Diseño de un sistema de gestión y asignación de equipos para una empresa

J2EE – TFC

Autor: José Ángel Pardillo Vela
Ingeniería técnica en informática de sistemas

Consultor: Oscar Escudero Sanchez

Índice de contenido

Introducción.

El proyecto consistirá en un sistema de gestión de los equipos pertenecientes a un departamento o a una empresa.

Motivación y necesidades

En la actualidad, la metodología de trabajo en grandes empresas (banca, telecomunicaciones, eléctricas, etc), llega consigo que una gran cantidad de personas formen parte del mismo departamento. Y esto implica que nos encontremos con la situación de que en la propia empresa la información de los recursos disponibles se encuentre dispersada y difícil de controlar para su gestión.

Otras veces, se realizan auditorías de software y hardware con el fin de conocer el estado de las máquinas y departamentos: control de instalación de programas “piratas” o no corporativos, funcionamiento correcto de cada uno de los equipos, instalación de actualizaciones y control de los equipos asignados a cada departamento (esto cobra más importancia cuando los equipos no son físicos, si no que se trata de máquinas virtuales que pueden estar asignadas durante periodos de tiempo necesarios para la realización de ciertas tareas).

Todas estas tareas entran dentro de la metodología de la gestión de los recursos y ponen de manifiesto la necesidad de tener un software que pueda dar cabida a esta gestión. Este software debe ser accesible desde cualquier punto de la organización donde se necesite (siempre con determinados permisos), con lo que una aplicación web se muestra como la mejor opción para la implementación del mismo, dadas sus características. Y para ello se utilizará la tecnología J2EE.

Objectivos generales y específicos del TFC.

A continuación se exponen los objetivos del trabajo realizado.

Objetivos personales.

Como finalidad última de este trabajo está obtener el conocimiento de la tecnología J2EE a través del diseño de una aplicación WEB con la misma. Con lo que el objetivo principal es obtener una sólida base sobre como diseñar una aplicación basada en esta tecnología:

http://java.sun.com/javaee/6/docs/tutorial/doc/docinfo.html
http://java.sun.com/javaee/6/docs/tutorial/doc/bnbqa.html

Objetivos técnicos.

Se trata de dar solución a las necesidades indicadas a través de J2EE. Con lo que para ello se ha priorizado la utilización de las últimas herramientas basadas en el estándar. Entre estas herramientas cabe destacar el framework de la capa de presentación (JSF 2.0) y el uso de JDBC para la capa de persistencia de los datos.

1. La capa de presentación

Se estudiaron una serie de diferentes frameworks. Para ello se hizo uso del amplio catálogo de trabajos de fin de carrera (TFC) de la biblioteca de la UOC . Unos ejemplos de los trabajos consultados son:

Gestión de expedientes para ayuntamientos
J2EE MVC
PFC : Diseño de un Framework de presentación
Diseño e implementación de un marco de trabajo de presentación para aplicación J2EE

Con ello se pudo ver las ventajas y desventajas de usar uno u otro, o ninguno de ellos. Hoy en día la potencia y versatilidad que ofrecen los principales frameworks hace que el desarrollo de una aplicación web sea mucho más sencillo y potente con el uso de los mismos. Es por ello que se realizó un estudio previo a través de los TFC de los tres más conocidos, Structs, Spring y JSF, optando por realizar la implementación con este último debido a las siguiente razones:

- Forma parte del propio estándar de J2EE.
- Está cada vez más extendido.
- Basado en el patrón modelo – vista – controlador.

Este framework, al contar con componentes para la interfaz gráfica de usuario (GUI), asemeja el diseño de la aplicación web a lo que es una aplicación de escritorio:

http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=IntroduccionJSFJava

Con este framework nos aseguraremos separar la parte de presentación del negocio, utilizando así mismo para el desarrollo de las páginas plantillas basadas en facelets:

http://facestutorials.icefaces.org/tutorial/facelets-tutorial.html
http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=faceletsEclipse
http://www.danilat.com/weblog/2007/07/06/empezando-con-facelets/
http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=migrateJSF2Facelets
http://refcardz.dzone.com/refcardz/core-java?oid=ban00025-0

Lo que nos ayudará a reducir la duplicidad del código y hacer una presentación más homogénea de la web.

2. La capa de persistencia

Una vez realizado el estudio de las herramientas para la capa de presentación, podemos pasar a ver otro de los pilares de la aplicación, como es el caso de la capa de persistencia. Para ello existen diferentes soluciones como es directamente utilizar JDBC, Hibernate o JPA. Se realizaron diversas pruebas para ver las ventajas e inconvenientes de utilizar frameworks como Hibernate o JPA frente a JDBC en una aplicación como la que se ha realizado:

- La aplicación hace un uso constante de conexiones a bases de datos (BBDD) tanto para consultas como para actualizaciones e inserciones.
- Salvo en momentos puntuales en los que la conexión deba mantenerse durante un periodo de tiempo largo, la mayoría de las acciones serán con el objetivo de recuperar una cierta cantidad de datos.
- Para las pruebas con framework se utilizó JPA, ya que forma parte del propio estándar J2EE. Es un framework versátil que permite que los datos se guarden tanto en BBDD como en ficheros en disco directamente y utiliza su propio lenguaje para hacer consultas. En contrapartida podemos observar que el tiempo para la conexión con la BBDD es alto, esto es debido a las comprobaciones que se ejecutan al realizar la conexión con la BBDD.
- Para las pruebas con JDBC utilizamos, además, procedimiento almacenados en la BBDD, con lo que se consiguió optimizar la velocidad de consulta y a lo que se sumó el bajo tiempo de conexión a la BBDD que necesita.

Debido al tipo de consultas que se realizarán y a que el diseño de la BBDD formará también parte importante de la realización de proyecto, se optó por crear un gestor de BBDD directamente a través de JDBC, procedimientos almacenados y configuración de las conexiones a través de ficheros XML.

3. UML

Para estructurar correctamente la aplicación, es necesario, al menos en una primera etapa, apoyarse en una herramienta para el diseño del software, para ello se utilizará Magic Draw. Con esto se diseñará el modelo de datos, las clases y los diagramas más representativos de las funcionalidades.

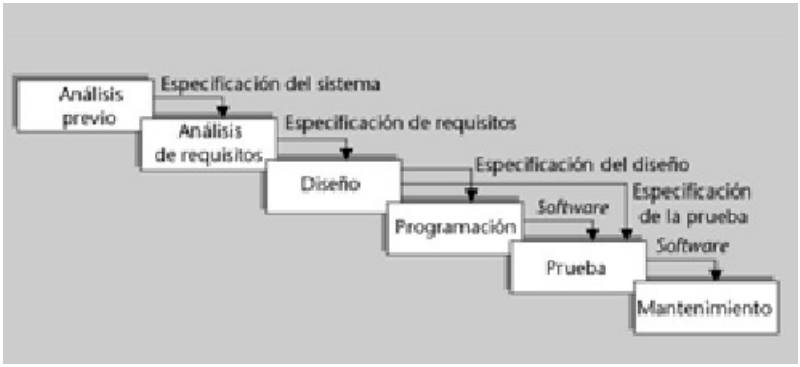
4. Objetivos funcionales

Como resumen de lo expuesto hasta el momento, en una primera aproximación, la aplicación contará con las siguiente funcionalidades:

- Gestión de la propiedad de cada equipo, tanto si es virtual como si es un equipo físico.
- Información de las tareas que se realizan en cada equipo.
- Control de la información de la plataforma del equipo.
- Niveles de permisos para la gestión y visualización de la información.

Metodología de desarrollo.

Como primera aproximación, se siguió para el desarrollo del software un ciclo de vida en cascada tal y como se puede ver en la figura siguiente:



Dado el enfoque del proyecto, hay que añadir al mismo una etapa de análisis de la tecnología a utilizar y pruebas a realizar en la misma. Esta etapa está definida en todo el ciclo de vida de la aplicación.

Posteriormente se siguió un ciclo de vida en cascada, donde cada una de las etapas se iba realimentando tanto con las anteriores como con las siguientes (junto con la etapa de análisis de la tecnología).

Como método de desarrollo se optó por obtener para las grandes bases del proyecto, persistencia, interfaz gráfica, controlador y negocio, un entorno reutilizable tanto en la realización de nuevas páginas como en cualquier tipo de proyectos.

Planificación.

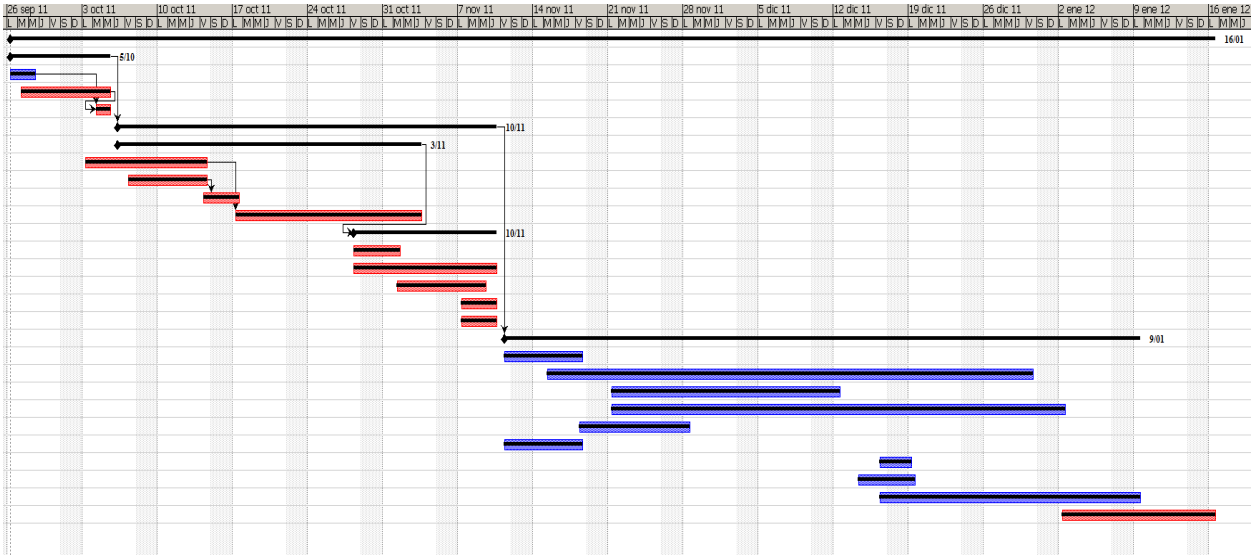
Se realizó una primera aproximación de la planificación basada en las entregas puntuales que se debían realizar a lo largo del semestre. Con esto se establecieron unos primeros plazos de entrega:

- Etapa previa. Hasta el 5 de Octubre de 2011. Elección del proyecto a realizar y presentación de una primera aproximación del plan de trabajo.
- Etapa de análisis y diseño (desde el 6 de Octubre hasta el 10 de Noviembre):
  - Estudio y consenso de las tecnologías a utilizar y finalización de la planificación.
  - Análisis y diseño. Se irán realimentando entre ellas junto con parte de la implementación. También se realizará en diseño del modelo de datos.
- Etapa de implementación (desde el 11 de Noviembre al 19 de Diciembre). Se terminará de implementar el modelo de datos y la propia aplicación con todas las funcionalidades requeridas.

Una vez establecida esta primera planificación, y pasada la etapa de estudio de las tecnologías a utilizar, se obtiene el siguiente plan para los meses de desarrollo:

	Nombre	Duración	Inicio	Terminado	Predecesores
1	TFC-J2EE-Realizacion	81 days	26/09/11 8:00	16/01/12 17:00	
2	Elección de proyecto y tecnologías	8 days	26/09/11 8:00	5/10/11 17:00	
3	Elección de proyecto	3 days	26/09/11 8:00	28/09/11 17:00	
4	Estudio de tecnologías	7 days	27/09/11 8:00	5/10/11 17:00	
5	PEC 1. Redacción	2 days	4/10/11 8:00	5/10/11 17:00	3;4
6	Análisis y diseño	26 days	6/10/11 8:00	10/11/11 17:00	2
7	Pruebas de las tecnologías a utilizar	21 days	6/10/11 8:00	3/11/11 17:00	
8	Pruebas JSF 2.0	10 days?	3/10/11 8:00	14/10/11 17:00	
9	Pruebas JPA	6 days	7/10/11 8:00	14/10/11 17:00	
10	Pruebas JDBC	2 days	14/10/11 8:00	17/10/11 17:00	9
11	Pruebas ICEFaces 2.0	14 days	17/10/11 8:00	3/11/11 17:00	8
12	Análisis y Diseño previos	10 days	28/10/11 8:00	10/11/11 17:00	7
13	Diseño del Gestor de BBDD	3 days	28/10/11 8:00	1/11/11 17:00	
14	Diseño del modelo de paginacion y visualizacion de tablas	10 days	28/10/11 8:00	10/11/11 17:00	
15	Diseño del controlador de páginas	6,875 days	1/11/11 9:00	9/11/11 17:00	
16	Diseño del modelo de datos	3,875 days	7/11/11 9:00	10/11/11 17:00	
17	PEC 2. Redacción	3,875 days	7/11/11 9:00	10/11/11 17:00	
18	Implementacion	41,875 days	11/11/11 9:00	9/01/12 17:00	6
19	Diseño de la interfaz gráfica de usuario	5,875 days	11/11/11 9:00	18/11/11 17:00	
20	Implementacion de la interfaz	33,875 days	15/11/11 9:00	30/12/11 17:00	
21	Implementación del modelo de tablas en la GUI	15,875 days	21/11/11 9:00	12/12/11 17:00	
22	Implementación del modelo de datos	30,875 da...	21/11/11 9:00	2/01/12 17:00	
23	Implementación del controlador de páginas	6,875 days?	18/11/11 9:00	28/11/11 17:00	
24	Implementación del gestor de BBDD	5,875 days	11/11/11 9:00	18/11/11 17:00	
25	Instalación en un servidor local	1 day	16/12/11 9:00	19/12/11 9:00	
26	PEC 3. Redacción	3,875 days	14/12/11 9:00	19/12/11 17:00	
27	Pruebas	16,875 days	16/12/11 9:00	9/01/12 17:00	
28	Entrega final. Memoria	10,875 da...	2/01/12 9:00	16/01/12 17:00	

Que tiene el correspondiente diagrama de Gantt:



Como se indicó anteriormente, las etapas se fueron realimentando las unas con la información de las demás, sean previas a las mismas o no, con lo que este diagrama muestra de una forma general el desarrollo del proyecto y los momentos en los que se llevó a cabo la mayor parte de la tarea especificada.

Productos obtenidos.

Como resultado del desarrollo, se obtuvieron los siguientes productos:

- Hostmanager.war: Fichero WAR con el código fuente y compilado para poder ser instalado en el servidor.
- SQLScripts.zip: Fichero con los correspondientes scripts para la creación de la BBDD, tablas y funciones.
- Memoria del proyecto: PardilloVela_JoseAngel_TFC_Memoria.pdf
- Presentación: PardilloVela_JoseAngel_TFC_Presentacion.pps.

Especificación y requerimientos.

Información previa

La empresa está formada por diferentes departamentos, todos ellos tienen acceso a la misma red y hay tanto equipos físicos en cada puesto de trabajo como equipos virtuales a los que se accede mediante aplicaciones del tipo “escritorio remoto”.

Los equipos virtuales tienen su propia subred y dominio, así como los equipos físicos. Con esto se busca que existan diferentes tipos de permisos para cada uno de ellos. Los equipos físicos pueden conectarse a internet, mientras que los virtuales, que están diseñados únicamente para tareas específicas, sólo tienen acceso a la intranet de la empresa.

Los equipos físicos están asignados a una persona en concreto, que pertenece a un departamento determinado, mientras que los virtuales, están asignados directamente a un departamento. Los usuarios no tienen permisos de administrador en ninguno de los equipos, pero tienen permisos de lectura/escritura en una unidad de disco particular (unidad D).

Cada equipo tiene una determinada plataforma de software, corriendo todos ellos bajo Windows.

Habrá dos perfiles para la gestión de los equipos: administrador y gestor de sistemas del departamento. El gestor de departamento podrá, como principales funciones para los equipos asignados al mismo:

- Logarse en la aplicación.
- Generar un informe (listado) de los equipos asignados.
- Tener acceso información referente a los equipos que están únicamente en su propio departamento. La información fundamental que podrá ver es la siguiente:
    - Nombre, IP y tipo de equipo (virtual o físico).
    - Si el equipo es físico, información de la persona a la que está asignado.
    - Si el equipo es virtual, finalidad del mismo y el departamento al que está asignado.
    - Renovar Ips del equipo.

Si se trata del usuario administrador, además de la información anterior, podrá acceder a los siguientes servicios de cualquier equipo:

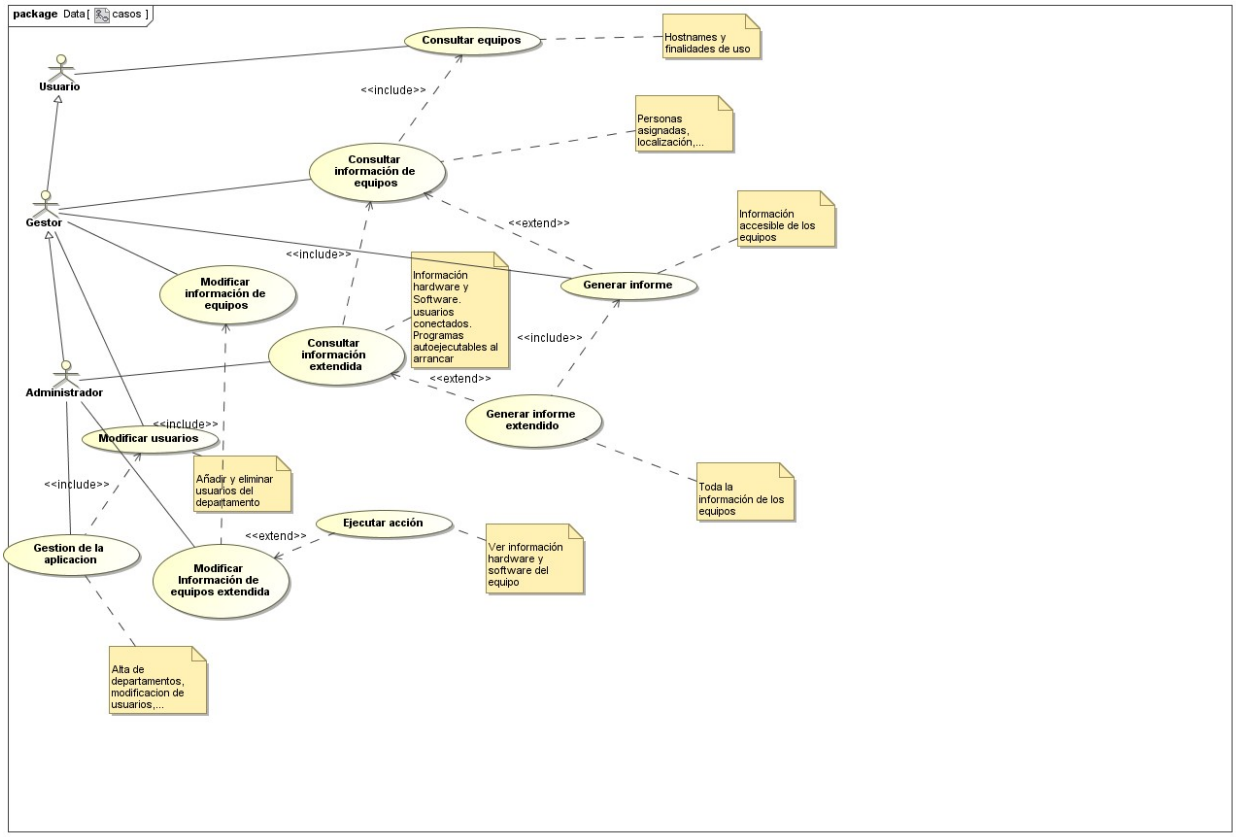
- Información espefícica de las máquinas.
- Dar de alta o baja un equipo virtual para un determinado departamento.
- Dar de alta un equipo.
- Dar de alta un departamento.
- Cambiar el rol de los usuarios.

Además, las personas integrantes del departamento podrán acceder a la aplicación para poder obtener información de nombres de equipo y uso del mismo.

Todo usuario logado, podrá modificar la información de sus datos.

El modelo de negocio

Aquí expondremos los diferentes casos de uso que se pueden deducir de la información previa. Podemos ver que existen los siguientes actores: Usuario, Gestor y Administrador. Podemos ver los casos de uso en el siguiente diagrama:



Como se muestra, todo usuario tiene un caso de uso para consultar la información y el gestor y el administrador tienen ambos un caso de uso para generar informes y gestionar. Dado que un administrador puede acceder a la misma información que un gestor y un gestor a la misma que un usuario (pero ampliada), se ha optado por poner estos casos como una inclusión con respecto al caso del administrador.

Desde el Modelo-Vista-Controlador, el controlador de cada página será el encargado de visualizar más o menos información dependiendo de los permisos que tenga el usuario logado en cuestión.

Tampoco se ha puesto aquí el caso de uso referente a la operación de logon y modificación de los datos del propio usuario, debido a su sencillez y generalidad.

Por todo lo indicado, pasamos a describir los casos de uso referentes al administrador.

1. Caso de uso: Alta de usuario

Resumen de la funcionalidad:

Permitir que se pueda dar de alta cualquier usuario en la aplicación. Salvo el usuario administrador, todos los demás usuarios deberán acceder a la aplicación a través de esta paso previo.

Actores:

Gestor y Usuario

Precondición:

No existe el usuario en la BBDD.

Postcondición:

El usuario está dado de alta en el sistema.

El nuevo usuario introduce información referente al mismo y se actualiza la BBDD con la misma para permitir el futuro acceso.

2. Caso de uso: Logado en la aplicación

Resumen de la funcionalidad:

Permitir que un usuario ya dado de alta en el sistema, acceda mediante su alias y su contraseña.

