



**ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA**

**INGENIERÍA INFORMÁTICA**

***SPORT PLANS GENERATOR***

**GESTIÓN DE PLANES DE ENTRENAMIENTO**

**EN LA NUBE USANDO SPRING E HIBERNATE**

**Realizado por  
Francisco López Sánchez  
48989266W**

**Dirigido por  
Sergio Segura Rueda**

**Departamento  
Lenguaje y Sistemas Informáticos**

**Sevilla, septiembre 2013**



# Resumen

A partir de las necesidades de un cliente real nace la idea de automatizar todo el proceso de diseño de planes de entrenamiento deportivo creados a medida, incluido dentro de un sistema de gestión de centros deportivos.

Estas necesidades surgen del hecho de que cada cliente tiene unas características físicas, objetivos, plazos y contraindicaciones (asma, lesiones musculares, etc) distintas al resto de usuarios. Además, un plan de entrenamiento consta de gran cantidad de parámetros como por ejemplo: series, repeticiones, periodizaciones o material a utilizar. Por lo tanto, hace que el diseño de planes de entrenamiento personalizados sea un trabajo difícil, tedioso y repetitivo.

Este proyecto presenta *Sport Plans Generator*, una aplicación *Web* para la gestión de centros deportivos, donde los monitores deportivos puedan crear planes de entrenamiento a los clientes asociados. Entre otras características, esta aplicación permite la generación tanto automática como manual de planes deportivos personalizados a partir del perfil del usuario y las indicaciones del monitor.

Una vez desplegada en la nube, la aplicación permite a monitores y clientes gestionar sus planes de entrenamiento desde cualquier lugar. Además, el sistema facilita el trabajo a los monitores durante el proceso de creación de los planes deportivos sugiriendo en cada momento los parámetros más adecuados para cada cliente en concreto.

La aplicación ha sido construida usando tecnologías de desarrollo *Web* punteras lo que facilita su desarrollo y futuro mantenimiento. Cabe destacar entre otras los frameworks *Spring* e *Hibernate* para la capa de lógica y datos, *Maven* para la gestión y construcción del proyecto, un repositorio *Git* para la gestión del código fuente y control de versiones, *Projectsii* para la gestión de tareas, *Selenium* para la automatización de pruebas funcionales, *JMeter* para pruebas de rendimiento y *Sonar* para la evaluación de métricas de calidad del código fuente.

El resultado es una aplicación que cubre las necesidades del cliente generando planes deportivos a medida para cada usuario y acorde a sus necesidades.



<b>Resumen .....</b>	3
<b>Parte I – Introducción y planificación .....</b>	11
<b>1.- Introducción .....</b>	13
1.1.- Motivación .....	13
1.2.- Objetivos del proyecto .....	13
1.2.1.- Objetivos docentes .....	13
1.2.2.- Objetivos técnicos .....	14
1.3.- Desarrollo del proyecto .....	14
1.3.1.- Projectsii.....	14
1.4.- Estructura del documento .....	15
1.5.- Abreviaturas y acrónimos utilizados.....	16
<b>2.- Planificación .....</b>	18
2.1.- Planificación temporal .....	18
2.1.1.- Estimación de tiempos .....	18
2.1.2.- Desviación respecto a la estimación de tiempos .....	19
2.2.- Planificación de costes .....	20
2.2.1.- Estimación de costes .....	20
2.2.2.- Desviación respecto de la estimación de costes.....	21
<b>Parte II – Materias relacionadas .....</b>	23
<b>3.- Materias relacionadas .....</b>	25
3.1.- <i>Spring</i> .....	25
3.2.- <i>Spring Security</i> .....	30
3.3.- <i>Hibernate</i> .....	32
3.4.- <i>JSTL – Java Server Pages Standard Tag Library</i> .....	33
3.5.- <i>AJAX (Asynchronous JavaScript and XML)</i> .....	34
3.6.- Control de versiones .....	36
3.6.1.- <i>Subversion</i> .....	36
3.6.1.1.- Alojar el proyecto en Projectsii .....	36
3.6.2.- <i>Git</i> .....	38
3.6.2.1.- Alojar el proyecto en <i>github.com</i> .....	39
3.7.- Patrones de diseño .....	40
<b>Parte III - Sistema Desarrollado .....</b>	43

<b>4.- Dominio del problema.....</b>	<b>45</b>
4.1.- Descripción del problema.....	45
4.2.- Descripción del sistema actual .....	45
4.3.- Glosario de términos .....	46
<b>5.- Visión general del sistema.....</b>	<b>48</b>
5.1.- Participantes en el proyecto .....	48
5.1.1.- Participantes .....	48
5.1.2.- Organizaciones.....	48
5.2.- Objetivos del sistema.....	49
<b>6.- Elicitación de requisitos.....</b>	<b>51</b>
6.1.- Requisitos de información .....	51
6.2.- Reglas de negocio y restricciones .....	53
6.3.- Requisitos funcionales .....	53
6.3.1.- Diagramas de casos de uso .....	54
6.3.1.1.- Subsistema de gestión de planes de entrenamiento.....	54
6.3.1.2.- Subsistema de gestión de monitores y centros deportivos.....	55
6.3.1.2.- Subsistema de gestión de clientes y perfiles .....	56
6.3.1.4.- Subsistema de gestión de usuarios.....	57
6.3.2.- Definición de actores .....	57
6.3.3.- Casos de uso del sistema .....	58
6.3.3.1.- Subsistema de gestión de planes de entrenamiento.....	58
6.3.3.2.- Subsistema de gestión de monitores y centros deportivos.....	61
6.3.3.3.- Subsistema de gestión de clientes y perfiles .....	63
6.3.3.4.- Subsistema de gestión de usuarios.....	66
6.4.- Requisitos no funcionales .....	67
6.5.- Matriz de rastreabilidad requisitos-objetivos.....	68
6.6.- Matriz de rastreabilidad casos de uso – objetivos .....	69
<b>7.- Análisis de requisitos.....</b>	<b>70</b>
7.1.- Modelo estático del sistema.....	70
7.1.1.- Diagramas de tipos.....	71
7.1.1.1.- Subsistema de gestión de planes de entrenamiento.....	71
7.1.1.2.- Subsistema de gestión de monitores y centros deportivos.....	72
7.1.1.3.- Subsistema de gestión de clientes y perfiles .....	73
7.1.1.4.- Subsistema de gestión de usuarios .....	74
7.1.2.- Escenario de prueba.....	75
7.1.3.- Tipos .....	76
7.1.2.1.- Subsistema de gestión de planes de entrenamiento.....	76

7.1.2.2.- Subsistema de gestión de monitores y centros deportivos.....	78
7.1.2.3.- Subsistema de gestión de clientes y perfiles .....	78
7.1.2.4.- Subsistema de gestión de usuarios.....	81
7.2.- Modelo dinámico del sistema.....	82
7.2.1.- Subsistema de gestión de planes de entrenamiento.....	82
7.2.2.- Subsistema de gestión de monitores y centros deportivos.....	84
7.2.3.- Subsistema de gestión de clientes y perfiles .....	86
7.2.4.- Subsistema de gestión de usuarios.....	89
7.3.- Prototipos de interfaz de usuario .....	90
<b>8.- Diseño del sistema.....</b>	<b>100</b>
8.1.- Arquitectura del sistema .....	100
8.2.- Grafo de navegabilidad.....	101
8.3.- Diseño detallado .....	102
8.3.1. - Capa de presentación.....	102
8.3.1.1.- Internacionalización.....	104
8.3.2. - Capa de lógica .....	104
8.3.2.1.- Capa de Servicio.....	105
8.3.2.2.- DTO – Data Transfer Object.....	107
8.3.3. - Capa de datos.....	108
8.3.4. - Patrones de diseño utilizados .....	111
8.4.- Condicionantes .....	113
8.4.1.- Puntos de variación.....	113
8.4.2.- Puntos de evolución.....	114
8.5.- Modelo de datos.....	115
8.6.- Diagrama de despliegue .....	117
<b>9.- Implementación.....</b>	<b>118</b>
9.1.- Entorno de desarrollo .....	118
9.2.- Aspectos relevantes de la implementación.....	120
9.2.1.- Capa de datos.....	120
9.2.2.- Capa de lógica .....	121
9.2.3.- Capa de presentación .....	123
9.2.4.- Métricas del código fuente .....	124
<b>10.- Pruebas .....</b>	<b>129</b>
10.1.- Pruebas de interfaces y contenidos.....	129
10.1.1.- Verificación de contenidos.....	129
10.1.2.- Verificación de sitios en construcción .....	129
10.1.3.- Verificación de estándares.....	130

10.1.4.- Verificación de interfaces.....	130
10.1.4.1.- Librerías y plug-ins necesarios .....	131
10.1.4.2.- Imágenes escaladas .....	131
10.1.4.3.- Imágenes sin atributo ALT.....	132
10.2.- Pruebas funcionales y de operación.....	132
10.3.- Pruebas de carga y rendimiento.....	135
10.3.1.- Prueba de carga número 1.....	136
10.3.2.- Prueba de carga número 2.....	138
10.3.3.- Prueba de carga número 3.....	139
<b>11.- Manual de instalación .....</b>	<b>141</b>
11.2.- Requisitos.....	141
11.3.- Proceso de instalación .....	141
11.3.1.- Despliegue de la aplicación.....	141
11.3.2.- Instalación de la base de datos.....	142
<b>12.- Manual de uso del sistema.....</b>	<b>143</b>
12.1.- Manual de funcionamiento .....	143
12.2.- Manual de centro deportivo.....	144
12.2.1.- Dashboard.....	144
12.2.2.- Registrar un nuevo monitor deportivo .....	145
12.2.3.- Modificar la información personal de un monitor deportivo.....	146
12.2.4.- Registrar un nuevo socio.....	147
12.2.5.- Modificar la información personal socio .....	148
12.2.6.- Eliminación de monitores deportivos o socios .....	148
12.2.7.- Configuración .....	148
12.3.- Manual de monitor deportivo .....	149
12.3.1.- Dashboard.....	149
12.3.2.- Registrar un nuevo socio.....	150
12.3.3.- Modificar la información personal de un socio .....	150
12.3.4.- Modificar el perfil de un socio .....	151
12.3.5.- Generar un plan deportivo de forma automática.....	152
12.3.5.1.- Generar un plan deportivo automáticamente: Paso 1 .....	153
12.3.5.2.- Generar un plan deportivo automáticamente: Paso2 .....	154
12.3.5.3.- Consulta del plan deportivo generado.....	155
12.3.6.- Generar un plan deportivo de forma manual.....	158
12.3.6.1.- Generar un plan deportivo manualmente: Paso 1 .....	159
12.3.6.2.- Generar un plan deportivo manualmente: Paso 2 .....	160
12.3.6.3.- Generar un plan deportivo manualmente: Paso 3 .....	161
12.3.6.4.- Consulta de plan deportivo generado .....	162

12.3.7.- Configuración .....	162
12.4.- Manual de socio.....	163
12.4.1.- Dashboard.....	163
12.4.2.- Consulta de planes deportivos.....	164
<b>Parte IV – Comentarios finales .....</b>	<b>165</b>
<b>13.- Conclusiones .....</b>	<b>167</b>
<b>14.- Trabajo futuro .....</b>	<b>168</b>
<b>Anexo I – Referencias .....</b>	<b>170</b>
<b>Enlaces.....</b>	<b>170</b>
<b>Referencias Bibliográficas.....</b>	<b>172</b>
<b>Anexo II – Pruebas realizadas .....</b>	<b>173</b>
<b>Verificación de contenidos .....</b>	<b>173</b>
<b>Verificación de sitios en construcción .....</b>	<b>173</b>
<b>Imágenes escaladas .....</b>	<b>174</b>
<b>Imágenes sin atributo ALT .....</b>	<b>175</b>



## **Parte I – Introducción y planificación**



## 1.- Introducción

Hoy en día, la salud y el bienestar personal son aspectos importantes en la mayoría de los países desarrollados. Tener buenos hábitos a la hora de la alimentación y practicar deporte con asiduidad son tareas que nos ayudan a mantener una vida saludable.

Podemos ver como gimnasios y centros deportivos tienen una gran demanda en las últimas décadas. Es común que muchas de las personas que acuden diariamente a estos centros no hayan practicado deporte en ningún momento de su vida. Al llegar a este tipo de centros, probablemente reciban alguna indicación por parte de un especialista para comenzar a desarrollar su actividad. El problema es que este tipo de recomendaciones suelen ser muy generales, y sin tener en cuenta detalles que cada usuario puede tenerlos cuales le impidan seguir esas indicaciones con normalidad o le lleven a unos resultados no deseados.

### 1.1.- Motivación

La idea base de la aplicación a desarrollar surge a través de una persona ajena al proyecto, un Licenciado en Ciencias del Deporte el cual ejerció durante todo el desarrollo del proyecto como cliente del mismo. Es por lo tanto una gran oportunidad para realizar un proyecto a partir de una necesidad de un cliente real y adquirir así una experiencia muy valiosa para un futuro próximo, en el que el trato con el cliente forma parte del día a día de un ingeniero informático.

Uno de los principales motivos por los que se decide abordar este proyecto es la oportunidad de cubrir esta carencia existente en centros deportivos y gimnasios. Para ello, la aplicación desarrollada permite a los empleados de cualquier centro deportivo, elaborar planes de entrenamientos personalizados para cada cliente. Cada plan deportivo será específico para cada usuario teniendo en cuenta aspectos de éste como pueden ser la asiduidad con la que se practica deporte o los objetivos que éste quiere conseguir.

A pesar de existir en la actualidad un gran número de aplicaciones *Web* o móviles que se dedican a esta tarea, uno de los objetivos principales de nuestra aplicación es enfocarla en el usuario, y que cada plan de entrenamiento creado sea específico para las necesidades concretas de ese cliente.

Otra gran motivación que nos lleva a desarrollar este proyecto es la utilización de tecnologías no abordadas durante la carrera, pero que tienen en la actualidad una gran demanda en el mercado laboral. El *framework* principal de desarrollo elegido es *Spring Framework*, muy utilizado actualmente en las empresas. Con ello, conseguimos ampliar nuestro currículum y a la vez, ganar experiencia en el desarrollo de proyectos a partir de la idea de un cliente real, y usando una tecnología moderna y muy demandada.

### 1.2.- Objetivos del proyecto

#### 1.2.1.- Objetivos docentes

- Aplicar las técnicas y metodologías estudiadas para la planificación, diseño y ejecución de un proyecto software.
- Aprender nuevas tecnologías que no se estudian durante la carrera y que actualmente son muy demandadas en el mercado laboral.

## Parte I – Introducción y planificación

- Adquirir conocimiento, destreza y experiencia a la hora de planificar y desarrollar un proyecto software real en todas sus etapas.

### 1.2.2.- *Objetivos técnicos*

- Estudiar y profundizar en el uso de algunas tecnologías que no se estudian durante la carrera o que se ven muy por encima, como es el caso del *framework* de desarrollo *Spring*, *Hibernate* y *AJAX*, y emplearlas en el desarrollo del proyecto.
- Aplicar los conocimientos adquiridos sobre patrones de diseño y evaluar los beneficios obtenidos tras su uso.
- Profundizar en fases de desarrollo del proyecto dentro del ciclo de vida del software haciendo hincapié en el apartado de pruebas.
- Adquirir experiencia a la hora de desarrollar un proyecto software a partir de las necesidades de un cliente. Es un aspecto bastante importante para cualquier ingeniero informático y quizás de los menos tratados durante la carrera.

## 1.3.- Desarrollo del proyecto

El proyecto que nos ocupa surge de una idea externa al alumno. Un licenciado en Ciencias del Deporte (en adelante *el cliente*) se pone en contacto con el alumno para llevar a cabo el desarrollo de una idea que tiene en mente. El alumno ve interesante la propuesta, y decide contactar con el futuro tutor del proyecto para ver la viabilidad de dicha idea y emplearla como base para el desarrollo del proyecto fin de carrera.

Una vez aceptada, las reuniones entre tutor del proyecto y alumno se han ido sucediendo a lo largo de todo el proyecto. De éstas, se extraían una serie de tareas a realizar por el alumno. A dichas tareas se les asignaba un tiempo razonable en el cual debían ser realizadas, y se fijaba también una posible fecha para la siguiente reunión. En la siguiente reunión, las tareas que habían sido realizadas, eran revisadas por el tutor. Y se decidía si debían ser modificadas o bien se daban por concluidas.

Según las necesidades que iba requiriendo el proyecto, el alumno concertaba ciertas reuniones con el cliente. De estas reuniones se extraían las necesidades que el éste exigía para su producto, las cuales eran luego comentadas en las reuniones con el tutor para ver su viabilidad.

### 1.3.1.- *Projectsii*

Para la gestión del proyecto que nos ocupa, vamos a utilizar la herramienta ofrecida por la Escuela Técnica de Ingeniería Informática *Projectsii* [2].

Usaremos esta herramienta para llevar un control de las tareas que vamos planeando realizar en cada etapa del proyecto. Indicaremos la fecha de comienzo de la tarea y una vez terminada, señalaremos el tiempo que hemos empleado en total en su realización.

Esto nos servirá para realizar el cálculo de la desviación de tiempos que comentaremos en el apartado 2.2.- *Planificación temporal*.

## 1.4.- Estructura del documento

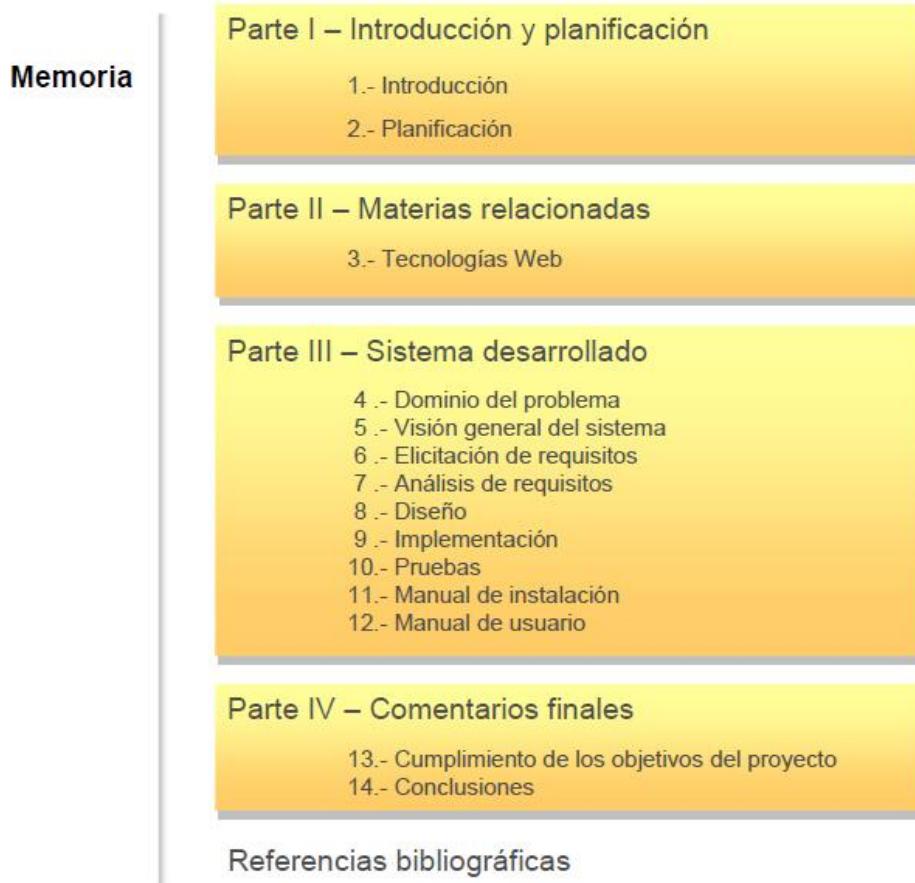
Para comprender rápidamente como está estructurado el presente documento, mostramos a continuación como está compuesto.

**PARTE I – Introducción y planificación:** en este apartado haremos una breve descripción de los motivos que me llevaron a la realización de este proyecto, así como los objetivos que se persiguen. También mostraremos aquí la planificación realizada tanto para el tiempo como para los costes.

**PARTE II – Materias relacionadas:** mencionaremos aquí aquellas tecnologías empleadas para el desarrollo de nuestro proyecto. Principalmente el *framework* de desarrollo *Spring*.

**PARTE III – Sistema desarrollado:** este apartado es la parte principal del documento. Aquí se mostrará toda la documentación de la aplicación desarrollada: elicitation y análisis de requisitos, diseño, implementación, pruebas y manuales.

**PARTE IV – Comentarios finales:** conclusiones finales sobre el proyecto realizado.



**Figura 1:** estructura del documento

## 1.5.- Abreviaturas y acrónimos utilizados

A continuación mostraremos ordenados alfabéticamente todas las abreviaturas y acrónimos utilizados en la presente documentación.

- **AJAX:** Asynchronous JavaScript And XML
- **AMETIC:** Asociación Multisectorial de Empresas de la Electrónica, Tecnologías de la Información y Comunicación.
- **API:** Application Programming Interface
- **CMP:** Container Managed Persistence
- **CSS:** Cascade Style Sheet
- **DAO:** Data Access Object
- **DDR:** Diseñador De Rutinas
- **DI:** Dependency Injection
- **DOM:** Document Object Model
- **DTO:** Data Transfer Object
- **EJB:** Enterprise JavaBeans
- **FK:** Foreign Key
- **GWT:** Google Web Toolkit
- **HQL:** Hibernate Query Language
- **HTML:** HyperText Markup Language
- **HTTP:** HyperText Transfer Protocol
- **IDE:** Integrated Development Environment
- **IoC:** Inversion of Control
- **J2EE:** Java Platform Enterprise Edition
- **JAAS:** Java Authentication and Authorization Service
- **JDBC:** Java DataBase Connectivity
- **JDO:** Java Data Object
- **JPA:** Java Persistence API
- **JSON:** JavaScript Object Notation

- **JSP:** JavaServer Pages
- **JSTL:** JavaServer Pages Standard Tag Library
- **LDAP:** Lightweight Directory Access Protocol
- **MVC:** Model View Control
- **ORM:** Object Relational Mapping
- **PDF:** Portable Document Format
- **PFC:** Proyecto Fin de Carrera
- **PK:** Primary Key
- **POJO:** Plain Old Java Object
- **POO:** Programación Orientada a Objetos
- **SQL:** Structured Query Language
- **UML:** Unified Modeling Language
- **URL:** Uniform Resource Locator
- **WAR:** Web application ARchive
- **XAMPP:** X(any of four different operating systems), Apache, MySQL, PHP and Perl
- **XML:** Extensible Markup Language
- **XSL:** Extensible Stylesheet Language

## 2.- Planificación

### 2.1.- Planificación temporal

La planificación temporal corresponde la identificación de tareas, asignación de tiempos y recursos a dichas tareas y planificación de la secuencia de ejecución de forma que el tiempo de desarrollo del proyecto sea mínimo.

En el mundo laboral, una buena planificación es muy importante para aumentar las probabilidades de éxito de un proyecto, ya que si ésta no es adecuada, puede repercutir en una reducción del impacto en el mercado, generar clientes insatisfechos o aumentar los costes internos.

En el proyecto que nos atañe, la tarea de planificar se simplifica un poco al estar formado el equipo de trabajo por una sola persona, con lo cual, no tenemos que preocuparnos por planificar tareas paralelas ni de asignar distintas tareas a distintos técnicos según sus habilidades.

Pero por otro lado, al tratarse de un proyecto en el que se usarán unas tecnologías novedosas para el autor, es difícil hacerse una idea a priori del tiempo y el esfuerzo necesario que se necesitará para realizar cada tarea.

Las distintas fases por las que va a pasar el proyecto, las hemos dividido en hitos o eventos.

Los hitos que hemos definido para este proyecto se muestran en la siguiente tabla.

Hito	Descripción
H1	Estudio del funcionamiento y uso de las tecnologías y herramientas
H2	Documento de elicitación de requisitos
H3	Documento de análisis de requisitos
H4	Documento de diseño
H5	Desarrollo de la aplicación
H6	Implantación de la aplicación
H7	Finalización del proyecto

#### 2.1.1.- Estimación de tiempos

El proyecto comenzó en marzo de 2013 y terminará oficialmente el 4 de septiembre de este mismo año, fecha en la que se cierra el plazo para la entrega de documentaciones. No obstante, la planificación será realizada teniendo como fecha límite la última semana del mes de julio de 2013, ya que el mes de agosto no es laborable para el personal universitario.

Para realizar la estimación de tiempos, se deben tener en cuenta tres aspectos, que son, como comentamos anteriormente, la *disponibilidad del equipo de desarrollo*, las *tareas que se pueden realizar en paralelo* y la normativa de *número mínimo de horas impuesta por la asignatura Proyecto Informático*.

En cuanto a la disponibilidad, como único integrante del equipo de desarrollo, se estima que se podrá trabajar unas 15 horas semanales. Esta estimación es una media, ya que según las épocas de año, se podrá dedicar más o menos tiempo. Y en cuanto a las tareas en paralelo, no es necesario tener este aspecto en cuenta, ya que todas serán desarrolladas por la misma persona.

Mostramos a continuación la descomposición en tareas del proyecto y la estimación de tiempos para cada una de ellas.

Tarea	Estimación (días)
T1. Estudio del problema	15
T2. Familiarización y configuración del entorno de desarrollo	6
T3. Desarrollo de la aplicación	106
T3.1. Elicitación	20
T3.2. Análisis	20
T3.3. Diseño	10
T3.4. Implementación	50
T3.5. Pruebas	3
T3.6. Implantación	3
T4. Documento del proyecto	20
T5. Manual de usuario	5
<b>TOTAL</b>	<b>152 días</b>

En total estimamos que emplearemos en la realización del proyecto unos 152 días.

Según la normativa de la asignatura *Proyecto Informático* del departamento de *Lenguajes y Sistemas Informáticos*, el tiempo total de dedicación al mismo debe ser de unas **450 horas**. La estimación realizada anteriormente de las horas que podremos dedicar semanalmente al proyecto es de 15 horas semanales, por lo que tenemos una media de 3 horas al día. Con estos datos realizamos la siguiente operación:  $152 \text{ días} * 3 \text{ h / día} = 456 \text{ horas}$ .

A continuación mostramos un diagrama de Gantt realizado con la herramienta online que ofrece la aplicación *Web Tomsplanner* [\[72\]](#).

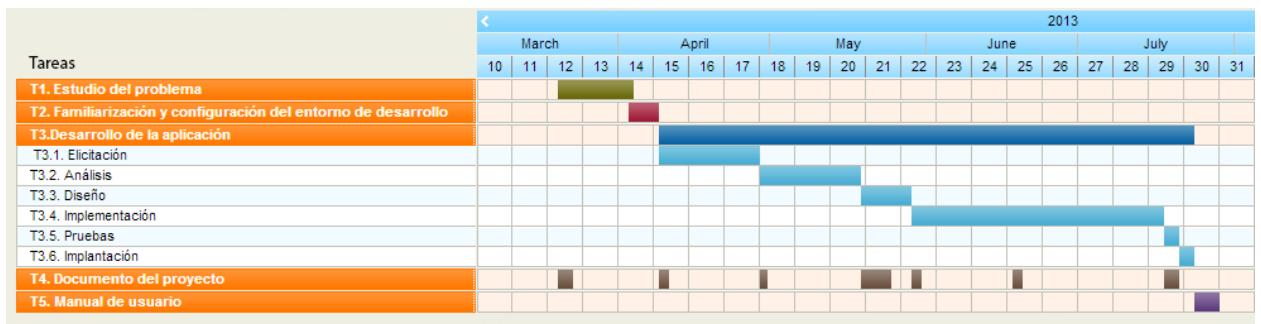


Figura 2: diagrama de Gantt

### 2.1.2.- Desviación respecto a la estimación de tiempos

A continuación mostraremos el tiempo que hemos empleado en cada una de las tareas que nos fijamos cuando comenzamos con el proyecto.

Tarea	Tiempo empleado(días)
T1. Estudio del problema	10 (-5)
T2. Familiarización y configuración del entorno de desarrollo	9 (+3)
T3. Desarrollo de la aplicación	118 (+12)
T3.1. Elicitación	18 (-2)
T3.2. Análisis	22 (+2)
T3.3. Diseño	14 (+4)
T3.4. Implementación	59 (+9)
T3.5. Pruebas	3 (0)
T3.6. Implantación	2 (-1)
T4. Documento del proyecto	24 (+9)
T5. Manual de usuario	1 (-4)
<b>TOTAL</b>	<b>162 días (+10)</b>

En total hemos empleado casi **1 semanas y 3 días** más que el tiempo total planificado al inicio del proyecto.

Aunque el cómputo global de días que hemos empleado en la realización del proyecto no se aleje mucho del planificado al inicio del proyecto, existen varias tareas que por sí solas han visto reflejadas un aumento en el tiempo empleado para su ejecución.

Es el caso por ejemplo de la tarea *T1. Estudio del problema*, a la que se le asignó un tiempo de 15 días para su realización y finalmente el tiempo empleado ha sido menor. Y el caso de la tarea *T3.4. Implementación* y la tarea *T4. Documentación del proyecto* a las que fueron asignadas un tiempo menor del que luego ha sido necesario emplear para finalizarlas.

Finalmente, podemos decir que al realizar el recuento total de los días empleados para la ejecución del proyecto (**162 días**), no se aleja demasiado de la planificación realizada inicialmente (**152 días**), dejándonos una diferencia de **10 días** empleados por encima de lo que planificamos. En total, teniendo en cuenta el tiempo medio por horas empleado al día, unas 3 horas al día, obtendríamos unas 30 horas más de lo planificado. Teniendo en cuenta la magnitud del proyecto consideramos esta desviación dentro de lo normal.

## 2.2.- Planificación de costes

En la planificación de costes realizaremos una predicción sobre los recursos tanto humanos como materiales que necesitaremos para la realización del proyecto así como del esfuerzo necesario para llevar a cabo el proyecto.

### 2.2.1.- Estimación de costes

En la siguiente tabla mostramos el coste estimado tanto para los recursos humanos como materiales que creemos necesarios para llevar a cabo la realización del proyecto. Los salarios de cada perfil involucrado en el proyecto han sido obtenidos del siguiente estudio retributivo del AMETIC [\[1\]](#).

Concepto	Perfil	Nº Personas	Nº Horas	€/hora	Cuantía
<b>Personal</b>	Jefe de Proyecto	1	15	25	375€
	Analista	1	75	20	1500€
	Programador	1	350	10	3500€
	Subtotal				<b>5375€</b>
<b>Implantación Sistema</b>	Intel® Core™ i3 2,27 GHz 4,00 GB RAM				650€
	Conexión fibra óptica				47,50€
	Servidor web				60€
	Dominio web				10€
	Subtotal				<b>767,5€</b>
<b>Gastos PFC</b>	Conexión Internet				100€
	Matrícula PFC				219,6€
	Encuadernación (Documentación + CD)				20€
	Subtotal				<b>339,6€</b>
<b>Total</b>					<b>6482,1€</b>

### 2.2.2.- Desviación respecto de la estimación de costes

En esta tabla mostramos la desviación observada con respecto a la estimación realizada a priori.

Concepto	Perfil	Nº Personas	Nº Horas	€/hora	Cuantía
<b>Personal</b>	Jefe de Proyecto	1	16	25	400€ (+25€)
	Analista	1	83	20	1660€ (+160€)
	Programador	1	359	10	3590€ (+90€)
	Subtotal				<b>5650€ (+275€)</b>
<b>Implantación Sistema</b>	Intel® Core™ i3 2,27 GHz 4,00 GB RAM				650€ (0)
	Conexión fibra óptica				47,50€ (0)
	Servidor web				60€ (0)
	Dominio web				10€ (0)
	Subtotal				<b>767,5€ (0)</b>
<b>Gastos PFC</b>	Conexión Internet				78,5€ (-21,5€)
	Matrícula PFC				219,6€ (0)
	Encuadernación (Documentación + CD)				0€ (-20€)
	Subtotal				<b>298,1€ (-41,5€)</b>
<b>Total</b>					<b>6715,6€ (+233,5€)</b>

El coste total del proyecto previsto al inicio del proyecto fue de **6482,1€**. El coste obtenido una vez finalizado dicho proyecto es de **6715,6€**. La diferencia asciende a **233,5€** más de los previstos inicialmente.

Esta diferencia se debe directamente a la desviación ocurrida en la planificación de tiempos, ya que los salarios del personal que ha trabajado en el proyecto fué calculado en función de las horas que se le iba a dedicar al mismo. Al haberse visto incrementadas dichas horas de dedicación en el proyecto, se ve afectado por tanto también el presupuesto económico.

Por ejemplo, vemos como se han visto incrementadas las horas empleadas como “Analista” en 8 horas, por lo que supone un gasto no planificado de 160€ más. Y ocurre lo mismo con las horas empleadas como “Programador”. Se han visto incrementadas en 9 horas, por lo que supone un gasto de 90€ más de lo estimado.

En los gastos referentes al propio Proyecto Fin de Carrera, uno de los hechos que ha producido el abaratamiento del presupuesto planeado para este apartado es el cambio de país por parte del alumno, en el cual se encontró con una conexión más económica que la que tenía contratada en España. También el hecho de no ser necesaria la impresión del documento del proyecto ha abaratado este apartado.

## **Parte II – Materias relacionadas**



### 3.- Materias relacionadas

En el presente capítulo, haremos un repaso de las tecnologías *Web* que emplearemos en el proyecto que nos ocupa, señalando sus características principales, algunas de las ventajas que nos aportan y la razón que nos ha llevado a elegirla.

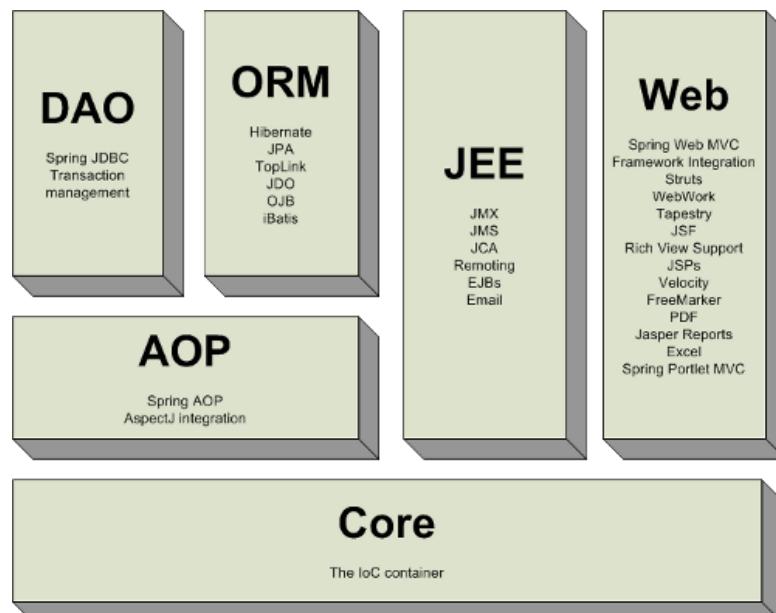
#### 3.1.- *Spring*

Los principales motivos que nos llevan a elegir este *framework* para el desarrollo de nuestra aplicación *Web* son varios. Su creciente demanda en el mercado laboral por ser una tecnología moderna, ágil y útil es uno de los motivos. También el hecho de no haber visto durante la carrera un *framework* de esta índole ha influido en la decisión.

*Spring* es un *framework* de código abierto de desarrollo de aplicaciones para la plataforma Java ([\[3\]](#), [\[4\]](#), [\[5\]](#), [\[6\]](#), [\[50\]](#)). Existe también una versión para la plataforma .NET.

*Spring* contiene muchas características que le dan una funcionalidad muy amplia; dichas características están organizadas en grandes módulos. Seguidamente se describen someramente las características de cada módulo.

#### Módulos de Spring



**Figura 3:** módulos de *Spring*

- **Core:** Parte fundamental del *framework*. Éste provee toda la funcionalidad de Inyección de Dependencias (se describirá más adelante) permitiéndole administrar la funcionalidad del contenedor de beans. Encima del módulo *Core* se encuentra el módulo *Context*, el cual provee de herramientas para acceder a los beans de una manera elegante. El paquete de contexto hereda sus características del paquete de beans y añade soporte para mensajería de texto, como son “*resource bundles*” (para internacionalización), propagación de eventos, carga de recursos y creación transparente de contextos por contenedores.

- **DAO:** provee una capa de abstracción de *JDBC* (*Java DataBase Connectivity*) que elimina la necesidad de teclear código *JDBC* tedioso y redundante así como el parseo de códigos de error específicos de cada proveedor de base de datos. También, el paquete *JDBC* provee de una manera de administrar transacciones tanto declarativas como programáticas, no solo para clases que implementen interfaces especiales, pero para todos sus *POJOs*.
- **ORM:** provee capas de integración para *APIs* de mapeo objeto – relacional, incluyendo, *JDO*, *Hibernate* e *iBatis*. Usando el paquete *ORM* es posible usar esos mapeadores en conjunto con otras características que *Spring* ofrece, como la administración de transacciones mencionada con anterioridad.
- **AOP:** provee una implementación de programación orientada a aspectos compatible con AOP Alliance, permitiendo definir *pointcuts* e interceptores de métodos para desacoplar el código de una manera limpia implementando funcionalidad que por lógica y claridad debería estar separada. Usando metadatos a nivel de código fuente se pueden incorporar diversos tipos de información y comportamiento al código.
- **Web:** provee características básicas de integración orientadas a la *Web*, como funcionalidad multipartes (para realizar la carga de archivos), inicialización de contextos mediante servlet listeners y un contexto de aplicación orientado a *Web*. Cuando se usa *Spring* junto con otro *Framework* de desarrollo como por ejemplo *Struts*, para la capa de presentación, este es el paquete que te permite una integración sencilla.
- **Web MVC:** provee de una implementación *Modelo – Vista – Controlador* para las aplicaciones *Web*. La implementación de *Spring MVC* permite una separación entre código de modelo de Dominio y las formas *Web* y permite el uso de otras características de *Spring Framework* como lo es la validación.

### Escenario de uso

Con los bloques descritos anteriormente es posible usar *Spring* en una multitud de escenarios, desde applets hasta aplicaciones empresariales complejas usando la funcionalidad de *Spring* para el manejo transaccional y el *framework Web*.

En nuestro caso, el escenario que encontraremos en nuestro proyecto será el de una aplicación *Web* usando el módulo *Spring Web MVC* (*Modelo – Vista – Controlador*), que es el patrón arquitectónico elegido para el desarrollo.

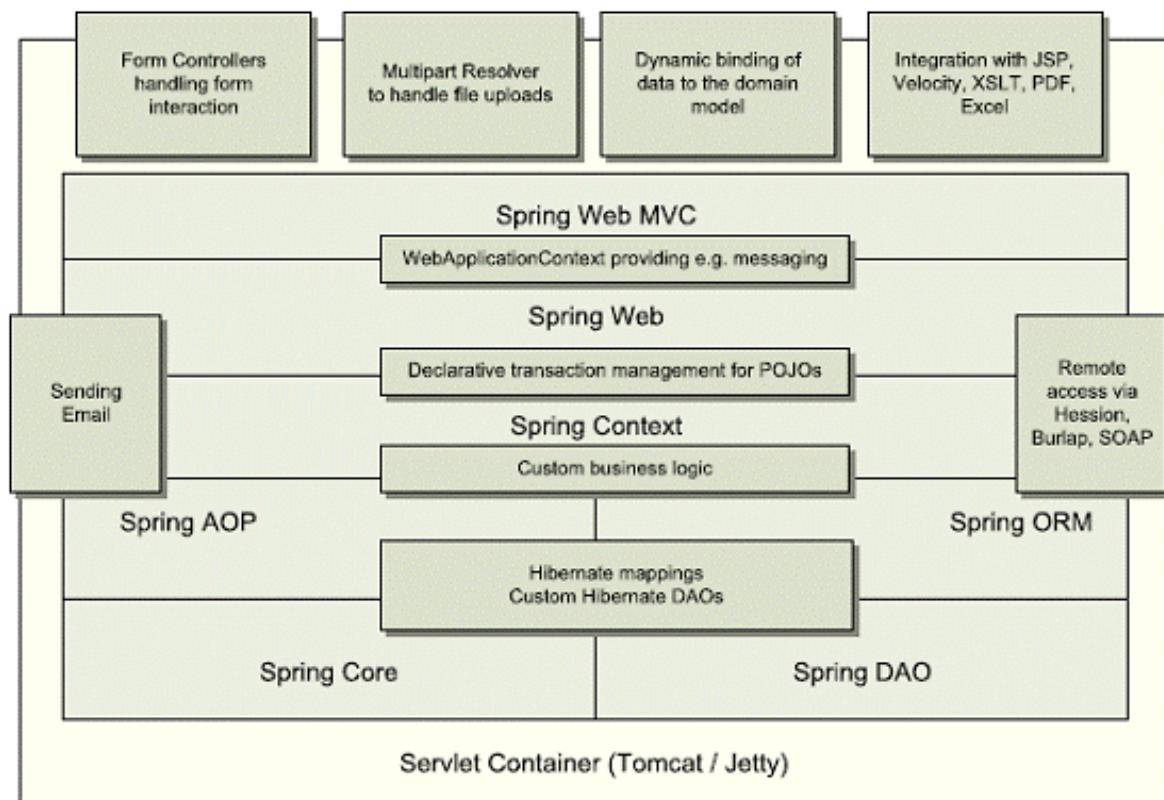
### Aplicación Web

Una aplicación *Web* típica usa gran parte de las características de *Spring*. Por medio de los *TransactionProxyFactoryBeans* la aplicación *Web* se vuelve totalmente transaccional, tal y como pudiera ser si se usaran las transacciones administradas por el contenedor como lo manejan los *Enterprise JavaBeans*. Toda la lógica de negocio puede ser implementada usando *POJOs*, administrados por el contenedor de Inyección de Dependencias de *Spring*.

Servicios adicionales como el envío de mails y la validación, independientes a la capa *Web* permiten escoger donde y cuando ejecutar las reglas de validación. El soporte *ORM* de *Spring* está integrado con *Hibernate*, *JDO* e *iBatis*. Si se usa el *HibernateDaoSupport*, es posible reutilizar los mapeos de *Hibernate* existentes.

Los controladores de formas permiten integrar la capa *Web* con el modelo de dominio, eliminando la necesidad de ActionForms o cualquier clase que transforme los parámetros *HTTP* en valores para el modelo de dominio.

### Esquema aplicación Web con Spring



**Figura 4:** esquema de aplicación Web con Spring

### Características más reseñables

#### **Inversión de control**

El módulo Core es el más importante de *Spring*. Es el que provee el Contenedor IoC (Inversion of control). Este contenedor nos permite aplicar el patrón Dependency Injection en nuestras aplicaciones.

En forma muy resumida, el objetivo del contenedor IoC es encargarse de instanciar los objetos de nuestro sistema, denominados beans, y asignarle sus dependencias. Para que el contenedor pueda llevar a cabo esta tarea, debemos, mediante archivos de información de configuración, indicarle dónde se encuentran dichos beans.

## Inyección de dependencias

Es una de las implementaciones del concepto Inversión de Control. Desde la perspectiva de la Inversión de Control se reconoce que los objetos involucrados en la lógica de negocio de una aplicación dependen de otros objetos de negocio, objetos de acceso a datos y recursos compartidos. Todos ellos se ejecutan en un ambiente denominado Contenedor Ligero, el cual es responsable de establecer las dependencias entre los objetos.

Para que el patrón Dependency Injection consiga plasmar los planteamientos de la Inversión de Control en el desarrollo de aplicaciones, se establece que todos los objetos de negocio deben proveer métodos públicos que permitan que el Contenedor determine las dependencias entre los objetos y los vincule a través de constructores y propiedades, los cuales reciben como argumentos los recursos que necesita el objeto de negocio. El primer método es denominado Constructor Injection y el segundo Setter Injection.

## Interfaz BeanFactory

El principal concepto es la interfaz BeanFactory. Es una sofisticada implementación del patrón “Factory”, nos permite instanciar objetos de forma muy rápida y fácil, liberándonos de la necesidad de especificar las dependencias en la lógica de nuestro programa y, por lo tanto, desacoplar esta dependencia en tiempo de codificación, quedando ligada al tiempo de “*inicialización*” o incluso al “*ejecución*” ya que pueden cambiarse las implementaciones en tiempo de ejecución.

El contenedor IoC de *Spring* administra uno o muchos beans. Estos beans son creados utilizando las instrucciones declaradas en la información de configuración del contenedor.

## Soporte DAO Y JDBC

El paquete *DAO* provee una capa de abstracción de *JDBC*, el patrón *DAO* es uno de los más comúnmente utilizados en aplicaciones *J2EE*, y la arquitectura de acceso a datos de *Spring* proporciona un sofisticado pero aún sencillo soporte la misma.

Los *DAO* son un patrón de diseño *J2EE* y considerados una buena práctica. La ventaja de usar objetos de acceso a datos es que cualquier objeto de negocio no requiere conocimiento directo del destino final de la información que manipula.

Los objetos *DAO* pueden usarse en Java para aislar a una aplicación de la tecnología de persistencia Java subyacente, la cual podría ser *JDBC*, *JDO*, *EJB CMP*, *TopLink*, *Hibernate*, *iBATIS*, o cualquier otra tecnología de persistencia. Usando *DAO* significa que la tecnología subyacente puede ser actualizada o cambiada sin cambiar otras partes de la aplicación.

*JDBC* es el acrónimo de Java Database Connectivity, un *API* que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java independientemente del sistema de operación donde se ejecute o de la base de datos a la cual se accede utilizando el dialecto *SQL* del modelo de base de datos que se utilice.

El *API JDBC* se presenta como una colección de interfaces Java y métodos de gestión de manejadores de conexión hacia cada modelo específico de base de datos. Un manejador de conexiones hacia un modelo de base de datos en particular es un conjunto de clases que implementan las interfaces Java y que utilizan los métodos de registro para declarar los tipos de localizadores a base de datos (*URL*) que pueden manejar.

Para utilizar una base de datos particular, el usuario ejecuta su programa junto con la librería de conexión apropiada al modelo de su base de datos, y accede a ella estableciendo una conexión, para ello

provee en localizador a la base de datos y los parámetros de conexión específicos. A partir de allí puede realizar con cualquier tipo de tareas con la base de datos a las que tenga permiso: consultas, actualizaciones, creado modificado y borrado de tablas, ejecución de procedimientos almacenados en la base de datos, etc.

*JDBC* cumple su objetivo mediante un conjunto de interfaces de java, cada una implementada de manera diferente por distintos distribuidores. El conjunto de clases que la componen se denomina el controlador *JDBC*.

## Spring Web MVC

El Modelo Vista Controlador (*MVC*) es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos.

El patrón *MVC* se ve frecuentemente en aplicaciones *Web*, donde la vista es la página *HTML* y el código que provee de datos dinámicos a la página, el modelo es el Sistema de Gestión de Base de Datos y el controlador representa la Lógica de negocio.

### Descripción del patrón *MVC*

- **Modelo:** Esta es la representación específica de la información con la cual el sistema opera. La lógica de datos asegura la integridad de éstos y permite derivar nuevos datos.
- **Vista:** Este presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario.
- **Controlador:** Este responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista.

Muchos sistemas informáticos utilizan un Sistema de Gestión de Base de Datos para gestionar los datos. En *MVC* corresponde al modelo.

Aunque se pueden encontrar diferentes implementaciones de *MVC*, el flujo que sigue el control generalmente es el siguiente:

El usuario interactúa con la interfaz de usuario de alguna forma (por ejemplo, el usuario pulsa un botón, enlace).

El controlador recibe (por parte de los objetos de la interfaz-vista) la notificación de la acción solicitada por el usuario. El controlador gestiona el evento que llega, frecuentemente a través de un gestor de eventos (handler) o callback.

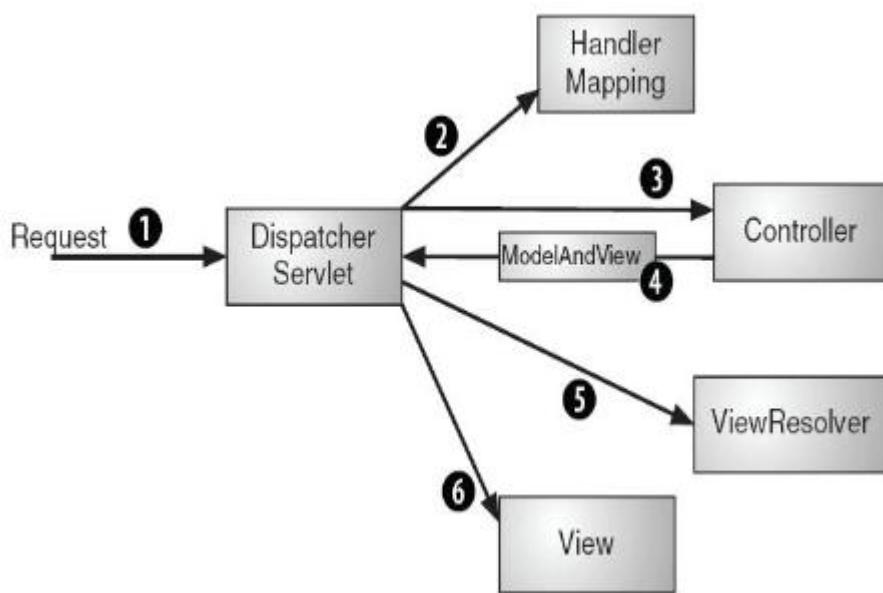
El controlador accede al modelo, actualizándolo, posiblemente modificándolo de forma adecuada a la acción solicitada por el usuario (por ejemplo, el controlador actualiza el carro de la compra del usuario). Los controladores complejos están a menudo estructurados usando un patrón de comando que encapsula las acciones y simplifica su extensión.

El controlador delega a los objetos de la vista la tarea de desplegar la interfaz de usuario. La vista obtiene sus datos del modelo para generar la interfaz apropiada para el usuario donde se refleja los cambios en el modelo. El modelo no debe tener conocimiento directo sobre la vista. Sin embargo, el

patrón de observador puede ser utilizado para proveer cierta indirección entre el modelo y la vista, permitiendo al modelo notificar a los interesados de cualquier cambio. Un objeto vista puede registrarse con el modelo y esperar a los cambios, pero aun así el modelo en sí mismo sigue sin saber nada de la vista. El controlador no pasa objetos de dominio (el modelo) a la vista aunque puede dar la orden a la vista para que se actualice.

La interfaz de usuario espera nuevas interacciones del usuario, comenzando el ciclo nuevamente.

#### Esquema del patrón MVC implementado por Spring



**Figura 5:** esquema del patrón *MVC* implementado por *Spring*

*Spring Web* es el componente específico para el desarrollo de aplicaciones *Web* en *Spring*. El desarrollo de aplicaciones *Web* con este módulo se realiza utilizando el patrón *MVC* y puede usarse conjuntamente con todas las funcionalidades que proporciona el *Framework*. Algunas características de *Spring Web* son:

- Proporciona una jerarquía de controladores para ser usados en diferentes escenarios.
- Su librería de tags *JSP* permite la escritura de las vistas mucho más clara y sencilla.
- Mapeador de peticiones muy configurable.
- Permite trabajar con cualquier tecnología en la vista como *JSP/JSTL*, *XSL*, *Freemarker*, *Velocity* o *PDF*.
- Se puede integrar con otros *Framework Web* como *Tapestry* o *Struts*.

### 3.2.- *Spring Security*

*Spring Security* ([\[7\]](#), [\[8\]](#), [\[9\]](#), [\[10\]](#)) proporciona servicios de seguridad para aplicaciones de software empresariales basados en *J2EE*, enfocado particularmente sobre proyectos construidos usando *Spring Framework*.

Es un subproyecto del *Framework Spring*, que permite gestionar completamente la seguridad de nuestras aplicaciones Java, y cuyas ventajas principales son las siguientes:

Es capaz de gestionar seguridad en varios niveles: *URLs* que se solicitan al servidor, acceso a métodos y clases Java, y acceso a instancias concretas de las clases.

Permite separar la lógica de nuestras aplicaciones del control de la seguridad, utilizando filtros para las peticiones al servidor de aplicaciones o aspectos para la seguridad en clases y métodos.

