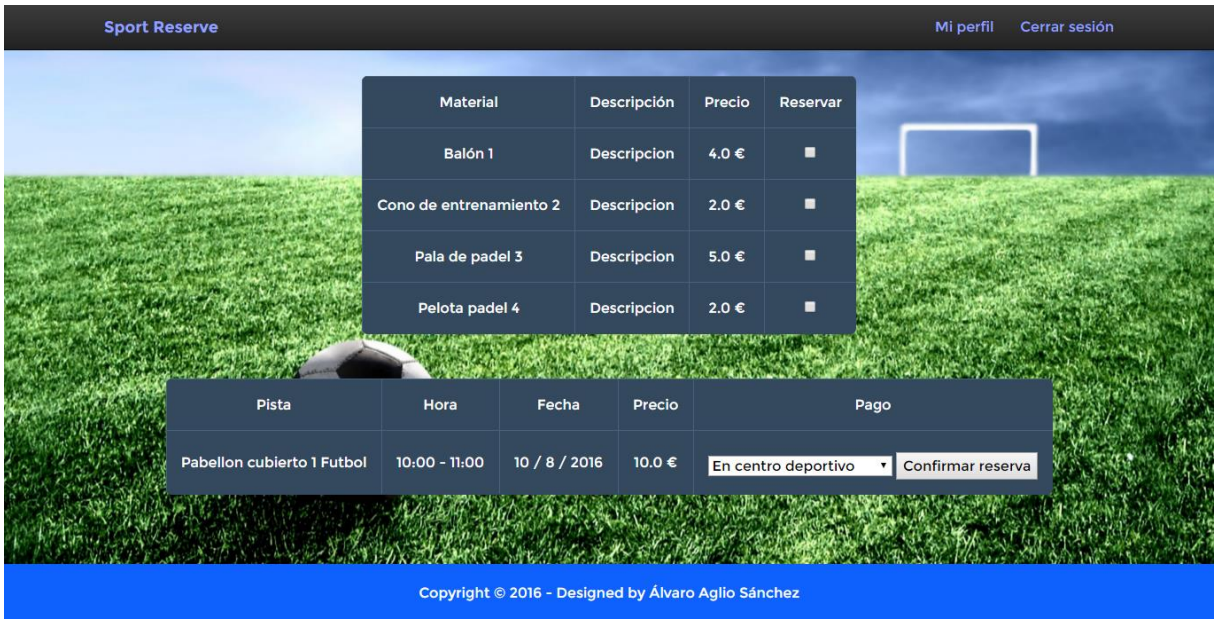

Ilustración 65- Realizar reserva

Tras pulsar la pista requerida, se muestra una tabla con materiales, los cuales podemos reservas pinchando sobre los *checkbox* de los mismos. En la tabla inferior se muestran los datos de la reserva. Para realizar la reserva, seleccione una forma de pago en el desplegable de la columna “Pago” y pulse el botón “Confirmar reserva”.

- Administrador

1. Menú administrador

Ilustración 66- Menú de administrador

Tras registrarnos como administrador, el menú de administrador permite:

- Gestionar pistas
- Gestionar usuarios
- Gestionar reservas
- Gestionar material
- Visualizar las reservas del día

Pulsando sobre los distintos botones accederemos a dichas acciones.

2. Gestión de pistas

Ilustración 67- Gestión de pistas

En el apartado gestión de pistas podemos realizar las siguientes opciones:

- Crear tipo de pista
- Editar tipo de pista
- Eliminar tipo de pista
- Crear pista
- Editar pista
- Eliminar pista

Pulsando sobre los distintos botones accederemos a dichas acciones. A continuación se explican cada una de ellas.

3. Crear tipo de pista

Ilustración 68- Crear tipo de pista

Para crear un tipo de pista rellene el formulario y pulse sobre el botón “Crear”, si los datos son correctos se le redirigirá al menú de gestión de pistas, si no son correctos, se le indicará para corregirlos.

Para crear una pista, la forma es análoga a la creación de un tipo de pista, pulse el botón “Crear pista” en el apartado gestión de pistas, rellene el formulario y pulse sobre el botón “Crear”, si los datos son correctos se le redirigirá al menú de gestión de pistas, si no son correctos, se le indicará para corregirlos.

4. Editar tipo de pista

Ilustración 69- Editar tipo de pista

Para editar un tipo de pista, pulse sobre el tipo de pista que desee editar, a continuación se le mostrará un formulario con los datos disponibles del tipo de pista seleccionado, introduzca los cambios que se requieran en el formulario y pulse el botón “Guardar”, si los datos son correctos se actualizarán los datos, si no son correctos, se le indicará para corregirlos.

Para editar una pista, la forma es análoga a la edición de un tipo de pista, pulse el botón “Editar pista” en el apartado gestión de pistas, pulse sobre la pista que desee editar, a continuación se le mostrará un formulario con los datos disponibles de la pista seleccionada, introduzca los cambios que se requieran en el formulario y pulse el botón “Guardar”, si los datos son correctos se actualizarán los datos, si no son correctos, se le indicará para corregirlos.

5. Borrar tipo de pista

Ilustración 70- Borrar tipo de pista

Para eliminar un tipo de pista, pulse sobre el tipo de pista que desee borrar, a continuación se eliminará de la base de datos.

Para eliminar una pista, la forma es análoga a la eliminación de un tipo de pista, pulse el botón “Eliminar pista” en el apartado gestión de pistas, a continuación pulse sobre la pista que desee borrar, a continuación se eliminará de la base de datos.