Actores:

Administrador, Gestor y Usuario

Precondición:

El usuario está dado de alta en la BBDD y no está logado en el sistema.

Postcondición:

El usuario está logado en el sistema.

El usuario introducirá su usuario y contraseña y pedirá la confirmación de los datos. Si los datos son correctos, se navegará hasta la página principal, si no es así, se indicará que existe un error.

3. Caso de uso: Consultar información extendida

Resumen de la funcionalidad:

Poder visualizar en una tabla la información de los equipos dados de alta en el sistema a través de una tabla

Actores:

Administrador

Precondición:

Estamos logados en el sistema.

Postcondición:

Obtenemos una tabla con la información de los equipos.

Al navegar a la página correspondiente, desde el menu principal, visualizaremos la tabla. Este caso de uso es el equivalente a consultar información del Gestor o del usuario.

4. Caso de uso: Generar Informe extendido

Resumen de la funcionalidad:

Se debe poder exportar la información de la tabla de los equipos en un fichero con extensión CSV.

Actores:

Administrador

Precondición:

Estamos logados en el sistema.

Postcondición:

Se descarga un fichero con la información de la tabla

Al navegar a la página correspondiente, desde el menú principal, se visualizará la tabla y habrá una opción para descargar la misma. Este caso de uso es el equivalente a generar informe del Gestor.

5. Caso de uso: Modificar información de equipos extendida

Resumen de la funcionalidad:

Conjunto de acciones sobre la información del equipo: recupera información del hardware y software del equipo seleccionado (caso de uso ejecutar acción), renueva Ips, modifica finalidad del equipo.

Actores:

Administrador

Precondición:

Estamos logados en el sistema.

Postcondición:

Obtenemos una pantalla con información referente al equipo. Esta información se recupera en tiempo real y, si es el caso, se actualiza en la BBDD.

Al navegar a la página correspondiente, desde el menú principal, se visualizará la tabla y, al seleccionar una columna, obtendremos un menú donde se podrán ver las diferentes opciones. Este caso de uso es el equivalente a modificar usuarios del Gestor.

6. Caso de uso: Gestionar aplicación

Resumen de la funcionalidad:

Conjunto de acciones sobre la información de la aplicación: modificación de datos de usuarios, roles, alta de equipos,...

Actores:

Administrador

Precondición:

Estamos logados en el sistema.

Postcondición:

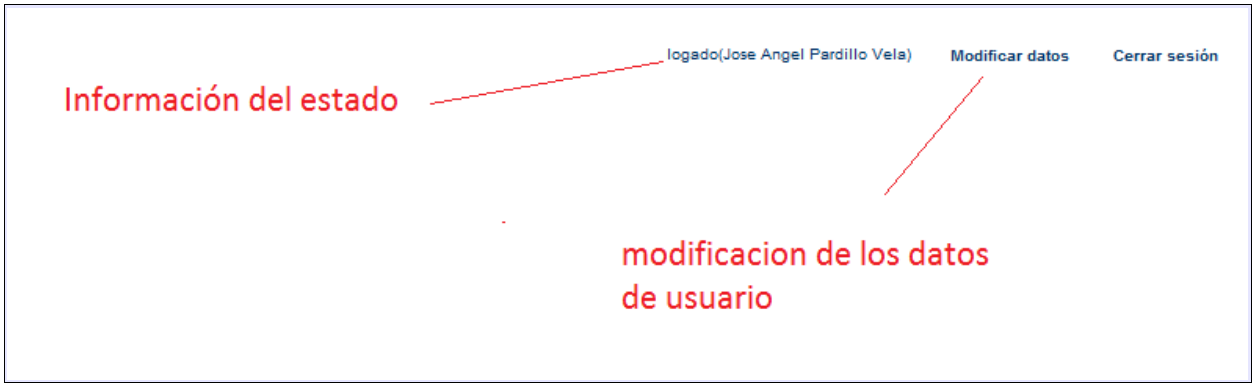
La información solicitada está modificada o dada de alta.

Al navegar a la página correspondiente, desde el menú principal, se visualizará una tabla y un menú con las opciones referentes a cada acción.

Requisitos de la interfaz de usuario

Tanto el logón como las páginas de la aplicación parten de la misma base:

- Creación de un menú en la esquina superior derecha. Este menú servirá para gestionar los datos de usuario:



- Opciones principales:

Una vez logados, aparte de este menu, tendremos otro menu que gestionará las funcionalidades de la aplicación. Haciendo click en cada una de sus opciones, se deberá mostrar las opciones correspondientes del mismo, que forma un segundo menu. Esta estará situado y justificado a la izquierda:

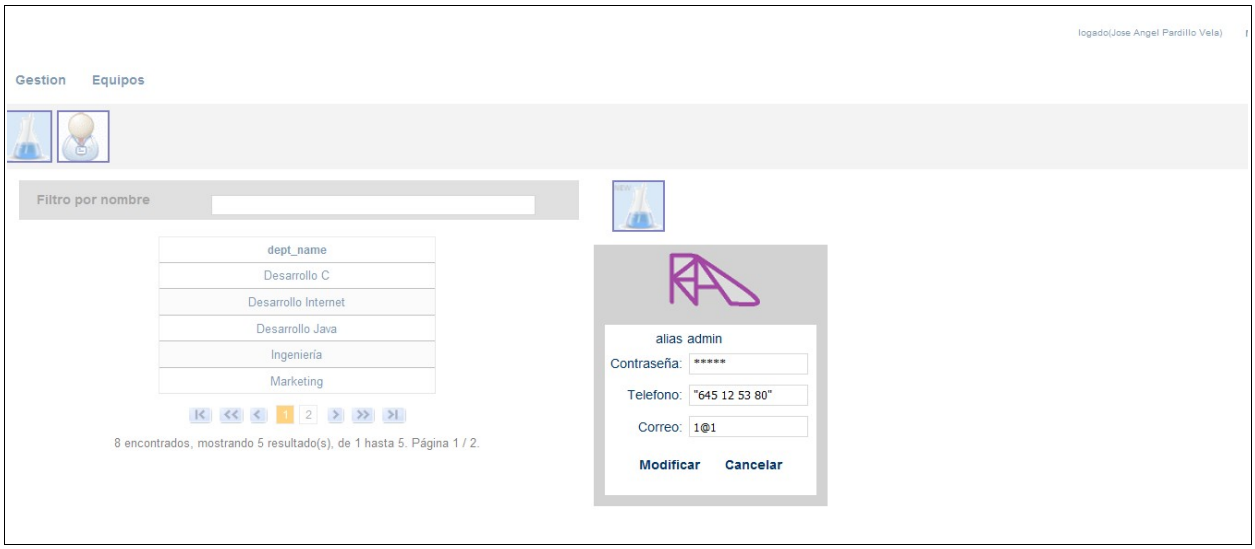
Con esto accederemos a las páginas.

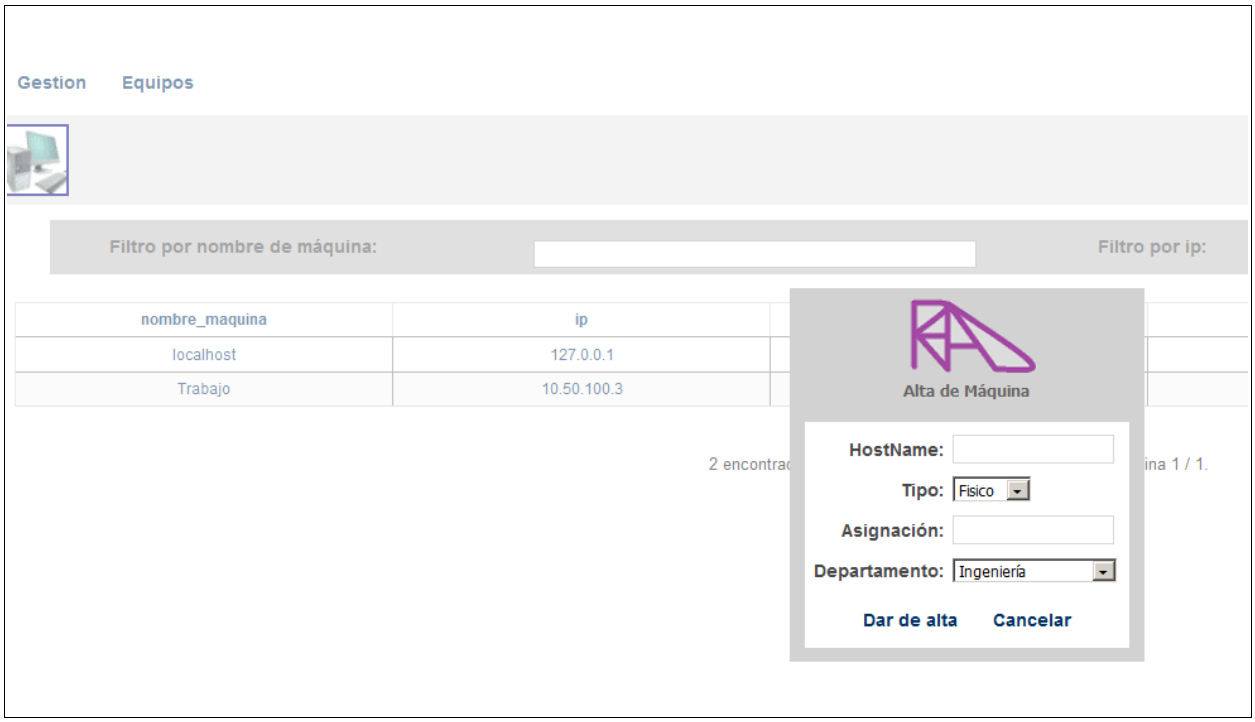
- Aspecto general de una página de la aplicación

El principal elemento de la página será una tabla en la parte central, dotada de paginación y un menu a su derecha con opciones:

- Opciones emergentes

Cuando sea necesario mostrar un menú emergente, se mostrará en forma modal con las opciones correspondientes y el logotipo de la aplicación. Todos los menús tienen el mismo aspecto. Unos ejemplos son:





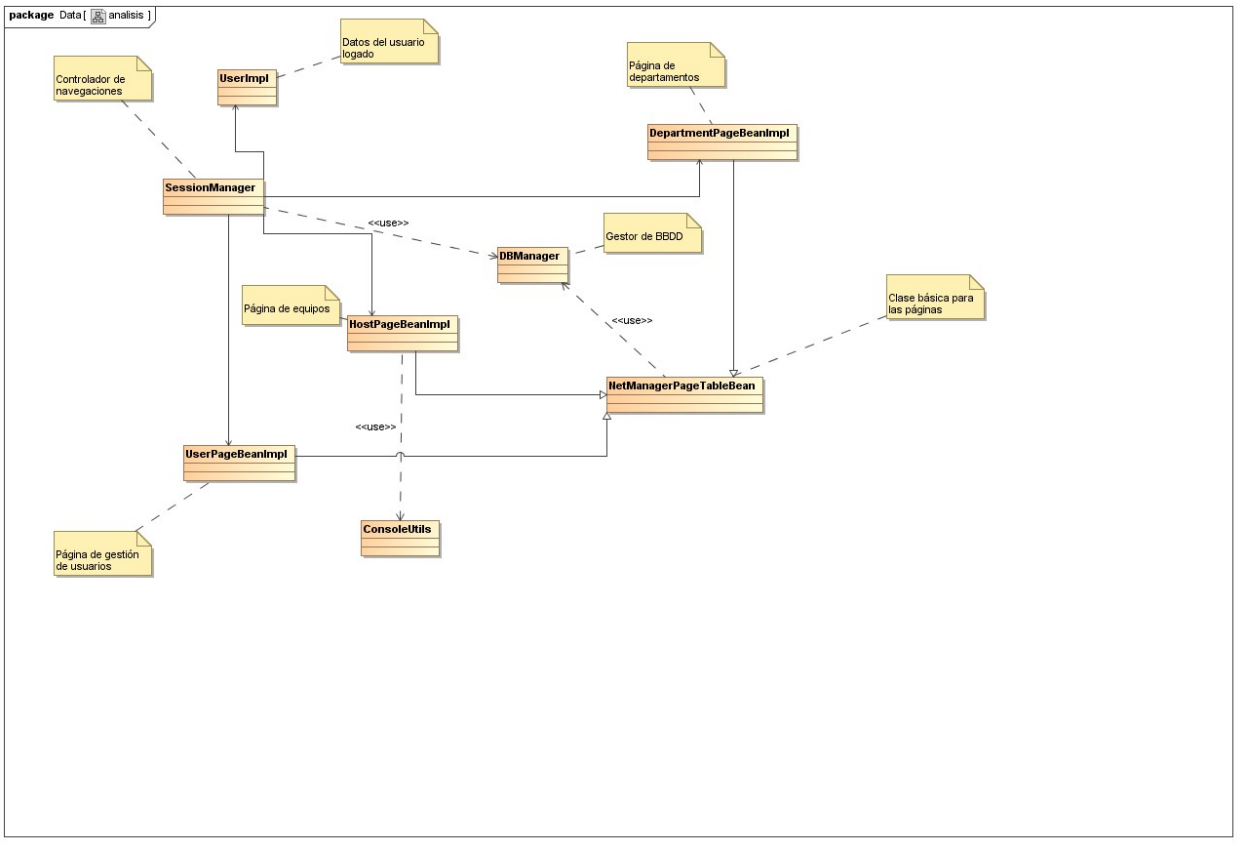
Análisis.

Identificación de las clases y atributos principales

Dado el MVC que se utiliza, se decidió crear la siguiente arquitectura:

- Una clase que gestionará las navegaciones.
- Un controlador por página.
- Un gestor para el acceso a la BBDD.
- Un gestor para el modelo de datos en la tabla. Este gestor incluye el control del filtro de las páginas y el número de columnas a visualizar.
- Una clase para gestionar los datos del usuario logado.

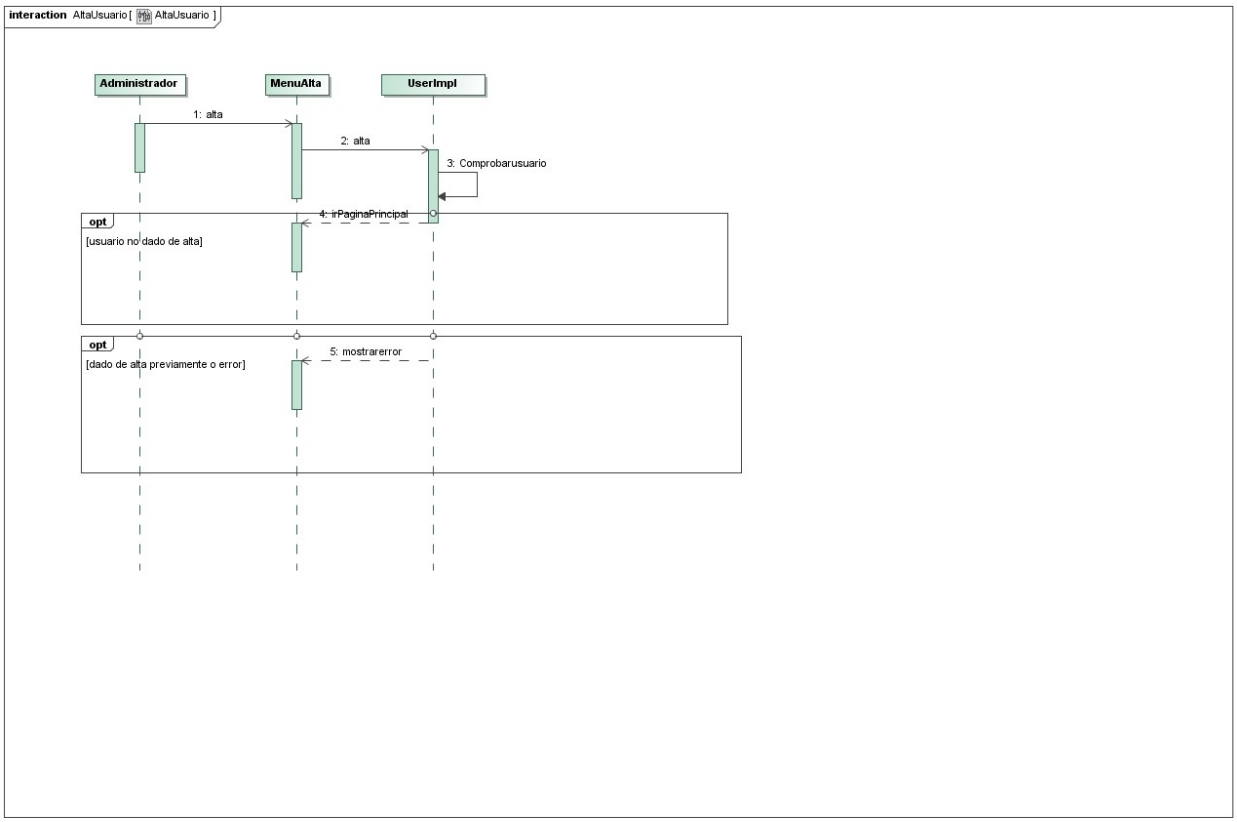
Con ello, aunque existe un único diagrama de clases que se mostrará en el apartado de diseño técnico, el diagrama de análisis de las clases quedaría de la siguiente forma (se han eliminado los atributos en esta primera aproximación al diagrama final ya que en este punto interesa sobre todo la relación entre las clases):



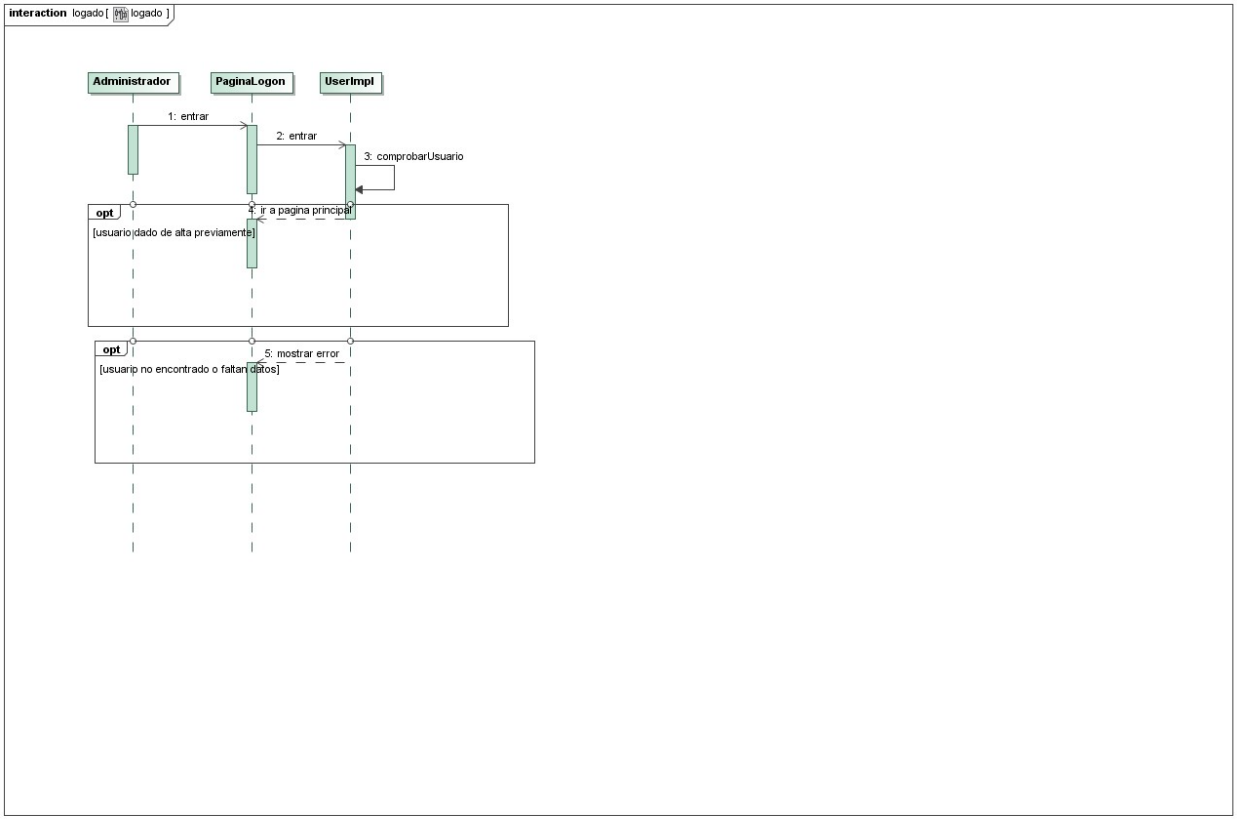
Diagramas identificativos de los casos de uso

A continuación se muestran los diagramas de secuencia de los casos de uso tratados anteriormente.

