iPalaestra

José Luis Carretero Galán
ETIG / ETIS

Salvador Campos Mazarico

13/01/2014

A mi madre, porque lleva mucho tiempo esperando esto
A mi tía, porque sé que puede ver esto
A mi mujer, porque lleva demasiado tiempo soportando esto
y a Ultano, porque él me metió en esto

1. Introducción

1.1 Resumen

Dentro de un mundo tan cambiante e informatizado como en el que vivimos existen pocos ámbitos donde no se apliquen las nuevas tecnologías. Este hecho es más remarcable si cabe dentro del mundo de la gestión comercial, dentro del cual la introducción de las mismas ha redundado en una mejora en la calidad y en la velocidad de los servicios. Uno de los campos donde más se ha notado dicho cambio es en la gestión de instalaciones deportivas, los cuales mas que bienes, venden servicios.

Antaño el hecho de acudir a un centro polideportivo significaba rellenar formularios, fichas y carnets de papel para poder acceder al mismo y disfrutar de los distintos servicios e instalaciones que estos lugares ofrecen, con la falta de inmediatez que eso significaba. Si además de esto tenemos en cuenta que, aparte de los usuarios (por así decirlo, los clientes) hay más gente que desarrolla sus actividades en estos centros (imaginemos al pobre profesor manejando distintas listas de papel para saber que alumnos acuden al curso de natación) podemos concluir que la existencia de una herramienta informática que ayude en estas tareas puede de ser de gran utilidad.

Si a todo lo anterior unimos el hecho de poder utilizar la tecnología Java/J2EE y la web, podemos tener, además, una herramienta distribuida que permitirá, a todos los actores que desarrollan su actividad en la instalación deportiva, poder trabajar desde cualquier ordenador del centro (e incluso de fuera de él) sin realizar instalación alguna de ningún programa especifico.

1.2 Objetivos

El principal objetivo de este TFC es realizar una aplicación dentro de un entorno Java/J2EE que permita, de manera distribuida, realizar las tareas básicas de gestión de un centro polideportivo. Se incluyen funcionalidades para los usuarios, los profesores y los gestores del mismo.

1.3 Alcance del proyecto

La aplicación a desarrollar debe permitir gestionar, en primera instancia de una manera básica, un centro polideportivo. Permitirá las siguientes acciones:

- Permitir la inscripción en un curso de los ofertados en el centro.

- Permitir la modificación de una inscripción ya realizada, ya sea para eliminarla o para realizar un cambio en la misma.

- Realizar el alquiler de una instalación del centro.

- Modificar un alquiler ya realizado, ya sea para suprimirlo o para cambiar la hora del mismo.

- Consultar los cursos de un profesor, así como los alumnos inscritos a estos.

- Consultar la asistencia a la última clase realizada de un determinado curso.

- Generar la asistencia para la última clase impartida de un determinado curso.

- Añadir nuevos cursos a la aplicación, modificarlos o eliminarlos.

- Añadir nuevas instalaciones a la aplicación, modificarlas o eliminarlas.

- Añadir nuevas personas (usuarios) a la aplicación, modificarlas, bloquearlas o eliminarlas.

1.4 Enfoque y método seguido

El desarrollo de este proyecto ha seguido las siguientes fases:

1. Investigación y concepción: La principal acción a realizar en esta fase es la de búsqueda y toma de requisitos para el posterior desarrollo, así como el análisis funcional y definición de la estructura básica del proyecto.

2. Desarrollo de la aplicación: El objetivo de esta fase es la confección del código de la aplicación así como una primera batería de pruebas para asegurar el correcto funcionamiento de la misma.

3. Fase de calidad y pruebas: En esta fase se procede a la prueba exhaustiva del aplicativo para encontrar tanto posibles fallos como modos de mejora del funcionamiento del mismo. Aquí también incluimos el tiempo de la realización de las mejoras introducidas.

4. Entrega de la aplicación: El objetivo de esta fase es la entrega y aceptación de la aplicación desarrollada.

1.5 Planificación del proyecto

Para la planificación se han seguido los siguientes pasos:

- Plan de trabajo: Incluye una descripción general, unos objetivos y la planificación temporal del proyecto.
- Maqueta: Diseño de la interfaz gráfica de la aplicación
- Especificaciones y casos de uso: Especificación de todos los roles incluidos en el aplicativo así como de todas las posibles acciones a realizar por estos. También incluye los requerimientos no funcionales y de seguridad.
- Diseño técnico: Incluye la descripción tecnológica del sistema así como los diagramas necesarias para su entendimiento: de casos de uso, de clase de estado, de paquetes…etc.

- Desarrollo: Aquí se adjunta toda la parte de código Java desarrollada para la aplicación así como las instrucciones de instalación y manejo. Será el "alma" de nuestro proyecto.
- Memoria: Documento que incluirá, de la manera más clara posible, todo lo referido en todos los puntos anteriores.
- Presentación: Documento en formato adecuado para poder realizar una presentación resumida del proyecto.
- Preguntas del tribunal: Si se tercia, se incluirá un documento respondiendo las preguntas o dudas presentadas por el tribunal, si las hubiere.

Se puede poner aquí un diagrama de Gantt….

1.6 Productos obtenidos

El producto obtenido es una aplicación web desarrollada con tecnología Java/J2ee destinada a la gestión de centros deportivos polivalentes. El acceso y la ejecución de la misma se podrá realizar desde cualquier navegador web, sin menoscabo en su funcionalidad.

2. Composición del programa

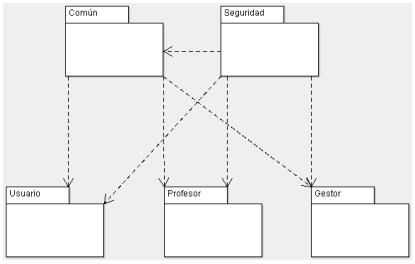
El sistema está compuesto por tres módulos en función del tipo de usuario que utilice la aplicación, uno común a todos estos roles y un último modulo de seguridad:

- Modulo de usuario: Permite al usuario inscribirse en los cursos ofertados en el centro, modificar estas inscripciones, realizar alquileres de instalaciones y modificar dichos alquileres.
- Modulo de profesor: Permite al profesor consultar los cursos donde él imparte clase, los alumnos de las mismas, generar la asistencia a la última clase impartida por él y consultar la última hoja de asistencia generada.
- Modulo de gestor: El gestor será capaz de crear nuevos cursos para el centro y modificarlos, crear nuevas pistas/instalaciones y modificarlas y por último, crear nuevos usuarios y modificarlos.
- Modulo común: Este modulo se pensó para agrupar las funcionalidades comunes a los usuarios, pero de momento, solo gestiona el proceso de acceso a la aplicación.
- Modulo de seguridad: Es el encargado de que ningún usuario entre por ninguna "puerta de atras" del programa. En ese caso, se detecta el caso y se redirige a la pantalla de acceso a la aplicación.

3. Diagrama de paquetes

El sistema se compone de diferentes paquetes de forma organizada que contienen las distintas funcionalidades del mismo y ayudan a su mejor comprensión:

- Paquete de usuario: Contiene las funcionalidades permitidas a un usuario normal del centro deportivo.
- Paquete de profesor: Agrupa las tareas permitidas a los profesores del centro.
- Paquete de gestor: Incluye las operaciones asignadas a los gestores de la aplicación.
- Paquete común: Agrupa funcionalidades comunes a todos los roles del programa.
- Paquete de seguridad: Contiene la clase que gestiona la seguridad de la aplicación



4. Especificaciones de las funcionalidades por modulo

En este apartado describiremos en forma general las funcionalidades de los distintos módulos que componen el sistema.

4.1 Modulo de usuario.

Este modulo contiene las funciones a realizar por el usuario de la aplicación, concretamente:

- Inscripción en cursos
- Modificación o borrado de las citadas inscripciones
- Alquiler de instalaciones del centro polideportivo
- Modificación o borrado de dichos alquileres

4.2 Modulo de profesor.

Este modulo contiene las funciones a realizar por un profesor del centro deportivo, concretamente:

- Consulta de cursos impartidos por ese profesor
- Consulta de alumnos de los citados cursos
- Generación de la asistencia a la última clase de uno de los cursos impartidos por el profesor
- Consulta de la última asistencia generada para un determinado curso impartido por el profesor.

4.3 Modulo de gestor.

Este modulo contiene las funciones a realizar por un gestor de la aplicación, concretamente:

- Creación de nuevos cursos
- Modificación o borrado de los mismos
- Creación de nuevas pistas en la instalación
- Modificación o borrado de dichas pistas.
- Creación de nuevos usuarios de la aplicación.

- Modificación, borrado, bloqueo o desbloqueo de estos usuarios.

4.4 Modulo común.

Pensado para incluir funcionalidades comunes a cada rol, de momento solo incluye el proceso de login a la aplicación.