La configuración de la seguridad es portable de un servidor a otro, ya que se encuentra dentro del WAR o el EAR de nuestras aplicaciones.

Soporta muchos modelos de identificación de los usuarios (*HTTP BASIC*, *HTTP Digest*, basada en formulario, *LDAP*, *OpenID*, *JAAS* y muchos más). Además podemos ampliar estos mecanismos implementando nuestras propias clases que extiendan el modelo de *Spring Security*.

### Arquitectura de *Spring Security*



**Figura 6:** arquitectura de *Spring Security*

Para comenzar a usar *Spring Security* en nuestra aplicación, la configuración que debemos realizar es mínima. Resumimos a continuación los pasos más importantes:

- Configurar un filtro interceptor de las peticiones *Web*, dentro de nuestro archivo *web.xml*.
- Crear el fichero para la configuración de seguridad (*spring-security.xml*) con una configuración mínima y cargarlo en el *web.xml*.
- Configurar el *login* y el *logout* explícitamente.

Con este módulo que *Spring* nos proporciona, nos ahorraremos cantidad de código y lógica para controlar todo lo referente a login de usuarios, validaciones de contraseñas encriptadas, etc. Además de ofrecernos la posibilidad de restringir el acceso a distintas páginas y contenidos según el tipo de usuario (*ROLE*) que esté solicitando el contenido en ese momento.

### 3.3.- Hibernate

Hibernate es una herramienta de Mapeo objeto-relacional (*ORM*) para la plataforma Java que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación, mediante archivos declarativos (*XML*) o anotaciones en los beans de las entidades que permiten establecer estas relaciones ([\[11\]](#), [\[12\]](#), [\[13\]](#)).

La elección de esta herramienta para el desarrollo de la capa de datos de nuestra aplicación es debido a su previo conocimiento al haber sido estudiada en la asignatura “Ingeniería del Software II”. En dicha asignatura pudimos comprobar su gran utilidad y facilidad de uso. Y las ventajas que ofrecía con muy poco esfuerzo por parte del desarrollador. Es por ello por lo que hemos decidido usar esta herramienta para nuestra capa de datos.

Como todas las herramientas de su tipo, Hibernate busca solucionar el problema de la diferencia entre los dos modelos de datos coexistentes en una aplicación: el usado en la memoria de la computadora (orientación a objetos) y el usado en las bases de datos (modelo relacional).

Para lograr esto permite al desarrollador detallar cómo es su modelo de datos, qué relaciones existen y qué forma tienen. Con esta información Hibernate le permite a la aplicación manipular los datos en la base de datos operando sobre objetos, con todas las características de la *POO* (Programación Orientada a Objetos).

Hibernate convertirá los datos entre los tipos utilizados por Java y los definidos por *SQL*. Genera las sentencias *SQL* y libera al desarrollador del manejo manual de los datos que resultan de la ejecución de dichas sentencias, manteniendo la portabilidad entre todos los motores de bases de datos con un ligero incremento en el tiempo de ejecución.

Está diseñado para ser flexible en cuanto al esquema de tablas utilizado, para poder adaptarse a su uso sobre una base de datos ya existente. También tiene la funcionalidad de crear la base de datos a partir de la información disponible.

Ofrece también un lenguaje de consulta de datos llamado *HQL* (Hibernate Query Language), al mismo tiempo que una *API* para construir las consultas programáticamente (Criteria). Ambas usadas en este proyecto.

### Integración de *Hibernate* con *Spring*

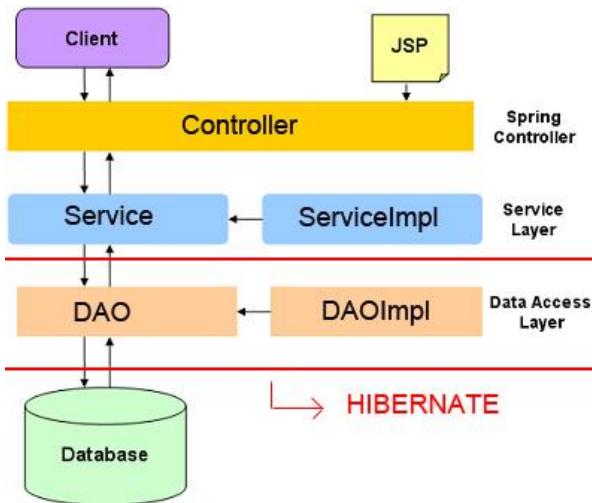


Figura 7: integración de *Hibernate* con *Spring*

En la imagen anterior podemos observar cómo se integra *Hibernate* en la arquitectura *MVC* usada en nuestro proyecto. Vemos como el flujo comienza en el cliente, que realiza una llamada a un controlador declarado usando los medios que *Spring* nos proporciona.

Posteriormente, este controlador, por medio de la capa de servicio, hace una llamada al método correspondiente en la capa de acceso a datos, que es donde entra en acción *Hibernate*. Éste realiza las consultas necesarias a base de datos y se realiza de nuevo el flujo anterior pero en el sentido contrario, devolviendo los datos correspondientes a la vista.

### 3.4.- JSTL – Java Server Pages Standard Tag Library

Es un componente de *Java EE*. Extiende las *JavaServer Pages* proporcionando cuatro bibliotecas de etiquetas con utilidades ampliamente utilizadas en el desarrollo de páginas *Web* dinámicas ([\[14\]](#), [\[15\]](#), [\[16\]](#)). Este componente lo usaremos a la hora de elaborar nuestras vistas usando páginas *JSP*.

Las bibliotecas que engloba *JSTL* son:

- **core**: iteraciones, condicionales, manipulación de *URL* y otras funciones generales.
- **xml**: para manipulación de *XML*.
- **sql**: para gestión de conexiones a base de datos.
- **fmt**: para la internacionalización y formateo de las cadenas de caracteres como cifras.

Para poder hacer uso de una de estas librerías en una página *JSP*, necesitamos hacer referencia a ella previamente. Por ejemplo mediante el siguiente código estamos cargando la biblioteca *JSLT* “*core*” y para utilizarla posteriormente, vamos a hacerlo a través del prefijo “*c*”.

```
<%@ taglib uri="HTTP://java.sun.com/JSP/JSTL/core" prefix="c" %>
```

## Parte II – Materias relacionadas

Algunas de las ventajas de usar *JSTL* son:

- **Uso de etiquetas estandarizadas:** *JSTL* es publicado por *SUN* como una parte de *JSP*, por lo que los desarrolladores se benefician de la estandarización de las etiquetas además de una comunidad de desarrolladores activa.
- **Funcionalidad:** provee la mayoría de funcionalidad necesaria para desarrollar páginas *JSP*, por lo que no es necesario crear etiquetas propias.
- **XML:** provee soporte para *XML*.
- **Fácil aprendizaje:** *JSTL* es fácil de aprender y utilizar.
- **Etiquetas personalizadas:** *JSTL* provee de mecanismos que permiten al desarrollador crear sus propias etiquetas en función de las necesidades de éste.

## 3.5.- AJAX (Asynchronous JavaScript and XML)

Las aplicaciones *Web* proliferan debido a su simplicidad, pero ofrecen una menor interactividad y usabilidad en comparación con las aplicaciones de escritorio, debido a que la interacción del usuario con una aplicación *Web* se interrumpe cada vez que se necesita algo del servidor.

Varias tecnologías han sido diseñadas para resolver este problema, Java Applets, FLASH...

AJAX es una nueva solución que no requiere plugins o capacidades específicas de ciertos navegadores ([\[17\]](#), [\[18\]](#), [\[19\]](#), [\[20\]](#), [\[21\]](#), [\[22\]](#), [\[23\]](#)). Es una técnica de desarrollo *Web* que genera aplicaciones *Web* interactivas combinando:

- Document Object Model (*DOM*) para visualizar dinámicamente e interactuar con la información presentada.
- *XML*, *XSLT* para intercambiar y manipular datos.
- *CSS* para definir el aspecto del documento.
- *JSON* y *JSON-RPC* pueden ser alternativas a *XML/XSLT*.
- *XMLHttpRequest* para recuperar datos de forma asincrónica.
- *JavaScript* como nexo de unión de todas estas tecnologías.

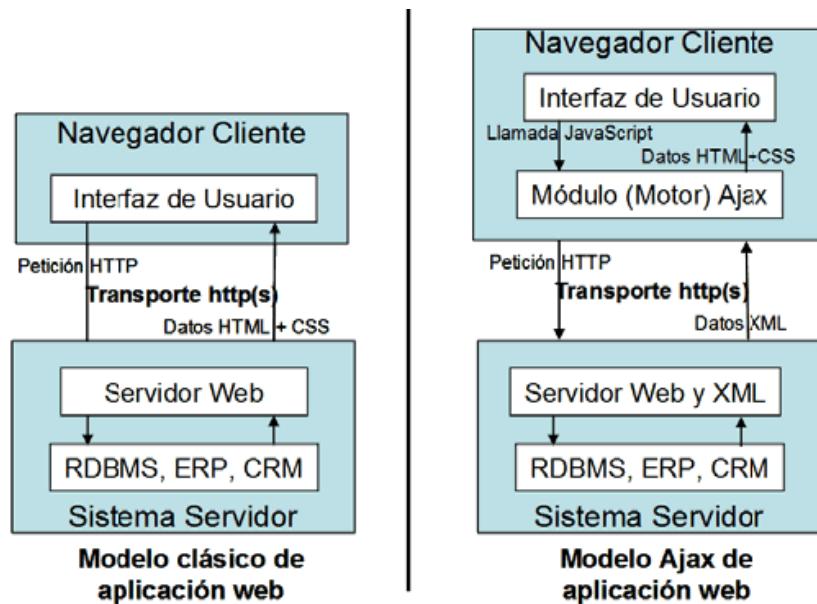
Con la idea de que la aplicación que vamos a desarrollar sea una experiencia agradable para el usuario, de fácil uso, y que no necesite grandes cantidades de tiempo para mostrar los datos al usuario, elegimos usar esta tecnología para nuestro proyecto.

Algunas de las principales ventajas que conseguimos usando *AJAX* son las siguientes:

- Las aplicaciones son más interactivas, responden a las interacciones del usuario más rápidamente, al estilo aplicaciones de escritorio.
- Estas aplicaciones tienen un aspecto (look and feel) muy similar a las aplicaciones de escritorio tradicionales sin depender de plugins o características específicas de los navegadores.
- Se reduce el tamaño de la información intercambiada.

- Se libera de procesamiento a la parte del servidor (se realiza en la parte cliente).
- AJAX actualiza porciones de la página en vez de la página completa.

En la siguiente imagen, podemos ver como es el comportamiento de una aplicación Web clásica, que realiza peticiones al servidor y recarga la página completa, en contraposición con otra aplicación Web que lo hace usando AJAX.



**Figura 8:** comparación entre aplicación Web clásica y aplicación Web usando AJAX

Como podemos observar, en el modelo de aplicación Web usando AJAX, colocamos un módulo que hace las funciones de motor de AJAX, y realiza las peticiones *HTTP* mediante eventos, scripts o rutinas al servidor *Web* en un segundo plano, por lo que el usuario puede continuar realizando su actividad normal en la página, y una vez que los datos a mostrar estén disponibles, son procesados por medio de *JavaScript* y renderizados en la página.

Esto redunda en una mayor interacción gracias a la reducción de información intercambiada entre servidor y cliente, y a que parte del proceso de la información se hace en el propio cliente, liberando al servidor de ese trabajo.

La contrapartida es que la descarga inicial de la página es más lenta al tenerse que bajar todo el código *JavaScript*.

En el proyecto que nos ocupa, usaremos esta tecnología por medio de la librería *jQuery*. Esta librería es de código abierto, libre y con gran cantidad de información, manuales y tutoriales en la red. Además existe una gran comunidad activa de desarrolladores *Web* que usan esta librería.

### 3.6.- Control de versiones

En este apartado trataremos las distintas tecnologías usadas para el control de versiones. Hemos optado por usar dos opciones distintas con el objetivo de probar diferentes tecnologías y ganar experiencia en ambas, ya que la demanda por parte de las empresas es equitativa en este caso.

El control de versiones software es un sistema que se encarga de la gestión de los diversos cambios que se realizan sobre los elementos de algún producto o configuración del mismo [\[64\]](#). Este tipo de herramientas nos ayudan a controlar las diversas modificaciones o actualizaciones que se van produciendo en un proyecto software.

Aunque se ha optado por probar dos opciones distintas para el control de versiones, *Subversion* y *Git*, la primera de ella la usaremos durante el proceso de desarrollo y la segunda sólo con fines académicos.

A continuación comentaremos un poco sobre cada tecnología y su uso.

#### 3.6.1.- Subversion



Es un sistema de control de versiones diseñado específicamente para reemplazar el popular *CVS*. Se trata de una herramienta de software libre conocida también como *svn* por ser el nombre de la herramienta utilizada en la línea de comando [\[62\]](#).

Para utilizar este método optaremos por la herramienta de software libre *TortoiseSVN*. Se trata de un cliente de *Subversion* que se integra perfectamente en el explorador de *Windows* y nos permite manejar las versiones de nuestro código cómodamente desde el propio explorador.

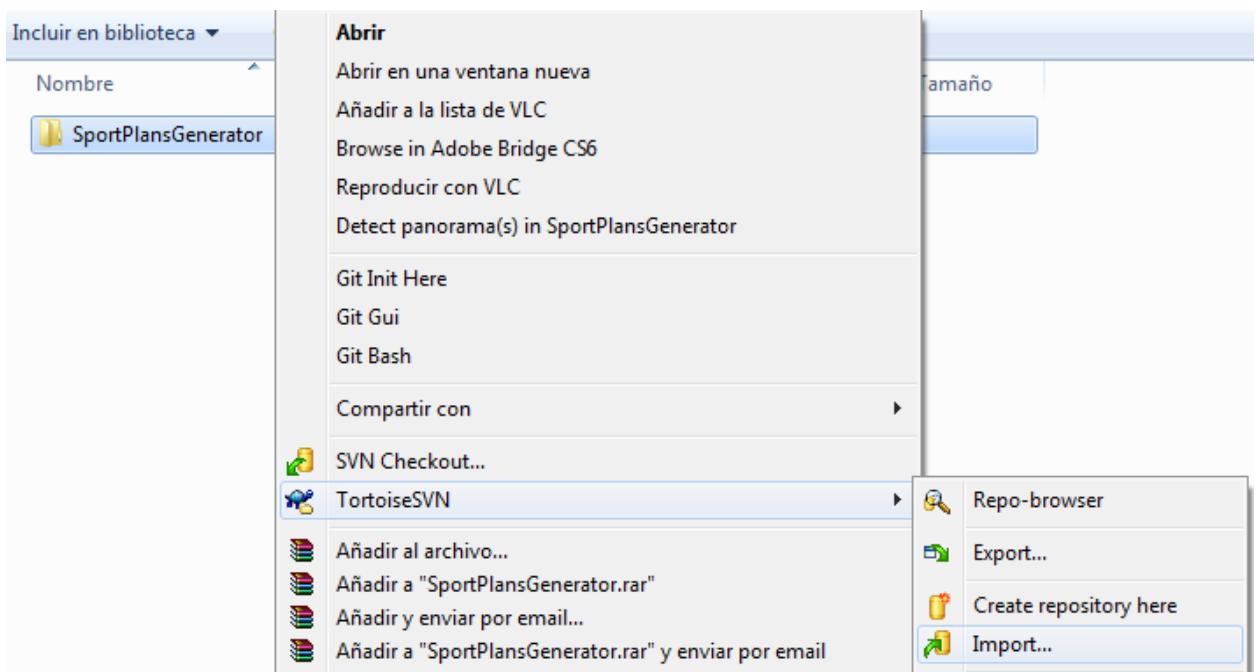


Como repositorio externo utilizaremos los servicios ofrecidos para este fin por la Universidad de Sevilla a través de la aplicación *Web Projectsii* [\[2\]](#).

##### 3.6.1.1.- Alojar el proyecto en *Projectsii*

Para alojar nuestro proyecto en el repositorio que nos ofrece *Projectsii* usando la herramienta *TortoiseSVN* realizaremos los siguientes pasos.

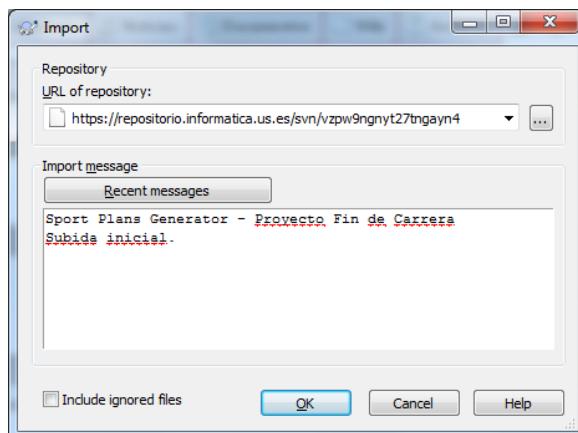
Como primer paso, debemos conectar nuestro código con el repositorio al que lo vamos a subir. Para esto, nos dirigiremos a la carpeta raíz de nuestro proyecto y haremos click con el botón secundario para ver las opciones que nos muestra el software *TortoiseSVN*.



**Figura 9:** alojar proyecto en repositorio usando la herramienta *TortoiseSVN*

Pulsando en la opción “*Import...*” nos aparecerá un cuadro de dialogo donde indicar la dirección del repositorio en el que queremos alojar nuestro proyecto.

Podemos verlo en la siguiente imagen.

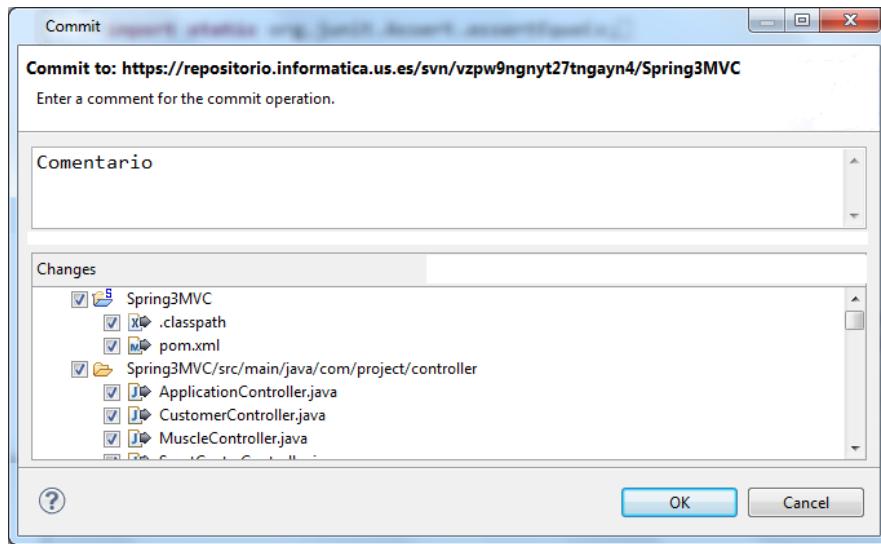


**Figura 10:** selección del repositorio donde alojar el código

Una vez indicado el repositorio donde alojar nuestro código, la herramienta *TortoiseSVN* creará automáticamente una carpeta *.svn* que será usada para llevar un control de las versiones de nuestro código.

A partir de este momento, podremos subir el código de nuestro proyecto al repositorio usando el cliente *TortoiseSVN* simplemente haciendo click con el botón secundario de nuevo sobre la carpeta raíz de nuestro proyecto y pulsar sobre la opción “*Commit...*”. Un nuevo dialogo nos mostrará los cambios que aportará la nueva versión y nos ofrecerá la posibilidad de incluir un comentario.

Podemos ver un ejemplo en la siguiente imagen.



**Figura 11:** cuadro de dialogo de la opción *Commit* de *TortoiseSVN*

Cada vez que queramos actualizar la versión existente en el servidor con los nuevos cambios en el código que tenemos en nuestra máquina, simplemente deberemos realizar el mismo paso descrito anteriormente. *TortoiseSVN* se encargará automáticamente de unir ambas versiones en una nueva versión más actualizada y además nos informará de cualquier conflicto existente.

### 3.6.2.- *Git*

Se trata de un software de control de versiones pensado para la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando estas tienen un gran número de archivos de código fuente [56].

El objetivo de usar este software de control de versiones para nuestro proyecto es principalmente académico. El creciente uso de esta herramienta por parte de la comunidad de programadores de software libre ha hecho que sea cada vez más solicitado por las empresas. Es por ello por lo que hemos decidido a usar esta herramienta para controlar las versiones de nuestro proyecto.

Para alojar nuestro proyecto en la *Web*, hemos usado los servicios ofrecidos por la aplicación *Web github.com* [58]. Se trata de la mayor comunidad de desarrolladores de proyectos de software libre online. Esta *Web* ofrece servicios para alojar tus repositorios gratuitamente con la condición de que serán proyecto públicos, es decir, que el código del mismo podrá ser accesible por toda la comunidad.

Existen en la actualidad muchas formas de usar este software para controlar las versiones de tu proyecto. Desde usar un plugin instalado en el propio *IDE* (*Integrated Development Environment*) o descargar un cliente y ejecutar nuestras acciones desde línea de comandos.

Hemos preferido realizar el control de versiones usando la línea de comandos.

A continuación mostramos los pasos seguidos para alojar nuestro proyecto en la aplicación *Web github.com*.

### 3.6.2.1.- Alojar el proyecto en [github.com](#)



El primer paso a realizar es crear una cuenta en la aplicación *Web*, para ello simplemente necesitamos una dirección de correo.

Una vez creada la cuenta, procederemos a la descarga del cliente *Git*. El cliente que hemos usado es el conocido por el mismo nombre que la propia tecnología: *Git*. Podemos descargarlo en la siguiente dirección *Web* [\[60\]](#).



Descargado el cliente, lo ejecutamos y nos dirigimos a la carpeta donde se encuentra nuestro proyecto. Esta consola está basada en *UNIX*, por lo tanto, los comandos a usar para navegar entre carpetas y listar directorios son los mismos que en el sistema *UNIX*. Los comandos típicos que usaremos serán:

- *cd*: para cambiar entre directorios
- *ls*: para listar directorios

Una vez nos encontramos en la carpeta donde se encuentra nuestro proyecto, el siguiente paso será iniciar el repositorio. Para ello usaremos el comando:

```
$ git init
```

Con este comando crearemos una carpeta *.git* en el directorio donde se encuentra el código de nuestro proyecto. Esta carpeta será usada por el cliente *Git* para llevar un control de las versiones de nuestro código.

El siguiente paso es indicar el repositorio donde vamos a alojar nuestro proyecto. Para realizar esta acción usaremos el siguiente comando:

```
$ git remote add origin git@github.com:tu_nick/tu_proyecto.git
```

Los parámetros a indicar en este comando serán el nombre de usuario que hemos usado al registrarnos en la aplicación *Web* *github.com* y el nombre que queremos darle al proyecto.

Un vez que hemos señalado el repositorio online en el que vamos a alojar nuestro proyecto, el siguiente paso consiste en indicar aquellos directorios y archivos que queremos subir al repositorio para que entren dentro de los archivos que *Git* mantendrá versionados. Para realizar este paso usamos el siguiente comando:

```
$ git remote add <nombre_fichero>
```

En este caso los parámetros a indicar son el nombre de los ficheros que queremos subir al repositorio. En nuestro caso se corresponde con la carpeta raíz del proyecto.

Una vez indicados todos los ficheros que queremos alojar en el repositorio, podemos comenzar con la subida. Un paso previo a realizar es indicar al cliente *Git* que vamos a realizar cambios. Todos los cambios realizados pasarán a ser una nueva versión. Para facilitar la posterior revisión de versiones asignaremos un comentario a cada nueva subida.

El comando que tenemos que ejecutar para realizar esta operación es el siguiente:

```
$ git commit -m "<Comentario>"
```

## Parte II – Materias relacionadas

En este punto, estaremos listos para comenzar la subida de nuestro proyecto al repositorio. Para realizar esta tarea, usaremos el siguiente comando:

```
$ git push origin master
```

Indicaremos con esta línea que queremos subir el código que tenemos en local y actualizar la que hay en el servidor. El parámetro *origin* indica que queremos subirlo al repositorio que indicamos anteriormente con el comando *git remote add origin*. El parámetro *master* indica que se trata de la rama principal a donde vamos a subir el código.

Una vez ejecutado este comando, comenzará la subida del código. A la finalización, se nos mostrará en la consola un mensaje confirmando que el proceso concluyó correctamente. Tendremos en este punto nuestro código alojado en el repositorio y listo para el control de versiones.

### 3.7.- Patrones de diseño

#### Patrón MVC – Model View Controller

El Modelo Vista Controlador (*MVC*) es un patrón de arquitectura de software que separa los *datos* y la *lógica de negocio* de una aplicación de la *interfaz de usuario* y el módulo encargado de gestionar los eventos y las comunicaciones [26].

El *framework* que vamos a usar para el desarrollo nos da la posibilidad de usar uno de sus módulos, *SpringMVC*, que se trata de una aplicación de este patrón dentro del *Framework* de *Spring*.

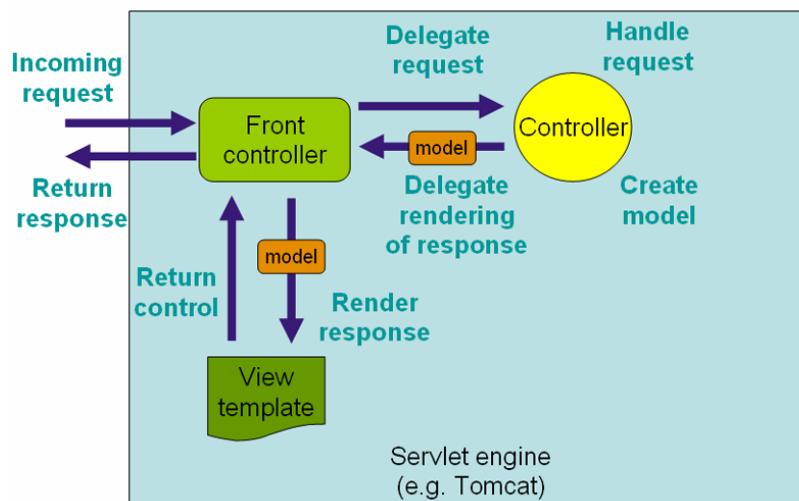


Figura 12: ejemplo de proceso de petición en *Spring MVC* [27]

En la imagen anterior podemos ver un ejemplo del proceso seguido durante una petición *HTTP* dentro de una aplicación *Web* creada usando este módulo de *Spring*.

A continuación explicaremos detalladamente la función de cada una de las capas que separa este patrón dentro del *framework Spring MVC*.

### Capa Modelo (Model)

El modelo representa los datos o reglas de negocio así como el estado de la aplicación. Permite que la capa de control interactúe con la base datos y acceda a información contenida en la misma.

### Capa Vista (View)

Esta capa representa los datos que obtiene desde el modelo a través del controlador al usuario en un formato especificado.

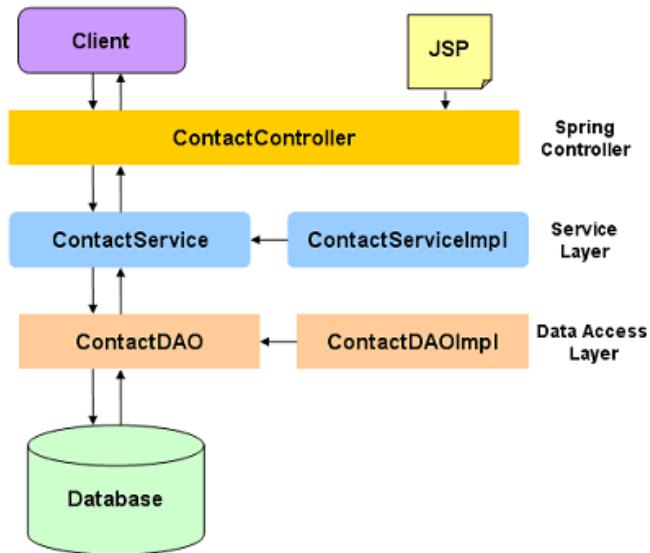
### Capa Controlador (Controller)

El controlador maneja las solicitudes de las acciones realizadas por el usuario en la vista, actualiza el modelo y dirige a los usuarios la vista apropiada basándose en el resultado de la ejecución.

### Patrón DAO – Data Access Object

El patrón de diseño Data Access Object implementa el mecanismo de acceso requerido para trabajar con la fuente de datos. Oculta completamente los detalles de implementación de la fuente de datos al cliente, ya que dicha fuente puede ser de diversos tipos (base de datos, ficheros o servicio Web). Desde el punto de vista de la aplicación, la fuente a la que se acceda es transparente, es decir, el *DAO* actúa como adaptador entre el componente y la fuente de datos consultada.

A continuación vemos una imagen de la aplicación de este patrón dentro del *framework* de desarrollo usado.



**Figura 13:** aplicación patrón *DAO* en *Spring MVC*

## Parte II – Materias relacionadas

## **Parte III - Sistema Desarrollado**



## 4.- Dominio del problema

### 4.1.- Descripción del problema

Actualmente existe un gran número de aplicaciones móviles o de escritorio con las que podemos crear en pocos pasos un completo plan de entrenamiento físico y poder empezar a desarrollar nuestra actividad física al instante, como por ejemplo *Xculpture* [73] para *iOS* o *Gym: Guía de ejercicios* [74] para *Android*. Este tipo de aplicaciones están pensadas para llegar a la mayor cantidad de público posible, por lo que los planes de entrenamiento que nos ofrecen o que nos posibilitan crear son de carácter general, es decir, puede que no se adapten adecuadamente a nuestras necesidades específicas.

Cada persona tiene unas características específicas, unos objetivos que desea cumplir, unos plazos, y también unas contraindicaciones que le impidan realizar unos tipos de ejercicios u otros. Es de esta idea de donde nace el proyecto que nos ocupa. Proponemos una aplicación centrada en las necesidades del usuario, tanto en lo que desea conseguir como también en lo que no le está indicado realizar. En definitiva, nuestra aplicación pretende focalizar mucho más las necesidades del usuario para que a través de la figura del monitor deportivo, desarrollar planes de entrenamiento “personalizados” para cada usuario.

### 4.2.- Descripción del sistema actual

Como ya se mencionó con anterioridad, la idea surge a partir una persona ajena al proyecto, que actúa como cliente. Se trata de un licenciado en Ciencias del Deporte, el cual trabaja como autónomo diseñando planes de entrenamiento físico de forma manual. Ésta era una tarea muy tediosa y prolongada en el tiempo. Había que decidir que ejercicios tenía que realizar el destinatario del plan deportivo en cada momento intentando no repetir con mucha frecuencia los ejercicios, para evitar así el aburrimiento. En resumen, un trabajo muy arduo para cada uno de los planes de entrenamiento que se quisieran crear.

Debido a esto, se decide se decide a usar algún tipo de software existente en el mercado. La herramienta elegida fue el diseñador de rutinas DDR 1.5 [24]. Esta herramienta es de uso libre y gratuito y cuenta con una amplia comunidad de usuarios.

DDR (Diseñador De Rutinas) es un conjunto de hojas de cálculo para Excel que permite crear y diseñar una rutina de musculación, analizando los grupos musculares implicados en los ejercicios, las repeticiones o el tiempo total estimado para el entrenamiento.

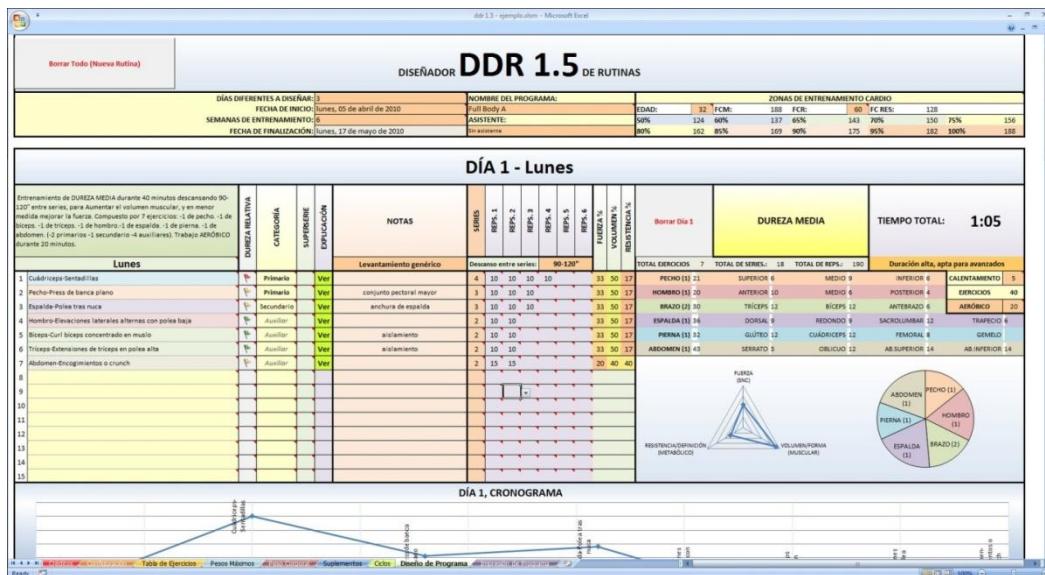
Con esta herramienta se redujo considerablemente el tiempo empleado para realizar cada rutina, pero el resultado obtenido seguía estando poco enfocado al usuario del plan deportivo además de ser poco intuitivo y manejable para éste, ya que seguían tratándose de hojas Excel.

Desde el punto de vista del cliente, la creación de rutinas seguía siendo en cierto modo manual, no existía la posibilidad de automatizarla además de seguir siendo necesarios unos conocimientos técnicos previos para la creación de estos planes.

Por lo que con la idea de poder automatizar aun más la generación de planes deportivos y poder llegar a la mayor cantidad de personas posibles con un esfuerzo menor, el cliente decide contactar con el alumno en cuestión para dar forma a su idea y crear una aplicación que cumpla sus necesidades.

### Parte III – Sistema desarrollado

Nace aquí la idea inicial de la aplicación que desarrollaremos en el proyecto que nos ocupa. Una herramienta *Web* que permita a monitores deportivos con unos conocimientos previos en la materia, crear planes de entrenamiento adaptados a cada cliente de una forma sencilla, rápida y efectiva.



**Figura 14:** diseñador de rutinas DDR 1.5

### **4.3.- Glosario de términos**

**Centro deportivo:** lugar que permite practicar deportes o hacer ejercicio.

**Monitor deportivo:** persona encargada de la seguridad de los usuarios del centro deportivo y de asesorar a éstos para el correcto desarrollo del ejercicio.

**Socio:** persona que realiza los planes de entrenamientos guiada por los monitores deportivos de cada centro deportivo.

**Ejercicio físico:** es la actividad física que se realiza con el fin de estar en buena forma física y adquirir resistencia, flexibilidad, fuerza, etc.

**Diseñador de Rutinas (DDR 1.5):** es un conjunto de hojas de cálculo para Excel que permite crear y diseñar con gran control una rutina de musculación.

**Perfil de cliente:** conjunto de características personales como el peso o la altura.

**“Principiante” “Intermedio” y “Avanzado”**

**Plan de entrenamiento:** sucesión detallada de ejercicios a realizar en un periodo de tiempo para cumplir unos determinados objetivos.

**Objetivo:** metas marcadas por el usuario a la finalización del plan. Puede tomar los valores “Adaptación”, “Aumento de la fuerza”, “Hipertrofia”, “Mantenimiento”, “Pérdida de peso” y “Salud”.

**Cualidad física:** facultad física a mejorar con el plan de entrenamiento. Puede tomar los valores “Fuerza”, “Resistencia” y “Elasticidad”.

**Periodo:** secciones en las que está dividido un plan de entrenamiento.

**Sesión:** día de la semana en el que se ha planificado un entrenamiento.

**Rutina:** conjunto de ejercicios que conforman una sesión de entrenamiento.

**Orden de la rutina:** forma en la que se realizan los ejercicios de una sesión. Puede ser del tipo “Círculo” en la que cada uno de los ejercicios se realiza sucesivamente o bien “Series” en la que cada ejercicio es realizado un número de veces consecutiva pasando luego al siguiente ejercicio.

**Tipo de la rutina:** grupo de ejercicios que engloba la rutina. Puede ser del tipo “Monoarticular” o “Poliarticular”.

**Velocidad de ejecución de la rutina:** rapidez con la que debe ejecutarse cada uno de los ejercicios de la rutina.

**Equipamiento:** material necesario para realizar un ejercicio físico.

**Repetición:** ejecución de un ejercicio físico.

**Serie:** conjunto de repeticiones de un ejercicio.

**Intensidad de entrenamiento:** número de repeticiones máximas a realizar de cada ejercicio por cada serie.

**Periodización:** variación en la intensidad de entrenamiento y en el descanso a lo largo de las sesiones de un plan de entrenamiento. Puede ser “Lineal” o “No lineal”.

## 5.- Visión general del sistema

### 5.1.- Participantes en el proyecto

En este apartado presentamos brevemente a las diferentes personas y organizaciones involucradas en el proyecto.

#### 5.1.1.- Participantes

<b>Participante</b>	Francisco López Sánchez
<b>Organización</b>	Alumno
<b>Rol</b>	Desarrollador
<b>Es desarrollador</b>	Sí
<b>Es cliente</b>	No
<b>Es usuario</b>	Sí
<b>Comentarios</b>	francisco.lopsan@gmail.com

<b>Participante</b>	Sergio Segura Rueda
<b>Organización</b>	Departamento de Lenguajes y Sistemas Informáticos
<b>Rol</b>	Tutor
<b>Es desarrollador</b>	No
<b>Es cliente</b>	No
<b>Es usuario</b>	Sí
<b>Comentarios</b>	sergiosegura@us.es

<b>Participante</b>	Jesús Domínguez Gutiérrez
<b>Organización</b>	Freelance
<b>Rol</b>	Cliente
<b>Es desarrollador</b>	No
<b>Es cliente</b>	Sí
<b>Es usuario</b>	Sí
<b>Comentarios</b>	Ninguno

#### 5.1.2.- Organizaciones

<b>Organización</b>	Alumno
<b>Dirección</b>	Ponce 16, Morón de la Frontera
<b>Teléfono</b>	955 850 690
<b>Fax</b>	-
<b>Comentarios</b>	Ninguno

<b>Organización</b>	Departamento de Lenguajes y Sistemas Informáticos
<b>Dirección</b>	Avda. Reina Mercedes s/n. 41012 Sevilla
<b>Teléfono</b>	954 557 139
<b>Fax</b>	954 557 139
<b>Comentarios</b>	E-mail: lsi@lsi.us.es Web: www.lsi.us.es

## 5.2.- Objetivos del sistema

<b>OBJ-0001</b>	Gestión de Centros Deportivos
<b>Versión</b>	1.0 ( 29/03/2012 )
<b>Autores</b>	Francisco López Sánchez
<b>Fuentes</b>	Jesús Domínguez Gutiérrez
<b>Descripción</b>	El sistema deberá gestionar toda la información referente a los centros deportivos, así como permitir la posibilidad de añadir, modificar o eliminar monitores deportivos.
<b>Importancia</b>	vital
<b>Urgencia</b>	inmediatamente
<b>Estado</b>	en construcción
<b>Estabilidad</b>	alta

<b>OBJ-0002</b>	Gestión de Monitores Deportivos
<b>Versión</b>	1.0 ( 08/04/2012 )
<b>Autores</b>	Francisco López Sánchez
<b>Fuentes</b>	Jesús Domínguez Gutiérrez
<b>Descripción</b>	El sistema deberá gestionar toda la información referente a los monitores deportivos, permitiendo a éstos añadir nuevos clientes. También tendrán la posibilidad de modificar el perfil de un cliente y crearles planes de entrenamiento a estos clientes.
<b>Importancia</b>	vital
<b>Urgencia</b>	inmediatamente
<b>Estado</b>	en construcción
<b>Estabilidad</b>	baja

<b>OBJ-0003</b>	Gestión de Clientes
<b>Versión</b>	1.0 ( 08/04/2012 )
<b>Autores</b>	Francisco López Sánchez
<b>Fuentes</b>	Jesús Domínguez Gutiérrez
<b>Descripción</b>	El sistema deberá gestionar toda la información referente a los clientes y sus perfiles.
<b>Importancia</b>	vital
<b>Urgencia</b>	inmediatamente
<b>Estado</b>	en construcción
<b>Estabilidad</b>	alta

<b>OBJ-0004</b>	<b>Gestión de Planes de Entrenamiento</b>
<b>Versión</b>	1.0 ( 08/04/2012 )
<b>Autores</b>	Francisco López Sánchez
<b>Fuentes</b>	Jesús Domínguez Gutiérrez
<b>Descripción</b>	<i>El sistema deberá posibilitar a los monitores deportivos la creación tanto automática como manual de un plan de entrenamiento específico para cada cliente según las necesidades observadas en la entrevista previa y en el perfil del cliente.</i>
<b>Importancia</b>	vital
<b>Urgencia</b>	inmediatamente
<b>Estado</b>	en construcción
<b>Estabilidad</b>	alta

## 6.- Elicitación de requisitos

La elicitation de requisitos es el conjunto de procesos previos al diseño e implementación del software mediante los cuales se elabora una lista de requisitos. Por requisito entendemos una característica que debe cumplir el producto que se va a desarrollar.

Los requisitos pueden ser de muy distinta índole, refiriéndose a aspectos diversos como la interfaz, la funcionalidad o la información a almacenar. Los resultados de la elicitation de requisitos quedan recogidos en el documento de elicitation de requisitos, que en nuestro caso constituye este capítulo.

A grandes rasgos, sabemos que el objetivo principal de este proyecto es el desarrollo de una aplicación *Web* que genere planes de entrenamiento específicos para las necesidades de cada usuario. A lo largo del capítulo iremos detallando requisitos más concretos que nos proporcionen una visión más detallada del sistema a desarrollar, y a los cuales nos ceñiremos en los próximos capítulos.

### 6.1.- Requisitos de información

Las siguientes tablas hacen referencia a la información que el sistema debe almacenar para cumplir los objetivos.

<b>IRQ-0001      Información sobre el Centro Deportivo</b>	
<b>Versión</b>	1.0 ( 14/05/2012 )
<b>Autores</b>	Francisco López Sánchez
<b>Fuentes</b>	Jesús Domínguez Gutiérrez
<b>Dependencias</b>	[OBJ-0001] Gestión de Centros Deportivos
<b>Descripción</b>	El sistema deberá almacenar la información correspondiente a <i>los centros deportivos</i> . En concreto:
<b>Datos específicos</b>	Datos propios de centro: nombre Datos de contacto: email, teléfono Datos de dirección: calle, ciudad, código postal, provincia, país
<b>Importancia</b>	vital
<b>Urgencia</b>	inmediatamente
<b>Estado</b>	en construcción
<b>Estabilidad</b>	alta

<b>IRQ-0002      Información sobre el Monitor Deportivo</b>	
<b>Versión</b>	1.0 ( 14/05/2012 )
<b>Autores</b>	Francisco López Sánchez
<b>Fuentes</b>	Jesús Domínguez Gutiérrez
<b>Dependencias</b>	[OBJ-0002] Gestión de Monitores Deportivos
<b>Descripción</b>	El sistema deberá almacenar la información correspondiente a <i>los monitores deportivos</i> . En concreto:
<b>Datos específicos</b>	Datos personales: nombre, apellidos, fecha de nacimiento Datos de contacto: email, teléfono Datos de centro: centro deportivo al que pertenece
<b>Importancia</b>	vital
<b>Urgencia</b>	inmediatamente
<b>Estado</b>	en construcción
<b>Estabilidad</b>	alta

<b>IRQ-0003</b>	<b>Información sobre el Cliente</b>
<b>Versión</b>	1.0 ( 14/05/2012 )
<b>Autores</b>	Francisco López Sánchez
<b>Fuentes</b>	Jesús Domínguez Gutiérrez
<b>Dependencias</b>	[OBJ-0003] Gestión de Clientes
<b>Descripción</b>	El sistema deberá almacenar la información correspondiente a <i>los clientes</i> .
<b>Datos específicos</b>	Datos personales: nombre, apellidos Datos de contacto: email, teléfono Otros datos: fecha ingreso
<b>Importancia</b>	vital
<b>Urgencia</b>	inmediatamente
<b>Estado</b>	en construcción
<b>Estabilidad</b>	alta

<b>IRQ-0004</b>	<b>Información sobre el Perfil</b>
<b>Versión</b>	1.0 ( 14/05/2012 )
<b>Autores</b>	Francisco López Sánchez
<b>Fuentes</b>	Jesús Domínguez Gutiérrez
<b>Dependencias</b>	[OBJ-0003] Gestión de Clientes
<b>Descripción</b>	El sistema deberá almacenar la información correspondiente a <i>el perfil asociado a cada cliente</i> .
<b>Datos específicos</b>	Datos personales: peso, altura, fecha de nacimiento, actividad laboral Datos del nivel: nivel (aclimatación, principiante, intermedio, avanzado) Datos de contraindicaciones: ejercicios y métodos a descartar
<b>Importancia</b>	vital
<b>Urgencia</b>	inmediatamente
<b>Estado</b>	en construcción
<b>Estabilidad</b>	alta

<b>IRQ-0005</b>	<b>Información sobre el Plan de Entrenamiento</b>
<b>Versión</b>	1.0 ( 14/05/2012 )
<b>Autores</b>	Francisco López Sánchez
<b>Fuentes</b>	Jesús Domínguez Gutiérrez
<b>Dependencias</b>	[OBJ-0004] Gestión de Planes de Entrenamiento
<b>Descripción</b>	El sistema deberá almacenar la información correspondiente a <i>el plan de entrenamiento</i> . En concreto:
<b>Datos específicos</b>	Datos de objetivos: objetivo (adaptación, mantenimiento, pérdida de peso, hipertrófia, salud, aumento de la fuerza) Datos de tiempo: número de días disponibles (a la semana), duración total del plan Datos de cualidades físicas: cualidades físicas a mejorar (fuerza, resistencia, flexibilidad) Datos de ejercicios: ejercicios a realizar en cada sesión, nº de series de cada ejercicio, orden (circuito o intercambiar hemisferio), intensidad del esfuerzo (porcentaje en relación a la F máxima, percepción del esfuerzo), densidad del esfuerzo (segundos de recuperación), velocidad de ejecución, periodización (si procede), técnicas de sobrecarga (si procede) Datos de rutina: tipo (global, por hemisferios, por grupo muscular) Datos de material: tipo (máquinas, peso libre, bosu, fitball, gomas, poleas, banco, balón medicinal, anillas) Otros datos: observaciones generales
<b>Importancia</b>	vital
<b>Urgencia</b>	inmediatamente
<b>Estado</b>	en construcción
<b>Estabilidad</b>	alta

## 6.2.- Reglas de negocio y restricciones

<b>CRQ-0001</b>	<b>Unicidad de Perfiles</b>
<b>Versión</b>	1.0 ( 14/05/2012 )
<b>Autores</b>	Francisco López Sánchez
<b>Fuentes</b>	Jesús Domínguez Gutiérrez
<b>Dependencias</b>	[OBJ-0003] Gestión de Clientes
<b>Descripción</b>	La información almacenada por el sistema deberá satisfacer la siguiente restricción: <i>Un cliente tendrá asociado un único perfil.</i>
<b>Importancia</b>	vital
<b>Urgencia</b>	inmediatamente
<b>Estado</b>	en construcción
<b>Estabilidad</b>	alta

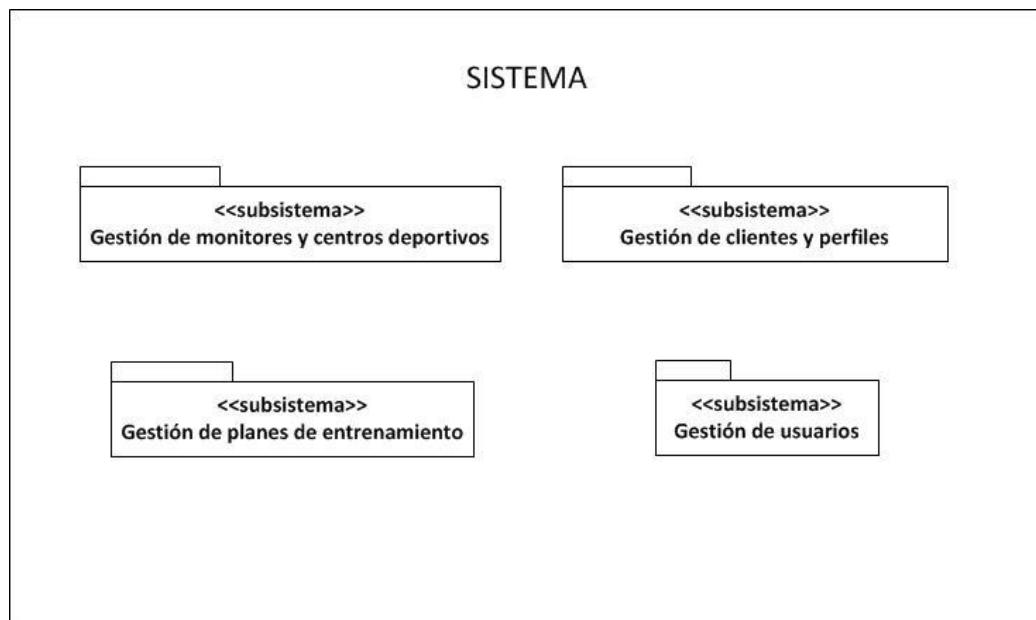
<b>CRQ-0002</b>	<b>Unicidad sobre el nombre de Usuario</b>
<b>Versión</b>	1.0 ( 28/05/2013 )
<b>Autores</b>	Francisco López Sánchez
<b>Fuentes</b>	Francisco López Sánchez
<b>Descripción</b>	La información almacenada por el sistema deberá satisfacer la siguiente restricción: <i>El nombre de usuario de cada cliente será único en el sistema</i>
<b>Importancia</b>	vital
<b>Urgencia</b>	inmediatamente
<b>Estado</b>	en construcción
<b>Estabilidad</b>	alta

<b>CRQ-0003</b>	<b>Restricción en la creación de planes de entrenamiento</b>
<b>Versión</b>	1.0 ( 28/05/2013 )
<b>Autores</b>	Francisco López Sánchez
<b>Fuentes</b>	Francisco López Sánchez
<b>Dependencias</b>	[OBJ-0004] Gestión de Planes de Entrenamiento
<b>Descripción</b>	Los planes de entrenamiento generados deberán respetar las restricciones impuestas por el sistema dependiendo del nivel del usuario y el tipo de entrenamiento deseado.
<b>Importancia</b>	vital
<b>Urgencia</b>	inmediatamente
<b>Estado</b>	en construcción
<b>Estabilidad</b>	alta

## 6.3.- Requisitos funcionales

En este apartado expondremos los requisitos funcionales detectados en forma de casos de uso. Estos casos de uso los clasificaremos en subsistemas que se caracterizan por la información con la que trabajan.