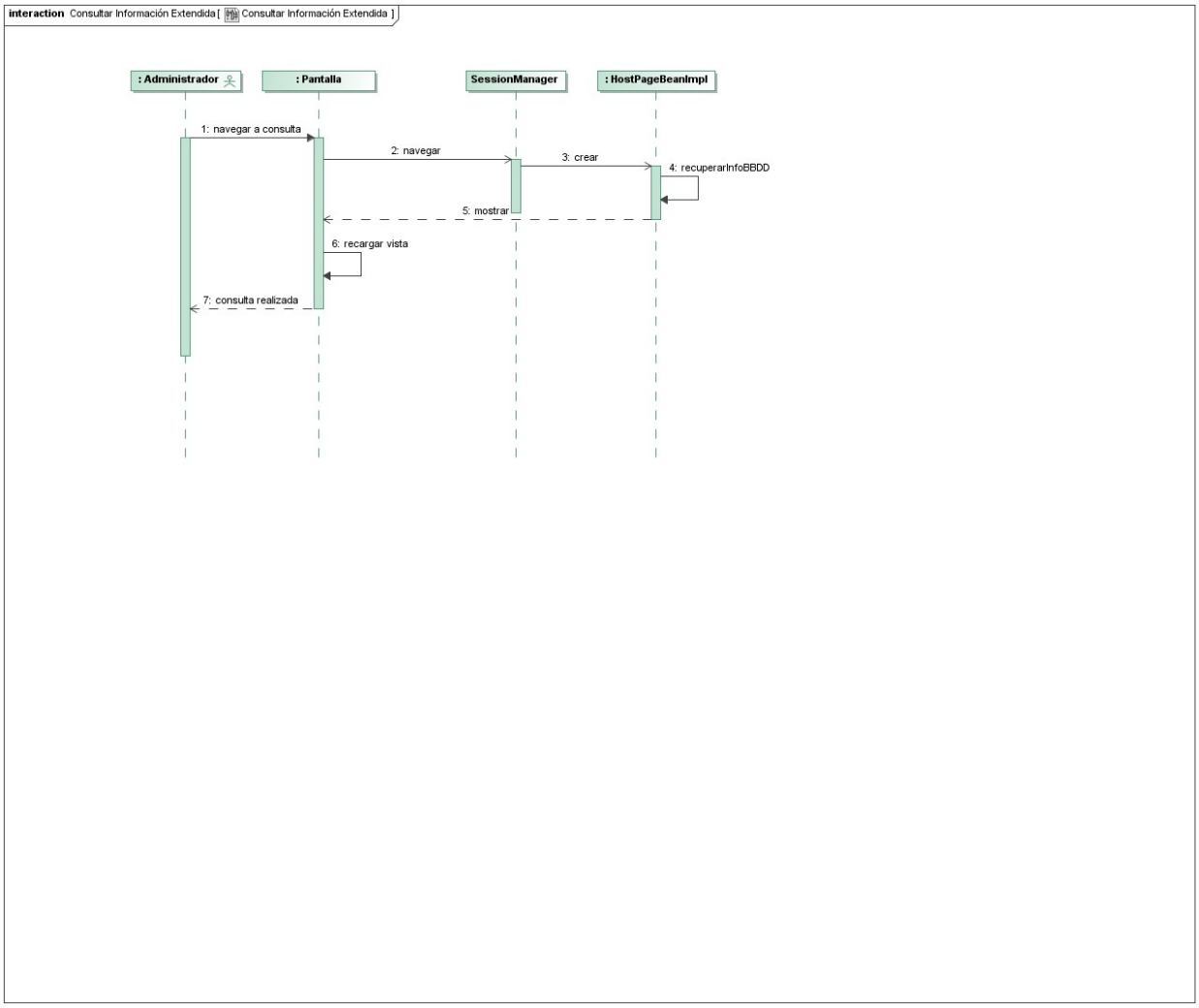
1. Caso de uso: Alta de usuario



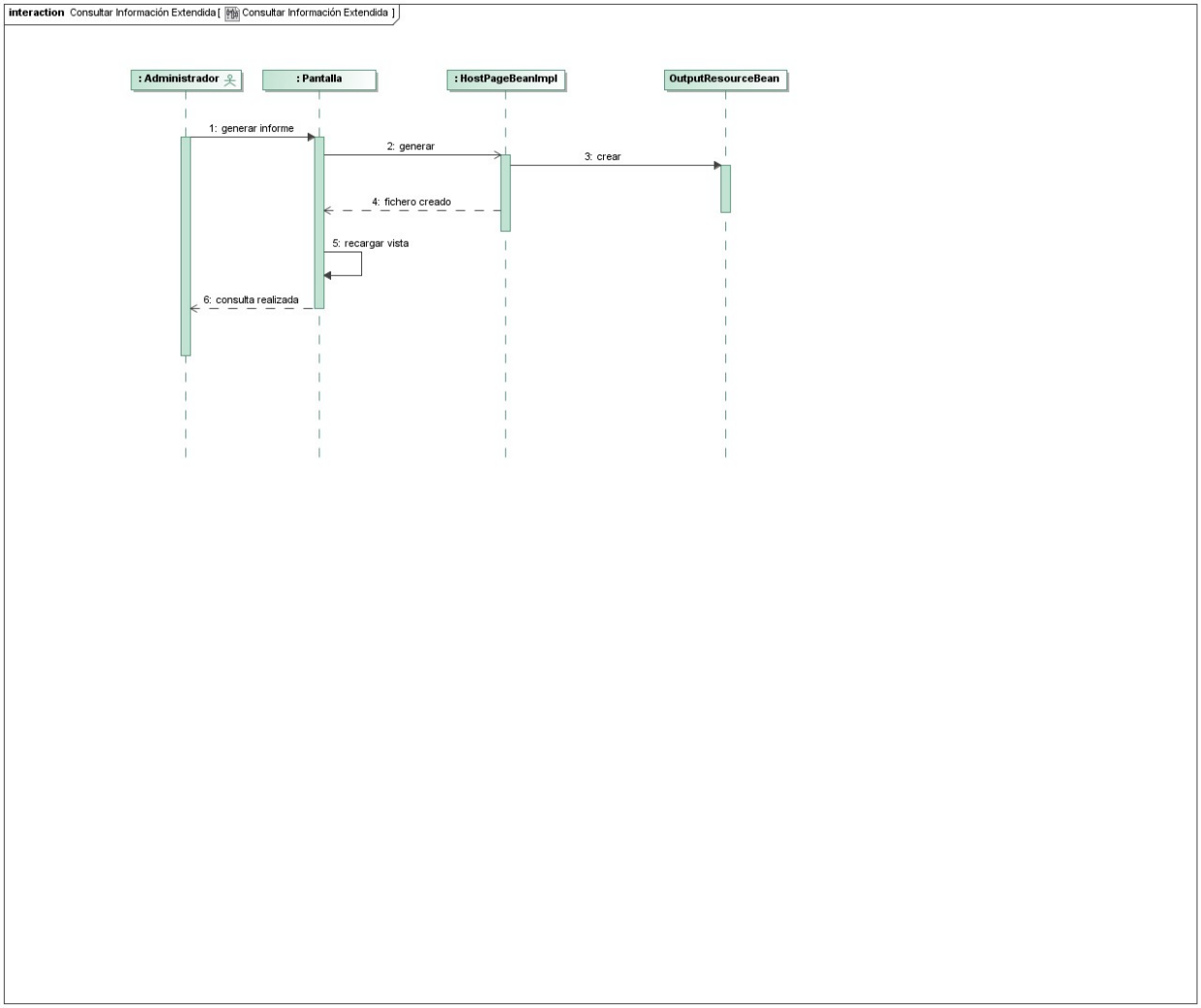
2. Caso de uso: Logado en la aplicación


interaction logado [logado]
Administrador
PaginaLogon
UserImpl
1: entrar
2: entrar
3: comprobarUsuario
opt
[usuario dado de alta previamente]
4: ir a pagina principal
opt
[usuario no encontrado o faltan datos]
5: mostrar error

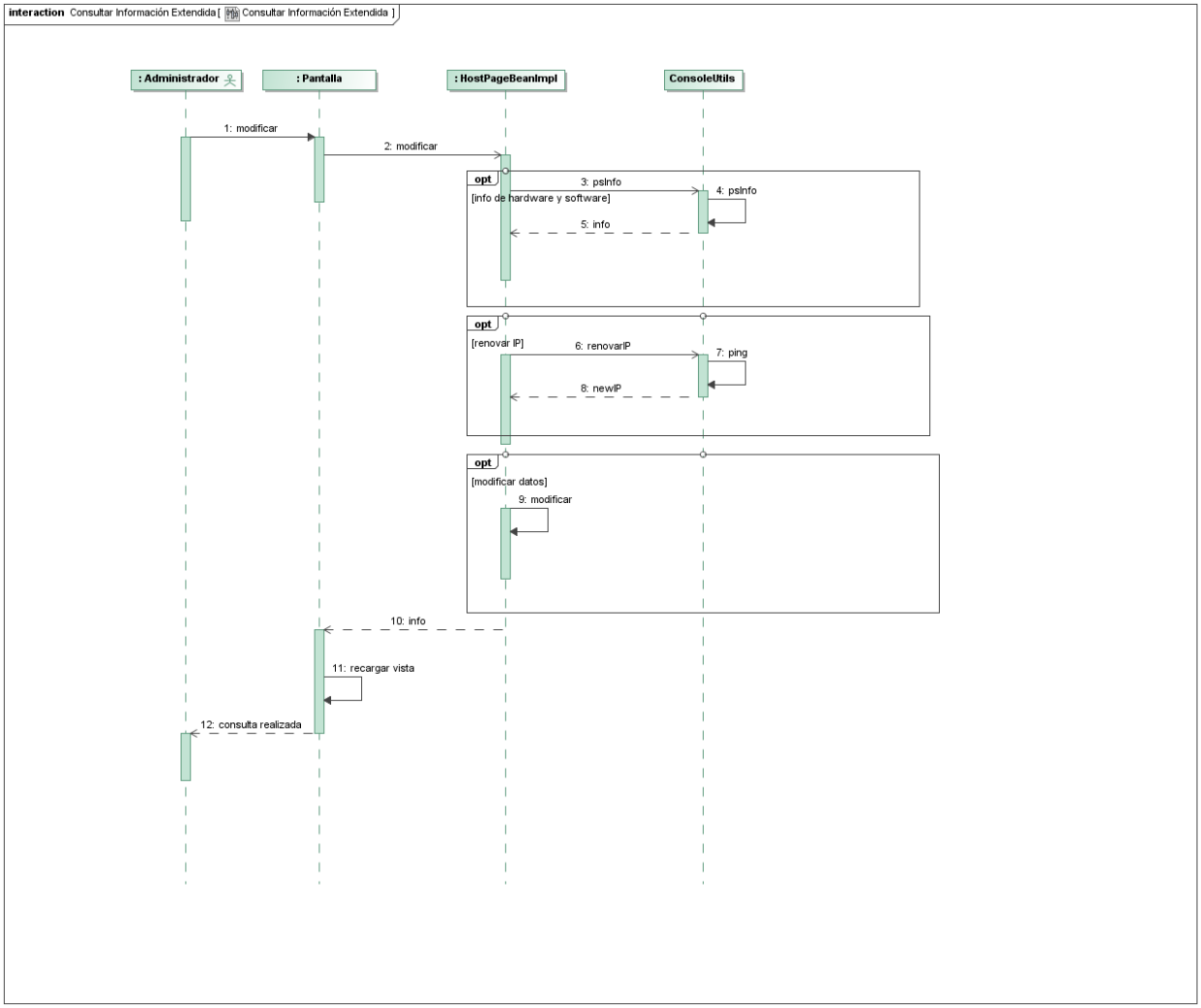
3. Caso de uso: Consultar información extendida



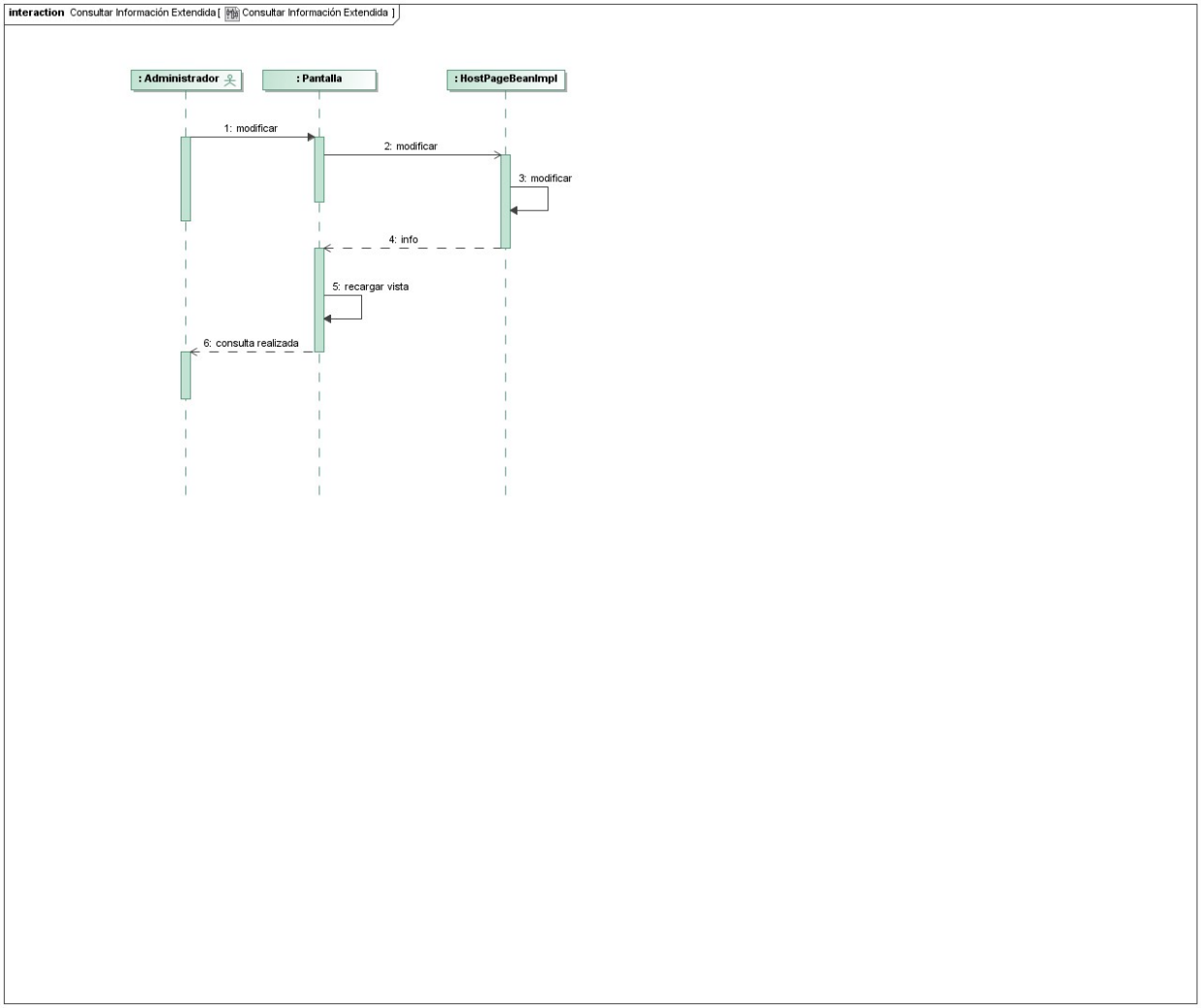
4. Caso de uso: Generar Informe extendido



5. Caso de uso: Modificar información de equipos extendida



6. Caso de uso: Gestionar aplicación

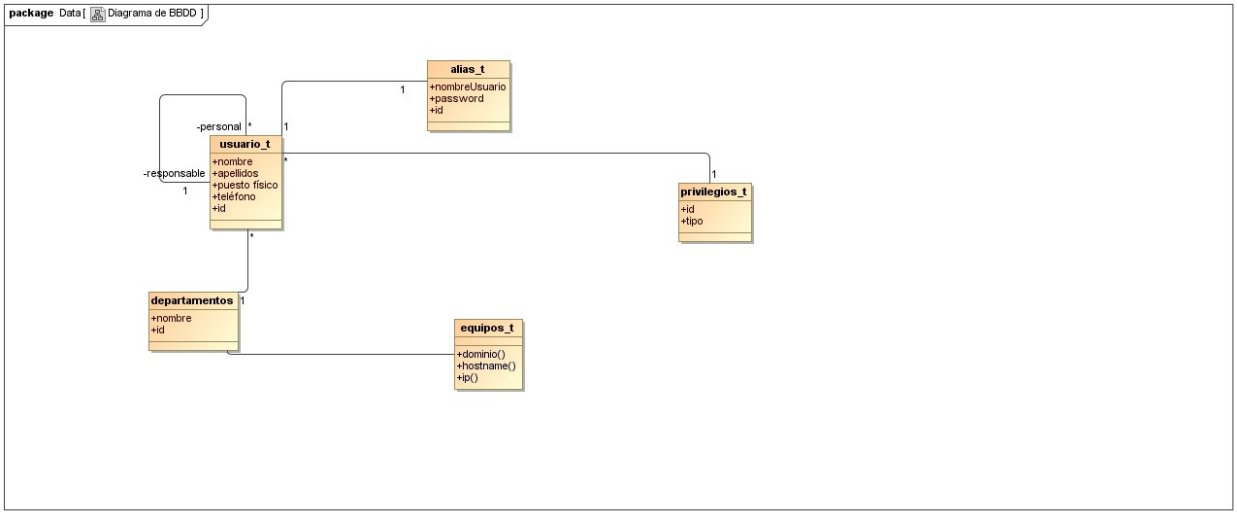


El modelo de datos

En la parte de análisis podemos destacar lo siguiente:

- Se utilizarán dos esquemas. Uno referente a los departamentos, donde se guardará información de los mismos y de los usuarios, y otro referente a los equipos.
- Para la visualización de las páginas, se crearán las correspondientes vistas.

Con todo ello tenemos el siguiente modelo para la etapa de análisis de los datos:



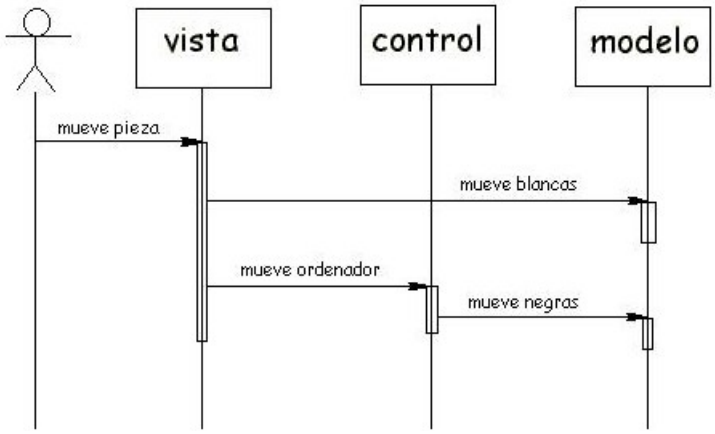
Diseño técnico.

Pasamos a desarrollar la parte de diseño de la aplicación. Aquí se verán varios de los anteriores diagramas con más profundidad y detalle que en la fase de análisis pura.

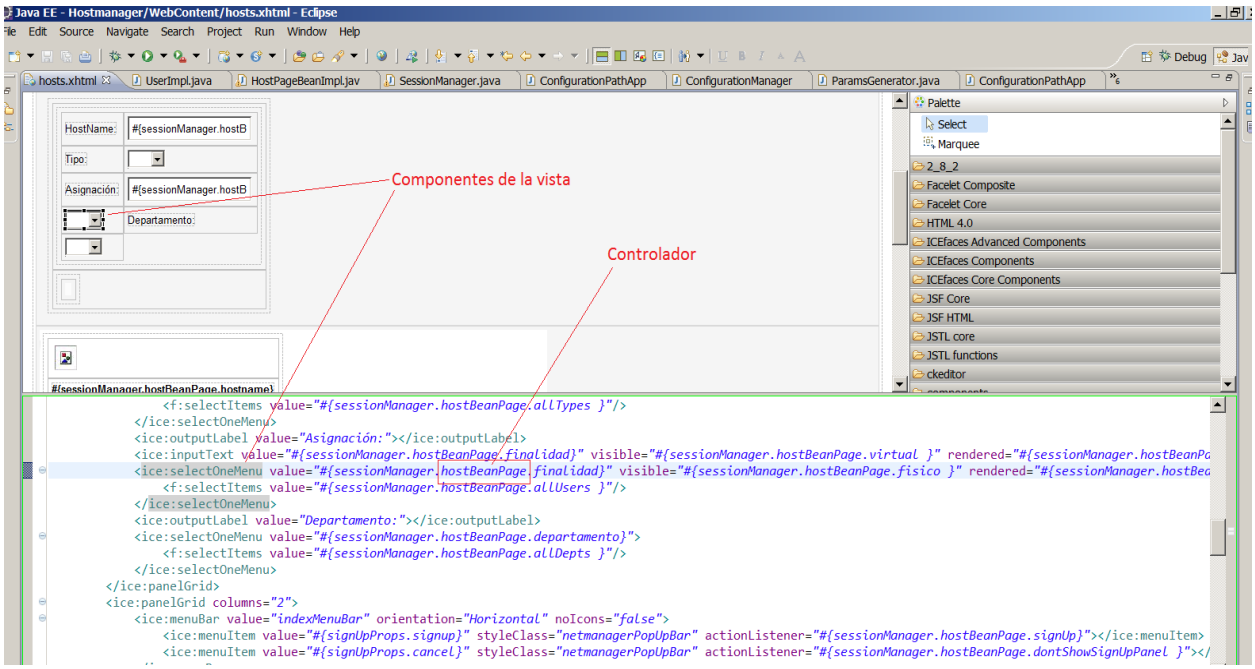
Arquitectura de la aplicación

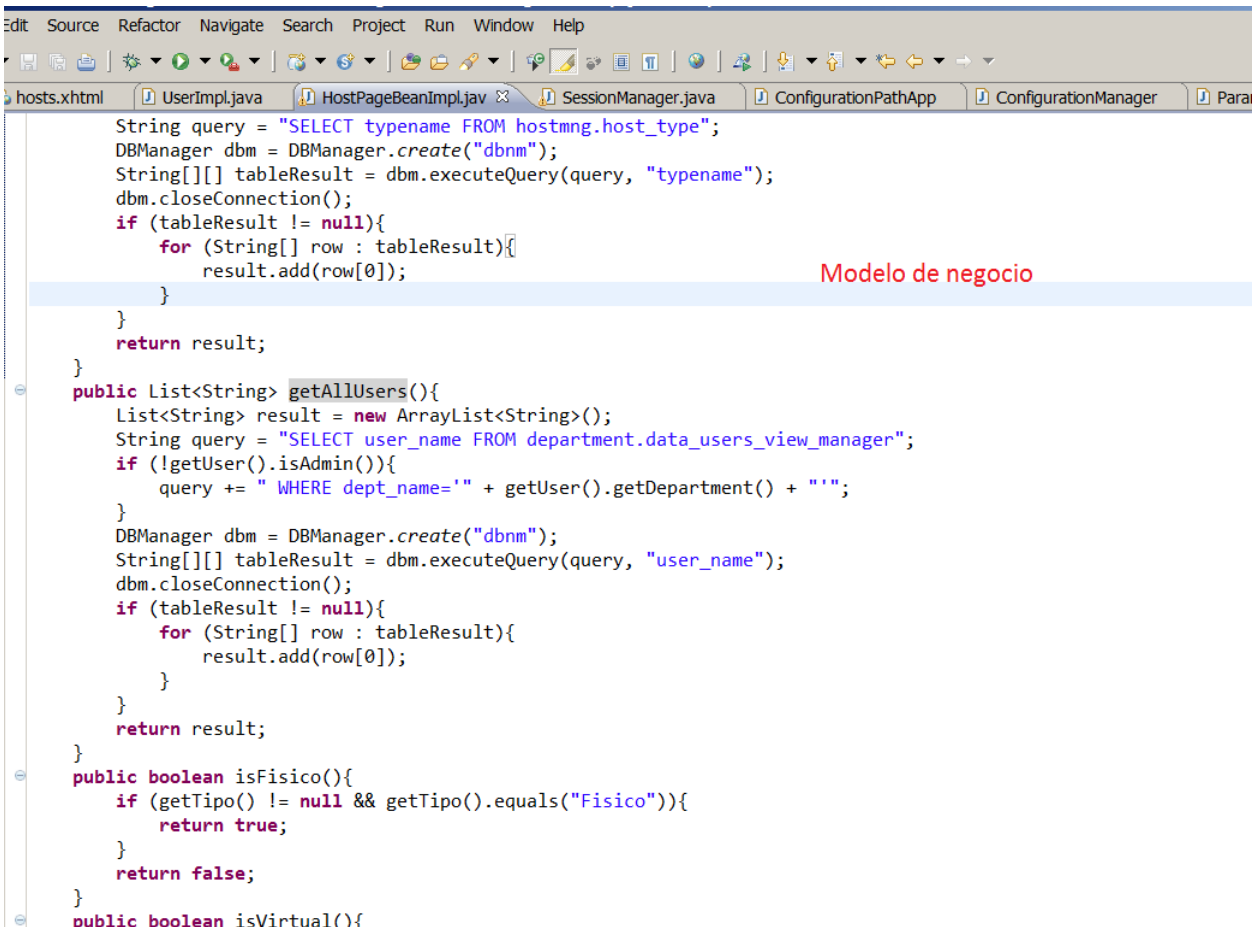
1. Introducción: bases de la arquitectura de desarrollo

La aplicación se desarrollará bajo la arquitectura J2EE, utilizando ICEFaces 2 (JSF) para la capa de presentación y JDBC para la de persisencia. Todo ello basado en el patrón Modelo-Vista-Controlador.