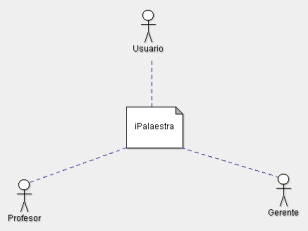
4.5 Modulo de seguridad.

Modulo encargado de gestionar la seguridad en la aplicación. Más concretamente, impide realizar ninguna acción sin haber accedido correctamente a la misma.

5. Casos de uso

5.1 Identificación de los actores

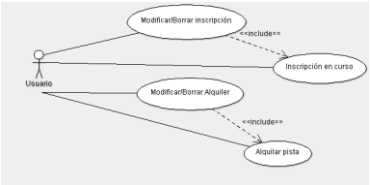
Los distintos actores que pueden interactuar con la aplicación son los siguientes:



- Usuario: Un usuario normal del centro deportivo. Aquel que acude a él a recibir algún curso o clase de alguna modalidad deportiva o similar; o aquel que acude a alquilar una pista deportiva para su posterior disfrute.
- Profesor: Persona que imparte los cursos del centro deportivo y que controla la asistencia a los mismos.
- Gerente: Persona que controla y gestiona los cursos, las pistas y los usuarios de la aplicación.

5.2 Casos de uso por modulo

5.2.1 Casos de uso de usuario



Caso de uso "Alquilar pista"

a. Actor principal: Usuario
b. Precondición: Debe existir, al menos, una pista creada en la instalación
c. Postcondición: ninguna
d. Casos de uso relacionados: Crear pista, Modificar/Borrar alquiler

Escenario principal:

a. El usuario selecciona la opción Alquileres>Nuevo Alquiler
b. El usuario selecciona la pista (entre las existentes en la instalación), la fecha y las horas de inicio y de fin del alquiler
c. El usuario selecciona lo que más le interese y pulsará "Enviar".
d. El sistema marcará dicha hora, pista y fecha como reservadas por el usuario.

Escenarios alternativos:

a. Si esa fecha, hora y pista están reservadas, se mostrará un mensaje de informando del hecho.

Caso de uso "Modificar/Eliminar alquiler"

a. Actor principal: Usuario
b. Precondición: El usuario debe haber alquilado alguna pista y por ende, esta debe existir en la aplicación.
c. Postcondición: Ninguna.
d. Casos de uso relacionados: Crear pista, Alquiler de pista

Escenario principal

a. El usuario elige la opción Alquileres>Modificar Alquiler
b. El sistema muestra los alquileres del usuario.
c. El usuario selecciona el que quiera modificar o eliminar.
d. El usuario elige "Modificar"
e. El sistema vuelve a mostrar el formulario de creación, en donde el usuario modificará lo que más le convenga y pulsará "Enviar".
f. El sistema marcará dicha hora y dicha pista como reservada por el usuario.

Escenarios alternativos:

a. Si esa fecha, hora y pista están reservadas, se mostrará un mensaje de informando del hecho.
b. El usuario sigue los pasos del escenario principal hasta el paso c, allí selecciona "Borrar", borrando dicho alquiler.

Caso de uso "Inscripción en curso"

a. Actor principal: Usuario
b. Precondición: Debe existir, al menos, un curso creado
c. Postcondición: ninguna
d. Casos de uso relacionados: Crear curso.

Escenario principal:

a. El usuario selecciona la opción Cursos>Inscripciones.
b. El sistema muestra los cursos con plazas libres.
c. El usuario selecciona el que más le interese y pulsará "Enviar".
d. El sistema marca al usuario como matriculado en el curso deseado.

Caso de uso "Modificar/eliminar curso"

a. Actor principal: Usuario
b. Precondición: El usuario debe estar matriculado en algún curso y este, por supuesto, debe existir
c. Postcondición: Ninguna.
d. Casos de uso relacionados: Crear curso, Inscripción en curso

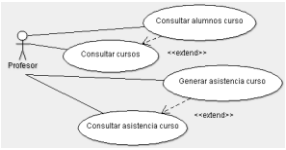
Escenario principal

a. El usuario selecciona la opción Cursos>Modificar Inscripción
b. El sistema muestra las inscripciones del usuario en cursos
c. El usuario selecciona la que quiera modificar o eliminar.
d. El usuario marcará "Modificar".
e. El sistema muestra los cursos con plazas disponibles
f. El usuario selecciona el que más le convenga y pulsará "Enviar".
g. El sistema marca al usuario como matriculado en el curso deseado.

Escenarios alternativos:

a. El usuario sigue los pasos del escenario principal hasta el paso c, allí seleccionará "Borrar", borrando dicha inscripción.

5.2.2 Casos de uso de profesor



Caso de uso "Consultar cursos"

a. Actor principal: Profesor
b. Precondición: El profesor debe tener asignado algún curso y este debe existir.
c. Postcondición: Ninguna.
d. Casos de uso relacionados: Crear curso, Modificar curso.

Escenario principal

a. El profesor selecciona la opción "Cursos".
b. El sistema muestra un listado de los cursos del profesor

Escenarios alternativos

a. El profesor no tiene asignado ningún curso, en ese caso, la pantalla aparecerá vacía.

Caso de uso "Consultar alumnos curso"

a. Actor principal: Profesor
b. Precondición: el profesor tiene que figurar como tal en algún curso.
c. Postcondición: Ninguna.
d. Casos de uso relacionados: Crear curso, Modificar curso.

Escenario principal

a. El profesor ejecuta el caso de uso "consultar curso".
b. El profesor selecciona el curso que más le convenga.

d. El sistema muestra un listado de los alumnos de ese curso.

Escenarios alternativos

a. Si ese curso aún no tiene alumnos, aparecerá la pantalla vacía.

Caso de uso "Consultar asistencia curso"

a. Actor principal: Profesor.
b. Precondición: El profesor debe figurar como tal en algún curso y este debe existir.
c. Postcondición: Ninguna
d. Casos de uso relacionados: Crear curso, Modificar curso.

Escenario principal

a. El profesor selecciona la opción Asistencia>Consultar asistencia.
b. El sistema muestra un listado de los cursos del profesor
c. El profesor selecciona el que más le convenga.
d. El sistema muestra un listado de los alumnos que hayan asistido a la última clase impartida del curso.

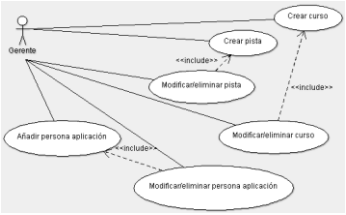
Caso de uso "Generar asistencia curso"

a. Actor principal: Profesor.
b. Precondición: El profesor debe figurar como tal en algún curso y este debe existir.
c. Postcondición: Ninguna
d. Casos de uso relacionados: Crear curso, Modificar curso.

Escenario principal

a. El profesor selecciona la aplicación Asistencia>Generar Asistencia.
b. El sistema muestra un listado de los cursos del profesor.
c. El profesor selecciona el que más le convenga.
d. El sistema muestra un listado de los alumnos del curso.
e. El profesor marca los alumnos que han asistido a la última clase impartida por el profesor (se supone que a medida que pasa el tiempo este listado se renueva una vez haya pasado una fecha donde haya habido clase).
f. El sistema graba la información de asistencia proporcionada para la última fecha en la que hubiese habido clase.

5.2.2 Casos de uso de gerente



Caso de uso "Crear curso"

a. Actor principal: Gerente.
b. Precondición: Ninguna
c. Postcondición: Ninguna
d. Casos de uso relacionados: Ninguno.

Escenario principal

a. El gerente selecciona la opción Cursos>Nuevo curso.
b. El sistema muestra formulario de creación de cursos.
c. El gerente proporciona los datos necesarios para la creación del curso entre los cuales se encuentran:

- Nombre
- Pista
- Fecha Inicio y fin del curso
- Hora Inicio y fin de la clase
- Días en los que se imparte el curso
- Las plazas disponibles
- El profesor que impartirá las clases entre los que existen en la aplicación

d. El sistema registra el curso en la aplicación, a partir de entonces se puede utilizar dentro de él.

Escenarios alternativos.

a. Si hay algún dato faltante o erróneo en el formulario, el sistema lo mostrará.

Caso de uso "Modificar/Eliminar curso"

a. Actor principal: Gerente.
b. Precondición: Que el curso exista
c. Postcondición: Ninguna
d. Casos de uso relacionados: Crear curso.