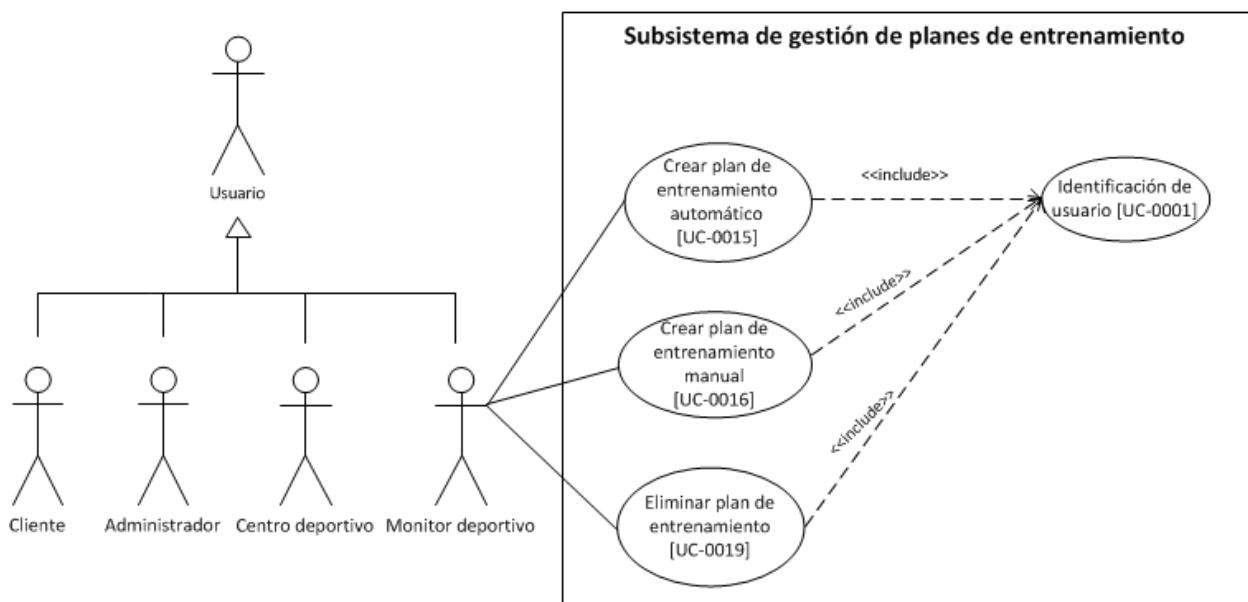
### 6.3.1.- Diagramas de casos de uso



**Figura 15:** Diagramas de subsistemas de la aplicación

#### 6.3.1.1.- Subsistema de gestión de planes de entrenamiento

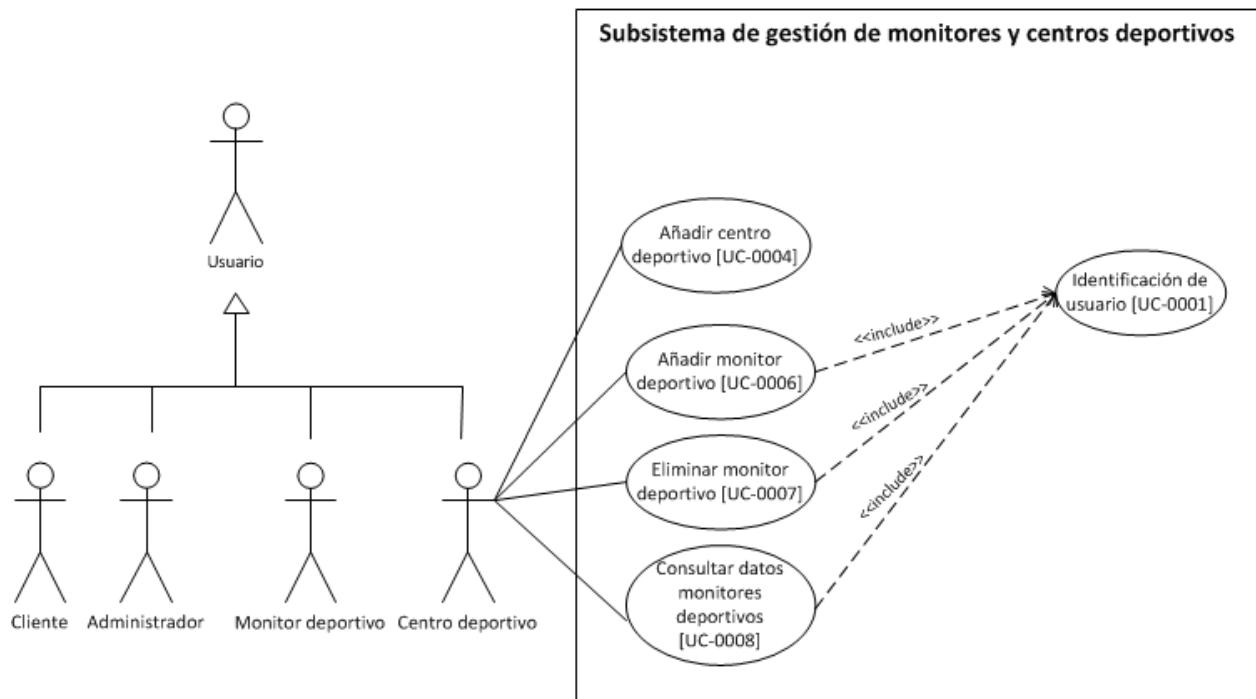
En este subsistema se gestionará todo lo referente a los planes de entrenamiento, posibilitando a los monitores su creación, modificación o eliminación.



**Figura 16:** Subsistema de gestión de planes de entrenamiento

### 6.3.1.2.- Subsistema de gestión de monitores y centros deportivos

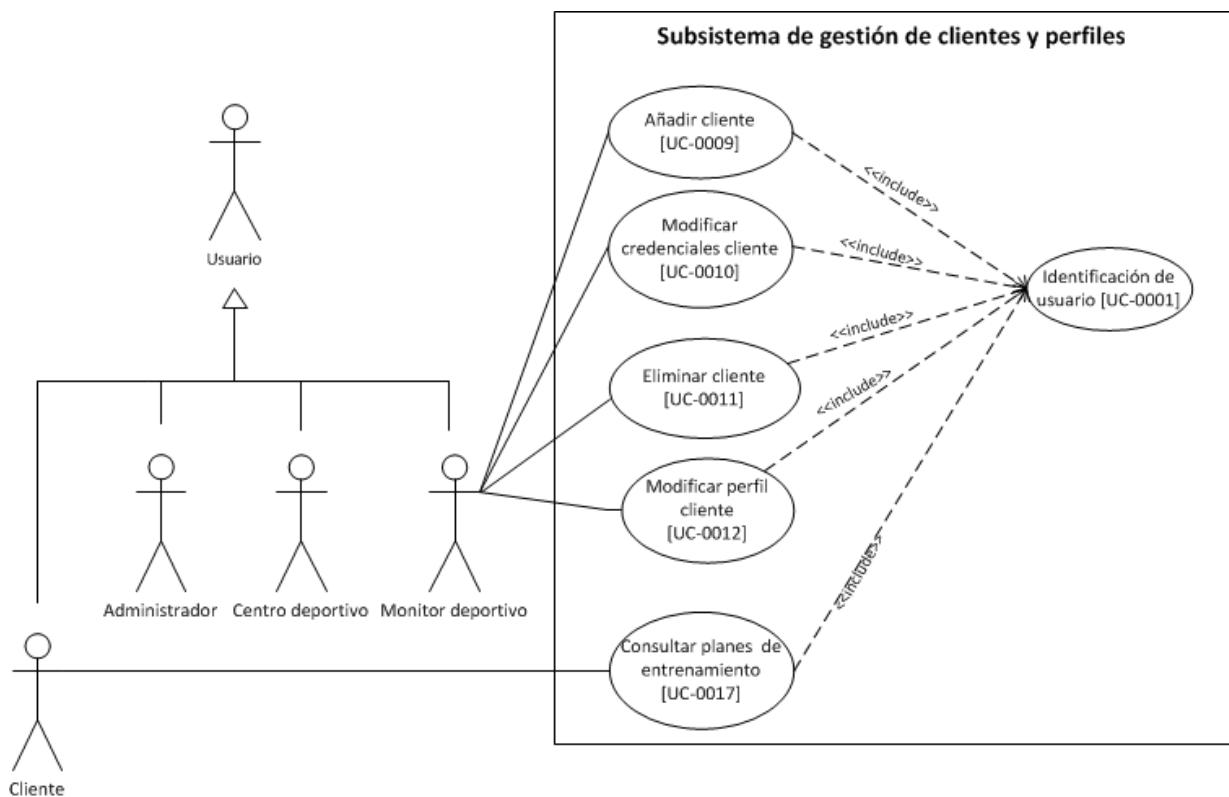
En este subsistema se gestionará todo lo referente a los monitores y centros deportivos. Controlaremos los centros deportivos nuevos que se añaden, sus modificaciones y bajas. También se controlará desde este subsistema losmonitores deportivos que se añadan desde cada centro deportivo, así como sus posibles modificaciones o bajas.



**Figura 17:** Subsistema de gestión de monitores y centros deportivos

### 6.3.1.2.- Subsistema de gestión de clientes y perfiles

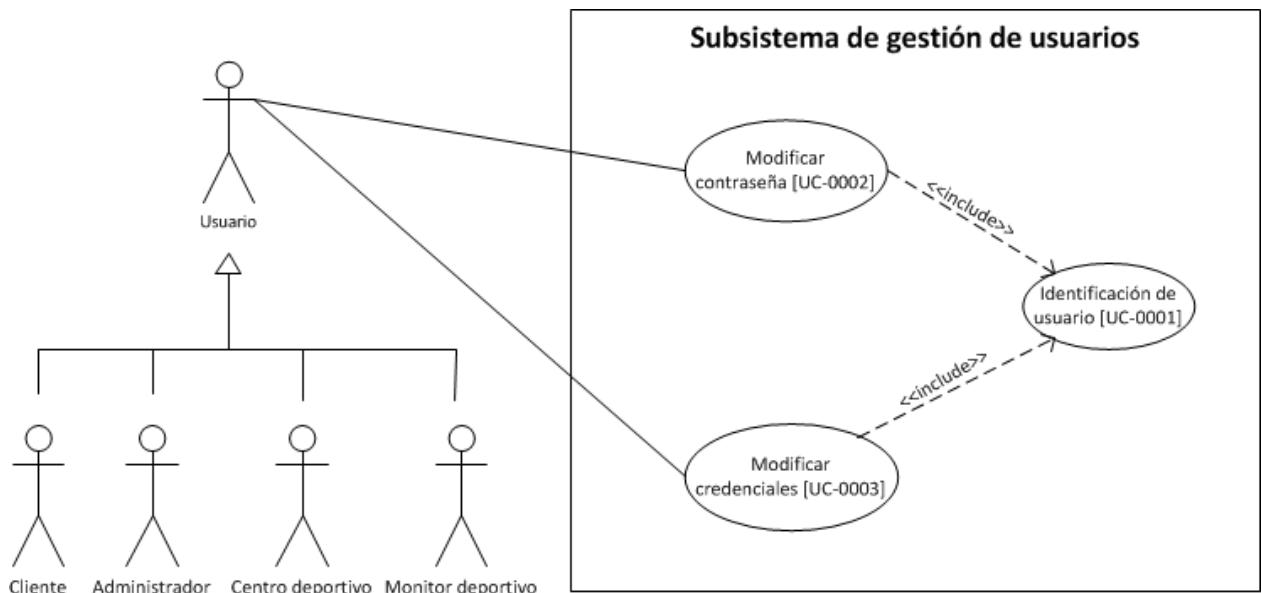
En este subsistema se gestionará todo lo referente a los clientes asociados a cada centro deportivo y sus perfiles. Se controlará que cada usuario tenga un sólo perfil asociado. Este subsistema posibilitará a los monitores deportivos modificar el perfil de cada cliente.



**Figura 18:** Subsistema de gestión de clientes y perfiles

### 6.3.1.4.- Subsistema de gestión de usuarios

Este subsistema gestionará todo lo referente a los usuarios de la aplicación Web.



**Figura 19:** Subsistema de gestión de usuarios

### 6.3.2.- Definición de actores

ACT-0001	Centro deportivo
<b>Versión</b>	1.0 ( 14/05/2012 )
<b>Autores</b>	Francisco López Sánchez
<b>Fuentes</b>	Jesús Domínguez Gutiérrez
<b>Descripción</b>	Este actor representa <i>al usuario que interacciona como centro deportivo</i>
<b>Comentarios</b>	Estos usuarios tienen derechos restringidos, es decir, aquellos derechos propios de los centros deportivos, como por ejemplo añadir monitores deportivos, modificar sus datos o darlos de baja.

ACT-0002	Monitor deportivo
<b>Versión</b>	1.0 ( 14/05/2012 )
<b>Autores</b>	Francisco López Sánchez
<b>Fuentes</b>	Jesús Domínguez Gutiérrez
<b>Descripción</b>	Este actor representa <i>al usuario que interacciona como monitor deportivo</i> .
<b>Comentarios</b>	Estos usuarios tienen derechos restringidos, es decir, aquellos derechos propios de los monitores deportivos.

ACT-0003	Administrador
<b>Versión</b>	1.0 ( 14/05/2012 )
<b>Autores</b>	Francisco López Sánchez
<b>Fuentes</b>	Sergio Segura Rueda
<b>Descripción</b>	Este actor representa <i>a la persona/s encargada/s de gestionar la aplicación web</i> .
<b>Comentarios</b>	Este usuario tiene derechos ilimitados, aunque no participa en la aplicación web.

<b>ACT-0004</b>	<b>Usuario</b>
<b>Versión</b>	1.0 ( 14/05/2012 )
<b>Autores</b>	Francisco López Sánchez
<b>Fuentes</b>	Sergio Segura Rueda
<b>Descripción</b>	Este actor representa a cualquier usuario de la aplicación web.
<b>Comentarios</b>	Ninguno

<b>ACT-0005</b>	<b>Cliente</b>
<b>Versión</b>	1.0 ( 16/05/2012 )
<b>Autores</b>	Francisco López Sánchez
<b>Fuentes</b>	Jesús Domínguez Gutiérrez
<b>Descripción</b>	Este actor representa al cliente al que se le crean planes de entrenamiento
<b>Comentarios</b>	Estos usuarios tienen derechos restringidos, es decir, consultar los planes de entrenamiento que le han sido creados y llevar un seguimiento de éstos.

### 6.3.3.- Casos de uso del sistema

#### 6.3.3.1.- Subsistema de gestión de planes de entrenamiento

<b>UC-0015</b>	<b>Crear Plan de Entrenamiento automático</b>																		
<b>Versión</b>	1.0 ( 15/05/2012 )																		
<b>Autores</b>	Francisco López Sánchez																		
<b>Fuentes</b>	Jesús Domínguez Gutiérrez																		
<b>Dependencias</b>	[IRQ-0005] Información sobre el plan de entrenamiento [OBJ-0004] Gestión de Planes de Entrenamiento																		
<b>Descripción</b>	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando un monitor deportivo desee crear un plan de entrenamiento a un cliente de forma automática																		
<b>Precondición</b>	Ninguna																		
<b>Secuencia normal</b>	<table border="1"> <thead> <tr> <th><b>Paso</b></th> <th><b>Acción</b></th> </tr> </thead> <tbody> <tr> <td>1</td> <td>Se realiza el caso de uso identificación de usuario (UC-0001)</td> </tr> <tr> <td>2</td> <td>El sistema ofrece una lista con los clientes asociados a ese monitor deportivo</td> </tr> <tr> <td>3</td> <td>El actor Monitor deportivo (ACT-0002) selecciona un cliente de la lista al cual se le desea generar el plan deportivo</td> </tr> <tr> <td>4</td> <td>Si el cliente tiene un perfil creado previamente, el sistema realiza el paso 6</td> </tr> <tr> <td>5</td> <td>Si el cliente no tiene creado un perfil previamente, se realiza el caso de uso Modificar perfil del cliente (UC-0012)</td> </tr> <tr> <td>6</td> <td>El sistema solicita al monitor deportivo información tales como el objetivo, la duración, la calidad física a entrenar o el material a usar en el plan de entrenamiento a generar</td> </tr> <tr> <td>7</td> <td>El actor Monitor deportivo (ACT-0002) introduce los datos a partir de la información obtenida del cliente</td> </tr> <tr> <td>8</td> <td>El sistema a partir de la información solicitada y el perfil previamente creado del cliente genera el plan de entrenamiento y se muestra desglosado</td> </tr> </tbody> </table>	<b>Paso</b>	<b>Acción</b>	1	Se realiza el caso de uso identificación de usuario (UC-0001)	2	El sistema ofrece una lista con los clientes asociados a ese monitor deportivo	3	El actor Monitor deportivo (ACT-0002) selecciona un cliente de la lista al cual se le desea generar el plan deportivo	4	Si el cliente tiene un perfil creado previamente, el sistema realiza el paso 6	5	Si el cliente no tiene creado un perfil previamente, se realiza el caso de uso Modificar perfil del cliente (UC-0012)	6	El sistema solicita al monitor deportivo información tales como el objetivo, la duración, la calidad física a entrenar o el material a usar en el plan de entrenamiento a generar	7	El actor Monitor deportivo (ACT-0002) introduce los datos a partir de la información obtenida del cliente	8	El sistema a partir de la información solicitada y el perfil previamente creado del cliente genera el plan de entrenamiento y se muestra desglosado
<b>Paso</b>	<b>Acción</b>																		
1	Se realiza el caso de uso identificación de usuario (UC-0001)																		
2	El sistema ofrece una lista con los clientes asociados a ese monitor deportivo																		
3	El actor Monitor deportivo (ACT-0002) selecciona un cliente de la lista al cual se le desea generar el plan deportivo																		
4	Si el cliente tiene un perfil creado previamente, el sistema realiza el paso 6																		
5	Si el cliente no tiene creado un perfil previamente, se realiza el caso de uso Modificar perfil del cliente (UC-0012)																		
6	El sistema solicita al monitor deportivo información tales como el objetivo, la duración, la calidad física a entrenar o el material a usar en el plan de entrenamiento a generar																		
7	El actor Monitor deportivo (ACT-0002) introduce los datos a partir de la información obtenida del cliente																		
8	El sistema a partir de la información solicitada y el perfil previamente creado del cliente genera el plan de entrenamiento y se muestra desglosado																		
<b>Postcondición</b>	El plan de entrenamiento queda generado y asociado al cliente																		
<b>Importancia</b>	vital																		
<b>Urgencia</b>	inmediatamente																		
<b>Estado</b>	en construcción																		
<b>Estabilidad</b>	alta																		

<b>UC-0016</b>	<b>Crear Plan de Entrenamiento manual</b>																						
<b>Versión</b>	1.0 ( 15/05/2012 )																						
<b>Autores</b>	Francisco López Sánchez																						
<b>Fuentes</b>	Jesús Domínguez Gutiérrez																						
<b>Dependencias</b>	[IRQ-0005] Información sobre el plan de entrenamiento [OBJ-0004] Gestión de Planes de Entrenamiento																						
<b>Descripción</b>	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando <i>un monitor deportivo desee crear manualmente un plan deportivo a un cliente</i>																						
<b>Precondición</b>	Ninguna																						
<b>Secuencia normal</b>	<table> <thead> <tr> <th>Paso</th> <th>Acción</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>Se realiza el caso de uso identificación de usuario (UC-0001)</td> </tr> <tr> <td>2</td> <td>El sistema <i>ofrece una lista con los clientes asociados a ese monitor deportivo</i></td> </tr> <tr> <td>3</td> <td>El actor Monitor deportivo (ACT-0002) <i>selecciona un cliente de la lista al cual se le desea generar el plan deportivo</i></td> </tr> <tr> <td>4</td> <td><i>Si el cliente tiene un perfil creado previamente, el sistema realiza el paso 6</i></td> </tr> <tr> <td>5</td> <td><i>Si el cliente no tiene creado un perfil previamente, se realiza el caso de uso Modificar perfil del cliente (UC-0012)</i></td> </tr> <tr> <td>6</td> <td><i>El sistema solicita al monitor deportivo información tales como el objetivo, la duración, la calidad física a entrenar o el material a usar en el plan de entrenamiento a generar</i></td> </tr> <tr> <td>7</td> <td><i>El actor Monitor deportivo (ACT-0002) introduce los datos a partir de la información obtenida del cliente</i></td> </tr> <tr> <td>8</td> <td><i>El sistema a partir de la información introducida por el monitor deportivo, muestra una lista con los ejercicios que pueden ser seleccionados para el plan de entrenamiento</i></td> </tr> <tr> <td>9</td> <td><i>El actor Monitor deportivo (ACT-0002) selecciona los ejercicios que desea incluir en el plan de entrenamiento</i></td> </tr> <tr> <td>10</td> <td><i>El sistema genera el plan de entrenamiento y lo muestra desglosado</i></td> </tr> </tbody> </table>	Paso	Acción	1	Se realiza el caso de uso identificación de usuario (UC-0001)	2	El sistema <i>ofrece una lista con los clientes asociados a ese monitor deportivo</i>	3	El actor Monitor deportivo (ACT-0002) <i>selecciona un cliente de la lista al cual se le desea generar el plan deportivo</i>	4	<i>Si el cliente tiene un perfil creado previamente, el sistema realiza el paso 6</i>	5	<i>Si el cliente no tiene creado un perfil previamente, se realiza el caso de uso Modificar perfil del cliente (UC-0012)</i>	6	<i>El sistema solicita al monitor deportivo información tales como el objetivo, la duración, la calidad física a entrenar o el material a usar en el plan de entrenamiento a generar</i>	7	<i>El actor Monitor deportivo (ACT-0002) introduce los datos a partir de la información obtenida del cliente</i>	8	<i>El sistema a partir de la información introducida por el monitor deportivo, muestra una lista con los ejercicios que pueden ser seleccionados para el plan de entrenamiento</i>	9	<i>El actor Monitor deportivo (ACT-0002) selecciona los ejercicios que desea incluir en el plan de entrenamiento</i>	10	<i>El sistema genera el plan de entrenamiento y lo muestra desglosado</i>
Paso	Acción																						
1	Se realiza el caso de uso identificación de usuario (UC-0001)																						
2	El sistema <i>ofrece una lista con los clientes asociados a ese monitor deportivo</i>																						
3	El actor Monitor deportivo (ACT-0002) <i>selecciona un cliente de la lista al cual se le desea generar el plan deportivo</i>																						
4	<i>Si el cliente tiene un perfil creado previamente, el sistema realiza el paso 6</i>																						
5	<i>Si el cliente no tiene creado un perfil previamente, se realiza el caso de uso Modificar perfil del cliente (UC-0012)</i>																						
6	<i>El sistema solicita al monitor deportivo información tales como el objetivo, la duración, la calidad física a entrenar o el material a usar en el plan de entrenamiento a generar</i>																						
7	<i>El actor Monitor deportivo (ACT-0002) introduce los datos a partir de la información obtenida del cliente</i>																						
8	<i>El sistema a partir de la información introducida por el monitor deportivo, muestra una lista con los ejercicios que pueden ser seleccionados para el plan de entrenamiento</i>																						
9	<i>El actor Monitor deportivo (ACT-0002) selecciona los ejercicios que desea incluir en el plan de entrenamiento</i>																						
10	<i>El sistema genera el plan de entrenamiento y lo muestra desglosado</i>																						
<b>Postcondición</b>	El plan de entrenamiento queda generado y asociado al cliente																						
<b>Importancia</b>	vital																						
<b>Urgencia</b>	inmediatamente																						
<b>Estado</b>	en construcción																						
<b>Estabilidad</b>	alta																						

<b>UC-0018</b>	<b>Consultar plan de entrenamiento en formato PDF</b>	
<b>Versión</b>	1.0 ( 15/05/2012 )	
<b>Autores</b>	Francisco López Sánchez	
<b>Fuentes</b>	Jesús Domínguez Gutiérrez	
<b>Dependencias</b>	[IRQ-0005] Información sobre el plan de entrenamiento [OBJ-0004] Gestión de Planes de Entrenamiento	
<b>Descripción</b>	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando <i>un monitor deportivo desee consultar el plan de entrenamiento en formato PDF</i>	
<b>Precondición</b>	Ninguna	
<b>Secuencia normal</b>	<b>Paso</b>	<b>Acción</b>
	1	El actor Monitor deportivo (ACT-0002) <i>selecciona un cliente del cual desea consultar un plan de entrenamiento en formato PDF</i>
	2	El sistema <i>muestra los planes deportivos que ese cliente tiene generados</i>
	3	Si <i>el cliente no tiene creado ningún plan deportivo previamente</i> , se realiza el caso de uso Crear un plan deportivo Automáticamente (UC-0015)
	4	El actor Monitor deportivo (ACT-0002) <i>indica el plan deportivo que desea consultar en formato PDF</i>
	5	El sistema <i>muestra el plan deportivo seleccionado en formato PDF</i>
<b>Postcondición</b>	Ninguna	
<b>Importancia</b>	estaría bien	
<b>Urgencia</b>	puede esperar	
<b>Estado</b>	en construcción	
<b>Estabilidad</b>	alta	

<b>UC-0019</b>	<b>Eliminar Plan de Entrenamiento</b>	
<b>Versión</b>	1.0 ( 16/05/2012 )	
<b>Autores</b>	Francisco López Sánchez	
<b>Fuentes</b>	Jesús Domínguez Gutiérrez	
<b>Dependencias</b>	[IRQ-0005] Información sobre el plan de entrenamiento [OBJ-0004] Gestión de Planes de Entrenamiento	
<b>Descripción</b>	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando <i>un monitor deportivo desee eliminar un plan de entrenamiento asociado a un cliente</i>	
<b>Precondición</b>	el plan de entrenamiento a eliminar debe estar asignado a algún cliente	
<b>Secuencia normal</b>	<b>Paso</b>	<b>Acción</b>
	1	Se realiza el caso de uso identificación de usuario (UC-0001)
	2	El sistema <i>muestra al monitor deportivo una lista con los planes de entrenamiento asociados a ese cliente</i>
	3	El actor Monitor deportivo (ACT-0002) <i>selecciona el plan de entrenamiento que desea eliminar</i>
	4	El sistema <i>elimina el/los monitor/es seleccionado/s y muestra un mensaje de confirmación</i>
<b>Postcondición</b>	el plan de entrenamiento queda eliminado del sistema	
<b>Importancia</b>	vital	
<b>Urgencia</b>	inmediatamente	
<b>Estado</b>	en construcción	
<b>Estabilidad</b>	alta	

### 6.3.3.2.- Subsistema de gestión de monitores y centros deportivos

<b>UC-0004</b>	<b>Añadir Centro Deportivo</b>	
<b>Versión</b>	1.0 ( 15/05/2012 )	
<b>Autores</b>	Francisco López Sánchez	
<b>Fuentes</b>	Jesús Domínguez Gutiérrez	
<b>Dependencias</b>	[OBJ-0001] Gestión de Centros Deportivos [IRQ-0001] Información sobre el centro deportivo	
<b>Descripción</b>	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando <i>un usuario desee dar de alta un nuevo centro deportivo</i>	
<b>Precondición</b>	Ninguna	
<b>Secuencia normal</b>	<b>Paso</b>	<b>Acción</b>
	1	El actor Usuario (ACT-0004) <i>solicita dar de alta un nuevo centro deportivo</i>
	2	El sistema <i>solicita los datos necesarios para el registro de un nuevo centro deportivo</i>
	3	El actor <u>Usuario</u> (ACT-0004) <i>introduce los datos correspondientes para dar de alta el centro deportivo en el sistema</i>
	4	El sistema <i>valida los datos introducidos por el usuario y se muestra un mensaje de confirmación informando del éxito del registro</i>
<b>Postcondición</b>	El centro deportivo queda registrado en el sistema	
<b>Excepciones</b>	<b>Paso</b>	<b>Acción</b>
	4	Si existe un centro deportivo añadido previamente con el mismo nombre y la misma ciudad, el sistema solicita al usuario que se introduzca un nombre distinto, a continuación este caso de uso continúa
	4	Si no se tiene acceso a la base de datos, el sistema informa del error, y ofrece la opción de reintentarlo, a continuación este caso de uso continúa
<b>Importancia</b>	vital	
<b>Urgencia</b>	inmediatamente	
<b>Estado</b>	en construcción	
<b>Estabilidad</b>	alta	

<b>UC-0006</b>	<b>Añadir Monitor Deportivo</b>	
<b>Versión</b>	1.0 ( 15/05/2012 )	
<b>Autores</b>	Francisco López Sánchez	
<b>Fuentes</b>	Sergio Segura Rueda	
<b>Dependencias</b>	[OBJ-0002] Gestión de Monitores Deportivos	
<b>Descripción</b>	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando <i>un centro deportivo solicite incluir un nuevo monitor deportivo</i>	
<b>Precondición</b>	Ninguna	
<b>Secuencia normal</b>	<b>Paso</b>	<b>Acción</b>
	1	Se realiza el caso de uso identificación de usuario (UC-0001)
	2	El sistema <i>solicita al centro deportivo los datos del nuevo monitor deportivo</i>
	3	El actor <u>Centro deportivo</u> (ACT-0001) <i>introduce los datos del nuevo monitor deportivo</i>
	4	El sistema <i>muestra de que se ha realizado satisfactoriamente la inserción del nuevo monitor deportivo</i>
<b>Postcondición</b>	El monitor deportivo es incluido en el sistema	
<b>Excepciones</b>	<b>Paso</b>	<b>Acción</b>
	3	Si existe un monitor deportivo añadido previamente con el mismo nombre, el sistema solicita al usuario que se introduzca un nombre distinto, a continuación este caso de uso continúa
<b>Importancia</b>	vital	
<b>Urgencia</b>	inmediatamente	
<b>Estado</b>	en construcción	
<b>Estabilidad</b>	alta	

<b>UC-0007</b>	<b>Eliminar Monitor Deportivo</b>	
<b>Versión</b>	1.0 ( 15/05/2012 )	
<b>Autores</b>	Francisco López Sánchez	
<b>Fuentes</b>	Sergio Segura Rueda	
<b>Dependencias</b>	[OBJ-0002] Gestión de Monitores Deportivos [IRQ-0002] Información sobre el monitor deportivo	
<b>Descripción</b>	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando <i>un centro deportivo solicite dar de baja a un monitor deportivo</i>	
<b>Precondición</b>	El monitor debe pertenecer a ese centro deportivo	
<b>Secuencia normal</b>	<b>Paso</b>	<b>Acción</b>
	1	Se realiza el caso de uso identificación de usuario (UC-0001)
	2	El sistema <i>muestra al centro deportivo una lista con los monitores deportivos pertenecientes a ese mismo centro deportivo</i>
	3	El actor <i>Centro deportivo (ACT-0001) selecciona el monitor que desea eliminar</i>
	4	El sistema <i>elimina el monitor seleccionado y muestra un mensaje de confirmación</i>
<b>Postcondición</b>	El monitor queda eliminado del sistema	
<b>Importancia</b>	vital	
<b>Urgencia</b>	inmediatamente	
<b>Estado</b>	en construcción	
<b>Estabilidad</b>	alta	

<b>UC-0008</b>	<b>Modificar información de Monitor Deportivo</b>	
<b>Versión</b>	1.0 ( 15/05/2012 )	
<b>Autores</b>	Francisco López Sánchez	
<b>Fuentes</b>	Sergio Segura Rueda	
<b>Dependencias</b>	[OBJ-0002] Gestión de Monitores Deportivos [IRQ-0002] Información sobre el monitor deportivo	
<b>Descripción</b>	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando <i>un centro deportivo solicite al sistema modificar la información personal de uno de sus monitores deportivos</i>	
<b>Precondición</b>	El monitor deportivo debe pertenecer al centro deportivo que realiza la acción	
<b>Secuencia normal</b>	<b>Paso</b>	<b>Acción</b>
	1	Se realiza el caso de uso identificación de usuario (UC-0001)
	2	El sistema <i>solicita al centro deportivo qué datos desea modificar del monitor deportivo seleccionado</i>
	3	El actor <i>Centro deportivo (ACT-0001) introduce los datos a modificar referente a ese monitor deportivo</i>
	4	El sistema <i>modifica la información introducida y muestra un mensaje confirmando la acción realizada</i>
<b>Postcondición</b>	Ninguna	
<b>Importancia</b>	importante	
<b>Urgencia</b>	puede esperar	
<b>Estado</b>	en construcción	
<b>Estabilidad</b>	alta	

### 6.3.3.3.- Subsistema de gestión de clientes y perfiles

<b>UC-0009</b>	<b>Añadir Cliente</b>	
<b>Versión</b>	1.0 ( 15/05/2012 )	
<b>Autores</b>	Francisco López Sánchez	
<b>Fuentes</b>	Jesús Domínguez Gutiérrez	
<b>Dependencias</b>	[IRQ-0003] Información sobre el cliente [OBJ-0003] Gestión de Clientes	
<b>Descripción</b>	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando un <i>monitor deportivo</i> desee dar de alta un nuevo <i>cliente</i> en el sistema	
<b>Precondición</b>	El monitor deportivo debe pertenecer al centro deportivo en el que se desea incluir el nuevo cliente	
<b>Secuencia normal</b>	<b>Paso</b>	<b>Acción</b>
	1	Se realiza el caso de uso identificación de usuario (UC-0001)
	2	El sistema solicita al <i>monitor deportivo</i> la información referente al nuevo <i>cliente</i> que se desea incluir
	3	El actor <i>Monitor deportivo</i> (ACT-0002) introduce la información solicitada
	4	El sistema valida la información introducida y muestra un mensaje de confirmación
<b>Postcondición</b>	El nuevo cliente queda registrado en el sistema	
<b>Importancia</b>	vital	
<b>Urgencia</b>	inmediatamente	
<b>Estado</b>	en construcción	
<b>Estabilidad</b>	alta	

<b>UC-0010</b>	<b>Modificar credenciales Cliente</b>	
<b>Versión</b>	1.0 ( 15/05/2012 )	
<b>Autores</b>	Francisco López Sánchez	
<b>Fuentes</b>	Jesús Domínguez Gutiérrez	
<b>Dependencias</b>	[IRQ-0003] Información sobre el cliente [OBJ-0003] Gestión de Clientes	
<b>Descripción</b>	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando un <i>monitor deportivo</i> desee modificar la información referente a un <i>cliente</i>	
<b>Precondición</b>	El monitor deportivo y el cliente cuyas credenciales se desean modificar deben pertenecer al mismo centro deportivo	
<b>Secuencia normal</b>	<b>Paso</b>	<b>Acción</b>
	1	Se realiza el caso de uso identificación de usuario (UC-0001)
	2	El sistema solicita al <i>monitor deportivo</i> las credenciales del <i>usuario</i> que quiere modificar
	3	El actor <i>Monitor deportivo</i> (ACT-0002) introduce los credenciales que desea modificar del <i>usuario</i>
	4	El sistema modifica los credenciales especificados por el <i>monitor deportivo</i> y muestra un mensaje de confirmación
<b>Postcondición</b>	Ninguna	
<b>Importancia</b>	vital	
<b>Urgencia</b>	hay presión	
<b>Estado</b>	en construcción	
<b>Estabilidad</b>	Alta	

<b>UC-0011</b>	<b>Eliminar Cliente</b>	
<b>Versión</b>	1.0 ( 15/05/2012 )	
<b>Autores</b>	Francisco López Sánchez	
<b>Fuentes</b>	Jesús Domínguez Gutiérrez	
<b>Dependencias</b>	[IRQ-0003] Información sobre el cliente [OBJ-0003] Gestión de Clientes	
<b>Descripción</b>	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando un <i>monitor deportivo</i> desee dar de baja un <i>cliente</i> del sistema	
<b>Precondición</b>	Tanto el monitor deportivo como el cliente que se desea dar de baja deberán pertenecer ambos al mismo centro deportivo	
<b>Secuencia normal</b>	<b>Paso</b>	<b>Acción</b>
	1	Se realiza el caso de uso identificación de usuario (UC-0001)
	2	El sistema muestra al <i>monitor deportivo</i> una lista con los <i>clientes</i> pertenecientes a ese <i>centro deportivo</i>
	3	El actor <i>Monitor deportivo</i> (ACT-0002) selecciona el <i>cliente</i> que desea eliminar
	4	El sistema elimina el <i>cliente</i> seleccionado y muestra un mensaje de confirmación
<b>Postcondición</b>	El cliente queda eliminado del sistema	
<b>Importancia</b>	vital	
<b>Urgencia</b>	hay presión	
<b>Estado</b>	en construcción	
<b>Estabilidad</b>	alta	

<b>UC-0012</b>	<b>Modificar Perfil del Cliente</b>	
<b>Versión</b>	1.0 ( 15/05/2012 )	
<b>Autores</b>	Francisco López Sánchez	
<b>Fuentes</b>	Jesús Domínguez Gutiérrez	
<b>Dependencias</b>	[IRQ-0004] Información sobre el perfil [OBJ-0006] Gestión de Perfil	
<b>Descripción</b>	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando un <i>monitor deportivo</i> desee modificar la información del perfil de un cliente o durante la realización de los siguientes casos de uso: [UC-0015] Crear plan de entrenamiento automático, [UC-0016] Crear plan de entrenamiento manual	
<b>Precondición</b>	El cliente y el monitor deportivo deben pertenecer al mismo centro deportivo	
<b>Secuencia normal</b>	<b>Paso</b>	<b>Acción</b>
	1	Se realiza el caso de uso identificación de usuario (UC-0001)
	2	El sistema solicita al <i>monitor deportivo</i> la nueva información del perfil que desea modificar.
	3	El actor <i>Monitor deportivo</i> (ACT-0002) introduce la información del perfil referente al cliente que desea modificar
	4	El sistema modifica el perfil asociado a ese cliente y muestra un mensaje de confirmación
<b>Postcondición</b>	Ninguna	
<b>Importancia</b>	vital	
<b>Urgencia</b>	inmediatamente	
<b>Estado</b>	en construcción	
<b>Estabilidad</b>	alta	

<b>UC-0017</b>	<b>Consultar Planes de Entrenamiento</b>	
<b>Versión</b>	1.0 ( 16/05/2012 )	
<b>Autores</b>	Francisco López Sánchez	
<b>Fuentes</b>	Jesús Domínguez Gutiérrez	
<b>Dependencias</b>	[IRQ-0003] Información sobre el cliente [IRQ-0005] Información sobre el plan de entrenamiento [OBJ-0003] Gestión de Clientes [OBJ-0004] Gestión de Planes de Entrenamiento	
<b>Descripción</b>	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando <i>un cliente solicite consultar los planes de entrenamiento que le han sido generados en el sistema</i>	
<b>Precondición</b>	Ninguna	
<b>Secuencia normal</b>	<b>Paso</b>	<b>Acción</b>
	1	Se realiza el caso de uso identificación de usuario (UC-0001)
	2	El sistema <i>solicita al cliente que tipo de plan de entrenamiento desea consultar</i>
	3	El actor Usuario (ACT-0004) <i>selecciona si desea consultar los planes de entrenamiento ya realizados o los que están activos actualmente</i>
	4	<i>Si el cliente selecciona consultar planes de entrenamiento ya finalizados, el sistema muestra dichos planes ordenados cronológicamente, a continuación se realiza el paso 6</i>
	5	<i>Si el cliente selecciona consultar los planes de entrenamiento actuales, el sistema muestra dichos planes ordenados cronológicamente, a continuación, se realiza el paso 6</i>
	6	El actor Usuario (ACT-0004) <i>selecciona el plan de entrenamiento a consultar</i>
	7	El sistema <i>muestra al cliente el plan de entrenamiento seleccionado separando cada periodo, semana y días del entrenamiento.</i>
<b>Postcondición</b>	Ninguna	
<b>Excepciones</b>	<b>Paso</b>	<b>Acción</b>
	2	<i>Si el cliente no tiene planes de entrenamiento previamente generados, el sistema informa de la situación, a continuación este caso de uso queda sin efecto</i>
	2	<i>Si el cliente no tiene planes de entrenamiento previamente generados, pero sí tiene planes de entrenamiento activos, , el sistema muestra únicamente los planes de entrenamientos activos en este momento, a continuación este caso de uso continúa</i>
<b>Importancia</b>	vital	
<b>Urgencia</b>	hay presión	
<b>Estado</b>	en construcción	
<b>Estabilidad</b>	alta	

### 6.3.3.4.- Subsistema de gestión de usuarios

<b>UC-0001</b>	<b>Identificación de Usuario</b>	
<b>Versión</b>	1.0 ( 15/05/2012 )	
<b>Autores</b>	Francisco López Sánchez	
<b>Fuentes</b>	Sergio Segura Rueda	
<b>Dependencias</b>	[OBJ-0001] Gestión de Centros Deportivos [OBJ-0002] Gestión de Monitores Deportivos [IRQ-0001] Información sobre el centro deportivo [IRQ-0002] Información sobre el monitor deportivo	
<b>Descripción</b>	El sistema deberá comportarse tal como se describe en el siguiente caso de uso abstracto durante la realización de los siguientes casos de uso: [UC-0002] Modificar contraseña, [UC-0003] Modificar credenciales, [UC-0006] Añadir monitor deportivo, [UC-0007] Eliminar monitor deportivo, [UC-0008] Modificar información de monitor deportivo, [UC-0009] Añadir cliente, [UC-0010] Modificar credenciales cliente, [UC-0011] Eliminar cliente, [UC-0012] Modificar perfil del cliente, [UC-0015] Crear plan de entrenamiento automático, [UC-0016] Crear plan de entrenamiento manual, [UC-0017] Consultar planes de entrenamiento, [UC-0019] Eliminar plan de entrenamiento	
<b>Precondición</b>	Ninguna	
<b>Secuencia normal</b>	<b>Paso</b>	<b>Acción</b>
	1	El sistema <i>solicita los datos de identificación al usuario</i>
	2	El actor <i>Usuario (ACT-0004) introduce los datos de identificación</i>
	3	El sistema <i>valida los datos introducidos por el usuario y muestra por pantalla un mensaje de confirmación</i>
<b>Postcondición</b>	El usuario es reconocido por la aplicación y se le aplican sus correspondientes permisos.	
<b>Excepciones</b>	<b>Paso</b>	<b>Acción</b>
	3	<i>Si uno de los datos aportados por el usuario no es correcto, el sistema muestra un mensaje informando del error y permite modificar el dato de acceso introducido erróneamente, a continuación este caso de uso continúa</i>
<b>Importancia</b>	vital	
<b>Urgencia</b>	inmediatamente	
<b>Estado</b>	en construcción	
<b>Estabilidad</b>	alta	
<b>Comentarios</b>	Ninguno	

<b>UC-0002</b>	<b>Modificar contraseña</b>	
<b>Versión</b>	1.0 ( 15/05/2012 )	
<b>Autores</b>	Francisco López Sánchez	
<b>Fuentes</b>	Sergio Segura Rueda	
<b>Dependencias</b>	[OBJ-0001] Gestión de Centros Deportivos [OBJ-0002] Gestión de Monitores Deportivos [OBJ-0003] Gestión de Clientes	
<b>Descripción</b>	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando <i>un usuario solicite cambiar su contraseña</i> .	
<b>Precondición</b>	Ninguna	
<b>Secuencia normal</b>	<b>Paso</b>	<b>Acción</b>
	1	Se realiza el caso de uso identificación de usuario (UC-0001)
	2	El sistema <i>solicita al usuario la nueva contraseña. Ésta se pide por duplicado para evitar confusiones</i>
	3	El actor <i>Usuario (ACT-0004) introduce la nueva contraseña</i>
	4	<i>El sistema modifica la contraseña del usuario y muestra un mensaje confirmando el cambio</i>
<b>Postcondición</b>	El sistema asigna la nueva contraseña al usuario	

Excepciones	Paso	Acción
	4	Si el usuario no introduce ambas contraseñas iguales, o no presentan la longitud mínima, el sistema pide la inserción de una nueva contraseña, a continuación este caso de uso continúa
Importancia	vital	
Urgencia	inmediatamente	
Estado	en construcción	
Estabilidad	alta	

UC-0003	Modificar credenciales	
Versión	1.0 ( 15/05/2012 )	
Autores	Francisco López Sánchez	
Fuentes	Sergio Segura Rueda	
Dependencias	[OBJ-0001] Gestión de Centros Deportivos [OBJ-0002] Gestión de Monitores Deportivos [OBJ-0003] Gestión de Clientes	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando <i>el usuario solicite modificar sus credenciales</i>	
Precondición	Ninguna	
Secuencia normal	Paso	Acción
	1	Se realiza el caso de uso identificación de usuario (UC-0001)
	2	El sistema <i>solicita al usuario las credenciales que quiere modificar</i>
	3	El actor <u>Usuario</u> (ACT-0004) <i>introduce los credenciales que desea modificar</i>
	4	El sistema <i>modifica los credenciales especificados por el usuario y muestra un mensaje de confirmación</i>
Postcondición	Ninguna	
Importancia	vital	
Urgencia	inmediatamente	
Estado	en construcción	
Estabilidad	alta	

## 6.4.- Requisitos no funcionales

NFR-0001	Seguridad	
Versión	1.0 ( 30/05/2013 )	
Autores	Francisco López Sánchez	
Fuentes	Francisco López Sánchez	
Dependencias	[OBJ-0003] Gestión de Clientes	
Descripción	El sistema deberá proveer de mecanismos de seguridad para un acceso seguro y proteger la información de los usuarios.	
Importancia	vital	
Urgencia	inmediatamente	
Estado	en construcción	
Estabilidad	alta	
Comentarios	Haremos especial hincapié a las principales vulnerabilidades de las aplicaciones Web. Parámetros sin validar, gestión modificada de sesión y de cuentas, problemas en la gestión de errores, configuración inadecuada de un servidor Web.	

NFR-0002	Portabilidad
<b>Versión</b>	1.0 ( 30/05/2013 )
<b>Autores</b>	Francisco López Sánchez
<b>Fuentes</b>	Francisco López Sánchez
<b>Dependencias</b>	Ninguno
<b>Descripción</b>	El sistema deberá poder ser accedido por los navegadores más actuales y más utilizados como son Google Chrome, Mozilla FireFox, Opera e Internet Explorer
<b>Importancia</b>	vital
<b>Urgencia</b>	inmediatamente
<b>Estado</b>	en construcción
<b>Estabilidad</b>	alta

NFR-0003	Escalabilidad
<b>Versión</b>	1.0 ( 30/05/2013 )
<b>Autores</b>	Francisco López Sánchez
<b>Fuentes</b>	Francisco López Sánchez
<b>Dependencias</b>	Ninguno
<b>Descripción</b>	El sistema deberá proveer de mecanismos para no ver comprometido el rendimiento de la aplicación con el acceso de, al menos, 100 usuarios registrados simultáneos a la aplicación Web
<b>Importancia</b>	vital
<b>Urgencia</b>	inmediatamente
<b>Estado</b>	en construcción
<b>Estabilidad</b>	alta

## 6.5.- Matriz de rastreabilidad requisitos-objetivos

TRM-0001	OBJ-0001	OBJ-0002	OBJ-0003	OBJ-0004
IRQ-0001	↑	-	-	-
IRQ-0002	-	↑	-	-
IRQ-0003	-	-	↑	-
IRQ-0004	-	-	↑	-
IRQ-0005	-	-	-	↑

**Matriz de rastreabilidad:** Requisitos de información - Objetivos

## 6.6.- Matriz de rastreabilidad casos de uso – objetivos

TRM-0002	OBJ-0001	OBJ-0002	OBJ-0003	OBJ-0004
UC-0001	↑	↑	-	-
UC-0002	↑	↑	↑	-
UC-0003	↑	↑	↑	-
UC-0004	↑	-	-	-
UC-0006	-	↑	-	-
UC-0007	-	↑	-	-
UC-0008	-	↑	-	-
UC-0009	-	-	↑	-
UC-0010	-	-	↑	-
UC-0011	-	-	↑	-
UC-0012	-	-	↑	-
UC-0015	-	-	-	↑
UC-0016	-	-	-	↑
UC-0017	-	-	↑	↑
UC-0018	-	-	-	↑
UC-0019	-	-	-	↑

**Matriz de rastreabilidad:** casos de uso – objetivos

## 7.- Análisis de requisitos

En este capítulo vamos a acometer una de las fases principales del proceso de desarrollo de un proyecto software, el análisis de requisitos. Esta fase consiste en la construcción de un modelo conceptual partiendo del documento delicitación de requisitos. Dicho modelo debe entenderse como una primera aproximación que será refinada en sucesivas etapas del desarrollo.

### 7.1.- Modelo estático del sistema

En este apartado describiremos el modelo estático del sistema, cuya finalidad es definir un conjunto de tipos abstractos de datos y asociaciones entre dichos tipos que satisfaga los requisitos de información detectados en la fase de elicitación. Hemos dividido el conjunto de tipos de datos y asociaciones en cuatro subsistemas cuyos diagramas se muestran en las siguientes páginas.