6. Gestión de usuarios

Ilustración 71- Gestión de usuarios

En el apartado gestión de usuarios se nos muestra una tabla con la información de los usuarios, la cual podemos modificar pulsando sobre el botón “Modificar”, a continuación se nos mostrará un formulario con la información del usuario. Introduzca los cambios que se requieran en el formulario y pulse el botón “Guardar”, si los datos son correctos se actualizarán los datos, si no son correctos, se le indicará para corregirlos.

Pulsando sobre el botón reservas, se nos muestran las reservas del usuario seleccionado, las cuales podemos eliminar pulsando sobre el botón “Cancelar reserva”.

Para eliminar el usuario, pulse sobre el botón “Eliminar” del usuario correspondiente y será eliminado de la base de datos.

Para crear un nuevo usuario, pulse sobre el botón “Crear usuario”, situado en la parte superior de la tabla, a continuación se le mostrará un formulario, rellene los datos y pulse el botón “Crear”, si los datos son correctos se creará el usuario, si no son correctos, se le indicará para corregirlos.

7. Gestión de reservas

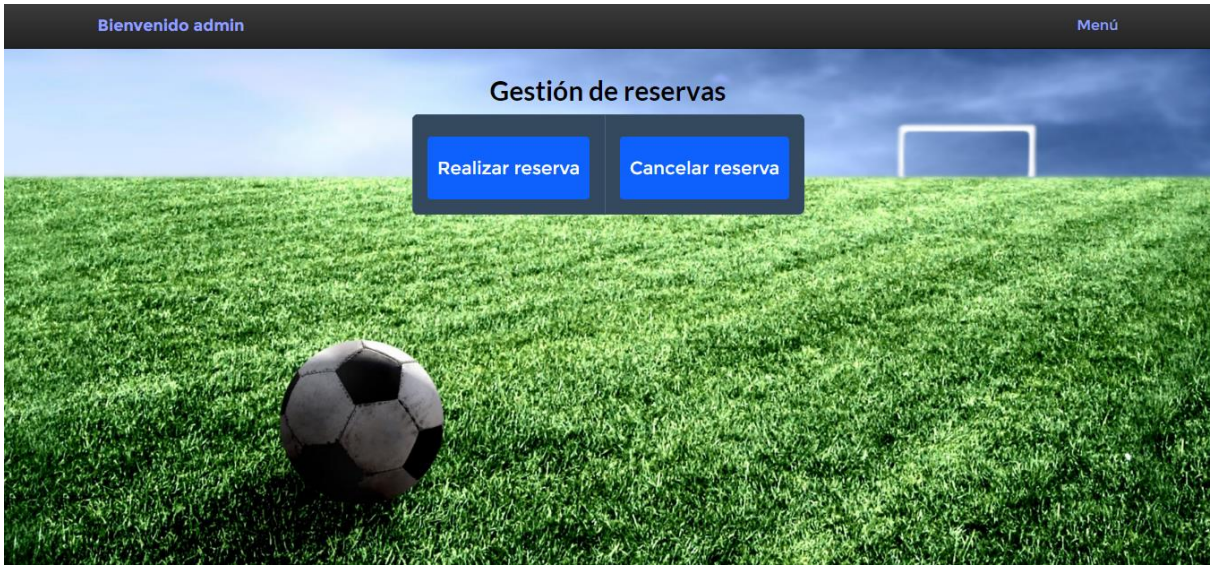

Ilustración 72- Gestión de reservas

En el apartado gestión de reservas podemos realizar una reserva pulsando sobre el botón “Realizar reserva”, a continuación se mostrará el formulario de consulta de pista indicado en el apartado 7 del usuario, ilustración 63. Tras realizar la consulta se visualizará la disponibilidad de las pistas, indicado en el apartado 8 del usuario, ilustración 64. Seleccionando una pista, se procederá a la realización de la reserva de la misma forma que se indica en el apartado 9 del usuario, ilustración 65; y además se pedirá el nombre y DNI de la reserva. Después de rellenarlo, pulse el botón “Confirmar reserva”.

Para cancelar reservas, pulse sobre el botón “Cancelar reservas”, a continuación se le pedirá que introduzca el DNI de la reserva como muestra la siguiente imagen:

Ilustración 73-Cancelar reserva

Tras introducir el DNI, si existe, se le mostrará las reservas asociadas al DNI que podrá cancelar pulsando sobre el botón “Cancelar reserva”. Si no existen reservas asociadas al DNI, se le indicará mediante un mensaje.

8. Gestión de material

Ilustración 74- Gestión de material

En el apartado gestión de material podemos realizar las siguientes opciones:

- Crear tipo de material
- Editar tipo de material
- Eliminar tipo de material
- Crear material
- Editar material
- Eliminar material

Pulsando sobre los distintos botones accederemos a dichas acciones. Los pasos a seguir para realizar dichas gestiones son análogos a los indicados en los apartados 3, 4 y 5 del administrador, ilustración 68, 69 y 70 respectivamente.

9. Reservas del día

Ilustración 75- Reservas del día

Para visualizar las reservas del día, tras pulsar el botón “Reservas del día” en el menú del administrador, seleccione un deporte y a continuación se mostrará las reservas existentes para el día actual en una tabla similar a la mostrada en la ilustración 64, pulsando sobre las reservas que nos aparecen como “Ocupadas” se visualizará la información de dicha reserva.