Escenario principal

a. El gerente selecciona la opción Cursos>Modificar curso
b. El sistema muestra un listado con los cursos activos dentro de la aplicación.
c. El gerente selecciona el curso a modificar.
d. El gerente selecciona la opción "Modificar".
e. El sistema muestra el formulario de creación de cursos con los datos del curso a modificar.
f. El gerente modifica lo que desee.
g. El sistema graba los datos del curso

Escenarios alternativos.

a. El gerente, en cualquier momento, ejecuta el caso de uso
b. El gerente, a partir del paso c, selecciona la opción "Borrar", eliminando dicho curso

Caso de uso "Crear pista"

a. Actor principal: Gerente
b. Precondición: Ninguna
c. Postcondición: Ninguna
d. Casos de uso relacionados: Ninguno

Escenario principal

a. El gerente selecciona la opción Pistas>Nueva Pista
b. El sistema muestra la pantalla de creación de pistas

c. El gerente proporciona el nombre de la pista, que es el único requerido.
d. El sistema registra la pista dentro de la aplicación pudiéndose utilizar en la misma desde ese momento

Escenarios alternativos
a. En caso de que el formulario de creación de pistas contenga datos erróneos o faltantes, el sistema informará del hecho.

Caso de uso "Modificar/Eliminar pista"
a. Actor principal: Gerente.
b. Precondición: Que la pista exista
c. Postcondición: Ninguna
d. Casos de uso relacionados: Crear pista.
Escenario principal
a. El gerente selecciona la opción Pistas>Modificar pista
b. El sistema muestra un listado con las pistas activas dentro de la aplicación.
c. El gerente selecciona la pista a modificar.
d. El sistema muestra los datos modificables de la pista
e. El gerente modifica lo que desee.
f. El sistema graba los datos de la pista
Escenarios alternativos.
a. En caso de que el formulario de creación de pistas contenga datos erróneos o faltantes, el sistema informará del hecho.
b. El gerente, a partir del paso c, elimina la pista, procediendo el sistema a registrar dicho cambio.

Caso de uso "Añadir usuario a la aplicación"
a. Actor principal: Gerente
b. Precondición: Ninguna
c. Postcondición: Ninguna
d. Casos de uso relacionados: Ninguno

Escenario principal
a. El gerente selecciona la opción Usuarios>Nuevo usuario.
b. El sistema muestra la pantalla de creación de usuario.
c. El gerente introduce los datos de la persona, que son:
- Nombre

- Apellidos
- DNI
- Rol
- Activo (indica si el usuario está bloqueado o no).
- Contraseña

Cabe indicar que el usuario nace con el estado "activo" de serie y que los roles son únicos: no se puede tener más de uno a la vez.
d. El sistema registra al nuevo usuario en el programa después de pulsar "Enviar".

Escenarios alternativos

a. En caso de que el formulario de creación de usuarios contenga datos erróneos o faltantes, el sistema informará del hecho

Caso de uso "Modificar/Eliminar persona"

a. Actor principal: Gerente.
b. Precondición: Que la persona exista
c. Postcondición: Ninguna
d. Casos de uso relacionados: Crear persona.

Escenario principal

a. El gerente selecciona la opción Usuarios>Modificar usuario.
b. El sistema muestra un listado con los usuarios de la aplicación.
c. El gerente selecciona el usuario a modificar.
d. El gerente selecciona la opción "Modificar"
d. El sistema muestra el formulario de creación de usuario con los datos del usuario seleccionado.
e. El gerente modifica lo que desee.
f. El sistema graba los datos del usuario después de pulsar "Enviar".

Escenarios alternativos.

a. El gerente, a partir del paso c, selecciona la opción "Borrar", eliminando al usuario de la aplicación.

5.2.4 Casos de uso comunes

Caso de uso "Entrar en la aplicación"

a. Actor principal: Todos
b. Precondición: Que la aplicación esté disponible y arrancada.
c. Postcondición: Ninguna.
d. Casos de uso relacionados: Ninguno.

Escenario principal:

a. El usuario abre un navegador y teclea la dirección de la aplicación: http://servidordelaaplicacion/Palestra
b. Aparecerá la portada de la aplicación en la cual vemos el identificador y la contraseña que necesitamos para entrar en la misma.
c. Introducimos el identificador del usuario (su Dni) y su contraseña.
d. Entramos a la aplicación

Escenarios alternativos:

a. Si se introducen mal los datos de acceso, el sistema informará debidamente del hecho, impidiendo el paso.

6. Requisitos de software y hardware:

Requisitos del cliente:
Básicamente tener conexión a internet (o a la LAN que tenga acceso IP al servidor de la aplicación) y tener un navegador que cumpla los estándares W3C (Firefox, Chrome, Opera o incluso, Explorer.

Requisitos del servidor:
La aplicación se basa en un sistema J2EE construido con Struts2 como gestor de las vistas (jsp fundamentalmente), Spring como gestor de los controladores e Hibernate como gestor de la persistencia.

El servidor de aplicaciones es Tomcat 7 aunque no debería haber ningún problema para migrarlo a cualquier otro, ya que se intentado seguir siempre el estándar de Java.

De manera más detallada podemos apuntar a más componentes utilizados:

- Sistema operativo: Windows 7 Ultimate (cliente), Linux Ubuntu 12 (servidor).
- Base de datos: PostgreSQL 9.
- No se ha usado servidor http, aunque no debería haber ningún problema para usar Apache en cualquiera de sus versiones si se deseare.
- Servidor de aplicaciones: Tomcat 7
- Java J2EE 1.7 JDK
- Maven 2 para la gestión de dependencias del proyecto.
- Desarrollado con el IDE Eclipse Indigo.

7. Seguridad del sistema

Para la gestión de la seguridad del sistema se ha usado la capacidad de Struts2 de operar con interceptores. Estos son una especie de filtros que se pueden ejecutar a voluntad del programador bajo ciertas condiciones. El interceptor que se ha creado para la seguridad de la aplicación rastrea todas las peticiones a struts2 (las actions), si quien intenta ejecutar dicha acción no está logado en la aplicación, dicha petición será rechazada y será redirigido a la pantalla de inicio. Con esto evitamos que un usuario malicioso escriba en la barra de direcciones una determinada acción para poder entrar por una puerta trasera.

Al usar hibérnate se evita el peligro de un ataque por SQL injection, ya que este framework está preparado contra ello.

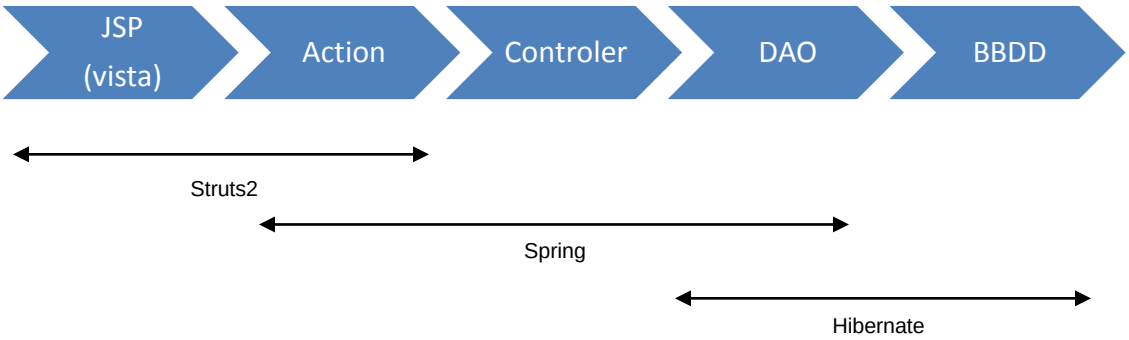
Por último, si en vez de teclear una acción, tecleamos la dirección de una jsp (difícil porque no se muestran en la barra de direcciones) nos veremos otra vez en la pantalla de login: se ha incluido una comprobación de identidad para que no se pueda mostrar ninguna página si no está ese usuario logado.

8. Funciones a incluir en próximas versiones

- Mejora en los alquileres: Pantalla/Calendario que muestre las fechas ocupadas en vez de introducir los datos y que el sistema te avise. Se incluiría tanto en el alquiler como en la modificación del mismo.
- Mejora en la generación de asistencia: Inclusión de un calendario que permita elegir para que fecha queremos generar la asistencia de un curso.
- Mejora en la consulta de asistencia: Inclusión de un calendario que permita consultar una fecha cualquiera de una determinada clase, y no solo la última.
- Mejora del aspecto visual de la aplicación.

9. Arquitectura

La arquitectura de la aplicación se puede resumir esquemáticamente de la siguiente manera:



La versión de Struts2 es la 2.2.3, la de Spring es la 3.0.5 y para Hibernate se ha usado la 3.1.3.

Como puede verse en el gráfico presentado el patrón de diseño seguido ha sido el MVC (modelo vista controlador), de manera más detallada:

- Capa de presentación: Se han utilizado Java Server Pages (jsp) para la interfaz de usuario, mientras que la parte de los action que recogen las peticiones de los usuarios son gestionadas por Struts2.
- Lógica de negocio: Se ha hecho uso del framework Spring que con su paradigma IoC (inversión de control) junto con otros patrones de diseño como los VO (value objects) han servido para el desarrollo de esta capa.
- Persistencia: Se ha hecho uso del framework Hibernate el cual permite un acceso limpio y claro a la base de datos simplificando el manejo de tablas mediante el uso de Data Access Objects (DAO).
- Seguridad: La seguridad se delega un interceptor de Struts2 especialmente creado para tal fin. Dicho interceptor impide accesos no deseados a la aplicación.