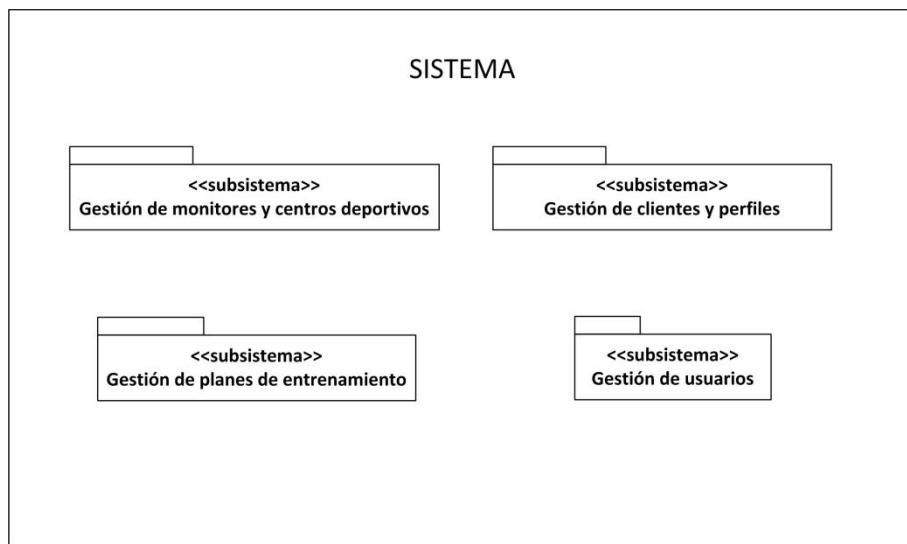


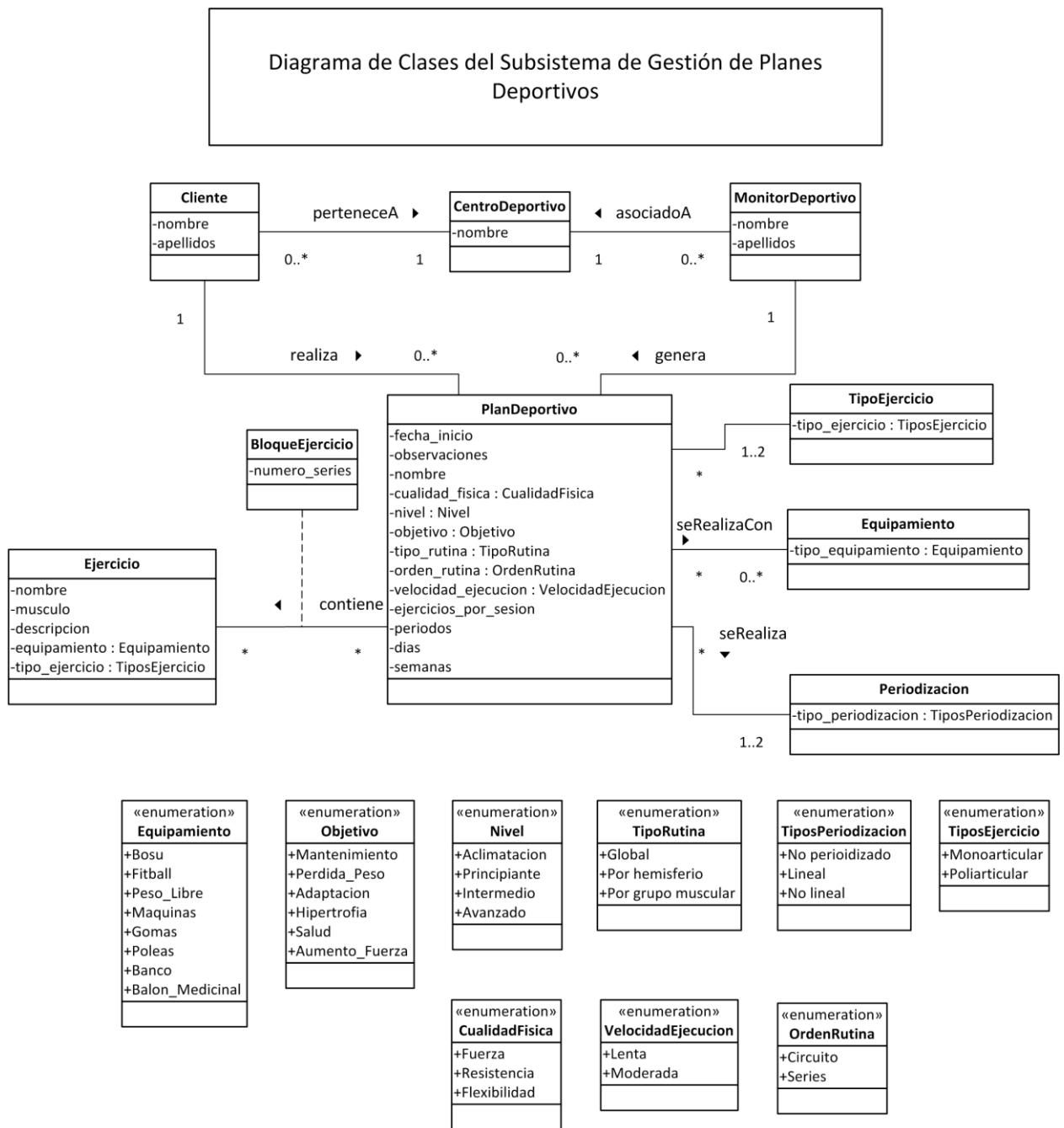
Figura 20: Diagrama de clases del sistema

- **Subsistema de gestión de usuarios:** tipos y asociaciones relacionados con la relación de usuarios, sus permisos y privilegios.
- **Subsistema de gestión de clientes y perfiles:** tipos y asociaciones relacionados con los usuarios finales de la aplicación y sus perfiles personalizados.
- **Subsistema de gestión de monitores y centros deportivos:** tipos y asociaciones relacionados con los monitores y los centros deportivos a los que están asociados.
- **Subsistema de gestión de planes de entrenamiento:** tipos y asociaciones relacionados con los planes de entrenamiento generados

### 7.1.1.- Diagramas de tipos

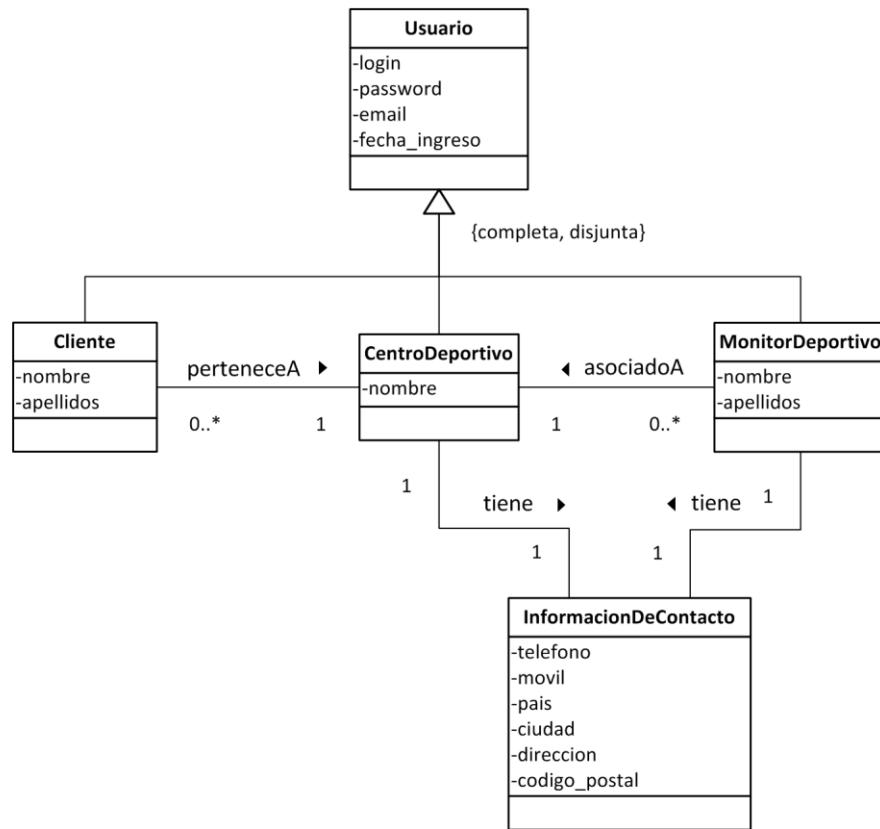
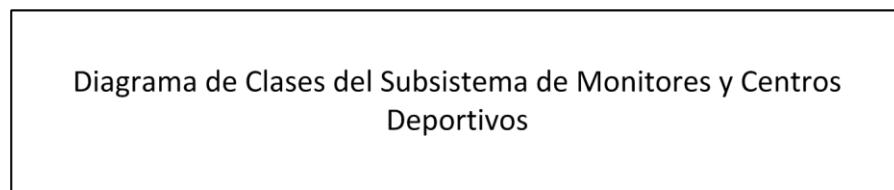
A continuación mostraremos cada uno de los diagramas de tipos separados por subsistemas. Los documentos que hemos consultados para la realización de estos diagramas son [\[31\]](#) y [\[32\]](#).

#### 7.1.1.1.- Subsistema de gestión de planes de entrenamiento



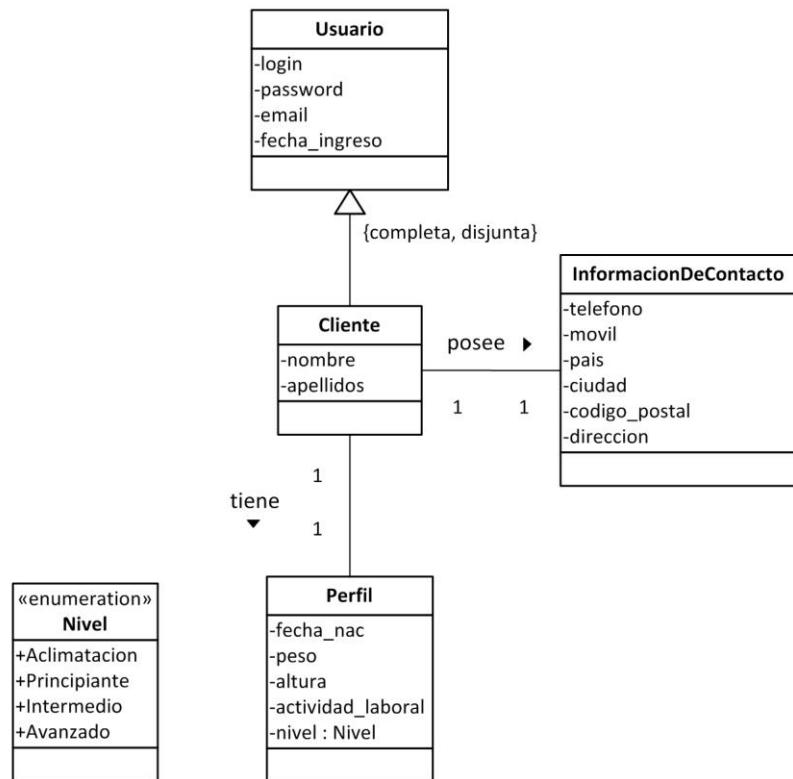
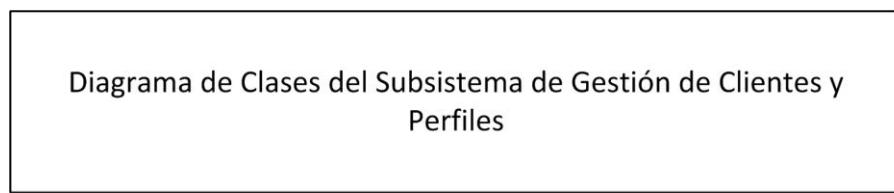
**Figura 21:** Diagrama de clases del subsistema de gestión de planes de entrenamiento

### 7.1.1.2.- Subsistema de gestión de monitores y centros deportivos



**Figura 22:** Diagrama de clases del subsistema de gestión de monitores y centros deportivos

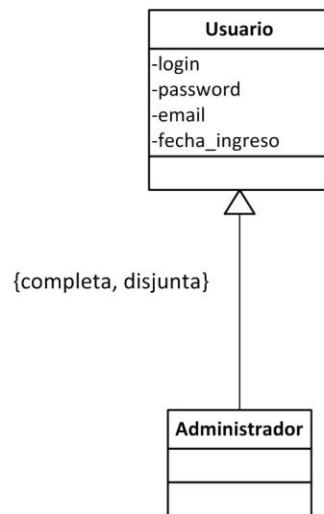
### 7.1.1.3.- Subsistema de gestión de clientes y perfiles



**Figura 23:** Diagrama de clases del subsistema de gestión de clientes y perfiles

**7.1.1.4.- Subsistema de gestión de usuarios**

Diagrama de Clases del Subsistema de Gestión de Usuarios



**Figura 24:** Diagrama de clases del subsistema de gestión de usuarios

### 7.1.2.- Escenario de prueba

Definimos a continuación un escenario de prueba para comprobar la integridad de los diagramas anteriormente realizados. En este escenario definimos un centro deportivo: *cd1*, al cual están asociados dos monitores deportivos: *m1* y *m2*, y dos clientes: *c1* y *c2*, cada uno con un perfil asociado: *p1* y *p2*.

El cliente *c1* tiene dos planes de entrenamiento generados: *pd1* y *pd2*. El primer plan deportivo, *pd1*, generado por el monitor deportivo *m1* y el segundo plan deportivo, *pd2*, generado por el otro monitor deportivo *m2*.

El cliente *c2* tiene un plan deportivo generado *pd3*, creado por el monitor deportivo *m2*.

Cada uno de los planes deportivos que se muestran a continuación en el escenario de prueba tienen una duración asociada (*d1*, *d2* y *d3* respectivamente), y un conjunto de ejercicios (*ej1*, *ej2*, *ej3*, *ej4*, *ej5*, *ej6*).

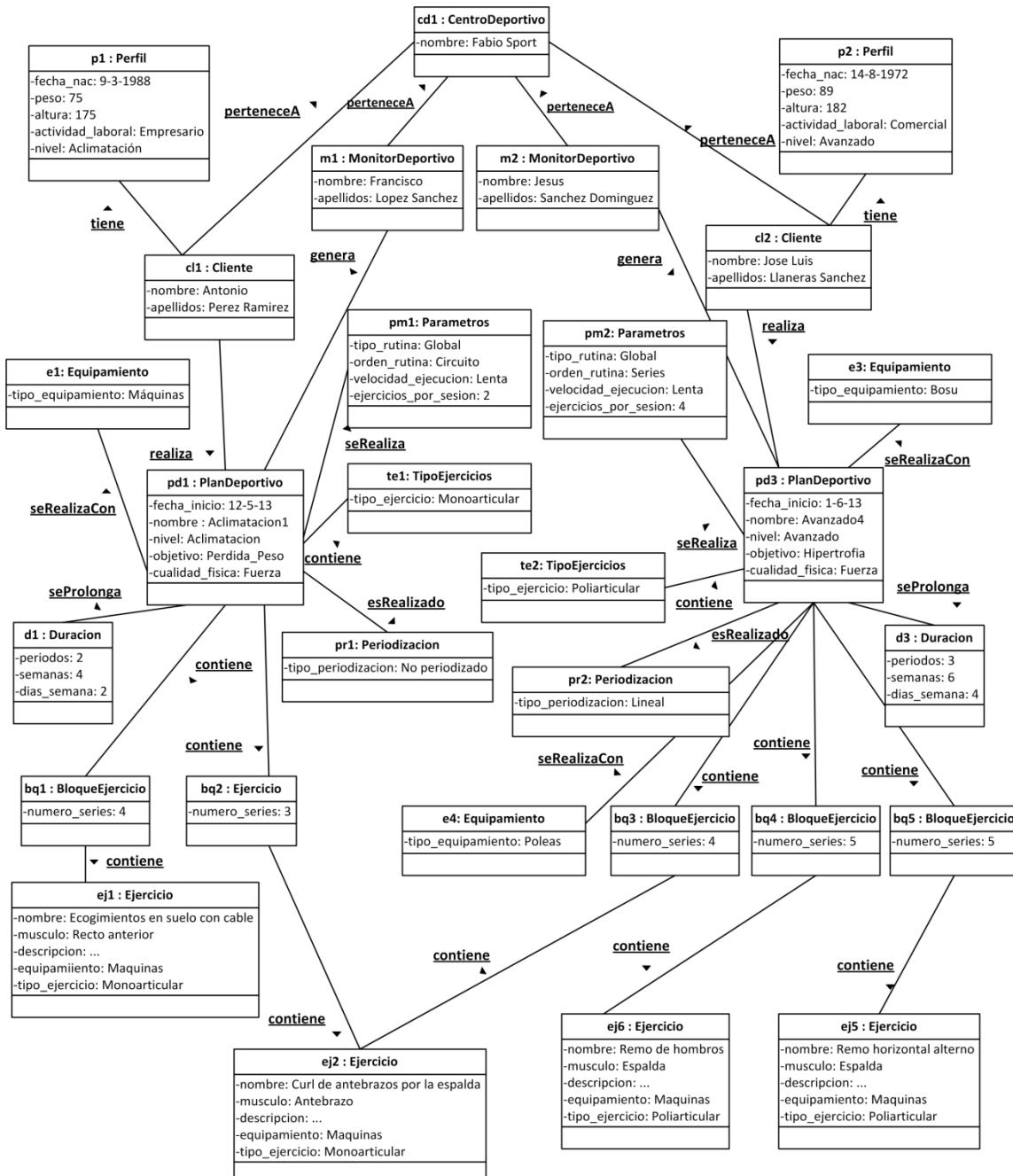


Figura 25: escenario de prueba

### 7.1.3.- Tipos

#### 7.1.2.1.- Subsistema de gestión de planes de entrenamiento

<b>TYP-0008</b>	<b>PlanDeportivo</b>		
<b>Versión</b>	1.0 ( 12/06/2013 )		
<b>Autores</b>	Francisco López Sánchez		
<b>Dependencias</b>	[OBJ-0004] Gestión de Planes de Entrenamiento		
<b>Descripción</b>	Este tipo de objetos representa <i>al plan deportivo generado por el sistema</i>		
<b>Componentes</b>	<b>Nombre</b>	<b>Tipo</b>	<b>Multiplicidad</b>
	cliente	Cliente	1
	monitor_deportivo	MonitorDeportivo	1
	nivel	Nivel	1
	objetivo	Objetivo	1
	cualidad_fisica	CualidadFisica	1
	duracion	Duración	1
	equipamiento	Set( Equipamiento )	1.. *
	ejercicios	Set( Ejercicio )	1.. *

<b>Atributo variable</b>	<b>PlanDeportivo:: fecha_inicio</b>
<b>Descripción</b>	Este atributo representa <i>a la fecha de comienzo del plan de entrenamiento</i>
<b>Tipo</b>	Date

<b>Atributo variable</b>	<b>PlanDeportivo:: observaciones</b>
<b>Descripción</b>	Este atributo representa <i>a las observaciones realizadas por el monitor deportivo</i>
<b>Tipo</b>	String

<b>TYP-0009</b>	<b>Duración</b>
<b>Versión</b>	1.0 ( 12/06/2013 )
<b>Autores</b>	Francisco López Sánchez
<b>Dependencias</b>	[OBJ-0004] Gestión de Planes de Entrenamiento
<b>Descripción</b>	Este tipo de objetos representa <i>a la duración del plan deportivo</i>

<b>Atributo variable</b>	<b>Duración:: periodos</b>
<b>Descripción</b>	Este atributo representa <i>numero de periodos en los que se divide el plan de entrenamiento</i>
<b>Tipo</b>	Integer

<b>Atributo variable</b>	<b>Duración:: semanas</b>
<b>Descripción</b>	Este atributo representa <i>número de semanas que comprende cada periodo</i>
<b>Tipo</b>	Integer

<b>Atributo variable</b>	<b>Duración:: dias_semanas</b>
<b>Descripción</b>	Este atributo representa <i>número de días que se entrenará cada semana</i>
<b>Tipo</b>	Integer

<b>TYP-0010</b>	<b>Ejercicio</b>
<b>Versión</b>	1.0 ( 12/06/2013 )
<b>Autores</b>	Francisco López Sánchez
<b>Dependencias</b>	[OBJ-0004] Gestión de Planes de Entrenamiento
<b>Descripción</b>	Este tipo de objetos representa <i>al ejercicio físico incluido en el plan de entrenamiento</i>

<b>Atributo variable</b>	<b>Ejercicio:: nombre</b>
<b>Descripción</b>	Este atributo representa <i>al nombre que caracteriza al ejercicio</i>
<b>Tipo</b>	String

<b>Atributo variable</b>	<b>Ejercicio:: músculo</b>
<b>Descripción</b>	Este atributo representa <i>al músculo que implica el ejercicio</i>
<b>Tipo</b>	String

<b>Atributo variable</b>	<b>Ejercicio:: descripción</b>
<b>Descripción</b>	Este atributo representa <i>a la explicación de la realización del ejercicio</i>
<b>Tipo</b>	String

<b>Atributo variable</b>	<b>Ejercicio:: comentarios</b>
<b>Descripción</b>	Este atributo representa <i>a los comentarios referentes al ejercicio</i>
<b>Tipo</b>	String

<b>TYP-0012</b>	<b>Equipamiento</b>
<b>Versión</b>	1.0 ( 12/06/2013 )
<b>Autores</b>	Francisco López Sánchez
<b>Dependencias</b>	[OBJ-0004] Gestión de Planes de Entrenamiento
<b>Descripción</b>	Este tipo de objetos representa <i>al tipo de equipamiento utilizado en cada ejercicio</i>
<b>Comentarios</b>	Enumerado de los tipos de Equipamiento

<b>TYP-0013</b>	<b>CualidadFisica</b>
<b>Versión</b>	1.0 ( 12/06/2013 )
<b>Autores</b>	Francisco López Sánchez
<b>Dependencias</b>	[OBJ-0004] Gestión de Planes de Entrenamiento
<b>Descripción</b>	Este tipo de objetos representa <i>a la calidad física que se ejercita</i>
<b>Comentarios</b>	Enumerado de los tipos de CualidadFisica

<b>TYP-0014</b>	<b>Objetivo</b>
<b>Versión</b>	1.0 ( 12/06/2013 )
<b>Autores</b>	Francisco López Sánchez
<b>Dependencias</b>	[OBJ-0004] Gestión de Planes de Entrenamiento
<b>Descripción</b>	Este tipo de objetos representa <i>al objetivo que se desea conseguir con el entrenamiento</i>
<b>Comentarios</b>	Enumerado de los tipos de Objetivo

**Atributo variable Cliente:: nombre**

<b>Descripción</b>	Este atributo representa <i>al nombre completo del cliente</i>
<b>Tipo</b>	String

**7.1.2.2.- Subsistema de gestión de monitores y centros deportivos**

<b>TYP-0006 CentroDeportivo</b>			
<b>Versión</b>	1.0 ( 12/06/2013 )		
<b>Autores</b>	Francisco López Sánchez		
<b>Dependencias</b>	[OBJ-0001] Gestión de Centros Deportivos		
<b>Descripción</b>	Este tipo de objetos representa <i>al centro deportivo dado de alta en el sistema</i>		
<b>Componentes</b>	<b>Nombre</b>	<b>Tipo</b>	<b>Multiplicidad</b>
	informacion_contacto	InformacionDeContacto	1
	monitores_deportivos	Set( MonitorDeportivo )	0.. *
	clientes	Set( Cliente )	0.. *

**Atributo variable CentroDeportivo:: nombre**

<b>Descripción</b>	Este atributo representa <i>al nombre del centro deportivo</i>
<b>Tipo</b>	String

**TYP-0007 MonitorDeportivo**

<b>TYP-0007 MonitorDeportivo</b>			
<b>Versión</b>	1.0 ( 12/06/2013 )		
<b>Autores</b>	Francisco López Sánchez		
<b>Dependencias</b>	[OBJ-0002] Gestión de Monitores Deportivos		
<b>Descripción</b>	Este tipo de objetos representa <i>al monitor deportivo dado de alta en el sistema</i>		
<b>Componentes</b>	<b>Nombre</b>	<b>Tipo</b>	<b>Multiplicidad</b>
	informacion_contacto	InformacionDeContacto	1
	planes_deportivos	Set( PlanDeportivo )	0.. *

**7.1.2.3.- Subsistema de gestión de clientes y perfiles**

**TYP-0002 Cliente**

<b>TYP-0002 Cliente</b>			
<b>Versión</b>	1.0 ( 12/06/2013 )		
<b>Autores</b>	Francisco López Sánchez		
<b>Dependencias</b>	[OBJ-0003] Gestión de Clientes		
<b>Descripción</b>	Este tipo de objetos representa <i>a los usuarios para los que se crearán los planes de entrenamiento</i>		
<b>Componentes</b>	<b>Nombre</b>	<b>Tipo</b>	<b>Multiplicidad</b>
	informacion_contacto	InformacionDeContacto	1
	planes_deportivos	Set( PlanDeportivo )	0.. '
	centro_deportivo	CentroDeportivo	1

<b>Atributo variable</b>	<b>Cliente:: apellidos</b>
<b>Descripción</b>	Este atributo representa <i>a los apellidos del cliente</i>
<b>Tipo</b>	String

<b>Atributo variable</b>	<b>InformacionDeContacto:: telefono</b>
<b>Descripción</b>	Este atributo representa <i>al teléfono fijo del cliente</i>
<b>Tipo</b>	String

<b>Atributo variable</b>	<b>InformacionDeContacto:: móvil</b>
<b>Descripción</b>	Este atributo representa <i>al teléfono móvil del cliente</i>
<b>Tipo</b>	String

<b>TYP-0003</b>	<b>InformacionDeContacto</b>						
<b>Versión</b>	1.0 ( 12/06/2013 )						
<b>Autores</b>	Francisco López Sánchez						
<b>Dependencias</b>	[OBJ-0003] Gestión de Clientes						
<b>Descripción</b>	Este tipo de objetos representa <i>a la información de contacto de cada cliente</i>						
<b>Componentes</b>	<table> <thead> <tr> <th><b>Nombre</b></th> <th><b>Tipo</b></th> <th><b>Multiplicidad</b></th> </tr> </thead> <tbody> <tr> <td>direccion</td> <td>Dirección</td> <td>1</td> </tr> </tbody> </table>	<b>Nombre</b>	<b>Tipo</b>	<b>Multiplicidad</b>	direccion	Dirección	1
<b>Nombre</b>	<b>Tipo</b>	<b>Multiplicidad</b>					
direccion	Dirección	1					

<b>TYP-0004</b>	<b>Dirección</b>
<b>Versión</b>	1.0 ( 12/06/2013 )
<b>Autores</b>	Francisco López Sánchez
<b>Dependencias</b>	[OBJ-0003] Gestión de Clientes
<b>Descripción</b>	Este tipo de objetos representa <i>a la dirección de contacto del cliente</i>

<b>TYP-0005</b>	<b>Perfil</b>
<b>Versión</b>	1.0 ( 12/06/2013 )
<b>Autores</b>	Francisco López Sánchez
<b>Dependencias</b>	[OBJ-0003] Gestión de Clientes [OBJ-0006] Gestión de Perfil
<b>Descripción</b>	Este tipo de objetos representa <i>a la información referente a las características del cliente</i>

Atributo variable	Dirección:: país
Descripción	Este atributo representa <i>al país donde se encuentra la dirección del cliente</i>
Tipo	String

Atributo variable	Dirección:: ciudad
Descripción	Este atributo representa <i>a la ciudad donde se encuentra la dirección del cliente</i>
Tipo	String

Atributo variable	Dirección:: codigo_postal
Descripción	Este atributo representa <i>al código postal de la dirección del cliente</i>
Tipo	String

Atributo variable	Dirección:: dirección
Descripción	Este atributo representa <i>al nombre de la calle, número de vivienda y otros datos relativos a la dirección( piso, puerta)</i>
Tipo	String

Atributo variable	Perfil:: fecha_nac
Descripción	Este atributo representa <i>a la fecha de nacimiento del cliente</i>
Tipo	Date

Atributo variable	Perfil:: peso
Descripción	Este atributo representa <i>al peso del cliente</i>
Tipo	Real

Atributo variable	Perfil:: altura
Descripción	Este atributo representa <i>a la altura del cliente</i>
Tipo	Real

Atributo variable	Perfil:: actividad_laboral
Descripción	Este atributo representa <i>a la ocupación del cliente</i>
Tipo	String

TYP-0011	Nivel
Versión	1.0 ( 12/06/2013 )
Autores	Francisco López Sánchez
Dependencias	[OBJ-0004] Gestión de Planes de Entrenamiento
Descripción	Este tipo de objetos representa <i>al nivel de entrenamiento del cliente</i>
Comentarios	Enumerado de los tipos de Nivel

### 7.1.2.4.- Subsistema de gestión de usuarios

<b>TYP-0001</b>	<b>UsuarioRegistrado</b>
<b>Versión</b>	1.0 ( 12/06/2013 )
<b>Autores</b>	Francisco López Sánchez
<b>Dependencias</b>	[OBJ-0003] Gestión de Clientes
<b>Descripción</b>	Este tipo de objetos representa <i>a los usuarios que se han dado de alta en el sistema</i>

<b>Atributo variable</b>	<b>UsuarioRegistrado:: login</b>
<b>Descripción</b>	Este atributo representa <i>al nombre de usuario con el que accederá a la aplicación</i>
<b>Tipo</b>	String

<b>Atributo variable</b>	<b>UsuarioRegistrado:: password</b>
<b>Descripción</b>	Este atributo representa <i>la contraseña para acceder al sistema</i>
<b>Tipo</b>	String

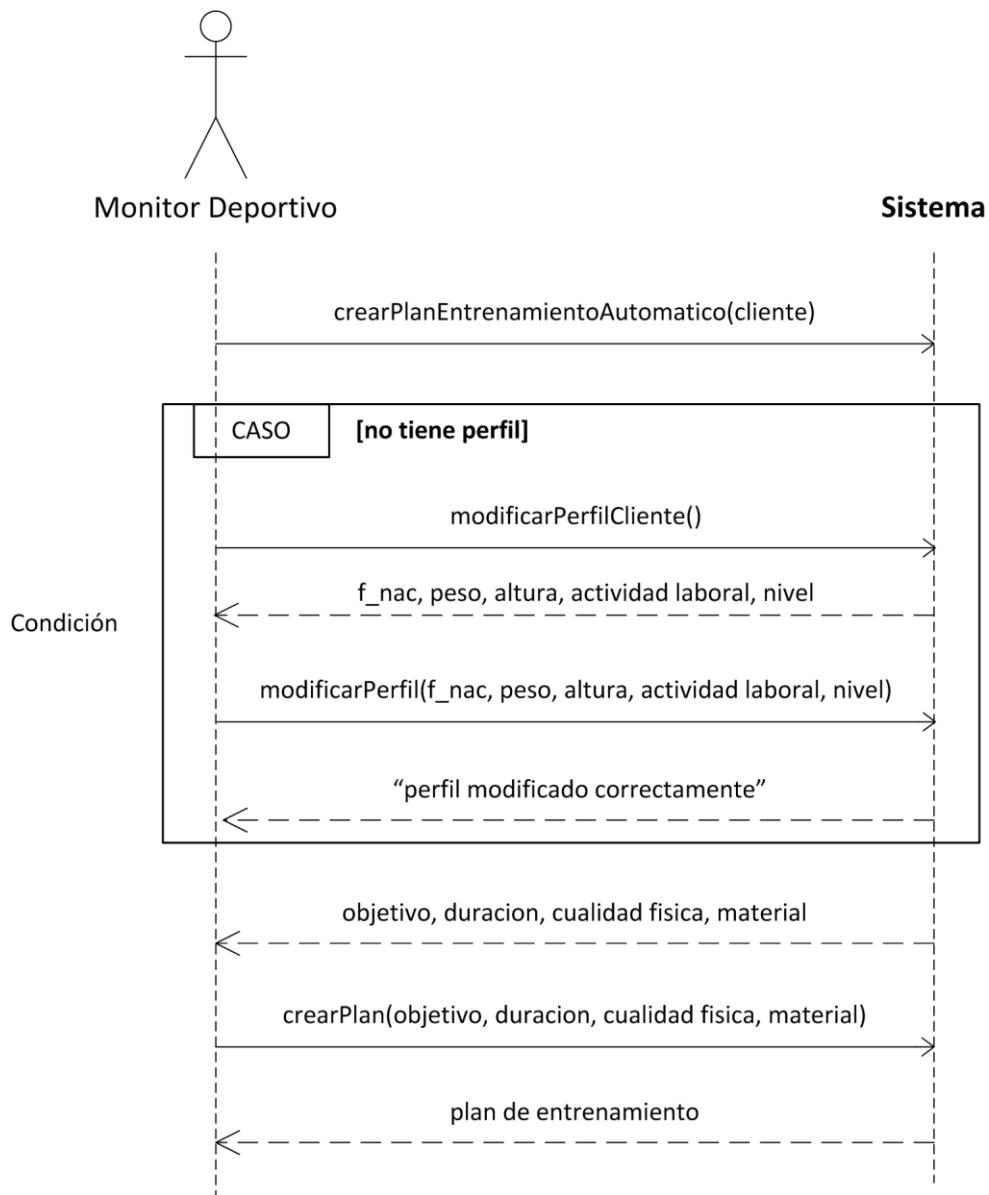
<b>Atributo variable</b>	<b>UsuarioRegistrado:: email</b>
<b>Descripción</b>	Este atributo representa <i>la dirección de correo electrónico</i>
<b>Tipo</b>	String

<b>Atributo variable</b>	<b>UsuarioRegistrado:: fecha_ingreso</b>
<b>Descripción</b>	Este atributo representa <i>la fecha en la que el usuario fue dado de alta en el sistema</i>
<b>Tipo</b>	Date

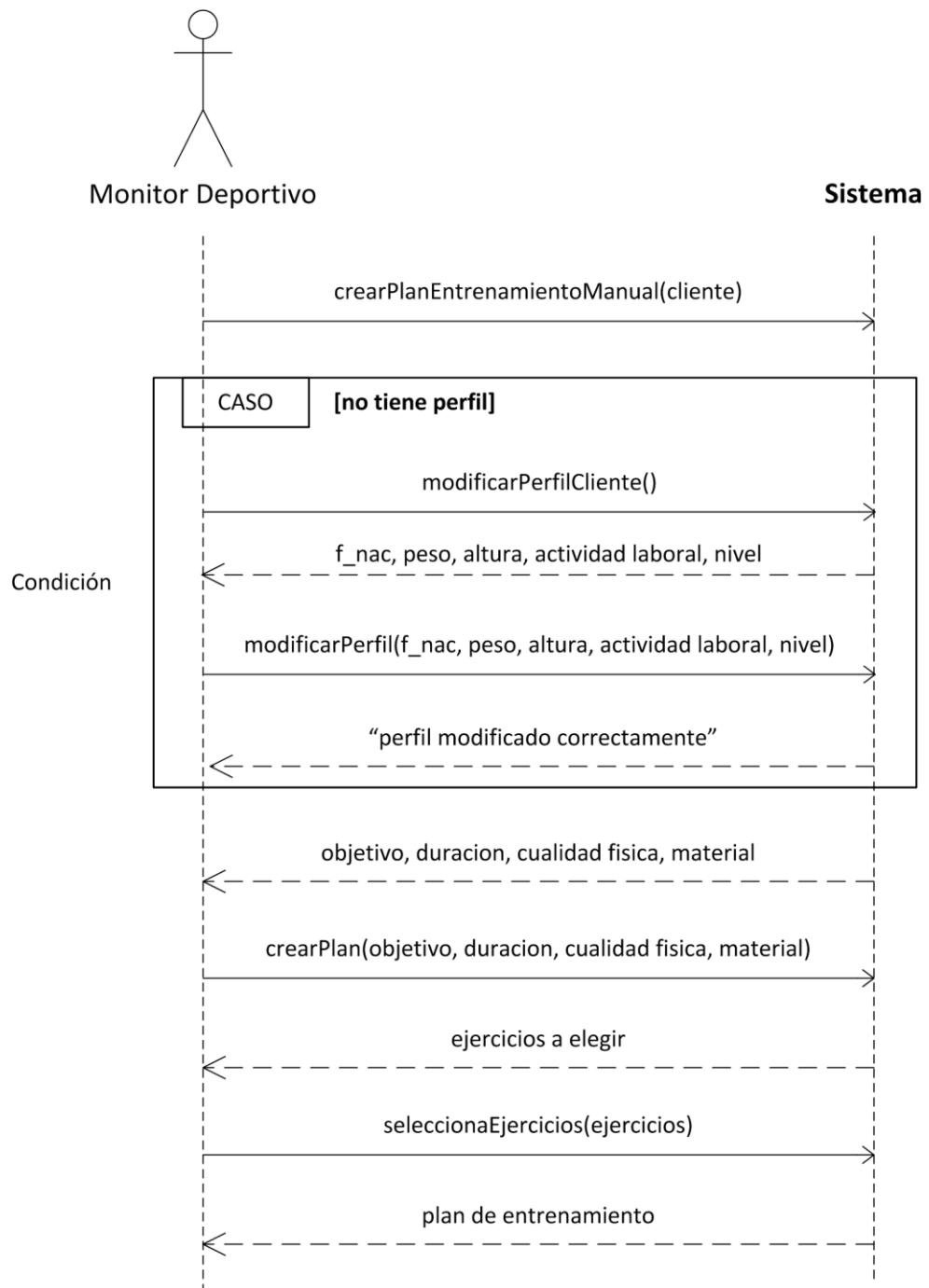
<b>Expresión de invariante</b>	<b>UsuarioRegistrado::Unicidad de nombre de usuario</b>
<b>Descripción</b>	no pueden existir dos usuarios con el mismo login
<b>Comentarios</b>	el nombre de usuario es el atributo login

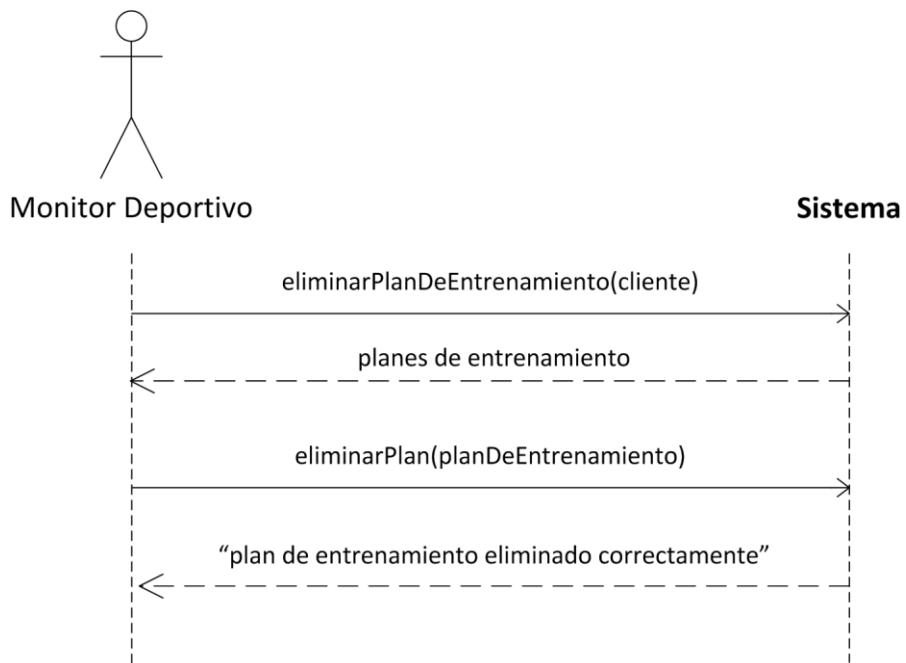
## 7.2.- Modelo dinámico del sistema

### 7.2.1.- Subsistema de gestión de planes de entrenamiento



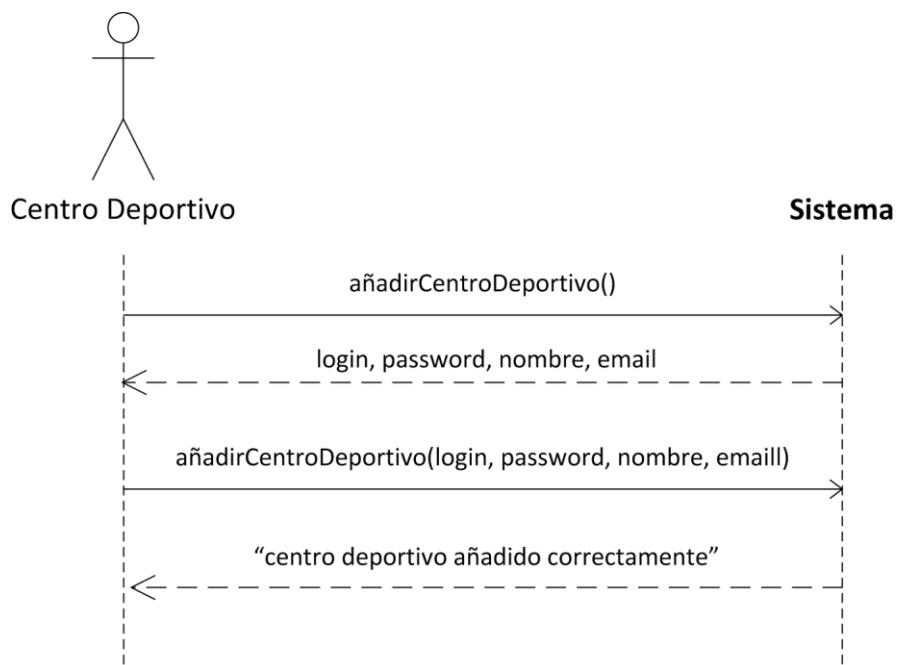
**Figura 26:** crearPlanEntrenamientoAutomatico()

**Figura 27:** crearPlanEntrenamientoManual()

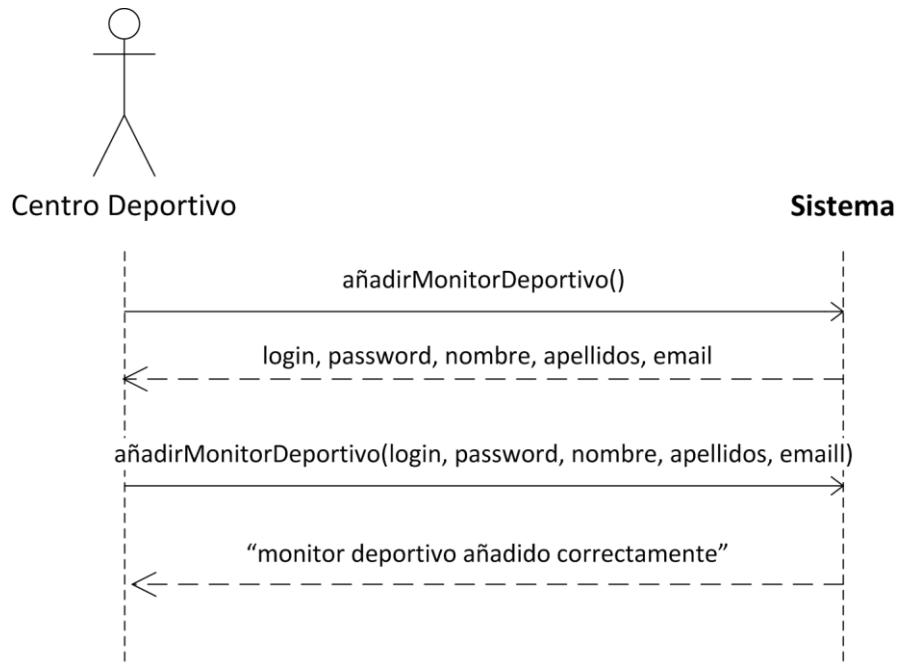


**Figura 28:** eliminarPlanDeEntrenamiento()

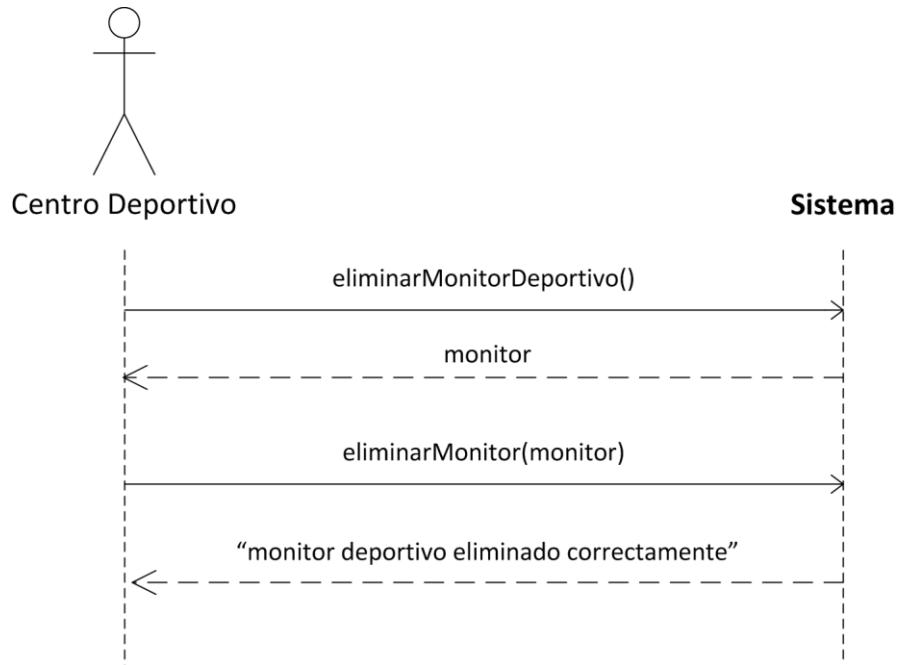
### 7.2.2.- Subsistema de gestión de monitores y centros deportivos



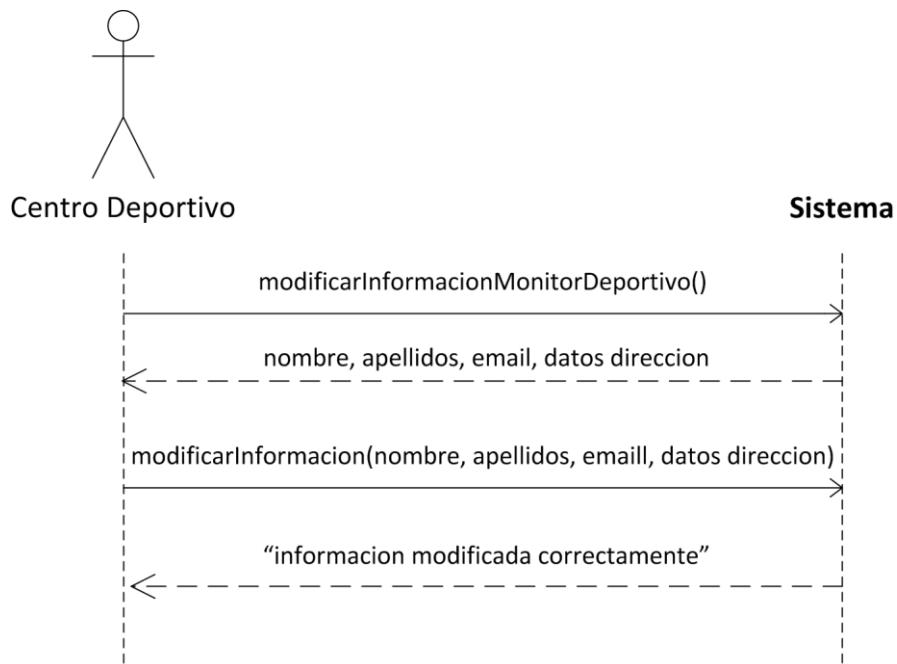
**Figura 29:** añadirCentroDeportivo()



**Figura 30:** añadirMonitorDeportivo()

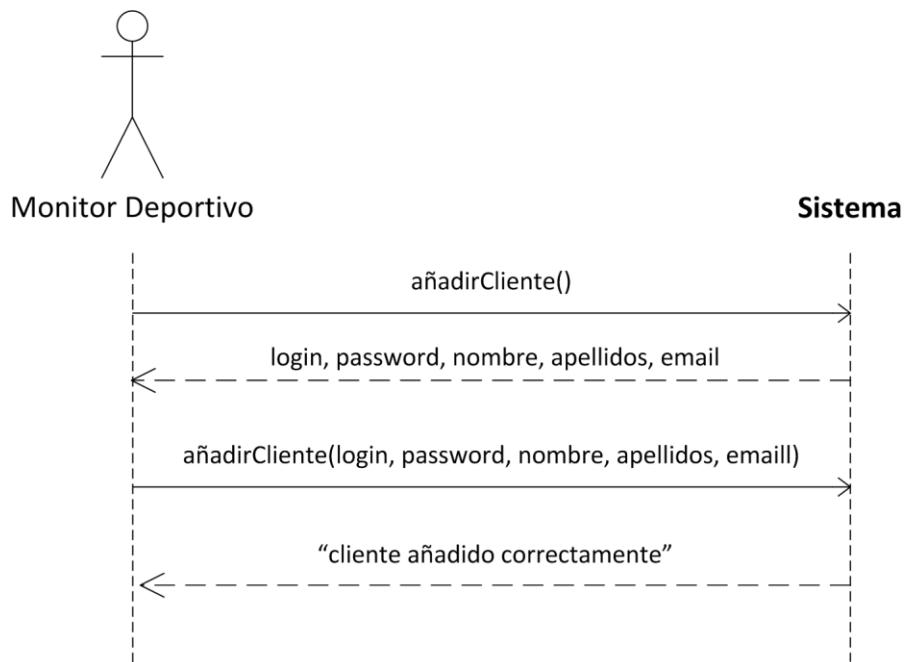


**Figura 31:** eliminarMonitorDeportivo()

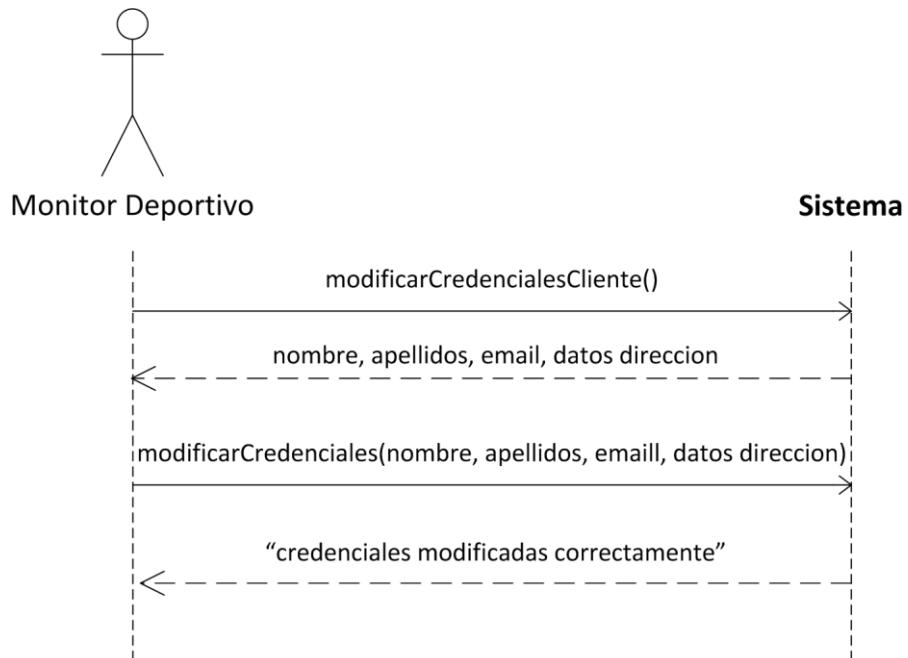


**Figura 32:** modificarInformacionMonitorDeportivo()

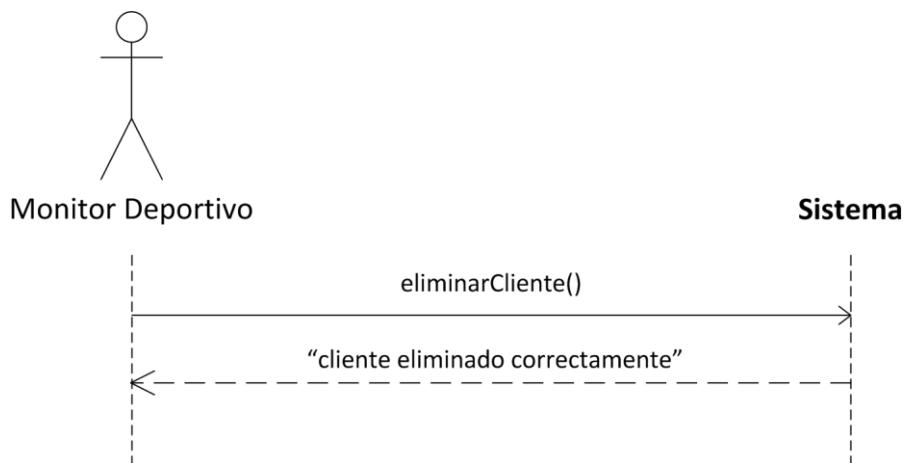
### 7.2.3.- Subsistema de gestión de clientes y perfiles



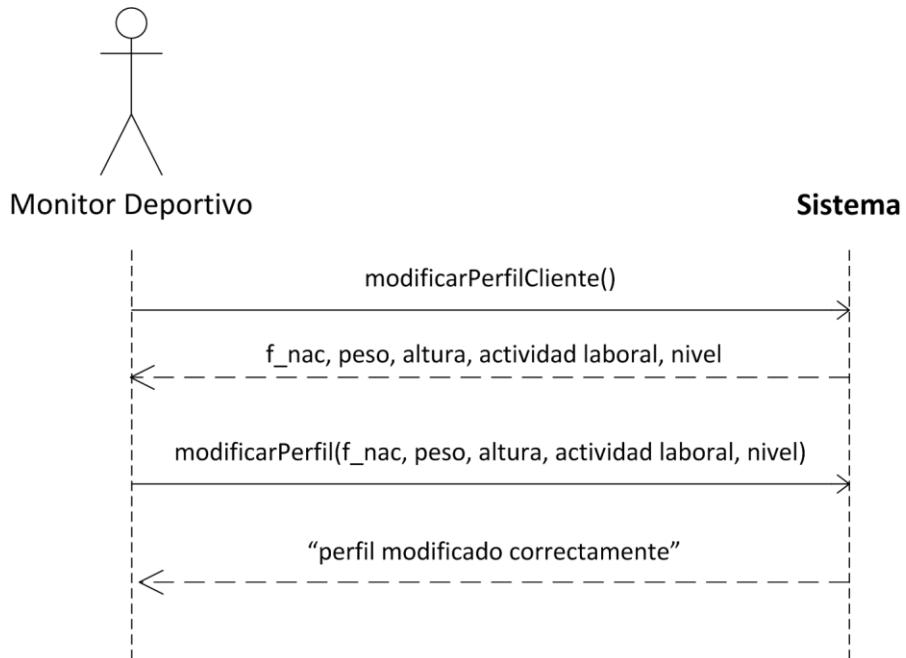
**Figura 33:** añadirCliente()



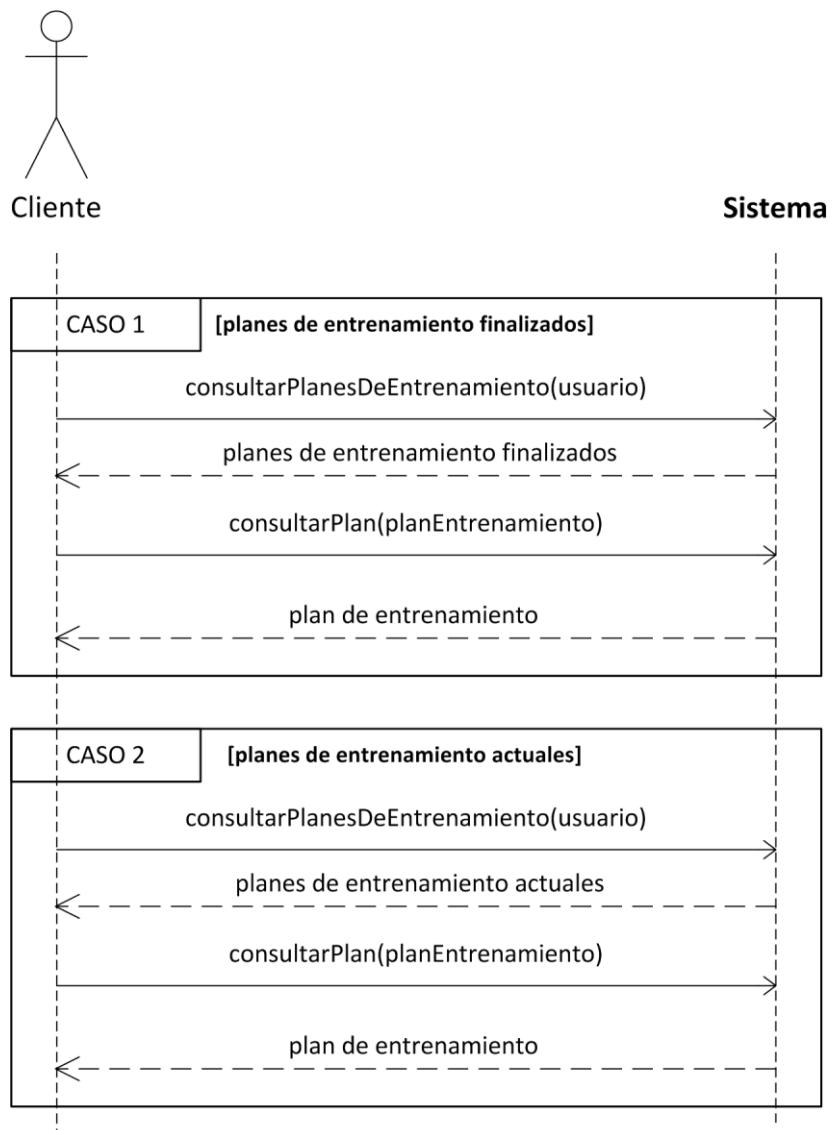
**Figura 34:** modificarCredencialesCliente()