JSF nos permite diferenciar completamente entre cada uno de estos componentes. Podemos ver esto en cualquier página o componente de la interfaz gráfica.






```
        String query = "SELECT typename FROM hostmng.host_type";
        DBManager dbm = DBManager.create("dbnm");
        String[][] tableResult = dbm.executeQuery(query, "typename");
        dbm.closeConnection();
        if (tableResult != null){
            for (String[] row : tableResult){
                result.add(row[0]);
            }
        }
        return result;
    }
    public List<String> getAllUsers(){
        List<String> result = new ArrayList<String>();
        String query = "SELECT user_name FROM department.data_users_view_manager";
        if (!getUser().isAdmin()){
            query += " WHERE dept_name='" + getUser().getDepartment() + "'";
        }
        DBManager dbm = DBManager.create("dbnm");
        String[][] tableResult = dbm.executeQuery(query, "user_name");
        dbm.closeConnection();
        if (tableResult != null){
            for (String[] row : tableResult){
                result.add(row[0]);
            }
        }
        return result;
    }
    public boolean isFisico(){
        if (getTipo() != null && getTipo().equals("Fisico")){
            return true;
        }
        return false;
    }
    public boolean isVirtual(){
```

Como Sistema gestor de base de datos (SGBD) se utilizó PostgresSQL en versión 9. Con lo que fue necesario descargarse los drivers correspondientes para su uso con JDBC.

2. JSF 2

Dado la pertenencia del mismo al estándar J2EE fue la opción escogida para desarrollar la capa gráfica. Permite la realización de página mediante un patrón de composición y el uso de plantillas xhtml. Todo ello facilita la reutilización de código y la limpieza de las páginas.

Podemos ver, por ejemplo, la siguiente plantilla:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
xmlns:icecore="http://www.icefaces.org/icefaces/core"
xmlns:ace="http://www.icesoft.org/icefaces/components"
xmlns:ice="http://www.icesoft.com/icefaces/component"
xmlns:h="http://java.sun.com/jsf/html"
xmlns:ui="http://java.sun.com/jsf/facelets"
xmlns:f="http://java.sun.com/jsf/core">
<h:head>
<title><ui:insert name="title">Default title</ui:insert></title>
<!--rime is for ice components-->
<link rel="stylesheet" type="text/css" href="../resources/css/netmanager.css"/>
<f:loadBundle basename="com.tfc.net.manager.properties.logonPage" var="logonProps"></f:loadBundle>
<f:loadBundle basename="com.tfc.net.manager.properties.signUpPage" var="signUpProps"></f:loadBundle>
</h:head>
<!--styleclass ice-skin-rime is for icefaces ace components-->
<h:body>
    <div id="header">
        <ui:insert name="header">
        </ui:insert>
    </div>
    <div id="content">
        <ui:insert name="content">
        </ui:insert>
    </div>
    <div id="footer">
        <ui:insert name="footer">
        </ui:insert>
    </div>
</h:body>
</html>
```

en ella no solo definimos la estructura de la página, sino que podemos dar de alta properties con las etiquetas que se utilizarán en la aplicación. Es utilizada en todas las páginas a través del patron de composición. Este patrón nos sirve para poder declarar componentes propios con las características que necesitemos (así se creó ICEFaces):

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<ui:component xmlns="http://www.w3.org/1999/xhtml"
              xmlns:f="http://java.sun.com/jsf/core"
              xmlns:h="http://java.sun.com/jsf/html"
              xmlns:ui="http://java.sun.com/jsf/facelets"
              xmlns:c="http://java.sun.com/jsp/jstl/core"
              xmlns:composite="http://java.sun.com/jsf/composite"
              xmlns:ice="http://www.icesoft.com/icefaces/component"
              xmlns:nm="http://java.sun.com/jsf/composite/components" >

<composite:interface>
</composite:interface>

<composite:implementation>

<ice:panelGrid columns="1" visible="true" styleClass="netmanagerLogonPnl">
    <nm:main_Header_configMenu_mainpages></nm:main_Header_configMenu_mainpages>
</ice:panelGrid>
</composite:implementation>

</ui:component>
```

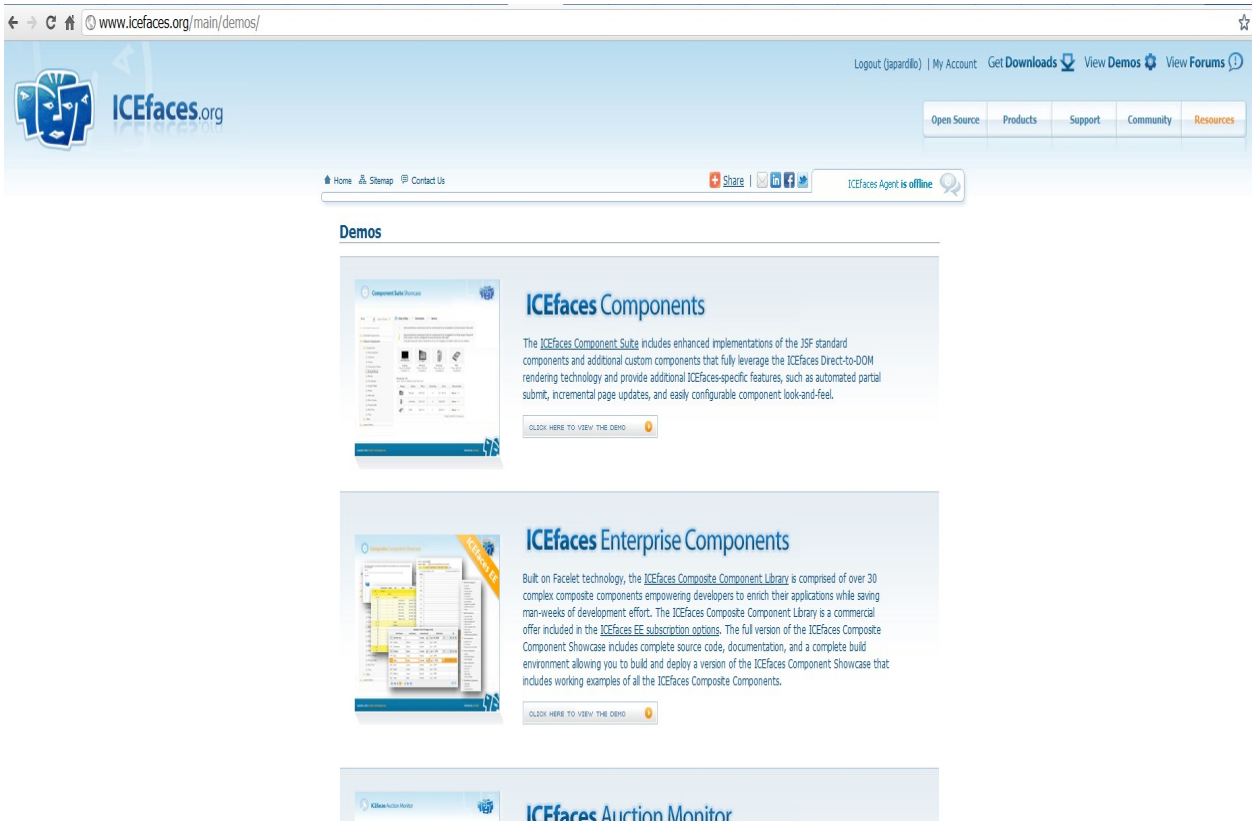
Con ello la página puede reutilizar código:

```
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:ui="http://java.sun.com/jsf/facelets"
    xmlns:ice="http://www.icesoft.com/icefaces/component"
    xmlns:h="http://java.sun.com/jsf/html"
    xmlns:f="http://java.sun.com/jsf/core"
    xmlns:ace="http://www.icefaces.org/icefaces/components"
    xmlns:nm="http://java.sun.com/jsf/composite/components">

<ui:composition template="WEB-INF/templates/mainTemplate.xhtml">

<ui:define name="content">
<ice:form id="hosts_form">
    <nm:main_Header_mainpages></nm:main_Header_mainpages>
    <nm:main_Header_PanelPop_mainpages></nm:main_Header_PanelPop_mainpages>
    <nm:main_header_optionsMenu></nm:main_header_optionsMenu>
    <ice:panelBorder>
        <f:facet name="east">
            <ice:panelGrid border="0" columns="1" visible="#{sessionManager.user.groupManager}" rendered="#
                <ice:commandLink actionListener="#{sessionManager.hostBeanPage.showSignUpPanel }" visible="
                    <ice:graphicImage url='resources/images/icons/ordenador.gif' alt="Alta de equipo"></ice
                </ice:commandLink>
```

3. Framework ICEFaces 2

El estándar JSF ha permitido que se desarrollen a partir del mismo potentes frameworks para el desarrollo de la interfaz gráfica de las aplicaciones web, entre ellos podemos encontrar ICEFaces (utilizado en su versión 2.0, http://www.icefaces.org).

Se eligió este framework debido al gran número de tutoriales y ejemplos que se encuentran en la propia página. Todos ellos sencillos y que dan una visión de todas las posibilidades que aporta al desarrollo.



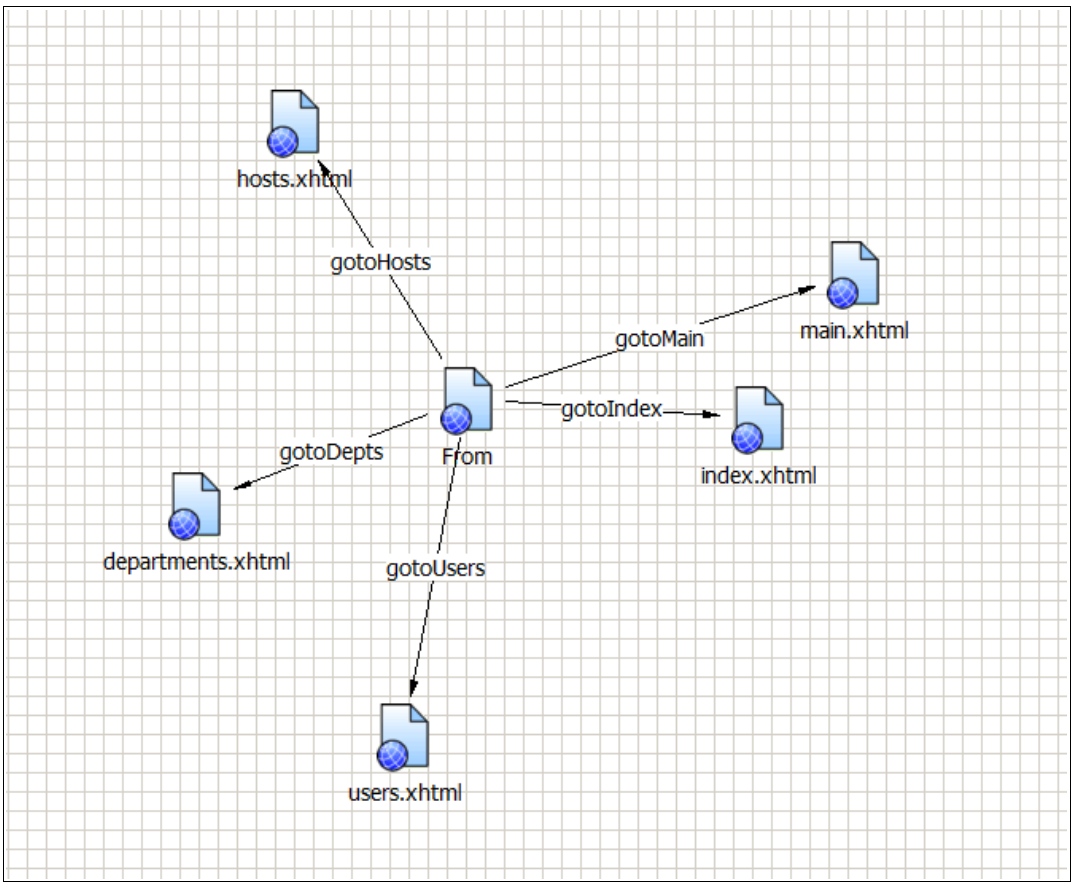
Tratándose de una comunidad de desarrollo con infinidad de cursos y documentos.

4. El Bean de sesión SessionManager

JSF permite definir Beans con diferentes tipos de “scope”, entre ellos se encuentra el tipo sesión, persistente mientras la sesión del usuario en curso es válida. Como aportación a la arquitectura de la aplicación se creó una clase de tipo sesión llamada SessionManager que controla las navegaciones de las páginas y el uso de un modelo de negocio u otro en las mismas.

Como características principales tiene lo siguiente:

- Métodos “goto”. Definen a qué página se navegará. La navegación en JSF se define a través de métodos que devuelven un String indicando la página siguiente. Con ello podemos enseñar aquí el diagrama de navegación de las páginas:



- El atributo “beans[]”. También controla qué modelo de negocio se utilizará en cada página sin necesidad de tener un atributo por cada página. Dado el hecho de que JSF reconoce como atributos todo método que empiece con el prefijo “get” y “set”, y uniendo a esto las características de la herencia en la programación orientada a objetos, se ha creado un patrón para la instanciación de la clase del modelo de negocio correcto en cada momento.

```
    }
    /**
     * Va a la página de equipos
     * @return
     */
    public String gotoHostsPage(){
        String result = "gotoHosts";
        setBeans(new Object[1]);
        String where = null;
        /*if (rol.toLowerCase().equals("manager") || rol.toLowerCase().equals("user")){
            where = "dept_name='" + getUser().getDepartment() + "' OR dept_name='Ninguno'";
        }*/
        where = " departamento='" + getUser().getDepartment() + "'";
        if (getUser().isAdmin()){
            getBeans()[0] = new HostPageBeanImpl("hosts",
                    "hostmng.hosts_view_main",
                    "nombre_maquina,ip, finalidad, tipo, departamento",
                    null, getUser());
        } else if (getUser().isGroupManager()){
            getBeans()[0] = new HostPageBeanImpl("hosts",
                    "hostmng.hosts_view_main",
                    "nombre_maquina,ip, tipo, finalidad",
                    where, getUser());
        } else {
            getBeans()[0] = new HostPageBeanImpl("hosts",
                    "hostmng.hosts_view_main_user",
                    "nombre_maquina,ip, finalidad",
                    where, getUser());
        }

        return result;
    }
```

se instancia la clase de la página en beans

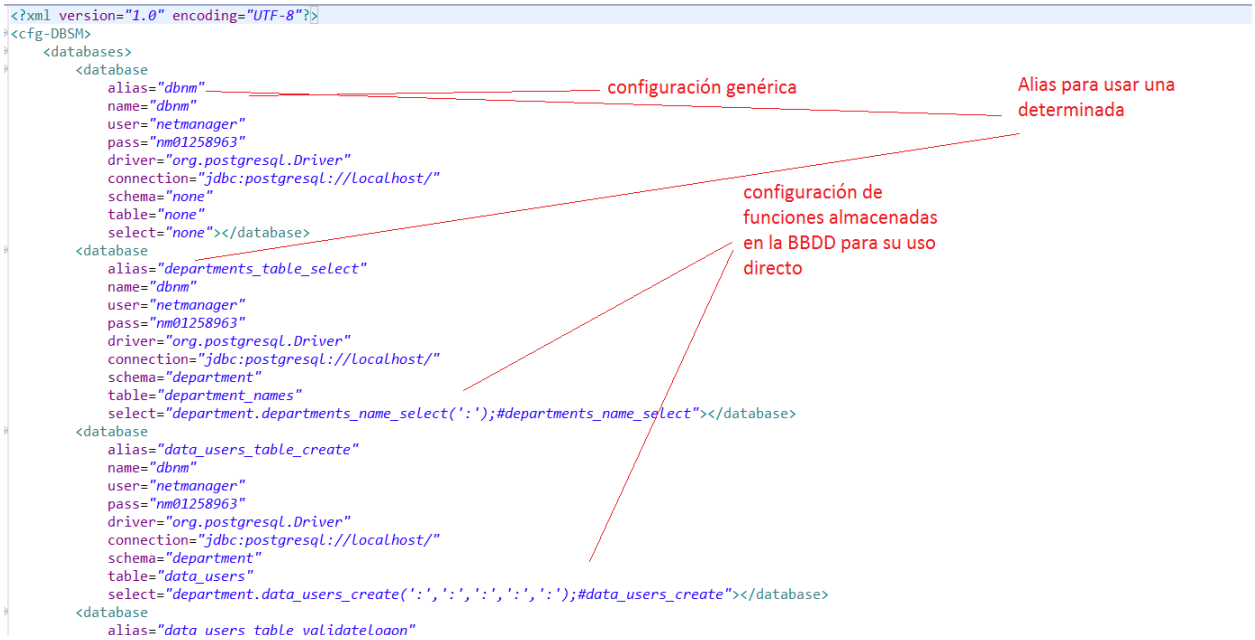
5. El gestor de BBDD DBManager

Para el uso de JDBC se optó por la creación de una clase que manejara las conexiones y devolviera los datos de forma homogénea en toda la aplicación. Esta clase fue el DBManager.