9.1 Diseño de la capa cliente

La parte cliente de la aplicación será, básicamente, un navegador web cualquiera (que respete los estándares). Evidentemente este

tipo de arquitecturas nos ahorran una instalación por cada cliente y nos simplifican el proceso general de implantación del aplicativo.

9.2 Diseño de la capa web

La capa web es la encargada de comunicar las peticiones del usuario a la lógica de negocio y a su vez, enviar los resultados de las mismas hacia el cliente para informar del éxito o el fallo de las mismas.

La principal directriz seguida para el diseño de esta capa web ha sido el patrón de diseño modelo vista controlador (MVC).

9.3 Diseño de la capa de lógica de negocio

Gestiona las reglas de negocio de nuestra aplicación, fundamentalmente reglas y entidades que intervienen en la gestión de nuestra entidad (en nuestro caso un centro deportivo).

Para su implementación se ha utilizado el framework Spring el cual, mediante el patrón de inyección de dependencias y los objetos POJO permite suministrar objetos a las clases y no que sean ellas quienes tengan que crear el objeto.

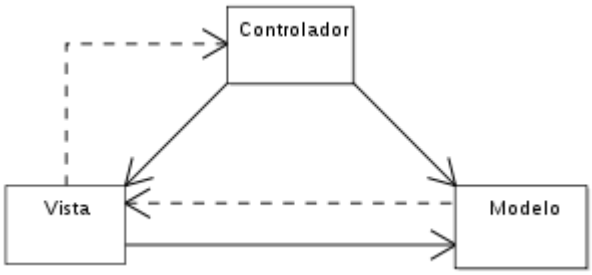
9.4 Diseño de la capa de persistencia

La capa persistencia es la encargada del acceso a la base de datos así como del control de las transacciones para con ella (escrituras, lecturas y borrados, fundamentalmente).

Para su implementación y diseño se ha usado el framework hibérnate el cual permite, mediante mapeos XML y DAOS, manejar las tablas de las base de datos sin necesidad de usar SQL.

9.5 Utilización de patrones

Patrón MVC: Podemos esquematizar este modelo con el siguiente esquema:



Estos componentes de podrían definir de la siguiente manera:

- El Modelo: Es la representación de la información que el sistema utiliza, gestiona todos los accesos a dicha información, tanto consultas como actualizaciones, ejecutando también los privilegios de acceso que se hayan descrito en la lógica de negocio. Envía a la 'vista' aquella parte de la información que en cada momento se le solicita para que sea mostrada. Las peticiones de acceso o manipulación de información llegan al 'modelo' a través del 'controlador'.
- El Controlador: Gestiona las peticiones del usuario y realiza peticiones al 'modelo' cuando se hace alguna solicitud de información. También puede enviar comandos a la 'vista' que tenga asociada si se solicita un cambio en la forma en que se presenta la información, por tanto se podría decir que el 'controlador' hace de intermediario entre la 'vista' y el 'modelo'.
- La Vista: Es la encargada de presentar el modelo de manera que los seres humanos sean capaces de interpretar dicha información.

Patrón IoC (Inversión de control): Es un método de programación que nos permite invertir el flujo de ejecución del programa respecto a los métodos tradicionales de programación.

Mediante la inversión de control se definen respuestas a sucesos o solicitudes de datos determinadas, dejando que algún tipo de entidad externa lleve a cabo las acciones de control que se requieran en el orden necesario y para el conjunto de sucesos que deban ocurrir.

En nuestro caso hemos utilizado el framework Spring para implementar dicho patrón de diseño.

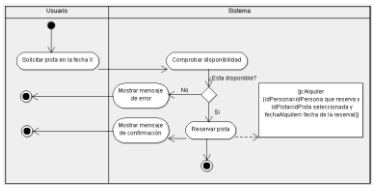
Patrón VO: El patrón Value Object (VO), Transfer Object (TO) e incluso DTO (Data transfer object) nos permite gestionar los datos de una determinada entidad (generalmente una tabla, real o virtual) volcándolos en un objeto construido a semejanza de esta.

El uso de estos objetos viene acompañado del uso de DAO's, que se encargan, generalmente, de la gestión de los VO.

10. Diagramas de actividad

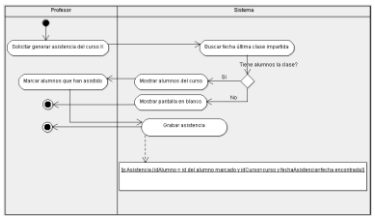
10.1 Alquilar pista

A continuación se muestra el diagrama para la acción "Alquilar pista":



10.2 Generar asistencia

A continuación se muestra el diagrama para el caso de uso "Generar asistencia"

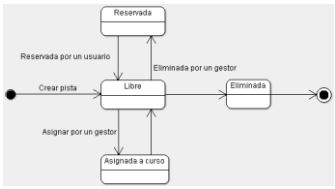


11. Diagramas de estados

11.1 Estados de un curso

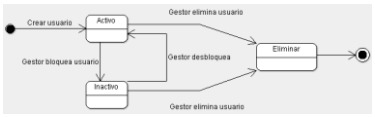
- Activo: El curso esta creado y su periodo de actividad aún está vigente.
- En curso: El curso tiene alumnos asignados, está en periodo de actividad.
- Finalizado: El curso ha sobrepasado su periodo de vigencia o ha sido borrado por un gestor.

11.2 Estado de una pista



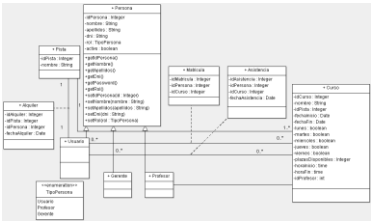
- Libre: La pista está creada y sin ocupar.
- Reservada: Un usuario la ha alquilado.
- Asignada: Un curso ocupa la pista en un determinado tiempo.
- Eliminada: Un gestor la borra, si no está ocupada

11.3 Estado de un usuario



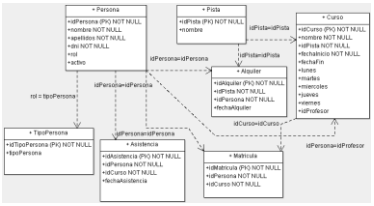
- Activo: El usuario esta creado y puede entrar en la aplicación y utilizarla:
- Inactivo: El gerente lo ha bloqueado y no puede hacer uso de la aplicación pero no está eliminado.
- Eliminado: El gerente borrar al usuario de la base de datos.

12. Diagrama de clases estático



13. Diseño de la persistencia

A continuación se muestra un diagrama de las tablas de BBDD:



14. Valoración económica:

La estimación realizada para el proyecto desde un punto de vista económico se cifra en 17061 euros, IVA incluido.

El detalle se presenta a continuación:

Perfil	Dedicación	Importe/Hora	Importe total	Importe total con IVA (+21%)
Analista funcional	40	45	1800	2178
Analista Programador	120	35	4200	5082
Diseñador gráfico	20	30	600	726
Programador	250	30	7500	9075
Total	430	140	14100	17061

15. Conclusiones:

15.1 Objetivos alcanzados

El objetivo principal era el realizar una pequeña aplicación que sirviese de base para después poder ampliar y añadir más funcionalidades ajustadas a las peticiones del cliente. Cada centro deportivo es un mundo. Donde uno apenas alquila sus instalaciones (un gimnasio privado), otro basa su negocio en este hecho (un polideportivo municipal); cada uno de estos pedirá una expansión hacia de una parte de la aplicación. El hecho de tener una base solida nos permitirá hacerla crecer hacia una solución a medida de cada cliente, lo cual puede redundar en un mayor beneficio económico. A ese respecto se cree cumplido ese objetivo.

15.2 Conclusiones personales

A la conclusión de este trabajo puedo decir que he aprendido más sobre las tecnologías Java en general y de programación de computadores en general. Pero creo que lo que más he aprendido ha sido a gestionar un proyecto desde cero. Hasta ahora me había encontrado con una parte del proyecto empezado: o la idea inicial, o el diseño, el aspecto gráfico… etc. En esta ocasión he sido capataz y obrero y he podido entender la verdadera dimensión y los mecanismos de un proyecto real, lo cual espero que me sirva en un futuro muy cercano.