**Figura 35:** eliminarCliente()

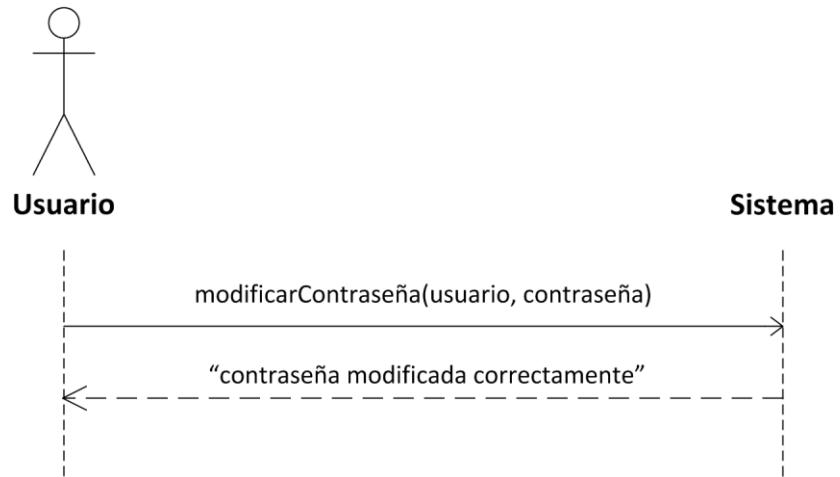


**Figura 36:** modificarPerfilCliente()

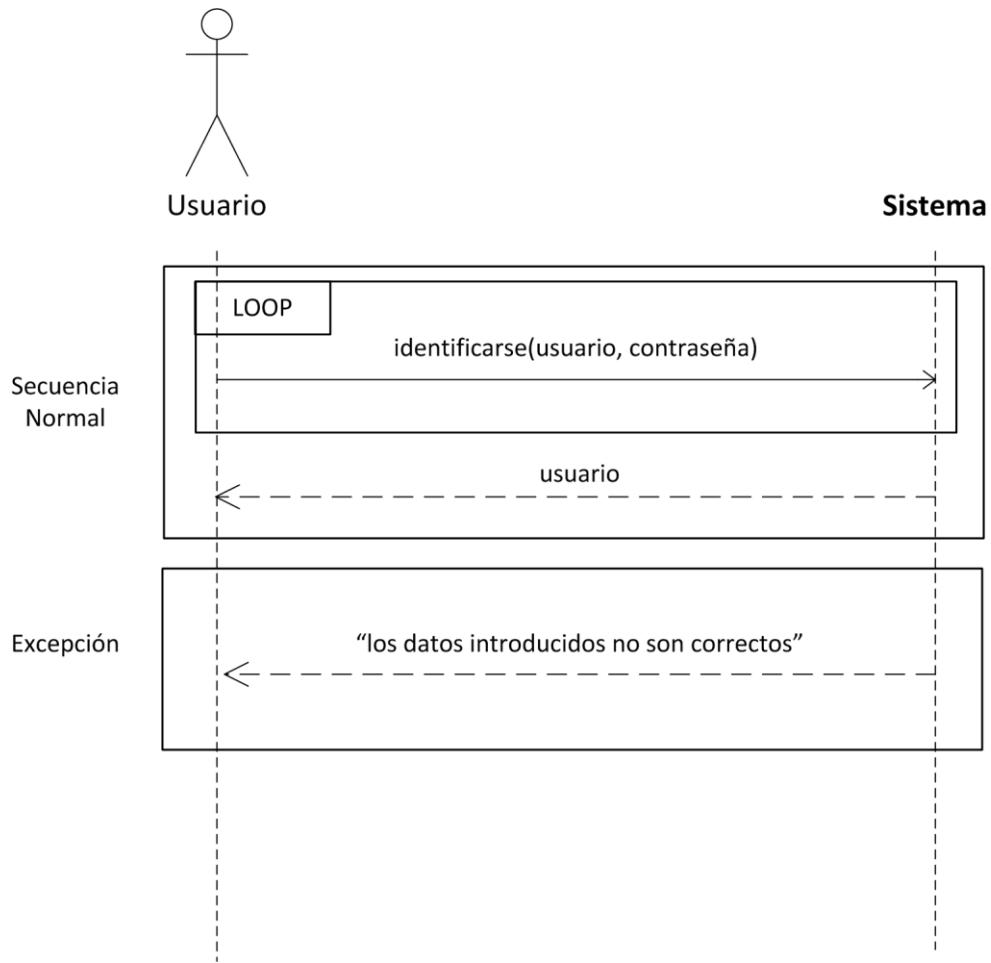


**Figura 37:** consultarPlanesDeEntrenamiento()

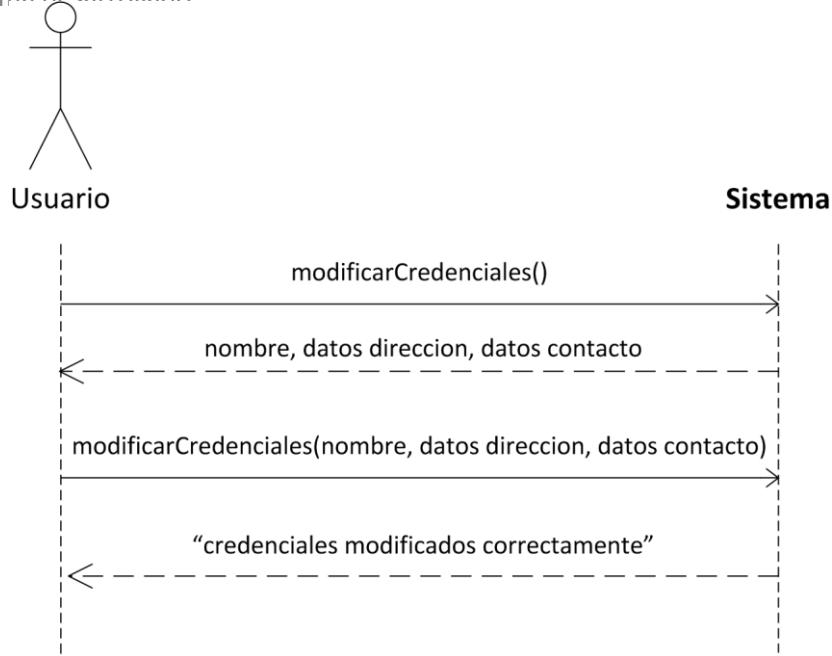
#### 7.2.4.- Subsistema de gestión de usuarios



**Figura 38:** modificarContraseña(usuario, contraseña)



**Figura 39:** identificacionUsuario(usuario, contraseña)

**Figura 40:** modificarCredenciales()

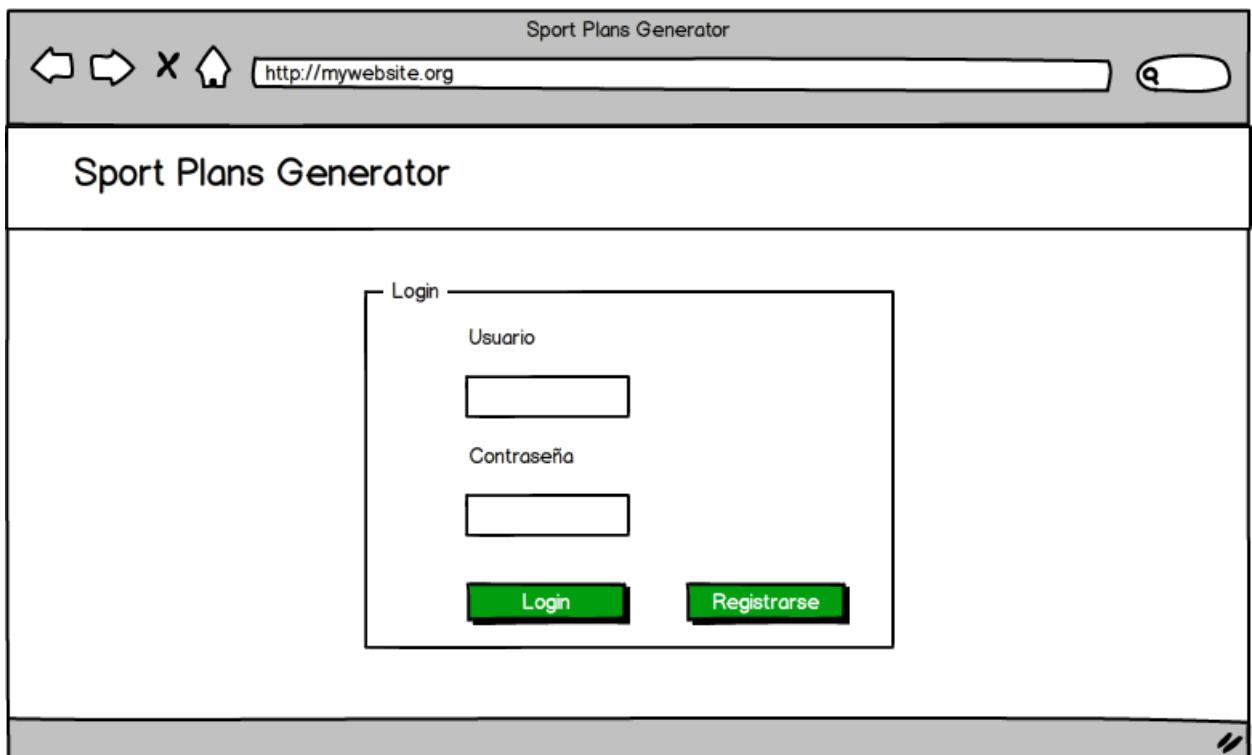
### 7.3.- Prototipos de interfaz de usuario

A continuación mostraremos las vistas más importantes que conformarán nuestra aplicación. Para la realización de estos diseños hemos usado el software *Balsamiq Mockups* en su versión de prueba [40].

Estos prototipos han sido obtenidos tras varias reuniones con el cliente y a partir de los requisitos obtenidos. Serán utilizados como medio de comunicación con el cliente, como una forma sencilla para él de comprobar que los requisitos que nos está pidiendo aparecen todos en las interfaces mostradas. Por nuestra parte, serán usados para cerciorarnos de que lo que nos está pidiendo se adecua a lo que hemos entendido a partir de los requisitos, y también para comprobar si existen inconsistencias en los requisitos obtenidos o si es necesario añadir o eliminar alguno.

### Pantalla de login

Se trata de la pantalla de inicio de la aplicación. Desde ella los distintos usuarios accederán al sistema.



**Figura 41:** diseño de la interfaz de la pantalla de login.

### Pantalla de registro

Se trata de la vista de registro de nuevos usuarios de la aplicación. A través de ella, los nuevos usuarios pasarán a formar parte de nuestro sistema introduciendo los campos requeridos.

The screenshot shows a web browser window titled "Sport Plans Generator". The address bar contains the URL "http://mywebsite.org". The main content area is titled "Sport Plans Generator" and features a form for "Nuevo Centro Deportivo". The form includes fields for "Usuario" (with an input box), "Password" (with an input box), "Repita Password" (with an input box), "Nombre Centro Deportivo" (with an input box), and "Email" (with an input box). A green button labeled "Dar de alta" (Register) is at the bottom of the form. The entire form is enclosed in a black rectangular border.

**Figura 42:** diseño de la interfaz de registro.

### Panel de centro deportivo

En esta vista podemos ver el panel de control de un centro deportivo. Se muestran algunos datos de ese centro deportivo tales como el número de monitores deportivos, de clientes y de planes deportivos que se han generado en ese centro deportivo. También se muestra una lista con todos los monitores deportivos y clientes que ese centro deportivo tiene registrados, además del formulario para el registro de nuevos monitores.

The screenshot shows the 'Sport Plans Generator' application interface. At the top, there is a header bar with icons for back, forward, and search, and a URL field containing 'http://mywebsite.org'. The main title 'Sport Plans Generator' is displayed, along with a welcome message 'Bienvenido usuario' and the date '10 de mayo de 2013'. A 'Cerrar sesión' (Logout) link is also present.

In the center, there is a summary section for the sports center, showing:

- Centro Deportivo - Nombre: \_\_\_\_\_
- Nº monitores : 4
- Nº clientes : 29
- Nº planes deportivos generados : 36

Below this is a table titled 'Monitores Deportivos' with columns: Nombre, Apellido, Información, and Eliminar. The data is as follows:

Nombre	Apellido	Información	Eliminar
Monitor1	monitor1		
Monitor2	monitor2		
Monitor3	monitor3		
Monitor4	monitor4		
Monitor5	monitor5		
Monitor6	monitor6		
Monitor7	monitor7		

At the bottom, there is a form titled 'Nuevo Monitor Deportivo' with fields for Nombre, Usuario, Password, Apellidos, and Email, and a 'Registrar Monitor' button with a checkmark icon.

**Figura 43:** diseño de la interfaz del panel de control del centro deportivo.

### Panel de monitor deportivo

En esta vista podemos ver el panel de control de un monitor deportivo. En él se muestran algunos datos de ese monitor deportivo tales como los clientes asociados a su mismo centro deportivo y de planes deportivos que ha generado. También se muestra una lista con todos los clientes además del formulario para el registro de nuevos clientes.

The screenshot shows the 'Sport Plans Generator' control panel. At the top, there are navigation icons (back, forward, search, home) and a URL bar showing 'http://mywebsite.org'. The title 'Sport Plans Generator' is displayed above a welcome message: 'Bienvenido usuario' and the date '10 de mayo de 2013'. A 'Cerrar sesión' (Logout) link is also present.

A large text box displays monitoring information:

- Monitor Deportivo - Nombre: \_\_\_\_\_
- Centro Deportivo - Fabio Sport
- Nº clientes : 12
- Nº planes deportivos generados : 18

Below this is a table titled 'Socios' (Clients) with a 'Panel de Control' (Control Panel) tab selected. The table lists 7 clients with the following details:

Nombre	Apellido	F_nac	Peso	Altura	Act_laboral	Nivel	Perfil	Info	Eliminar
Socio1	socio1	12/6/200	68kg	176cm	Estudiante	Intermedio			
Socio2	socio2	12/6/1988	78kg	186cm	Profesor	Avanzado			
Socio3	socio3	12/6/1990	88kg	198cm	Tendero	Aclimatación			
Socio4	socio4	12/6/1995	99kg	188cm	Taxista	Intermedio			
Socio5	socio5	12/6/1976	58kg	190cm	Estudiante	Principiante			
Socio6	socio6	12/6/1980	67kg	165cm	Sector Informatico	Avanzado			
Socio7	socio7	12/6/1990	69kg	176cm	Doctor	Intermedio			

At the bottom, a 'Nuevo Cliente' (New Client) form is shown with fields for Nombre, Apellidos, Usuario, Email, Password, and a 'Registrar Cliente' (Register Client) button with a checkmark icon.

**Figura 44:** diseño de la interfaz del panel de control del monitor deportivo.

### Pantalla de modificación de perfil

Por medio de esta pantalla los monitores deportivos tendrán acceso al perfil del cliente, si éste hubiese sido creado con anterioridad, y podrán modificar dicho perfil.

The screenshot shows a web browser window for 'Sport Plans Generator' at the URL <http://mywebsite.org>. The page title is 'Sport Plans Generator'. On the right, it says 'Bienvenido usuario' and '10 de mayo de 2013', with a 'Cerrar sesión' button. The main content area is titled 'Modificar Perfil - Nombre Cliente'. It contains fields for 'Peso (kg)' and 'Altura (cm)', each with an input field. Below these are fields for 'Actividad Laboral' (with an input field) and 'Fecha de Nacimiento' (with an input field and a calendar icon). A 'Nivel' section includes radio buttons for 'Aclimatación' (selected), 'Principiante', 'Intermedio', and 'Avanzado', followed by a green 'Guardar Perfil' button.

**Figura 45:** diseño de la interfaz del panel de modificación del perfil del cliente.

### Panel de generación de plan deportivo (primer paso)

Esta vista muestra el primer paso para la creación de un nuevo plan deportivo. En ella podemos ver los primeros parámetros que el monitor deportivo debe introducir para la creación de dicho plan.

The screenshot shows a web-based application titled "Sport Plans Generator". At the top, there are navigation icons (back, forward, search, etc.) and a URL bar containing "http://mywebsite.org". Below the header, the title "Sport Plans Generator" is displayed, along with a welcome message "Bienvenido usuario" and the date "10 de mayo de 2013". A "Cerrar sesión" (Logout) link is also present.

The main form is titled "Generar Plan Deportivo - PASO 1". It includes the following fields:

- Nivel Cliente:** A dropdown menu currently set to "Aclimatación".
- Objetivo:** Radio buttons for "Adaptación" (selected), "Pérdida de peso", and "Mantenimiento".
- Duración:** Text input "1 periodo de" followed by a dropdown menu "semanas" containing the value "6".
- Cualidad Física a entrenar:** Radio buttons for "Fuerza" (selected), "Resistencia", and "Elasticidad".
- Sesiones por semana:** Text input "3" followed by a dropdown menu "sesiones".
- Días de entrenamiento:** A list of checkboxes for days of the week: Lunes (checked), Martes, Miércoles (checked), Jueves, Viernes (checked), Sábado, and Domingo.

A green "Continuar" (Continue) button is located at the bottom left of the form area.

**Figura 46:** diseño de la interfaz de generación de planes de entrenamiento.

### Panel de generación de plan deportivo (segundo paso)

En esta vista se muestra el segundo paso para la creación de un plan deportivo. Aquí podemos ver como se muestra tanto información de confirmación introducida en el paso anterior como nuevos parámetros necesarios para la creación del plan de entrenamiento.

The screenshot shows a web-based application titled "Sport Plans Generator". At the top, there are navigation icons (back, forward, search) and a URL bar containing "http://mywebsite.org". The main header says "Sport Plans Generator" and includes a welcome message "Bienvenido usuario" and the date "10 de mayo de 2013", with a "Cerrar sesión" link.

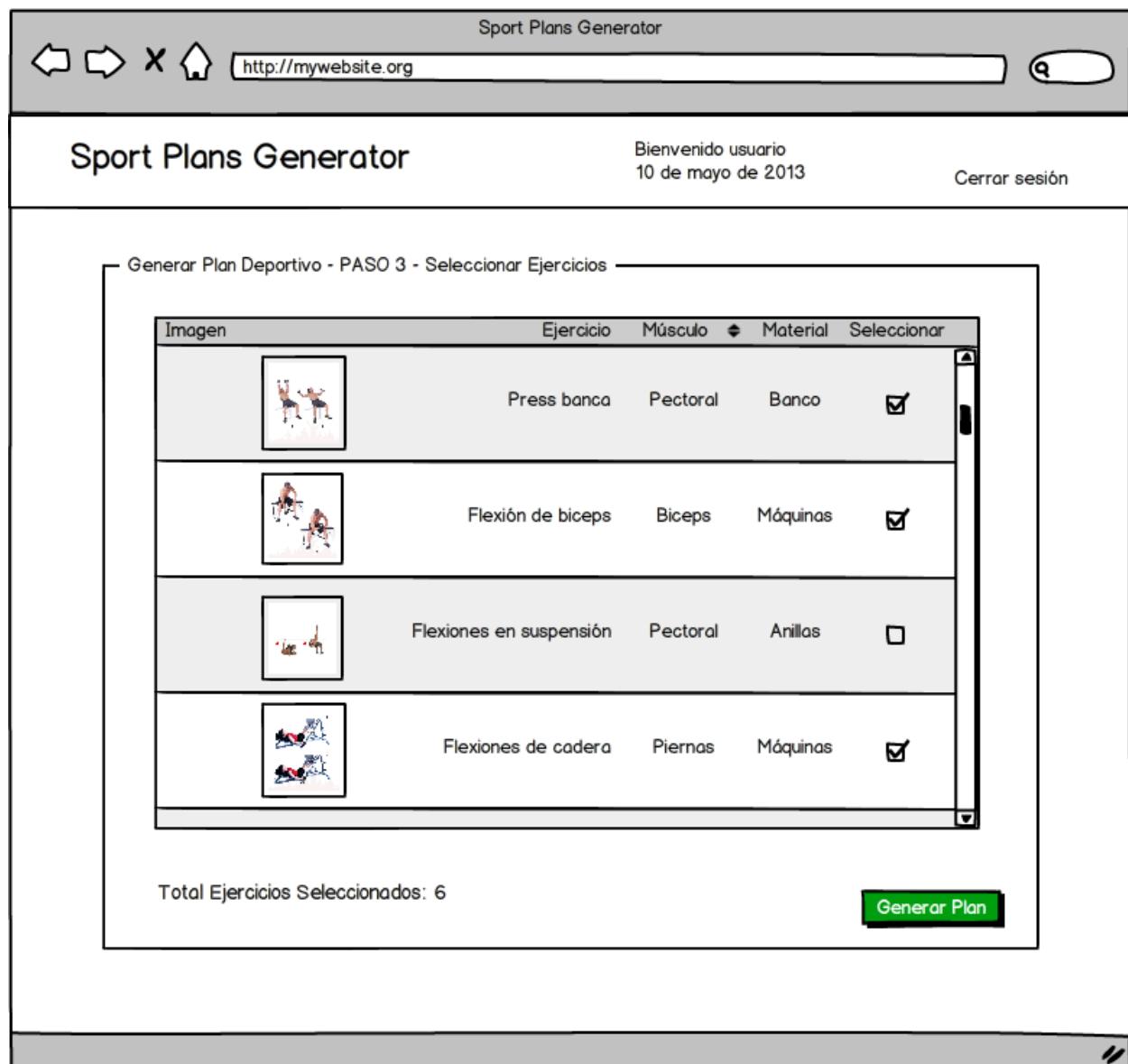
The central area is titled "Generar Plan Deportivo - PASO 2". It contains several configuration sections:

- Nivel Cliente:** Aclimatación (selected), Objetivo, Adaptación, Duración.
- Duración:** 1 periodo de 6 semanas (3 sesiones a la semana).
- Orden de la rutina:** Circuito (selected), Series.
- Tipo de rutina:** Global (selected), Por hemisferios, Por grupo muscular.
- Ejercicios por sesión:** 7 ejercicios.
- Material a descartar:** Bosu, Fitball, Peso libre, Máquinas, Gomas, Poleas, Banco, Balón medicinal, Anillas. Several items have checkboxes checked.
- Continuar:** A green button at the bottom right of the configuration area.

**Figura 47:** diseño de la interfaz de generación de planes de entrenamiento (segundo paso).

### Panel de generación de plan deportivo – selección de ejercicios (tercer paso)

En este último paso, el monitor deportivo selecciona aquellos ejercicios que desea incluir en el plan deportivo.



**Figura 48:** diseño de la interfaz de generación de planes de entrenamiento.

### Panel del cliente – consulta de planes de entrenamiento

En esta vista se muestra la visión que tendría el cliente en nuestra aplicación. En ella se ve una tabla con todos los planes que le han sido generados, y además puede consultar cualquiera de ellos en detalle.

The screenshot shows a web browser window titled "Sport Plans Generator". The address bar displays the URL <http://mywebsite.org>. The main content area has a header "Sport Plans Generator" and a welcome message "Bienvenido usuario" followed by the date "10 de mayo de 2013" and a "Cerrar sesión" link. Below this, a table lists three training plans:

Nombre	Fecha	Nivel	Objetivo	Monitor	Finalizado	Consultar
PlanEntrenamiento1	2/5/2013	Aclimataci	Pérdida de Peso	Monitor1	✓	<input type="button" value="Detalles"/>
PlanEntrenamiento2	6/6/2013	Principiante	Salud	Monitor2	✓	<input type="button" value="Detalles"/>
PlanEntrenamiento3	7/8/2013	Intermedio	Hipertrofia	Monitor2	<input type="radio"/>	<input type="button" value="Detalles"/>

Clicking on "PlanEntrenamiento3" reveals its details:

**PlanEntrenamiento3 - Generado por Monitor2**

Fecha Comienzo: **7/8/2013**  
 Nivel: **Intermedio**  
 Objetivo: **Hipertrofia** En progreso

**Periodo 1 Semana 1**

Sesión Lunes (25 - 20 Repeticiones por minuto)

	Nombre: Flexión de biceps Músculo: Bíceps Equipamiento: Máquinas	Series: 4 Descanso: 90 segundos
	Nombre: Press en banca Músculo: Pectoral Equipamiento: Máquinas	Series: 4 Descanso: 90 segundos

**Figura 49:** diseño de la interfaz de consulta de planes de entrenamiento.

## 8.- Diseño del sistema

### 8.1.- Arquitectura del sistema

A continuación mostramos un diagrama de la arquitectura que hemos diseñado para nuestro sistema. Hemos optado por el estilo arquitectónico basado en la separación en capas, tres capas en este caso (Presentación, Lógica y Datos) muy apropiada para las aplicaciones *Web* y además, en nuestro caso, sencillo de aplicar gracias al módulo del *framework* de desarrollo que hemos usado, *Spring MVC (Model – View – Controller)*.

La *Capa de Presentación* es la encargada de mostrar los datos al usuario de la aplicación mediante cada una de las vistas. Esta capa se comunica directamente con la *Capa de Lógica*, la cual es la encargada de procesar la información recibida de la *Capa de Presentación*, realizar el tratamiento necesario a estos datos, y o bien devolverlos a la *Capa de Presentación* o bien enviarlos a la *Capa de Datos / Recursos*. Esta última capa es la encargada de almacenar los datos recibidos de las capas superiores en algún medio de forma permanente, ya sea una base de datos o ficheros externos.

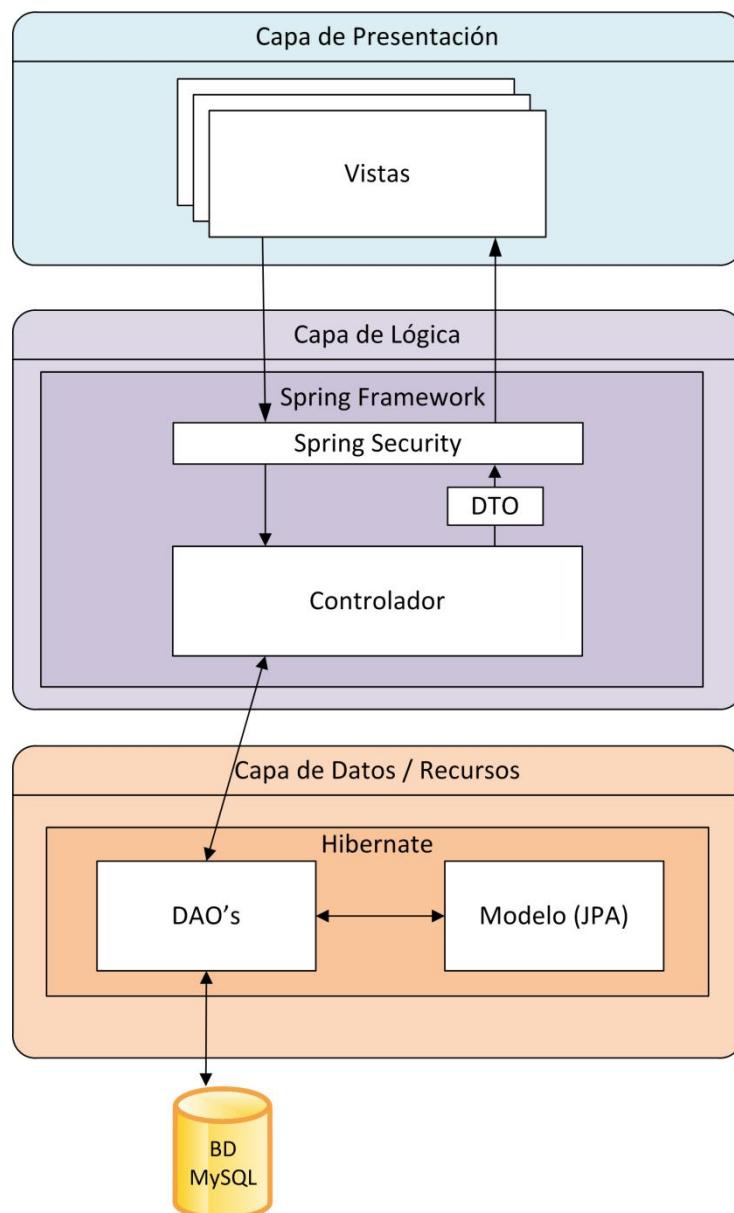


Figura 50: Arquitectura del sistema

## 8.2.- Grafo de navegabilidad

El grafo de navegabilidad es una representación gráfica en la que mostraremos de una forma esquemática los principales conceptos incluidos en el espacio de la información de nuestra aplicación y las interrelaciones que existen entre ellos.

El siguiente grafo se corresponde a la navegabilidad de nuestra aplicación *Web*. Podemos ver que el primer paso para hacer uso de nuestra aplicación es registrarse en el sistema (*Login*). Un usuario que no esté dado de alta en la aplicación no podrá hacer ningún uso de ella.

Una vez el usuario haya realizado el *Login*, tendrá acceso a un panel de control según el tipo de usuario que sea: *Centro Deportivo*, *Monitor Deportivo* o *Cliente*. La funcionalidad ofrecida para cada uno de los distintos usuarios es distinta y se puede consultar observando el siguiente grafo.

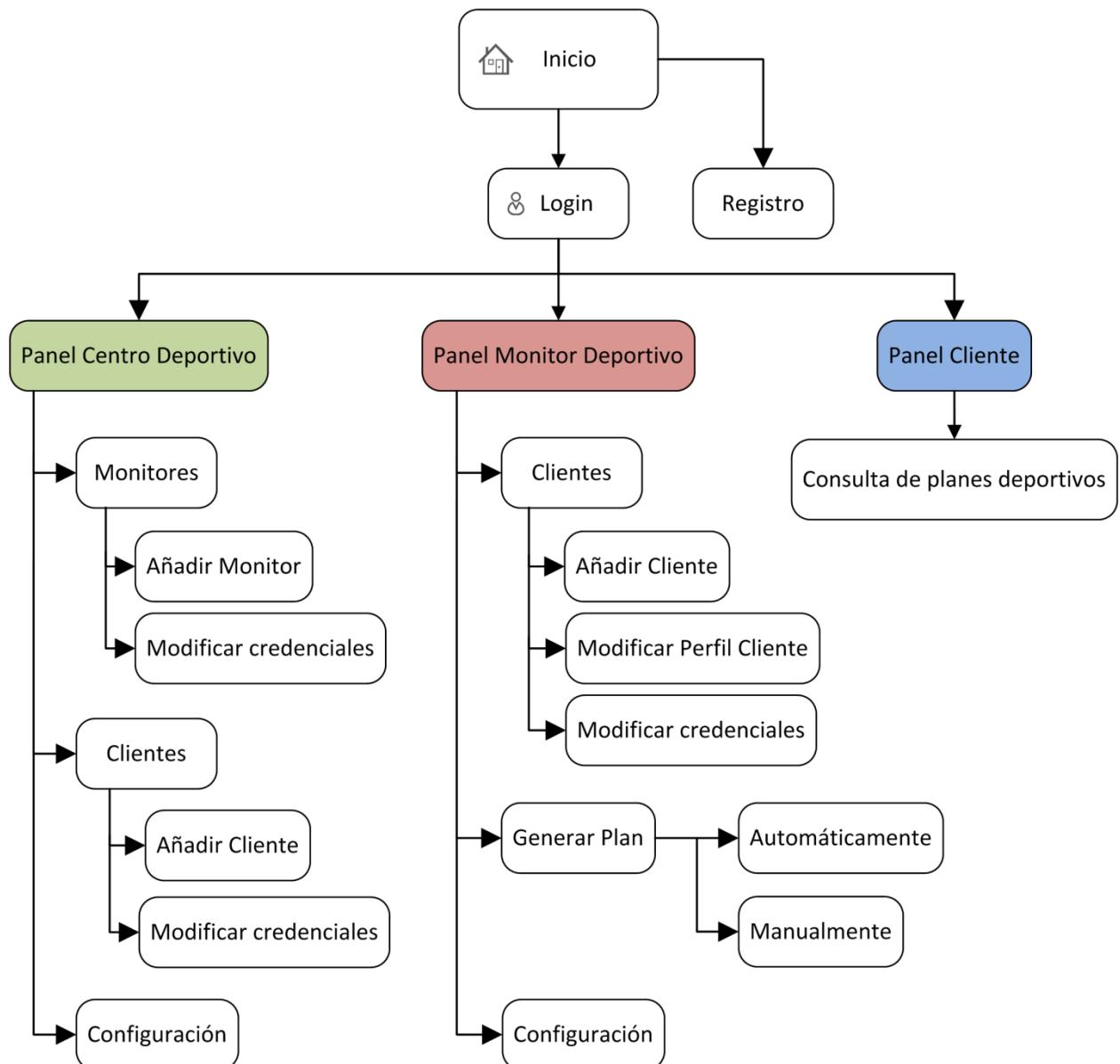


Figura 51: grafo de navegabilidad de la aplicación

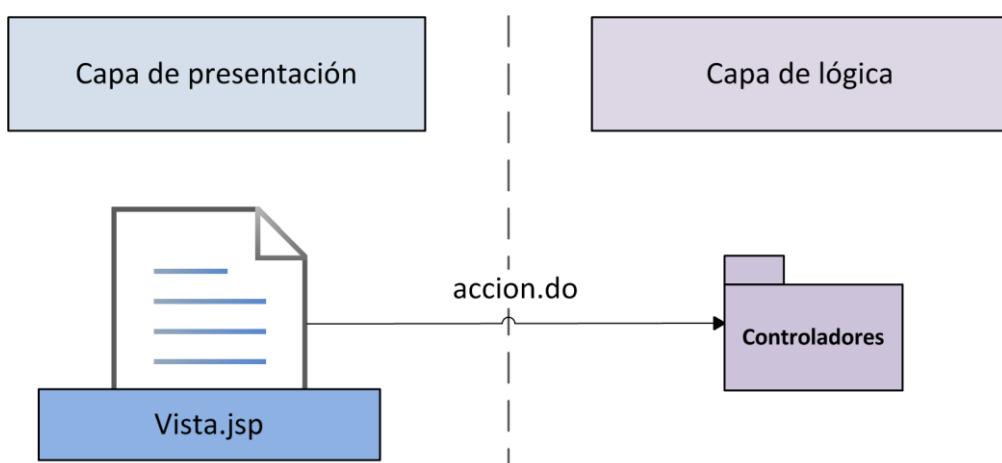
### 8.3.- Diseño detallado

En este apartado comentaremos en detalle el diseño seguido en cada una de las capas que componen la aplicación.

#### 8.3.1. - Capa de presentación

La capa de presentación es la encargada de mostrar los datos al usuario. Con esta capa conseguimos aislar la parte visual de toda la lógica de la aplicación. Así, si en el futuro queremos cambiar por completo el diseño de nuestra aplicación, el proceso no es tan costoso al estar todos los componentes pertenecientes a la vista desacoplados de la lógica de negocio.

Esta capa estará compuesta principalmente de cada una de las vistas de la aplicación. En el siguiente diagrama mostramos el flujo normal que seguirán los usuarios de la aplicación cuando interactúen con ella por medio de las vistas.



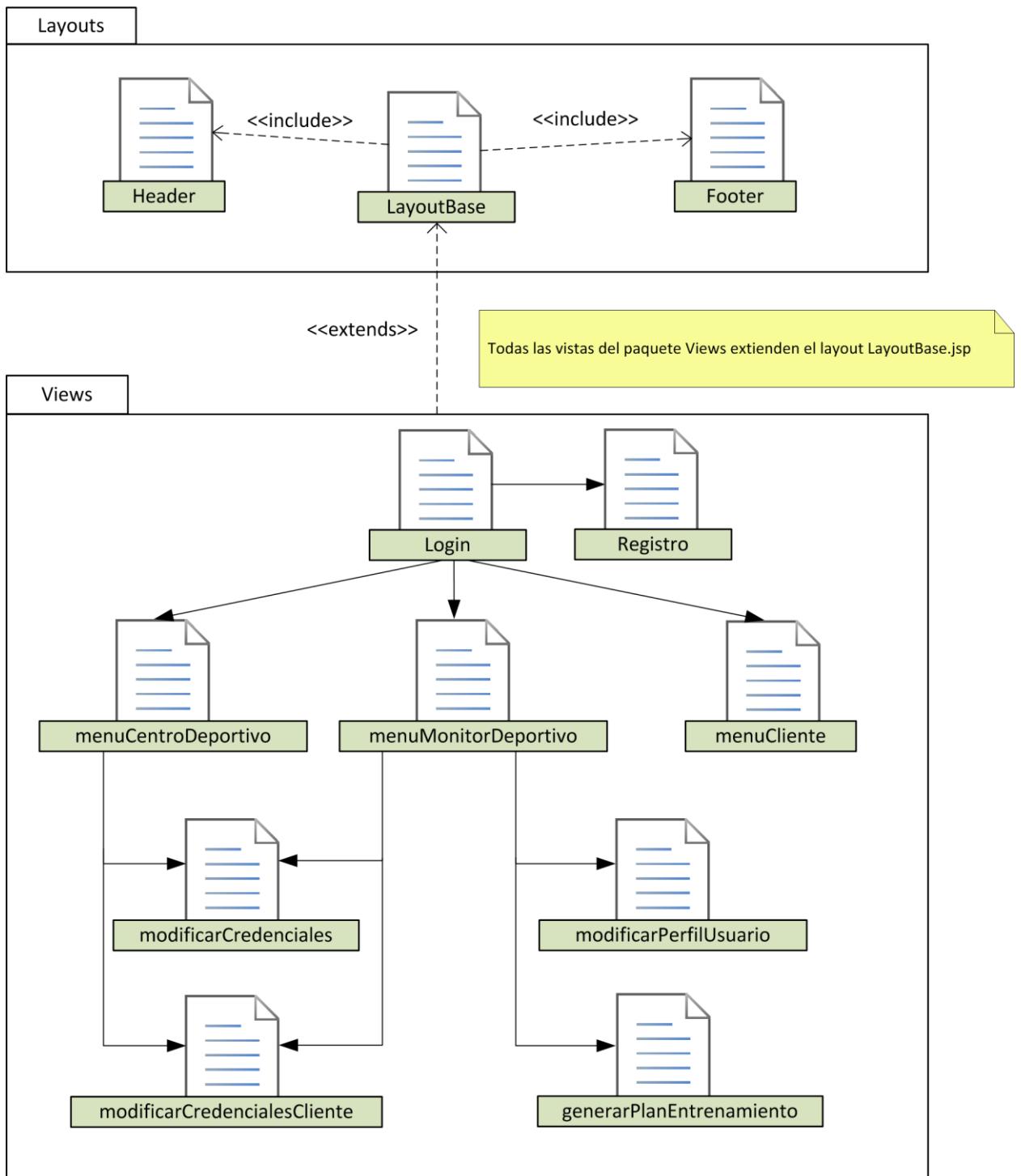
**Figura 52:** Flujo de las peticiones desde la capa de presentación.

Las acciones que el usuario lleve a cabo en la vista producirán un evento que será recogido por el controlador que tenga registrado dicho evento, y se procederá a llevar a cabo la acción correspondiente. El flujo que siguen estas peticiones será comentado en detalle en el siguiente apartado.

Nuestra aplicación estará compuesta por una serie de vistas que han sido diseñadas para cumplir los propios requisitos de la aplicación. Para mantener una homogeneidad en la apariencia de nuestra aplicación y que todas las vistas sigan el mismo diseño, definiremos unas plantillas (*layout*) que serán extendidas por cada una de las vistas. En estas plantillas crearemos la estructura básica que será heredada por cada vista que extienda a la plantilla.

Para nuestra aplicación crearemos una plantilla, compuesta por una cabecera (*header*), el propio contenido que se proporciona en la vista y un pie de página (*footer*). Se ha optado por este diseño para poder cambiar en cualquier momento la estructura de todas las vistas de las páginas de una forma rápida, es decir, modificando sólo la plantilla, sin ser necesario realizar el mismo cambio en cada una de las vistas que compongan la aplicación. Este diseño nos permite a su vez crear otras plantillas distintas por si en el futuro necesitamos incluir algún otro elemento en todas las vistas de la aplicación como por ejemplo un menú lateral.

En el siguiente diagrama mostramos cada una de las vistas que compondrán nuestra capa de presentación. Además se indica la navegabilidad que existirá entre las vistas.



**Figura 53:** esquema de vistas de la aplicación y conexión entre ellas.

Es importante que las vistas de nuestra aplicación sean sencillas de implementar, con una interfaz intuitiva para el usuario y que conformen una experiencia gratificante para éste.

Para cumplir la primera premisa, usaremos un *framework* de desarrollo CSS para diseñar nuestras vistas, *PageOne* [37], ofrecido gratuitamente en la *Web styleshout.com* [38]. Este *framework* nos provee de una serie de componentes para ser usados en nuestras vistas, lo que nos ahorra diseñar nuestros propios elementos desde cero, rebajando así el tiempo necesario para el diseño y codificación de cada una de las vistas. Además conseguiremos que el código de nuestras vistas esté totalmente desacoplado del diseño y la maquetación.

### 8.3.1.1.- Internacionalización

Un aspecto importante a tener en cuenta en la etapa de diseño es pensar en el futuro de la aplicación que estamos diseñando y en el alcance que esta pueda tener. Por ello, si en futuras versiones queremos ofrecer nuestra aplicación en otros idiomas, necesitamos realizar un diseño que nos ayude a poder dar este paso de la manera más fácil posible y sin tener que realizar grandes cambios ni en la arquitectura de la aplicación ni en el código. Es el concepto conocido como *Internacionalización* [42].

El *framework* de desarrollo que vamos a utilizar nos provee de mecanismos para poder realizar esta tarea de una forma rápida y sencilla, facilitando la labor al programador.

Para aplicar esta técnica, el texto mostrado en cada una de las vistas de la aplicación no será introducido directamente en la propia vista. Si no que serán injectados en la vista dinámicamente según el idioma en que sea solicitada la aplicación en el momento de iniciar ésta. De esta forma, el hecho de que nuestra aplicación soporte un nuevo idioma será cuestión de traducir cada uno de los términos que son mostrados en las vistas, sin tener que realizar ningún cambio en el código de las vistas.

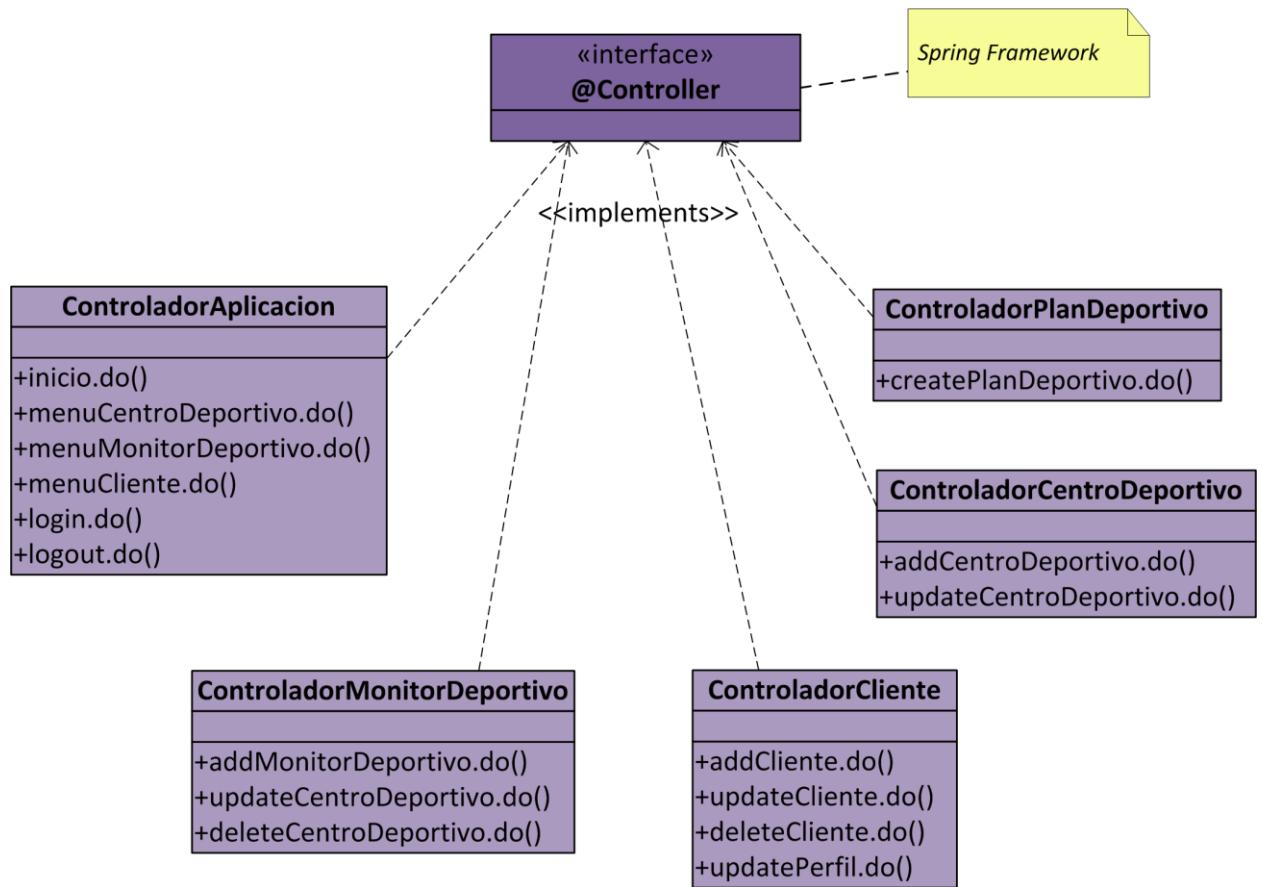
### 8.3.2. - Capa de lógica

La capa de lógica, también conocida como nivel de aplicación, es el encargado de recibir las peticiones del usuario realizadas en la capa de presentación, llamar a las funciones de acceso a los datos si es necesario, realizar el procedimiento o cálculo requerido y devolver la información al nivel de presentación para que sea visualizada por el usuario.

Para manejar las peticiones que el usuario realiza desde la capa de presentación, crearemos una serie de clases cuyo cometido será controlar el flujo de estas peticiones, es decir, serán las encargadas de decidir qué hacer con cada solicitud del usuario en cada momento.

Para realizar esta abstracción en la lógica de nuestra aplicación, *Spring* nos facilita la funcionalidad de la interfaz `@Controller`. Crearemos por tanto distintas clases controladoras, que se ocuparán de las peticiones de los usuarios en los distintos ámbitos de la aplicación. Estas clases implementarán a la interfaz `@Controller` de *Spring*, y cada uno de los métodos que contengan estas clases corresponderán a cada una de las peticiones que el usuario pueda realizar desde la capa de presentación.

En el siguiente diagrama se muestran todos los controladores que compondrán nuestra aplicación. En cada uno de estos controladores mostramos las acciones más importantes que el usuario puede realizar desde la capa de presentación. Cada una de estas funciones realizará las operaciones pertinentes para cumplir las exigencias del cliente.



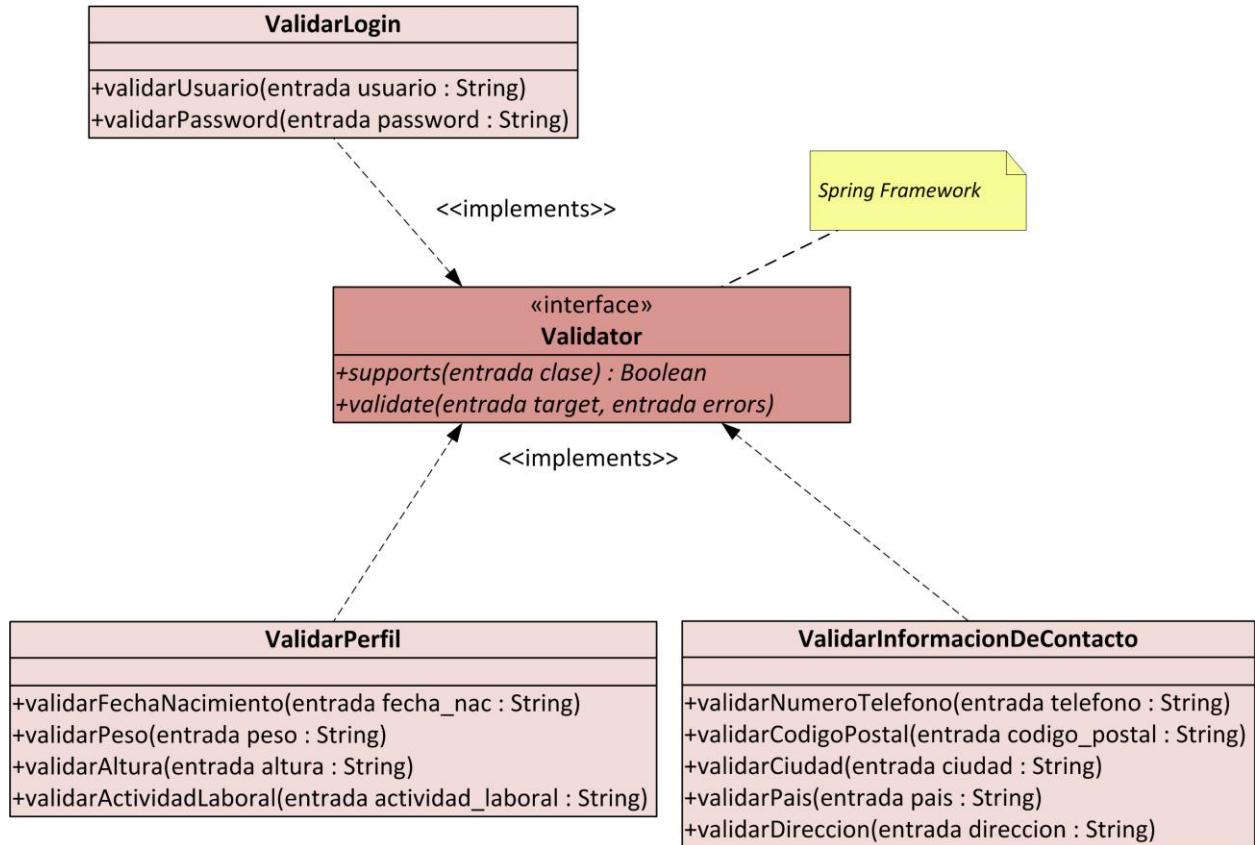
**Figura 54:** controladores pertenecientes a la capa de lógica.

### 8.3.2.1.- Capa de Servicio

Dentro de la capa de lógica se encuentran localizadas todas las clases que ofrecen algún tipo de servicio a otras clases. Componen por tanto la capa de servicio dentro de la capa de lógica.

Tenemos en primer lugar todas las clases ocupadas de la validación de datos. Los servicios de estas clases serán usados por los controladores para comprobar que los datos introducidos por el cliente se adecuan a lo permitido por nuestra aplicación.