- Configuración. Para la misma se utiliza el fichero cfg-DBSM.xml y la clase ConfigurationPathAppManager, que nos indica donde se encuentra dicho fichero:

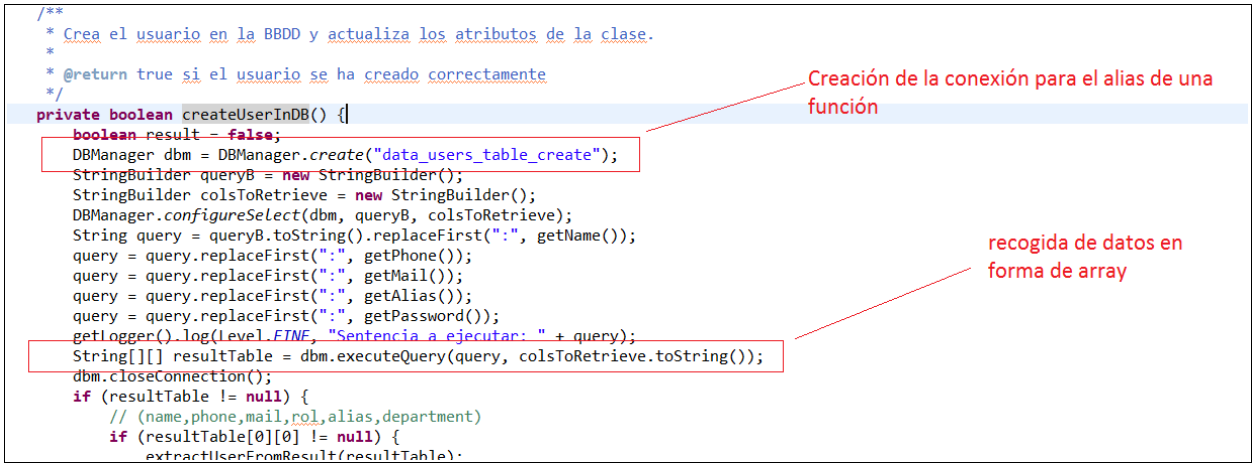
```
 * Esta clase será la única modificable si las rutas de los ficheros de configuración cambian
 */
public abstract class ConfigurationPathAppManager {

    public static final String ERROR_LOADING_CONFIGURATION_FILE = "Error loading configuration file";

    /**
     * Recupera el fichero de configuracion
     * @param configurationType
     * @return File
     * @throws ConfigurationManagerException
     */
    protected static File managerFile(ConfigurationType configurationType)
            throws ConfigurationManagerException {
                String fileStr = getConfigurationPath(configurationType);
                File result = new File(fileStr);
                if (!result.exists()){
                    throw new ConfigurationManagerException(
                            ConfigurationPathAppManagerImpl.class.getName() + ": " + fileStr);
                }
                return result;
            }

    /**
     * Retorna el servicio de configuración que se recuperará
     * @param configurationType
     * @return
     */
    private static String getConfigurationPath(ConfigurationType configurationType) {
        Map<ConfigurationType, String> conf = new HashMap<ConfigurationType, String>();
        conf.put(ConfigurationType.DB_MANAGER, "C:/eclipse_WS/WS_TFC/Hostmanager/WebContent" + File.separator + "resources" +
                File.separator + "xml" + File.separator +
                "configuration" + File.separator + "cfg-DBSM.xml");
        return conf.get(configurationType);
    }
```

En el fichero podemos configurar alias de conexión (que es lo que usa DBManager para crearla), así como consultas modelo y genéricas:




```
<?xml version="1.0" encoding="UTF-8"?>
<cfg-DBSM>
    <databases>
        <database
            alias="dbnm"
            name="dbnm"
            user="netmanager"
            pass="nm01258963"
            driver="org.postgresql.Driver"
            connection="jdbc:postgresql://localhost/"
            schema="none"
            table="none"
            select="none"></database>
        <database
            alias="departments_table_select"
            name="dbnm"
            user="netmanager"
            pass="nm01258963"
            driver="org.postgresql.Driver"
            connection="jdbc:postgresql://localhost/"
            schema="department"
            table="department_names"
            select="department.departments_name_select(':');#departments_name_select"></database>
        <database
            alias="data_users_table_create"
            name="dbnm"
            user="netmanager"
            pass="nm01258963"
            driver="org.postgresql.Driver"
            connection="jdbc:postgresql://localhost/"
            schema="department"
            table="data_users"
            select="department.data_users_create(':',':',':',':',':');#data_users_create"></database>
        <database
            alias="data_users_table_validateloaon"
```

con ello, la conexión y recogida de datos se encapsula y simplifica:




```
/**
 * Crea el usuario en la BBDD y actualiza los atributos de la clase.
 *
 * @return true si el usuario se ha creado correctamente
 */
private boolean createUserInDB() {
    boolean result = false;
    DBManager dbm = DBManager.create("data_users_table_create");
    StringBuilder queryB = new StringBuilder();
    StringBuilder colsToRetrieve = new StringBuilder();
    DBManager.configureSelect(dbm, queryB, colsToRetrieve);
    String query = queryB.toString().replaceFirst(":", getName());
    query = query.replaceFirst(":", getPhone());
    query = query.replaceFirst(":", getMail());
    query = query.replaceFirst(":", getAlias());
    query = query.replaceFirst(":", getPassword());
    getLogger().log(Level.FINE, "Sentencia a ejecutar: " + query);
    String[][] resultTable = dbm.executeQuery(query, colsToRetrieve.toString());
    dbm.closeConnection();
    if (resultTable != null) {
        // (name,phone,mail,rol,alias,department)
        if (resultTable[0][0] != null) {
            extractUserFromResult(resultTable);
```

6. El modelo de visualización y autocompletado

A partir de los tutoriales de ICEFaces sobre la creación de tablas con autocompletado. Se creó un modelo para este propósito que tiene como fin la generación dinámica de tablas independientemente del número de columnas que se especifique y una función de autocompletado para cada uno de las columnas.

Con ello, lo único que debemos indicar a la correspondiente clase que controlará la página es simplemente el nombre de la tabla y las columnas a recuperar. Todo ello se implemente el la clase NetManagerPageTableBean y en sus correspondientes clases hijas (de hecho, se puede generalizar e indicar varias tablas):

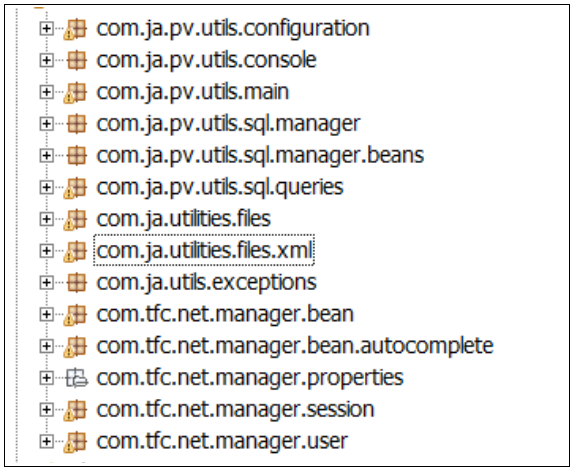
```
/**
 *
 * @param pageName
 * @param tableNames
 * @param tableCols
 * @param wf filtro where para la tabla
 * @param user
 */
public HostPageBeanImpl(String pageName, String tableNames, String tableCols, String wf, UserImpl user) {
    // TODO Auto-generated constructor stub
    super(pageName, tableNames, tableCols, wf);
    outputResource = new OutputResourceBean(user);
    this.user = user;
}
```

Su forma de utilización es la siguiente:

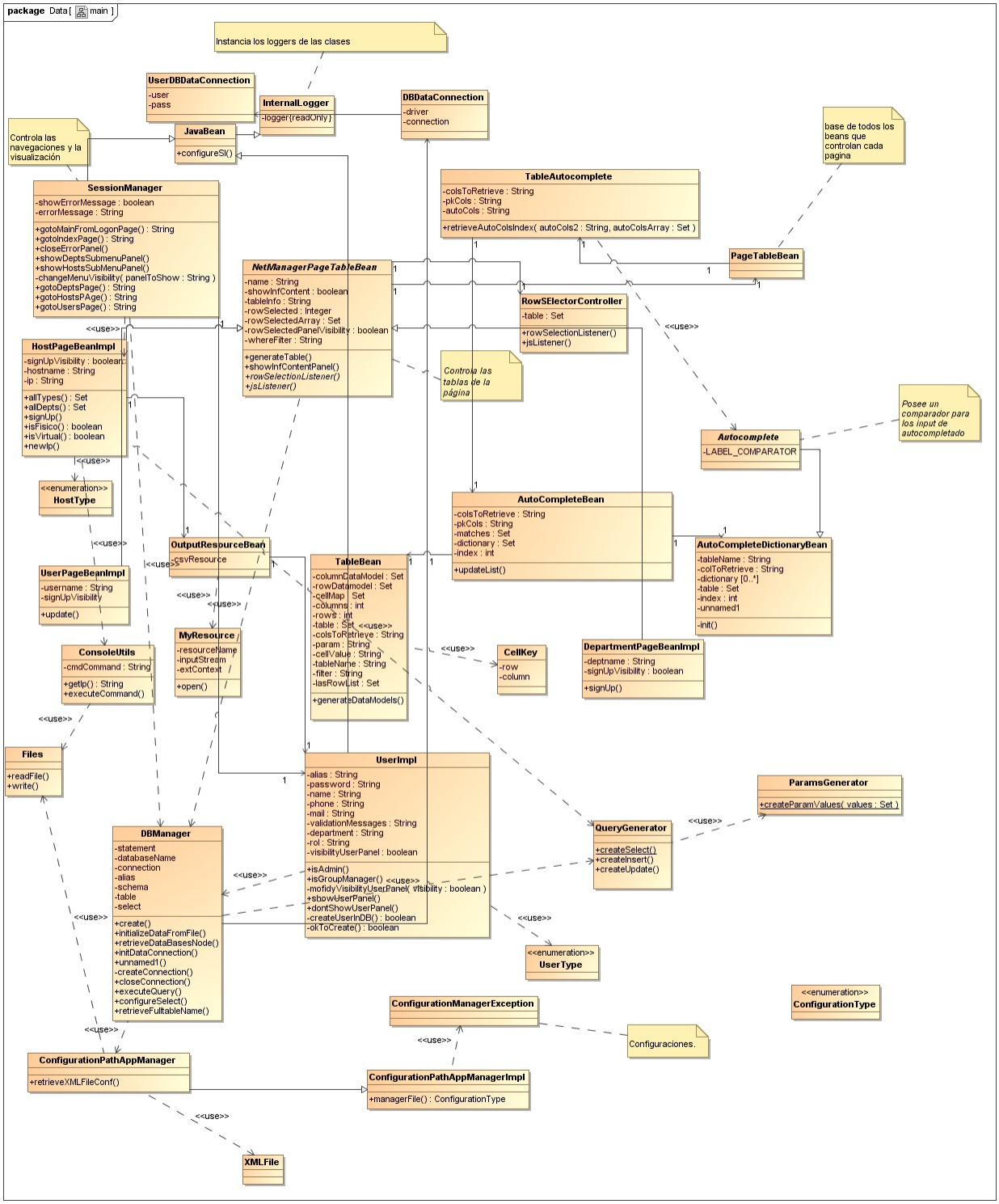
```
setBeans(new Object[1]);
String where = "dept_name='Ninguno' OR dept_name='" + getUser().getDepartment() + "'";
if (getUser().isAdmin()){
    getBeans()[0] = new UserPageBeanImpl("users", "department.data_users_view_manager",
            "user_name,aliasname,rol_name,dept_name", null, getUser());
} else if (getUser().isGroupManager()){
    getBeans()[0] = new UserPageBeanImpl("users", "department.data_users_view_manager",
            "user_name,dept_name", where, getUser());
}
```

Diagrama de clases del diseño

Ya se ha comentado las principales funcionalidades. Todas ellas se dividen en los siguientes paquetes:

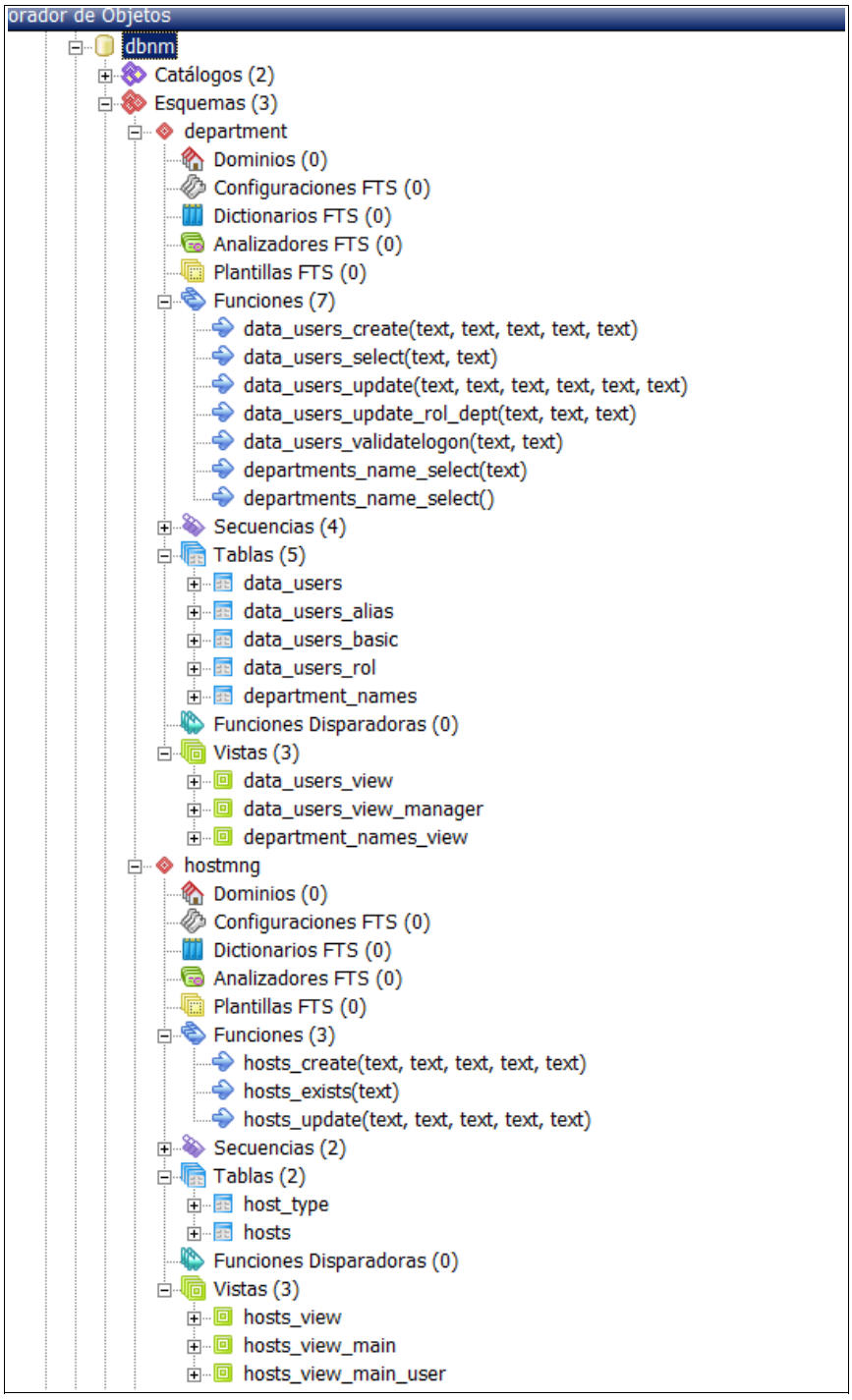


con ello, el diagrama de clases correspondiente es:


package Data [main]
Instancia los loggers de las clases
UserDBDataConnection
-user
-pass
InternalLogger
-logger{readOnly}
DBDataConnection
-driver
-connection
Controla las navegaciones y la visualización
JavaBean
+configureSI()
base de todos los beans que controlan cada pagina
SessionManager
-showErrorMessage : boolean
-errorMessage : String
+gotoMainFromLogonPage() : String
+gotoIndexPage() : String
+closeErrorPanel()
+showDeptsSubmenuPanel()
+showHostsSubMenuPanel()
-changeMenuVisibility(panelToShow : String)
+gotoDeptsPage() : String
+gotoHostsPAge() : String
+gotoUsersPage() : String
TableAutocomplete
-colsToRetrieve : String
-pkCols : String
-autoCols : String
+retrieveAutoColsIndex(autoCols2 : String, autoColsArray : Set)
PageTableBean
NetManagerPageTableBean
-name : String
-showInfContent : boolean
-tableInfo : String
-rowSelected : Integer
-rowSelectedArray : Set
-rowSelectedPanelVisibility : boolean
-whereFilter : String
+generateTable()
+showInfContentPanel()
+rowSelectionListener()
+jsListener()
RowSElectorController
-table : Set
+rowSelectionListener()
+jsListener()
HostPageBeanImpl
-signUpVisibility : boolean
-hostname : String
-ip : String
+allTypes() : Set
+allDepts() : Set
+signUp()
+isFisico() : boolean
+isVirtual() : boolean
+newIp()
Controla las tablas de la página
Autocomplete
-LABEL_COMPARATOR
Posee un comparador para los input de autocompletado
<<enumeration>>
HostType
AutoCompleteBean
-colsToRetrieve : String
-pkCols : String
-matches : Set
-dictionary : Set
-index : int
+updateList()
AutoCompleteDictionaryBean
-tableName : String
-colToRetrieve : String
-dictionary [0..*]
-table : Set
-index : int
-unnamed1
-init()
OutputResourceBean
-csvResource
TableBean
-columnDataModel : Set
-rowDatamodel : Set
-cellMap : Set
-columns : int
-rows : int
-table : Set <<use>>
-colsToRetrieve : String
-param : String
-cellValue : String
-tableName : String
-filter : String
-lasRowList : Set
+generateDataModels()
UserPageBeanImpl
-username : String
-signUpVisibility
+update()
MyResource
-resourceName
-inputStream
-extContext
+open()
CellKey
-row
-column
DepartmentPageBeanImpl
-deptname : String
-signUpVisibility : boolean
+signUp()
ConsoleUtils
-cmdCommand : String
+getIp() : String
+executeCommand()
Files
+readFile()
+write()
UserImpl
-alias : String
-password : String
-name : String
-phone : String
-mail : String
-validationMessages : String
-department : String
-rol : String
-visibilityUserPanel : boolean
+isAdmin()
+isGroupManager() <<use>>
-mofidyVisibilityUserPanel(visibility : boolean)
+showUserPanel()
+dontShowUserPanel()
-createUserInDB() : boolean
-okToCreate() : boolean
ParamsGenerator
+createParamValues(values : Set)
QueryGenerator
+createSelect()
+createInsert()
+createUpdate()
DBManager
-statement
-databaseName
-connection
-alias
-schema
-table
-select
+create()
+initializeDataFromFile()
+retrieveDataBasesNode()
+initDataConnection()
+unnamed1()
-createConnection()
+closeConnection()
+executeQuery()
+configureSelect()
+retrieveFulltableName()
<<enumeration>>
UserType
<<enumeration>>
ConfigurationType
ConfigurationManagerException
Configuraciones.
ConfigurationPathAppManager
+retrieveXMLFileConf()
ConfigurationPathAppManagerImpl
+managerFile() : ConfigurationType
XMLFile
<<use>>

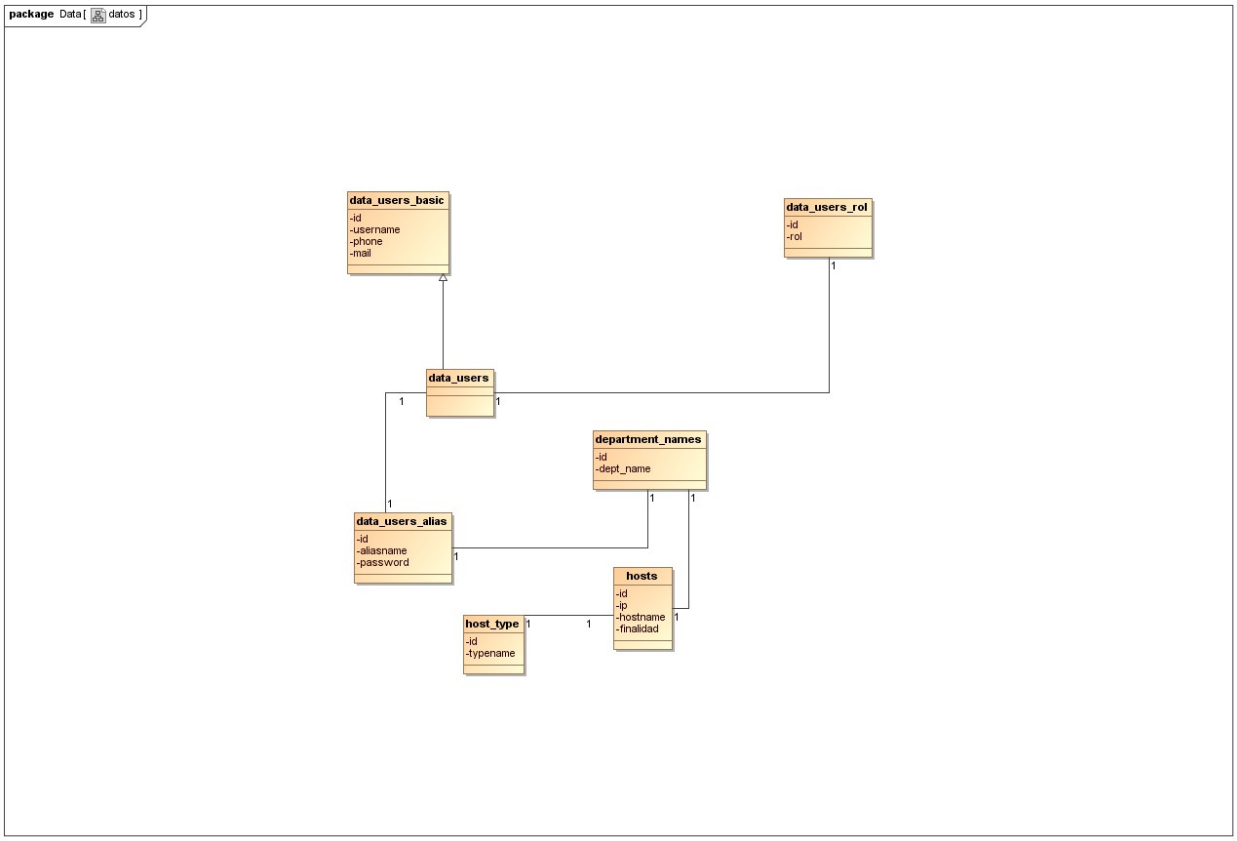
Arquitectura del modelo completo de datos

Se tratará de una única BBDD con dos esquemas, procedimientos almacenados y vistas tal y como se comentó en la etapa de análisis. A continuación se muestra la estructura de la base de datos:



con los procedientos almacenados ganamos velocidad de respuesta y homogeneidad a la hora de hacer las consultas. Mientras, de las vistas será de donde se recupere la información a mostrar en las tablas de cada una de las páginas.

Con todo ello, podemos ver el siguiente modelo ER completo para la capa de persistencia de datos:



Implementación.

Pasamos a describir el entorno de despliegue de la aplicación. Como producto final hemos obtenido el fichero WAR, los jar de dependencias no incluidos y los scripts de BBDD.