16. Glosario:

- **Centro polideportivo**: Instalación en la cual una persona ajena a ella puede acudir a la misma a realizar un determinada actividad deportiva que dicho centro facilite.
- **Usuario**: Persona que usa la aplicación desde el punto de vista del usuario de la instalación polideportiva. Realiza las inscripciones a cursos y el alquiler de pistas.
- **Profesor**: Persona que usa la aplicación desde el punto de vista de los profesores de los cursos. Su función principal en la aplicación es gestionar la asistencia a los cursos.
- **Gerente**: Superusuario encargado de la gestión de los principales objetos de la aplicación: personas, cursos y pistas.
- **Curso**: Distintos tipos de clases que se imparten en la instalación polideportiva.
- **Inscripción en curso**: Inscripción que realiza un usuario en una clase determinada.
- **Pista**: Instalación concreta donde se realiza un curso o que es susceptible de ser utilizada por un usuario.
- **Alquiler**: Reserva de una pista por parte de un usuario durante un tiempo concreto

17. Bibliografía:

SCHILDT, HERBERT. Fundamentos de Java. McGraw Hill

LIMA DIAZ, FELIPE. Java 6, Manual avanzado. Anaya multimedia.

FALKNER, JAYSON; GALBRAITH, BEN; IRANI, ROMIN; KOCHMER, CASEY; TIMNEY, JOHN. Desarrollo web con JSP. Anaya multimedia.

ZAKAS, NICHOLAS. JavaScript para desarrolladores Web. Anaya multimedia.

CAMPDERRICH, BENET. Ingeniería del software, UOC (material de la asignatura).

http://www.javatutoriales.com/. Tutorial de Struts2

Anexo 1. Instrucciones de creación y relleno de la base de datos

```
CREATE DATABASE "Palaestra"
  WITH OWNER = postgres
       ENCODING = 'UTF8'
       TABLESPACE = pg_default
       LC_COLLATE = 'Spanish_Spain.1252'
       LC_CTYPE = 'Spanish_Spain.1252'
       CONNECTION LIMIT = -1;

-- Table: tipopersona

-- DROP TABLE tipopersona;

CREATE TABLE tipopersona
(
  idtipopersona serial NOT NULL,
  tipopersona character varying(10),
  CONSTRAINT tipopersona_pkey PRIMARY KEY (idtipopersona),
  CONSTRAINT tipopersona_uk UNIQUE (tipopersona)
)
WITH (
  OIDS=FALSE
);

-- Table: persona

-- DROP TABLE persona;

CREATE TABLE persona
(
  idpersona serial NOT NULL,
  nombre character varying(20) NOT NULL,
  apellidos character varying(30) NOT NULL,
  rol character varying(10) NOT NULL,
  activo boolean NOT NULL DEFAULT true,
  dni character varying(9) NOT NULL,
  password character varying(6) NOT NULL,
  CONSTRAINT persona_pkey PRIMARY KEY (idpersona),
  CONSTRAINT tipopersona_fk FOREIGN KEY (rol)
      REFERENCES tipopersona (tipopersona) MATCH SIMPLE
```

```
      ON UPDATE NO ACTION ON DELETE NO ACTION
)
WITH (
  OIDS=FALSE
);
-- Table: pista

-- DROP TABLE pista;

CREATE TABLE pista
(
  idpista serial NOT NULL,
  nombre character varying(20),
  CONSTRAINT pista_pkey PRIMARY KEY (idpista)
)
WITH (
  OIDS=FALSE
);

-- Table: curso

-- DROP TABLE curso;

CREATE TABLE curso
(
  idcurso serial NOT NULL,
  nombre character varying(20) NOT NULL,
  idpista integer NOT NULL,
  fechainicio date NOT NULL,
  fechafin date,
  lunes boolean DEFAULT false,
  martes boolean DEFAULT false,
  miercoles boolean DEFAULT false,
  jueves boolean DEFAULT false,
  viernes boolean DEFAULT false,
  plazasdisponibles integer NOT NULL,
  horainicurso time without time zone,
  horafincurso time without time zone,
  idprofesor integer,
  CONSTRAINT curso_pkey PRIMARY KEY (idcurso),
  CONSTRAINT curso_idpista_fkey FOREIGN KEY (idpista)
      REFERENCES pista (idpista) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT curso_idprofesor_fkey FOREIGN KEY (idprofesor)
      REFERENCES persona (idpersona) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION
```

```
)
WITH (
  OIDS=FALSE
);

-- Table: matricula

-- DROP TABLE matricula;

CREATE TABLE matricula
(
  idmatricula serial NOT NULL,
  idpersona integer NOT NULL,
  idcurso integer NOT NULL,
  CONSTRAINT matricula_pkey PRIMARY KEY (idmatricula),
  CONSTRAINT matricula_idpersona_fkey FOREIGN KEY (idpersona)
      REFERENCES persona (idpersona) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION
)
WITH (
  OIDS=FALSE
);

-- Table: alquiler

-- DROP TABLE alquiler;

CREATE TABLE alquiler
(
  idalquiler serial NOT NULL,
  idpista integer NOT NULL,
  idpersona integer NOT NULL,
  fechaalquiler date,
  horainialquiler time without time zone,
  horafinalquiler time without time zone,
  CONSTRAINT alquiler_pkey PRIMARY KEY (idalquiler),
  CONSTRAINT alquiler_idpersona_fkey FOREIGN KEY (idpersona)
      REFERENCES persona (idpersona) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT alquiler_idpista_fkey FOREIGN KEY (idpista)
      REFERENCES pista (idpista) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION
)
WITH (
  OIDS=FALSE
);
```

```
-- Table: asistencia

-- DROP TABLE asistencia;

CREATE TABLE asistencia
(
  idasistencia serial NOT NULL,
  idpersona integer NOT NULL,
  idcurso integer NOT NULL,
  fechaasistencia date,
  CONSTRAINT asistencia_pkey PRIMARY KEY (idasistencia),
  CONSTRAINT asistencia_idpersona_fkey FOREIGN KEY (idpersona)
      REFERENCES persona (idpersona) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION
)
WITH (
  OIDS=FALSE
);

-- Datos de prueba

INSERT INTO tipopersona(tipopersona) VALUES ('usuario');
INSERT INTO tipopersona(tipopersona) VALUES ('profesor');
INSERT INTO tipopersona(tipopersona) VALUES ('gerente');

INSERT INTO persona(nombre, apellidos, rol, activo, dni, password)
VALUES ('Juan','Arbós','usuario',TRUE,'12','12');
INSERT INTO persona(nombre, apellidos, rol, activo, dni, password)
VALUES ('Pepe','Perez','gerente',TRUE,'101','101');
INSERT INTO persona(nombre, apellidos, rol, activo, dni, password)
VALUES ('Miguel','Rojo','profesor',TRUE,'10','10');
INSERT INTO persona(nombre, apellidos, rol, activo, dni, password)
VALUES ('Emilio','Coronel','usuario',TRUE,'123','123');

INSERT INTO pista(nombre) VALUES ('campo de futbol');
INSERT INTO pista(nombre) VALUES ('pista de tenis');
INSERT INTO pista(nombre) VALUES ('cancha de baloncesto');
INSERT INTO pista(nombre) VALUES ('sala de juegos');

INSERT INTO curso(nombre, idpista, fechainicio, fechafin, lunes,
martes, miercoles, jueves, viernes, plazasdisponibles,
horainicurso,
            horafincurso, idprofesor)
    VALUES ('Curso nuevo',1,'2013-11-01','2013-12-
30',TRUE,TRUE,TRUE,TRUE,TRUE,14,'09:00:00','10:30:00',3);
```

```
INSERT INTO curso(nombre, idpista, fechainicio, fechafin, lunes,
martes,
            miercoles, jueves, viernes, plazasdisponibles,
horainicurso,
            horafincurso, idprofesor)
    VALUES ('Curso viejo',1,'2012-11-01','2012-12-
30',TRUE,TRUE,TRUE,TRUE,TRUE,14,'10:00:00','10:30:00',3);--Este
curso no está vigente y NO aparecerá
INSERT INTO curso(nombre, idpista, fechainicio, fechafin, lunes,
martes,
            miercoles, jueves, viernes, plazasdisponibles,
horainicurso,
            horafincurso, idprofesor)
    VALUES ('Futbol básico',1,'2013-11-01','2014-02-
28',TRUE,TRUE,TRUE,TRUE,TRUE,14,'10:00:00','11:30:00',3);--Este
último tres es supuesto, si el script se ejecuta correctamente,
ese deberá ser el idpersona de Miguel Rojo.
```

Anexo 2. Manual de instalación.