Estas clases se servirán de la funcionalidad ofrecida por la interfaz *Validator* de *Spring*. En el siguiente esquema podemos ver el conjunto de validadores que pertenecerán a la capa de servicio y su relación con la interfaz *Validator*.



**Figura 55:** clases validadoras pertenecientes a la capa de servicio

En segundo lugar tenemos las clases que serán las encargadas de realizar otro tipo de operaciones distinto a la validación de datos, como por ejemplo la creación de los planes de entrenamiento y que pertenecen también a la capa de servicio.

La generación de los planes de entrenamiento podrá realizarse de dos maneras posibles: automática y manual. Ambos procedimientos de creación de planes de entrenamiento comparten métodos comunes para la creación de dichos planes, y tienen cada uno de ellos métodos propios. Es aquí donde decidimos aplicar el patrón *Plantilla*.

Crearemos una clase abstracta, *PlanDeportivoManager*, que contendrá una definición abstracta de cada uno de los métodos que conformarán la generación de planes deportivos. Además implementará otros métodos que definirán las dos formas de generar un plan deportivo: generación automática y generación manual. Es aquí donde se aprovechan las ventajas de la aplicación del patrón *Plantilla*. Podremos compartir la funcionalidad de aquellos métodos que ambos procedimientos usen. En el diagrama siguiente se explica con detalle la aplicación de este patrón de diseño.



**Figura 56:** implementación patrón plantilla. Clases generadoras del plan deportivo.

### 8.3.2.2.- DTO – Data Transfer Object

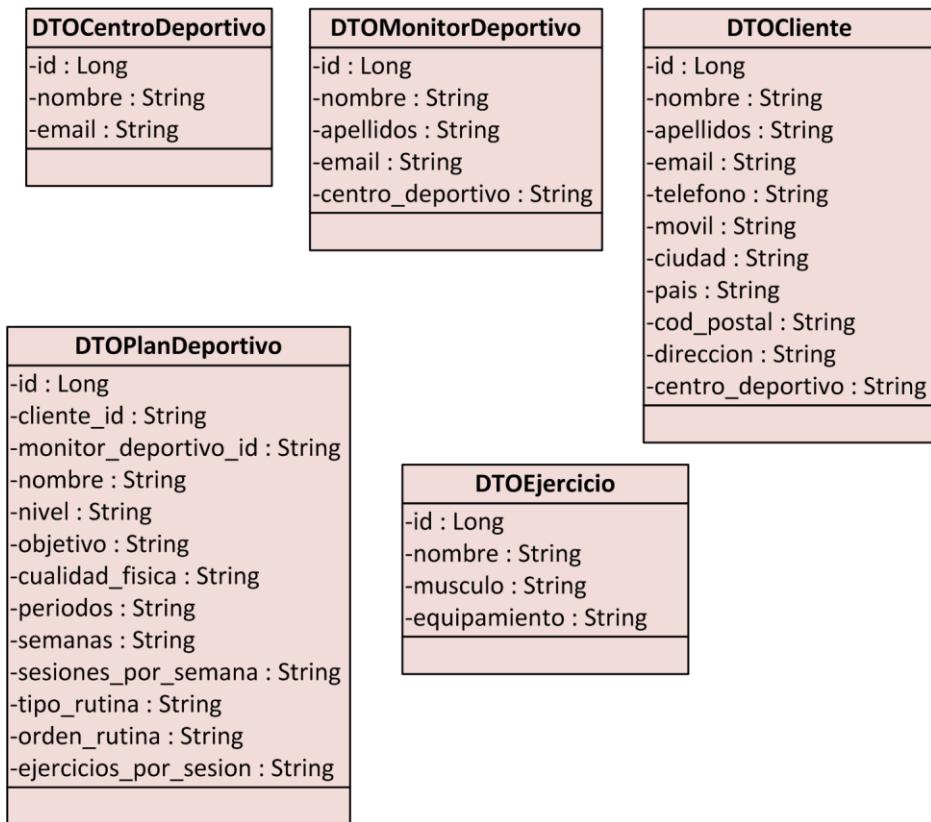
Las clases que conformarán el modelo de datos albergarán una gran cantidad de datos. Estos datos a menudo pueden ser colecciones de elementos que a su vez contendrán en sus atributos otras colecciones.

Estos objetos van a ser obtenidos por la capa de datos del modelo de persistencia, enviados a la capa de lógica para que ésta realice las operaciones pertinentes y finalmente mostrados al usuario en la capa de presentación. El problema recae en que al ser objetos tan pesados, la eficiencia de la aplicación empieza a caer debido al consumo de memoria que se realiza al tener que traspasar entre capas objetos tan pesados.

Como solución a este problema se recurre al patrón de diseño *DTO (Data Transfer Object – Objeto de Transferencia de Datos)*. Este patrón consiste en la encapsulación de objetos en objetos más pequeños y reducidos, en los que almacenaremos la información que realmente queremos mostrar al usuario y que nos es útil de algún modo en la capa de presentación.

Este patrón de diseño es usado también para ocultar información que no desea ser compartida con la capa de presentación para evitar así su posible acceso.

En el siguiente diagrama mostramos las clases *DAO* que hemos creado para nuestra aplicación y que encapsulan la información que realmente necesitamos mostrar en las vistas.



**Figura 57:** Clases *DAO* pertenecientes a la capa de lógica.

### 8.3.3. - Capa de datos

Esta capa será la encargada de proporcionar las funciones de acceso a los datos que nuestra aplicación almacene. Será diseñada como un conjunto de funciones que ofrecen un servicio de almacenamiento abstracto. Así será más fácil acceder a estos datos desde la capa de lógica, y además estaremos desacoplando nuestro código para futuros cambios que puedan producirse.

A continuación mostramos un diagrama en el que aparecen las clases involucradas en esta capa. En ella aparecen las interfaces *DAO* para el tratamiento de los datos de nuestra aplicación y una referencia a sus implementaciones, las cuales se sirven de la funcionalidad de la clase *SessionFactory* de *Spring*. Esta clase es la encargada de las conexiones con la base de datos o los recursos y también del manejo de las transacciones.

También se muestra la abstracción realizada con la interfaz *IDAOComun*. Esta interfaz contiene métodos comunes, como inserciones o búsquedas, por lo tanto los agrupamos en una única interfaz.

Se muestra también el uso que haremos aquí de la utilidad que nos proporciona *Spring* para los métodos que realizan operaciones con la base de datos. Nos referimos a la interfaz *@Transactional*. Nuestros clases *DAO* implementarán esta interfaz para que el *framework* se ocupe de que los métodos de estas clases se ejecuten como transacciones, es decir, que deben ser ejecutados en su totalidad para evitar estados inconsistentes en la base de datos.

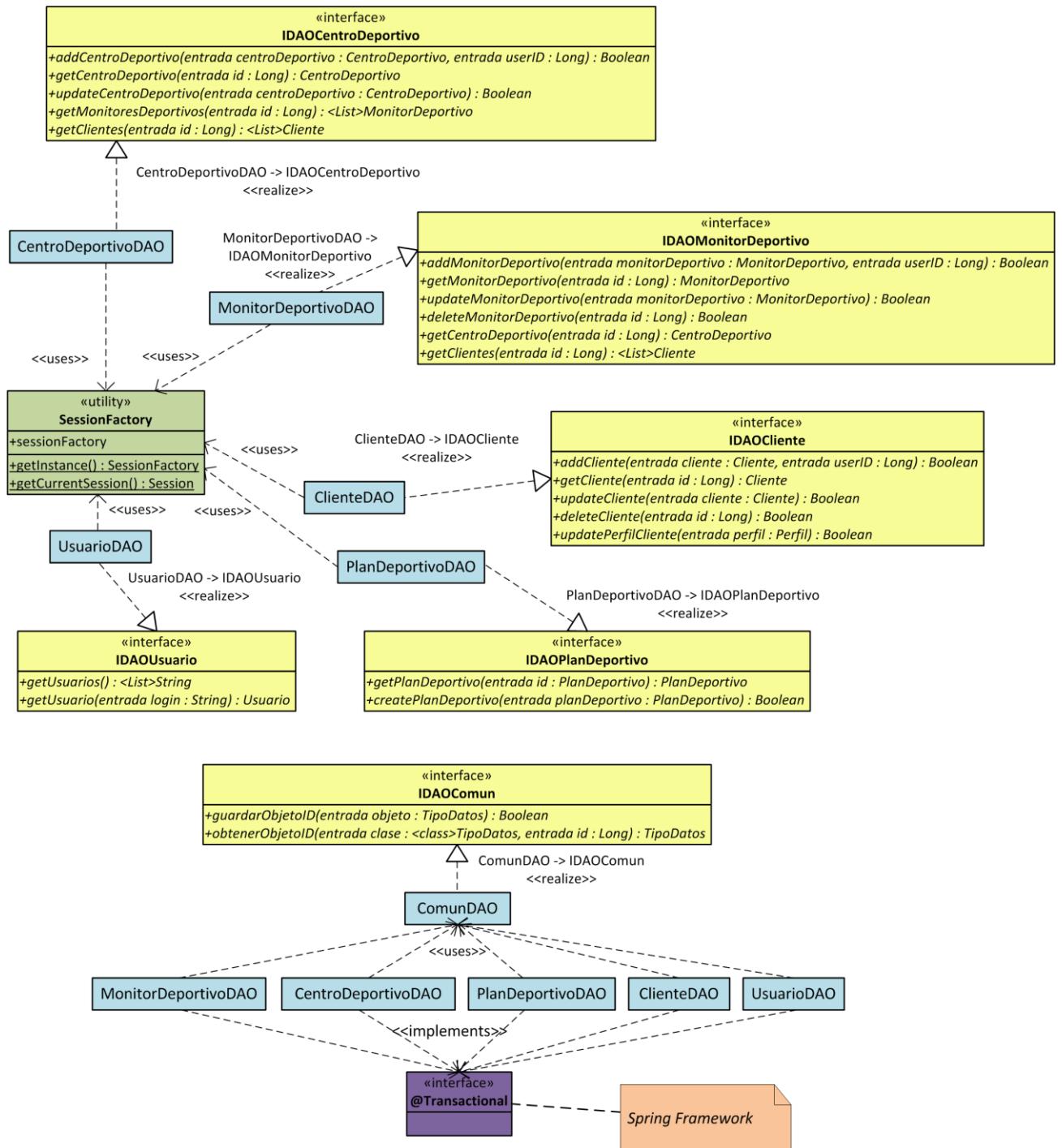


Figura 58: Diseño de la capa de datos.

En esta capa se encuentran también todas las clases que forman el modelo de datos, es decir, toda la información que queremos guardar de cada una de las clases del modelo. Aquí vuelve a tomar protagonismo el *framework* de persistencia que usaremos, ya que nos permitirá crear automáticamente

nuestra base de datos del sistema a partir de las clases que definamos como parte del modelo. Para ello, usaremos la funcionalidad ofrecida por la interfaz *Entity* de *Hibernate*.

En el siguiente diagrama se pueden ver todas las clases que forman el modelo y su relación con la interfaz *Entity*. Todas estas clases contienen los métodos consultores y modificadores que no incluimos por simplicidad en los diagramas. Por simplicidad hemos separado el diagrama en dos partes, una primera con las propias clases del modelo y otra con los tipos enumerados que serán persistidos en la base de datos también.

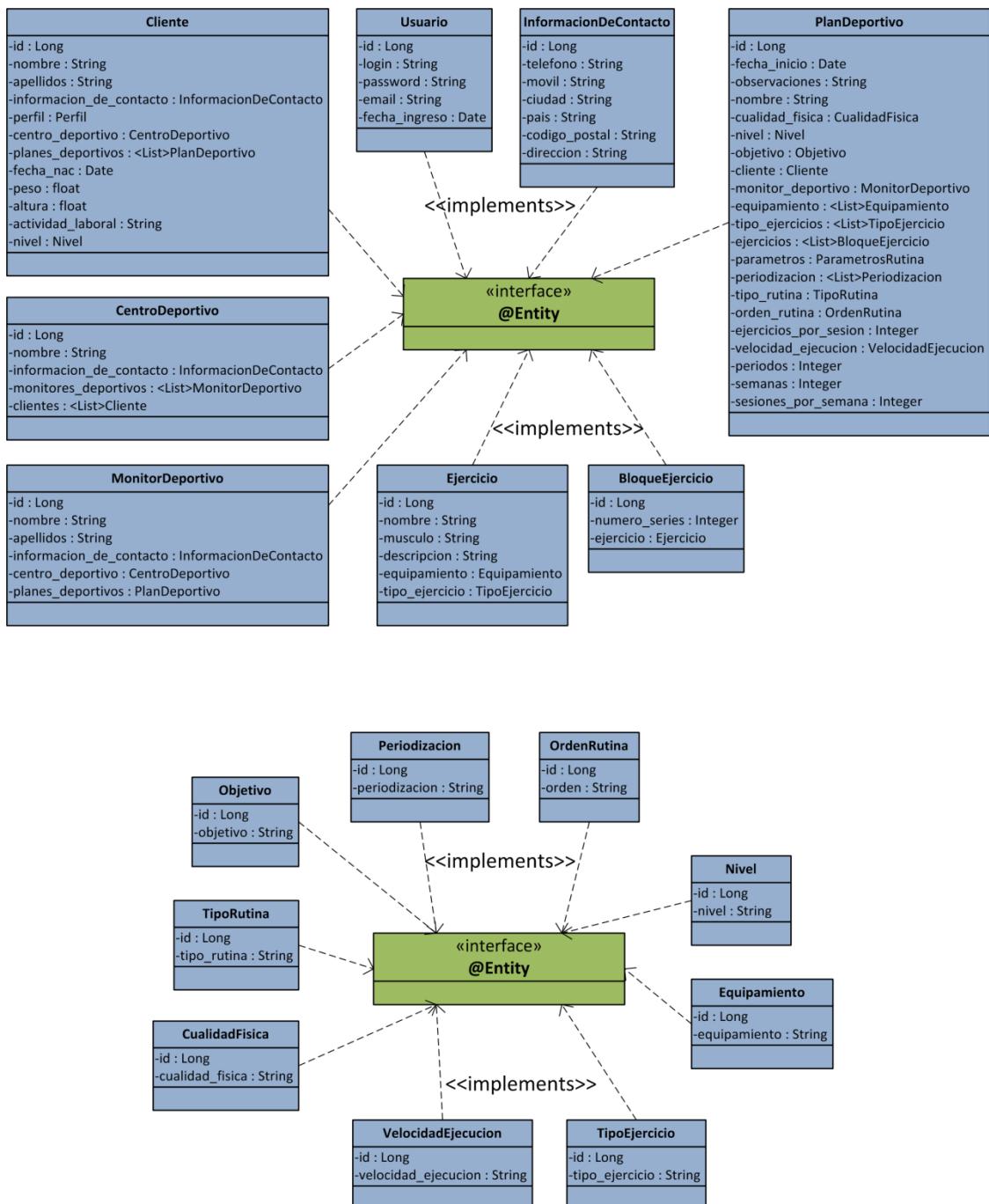


Figura 59: diagrama de clases pertenecientes al modelo de la aplicación

### 8.3.4. - Patrones de diseño utilizados

A continuación, se explicarán los distintos patrones de diseño usados para el diseño del sistema, y donde han sido aplicados.

#### Patrón MVC – Model View Controller

Hemos decidido aplicar este patrón de diseño en nuestra aplicación por varios motivos. Primeramente para aprovechar la clara separación que hace de cada una de las capas. Esto nos permitirá desarrollar cada una de ellas como un módulo independiente, favoreciendo así el desacople de el código, y permitiendo en el futuro la sustitución de una de las capas sin tener que modificar ninguna de las otras dos. Además, teniendo cada capa con su funcionalidad por separado, nos permitirá desarrollar pruebas unitarias a cada componente de cada una de las capas de una manera más sencilla. El mantenimiento de cada una de las capas será también más sencillo y si en el futuro la aplicación crece y se necesitan varias personas trabajando en ella a la vez, es más fácil si tenemos cada una de las capas separadas.

Otro motivo que nos mueve a usar este patrón es la facilidad para utilizar este patrón de diseño que nos proporciona el *framework* que vamos a usar para el desarrollo de nuestra aplicación (*Spring*). Para ello tendremos que hacer uso del modulo *Spring MVC*, el cual provee de los mecanismos necesarios para el uso de este patrón.

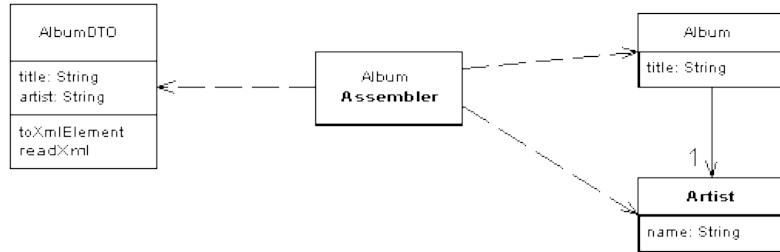
#### Patrón DAO – Data Access Object

Este patrón busca resolver el problema de contar con varias fuentes de datos en una misma aplicación. Por ello, se encapsula la forma de acceder a esta información. Esto nos permite además cambiar en cualquier momento el tipo de fuente de datos que estemos usando y gracias al uso de este patrón, los cambios que será necesario realizar serán mínimos, ya que la forma en la que se accede a la información no está orientada a ningún gestor de la información en concreto. Podemos ver la aplicación de este patrón en nuestra aplicación en el apartado 8.3.3.- *Capa de datos*.

*Spring MVC* nos provee además de varias formas de poder aplicar este patrón en nuestra aplicación de una manera sencilla y cómoda. El gestor de base de datos que usaremos será *Hibernate*, y para ello, *Spring* nos permite usar la solución basada en la superclase *HibernateDAOsupport*. Y para hacer uso de esta implementación es necesario el uso de un objeto *SessionFactory*. La aplicación de este patrón es mostrada en la figura 58 del apartado 8.3.3.- *Capa de datos*.

#### Patrón DTO – Data Transfer Object

Un objeto de transferencia de datos (Data Transfer Object) es usado para encapsular información y enviarla desde un subsistema de la aplicación a otro. Esta estrategia es seguida para reducir la cantidad de datos a transferir entre subsistemas y disminuir así la memoria necesaria para esto además de ahorrarnos el traspaso de información no necesaria. Podemos ver la explicación en detalle de cómo ha sido aplicado este patrón en nuestra aplicación en el apartado 8.3.2.2.- *DTO – Data Transfer Object*.

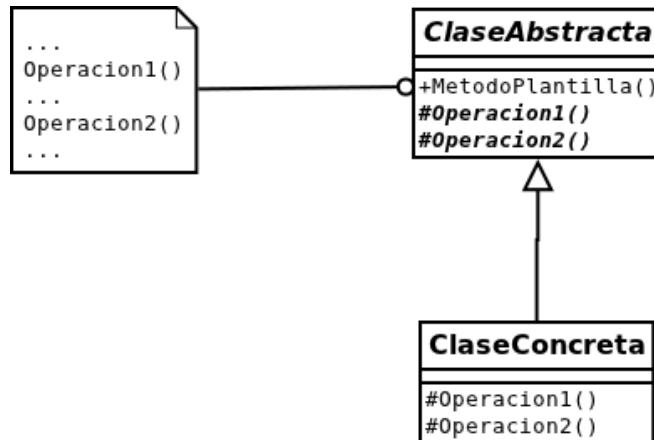


**Figura 60:** ejemplo del uso del patrón Data Transfer Object

En nuestra aplicación haremos uso de este patrón para “aligerar” el tamaño de los objetos que queremos mostrar en la vista. La información de estos objetos que queremos pasar a la *Vista* es un subconjunto de los datos totales que contienen dichos objetos. Por lo tanto este patrón de diseño nos permite reducir el tamaño de estos objetos y además nos possibilita “ocultar” aquella información del objeto que no nos interese mostrar.

#### Patrón Plantilla – (Template Method)

Es un patrón de diseño enmarcado dentro de los llamados patrones de comportamiento, que se caracteriza por la definición, dentro de una operación de una superclase, de los pasos de un algoritmo, de forma que todos o parte de estos pasos son redefinidos en las subclases herederas de la citada superclase [29]. Podemos ver la explicación detallada de cómo ha sido incluido este patrón en nuestra aplicación en el apartado 8.3.2.- *Capa de lógica*.



**Figura 61:** estructura que sigue el patrón plantilla

En nuestra aplicación hemos aplicado este patrón a la hora de generar los planes de entrenamiento. Para realizar dicha tarea ofrecemos varias alternativas, creación del plan automáticamente o de forma manual, y cada una de ellas, sigue un orden de tareas específico. Para ello este patrón nos ayuda a reutilizar código, ya que las tareas a realizar en la creación tanto automática como manual de los planes de entrenamiento son muy similares variando tan sólo alguna de estas tareas y el orden.

## 8.4.- Condicionantes

### 8.4.1.- Puntos de variación

En este apartado comentaremos los posibles cambios a los que puede sufrir nuestra aplicación, como afectarían estas modificaciones a nuestro sistema y como nuestro diseño se adaptaría a estos cambios.

#### Soporte de nuevos idiomas

En el caso de que nuestra aplicación quisiera soportar en el futuro nuevos idiomas, no tendríamos graves problemas, ya que hemos diseñado nuestra aplicación pensando en la internacionalización de la misma, haciendo uso de las funcionalidades que para esto nos provee el *framework* que usaremos en el desarrollo de nuestra aplicación. Por ello, si en el futuro queremos que nuestra aplicación sea mostrada en distintos idiomas, el trabajo a realizar será únicamente de traducción de términos. No será necesario realizar ningún tipo de cambios en el diseño o en la arquitectura de nuestra aplicación.

#### Uso de un gestor de base de datos distinto

En el futuro puede que necesitemos o que nos veamos obligados a usar un gestor de base de datos distinto, lo que podría suponer un gran problema para una aplicación que no hubiera tenido esto en cuenta. En el diseño de nuestra aplicación, hemos hecho uso del patrón *DAO* por lo que creamos una capa intermedia entre el modelo y la propia base de datos permitiéndonos así modificar esta última sin grandes problemas en el resto de la aplicación. Es una de las grandes ventajas que nos proporciona el uso del patrón de diseño *Data Access Object*.

#### Futuros cambios en el diseño

Los diseños de las interfaces de usuario de las aplicaciones *Web* están sujetos a las modas o tendencias que surgen con el tiempo. Por ello es probable que el diseño que hagamos inicialmente para nuestra aplicación sea sometido a cambios en el futuro para adaptar nuestra aplicación a las tendencias actuales y que el usuario sienta que está usando una aplicación moderna y novedosa.

Nuestra aplicación está diseñada para que estos futuros cambios en el diseño de la interfaz de usuario, tengan la menor repercusión en lo que al propio código de las vistas se refiere. Gracias a la decisión a la hora del diseño de usar el *framework* de maquetación de las vistas que usaremos, *PageOne*, nuestro diseño estará completamente desacoplado del código de las vistas, por lo que simplemente habrá que actualizar o cambiar por completo este *framework* y nuestra aplicación seguirá luciendo una moderna interfaz de usuario.

#### Inclusión de nuevos módulos o funcionalidades

Nuestro cliente de la aplicación posee estudios en Dietética y Nutrición, por lo que es probable que en el futuro decida incluir en nuestra aplicación un módulo que permita a partir del perfil del cliente y de sus objetivos, generar una dieta de comidas que se adecue al plan de entrenamiento generado y que ayude al usuario del plan a conseguir sus objetivos.

Gracias al diseño propuesto y al *framework* de desarrollo elegido, la inclusión de un nuevo módulo en nuestro sistema no sería muy problemático ya que el diseño en capas que hemos seguido nos proporcionará un código poco acoplado, las clases estarán separadas por módulos y funcionalidad. Por lo que la adición de nuevos módulos no supondría gran problema aparte del diseño e implementación del mismo.

#### **8.4.2.- Puntos de evolución**

En este apartado comentaremos como el diseño elegido se adaptaría a una serie de posibles mejoras y nuevas funcionalidades que añadiríamos a nuestro sistema en futuras versiones.

##### **Creación de ejercicios propios**

Los ejercicios que actualmente se incluyen en los planes de entrenamiento son ejercicios estándar, que cubren todos los grupos musculares de forma general. Puede que algún centro deportivo le interese crear sus propios ejercicios para distinguirse de otros centros que usen esta misma aplicación y generar así planes exclusivos. Sería interesante por lo tanto dar la posibilidad a los centros deportivos de crear sus propios ejercicios y que estos se incluyan en los planes deportivos que sus monitores generen.

Para ofrecer esta característica, el diseño que hemos aplicado a nuestro sistema nos da la posibilidad de implementar esta nueva funcionalidad fácilmente. En primera instancia, todos los ejercicios incluidos en nuestra base de datos pertenecerán a un mismo usuario, el usuario *Administrador*. Para ofrecer esta nueva funcionalidad, únicamente será necesario el formulario de creación de nuevos ejercicios por parte del centro deportivo, ya que las clases encargadas de la generación del plan buscarán los ejercicios pertenecientes al usuario *Administrador* además de los ejercicios propios del centro deportivo en el que se esté generando el plan.

##### **Explicación visual de los ejercicios**

En la primera versión de la aplicación, el ejercicio será explicado por medio de imágenes que mostrarán paso a paso la correcta ejecución del ejercicio. En futuras versiones sería interesante incluir un pequeño video de la ejecución del ejercicio, por si el usuario del plan de entrenamiento sigue teniendo dudas de la correcta forma de realizar el ejercicio después de ver las imágenes, pueda consultar el video explicativo y resolver así sus dudas.

Para cumplir esta futura mejora, el modelo de nuestra aplicación está diseñado de manera que la adición de nuevos atributos a las clases del modelo tenga la menor repercusión posible en el resto de la aplicación. Gracias al *framework* que aplicaremos en la capa de datos, *Hibernate*, esta nueva característica no supondrá cambios en el diseño, y sólo supondrá cambios en las vistas de nuestra aplicación para mostrar dichos videos.

##### **Aplicación móvil**

Sería muy interesante que en el futuro los usuarios de los planes de entrenamiento pudieran acceder a los planes que están realizando en la actualidad y consultarlos en directo. Así, un usuario durante su entrenamiento en su centro deportivo, podría acceder desde su dispositivo móvil a través de la aplicación y consultar los ejercicios que debe realizar en esa sesión.

Gracias al diseño en capas de nuestra aplicación, satisfacer este requisito no supondría ningún cambio en nuestro sistema, ya que para crear una aplicación móvil únicamente necesitaríamos añadir una capa de presentación completamente distinta. Las capas de lógica y datos quedarían totalmente intactas y podrían ser usadas por esta nueva capa creada especialmente para la aplicación de dispositivos móviles.

### **Uso de un framework en la capa Vista**

Nuestra capa de presentación está diseñada como un conjunto de vistas. Estas vistas serán codificadas manualmente usando para ello lenguajes como *JSP* o *JSTL*. Cuando la aplicación comience a crecer, el código que controle la funcionalidad en el cliente empezará a ser poco mantenible. Por lo que el uso de un *framework* para la capa *Vista*, como por ejemplo *GWT (Google Web Toolkit)* nos haría tener un código más mantenible, optimizado, más sencillo de programar, ya que el lenguaje puede ser el mismo que el que usaremos para el resto de la aplicación (Java) y soportado por todos los navegadores.

Para realizar esta mejora, al igual que el punto anterior, tendríamos que modificar la capa de presentación por completo. Esto supone un cambio importante, pero gracias a la separación en capas que implementaremos en nuestra aplicación, los cambios a realizar en las otras dos capas se reducen en teoría a cero.

## **8.5.- Modelo de datos**

En este apartado procederemos al diseño del esquema que seguirá la base de datos de nuestra aplicación teniendo en cuenta lo realizado anteriormente en el modelo estático del sistema y satisfaciendo los requisitos de almacenamiento de datos [\[51\]](#) [\[52\]](#).

Para el diseño de este esquema, se ha usado un perfil UML y se seguirá una serie de reglas que nos permitirán traducir el diagrama estático del sistema en el modelo de datos con pocos pasos [\[51\]](#).

La regla general a seguir es:

- Las clases se traducen a tablas.
- Los atributos se traducen en columnas.

A cada tabla creada se le añadirá un campo que actuará como identificador único de cada registro y podrá ser usado para referenciar ese registro desde otras tablas.

Para las relaciones entre clases, seguiremos la siguiente estrategia:

- Relaciones uno a uno y uno a muchos se relacionarán con clave ajena <>FK<>.
- Relaciones muchos a muchos se relacionarán creando una nueva tabla asociativa.

Para las especializaciones existen varias alternativas:

- Una tabla para todas las clases.
- Una tabla para cada clase concreta.
- Una tabla por cada clase.

Para este caso hemos optado por la última opción, creando una tabla por cada una de las clases implicadas en la especialización, ya que conceptualmente son entidades distintas y además estamos favoreciendo el desacoplamiento del código.

Teniendo en cuenta cada una de estas restricciones, el modelo de datos queda de la siguiente forma:

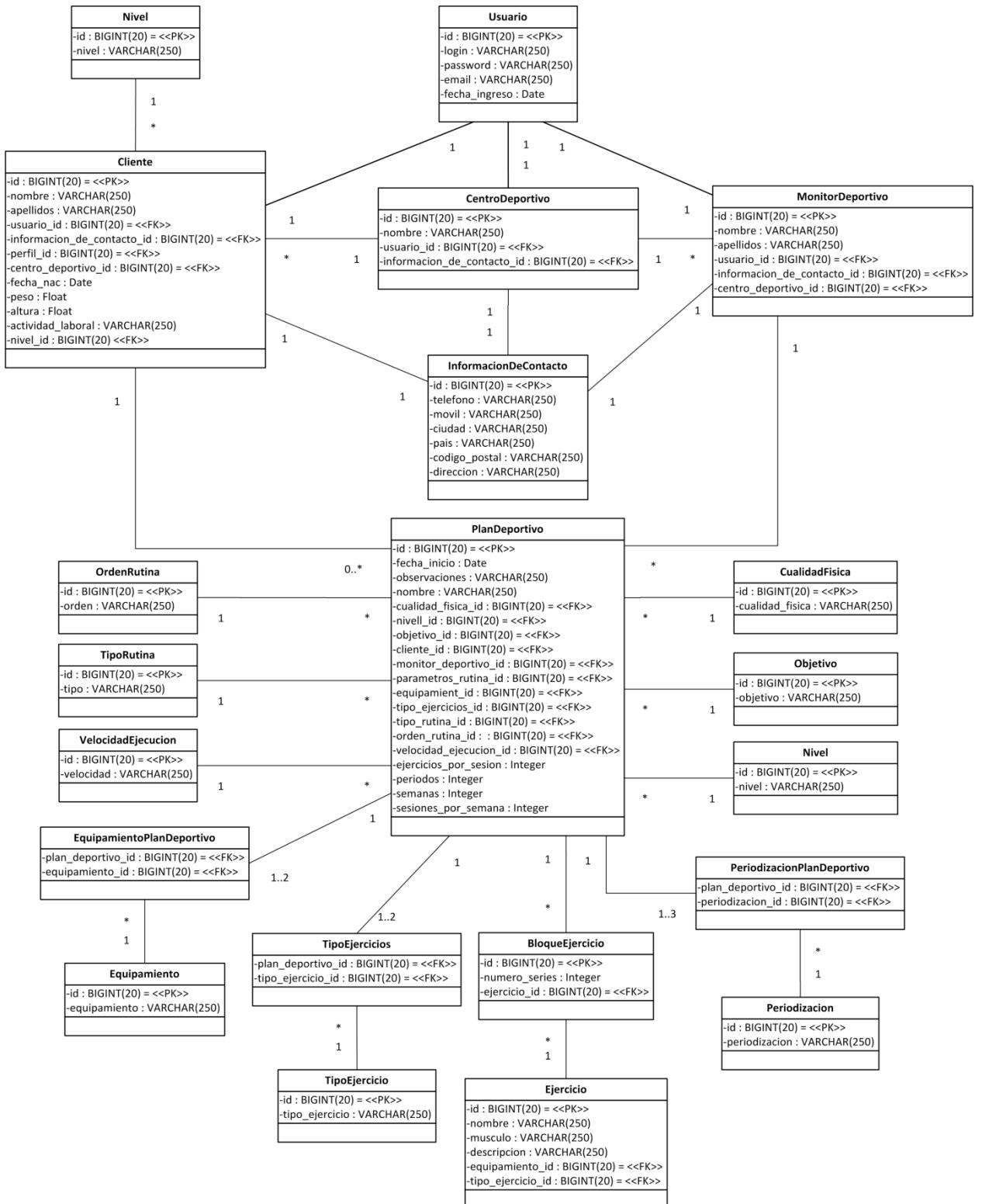


Figura 62: Modelo de datos.

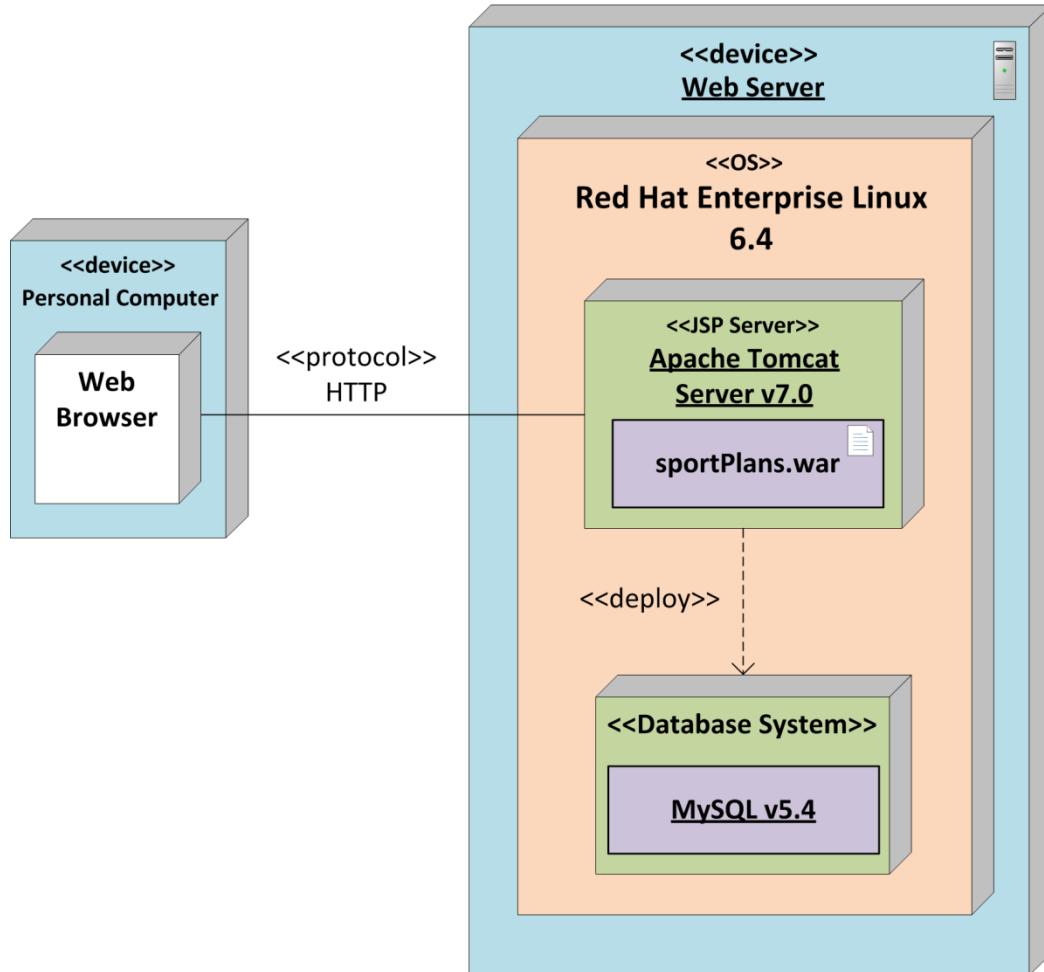
## 8.6.- Diagrama de despliegue

El Diagrama de Despliegue es un tipo de diagrama del *Lenguaje Unificado de Modelado (UML)* que se utiliza para modelar el hardware utilizado en las implementaciones de sistemas y las relaciones entre sus componentes. [34] [35]

A continuación mostramos el diagrama de despliegue de nuestra aplicación. En el cual se pueden diferenciar tres bloques principales. El bloque que se encuentra más a la izquierda en el diagrama se trata del equipo desde el que el cliente se conectará a nuestra aplicación, y lo hará usando un navegador *Web* a través del protocolo *HTTP*.

El bloque situado a la derecha del anterior se refiere al servidor donde correrá la aplicación. El sistema operativo especificado en el diagrama es el que se usará durante el desarrollo, aunque podrá usarse cualquier otro en el futuro gracias a la portabilidad que tienen las tecnologías usadas en la aplicación.

El último bloque, situado en la parte baja del diagrama, se trata del sistema de base de datos con el que se conectará la aplicación para almacenar toda la información necesaria.



**Figura 63:** diagrama de despliegue

## 9.- Implementación

En este capítulo mostraremos aquellos aspectos más relevantes que se han obtenido del proceso de implementación de la aplicación. Comentaremos también el entorno de desarrollo usado y la configuración realizada para su puesta a punto.

### 9.1.- Entorno de desarrollo

A continuación mostraremos cada uno de los apartados que componen todo el entorno de desarrollo que hemos usado para el desarrollo de nuestra aplicación.

Hardware	Detalles
Intel Core	i3 M350 2.27GHz 4GB RAM

**Cuadro 1 :** Entorno de desarrollo. Hardware

Sistema Operativo
Windows 7 Home Edition 64 bits

**Cuadro 2 :** Entorno de desarrollo. Sistema Operativo

Servidor Web	
Nombre	Versión
Apache HTTP Server	2.0
Tomcat	7.0

**Cuadro 3 :** Entorno de desarrollo. Servidor Web

Base de datos		
Nombre	Versión	Comentarios
MySQL Server	5.4	Sistema gestor de BD.
Heidi SQL	7.0.0.4053	Aplicación para interaccionar con las bases de datos de MySQL Server mediante consultas SQL.

**Cuadro 4 :** Entorno de desarrollo. Servidor Base de datos

Lenguajes de programación		
Nombre	Versión	Comentarios
JSP	2.1	Lenguaje de procesamiento en servidor.
HTML	5	Lenguaje de marcado que se interpreta en el cliente.
CSS	3	Hojas de estilo para maquetar el contenido de las páginas.
JavaScript	1.9	Lenguaje de script usado en el procesamiento en el cliente.
JAVA		Lenguaje de programación orientado a objetos.

**Cuadro 5 :** Entorno de desarrollo. Lenguajes de programación

Entorno de desarrollo		
Nombre	Versión	Comentarios
Eclipse	SpringSource	Plugin creado por <i>Spring</i> para el desarrollo de aplicaciones que usan este <i>framework</i> .
Notepad++	5.9	Procesador de textos para la rápida edición de hojas de estilos.

**Cuadro 6 :** Entorno de desarrollo. Entorno de programación

Frameworks	
Nombre	Versión
Spring Framework	3.0.5
Hibernate	3.6.10
PageOne CSS Framework	1.0

**Cuadro 7 :** Frameworks usados. Entorno de programación

Pruebas	
Nombre	Versión
Selenium	2.26.0
JMeter	2.9

**Cuadro 8 :** Fase de pruebas. Entorno de programación

Métricas del código fuente	
Nombre	Versión
SonarQube	3.6.1

**Cuadro 9 :** Métricas del código fuente. Entorno de programación

Control de versiones	
Nombre	Versión
Git	1.8.4

**Cuadro 10 :** Control de versiones. Entorno de programación

Navegadores web	
Nombre	Versión
Google Chrome	27.0
Google Chrome Developer Tools	27.0
Mozilla Firefox	10.0.2
Internet Explorer	7

**Cuadro 11 :** Entorno de desarrollo. Navegadores Web

Herramientas para la documentación		
Nombre	Versión	Comentarios
Microsoft Word	2007	Procesador de textos usado para la redacción y maquetación de la documentación del proyecto.
Microsoft Visio	2010	Aplicación usada para la edición de diagramas.
REM	1.2.2	Programa para la edición de los apartados de análisis ylicitación de requisitos.

**Cuadro 12 :** Entorno de desarrollo. Herramientas para la documentación

## 9.2.- Aspectos relevantes de la implementación

Como resumen del proceso de implementación de la aplicación, expondremos los aspectos más importantes y destacados que hemos tenido que resolver a la hora de crear la aplicación *Web*.

Para ordenar un poco la información, comentaremos cada una de las capas que componen la aplicación por separado.

### 9.2.1.- Capa de datos

Como comentamos en el capítulo de diseño, esta capa es la encargada de guardar el estado de nuestra aplicación. Para ello, almacenará toda la información en base de datos y poder así acceder a ella en el futuro.

Para la persistencia de los objetos en la base de datos usaremos la herramienta *Hibernate*. Nos ahorrará un gran esfuerzo a la hora de la creación de la base de datos, ya que con esta herramienta todo el proceso de la creación de las tablas de las bases de datos es automático. Además es de resaltar que *Spring* se integra bastante bien con *Hibernate*, por lo que su uso es bastante sencillo.

Para realizar el mapeo de las clases que conformarán el modelo de datos de la aplicación, *Hibernate* nos provee de un mecanismo que nos facilita mucho la tarea. Se trata de las anotaciones. Con el uso de las anotaciones, nos ahorramos tener que definir en ficheros de configuración cada una de las tablas que conformarán nuestra base de datos. Por el contrario, simplemente necesitamos “anotar” cada una de estas clases que conformarán nuestro modelo con las anotaciones correspondientes e *Hibernate* se encargará de hacer todo el trabajo.

A continuación mostraremos una de las clases del modelo a modo de ejemplo del uso de las anotaciones de *Hibernate*.

```

18 @Entity
19 @Table(name = Tabla.SPORT_CENTRE)
20 @Inheritance(strategy = InheritanceType.JOINED)
21 public class SportCentre extends User {
22
23     private static final long serialVersionUID = 1L;
24
25     // Constantes para referencias externas a atributos con hibernate.
26     public static final String A_NAME = "name";
27
28
29     private String name;
30
31     @OneToOne
32     @JoinColumn(name = "contactInfo_id")
33     @Cascade(CascadeType.SAVE_UPDATE)
34     private ContactInfo contactInfo;
35
36     @OneToMany(mappedBy = "sportCenter")
37     private List<SportTrainer> sportTrainers;
38
39     @OneToMany(mappedBy = "customerSportCenter")
40     private List<Customer> customers;

```

**Figura 64:** clase con anotaciones *Hibernate*

Una parte bastante grande de esta capa serán las consultas que ataqueen a la base de datos y que irán en esta capa, concretamente en las implementaciones *DAO* que realicemos. Es importante reseñar que para estas consultas usaremos otra funcionalidad que ofrece *Spring*, la *API Criteria*. Con esta *API* nuestras consultas serán realizadas de una forma más sencilla y la consulta resultante será mucho más fácil de interpretar, ya que el lenguaje usado por esta *API* es más fácil de entender que las típicas consultas *SQL*.

A continuación mostramos un ejemplo de una consulta realizada con esta *API*:

```

252     public MuscleType getMuscleType(String type, String muscle) {
253         Criteria crit = sessionFactory.getCurrentSession().createCriteria(MuscleType.class);
254         crit.add(Restrictions.eq(MuscleType.A_TYPE, type));
255         crit.createAlias(MuscleType.A_MUSCLE, "muscle");
256         crit.add(Restrictions.eq("muscle.muscle", muscle));
257
258         MuscleType mType = (MuscleType) crit.uniqueResult();
259         return mType;
260     }
261

```

**Figura 65:** método de consulta usando la *API Criteria*

### 9.2.2.- Capa de lógica

La capa de lógica, también conocida como nivel de aplicación, es el encargado de recibir las entradas del usuario, de llamar a las funciones de acceso a los datos si es necesario, realizar el procedimiento o cálculo requerido y devolver la información al nivel de presentación para que sea visualizada en el usuario.

En esta capa será donde hagamos un mayor uso del *framework* de desarrollo que vamos a usar en el proyecto que nos ocupa.

Usaremos la funcionalidad ofrecida por los “*Controller*” de *Spring*. Con ellos manejaremos las peticiones realizadas por el usuario desde la capa de presentación, y las dirigiremos hacia el servicio adecuado de nuestra aplicación. Ya sea devolviendo datos desde la capa de lógica, o realizando algún cálculo con estos datos para luego devolverlos a la vista.

Para definir qué clases van a realizar la función de controladores y que el *framework* tenga constancia de ello, usaremos de nuevo la funcionalidad ofrecida por las anotaciones. Por medio de estas anotaciones, nos ahorraremos el arduo trabajo de la definición de propiedades en los archivos de configuración *.xml*. Usaremos por tanto la anotación ofrecida por *Spring*, `@Controller`. Mediante la anotación `@RequestMapping` controlaremos cada una de las peticiones del usuario y definiremos que hacer en cada caso.

A continuación mostramos un ejemplo de una clase que realiza la función de controlador.

```

24 @Controller
25 public class ApplicationController extends GenericController {
26
27     @RequestMapping(value = "/welcome.do", method = RequestMethod.GET)
28     public String printWelcome(ModelMap model) {
29         String res = null;
30         User user = (User)SecurityContextHolder.getContext().getAuthentication().getPrincipal();
31         Collection<GrantedAuthority> authorities = user.getAuthorities();
32         for(GrantedAuthority g : authorities){
33             if(g.getAuthority().equals("ROLE_ADMIN")){
34                 res = "redirect:/menuAdmin.do";
35             }
36             else if(g.getAuthority().equals("ROLE_SPORT_CENTRE")){
37                 res = "redirect:/menuSportCentre.do";
38             }
39             else if(g.getAuthority().equals("ROLE_SPORT_TRAINER")){
40                 res = "redirect:/menuSportTrainer.do";
41             }
42             else if(g.getAuthority().equals("ROLE_CUSTOMER")){
43                 res = "redirect:/menuCustomer.do";
44             }
45         }
46         return res;
47     }

```

**Figura 66:** clase anotada con la anotación `@Controller`

En esta capa incluiremos también otra de las funcionalidades que nos brinda *Spring*. Se trata de la capa de seguridad ofrecida por el módulo *Spring Security*. Con este módulo realizaremos todo el control de usuarios y sus permisos. Esta capa nos dará además una seguridad extra, ya que implementa por defecto medidas de seguridad tales como ataques por inyección *SQL*, ahorrándonos así trabajo a la hora del desarrollo.

Para configurar este módulo, necesitaremos añadir un nuevo fichero de configuración, “*spring-security.xml*”, que será referenciado desde el fichero “*web.xml*” de la aplicación.

En este fichero de configuración definiremos las *urls* a las que podrá acceder cada tipo de usuario. En la siguiente imagen podemos ver una parte de este fichero de configuración donde se muestran dichas *urls* y los usuarios que pueden acceder a ellas.

```

25<security:http auto-config="true">
26    <security:intercept-url pattern="/welcome*" access="ROLE_SPORT_CENTRE,ROLE_ADMIN,ROLE_SPORT_TRAINER,ROLE_CUSTOMER" />
27    <security:intercept-url pattern="/private/*" access="ROLE_SPORT_CENTRE,ROLE_ADMIN,ROLE_SPORT_TRAINER,ROLE_CUSTOMER" />
28    <security:intercept-url pattern="/public/*" filters="none" />
29    <security:form-login login-page="/Login.do" default-target-url="/welcome.do"
30        authentication-failure-url="/LoginFailed.do" />
31    <security:logout logout-success-url="/Logout.do" />
32</security:http>

```

**Figura 67:** definición de urls en el fichero de configuración de Spring Security

Además de toda la funcionalidad general de la aplicación, esta capa será también la encargada de la creación de los planes de entrenamiento deportivos. Se albergará aquí por tanto toda la lógica que posibilitará la generación de planes de entrenamiento adecuados a las necesidades y objetivos de los clientes.

### 9.2.3.- Capa de presentación

La capa de presentación es la encargada de mostrar los datos al usuario. Con esta capa conseguimos aislar toda la lógica de la aplicación de la parte visual. Así, si en el futuro queremos cambiar por completo el diseño de nuestra aplicación, el proceso no es tan costoso al estar toda la lógica de negocio desacoplada de la vista.

Nuestra aplicación se trata de una aplicación *Web*, pero en el caso de que en el futuro queramos desarrollar una aplicación nativa para móviles usando algunos de los sistemas actuales como Android o iOS, únicamente tenemos que cambiar la capa de presentación pudiendo usar las capas de lógica y datos sin tener que realizar cambios.

Para la gestión de las vistas de nuestra aplicación, usaremos la funcionalidad que nos brinda *Spring Framework* por medio de la clase *ViewResolver*. Dicha clase será la encargada de elegir qué vista mostrar a partir de un archivo *.xml* de configuración en el cual definiremos cada una de las vistas que compondrán nuestra aplicación.

A continuación mostramos una parte de este archivo de configuración de las vistas en el que podemos ver la definición de varias vistas de la aplicación.

```

7<tiles-definitions>
8    <!-- Login -->
9    <definition name="Login" extends="Login-Layout">
10       <put-attribute name="titulo" value="etiqueta.titulo.bienvenido" />
11       <put-attribute name="contenido" value="/WEB-INF/pages/public/Login.jsp" />
12       <put-attribute name="idPagina" value="Login" type="string"/>
13   </definition>
14
15   <!-- Form User -->
16   <definition name="formSportCentre" extends="base-layout">
17       <put-attribute name="titulo" value="etiqueta.titulo.nuevoSportCentre" />
18       <put-attribute name="contenido" value="/WEB-INF/pages/public/formSportCentre.jsp" />
19       <put-attribute name="idPagina" value="formSportCentre" type="string"/>
20   </definition>

```

**Figura 68:** definición de vistas de la aplicación en fichero *.xml*

Para las vistas usaremos *JSP (JavaServer Pages)*. Nos serviremos también de la librería de etiquetas *JSTL* de Oracle, la cual nos ofrece una amplia funcionalidad a la hora de crear nuestras vistas.

Al código *HTML* generado, se le aplicará estilo usando hojas de estilo en cascada (*CSS*). Para ahorrar tiempo en la generación de estas hojas de estilo, nos serviremos de un *framework CSS*, *PageOne* [37], ofrecido gratuitamente en la *Web styleshout.com* [38].

Para dar funcionalidad a nuestras vistas, usaremos scripts usando el lenguaje soportado por la mayoría de los navegadores actuales, *JavaScript*. Para facilitar la codificación de la funcionalidad de las vistas usando este lenguaje, usaremos la librería *jQuery*.

Un aspecto importante a resaltar en las vistas de nuestra aplicación es el uso de *AJAX*. *AJAX* es el acrónimo de *Asynchronous JavaScript And XML* (*JavaScript* asíncrono y *XML*) [50]. Es una técnica de desarrollo *Web* para crear aplicaciones interactivas. Se trata de realizar peticiones asíncronas al servidor sin necesidad de recargar por completo la página en el cliente. Con esto se consigue una experiencia mucho más rica y agradable para el usuario.

Para aplicar este concepto en nuestra aplicación, usaremos la función que nos ofrece la librería *jQuery*: *getJSON()*. Con esta función haremos una petición concreta al servidor de manera asíncrona y éste nos responderá usando la anotación *JSON* [52]. *JSON* es un formato ligero usado para el intercambio de datos. En la siguiente imagen mostramos un ejemplo de uso de esta función para obtener los clientes pertenecientes a un centro deportivo de manera asíncrona. De esta manera, logramos mejorar el rendimiento de nuestra aplicación *Web*, ya que cuando se hace la petición para mostrar el *Dashboard* de un centro deportivo, estamos devolviendo la menor cantidad de información posible al cliente. Y una vez que la página *Web* ha sido cargada y mostrada por el navegador, se realiza la petición asíncrona usando la función comentada anteriormente.

Mejoramos así por tanto el rendimiento de nuestra aplicación *Web* al usar este tipo de llamadas asíncronas.