Apache Tomcat v7.0

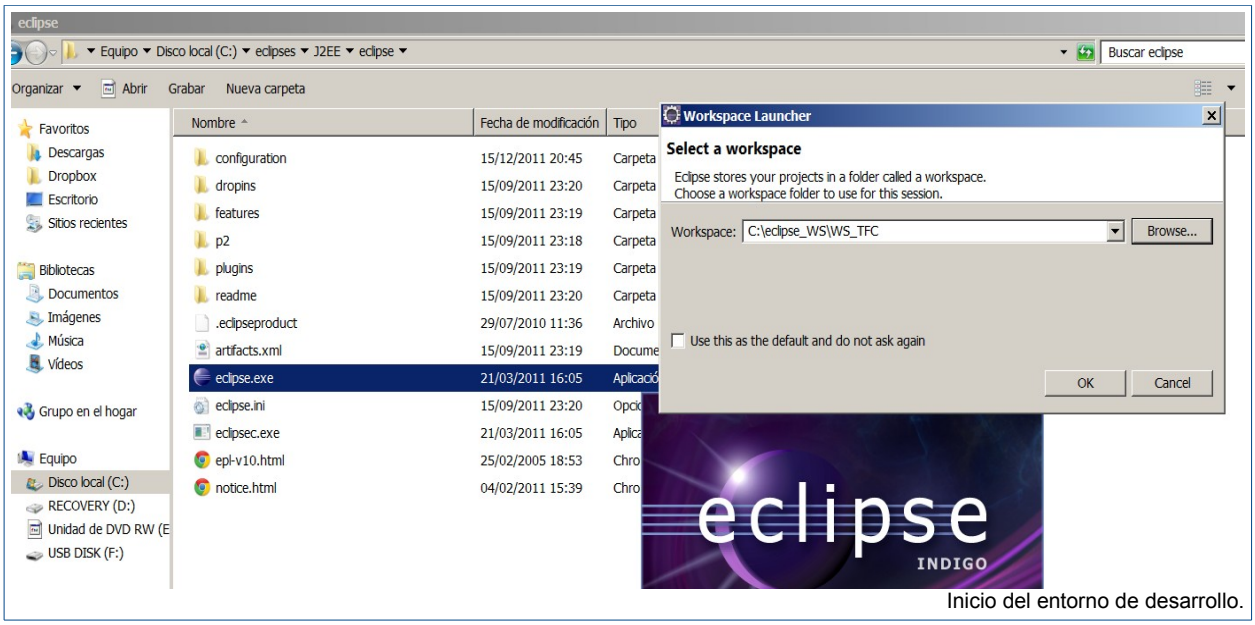
Será nuestro servidor para pruebas desde el entorno de desarrollo y en el que se colgará la aplicación. Para poderlo instalar, es necesario descargárselo de la web oficial. Una vez instalado el servidor apache podemos pasar a ver el entorno de desarrollo.

Eclipse

Será necesario descargar el entorno de desarrollo eclipse para desarrolladores J2EE. En nuestro caso no descargamos la versión indigo para sistemas Windows de 64 bit.

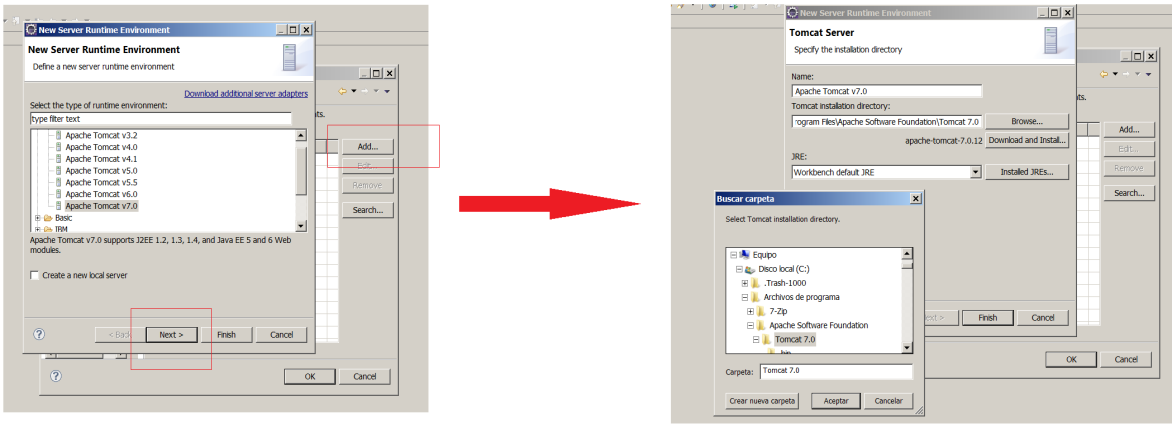
Una vez descargado, los descomprimimos en la carpeta donde deseemos tenerlo y accedemos a la carpeta, donde ejecutamos el programa eclipse.exe.

Elegimos el lugar donde queremos tener nuestro espacio de trabajo y aceptamos:



Inicio del entorno de desarrollo.

Una vez cargado el entorno, tendremos que elegir e instalar en el mismo el servidor utilizado, en nuestro caso un servidor Apache Tomcat en versión 7:



Instalación en eclipse de Tomcat v7.0

Con ello ya tendremos instalado el entorno de desarrollo y podemos importar el proyecto Hostmanager en el mismo, compilarlo y ejecutarlo. Si queremos hacer un nuevo proyecto con JSF, será necesario crear un nuevo proyecto WEB Dinámico y elegir la instalación de JSF 2.0. Para instalar ICEFaces con JSF2, debemos ir a la página de ICEFaces, bajarnos el plugin para eclipse e instalarlo en el entorno.

PostgesSQL

PostgresSQL puede ser descargado desde su sitio web. Aquí utilizamos la versión 9 del mismo. Para poder utilizarlo con la aplicación, debemos crear una base de datos llamada “dbnm”. Una vez hecho esto, debemos ejecutar los scripts facilitados para la creación de las tablas y procedimientos almacenados y crear desde aquí el usuario administrador para entrar en la aplicación. Los scripts han de ejecutarse en el siguiente orden:

1. schemas.sql
2. datatypes.sql
3. tables.sql
4. views.sql
5. functions.sql

PsTools

Se hace uso de este paquete de herramientas para recuperar información de los equipos. Por lo tanto es necesario bajarlo y copiarlo dentro de la carpeta system32 del directorio donde esté instalado Windows.

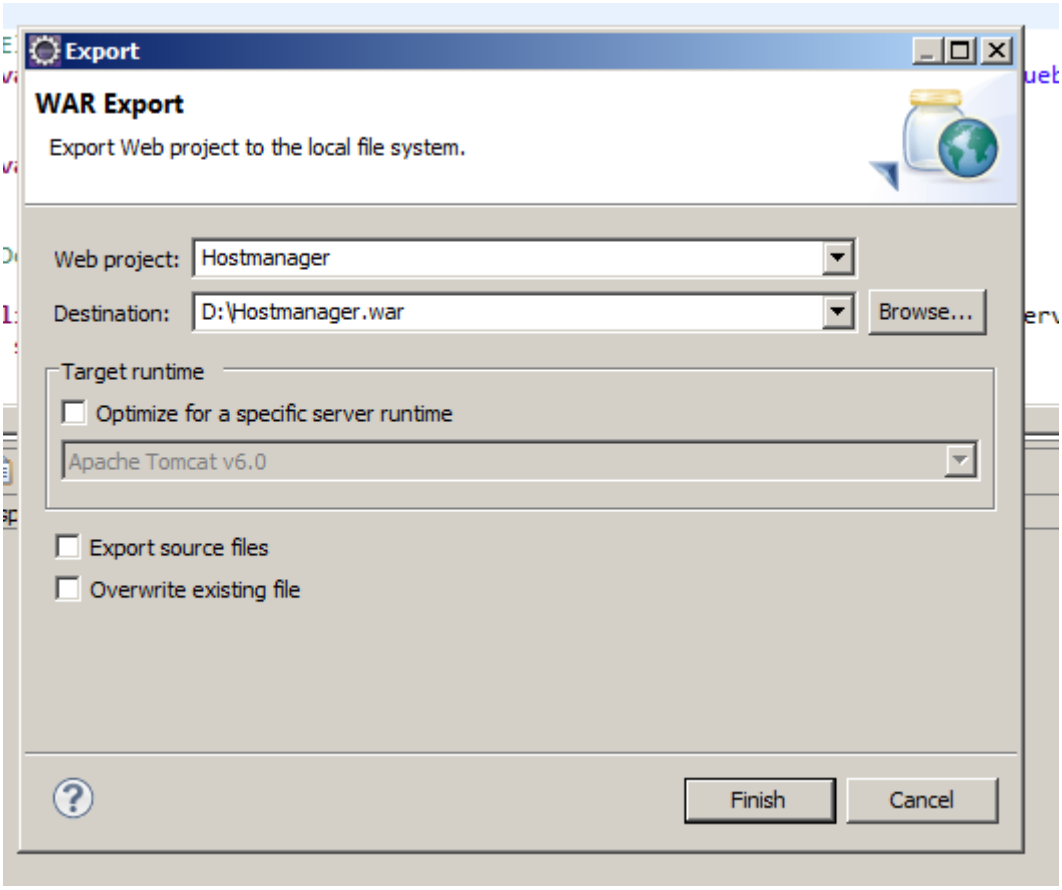
http://technet.microsoft.com/en-us/sysinternals/bb896649

Instalación del WAR

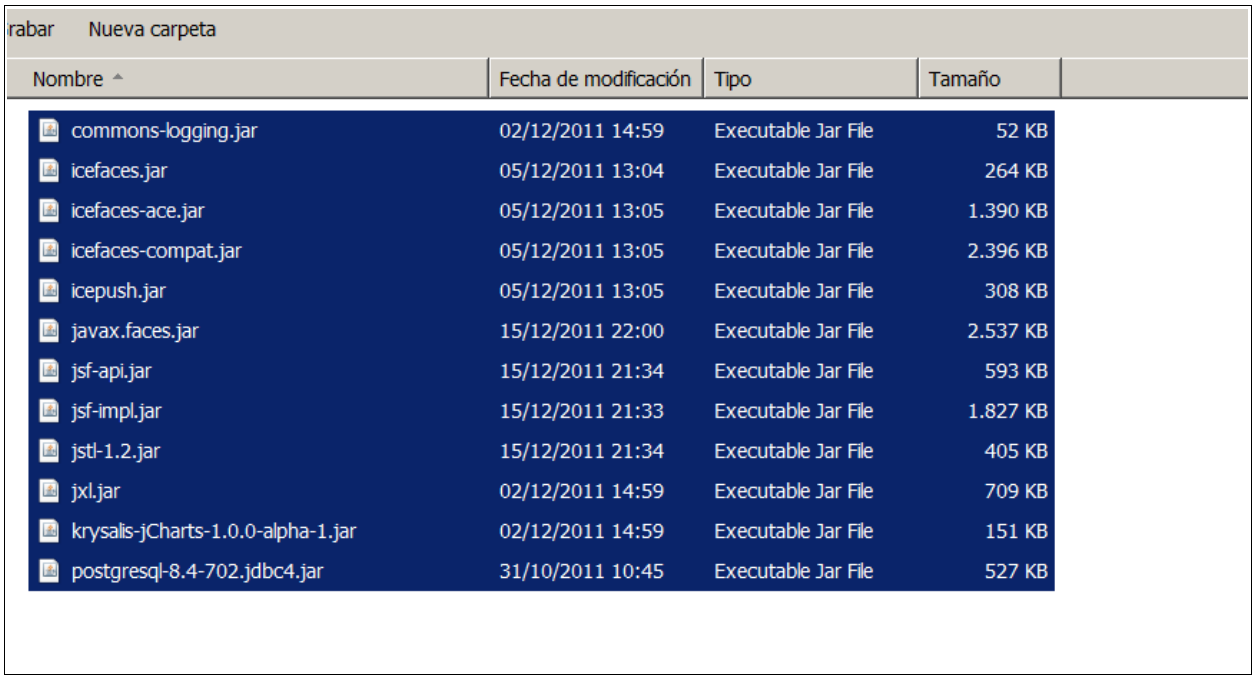
Una vez configurado el servidor apache y exportada la aplicación desde eclipse, es necesario instalar el WAR.

1. Exportación desde eclipse

Pulsamos encima de la carpeta del proyecto el botón derecho del ratón y damos a exportar como WAR:



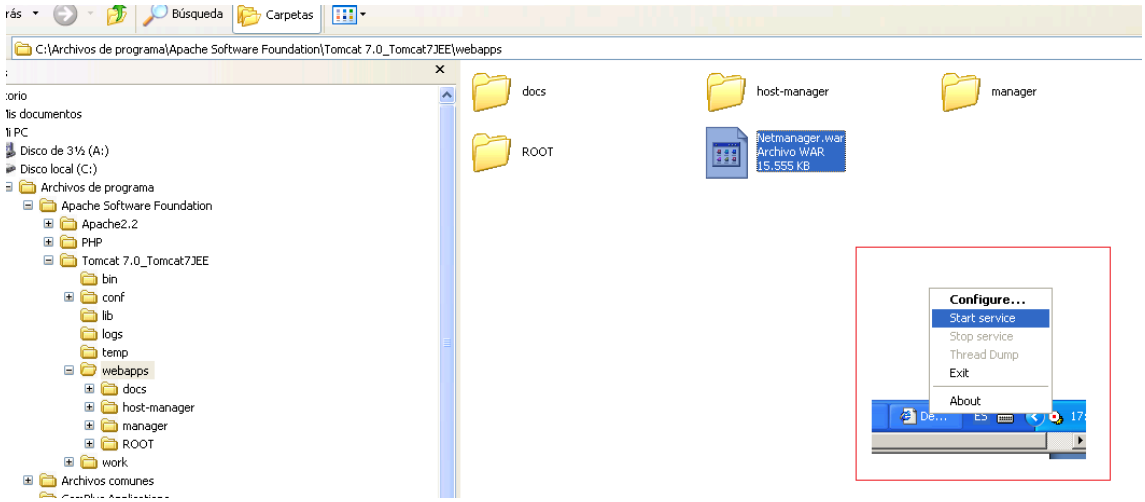
Una vez hecho esto, tenemos ya el WAR y podemos copiarlo dentro de la carpeta correspondiente del servidor Tomcat. Este WAR contiene dentro de la carpeta lib todos los jar de la aplicación, para conseguir y WAR más libiano, se descomprimió con WINZIP y se eliminaron de la misma esos ficheros:



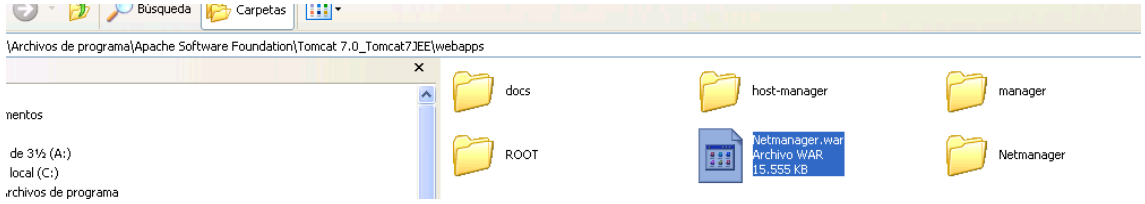
Con lo que a la hora de la instación deberán copiarse los mismos a la carpeta correspondiente. Muchos de ellos son de la biblioteca de ICEFaces. El jar de postgresSql debe bajarse de su página y los de jstl y jsf deben copiarse de la biblioteca correspondiente de eclipse (como los

de ICEFaces), ya que para la correcta ejecución en el servidor, es necesario que estén en esta carpeta.

2. Instalación en el servidor



Al arrancar el servidor se creará la correspondiente carpeta y se podrá ejecutar la aplicación:



Ahi que tener en cuenta la configuración de los ficheros y localización de los mismos definidas en el ConfigurationPathAppManager.java (por defecto son las que se utilizaron en el entorno de desarrollo Eclipse.

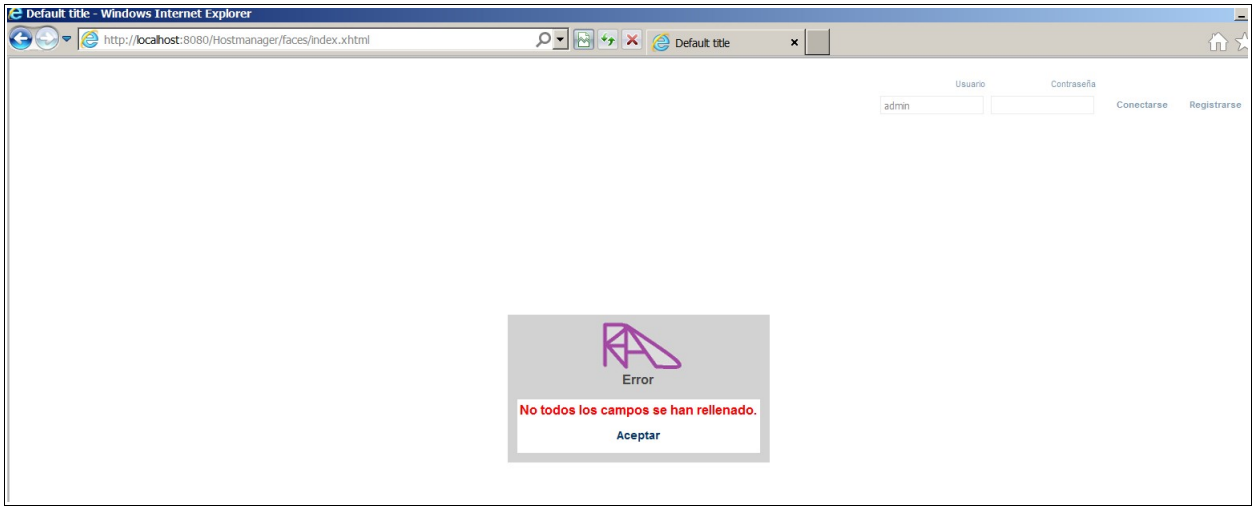
Juego de pruebas y manual de uso

Se realizará una recopilación de las correspondientes navegaciones y funcionalidades del projecto sirviendo también como manual de uso del mismo.

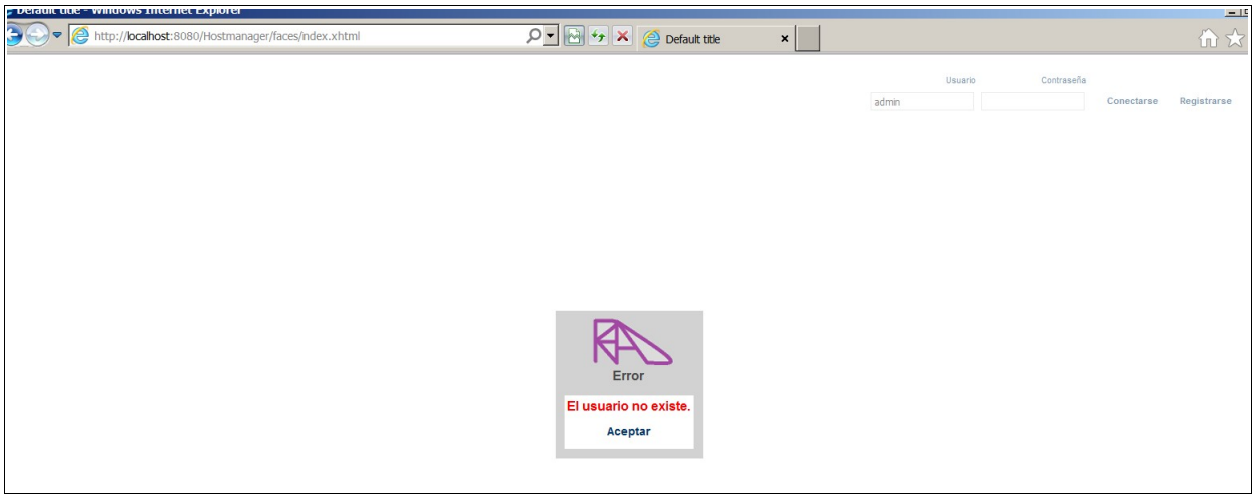
Comenzaremos por las opciones referentes al usuario administrador.

1. Logado en la aplicación

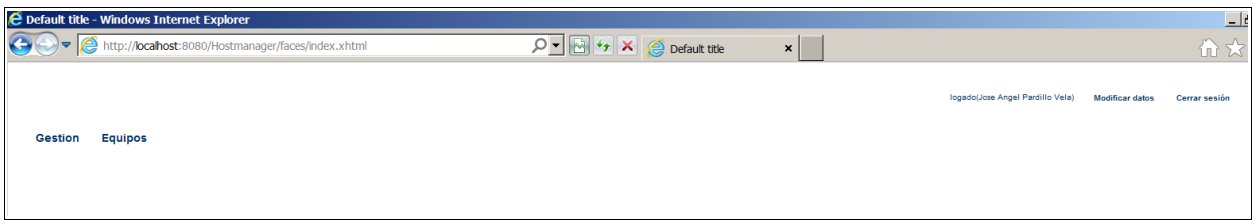
- Faltan datos por completar. Si dejamos sin rellenar alguno de los inputs, al dar al botón conectar, obtendremos el siguiente mensaje.



- Usuario/contraseña erróneos. Si rellenamos algún campo de forma errónea.

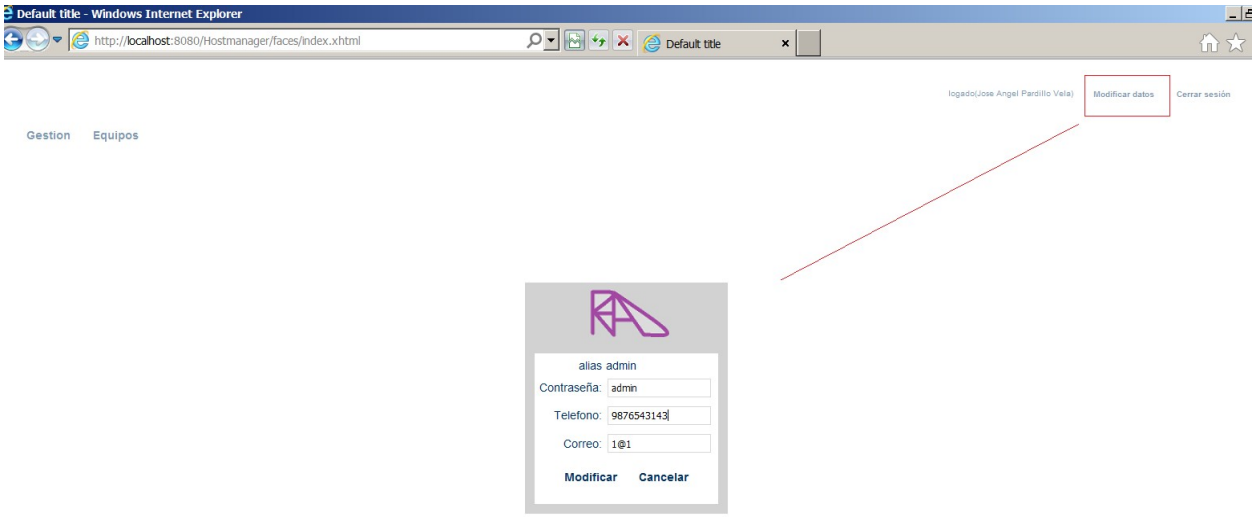


- Conexión correcta y página inicial. Si se ha producido un login correcto, pasaremos a la página inicial, donde veremos el menu de la aplicación y el del usuario.