Requerimientos de software:

En principio no hay ningún impedimento para que le aplicación se ejecute en cualquier servidor. No tiene porque ser un ordenador especialmente potente y en principio no tiene ni porque tener Windows (mi idea primera era meterlo en un Ubuntu Linux pero me ha pillado el toro, así que la explicación es para Windows). Básicamente necesitamos dos componentes que habrá que instalar y estos dos:

- Servidor de aplicaciones Tomcat 7

  El cual es fácilmente descargable en:

  http://tomcat.apache.org/download-70.cgi

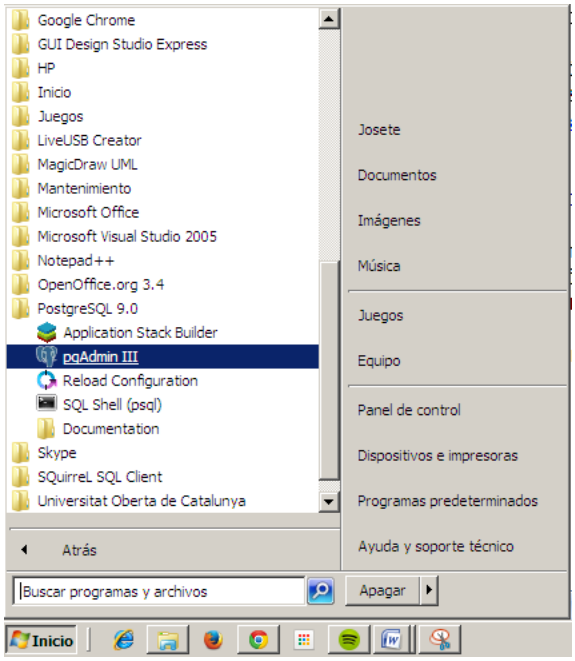
  Cabe apuntar que, aunque se ha desarrollado para esta versión, no hay nada que impida que se ejecute en una versión seis u ocho. Por debajo de esta primera no se garantiza un correcto funcionamiento por la antigüedad de la misma y las posibles incompatibilidades con el código de la aplicación.

- PostgreSQL versión 9.0, la cual está incluida en el material de la UOC (aunque se puede descargar la versión que queramos en http://www.postgresql.org.es/)

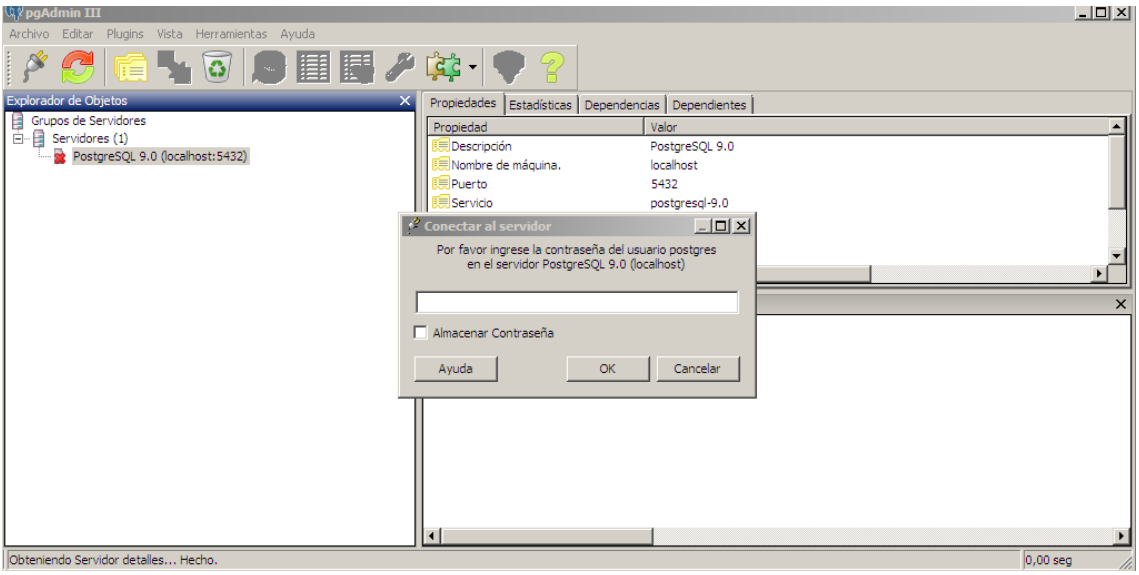
Servidor de base de datos:

No es el objetivo de esta memoria explicar cómo se instala PostgreSQL, pero si vamos a explicar cómo crear la base de datos de la aplicación de una manera sencilla y sin necesidad de terminales:

- El primer paso consiste en ejecutar el pgAdmin III el cual encontraremos con facilidad en el menú inicio de Windows:

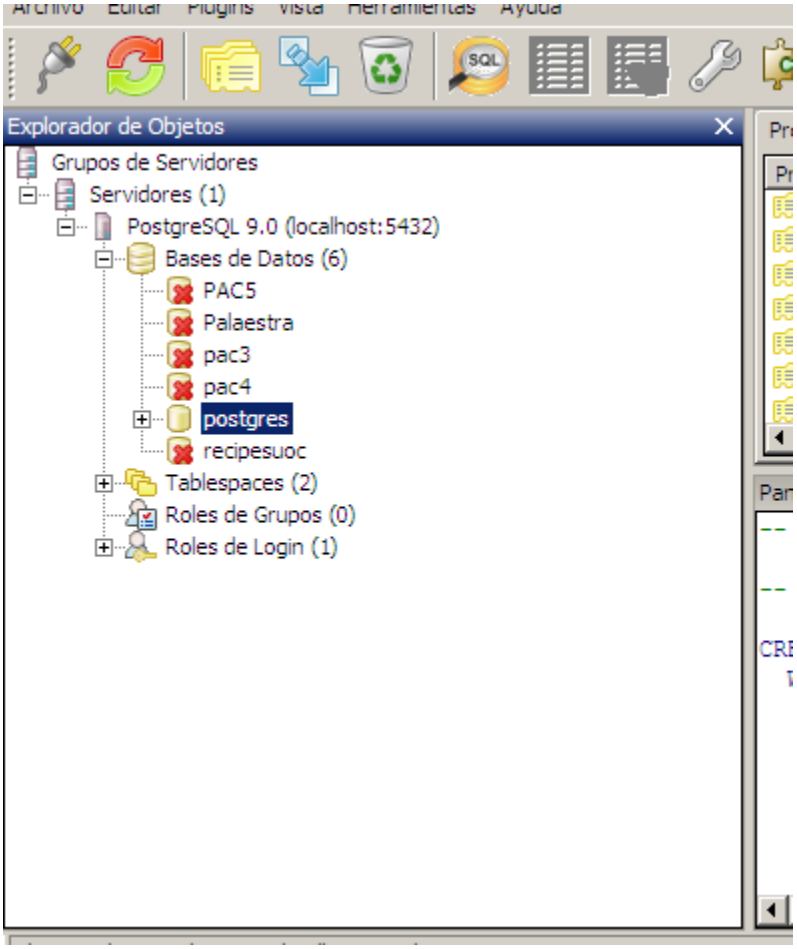


- Una vez dentro hay que conectarse al servidor que hayamos creado, para eso lo seleccionamos y o le damos al botón del enchufe o hacemos doble clic sobre él:

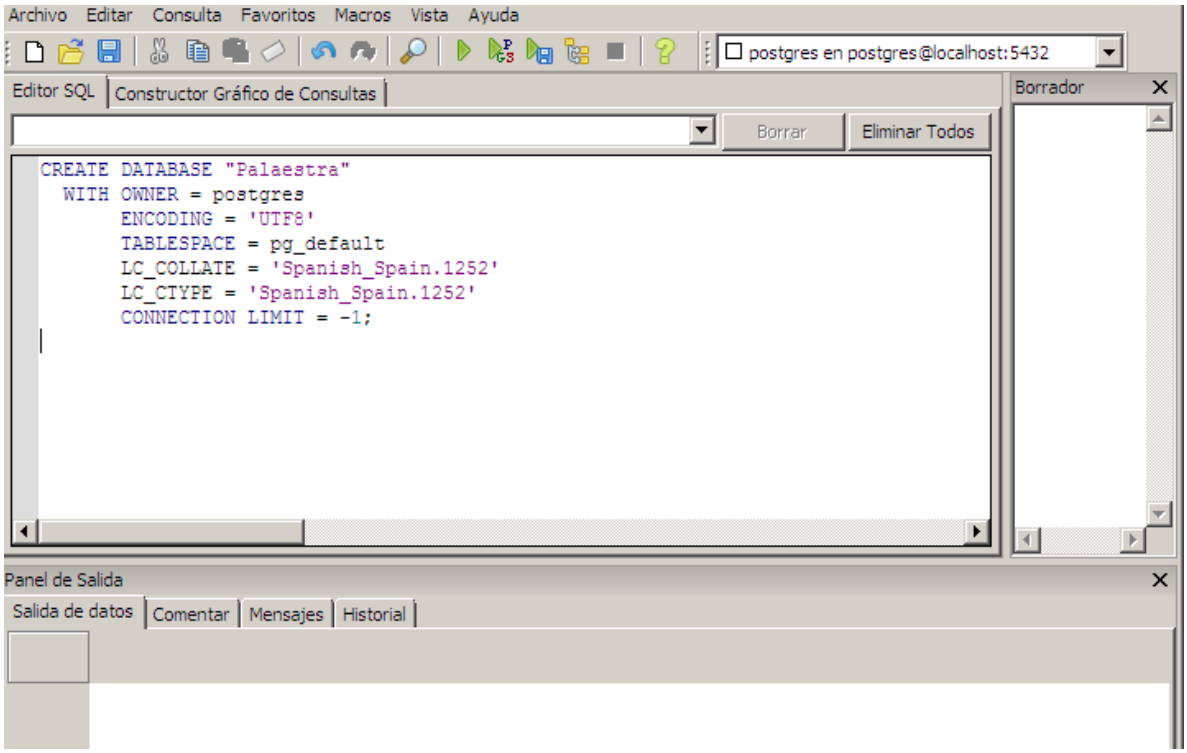


Le ponemos la contraseña que hayamos puesto en el proceso de instalación (en mi caso, es josete, y si no queremos recompilar la aplicación esa debería ser la que hay que poner para que el programa se pueda conectar bien) y entramos en el servidor.