Mostramos a continuación un ejemplo de un función *JavaScript* que realiza una llamada asíncrona usando la función *getJSON()* de *jQuery*.

```
83  function getCustomers() {  
84      jQuery.getJSON("getCustomers.do", function(json) {  
85          muestraCustomers(json);  
86      });  
87  }
```

**Figura 69:** ejemplo de llamada asíncrona

#### 9.2.4.- Métricas del código fuente

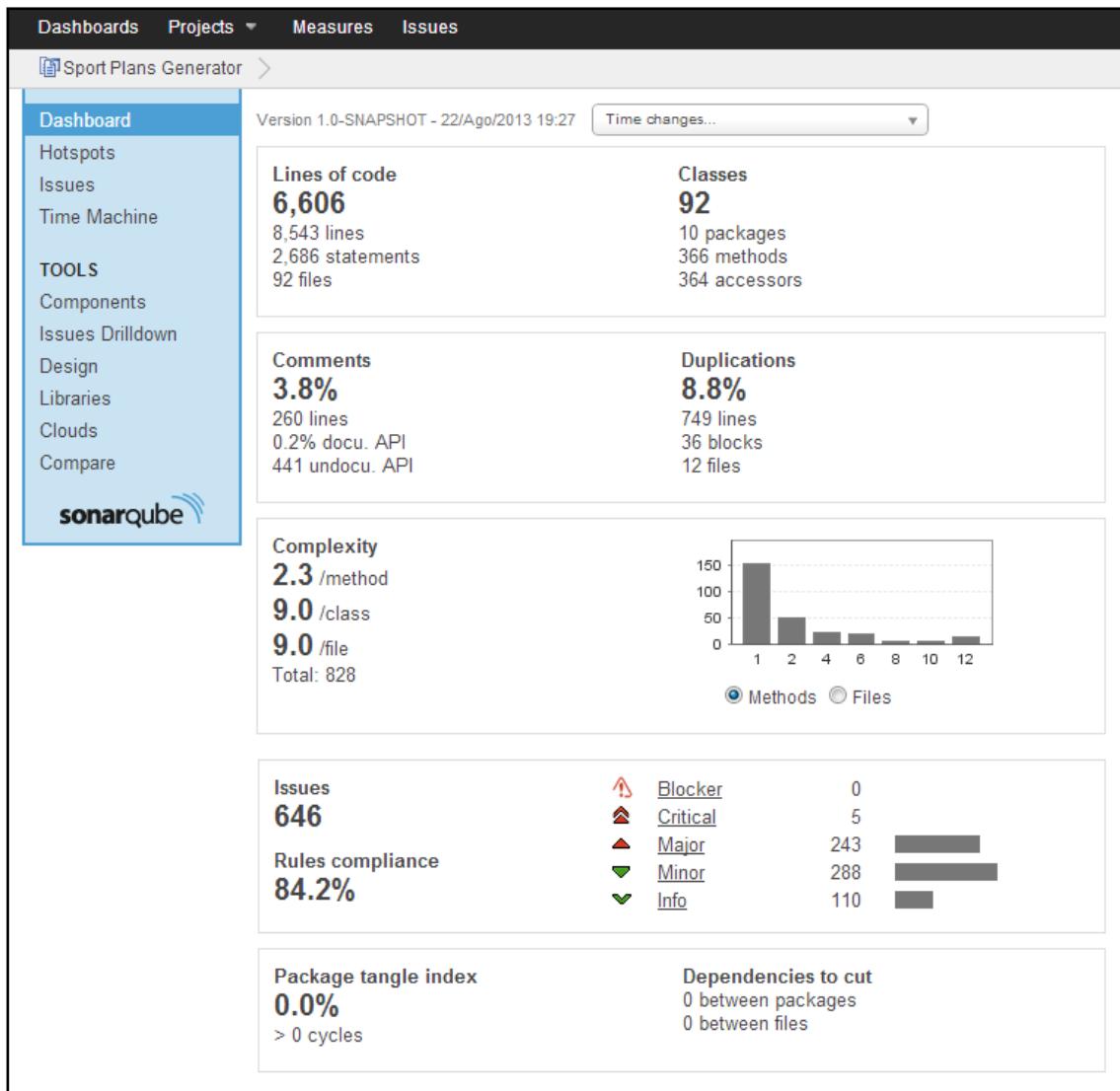


SonarQube es una herramienta de código libre para la evaluación del propio código fuente. Usa diversas herramientas de análisis estático de código fuente para obtener métricas que pueden ayudar a mejorar la calidad del código de un programa [54].

Esta herramienta te informa sobre la calidad de tu código usando diversas métricas. Una característica muy reseñable es que te permite comparar la evolución de tu código con versiones anteriores del mismo, por lo que en cada análisis, puedes saber cuánto has mejorado tu código y en qué.

Durante el desarrollo de nuestra aplicación, hemos ido realizando diversos test a nuestro código para evaluar su calidad con el objetivo de comparar si los cambios introducidos mejoran la calidad de la versión anterior.

Podemos ver en la siguiente imagen el cuadro de mandos principal de la herramienta *SonarQube* donde se nos muestra en una misma pantalla toda la información relevante sobre el código analizado de nuestro proyecto.



**Figura 70:** Dashboard principal de la herramienta *SonarQube*

El número de líneas de código que conforman nuestro proyecto (6.606), el número de clases (92), el número de paquetes (10) o el número de métodos (366) que tiene nuestro código son algunos de los datos que nos muestra este cuadro de mandos.

Podemos ver también el porcentaje de nuestro código que está comentado y el porcentaje de duplicaciones.

Otro interesante dato que nos muestra la herramienta es el grado de complejidad ciclomática de nuestra aplicación. Se trata de una métrica del software que proporciona una medición cuantitativa de la complejidad lógica de un programa. Nos permite saber por tanto “cómo de fácil es de probar y mantener nuestro software”.

La herramienta *SonarQube* realiza el cálculo de la complejidad ciclomática de los métodos de la siguiente forma:

- Cada método tiene una complejidad inicial de 1 excepto los de acceso y modificación de atributos.
- La aparición de cada una de las siguientes palabras reservadas incrementa la complejidad en 1:
  - if
  - for
  - while
  - case
  - catch
  - throw
  - return (cuando no es la última sentencia del método)
  - &&
  - ||
  - ?

A continuación mostramos un ejemplo de un método y el cálculo de su complejidad.

```
public void process(Car myCar){          // +1
    if(myCar.isNotMine()){              // +1
        return;                         // +1
    }
    car.paint("red");
    car.changeWheel();
    while(car.hasGazol() && car.getDriver().isNotStressed()){ // +2
        car.drive();
    }
    return;
}
```

Tras realizar el cálculo de la complejidad del método anterior, obtendríamos un resultado de 5.

Los diferentes grados de complejidad se distribuyen de la siguiente manera [\[78\]](#):

- 1-10 Programa simple, sin mucho riesgo
- 11-20 Más complejo, riesgo moderado
- 21-50 Complejo, programa de alto riesgo
- 50 Programa no testeable, muy alto riesgo

El grado de complejidad ciclomática de nuestra aplicación en el momento en que se realizó la captura es de **2.3%**. Esto se ha conseguido esforzándonos en el proceso de codificación para que nuestro código tenga el menor número de bucles y condiciones múltiples posible. Conseguimos así una complejidad baja y un código mantenible y fácil de testear.

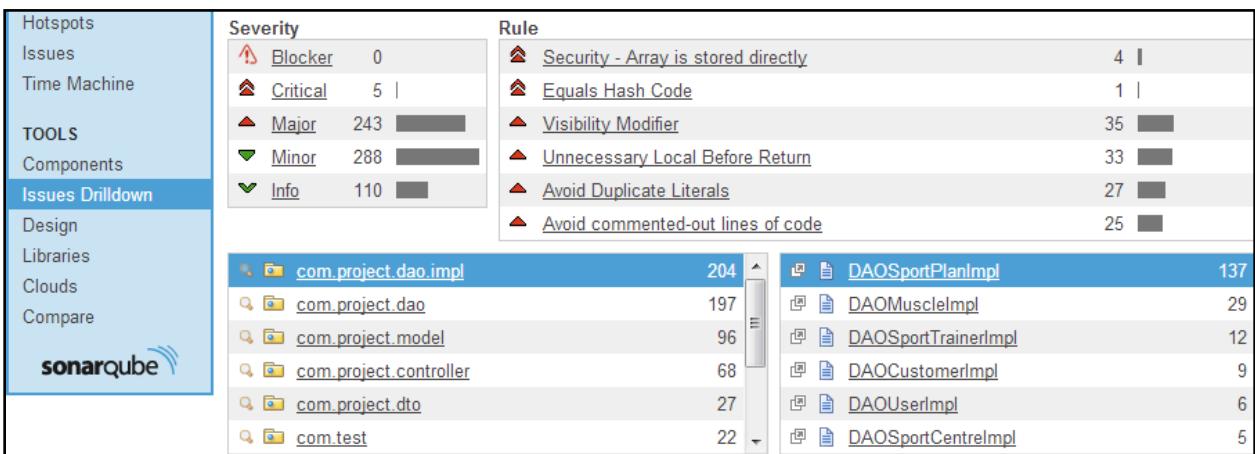
Otro dato interesante a resaltar es la dependencia entre paquetes de nuestro proyecto. Como muestra en el gráfico, hemos conseguido una dependencia de **0**, es decir, que no existen dependencias entre los paquetes, por lo que nuestro posee un bajo acoplamiento y por tanto es fácil de mantener.

Una funcionalidad muy útil de esta herramienta es la detección de fallos en el código. No se trata de fallos de compilación, ya que si no nuestro código no compilaría, si no de reglas de codificación que podemos realizar para que nuestro código sea más eficiente y esté más claro y organizado.

Si nos dirigimos al apartado “*Issues Drilldown*”, podremos ver todas las reglas de codificación que nos ha detectado la herramienta separadas por paquetes y clases.

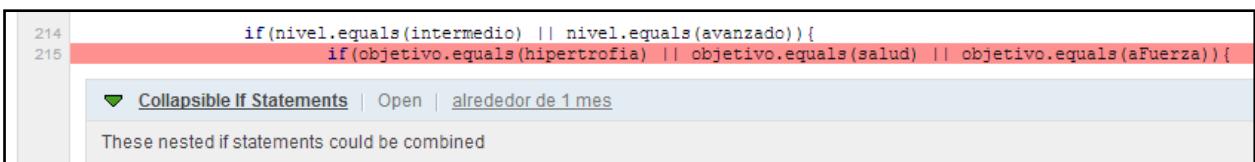
Este tipo de reglas nos ayudan a mejorar la calidad de nuestro programa, su eficiencia y legibilidad. Por ejemplo la regla: “*Unused local variable*” nos avisa de variables locales que tengamos definidas en algún método y que no se esté usando. Con esto reducimos la cantidad de memoria que se reserva inútilmente y así aumentamos la eficiencia de nuestro código.

En la siguiente imagen podemos ver el resumen de las reglas encontradas que nos realiza la herramienta. También nos ofrece una clasificación según la severidad de la regla o el tipo de ésta.



**Figura 71:** reglas de codificación encontradas en nuestro proyecto por la herramienta *SonarQube*

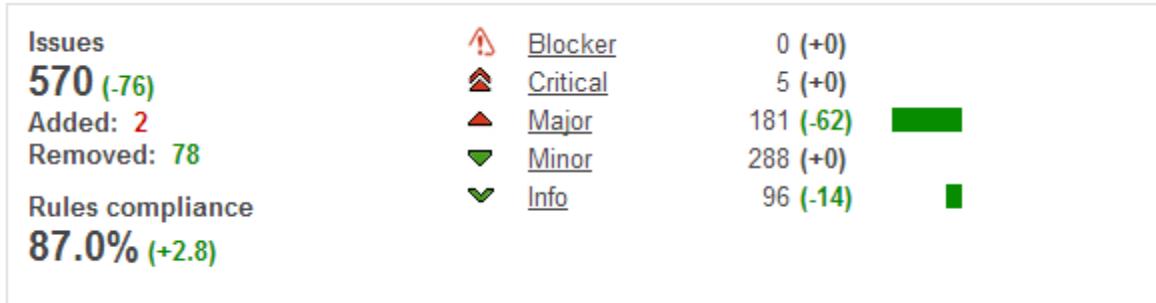
A continuación mostramos un ejemplo de un error detectado en el código y su posible mejora.



**Figura 72:** error detectado y posible mejora

Esta herramienta nos ha sido de gran ayuda, ya que nos ha ayudado a que nuestro código sea mucho más fácil de leer para los desarrolladores, nos ha ayudado a encontrar también trozos de nuestro código que estaban sin utilizar y a mejorar el rendimiento de nuestra aplicación.

En la siguiente imagen se puede comprobar las mejoras obtenidas tras ejecutar un análisis con esta herramienta y realizar varios de los cambios que nos sugería.



**Figura 73:** resumen de las mejoras obtenidas en el código tras ejecutar *SonarQube*

## 10.- Pruebas

La fase de pruebas software es una de las más importantes del proceso de desarrollo software. Probar una aplicación involucra la generación de casos de prueba, la ejecución de la aplicación contra esos casos de prueba y la observación del comportamiento de la aplicación para determinar su corrección (oráculo).

Los objetivos en esta fase serán detectar posibles errores que no hallan sido detectados durante la fase de implementación.

### 10.1.- Pruebas de interfaces y contenidos

En este apartado nos centraremos en revisar la forma en la que se muestran las distintas páginas de nuestra aplicación y su contenido. Realizaremos también diversos chequeos al código *HTML* generado por nuestras vistas *JSP* y a nuestras hojas de estilo en cascada *CSS* para comprobar que cumplen los estándares definidos para estos tipos de documentos.

Realizaremos la siguiente lista de comprobaciones:

#### 10.1.1.- Verificación de contenidos

Este tipo de comprobación consiste en probar que todas las vistas de nuestra aplicación contienen los contenidos adecuados para la propia vista, es decir, que estamos mostrando en cada página la información correspondiente. Como ejemplo clarificador, queremos ver en este apartado que no estamos ofreciendo información correspondiente a un monitor deportivo en la página de inicio de un socio o viceversa.

Por lo tanto, se trata de revisar cada uno de los tipos de vistas que ofrece nuestra aplicación y comprobar que los contenidos que se muestran, las imágenes, la ortografía y la redacción son los adecuados. Es importante verificar también en este apartado que los textos de los enlaces y los distintos botones sean correctos.

Tras comprobar cada una de las vistas de la aplicación de manera concienzuda usando para ello una lista a modo de “*checklist*” con todas las vistas ([Anexo II – Verificación de contenidos](#)), podemos verificar que nuestra aplicación ofrece una redacción correcta de los textos que se muestran. La información mostrada para cada tipo de usuario es la adecuada y los textos que aparecen en cada uno de los enlaces y botones que se muestran son los correctos.

#### 10.1.2.- Verificación de sitios en construcción

Se debe comprobar que la aplicación no contenga sitios vacíos, es decir, en el caso de que en el momento de creación de las vistas, dejásemos alguna sin terminar, llegados a este punto no debe existir ninguna vista sin contenido. No es adecuado encontrar espacios en nuestra aplicación bajo el título “*en construcción...*”. Es preferible eliminar ese apartado en incluirlo posteriormente cuando se tenga completamente terminado.

Para la realización de este tipo de pruebas hemos usado una lista a modo de “*checklist*” al igual que en el apartado anterior ([Anexo II – Verificación de sitios en construcción](#)), y tras realizar el test, podemos afirmar que no existen sitios en construcción o por terminar en nuestra aplicación.

### 10.1.3.- Verificación de estándares

Todos los lenguajes que hemos usado para la capa de presentación de nuestra aplicación siguen unos estándares definidos. Es por ello que llegados a este punto es adecuado testear si el código que compone nuestras vistas sigue dichos estándares. El cumplimiento de dichos estándares nos asegura, gracias al propio estándar, que nuestro código será accesible en todas las plataformas que soporta el estándar.

Para estas verificaciones usaremos las herramientas online que ofrece la W3C (*World Wide Web Consultorium*) [\[66\]](#).

Primeramente realizamos la validación de nuestras hojas de estilo en cascada CSS.

**Figura 74:** validación CSS usando la herramienta online ofrecida por W3C

Como podemos ver en la imagen anterior, el resultado del test fue positivo por lo que nuestras hojas de estilo cumplen los estándares establecidos para este lenguaje.

Realizamos también la validación de nuestro código HTML generado dinámicamente por nuestro código JSP.

**Figura 75:** validación HTML usando la herramienta online ofrecida por W3C

Como se puede observar en la imagen anterior, el resultado del test fue positivo por lo que el código HTML generado por nuestras vistas cumple los estándares establecidos para este lenguaje.

### 10.1.4.- Verificación de interfaces

A través de este test comprobaremos los aspectos gráficos de nuestra aplicación *Web*, para determinar si su despliegue es el correcto.

Realizaremos las siguientes verificaciones:

#### 10.1.4.1.- Librerías y plug-ins necesarios

Comprobaremos en este apartado que la importación de librerías en las vistas para la adición de elementos visuales se realiza correctamente y que los elementos añadidos se muestran adecuadamente.

Es el caso de la librería *jQuery-UI* [68]. Se trata de un conjunto de elementos gráficos que podemos incorporar fácilmente a nuestra aplicación *Web* y que ofrece gran funcionalidad y calidad a nuestras vistas.

En nuestro caso hemos usado dos componentes de esta librería en nuestras vistas.

“*Datepicker*” es el primer elemento utilizado. Se trata de un calendario que nos ayuda fácilmente a encontrar una fecha y brinda a nuestra aplicación de una gran usabilidad. El otro componente de esta librería que hemos usado es “*Accordion*”. Este elemento nos permite agrupar contenido en distintas pestañas con la posibilidad de comprimir las pestañas que no están activas. Por lo que mejora la navegación en nuestras páginas y ayuda al usuario a hacerse una idea del contenido global de cada vista. Este componente ha sido usado en las vistas que muestran el plan de entrenamiento recién generado. En cada pestaña agrupamos los períodos, semanas y sesiones que conforman cada uno de los planes.

Tras diversos tests, se ha comprobado que su importación es correcta y que los componentes son cargados y funcionan con éxito.

Para la comprobación de la importación de las librerías se ha usado la herramienta proporcionada por el navegador *Web Google Chrome: Chrome Developer Tools*.

Podemos ver en la siguiente imagen una de las comprobaciones realizadas durante la carga de una de las pantallas de la aplicación.

Elements Resources Network Sources Timeline Profiles Audits Console									
Name Path		Method	Status	Type	Initiator	Size Content	Time Latency	Timeline	
showSportPlan.do?sportPlanId=109 /SportPlansGenerator		GET	200 OK	text/html	Other	184 KB 184 KB	1.80 s 1.58 s		4.00 s
jquery-1.7.2.min.js /SportPlansGenerator/js/jQuery		GET	200 OK	200 OK	text/javascript	showSportPlan.do:40 Parser	(from cache) 1 ms 0		
jquery.form.js /SportPlansGenerator/js/jQuery		GET	200 OK	text/javascript	Parser	showSportPlan.do:41 Parser	(from cache) 1 ms 0		
jquery.min.js ajax.googleapis.com/ajax/libs/jquery/1.8.2		GET	200 OK	text/javascript	Parser	showSportPlan.do:42 Parser	(from cache) 1 ms 1 ms		
jquery-1.8.3.js code.jquery.com		GET	200 OK	application/...	Parser	showSportPlan.do:81 Parser	(from cache) 1 ms 0		
jquery-ui.css ajax.googleapis.com/ajax/libs/jqueryui/1.9.2/themes/ui-lightness		GET	200 OK	text/css	Parser	showSportPlan.do:80 Parser	6.1 KB 60 ms 30.3 KB 60 ms		
jquery-ui.min.js ajax.googleapis.com/ajax/libs/jqueryui/1.9.2		GET	200 OK	text/javascript	Parser	showSportPlan.do:82 Parser	61.6 KB 252 ms 232 KB 246 ms		
jsPDF.plugin.from_html.js /SportPlansGenerator/js/jPDF		GET	200 OK	text/javascript	Parser	showSportPlan.do:84 Parser	18.2 KB 127 ms 18.0 KB 125 ms		
jsPDF.js /SportPlansGenerator/js/jPDF		GET	200 OK	text/javascript	Parser	showSportPlan.do:83 Parser	76.6 KB 194 ms 76.4 KB 188 ms		

**Figura 75:** comprobación de carga de librerías usando *Chrome Developer Tools*

#### 10.1.4.2.- Imágenes escaladas

Se debe verificar en este apartado que las imágenes que estamos ofreciendo en nuestra aplicación *Web* no estén siendo manipuladas por las hojas de estilo, es decir, no queremos mostrar imágenes en nuestra aplicación cuyo ratio se distinga del ratio original de la imagen. También comprobaremos que no estamos mostrando imágenes pixeladas o con poca calidad.

Tras una comprobación exhaustiva revisando cada una de las imágenes que se muestran acompañando a cada ejercicio deportivo, no se han encontrado imágenes deformadas o de baja calidad. Consultar en el [Anexo II – Imágenes escaladas](#) para ver más detalles.

#### **10.1.4.3.- Imágenes sin atributo ALT**

Para cumplir con las normas de accesibilidad es necesario que todas las imágenes que se usen en un sitio *Web* tengan una descripción apropiada usando el atributo ALT de la misma en la etiqueta IMG del lenguaje HTML.

Han sido revisadas todas las imágenes que se muestran en nuestra aplicación *Web* y todas contienen el atributo *ALT* definido para cumplir con la norma de accesibilidad. *Consultar en el Anexo II – Imágenes sin atributo ALT*.

## **10.2.- Pruebas funcionales y de operación**

Las pruebas a realizar en esta sección se refieren a ejecutar chequeos completos a las funcionalidades que ofrece nuestra aplicación *Web*.

Para este tipo de pruebas usaremos la herramienta de código libre *Selenium* [76]. Se trata de un entorno de pruebas de software para aplicaciones basadas en la *Web*. Incluye además un lenguaje específico de dominio para escribir dichas pruebas en un amplio número de lenguajes, en nuestro caso *Java*. Las pruebas creadas podrán ser ejecutadas entonces usando la mayoría de los navegadores *Web* actuales, en nuestro caso *Mozilla Firefox*.

Una de las grandes ventajas que nos brinda el uso de esta herramienta es la posibilidad de automatizar nuestras pruebas. Así, cuando se continúe con la mejora y actualización de la aplicación añadiendo o mejorando componentes, y queramos realizar pruebas de regrasión, sólo tendremos que ejecutar nuestro conjunto de pruebas realizadas con *Selenium* y comprobar que todo nuestro anterior trabajo sigue funcionando. Es decir, una vez que tengamos un conjunto de pruebas funcionales realizadas y realicemos cualquier cambio en nuestra aplicación, podremos volver a ejecutar el conjunto de pruebas funcionales para comprobar que ninguna de las funcionalidades anteriores de nuestra aplicación han sido dañadas.

Para usar *Selenium* en nuestro proyecto, realizaremos una serie de casos de prueba en los que testearemos las principales funcionalidades de nuestro proyecto.

En la siguiente imagen podemos ver un ejemplo de su uso. Realizaremos nuestros tests integrándolos dentro de casos de prueba creados con la herramienta *JUNIT 4*.

```

1 package selenium;
2
3+ import static org.junit.Assert.assertEquals;+
11
12 public class LoginCustomerTest {
13
14     private WebDriver browser;
15
16@ Before
17 public void setup() {
18     browser = new FirefoxDriver();
19     browser.get("http://localhost:8080/SportPlansGenerator/");
20 }
21
22@ Test
23 public void LoginCliente() {
24     browser.findElement(By.id("usernameInput")).sendKeys("flopez");
25     browser.findElement(By.id("passwordInput")).sendKeys("flopez");
26
27     browser.findElement(By.id("submit")).click();
28
29     assertEquals("Cliente", browser.getTitle());
30 }
31
32@ After
33 public void tearDown() {
34     browser.close();
35 }
36 }

```

**Figura 76:** Caso de prueba funcional usando *Selenium*

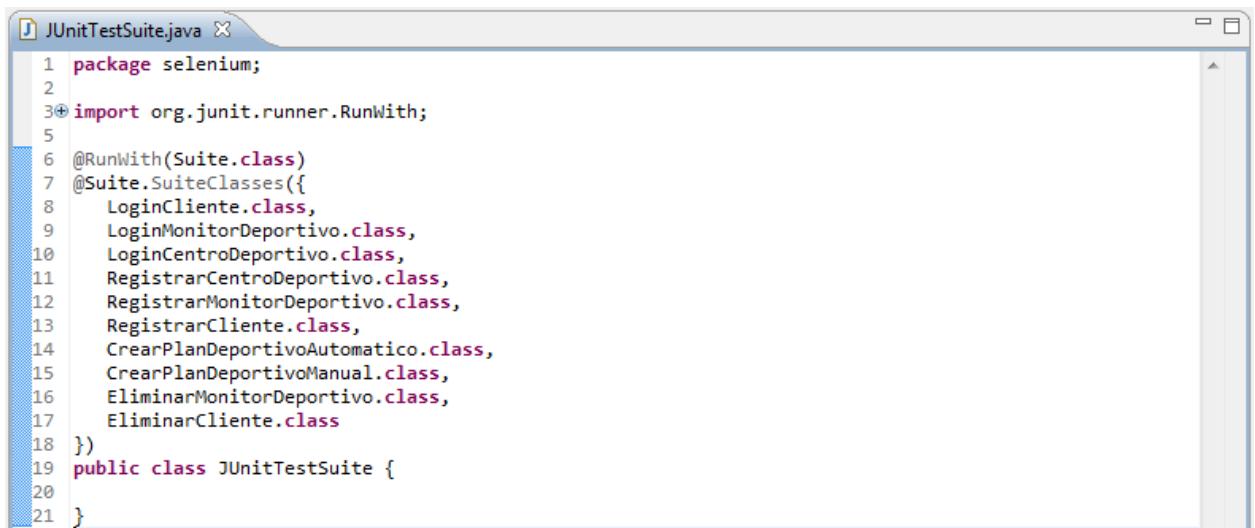
El caso de uso anterior usa el navegador *Mozilla Firefox*, que a partir del driver ofrecido por *Selenium* nos permite interactuar desde el código simulando el comportamiento de un usuario. Introducimos en el login el nombre de usuario y la contraseña de un usuario de prueba, en este caso un cliente, y pulsamos el botón de “*login*”. Si el login es correcto, la aplicación debería mostrar el menú de Cliente en el que el título de la página sería “*Cliente*”.

Al igual que este, realizaremos otros tests para probar cada una de las funcionalidades ofrecida por nuestra aplicación para cada uno de los tipos de usuarios que se soportan.

En la siguiente tabla podemos ver cada test realizado y el resultado obtenido.

Prueba	Salida esperada	Resultado Obtenido	¿Éxito?
Login Cliente	Login correcto	Login correcto	SI
Login Monitor Deportivo	Login correcto	Login correcto	SI
Login Centro Deportivo	Login correcto	Login correcto	SI
Registrar Centro Deportivo	Registro correcto	Registro correcto	SI
Registrar Monitor Deportivo	Registro correcto	Registro correcto	SI
Registrar Cliente	Registro correcto	Registro correcto	SI
Crear Plan Deportivo Automático	Creación correcta	Creación correcta	SI
Crear Plan Deportivo Manual	Creación correcta	Creación correcta	SI
Eliminar Monitor Deportivo	Eliminación correcta	Eliminación correcta	SI
Eliminar Cliente	Eliminación correcta	Eliminación correcta	SI

Para ejecutar automáticamente todo este conjunto de tests, hemos realizado un “*TestSuite*” usando las funcionalidades ofrecidas por *JUNIT*. De esta forma, sólo será necesario ejecutar la clase que forma el “*TestSuite*” para ejecutar todos los tests automáticamente. Podemos ver en la siguiente imagen la forma que tendrá este conjunto de pruebas.



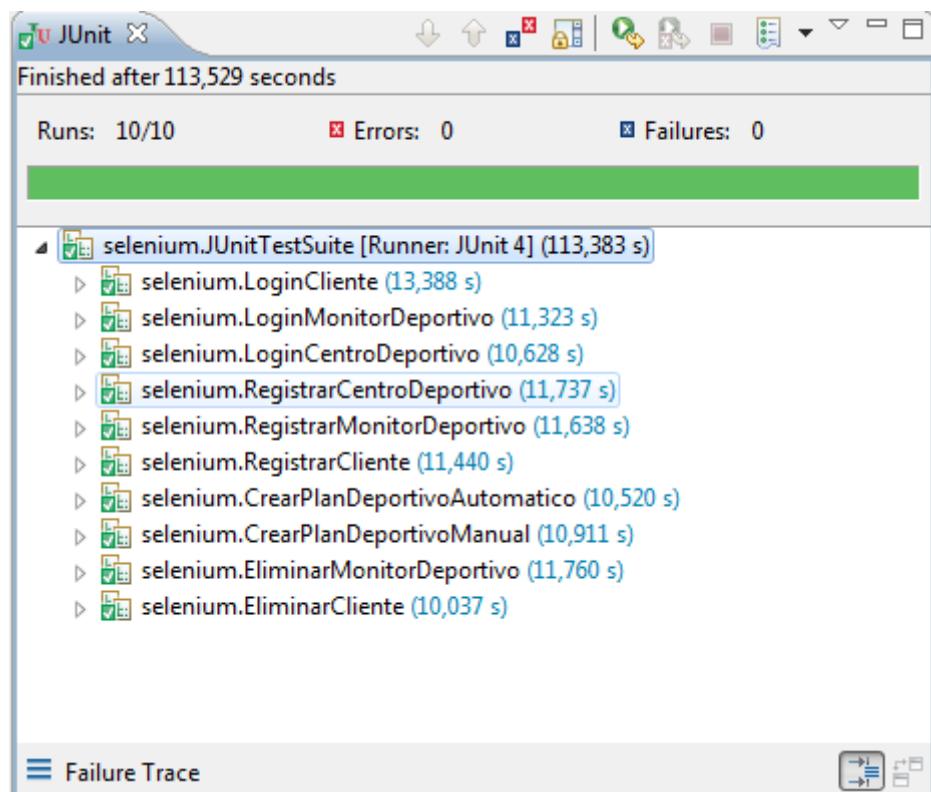
```

1 package selenium;
2
3 import org.junit.runner.RunWith;
4
5
6 @RunWith(Suite.class)
7 @Suite.SuiteClasses({
8     LoginCliente.class,
9     LoginMonitorDeportivo.class,
10    LoginCentroDeportivo.class,
11    RegistrarCentroDeportivo.class,
12    RegistrarMonitorDeportivo.class,
13    RegistrarCliente.class,
14    CrearPlanDeportivoAutomatico.class,
15    CrearPlanDeportivoManual.class,
16    EliminarMonitorDeportivo.class,
17    EliminarCliente.class
18 })
19 public class JUnitTestSuite {
20
21 }

```

**Figura 77:** *TestSuite* usando la herramienta *JUNIT*

Podemos ver además el resultado obtenido tras la ejecución de este conjunto de pruebas en la siguiente imagen.



**Figura 78:** resultados de las pruebas realizadas con *Selenium* y *JUNIT*

### 10.3.- Pruebas de carga y rendimiento

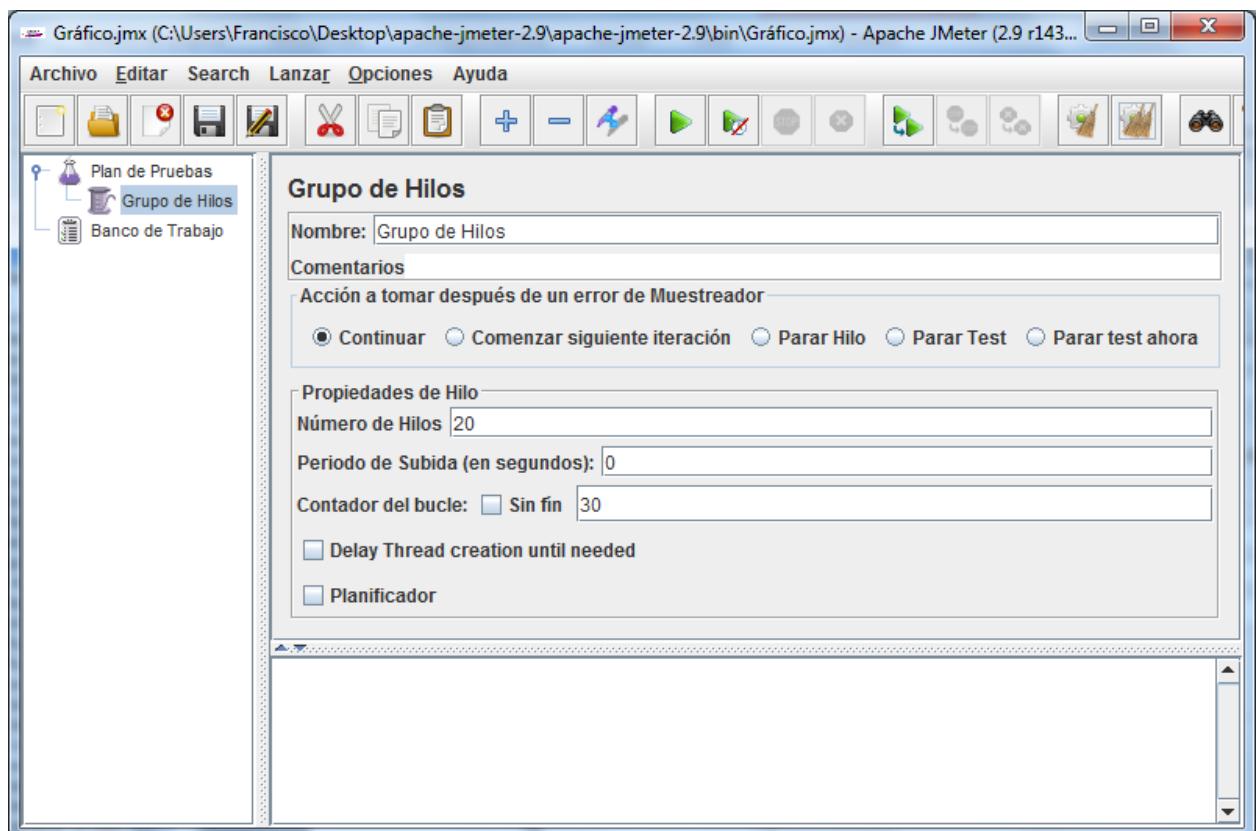
Para realizar las pruebas de carga y rendimiento, usaremos la herramienta de código libre *JMeter*. Se trata de un proyecto de *Apache* que se utiliza como una herramienta de carga para analizar y medir el desempeño de una variedad de servicios, en nuestro caso de aplicaciones Web [70].

Las primeras pruebas que vamos a realizar son pruebas de sobrecarga de la aplicación. Usando la herramienta *JMeter* simularemos una serie de conexiones simultáneas para ver como se comporta el servidor ante un número alto de peticiones.

Queremos simular con esta prueba una situación real en la que nuestra aplicación es accedida en un mismo instante de tiempo por una gran cantidad de usuarios.

Para ello, usaremos la aplicación de escritorio ofrecida gratuitamente en la *Web* de la fundación *Apache* [70]. Tras descargarla y guardar los archivos descargados, ejecutamos el fichero *bin/jmeter.bat* para iniciar la aplicación.

El primer paso a realizar es crear un grupo de hilos que simularán cada una de las conexiones que se realizarán simultáneamente al servidor. Para ello, hacemos pulsamos con el botón secundario sobre el *Plan de Pruebas* que aparece en el panel izquierdo y añadimos un grupo de hilos.



**Figura 79:** agregar *Grupo de Hilos* a *JMeter*

Realizaremos los siguientes tests:

Tests de sobrecarga del servidor			
Prueba	Hilos	Periodo de subida (seg)	Contador de bucle
Prueba 1	25	0	30
Prueba 2	50	0	30
Prueba 3	100	0	30

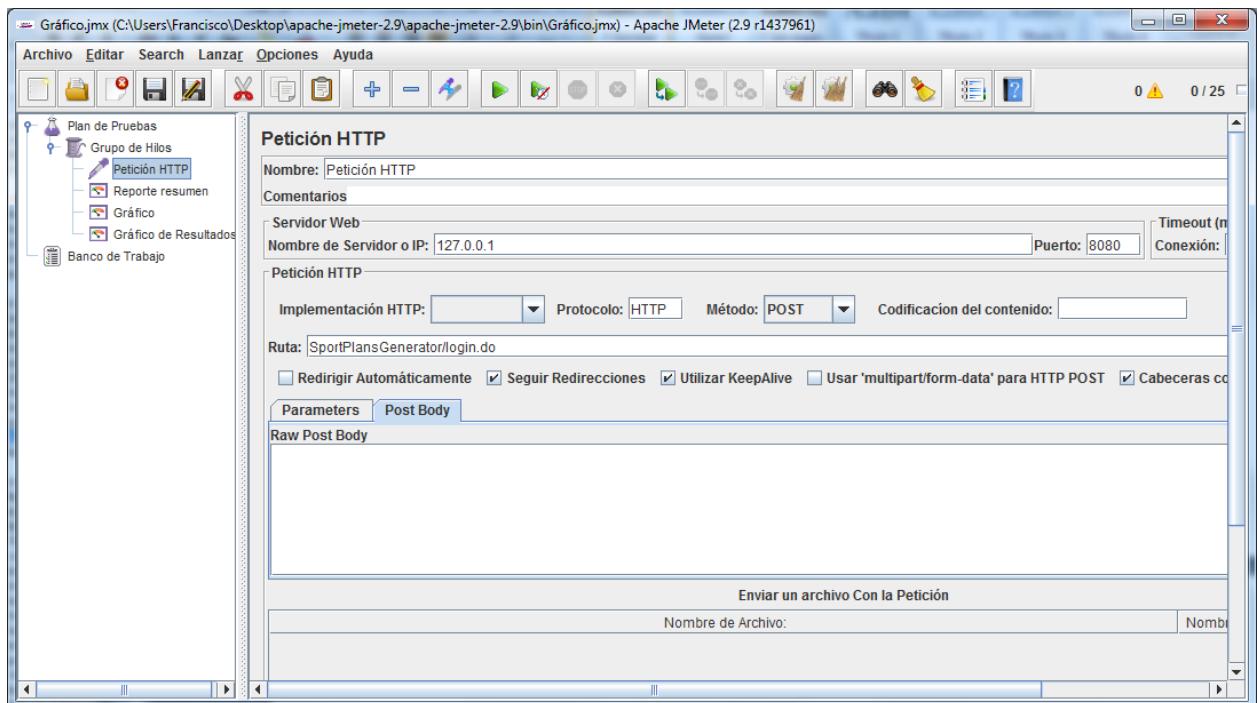
**Hilos:** número de conexiones simultáneas que se simularán en la prueba.

**Periodo de subida:** tiempo de espera entre la creación de un hilo y el siguiente.

**Contador de bucle:** número de veces que cada hilo realizará una conexión.

A continuación indicaremos a *JMeter* que el test que queremos realizar se trata de peticiones *HTTP* para comprobar así como responde nuestro servidor.

Para ello añadimos una *Petición HTTP* pulsando en el botón derecho del grupo de hilos recientemente añadido y configuramos los parámetros que nos aparecen en la ventana emergente como se muestra en la siguiente imagen.



**Figura 80:** añadir Petición *HTTP* al grupo de Hilos en *JMeter*

A continuación estamos listos para comenzar con las pruebas. Mostramos a continuación los resultados obtenidos en cada una de ellas.

### 10.3.1.- Prueba de carga número 1

**Nº de hilos:** 25

**Periodo de subida:** 0 segundos. Todos los hilos son creados simultáneamente

**Contador de bucle:** 30

A continuación se muestran los resultados obtenidos tras la ejecución de esta prueba. Para ello usamos el componente de *JMeter “Reporte – Resumen”*.

Etiqueta	# Muestras	Media	Mín	Máx	Desv. Estándar	% Error	Rendimiento	Kb/sec	Media de Bytes
Total	750	22	3	331	42,02	100,00%	275,5/sec	342,53	1273,0

Como podemos observar en los resultados, el tiempo medio de acceso a nuestra aplicación es 22 milisegundos, realizándose un total de 750 requerimientos al servidor.

Evaluaremos los resultados obtenidos a través de un intervalo de confianza para la distribución Normal al 95%. La fórmula que usaremos será la siguiente:

$$[ TP - Z_{0.95} * D / \sqrt{n}, TP + Z_{0.95} * D / \sqrt{n} ]$$

Donde:

**TP** (Tiempo de respuesta): 22

**D** (Desviación): 42,02

**n** (Tamaño de la muestra): 750

**Z<sub>0.95</sub>**: 1,96

Por lo que el intervalo obtenido es el siguiente:

$$[18,99 \text{ ms} - 25,00 \text{ ms}]$$

Concluimos por tanto tras la primera prueba que se puede esperar que el tiempo de respuesta promedio esté entre **18,99 y 25 milisegundos** para una cantidad de **25 usuarios simultáneos** realizando un total de **750 peticiones**.

Mostramos a continuación una gráfica construida con el módulo “*Gráfico de Resultados*” de la herramienta *JMeter*.



**Figura 81:** gráfica ofrecida por el módulo “*Gráfico de Resultados*” de la herramienta *JMeter*

### 10.3.2.- Prueba de carga número 2

**Nº de hilos:** 50

**Periodo de subida:** 0 segundos. Todos los hilos son creados simultáneamente

**Contador de bucle:** 30

A continuación se muestran los resultados obtenidos tras la ejecución de esta prueba. Para ello usamos el componente de *JMeter “Reporte – Resumen”*.

Etiqueta	# Muestras	Media	Mín	Máx	Desv. Estándar	% Error	Rendimiento	Kb/sec	Media de Bytes
Petición HTTP	1500	15	1	201	23,50	100,00%	415,4/sec	516,41	1273,0

Como podemos observar en los resultados, el tiempo medio de acceso a nuestra aplicación es 15 milisegundos, realizándose un total de 1500 requerimientos al servidor.

Evaluaremos los resultados obtenidos a través de un intervalo de confianza para la distribución Normal al 95%. La fórmula que usaremos será la siguiente:

$$[ TP - Z_{0.95} * D / \sqrt{n}, TP + Z_{0.95} * D / \sqrt{n} ]$$

Donde:

**TP** (Tiempo de respuesta): 15

**D** (Desviación): 23,50

**n** (Tamaño de la muestra: 1500

**Z<sub>0.95</sub>**: 1,96

Por lo que el intervalo obtenido es el siguiente:

$$[13,82 \text{ ms} - 16,18 \text{ ms}]$$

Concluimos por tanto tras la segunda prueba que se puede esperar que el tiempo de respuesta promedio esté entre **13,82 y 16,18 milisegundos** para una cantidad de **50 usuarios simultáneos** realizando un total de **1500 peticiones**.

Mostramos a continuación una gráfica construida con el módulo “*Gráfico de Resultados*” de la herramienta *JMeter*.



**Figura 82:** gráfica ofrecida por el módulo “Gráfica de Resultados” de la herramienta JMeter

### 10.3.3.- Prueba de carga número 3

Nº de hilos: 100

Periodo de subida: 0 segundos. Todos los hilos son creados simultáneamente

Contador de bucle: 30

A continuación se muestran los resultados obtenidos tras la ejecución de esta prueba. Para ello usamos el componente de JMeter “Reporte – Resumen”.

Etiqueta	# Muestras	Media	Mín	Máx	Desv. Estándar	% Error	Rendimiento	Kb/sec	Media de Bytes
Petición HTTP	3000	25	2	496	69,07	100,00%	522,4/sec	649,40	1273,0

Como podemos observar en los resultados, el tiempo medio de acceso a nuestra aplicación es 25 milisegundos, realizándose un total de 3000 requerimientos al servidor.

Evaluaremos los resultados obtenidos a través de un intervalo de confianza para la distribución Normal al 95%. La fórmula que usaremos será la siguiente:

$$[ TP - Z_{0.95} * D / \sqrt{n}, TP + Z_{0.95} * D / \sqrt{n} ]$$

Donde:

TP (Tiempo de respuesta): 25

D (Desviación): 69,07

n (Tamaño de la muestra): 3000

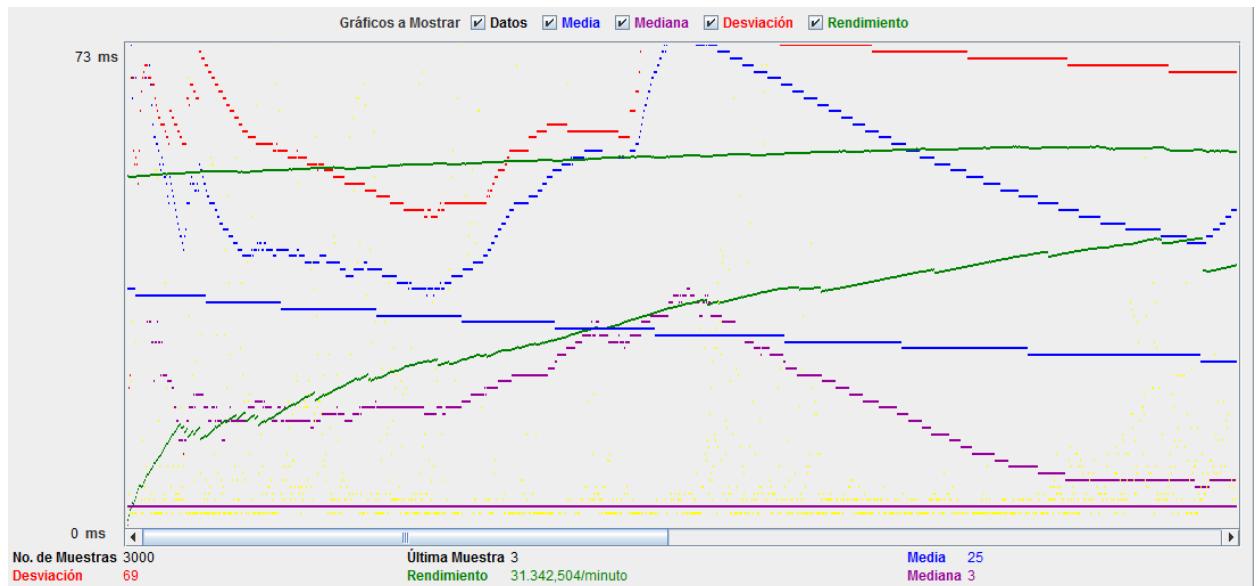
Z<sub>0.95</sub>: 1,96

Por lo que el intervalo obtenido es el siguiente:

$$[24,15 \text{ ms} - 25,84 \text{ ms}]$$

Concluimos por tanto tras la tercera prueba que se puede esperar que el tiempo de respuesta promedio esté entre **24,15 y 25,84 milisegundos** para una cantidad de **100 usuarios simultáneos** realizando un total de **3000 peticiones**.

Mostramos a continuación una gráfica construida con el módulo “*Gráfico de Resultados*” de la herramienta *JMeter*.



**Figura 83:** gráfica ofrecida por el módulo “*Gráfico de Resultados*” de la herramienta *JMeter*

## 11.- Manual de instalación

A continuación comentaremos los requisitos necesarios para poder ejecutar la aplicación *Sport Plans Generator*, y los pasos a seguir para instalarla con éxito.

### 11.2.- Requisitos

Para poder desplegar la aplicación *Sport Plans Generator* será necesario instalar previamente los siguientes elementos:

- Apache Tomcat versión 5.0 o superior [\[44\]](#)
- MySQL Server 5.0 o superior [\[46\]](#)

Los manuales de instalación de cada uno de los anteriores elementos pueden ser consultados en los enlaces que aparecen a la izquierda del nombre.

### 11.3.- Proceso de instalación

A continuación comentaremos los pasos a seguir para instalar con éxito nuestra aplicación. Para ello debemos de tener previamente instalados y configurados los requisitos necesarios que se han comentado en el apartado 11.2.- *Requisitos*.

#### 11.3.1.- Despliegue de la aplicación

Para desplegar nuestra aplicación necesitamos previamente arrancar el servidor de aplicaciones Tomcat. Para ello ejecutamos el archivo ***bin\startup.bat*** si nos encontramos bajo la plataforma *Windows*, o el archivo ***bin/startup.sh*** si nos encontramos en la plataforma *UNIX*.

Una vez ejecutado, tendremos nuestro servidor de aplicaciones funcionando. El siguiente paso es abrir un navegador *Web* e introducir la dirección *Web*: <http://127.0.0.1:8080/manager/html>, la cual nos llevará a la página de administración de aplicaciones de Tomcat, el cual podemos ver en la siguiente imagen.

<b>Desplegar</b>	
Desplegar directorio o archivo WAR localizado en servidor	
<input type="text" value="Trayectoria de Contexto (opcional):"/> <input type="text"/> <input type="text" value="URL de archivo de Configuración XML:"/> <input type="text"/> <input type="text" value="URL de WAR o Directorio:"/> <input type="text"/> <input type="button" value="Desplegar"/>	
<b>Archivo WAR a desplegar</b>	
<input type="text" value="Seleccione archivo WAR a cargar"/> <input type="button" value="Seleccionar archivo"/> No se ha seleccionado ningún archivo <input type="button" value="Desplegar"/>	

**Figura 84:** administrador de aplicaciones Tomcat

Para desplegar nuestra aplicación, únicamente deberemos seleccionar el archivo *.war* de la misma y pulsar el botón que aparece abajo con el texto “Desplegar”.

En este momento tendríamos nuestra aplicación desplegada en el servidor Tomcat. Para acceder a ella debemos introducir la siguiente dirección en un navegador Web:  
<HTTP://127.0.0.1:8080/SportPlansGenerator>.

### 11.3.2.- Instalación de la base de datos

Para el correcto funcionamiento de la aplicación *Sport Plans Generator* es necesario instalar previamente la base de datos que contendrá todos los ejercicios que se incluyen por defecto en la aplicación, además de otro tipo de datos imprescindibles.

El primer paso a realizar es ejecutar MySQL. Este paso puede realizarse de diversas formas.

La forma más directa es ejecutar el archivo **bin/mysqld.exe**.

Si tenemos instalado una herramienta del tipo *XAMPP*, simplemente pulsaremos en el botón “Start” que aparece a la derecha del módulo MySQL. Podemos verlo más claro en la siguiente imagen.

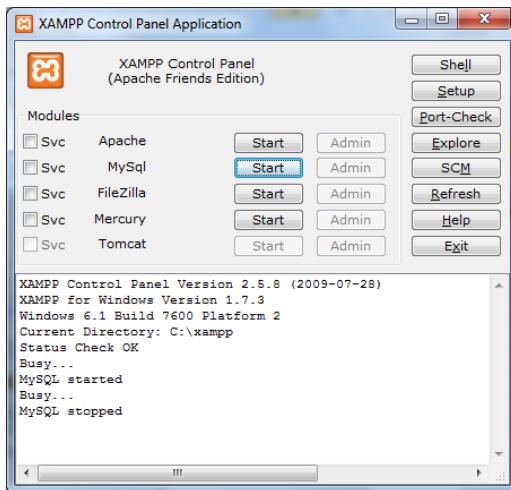


Figura 85: panel de control de XAMPP

Una vez arrancado el servicio de MySQL, abriremos una ventana de comandos y ejecutaremos la siguiente línea:

```
mysql -u #username -p #password MYBASE < #ruta_archivo_.sql
```

En la cual los parámetros a introducir son:

- **#username**: el nombre de usuario con el que nos conectaremos a la base de datos.
- **#password**: la contraseña con la que nos conectaremos a la base de datos.
- **#ruta\_archivo\_.sql**: la ruta hasta el archivo .sql que contiene la base de datos de la aplicación.