- Una vez dentro, desplegamos "bases de datos" y seleccionamos la "postgres" (es el esquema por defecto). Al hacerlo nos aparecerá activo el botón de SQL, al lado del de la papelera



- Si pulsamos se nos abrirá una ventana para poder introducir mandatos SQL. Aquí dentro podemos pegar los mandatos incluidos en el anexo 1



El orden a seguir sería: primero la sentencia de creación de base de datos (la que vemos en la figura), segundo la creación de tablas siguiendo el orden que aparece en este documento y por último los inserts con los datos de prueba. Para poder ejecutar como un script PL/PGSql hay que pulsar el botoncito con el triangulo verde y las letras PGS (está debajo de Ayuda).

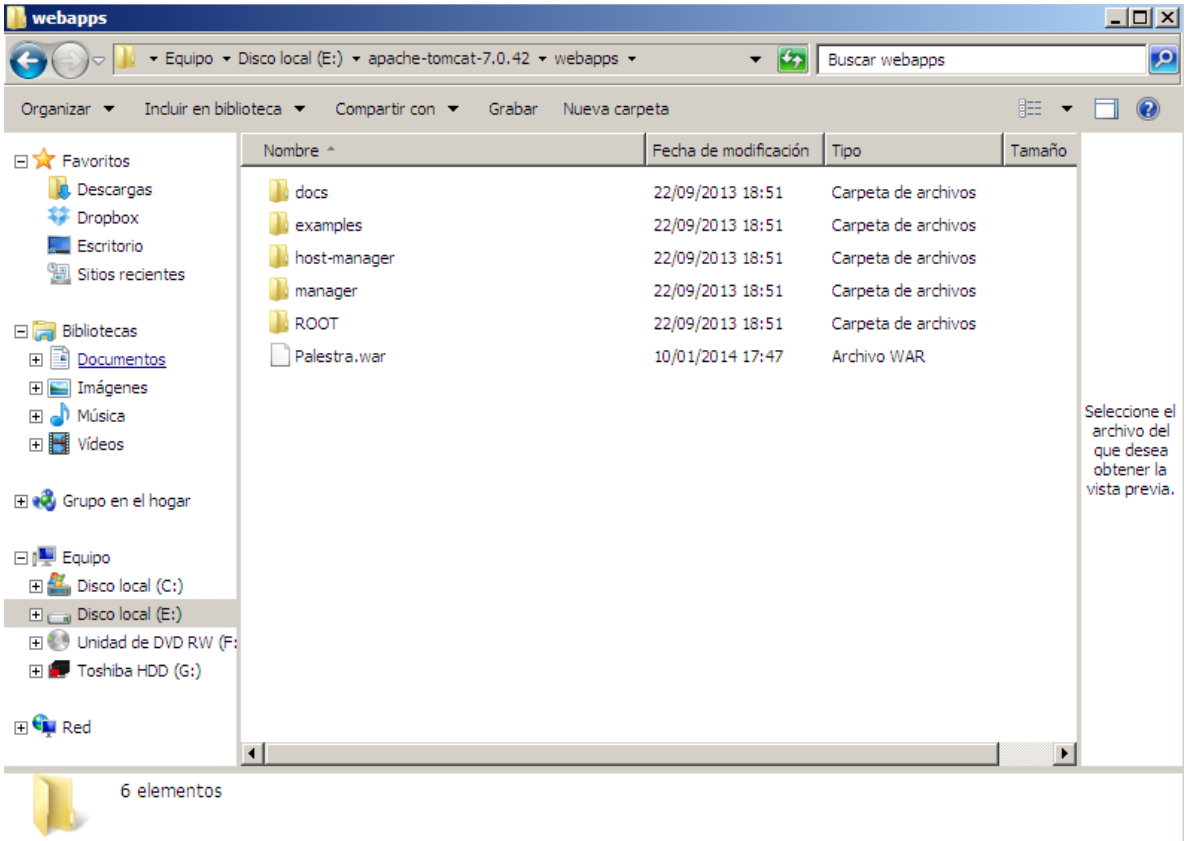
- Si todo va bien, al arrancar la aplicación conectará correctamente con la base de datos y podremos entrar en ella usando cualquiera de las personas de la tabla Persona (el nombre de usuario es el dni y la contraseña la idem).

Servidor de aplicaciones Apache Tomcat

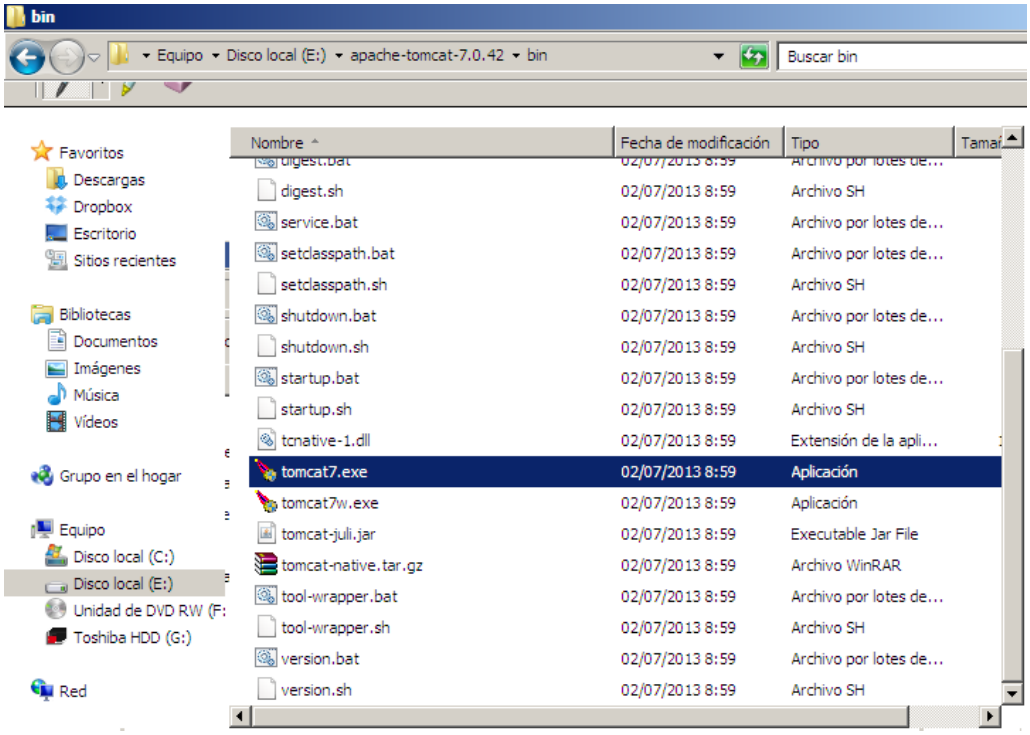
Como en el caso anterior no vamos a explicar cómo se instala este contenedor. Si indicaremos que existen versiones autoejecutables que no nos interesan: funcionan muy bien para usar dentro de eclipse, pero si queremos ejecutar Tomcat en un entorno Windows puro y duro necesitamos el instalador, que, entre otras cosas, registra los servicios de inicio en el sistema operativo.

Lo que explicaremos es donde como se mete nuestra recién creada aplicación dentro del contexto de Tomcat:

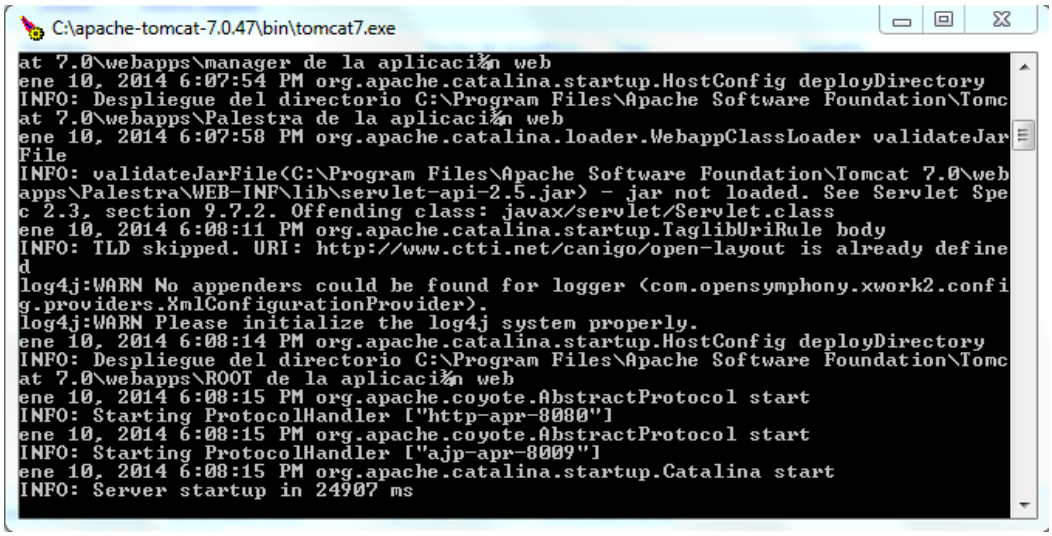
- Primero hay que copiar el fichero Palestra.war que se incluye junto a este documento dentro de \DirectorioDeNuestroTomcat\webapps



- Después iremos a \DirectorioDeNuestroTomcat\bin y allí ejecutamos el fichero tomcat7.exe



Existe la posibilidad de ejecutar el tomcat7w.exe, pero este segundo nos presenta la aplicación en modo Windows y no podremos ver la salida de nuestro programa. Una vez arrancado el servidor, si todo va bien, veremos algo así



Con lo cual ya podremos abrir nuestro navegador para entrar en la aplicación.

Acceso a la aplicación

- Ingresar la siguiente dirección en nuestro navegador

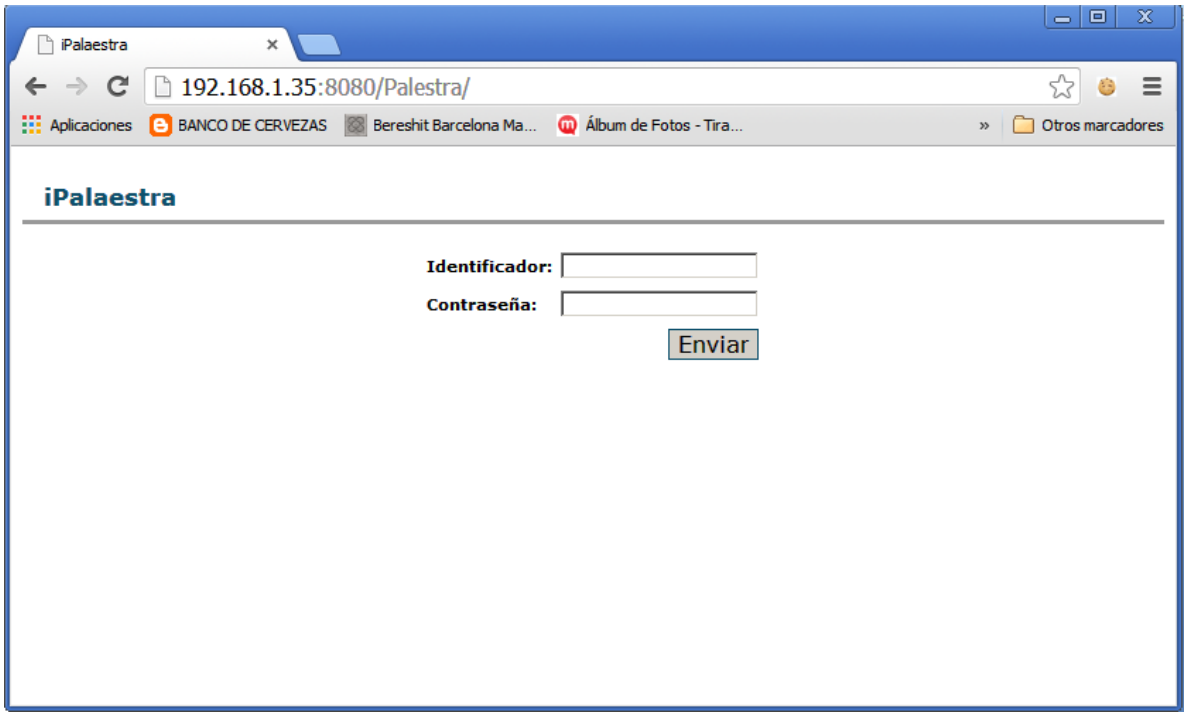
  http://IPDELSERVIDOR:8080/Palestra

  Si estamos en el mismo servidor (lo habitual) lo mejor es poner:

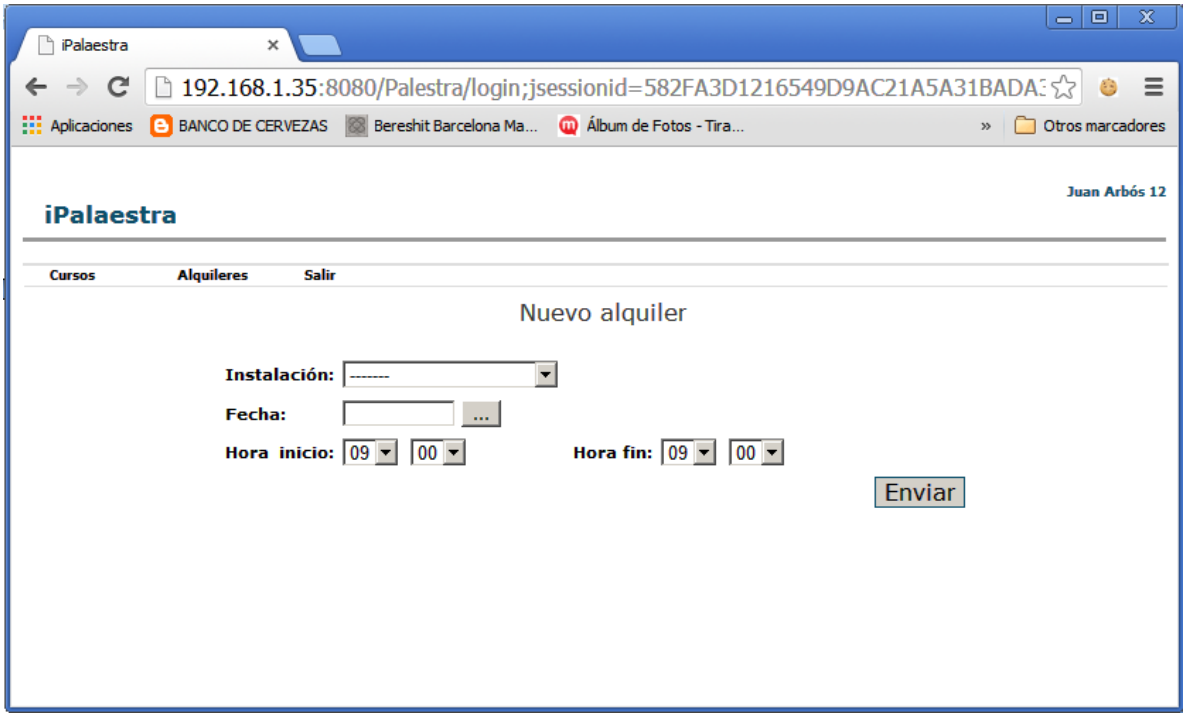
  http://localhost:8080/Palestra

- Una vez en la pantalla inicial usar uno de los usuarios de la tabla Persona para entrar, en función del rol, estos son:

  Usuario: identificador 12 y contraseña idem
  Gerente: identificador 101 y contraseña idem
  Profesor: identificador 10 y contraseña idem



- Una vez dentro solo tendremos que elegir una de las opciones disponibles y empezar a hacer uso de la aplicación



Apuntes finales

- **Acceso a la base de datos**: Si por cualquier circunstancia se hubiesen de modificar los datos de acceso a la base de datos habría que incluir la carpeta adjuntada Palestra dentro de un eclipse y modificar la clase /Palestra/resources/jdbc/jdbc.properties la cual contiene, mas o menos estos campos a modificar:

```
hibernate.dialect=org.hibernate.dialect.MySQL5Dialect
jdbc.driverClassName=org.postgresql.Driver
jdbc.url=jdbc:postgresql://localhost:5432/Palaestra
jdbc.host=localhost
jdbc.port=5432
jdbc.dbname=Palaestra
jdbc.username=postgres
jdbc.password=josete
```

Aquí podremos cambiar el username y password, la ubicación del servidor… etc. Despues solo hay que volver a generar un

war importando el proyecto como un fichero de ese tipo y ya lo tendríamos.

- **Logs de la aplicación**: En caso de errores u otras eventualidades, dentro del servidor, habrá una carpeta en C:\PalestraLogs (se crea si no existe) donde tendremos un log diario del programa. Aquí podemos consultar, por día de ejecución, los logs de la aplicación.