Una vez realizados estos pasos, tendremos nuestra base de datos instalada y lista para funcionar

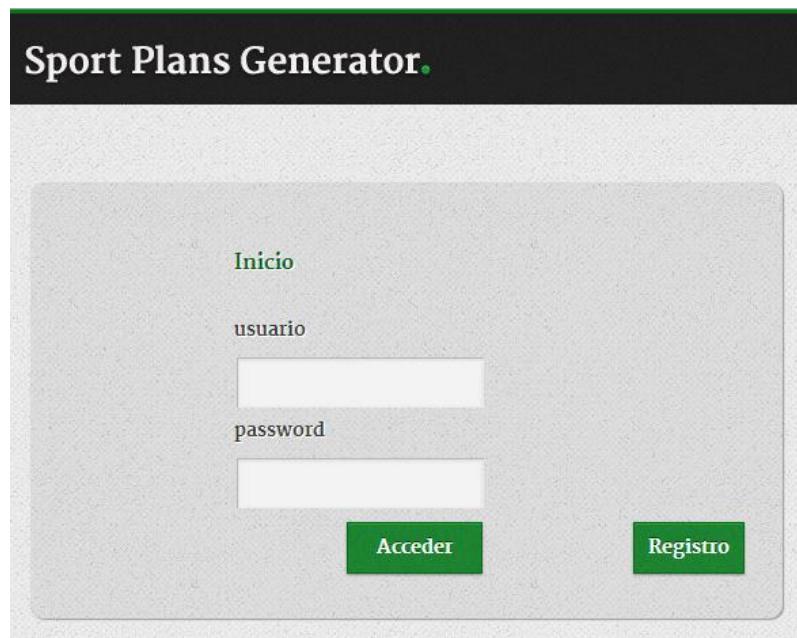
## 12.- Manual de uso del sistema

En este capítulo mostraremos cómo hacer uso de la aplicación desarrollada para cada uno de los tipos de usuarios que la utilizarán.

### 12.1.- Manual de funcionamiento

La aplicación que hemos desarrollado, *Sport Plans Generator* podrá ser usada por tres tipos de usuarios distintos: centros deportivos, monitores deportivos y socios. Como primer paso antes de empezar a hacer uso de los servicios que ofrece la aplicación, los distintos usuarios necesitarán acceder al sistema facilitando su usuario y contraseña. Este paso es común para todos los usuarios.

A continuación podemos ver el formulario de acceso a la aplicación (login).



**Figura 86:** pantalla de login de la aplicación

Para aquellos centros deportivos que deseen empezar a utilizar nuestra aplicación, podrán hacerlo accediendo al formulario de registro de nuevos centros deportivos a través del botón que aparece en la pantalla de acceso (login) de la aplicación.

A continuación mostramos dicho formulario de registro.

The screenshot shows a registration form titled "Nuevo Centro Deportivo". It contains five input fields: "usuario", "password", "repita password", "centro deportivo", and "email". Below the fields is a green button labeled "Dar de alta".

**Figura 87:** pantalla de registro de nuevos centros deportivos la aplicación

Los datos solicitados al principio del formulario, usuario y password, serán requeridos posteriormente para el acceso a la aplicación.

Una vez que el usuario realiza el login a la aplicación puede empezar a hacer uso de sus servicios. El resto de funcionalidades que ofrece la aplicación serán mostradas en cada uno de los apartados siguientes del manual.

## 12.2.- Manual de centro deportivo

A continuación describiremos las funcionalidades disponibles para los centros deportivos registrados en la aplicación *Sport Plans Generator*.

### 12.2.1.- Dashboard

Una vez que el centro deportivo accede a la aplicación, tendrá disponible el menú o *Dashboard* de centro deportivo. Desde este menú podrá acceder de una manera rápida a todas las funcionalidades disponibles para los centros deportivos.

A continuación mostramos el menú o *Dashboard* de centro deportivo.

Nombre	Apellido	Info	Eliminar
Carlos	Martinez	<a href="#">-&gt;</a>	<a href="#">X</a>
Ernesto	Valle	<a href="#">-&gt;</a>	<a href="#">X</a>
Rosalía	Rodríguez	<a href="#">-&gt;</a>	<a href="#">X</a>

**Figura 88:** menú o *Dashboard* de un centro deportivo

En este menú o Dashboard el usuario centro deportivo puede consultar toda la información relevante en una misma pantalla.

Se encuentra dividido en tres paneles, de los cuales el panel superior izquierda muestra información relevante: número de monitores y socios que ese centro deportivo tiene asociados y el número total de planes deportivos que han sido generados en ese centro deportivo.

El panel inferior izquierda nos permite ver rápidamente la lista de monitores deportivos o socios pertenecientes al centro deportivo. Desde ese panel podremos acceder a las pantallas de configuración.

El tercer panel situado en la mitad derecha de la pantalla se trata de un panel dinámico. Su contenido irá variando según la opción señalada en el *Panel de Control*.

### 12.2.2.- Registrar un nuevo monitor deportivo

Para registrar un nuevo monitor deportivo, en el *Dashboard* principal se debe acceder al formulario de registro de nuevo monitor deportivo pulsando en el enlace que aparece en la parte inferior del menú de "Monitores".

Nos aparecerá el formulario para registrar a un nuevo monitor deportivo. En la imagen que aparece a continuación se muestra el aspecto de dicho formulario.



Este formulario se titula "Nuevo Monitor Deportivo". Se divide en dos secciones principales: "usuario" y "monitor deportivo". La sección "usuario" contiene los campos "password" y "repita password". La sección "monitor deportivo" contiene los campos "apellidos" y "email". A la derecha de cada sección hay un cuadro vacío para introducir datos. En la parte inferior derecha del formulario hay un botón verde con el texto "Nuevo".

**Figura 89:** formulario de registro de nuevo monitor deportivo

Los campos usuario y password solicitados serán usados posteriormente para que dicho monitor deportivo pueda acceder a la aplicación. Se piden también otros datos como el nombre del monitor o su correo electrónico.

Una vez rellenados todos los campos que se nos solicitan, el nuevo monitor nos aparecerá en la lista de monitores en el menú principal y podrá acceder con sus credenciales a la aplicación para empezar a hacer uso de ella.

#### **12.2.3.- Modificar la información personal de un monitor deportivo**

Para consultar y/o modificar la información personal referente a un monitor deportivo perteneciente a nuestro centro deportivo tendremos que pulsar en el icono  que aparece al lado del nombre de cada monitor deportivo.

El formulario de modificación de datos personales es mostrado en la siguiente imagen.

**Modificar datos**

monitor deportivo	ciudad
Carlos	Sevilla
apellidos	país
Martinez	España
tlf. fijo	codigo postal
955850069	41005
tlf. móvil	dirección
34 616138264	Avenida Ciudad Jardín

**Guardar**

**Figura 90:** formulario de modificación de datos personales de un monitor deportivo

#### 12.2.4.- Registrar un nuevo socio

Para registrar un nuevo socio, en el *Dashboard* principal se debe acceder al formulario de registro de nuevo socio pulsando en el enlace que aparece en la parte inferior del menú de “Socios”.

Nos aparecerá el formulario para registrar a un nuevo socio. En la imagen que aparece a continuación se muestra el aspecto de dicho formulario.

**Nuevo Socio**

usuario	nombre
password	apellidos
repita password	email

**Nuevo**

**Figura 91:** formulario de registro de nuevo socio

Los campos usuario y password solicitados serán usados posteriormente para que dicho socio pueda acceder a la aplicación. Se piden también otros datos como el nombre del socio o su correo electrónico. Una vez rellenados todos los campos que se nos solicitan, el socio nos aparecerá en la lista de socios en el menú principal y podrá acceder con sus credenciales a la aplicación para empezar a hacer uso de ella.

### 12.2.5.- Modificar la información personal socio

Para consultar y/o modificar la información personal referente a un socio perteneciente a nuestro centro deportivo tendremos que pulsar en el icono  que aparece al lado del nombre de cada socio.

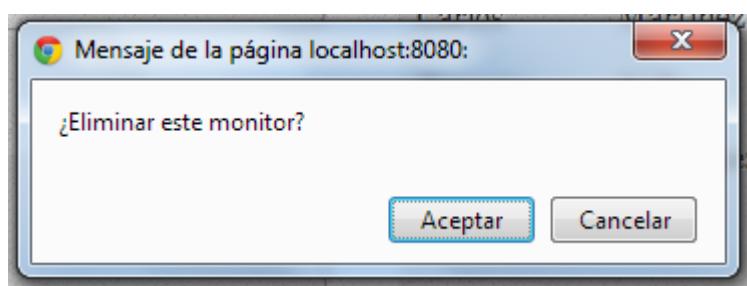
El formulario de modificación de datos personales es el mismo que el mostrado en el apartado 12.2.3.- *Modificar información personal de un monitor deportivo*. Los pasos a seguir para completar el formulario son iguales que los explicados en dicho apartado.

### 12.2.6.- Eliminación de monitores deportivos o socios

Para dar de baja de nuestro centro deportivo a un monitor deportivo o un socio, debemos pulsar en el icono  que aparece a la derecha de del nombre, según sea monitor deportivo o socio.

Debemos estar completamente seguros antes de realizar esta acción, ya que sus resultados son irreversibles y el monitor deportivo o socio quedará dado de baja de nuestro centro deportivo para siempre.

La aplicación realiza una última comprobación de la acción que vamos a realizar pidiéndonos de nuevo que confirmemos la eliminación.



**Figura 92:** comprobación ante la eliminación de un monitor deportivo

### 12.2.7.- Configuración

El apartado de configuración nos ofrece la posibilidad tanto de cambiar los datos de contacto referentes al centro deportivo así como la contraseña requerida para el acceso a la aplicación.

Estas opciones son mostradas en la siguiente imagen.



**Figura 93:** panel de configuración de un centro deportivo

El formulario de modificación de datos de contacto es similar al explicado en la sección 12.2.3.- *Modificar información personal de un monitor deportivo*. Puede consultarse esa sección para ver más detalles.

El formulario de modificación de la contraseña de acceso a la aplicación es mostrado en la siguiente imagen.

Este formulario titulado 'Modificar contraseña' tiene dos campos: 'contraseña' y 'repita la contraseña'. El campo 'contraseña' contiene '\*\*\*\*'. Una alerta roja dice: 'Please enter at least 6 characters.' El botón 'Guardar' está en el pie del formulario.

**Figura 94:** formulario de modificación de la contraseña

Podemos ver en la captura de pantalla como la aplicación nos alerta si la contraseña introducida no cumple con las restricciones de tamaño establecidas.

## 12.3.- Manual de monitor deportivo

A continuación describiremos las funcionalidades disponibles para los monitores deportivos registrados en la aplicación *Sport Plans Generator*.

### 12.3.1.- Dashboard

Una vez que el monitor deportivo accede a la aplicación, tendrá disponible el menú o *Dashboard* de monitor deportivo. Desde este menú podrá acceder de una manera rápida a todas las funcionalidades disponibles para los monitores deportivos.

A continuación mostramos el menú o *Dashboard* de monitor deportivo.

Nombre	Apellido	Edad	Peso	Altura	Actividad Laboral	Nivel
Francisco	Lopez	25 años	75 kg	175 cm	Ingeniero Informático	Aclimatación
Juan	Diaz					

**Figura 95:** menú o Dashboard de monitor deportivo

En este menú o Dashboard el monitor deportivo puede consultar toda la información relevante en una misma pantalla.

Se encuentra dividido en tres paneles, de los cuales el panel superior izquierda muestra información relevante: el centro deportivo al que pertenece, el número de socios que ese centro deportivo tiene asociados y el número total de planes deportivos que han sido generados en ese centro deportivo.

El panel que se encuentra a la derecha nos permite intercambiar entre la lista de socios y las opciones de configuración.

El tercer panel situado en la parte inferior de la pantalla se trata de un panel dinámico. Su contenido irá variando según la opción señalada en el *Panel de Control*. Por defecto aparecerá la lista de socios asociados al mismo centro deportivo al que pertenece el propio monitor. En esta lista se mostrarán los datos principales de los socios.

### 12.3.2.- Registrar un nuevo socio

Un monitor deportivo tiene la posibilidad al igual que los centros deportivos de registrar nuevos socios. Para realizar esta acción debe accederse al formulario de registro de nuevos socios que aparece en la parte inferior de la lista de socios pertenecientes al centro deportivo.

El formulario de registro de nuevo socio que será necesario llenar se corresponde con el explicado en la sección 12.2.4.- *Registrar nuevo socio*. Consultar dicha sección para más detalles.

### 12.3.3.- Modificar la información personal de un socio

Para consultar y/o modificar la información personal referente a un socio perteneciente a nuestro centro deportivo tendremos que pulsar en el icono que aparece a la izquierda del nombre de cada cliente.

El formulario de modificación de datos personales es el mismo que el mostrado en el apartado 12.2.3.- *Modificar información personal de un monitor deportivo*. Los pasos a seguir para completar el formulario son iguales que los explicados en dicho apartado.

#### **12.3.4.- Modificar el perfil de un socio**

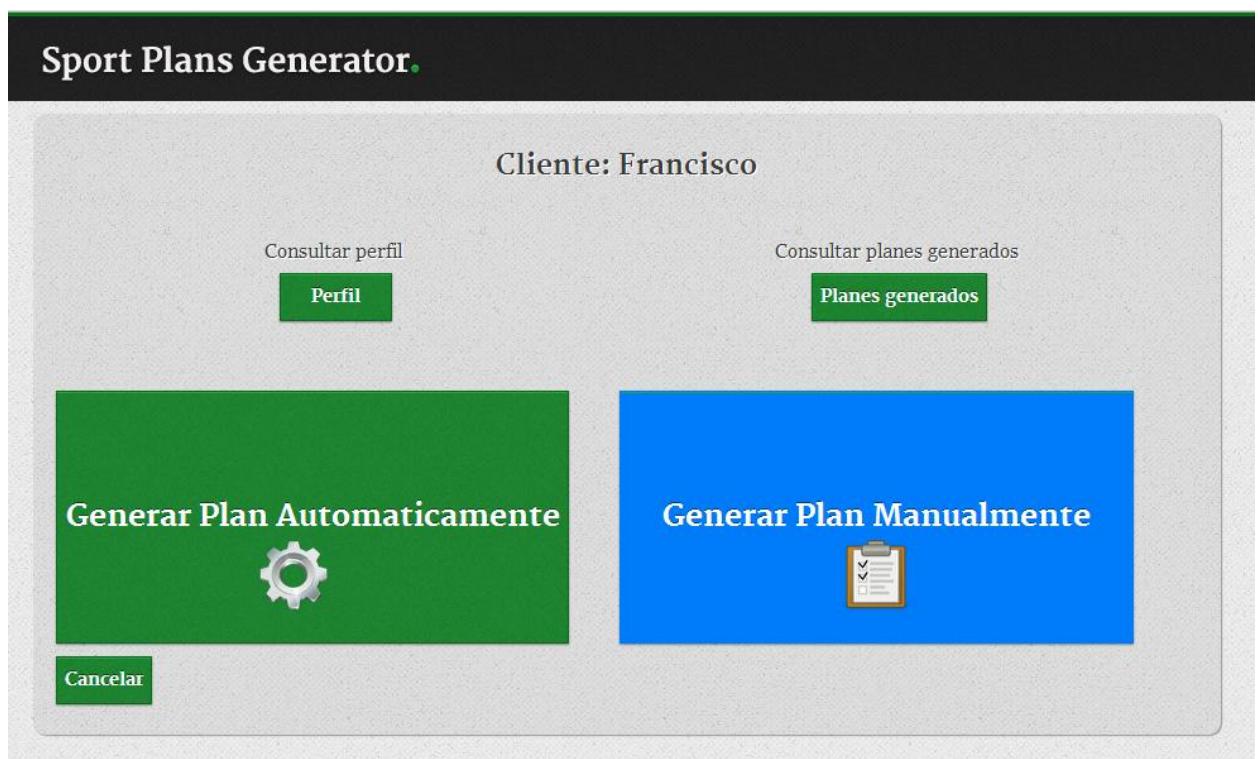
Para poder generar planes deportivos a los socios, necesitamos que éstos tengan previamente su perfil creado. El perfil consta de información básica referente a ese socio, como por ejemplo el peso o la altura.

Para acceder al formulario de edición del perfil de un socio, debemos pulsar en el icono  que aparece a la izquierda del nombre de cada socio.

En el caso de que dicho socio no tuviese un perfil definido con anterioridad, la aplicación nos mostrará directamente el formulario de edición del perfil.

Si por el contrario el socio ya tenía el perfil creado con anterioridad y deseamos modificarlo, deberemos acceder al formulario pulsando en el botón que aparecerá en la parte superior izquierda de la pantalla.

Se puede ver en la siguiente imagen.



**Figura 96:** menú generación de planes deportivos

El formulario de modificación del perfil de un socio es mostrado en la siguiente imagen.

The screenshot shows a user interface titled "Sport Plans Generator." Below it, a sub-titile "Modificar perfil" (Modify profile) is visible. The form contains several input fields and a date picker:

- fecha de nacimiento: 03/09/1988 (Birth date)
- peso(kg): 75.0 kg (Weight)
- altura(cm): 175 cm (Height)
- actividad laboral: Ingeniero Informático (Professional activity)
- nivel: Aclimatación (Level)

Below the input fields is a date picker calendar for March 1988, showing days from 1 to 31. At the bottom of the form are two buttons: "Guardar" (Save) and "Cancelar" (Cancel).

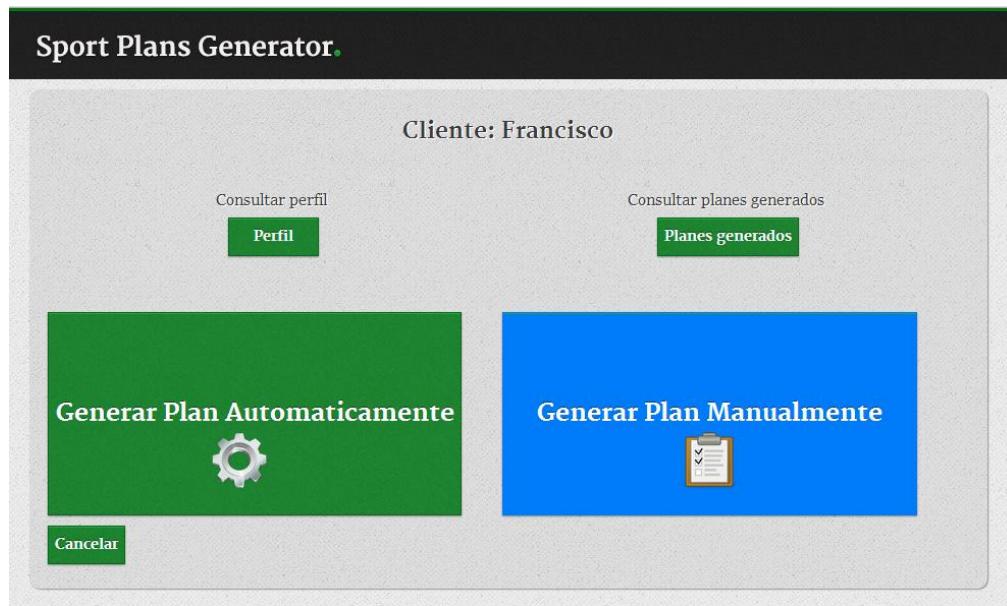
**Figura 97:** formulario de modificación de perfil de socio

Para introducir la edad, la aplicación nos facilita la tarea mostrándonos un calendario donde elegir la fecha.

#### *12.3.5.- Generar un plan deportivo de forma automática*

Para generar un plan deportivo a un socio perteneciente a nuestro centro deportivo, el paso previo que hemos debido realizar es la definición del perfil de dicho socio. Para ello se debe seguir los pasos comentados en el apartado 12.3.4.- *Modificar el perfil de un socio*.

Una vez que el socio tenga el perfil definido, para acceder al generador de planes deportivos de forma automática, se debe pulsar en el botón verde que aparece en la parte inferior izquierda del menú de generación de planes deportivos.



**Figura 98:** menú generación de planes deportivos

Para generar el plan deportivo, simplemente debemos seguir unos sencillos pasos y modificar los valores según consideremos conveniente.

#### *12.3.5.1.- Generar un plan deportivo automáticamente: Paso 1*

El primer paso para la generación automática de un plan deportivo consiste en introducir aquellos valores correspondientes a la duración del plan. También será necesario introducir la cualidad física que queremos entrenar y el objetivo que queremos conseguir.

La aplicación nos facilita la tarea de la creación del plan deportivo sugiriéndonos un valor por defecto para la mayoría de parámetros que son necesarios introducir en el primer paso.

Podemos ver el primer paso para la creación del plan automáticamente en la siguiente imagen.

The screenshot shows a web-based application titled "Sport Plans Generator." at the top. On the right, it displays a welcome message: "Bienvenido carlos" and the date "20 de Agosto de 2013," with a "Close session" link. The main area is titled "Generar Plan Deportivo Automáticamente" and is divided into sections:

- PASO 1**
- Nombre del Plan:** Plan Deportivo 1
- NIVEL:** Aclimatación
- Objetivo:** Adaptación (selected from a dropdown menu which also includes Adaptación, Pérdida de peso, and Mantenimiento)
- Duración:** 1 periodo de 6 semanas
- Cualidad física a entrenar:** Fuerza
- Numero de sesiones a la semana:** 2 sesiones
- TOTAL:** (6 semanas , 12 sesiones )
- Días de entrenamiento:** (Tiempo entre sesiones 48 horas)
  - Lunes (checked)
  - Martes
  - Miércoles (checked)
  - Jueves
  - Viernes
  - Sábado
  - Domingo
- Buttons:** Siguiente (Next) and Cancelar (Cancel).

**Figura 99:** primer paso de la generación automática de un plan de entrenamiento

#### 12.3.5.2.- Generar un plan deportivo automáticamente: Paso2

El segundo y último paso para la generación automática de un plan deportivo consiste en introducir aquellos valores correspondientes a las características que tendrá el futuro plan deportivo, así como el orden o tipo de rutinas a seguir y el número de ejercicios que se realizará en cada sesión.

La aplicación nos mostrará parte de los datos introducidos en el paso anterior a modo de recordatorio y confirmación. Nuevamente nos facilitará la tarea sugiriéndonos valores por defecto para cada dato requerido. Por lo que la tarea de generar un nuevo plan deportivo puede limitarse a comprobar que todos los campos contienen el valor correcto.

En la siguiente imagen mostramos el aspecto que tendrá el segundo paso para la generación automática de un plan deportivo.



**Figura 100:** segundo paso de la generación automática de un plan de entrenamiento

Podemos ver en la imagen mostrada anteriormente como la aplicación nos informa con un mensaje en la parte superior de que todos los cambios realizados en el paso anterior han sido guardados con éxito.

Una vez finalizado este paso, tenemos nuestro plan de entrenamiento generado por completo. En la siguiente pantalla se nos mostrará el plan separado en secciones para poder consultarla de una manera sencilla. Esto será explicado en la siguiente sección.

#### 12.3.5.3.- Consulta del plan deportivo generado

Tras la realización de los pasos anteriormente descritos para la creación de un plan deportivo de forma automática, podemos pasar ya a consultar el plan deportivo generado.

La aplicación nos mostrará la información más destacada en la cabecera de la pantalla a modo de resumen, y como prueba de que todos los datos introducidos son correctos.

En la siguiente imagen podemos ver la pantalla que aparecerá una vez que el plan de entrenamiento ha sido generado con éxito.

The screenshot shows a web-based application for generating training plans. At the top, there's a dark header bar with the text "Sport Plans Generator." on the left, and "Bienvenido carlos" followed by the date "20 de Agosto de 2013" and a "Close session" link on the right.

The main title "Plan de entrenamiento" is displayed prominently in green. Below it, the subtitle "Plan Deportivo 1" is shown. A "PDF" button with a file icon is located to the right of the subtitle.

A message below the subtitle states "Generado por el monitor deportivo Carlos el 20.08.2013" and a link "Como interpretar el plan de entrenamiento".

Below this, there's a table with four rows of data:

Nivel	Aclimatación	Objetivo	Adaptación
Cualidad física a entrenar	Fuerza	Duración total	6 semanas
Realizar los ejercicios a una velocidad lenta			
Ejercicios			

A blue horizontal bar labeled "Ejercicios" spans across the table. Below the table, a white box contains the text "Periodo 1" with a small orange arrow icon to its left.

At the bottom left of the main content area, there's a green button labeled "Volver Menu".

**Figura 101:** consulta de plan deportivo generado

El plan de entrenamiento generado nos aparecerá separado en períodos, semanas y días de la semana, facilitándonos así la consulta de dichos planes.

Para consultar una sesión de entrenamiento en concreto, simplemente debemos navegar a través del periodo al que pertenece pulsando en el nombre del periodo. A continuación se desplegará el contenido de dicho periodo con todas las semanas que contiene.

Podemos verlo con más claridad en la siguiente imagen.

The screenshot shows a software interface titled "Sport Plans Generator". At the top right, it says "Bienvenido carlos" and "20 de Agosto de 2013" with a "Close session" button. Below the title, a blue bar says "Ejercicios". Underneath, a yellow bar says "Periodo 1". The main area is divided into three sections labeled "Semana 1", "Semana 2", and "Semana 3". Each section contains two white boxes: "Sesión Lunes (25 – 20 RM)" and "Sesión Miércoles (25 – 20 RM)".

**Figura 102:** periodo desglosado en semanas

Para consultar el contenido de la sesión de entrenamiento, simplemente pulsaremos en el nombre de la sesión y se desplegarán los ejercicios debajo del nombre de la sesión que estamos consultando.

Podemos ver una sesión en detalle en la siguiente imagen.

The screenshot shows the "Sport Plans Generator" software interface. At the top right, it says "Bienvenido carlos" and "20 de Agosto de 2013" with a "Close session" button. Below the title, a blue bar says "Semana 1". Underneath, a white box lists "Sesión Lunes (25 – 20 RM)" and "Sesión Miércoles (25 – 20 RM)". To the right of "Sesión Miércoles (25 – 20 RM)", a yellow button says "7 ejercicios". Below this, there are three exercise descriptions with images and details:

- Diagonales hacia abajo con cable de rodillas**
  - Músculo: Abdominales
  - Equipamiento: Poleas
  - Número de series: 2
  - Descanso: 45 segundos
- Biceps en banco scott**
  - Músculo: Biceps
  - Equipamiento: Maquinas
  - Número de series: 1
  - Descanso: 45 segundos
- Remo horizontal en barra fija pies apoyados a una mano**
  - Músculo: Espalda
  - Equipamiento: Maquinas

**Figura 103:** ejercicios pertenecientes a una sesión de entrenamiento

Podemos ver el número de ejercicios en total que contiene esa sesión así como cada uno de los ejercicios con su imagen descriptiva a la izquierda. Se muestra además información de cómo debe ser realizado el ejercicio, con qué equipamiento, a qué músculo afecta y el tiempo de descanso que debemos esperar entre cada una de las series.

Se nos ofrecerá la posibilidad también de consultar el plan deportivo generado en formato *PDF*. Para ello sólo deberemos pulsar en el icono  que aparece a la derecha del nombre del plan de entrenamiento generado.

El aspecto que tendrá el fichero *PDF* que nos descarguemos es mostrado en la siguiente imagen.

## Plan de Entrenamiento Deportivo

### Plan Deportivo 1

Generado por el monitor deportivo Carlos el 20.08.2013.

Nivel: Aclimatación  
Objetivo: Adaptación  
Cualidad Física: Fuerza  
Duración: 6 semanas

---

- Ejercicios -

**Periodo 1**

**Semana 1**

**Sesión Lunes**

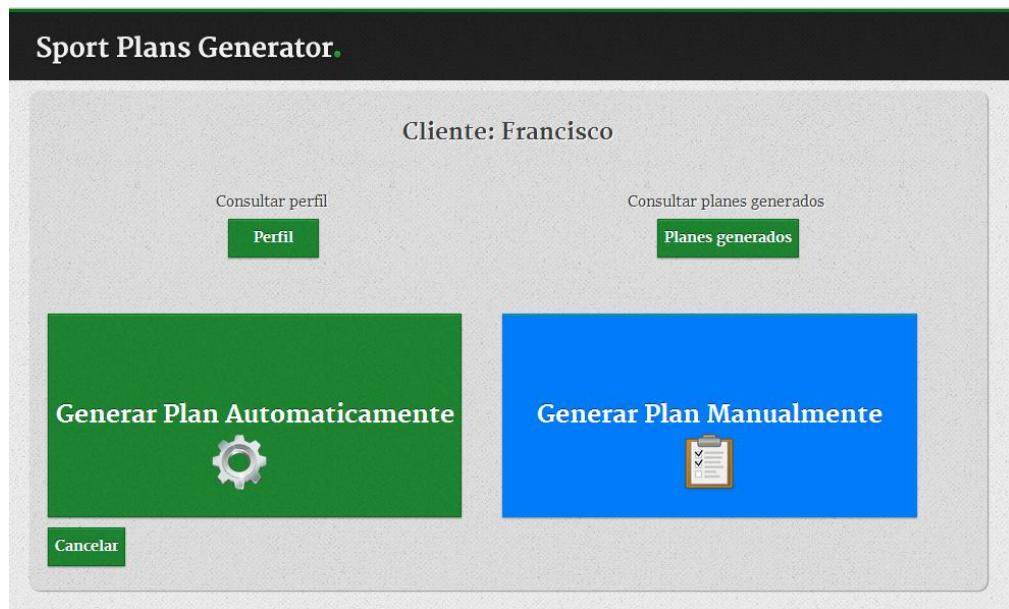
<p><b>Aperturas en contractor</b></p>  <p>Muscle: Pectoral Equipment: Machines Number of sets: 1 set Rest: 45 seconds</p>	<p><b>Encogimientos en máquina</b></p>  <p>Muscle: Abdominal Equipment: Machines Number of sets: 2 sets Rest: 45 seconds</p>
--	---

**Figura 104:** plan de entrenamiento en formato *PDF*

#### **12.3.6.- Generar un plan deportivo de forma manual**

Para generar un plan deportivo a un socio perteneciente a nuestro centro deportivo, el paso previo que hemos debido realizar es la definición del perfil de dicho socio. Para ello se debe seguir los pasos comentados en el apartado **12.3.4.- Modificar el perfil de un socio**.

Una vez que el socio tenga el perfil definido, para acceder al generador de planes deportivos de forma manual, se debe pulsar en el botón azul que aparece en la parte inferior derecha del menú de generación de planes deportivos.



**Figura 105:** menú generación de planes deportivos

Para generar el plan deportivo, simplemente debemos seguir unos sencillos pasos, modificar los valores según consideremos conveniente y seleccionar aquellos ejercicios que queremos incluir en nuestro plan deportivo.

#### *12.3.6.1.- Generar un plan deportivo manualmente: Paso 1*

El primer paso para la generación manual de un plan deportivo es similar al explicado en el apartado 12.3.5.1.- *Generar un plan deportivo automáticamente: Paso 1*. Consiste en introducir aquellos valores correspondientes a la duración del plan. También será necesario introducir la cualidad física que queremos entrenar y el objetivo que queremos conseguir.

La aplicación nos facilita la tarea de la creación del plan deportivo sugiriéndonos un valor por defecto para la mayoría de parámetros que son necesarios introducir en el primer paso.

Podemos ver el primer paso para la creación del plan manualmente en la siguiente imagen.

The screenshot shows the 'Sport Plans Generator' application interface. At the top right, it says 'Bienvenido carlos' (Welcome carlos), '20 de Agosto de 2013' (August 20, 2013), and 'Close session'. The main title 'Generar Plan Deportivo Manualmente' is at the top left. Below it, a large button labeled 'PASO 1' is centered. The form fields include:

- Nombre del Plan:** Plan Deportivo 2
- NIVEL:** Aclimatación
- Objetivo:** Adaptación (selected in a dropdown menu which also includes Adaptación, Pérdida de peso, and Mantenimiento)
- Duracion:** 1 periodo de 6 semanas
- Cualidad fisica a entrenar:** Fuerza
- Numero de sesiones a la semana:** 2 sesiones
- TOTAL:** (6 semanas , 12 sesiones )
- Dias de entrenamiento:** (Tiempo entre sesiones 48 horas)
  - Lunes
  - Martes
  - Miércoles
  - Jueves
  - Viernes
  - Sábado
  - Domingo

At the bottom left are two buttons: 'Siguiente' (Next) and 'Cancelar' (Cancel).

**Figura 106:** primer paso de la generación manual de un plan de entrenamiento

#### 12.3.6.2.- Generar un plan deportivo manualmente: Paso 2

El segundo paso para la generación manual de un plan deportivo es similar al explicado en la sección 12.3.5.2.- *Generar un plan deportivo automáticamente: Paso2.* Consiste en introducir aquellos valores correspondientes a las características que tendrá el futuro plan deportivo, así como el orden o tipo de rutinas a seguir y el número de ejercicios que se realizará en cada sesión.

La aplicación nos mostrará parte de los datos introducidos en el paso anterior a modo de recordatorio y confirmación. Nuevamente nos facilitará la tarea sugiriéndonos valores por defecto para cada dato requerido. Por lo que la tarea de generar un nuevo plan deportivo puede limitarse a comprobar que todos los campos contienen el valor correcto.

En la siguiente imagen mostramos el aspecto que tendrá el segundo paso para la generación manual de un plan de entrenamiento.

The screenshot shows a web-based application titled "Sport Plans Generator". At the top right, it displays a welcome message "Bienvenido carlos", the date "20 de Agosto de 2013", and a "Close session" link. The main title "Generar Plan Deportivo Manualmente" is at the top left. Below it, the section "PASO 2" is labeled. The form contains several input fields and radio buttons:

- NIVEL:** Aclimatación
- OBJETIVO:** Adaptación
- DURACIÓN:** 1 periodos de 6 semanas  
( 2 dias a la semana - 12 días en total )
- Orden de la rutina:**
  - Circuito
  - Series
- Tipo de rutina:**
  - Global
  - Por hemisferios
  - Por grupo muscular
- Ejercicios por sesión:** 7 ejercicios
- Material a descartar:** ( Seleccione aquellos materiales que desea incluir en su plan de entrenamiento )
  - Maquinas

At the bottom are two buttons: "Siguiente" (Next) and "Cancelar" (Cancel).

**Figura 107:** segundo paso de la generación manual de un plan de entrenamiento

Tras este paso, la aplicación nos sugerirá según los parámetros introducidos en los pasos previos, qué ejercicios son los más adecuados para el cliente al que estamos generando el plan. Se tendrán también en cuenta el perfil del socio y el objetivo que desea conseguir.

En el siguiente apartado se muestra con más detalle cómo realizar la selección de ejercicios que se tendrán en cuenta a la hora de generar el plan deportivo.

#### 12.3.6.3.- Generar un plan deportivo manualmente: Paso 3

En este tercer y último paso, el monitor deportivo será el encargado de seleccionar aquellos ejercicios que la aplicación deberá tener en cuenta a la hora de generar el plan deportivo.

Los ejercicios que aparecerán serán seleccionados según el objetivo que se ha marcado el socio y según el nivel de éste.

Para facilitar su localización, serán mostrados ordenados según el grupo muscular al que pertenecen. De esta forma, se facilita la tarea al monitor deportivo que desee eliminar algún ejercicio en concreto ya que se supone que éste conoce a qué grupo muscular pertenece dicho ejercicio.

En la siguiente pantalla podemos ver con más detalle la pantalla de selección de ejercicios.

## Sport Plans Generator.

Bienvenido carlos

20 de Agosto de 2013

[Close session](#)

### PASO 3

Seleccione los ejercicios que desea incluir en el plan.

#### Abdominales

- Diagonales hacia arriba con cable de rodillas



- Diagonales hacia abajo con cable de rodillas



- Encogimientos en máquina

**Figura 108:** tercer paso de la generación manual de un plan de entrenamiento

Una vez seleccionados los ejercicios que queremos incluir en nuestro plan de entrenamiento a generar, confirmamos el formulario y se procederá a la generación del plan.

#### *12.3.6.4.- Consulta de plan deportivo generado*

Tras la realización de los pasos anteriormente descritos para la creación de un plan deportivo de forma manual, podemos pasar ya a consultar el plan deportivo generado.

Para consultar el plan deportivo generado, podemos seguir los pasos descritos en el apartado 12.3.5.3.- *Consulta del plan deportivo generado*.

#### *12.3.7.- Configuración*

El apartado de configuración nos ofrece la posibilidad tanto de cambiar los datos de contacto referentes al monitor deportivo así como la contraseña requerida para el acceso a la aplicación.

Este apartado puede ser consultado en la sección 12.2.7.- *Configuración*.

## 12.4.- Manual de socio

A continuación describiremos las funcionalidades disponibles para los socios registrados en la aplicación *Sport Plans Generator*.

### 12.4.1.- Dashboard

Una vez que el socio accede a la aplicación, tendrá disponible el menú o *Dashboard* de socio. Desde este menú podrá acceder de una manera rápida a todas las funcionalidades disponibles para los socios registrados en la aplicación.

A continuación mostramos el menú o *Dashboard* de socios.

The screenshot shows the 'Sport Plans Generator' dashboard. At the top right, it says 'Bienvenido flopez' (Welcome flopez), '20 de Agosto de 2013' (August 20, 2013), and a 'Close session' button. Below this, the user's name 'Francisco Lopez' and location 'Centro deportivo O2 Sevilla' are displayed. Underneath, it shows 'Nivel: Aclimatación' (Level: Acclimation) and 'Planes generados: 3' (Generated plans: 3). The main content area displays two rows of training plans:

Plan Deportivo	Nivel	Monitor Deportivo
<b>Plan Deportivo 1</b>	<b>Aclimatación</b>	<b>Carlos Martinez</b>
plan1	Aclimatación	Carlos Martinez
<b>Plan Deportivo 2</b>	<b>Aclimatación</b>	<b>Carlos Martinez</b>

**Figura 109:** menú o Dashboard de socio

En este menú o *Dashboard* el socio puede consultar los planes de entrenamiento que le han sido generados.

Se encuentra dividido en dos paneles principales. El panel superior muestra información referente al socio, por ejemplo el número de planes que le han sido generados y el nivel que tiene actualmente en su perfil.

El panel que se encuentra en la parte inferior muestra cada uno de los planes, además se indica el nivel de ese plan y el monitor deportivo que lo generó.

Para consultar cualquiera de los planes que aparecen en el panel inferior, el socio debe pulsar sobre el nombre de uno de ellos y podrá acceder a su contenido.

#### **12.4.2.- Consulta de planes deportivos**

Para consultar el plan deportivo generado, podemos seguir los pasos descritos en el apartado 12.3.5.3.- *Consulta del plan deportivo generado.*

## **Parte IV – Comentarios finales**



## 13.- Conclusiones

Podemos decir que los principales objetivos tanto docentes como técnicos se han conseguido con éxito.

Es de resaltar que nuestro cliente ha quedado satisfecho con el resultado obtenido. Hemos sabido extraer las necesidades del cliente expresadas en las reuniones mantenidas al comienzo del proyecto y plasmarlas de manera adecuada. Se han resuelto con éxito los problemas que aparecían durante la generación de los planes, además de ofrecer nuevas funcionalidades como la automatización en la creación de los propios planes y la gestión de centros y monitores deportivos y sus socios y la posibilidad de ofrecer el plan deportivo en formato *.PDF*.

Hemos aplicado correctamente las técnicas y metodologías estudiadas para la planificación, diseño y planificación de un proyecto software y como resultado hemos obtenido este documento que plasma cada una de las fases. Además hemos adquirido una valiosa experiencia en el desarrollo de proyectos que creemos nos será de gran utilidad en el futuro próximo.

En cuanto a las tecnologías usadas para el desarrollo del proyecto, podemos decir que se ha hecho un uso profesional de ellas, profundizando en cada uno de los frameworks usados para sacar el mayor beneficio de ellas y aumentar la calidad de la aplicación en cuestión. Esto ha favorecido también el aumento de los conocimientos y experiencia del alumno. Se han obtenido conocimientos en tecnologías útiles para realizar la fase de pruebas como son *Selenium* y *JMeter*, y para obtener métricas acerca del código fuente, como es el caso de *SonarQube*.

Es remarcable señalar que gracias a los conocimientos y experiencia adquiridos en las tecnologías usadas, como es el caso de *Spring Framework* e *Hibernate*, el alumno ha podido optar a nuevas ofertas de trabajo a las que antes no tenía acceso al no tener estos conocimientos, y conseguir con éxito un puesto de trabajo en una empresa en la que usa dichas tecnologías (*Ludologic Ltd., Ipswich, Reino Unido*).

En el ámbito personal, podemos decir que la realización de este proyecto nos ha aportado una gran experiencia tanto en el aspecto teórico como en el aspecto técnico, la cual nos será de gran utilidad para el futuro próximo. Cabe destacar también las destrezas desarrolladas en lo que a planificación de proyectos se refiere.

## 14.- Trabajo futuro

En este apartado, hablaremos primeramente sobre el futuro a corto plazo de nuestra aplicación.

Ya que la idea sobre la que yace nuestra aplicación surge de un cliente real, es nuestro deber y nuestro compromiso continuar con el proyecto para seguir mejorándolo y poder lanzarlo al mercado en un futuro próximo.

Para esto, se continuará añadiendo nuevas funcionalidades a la aplicación además de testear y mejorar las ya existentes.

En cuanto a las mejoras en las posibilidades de configuración de la aplicación, se pretende incluir en el futuro el hecho de que las restricciones que cada centro deportivo desea imponer a las reglas de creación de los planes deportivos sea más editable. Es decir, que un centro deportivo pueda crear reglas del tipo: "Si un cliente es asmático, descartar entonces los objetivos de hipertrofia". De esta forma, los planes que sean generados a través de nuestra aplicación serán aún más centralizados en las características y necesidades de cada cliente.

Además de los cambios en lo que a generación de planes se refiere, una de las principales mejoras que el cliente pretende realizar es la inclusión de un módulo de dietética. Aprovechando los estudios que el cliente posee en este campo, se prevee incluir en la aplicación un módulo para la gestión de dietas que se adecuen a los planes de entrenamiento generados, potenciando así los resultados obtenidos tras su realización.

El objetivo es que además de generar un plan deportivo adaptado para los objetivos y necesidades específicas para cada cliente, acompañar dicho plan deportivo con una relación de comidas adecuada a sus necesidades.

El objetivo es poder lanzar al mercado una aplicación *Web* en la que un centro deportivo cualquiera pueda registrarse en unos simples pasos y a partir de ahí, comenzar a generar planes de entrenamiento específicos para las necesidades de sus clientes. Podrán en el futuro también configurar qué ejercicios no quieren incluir en los planes que se generen dentro de su centro además de añadir los suyos propios. Se pretende también ofrecer una versión *Premium*, con características exclusivas y poder así financiar parte del proyecto.

Para poder realizar toda esta tarea, será necesario un nuevo estudio de mercado, que se adapte a la situación real y que esté actualizado teniendo en cuenta otras aplicaciones de este mismo ámbito ya existentes en el mercado.

Será necesario también buscar algún tipo de socio capitalista que asuma los gastos que supondrá sacar este nuevo producto al mercado, teniendo en cuenta la inversión que será necesario realizar tanto en personal, como en equipamiento técnico.

En cuanto a la adición de nuevos módulos a nuestra aplicación, hemos sido concienzudos en este aspecto durante el periodo de diseño del sistema. Se han tenido en cuenta en el diseño realizado ideas que el cliente tenía pensadas realizar en un futuro próximo para que afectasen lo menos posible al sistema actual.

Con todas estas nuevas funcionalidades a añadir y la mejora de las existentes, tendremos lista una aplicación para competir con las existentes en el mercado actual.



# Anexo I – Referencias

## Enlaces

[1] Estudio Retributivo AMETIC

[http://www.conetic.info/Archivos/Descargas/Publicaciones/Estudio\\_Retributivo\\_CONETIC\\_version%20final.pdf](http://www.conetic.info/Archivos/Descargas/Publicaciones/Estudio_Retributivo_CONETIC_version%20final.pdf)

[2] Projectsii

<https://projetsii.informatica.us.es/>

[3] Spring Framework

[3.1] [http://es.wikipedia.org/wiki/Spring\\_Framework](http://es.wikipedia.org/wiki/Spring_Framework)

[4] <http://www.springsource.org/>

[5] <http://picandocodigo.net/>

[6] <http://www.sicuma.uma.es>

[7] Spring Security

[8] <http://carloscacique.blogspot.com.es>

[9] <http://lanyrd.com/2012/springio-2012/smtww/>

[10] <http://davizuco.wordpress.com/tag/spring-security/>

[11] Hibernate

[12] <http://www.hibernate.org/>

[13] <http://es.wikipedia.org/wiki/Hibernate>

[14] JSTL – Java Server Pages Standard Tag Library

[15] [http://es.wikipedia.org/wiki/JavaServer\\_Pages\\_Standard\\_Tag\\_Library](http://es.wikipedia.org/wiki/JavaServer_Pages_Standard_Tag_Library)

[16] <http://java.ciberaula.com>

[17] AJAX – Asynchronous JavaScript and XML

[18] <http://www.monografias.com/>

[19] <http://es.wikipedia.org/wiki/AJAX>

[20] <http://ajaxian.com/archives/>

[21] <http://webdesign.about.com/>

[22] <http://www.tufucion.com/>

[23] <http://www.proyectosbds.com/>

[24] DDR 1.5 – Diseñador de Rutinas

[http://es.fitness.com/forum/threads/77115-DDR-Dise%C3%B1ador-de-Rutinas-\(Hoja-de-C%C3%A1culo\)-HILO-OFICIAL](http://es.fitness.com/forum/threads/77115-DDR-Dise%C3%B1ador-de-Rutinas-(Hoja-de-C%C3%A1culo)-HILO-OFICIAL)

[25] Patrón MVC – Model View Controller

[26] [http://es.wikipedia.org/wiki/Modelo\\_Vista\\_Controlador](http://es.wikipedia.org/wiki/Modelo_Vista_Controlador)

[27] <http://static.springsource.org/spring/docs/2.0.x/reference/mvc.html>

[28] Patrón Plantilla

[29] [http://es.wikipedia.org/wiki/Template\\_Method\\_\(patr%C3%B3n\\_de\\_dise%C3%BAo\)](http://es.wikipedia.org/wiki/Template_Method_(patr%C3%B3n_de_dise%C3%BAo))

[30] *Modelado UML – Unified Modeling Language*

[31] <http://www.lsi.us.es/docencia/get.php?id=6307>

[32] <http://www.lsi.us.es/docencia/get.php?id=6390>

[33] *Diagrama de Despliegue*

[34] [http://es.wikipedia.org/wiki/Diagrama\\_de\\_despliegue](http://es.wikipedia.org/wiki/Diagrama_de_despliegue)

[35] <http://www.lsi.us.es/docencia/get.php?id=6558>

[36] *PageOne CSS Framework*

[37] <http://www.styleshout.com/templates/preview/PageOne10/index.html>

[38] <http://www.styleshout.com/>

[39] *Balsamiq Mockups*

[40] <http://www.balsamiq.com/products/mockups>

[41] *Internacionalización*

[42] [http://es.wikipedia.org/wiki/Internacionalizaci%C3%B3n\\_y\\_localizaci%C3%B3n](http://es.wikipedia.org/wiki/Internacionalizaci%C3%B3n_y_localizaci%C3%B3n)

[43] *Apache Tomcat*

[44] <http://www.tomcat.org>

[45] *MySQL Server*

[46] <http://www.mysql.com>

[47] *Spring Tool Suite*

[48] <http://www.springsource.org/sts>

[49] *AJAX – Asynchronous JavaScript And XML*

[50] <http://es.wikipedia.org/wiki/AJAX>

[51] *JSON – JavaScript Object Notation*

[52] <http://es.wikipedia.org/wiki/JSON>

[53] *SonarQube*

[54] [http://es.wikipedia.org/wiki/Sonar\\_\(software\)](http://es.wikipedia.org/wiki/Sonar_(software))

[55] *Git*

[56] <http://es.wikipedia.org/wiki/Git>

[57] *GitHub*

[58] <http://github.com>

[59] *Cliente Git*

[60] <http://git-scm.com/>

[61] *Subversion*

[62] [http://es.wikipedia.org/wiki/Subversion\\_\(software\)](http://es.wikipedia.org/wiki/Subversion_(software))

[63] *Control de versiones*

[64] [http://es.wikipedia.org/wiki/Control\\_de\\_versiones](http://es.wikipedia.org/wiki/Control_de_versiones)

[65] *W3C – World Wide Web Consultorium*

## Anexo I- Referencias

- [66] <http://validator.w3.org/>
- [67] jQueryUI
- [68] <http://jqueryui.com/>
- [69] JMeter
- [70] <http://es.wikipedia.org/wiki/JMeter>
- [71] Aplicación Web Tomsplanner
- [72] [www.tomsplanner.com](http://www.tomsplanner.com)
- [73] Xculture
- <https://itunes.apple.com/es/app/xculture-fitness/id455654634?mt=8>
- [74] Gym: Guía de Ejercicios
- [https://play.google.com/store/apps/details?id=appinventor.ai\\_nelsOnOs0ri0.MiRutina](https://play.google.com/store/apps/details?id=appinventor.ai_nelsOnOs0ri0.MiRutina)
- [75] Selenium
- [76] <http://docs.seleniumhq.org/>
- [77] SonarQube – Cyclomatic Complexity
- [78] <http://docs.codehaus.org/display/SONAR/Metrics+-+Complexity>

## Referencias Bibliográficas

- [50] Craig Walls, *Spring in Action, Third Edition, 2011*
- [51] Ambler, S.W. et al, *Refactoring Databases: Evolutionary database design, 2006, Addison-Wesley Professional*
- [52] Gamma, E. et al, *Design Patterns: Elements of Reusable Object-Oriented Software, 1994, Addison-Wesley Professional*

## Anexo II – Pruebas realizadas

### Verificación de contenidos

Se muestra en la siguiente tabla cada una de las vistas testeadas y a la derecha el resultado de la prueba.

Checklist	
Vista	Resultado
Login	✓
Registro	✓
Menu Centro Deportivo	✓
Menu Monitor Deportivo	✓
Menu Cliente	✓
Modificar Perfil Cliente	✓
Modificar Credenciales	✓
Generar Plan de entrenamiento	✓

### Verificación de sitios en construcción

Se muestra en la siguiente tabla cada una de las vistas testeadas y a la derecha el resultado de la prueba.

Checklist	
Vista	Resultado
Login	✓
Registro	✓
Menu Centro Deportivo	✓
Menu Monitor Deportivo	✓
Menu Cliente	✓
Modificar Perfil Cliente	✓
Modificar Credenciales	✓
Generar Plan de entrenamiento	✓

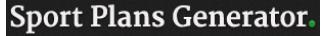
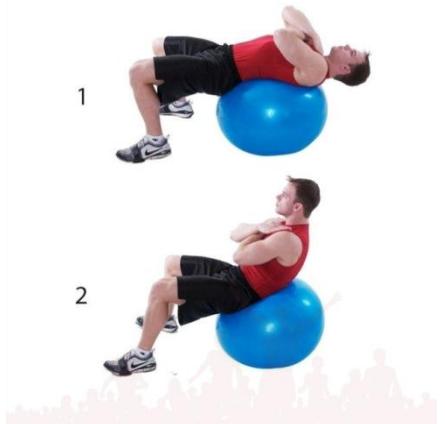
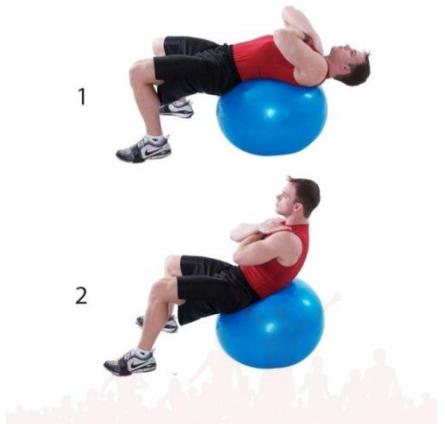
## Imágenes escaladas

Se muestra en la siguiente tabla las imágenes que han sido testeadas y a la derecha el resultado obtenido.

Checklist	
Imagen	Resultado
logo.png 	✓
generator.png 	✓
generarAutomatica.png 	✓
generarManual.png 	✓
pdf.png 	✓
Imágenes ejercicios	
Testeadas las 407 imágenes correspondientes a cada uno de los ejercicios que la aplicación ofrece por defecto. Ejemplo:	
1 	✓
2 	✓

## Imágenes sin atributo ALT

Se muestra en la siguiente tabla las imágenes que han sido testeadas y a la derecha el resultado obtenido.

Checklist	
Imagen	Resultado
logo.png 	✓
generator.png 	✓
generarAutomatica.png 	✓
generarManual.png 	✓
pdf.png 	✓
<b>Imágenes ejercicios</b>	
Testeadas las 407 imágenes correspondientes a cada uno de los ejercicios que la aplicación ofrece por defecto. Ejemplo:	
1 	✓
2 	✓