2. Modificación de los datos del usuario

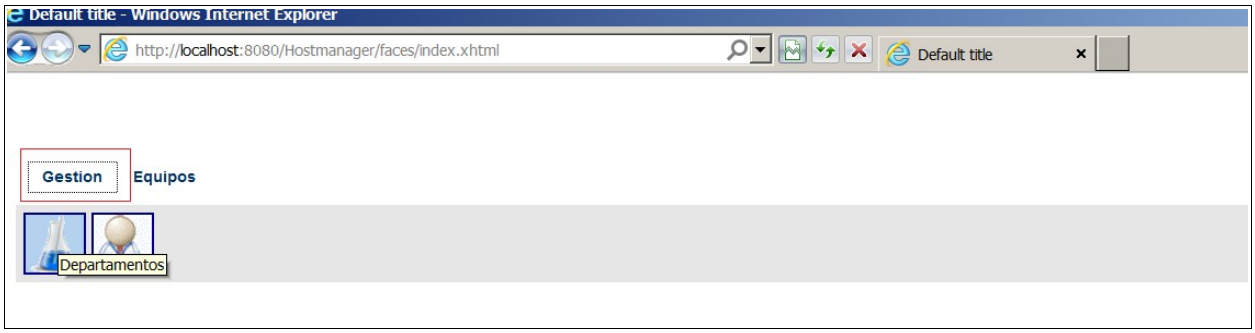
Al dar a modificar los datos, obtendremos un menú emergente donde podremos cambiar los datos del usuario o cancelar la operación.



3. Uso del menu de usuarios

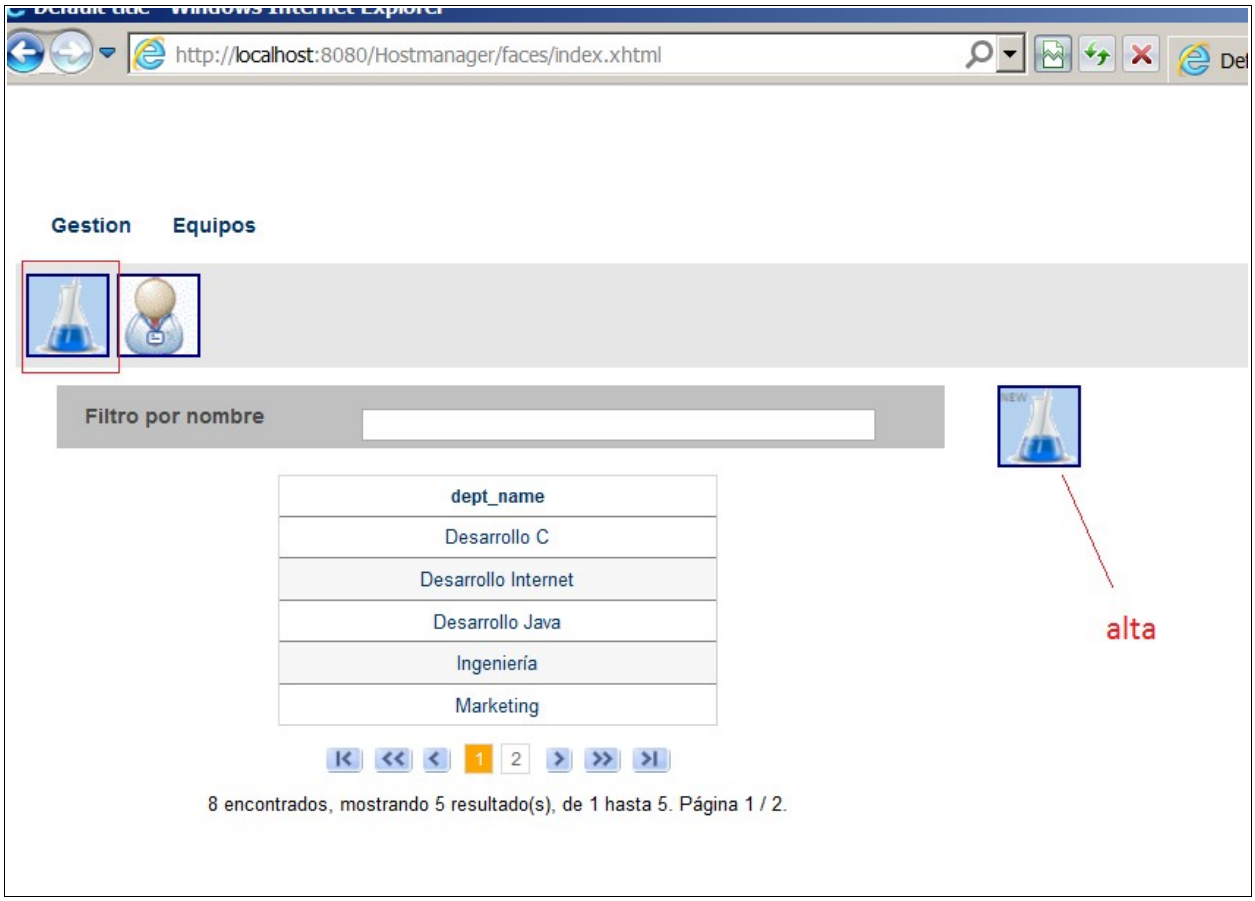
- Menú de gestión

Si pulsamos sobre el mismo, aparecerán dos nuevas opciones, referentes a departamentos (ver sus tooltips) y a usuarios:

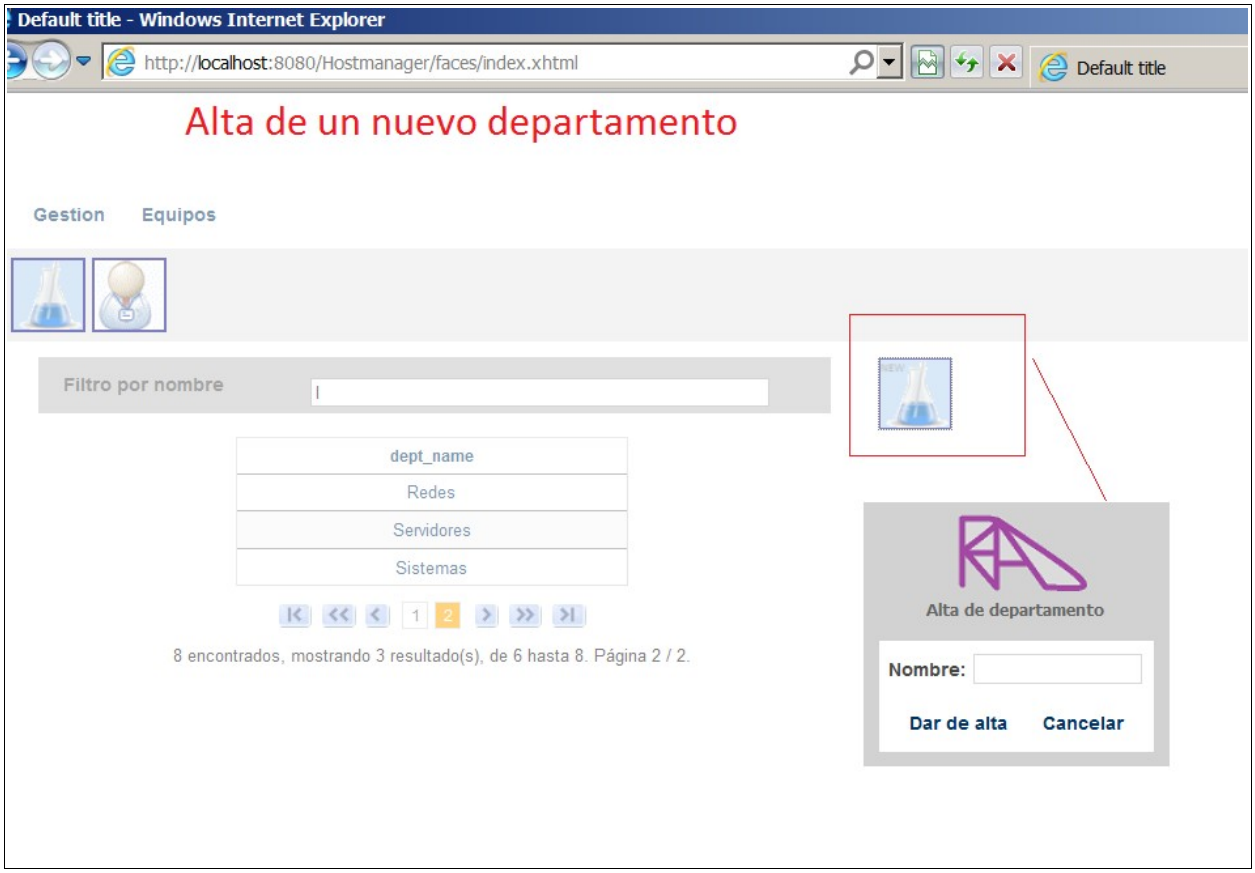


- Página de departamentos

Si pulsamos sobre el icono del laboratorio, iremos a la página de departamentos. Aquí tenemos las opciones de filtrar por departamento y dar de alta uno nuevo. También podemos ver la paginación.

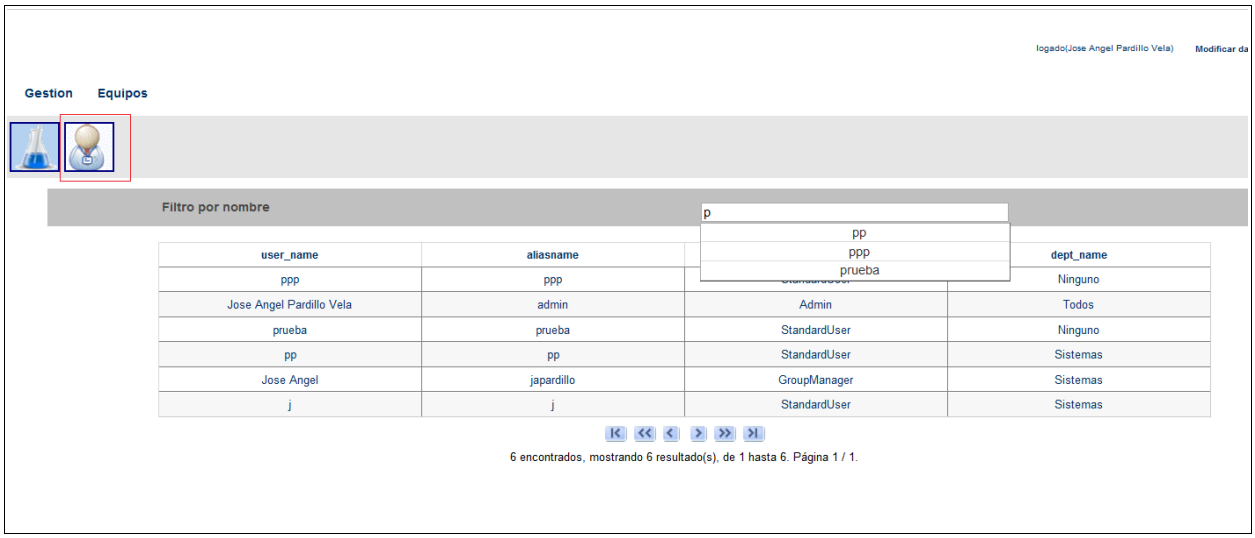



Default title - Windows Internet Explorer
http://localhost:8080/Hostmanager/faces/index.xhtml
Uso del filtro
Gestion
Equipos
Filtro por nombre
I
Ingeniería
Marketing
Redes
Servidores
Sistemas
Desarrollo Java
Ingeniería
Marketing
1
2
8 encontrados, mostrando 5 resultado(s), de 1 hasta 5. Página 1 / 2.

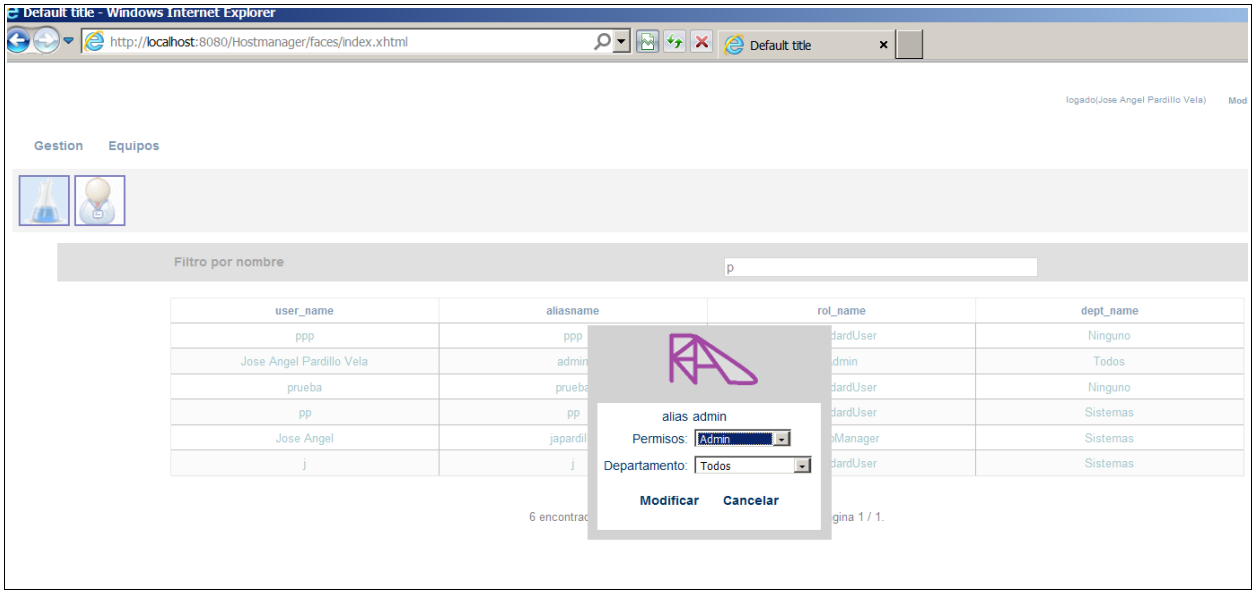


- Página de usuarios

En esta página podemos ver también funcionalidad similares en cuanto a filtros y paginación:



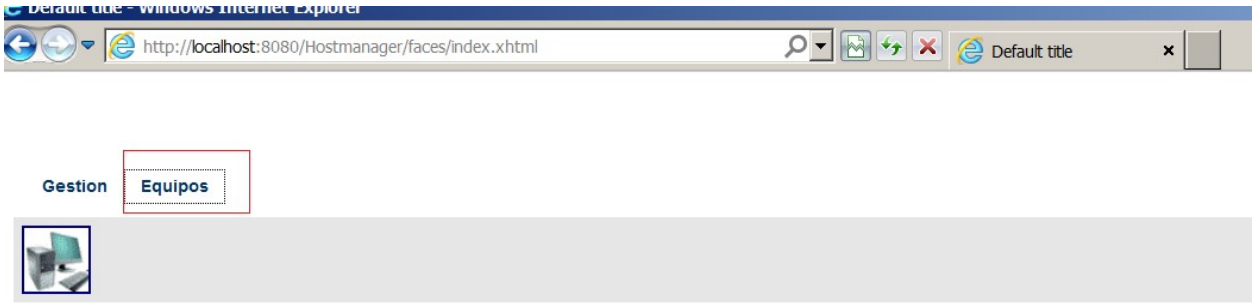
Pero además, las filas de la tabla son seleccionables y poseen un menú emergente:



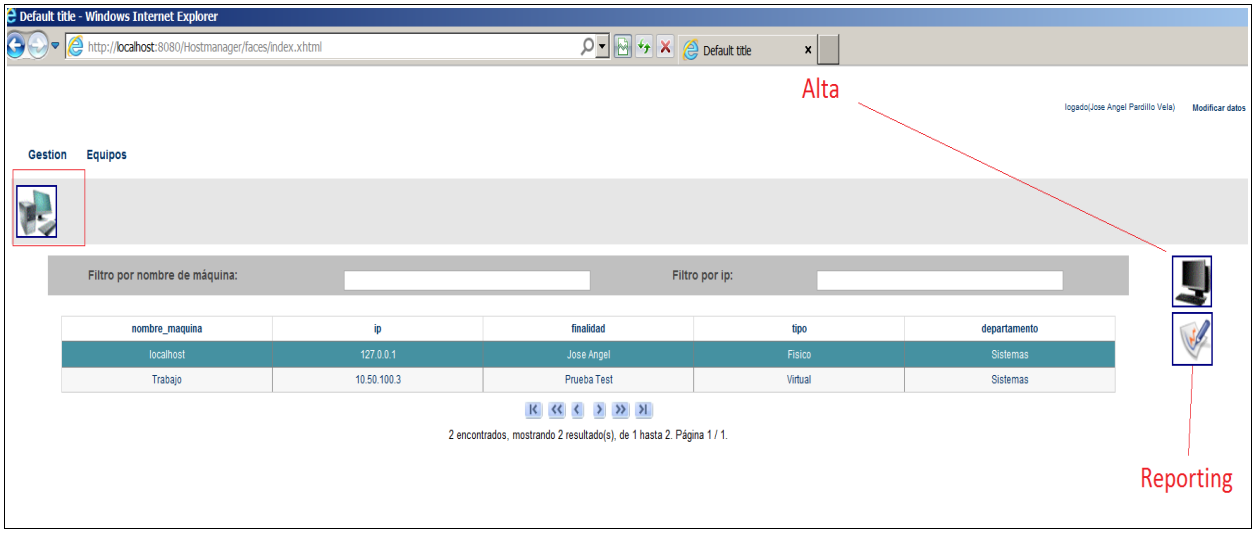
Donde podemos modificar los permisos de los usuarios y el departamento al que está asignados.

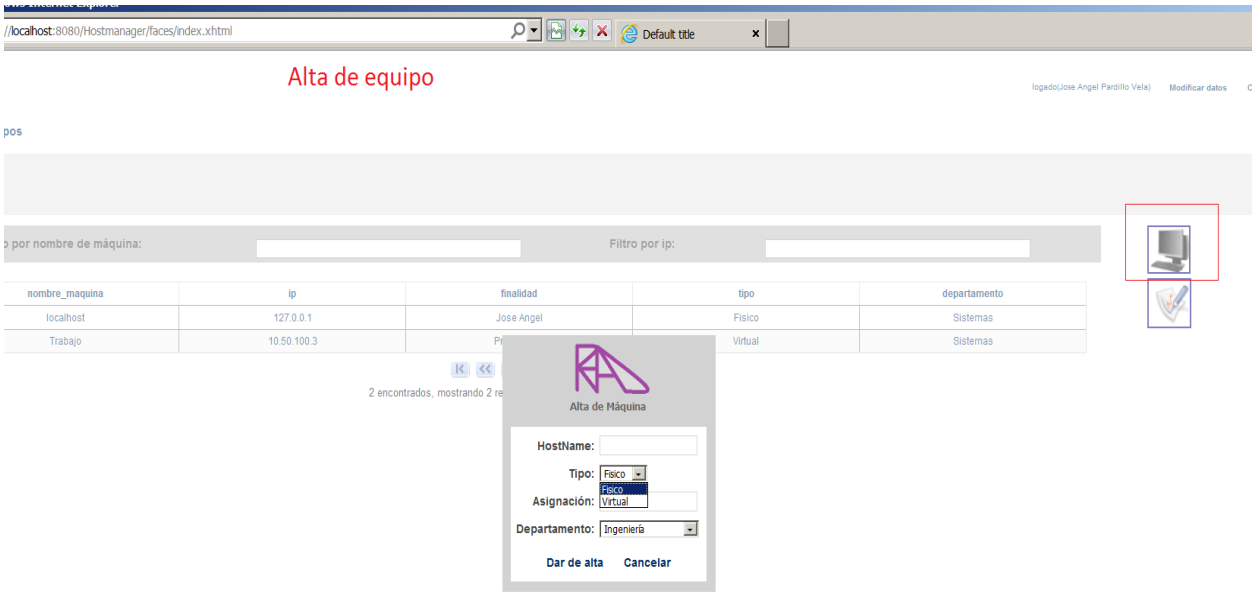
4. El menu de equipos

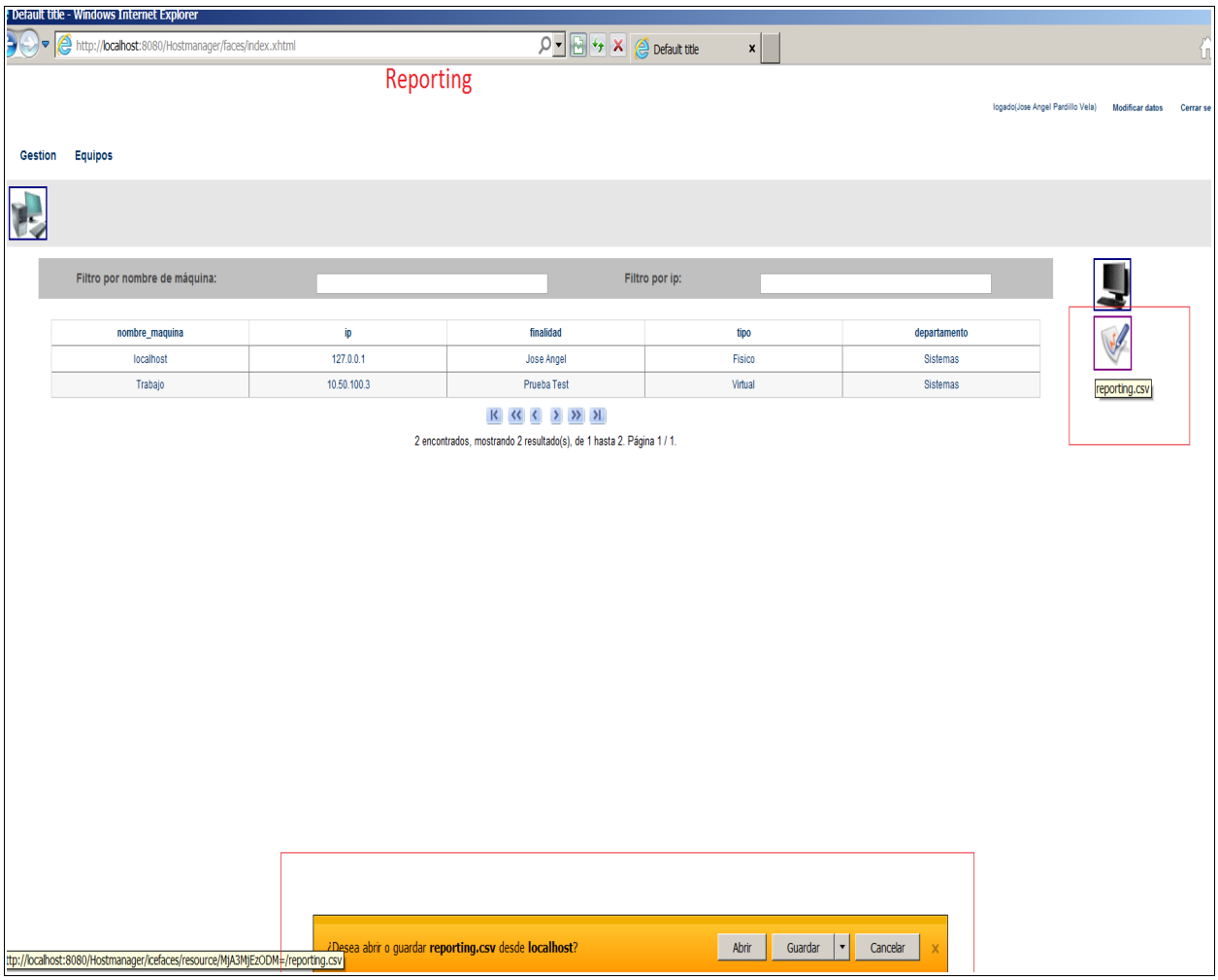
Si hacemos click sobre Equipos, podemos acceder al submenu correspondiente:



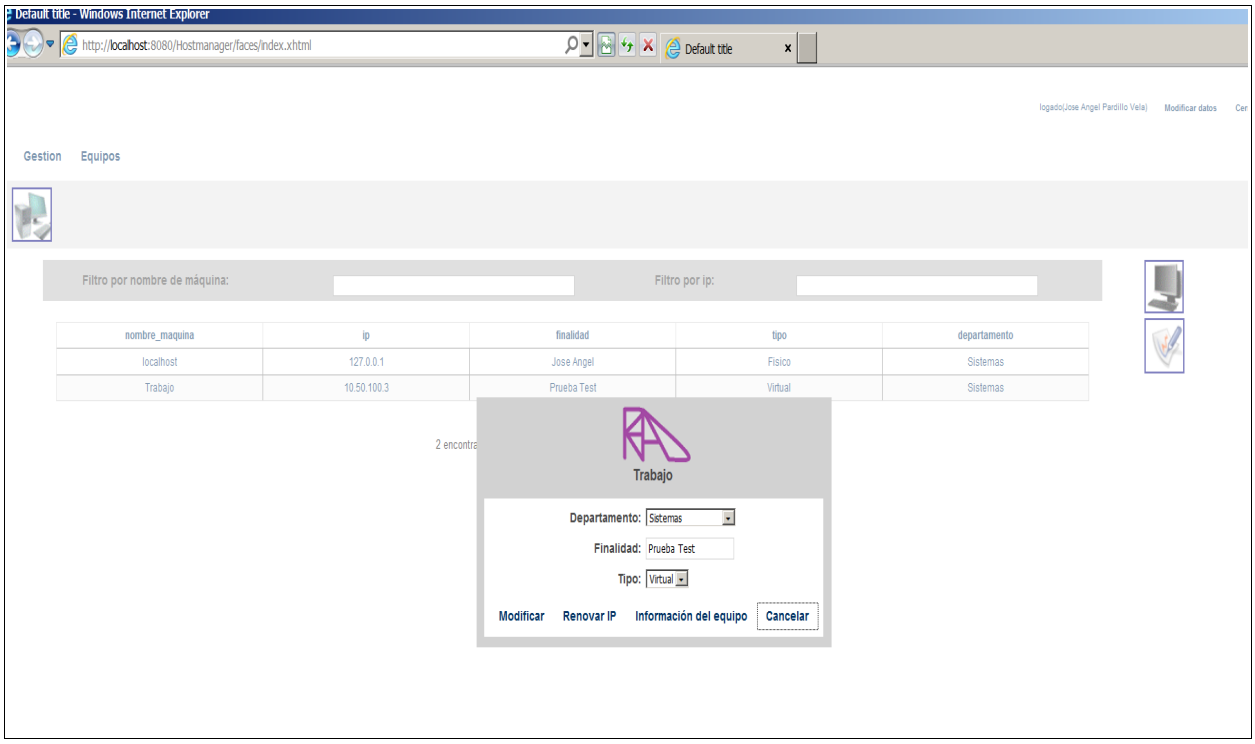
- La página de los equipos. Aquí tendremos varias opciones podremos dar de alta un nuevo equipo, generar un reporting de los equipos que existen ejecutar acciones sobre los mismos.

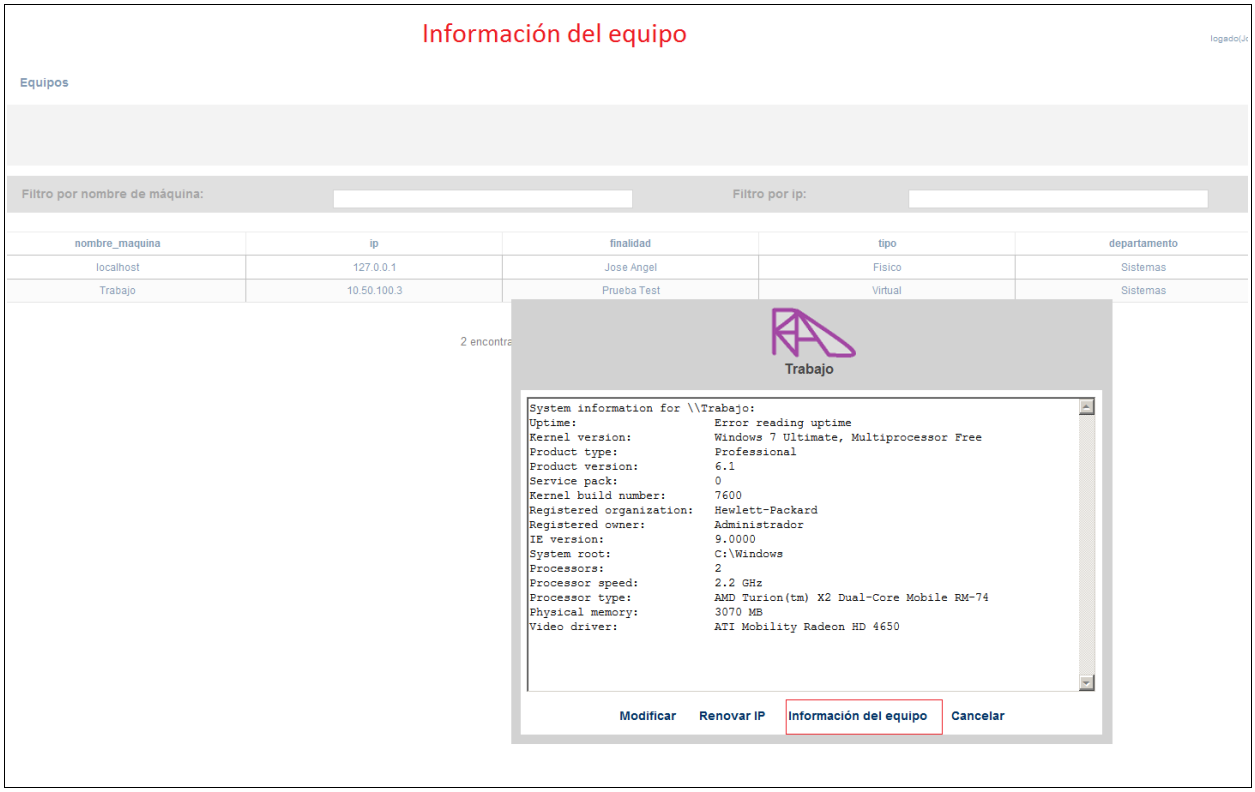






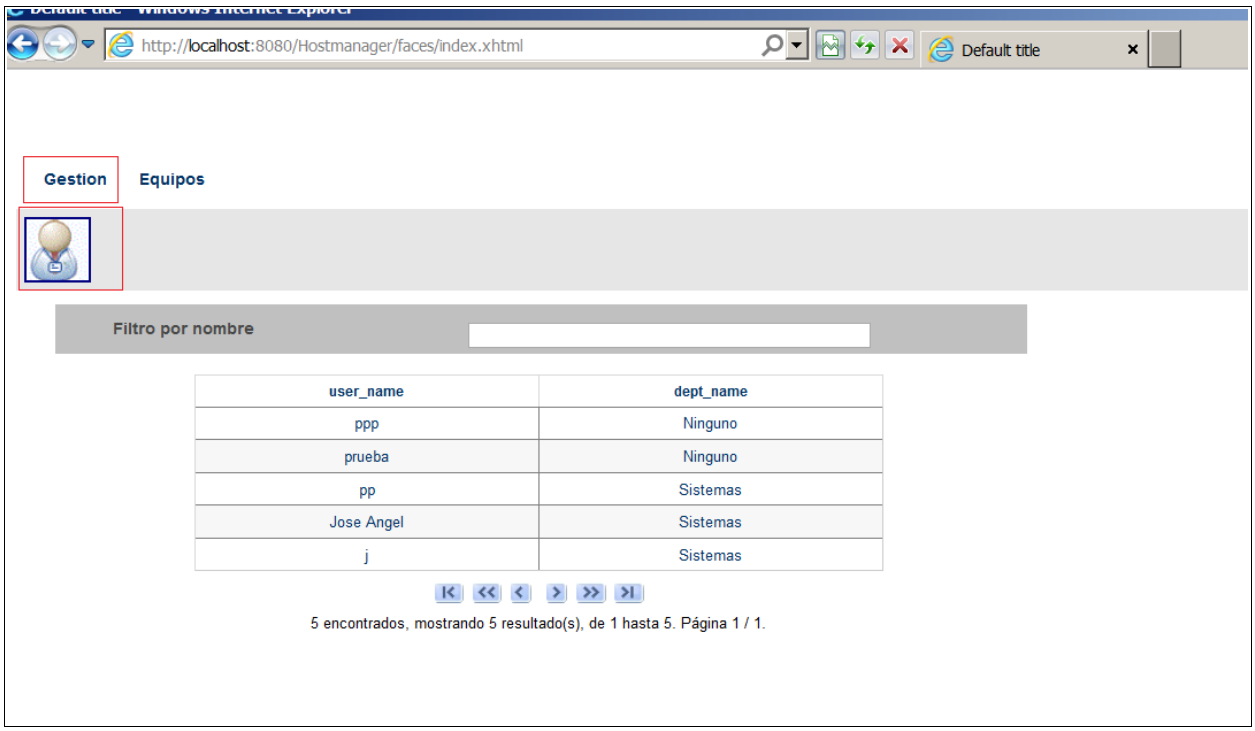
Al hacer click en una fila, obtenemos un menú emergente en el que podremos renovar la IP (se realiza un ping a la máquina), modificar datos y actualizar la BBDD o recuperar la información de los equipos mediante el uso de la herramienta psInfo:

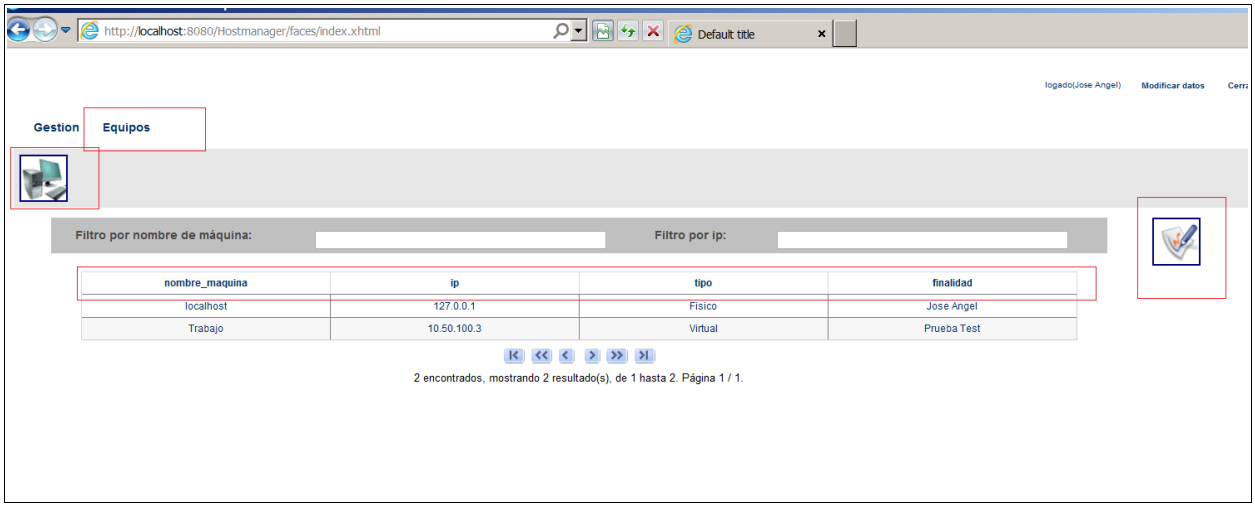


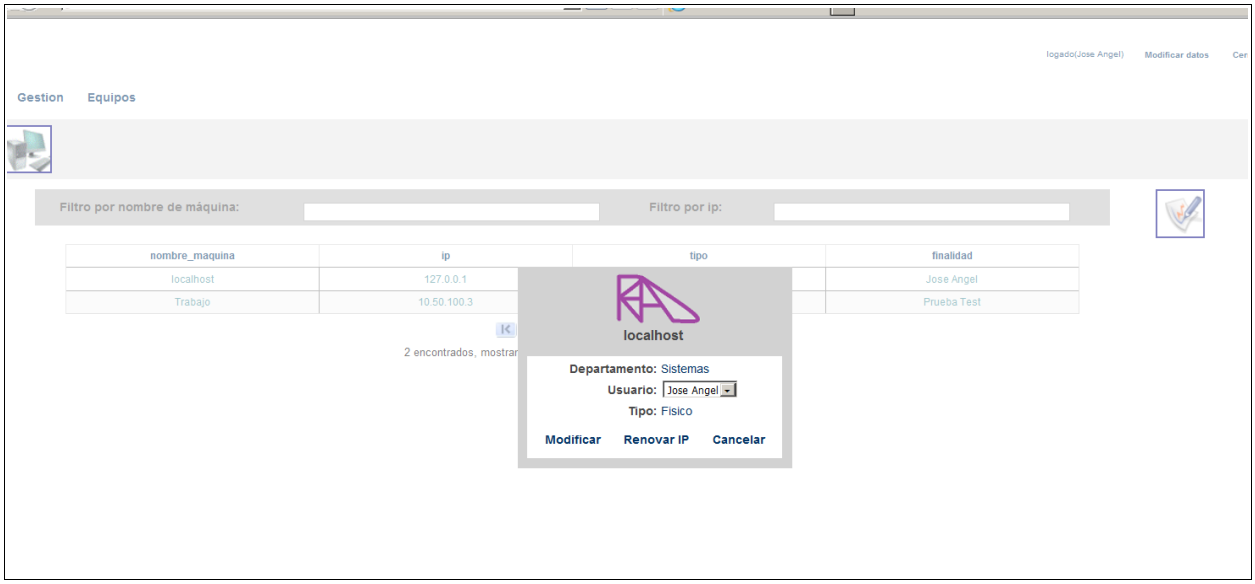


5. Navegación para un usuario tipo GroupManager

A continuación se muestran las pantallas que puede visualizar un GroupManager



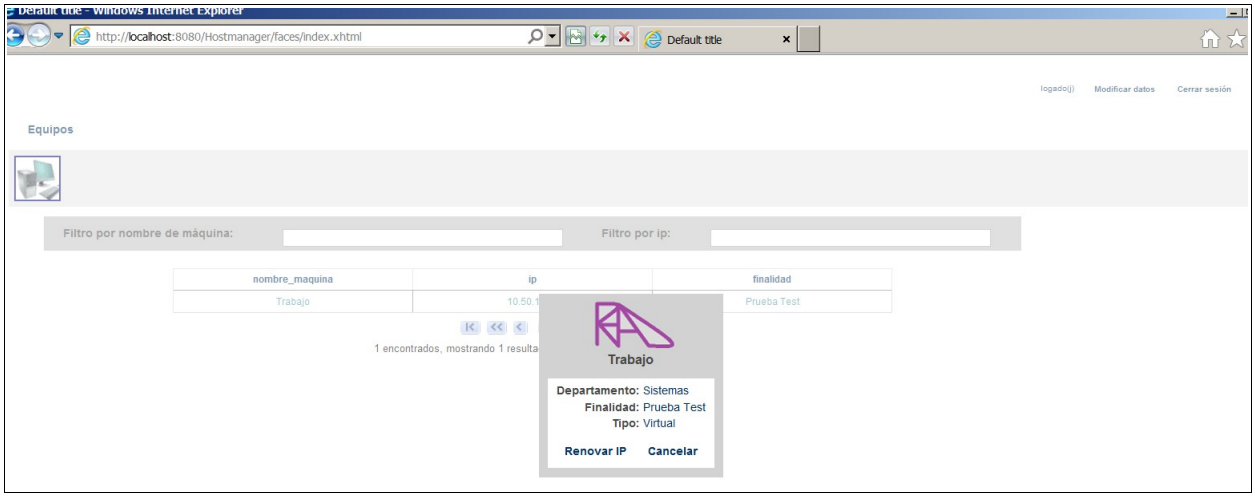




Como puede verse, se eliminan ciertas opciones tanto en los menús emergentes como en las propias páginas, apareciendo menos columnas en la tabla.

6. Navegación para un usuario tipo StandardUser

Como puede observarse, únicamente tiene acceso a la consulta de los equipos virtuales de su departamento, si no existe ninguno en su departamento, la tabla aparece vacía.



Conclusiones.

La utilización de patrones para cada una de las capas de la aplicación ha dado como resultado una simplificación del trabajo y ha favorecido la reutilización de componentes en cada una de las mismas.

Gracias al framework de JSF y al gran uso que hace del modelo vista controlador, se ha podido abstraer la presentación del desarrollo Java propiamente dicho. De hecho, en gran parte de las primeras etapas del proyecto se dedicó mucho más tiempo a la presentación que a la propia lógica de la aplicación.

Este hecho pone de manifiesto que pese a ofrecer una potente herramienta para el desarrollo, el aprendizaje de un framework concreto conlleva tiempo hasta conseguir un dominio lo suficientemente bueno como para poder desarrollar con coherencia una aplicación web.

Este es el mayor problema que se ha podido encontrar a la hora del desarrollo del mismo, junto con la modificación de las CSS que vienen por defecto en el framework. Para ello se tuvo que utilizar los depuradores ofrecidos por el navegador (FireBug para FireBox, el depurador JS de IE o la inspección de elementos de Chrome). Aunque es una CSS básica, posee muchos estilos que usan por defecto los componentes de ICEFaces, con lo que se pasó una gran cantidad de tiempo depurando con el navegador las hojas para obtener estilos propios a partir de los que cogen obligatoriamente los propios componentes. Esto puede llegar a hacer que aplicaciones desarrolladas por el mismo grupo de desarrollo obtengan similares aspectos en cuanto a usabilidad y aspecto.

Una vez se realizó gran parte del estudio del framework de presentación, se pasó a analizar la posibilidades de gestión de otro de los pilares de la aplicación el diseño de base de datos. En este aspecto la abstracción obtenida con el gestor DBManager fue muy alta, pudiendo cualquier con poco más de 3 lineas obtener los datos de una consulta sin esfuerzo.

Otro aspecto realmente crucial y crítico fue el nexo de unión entre la base de datos y las páginas. ICEFaces ofrece una gran cantidad de ejemplos y tutoriales, pero desgracidamente muchos de ellos no están totalmente actualizados, con lo que únicamente sirven como toma de

ideas y conceptos. En este aspecto, se quiso obtener un modelo para las tablas de la aplicación en la que la clase responsable de los datos fuera lo más genérica posible y que si quisiéramos cambiar las columnas a visualizar de la tabla fuera una tarea sencilla en la que hubiera que cambiar el mínimo número de elementos en la aplicación. Este nexo y objetivo fue conseguido con la clase NetManagerPageTableBean. Con ello un simple cambio de datos en el constructor de la página nos da como resultado un número distinto de columnas a visualizar sin tener que modificar ni añadir nuevos elementos a la presentación, lo que hace que el peso de la página se vea reducido.

Se estuvo haciendo gran cantidad de pruebas con los diferentes alcances (scope) de los Beans en JSF y el número de objetos que se crean con las peticiones y cual sería la forma más fácil de poder eliminar estas peticiones. Por ello utilizó en el SessionManager la herencia y casting de objetos con el fin de que esta clase pesara lo mínimo y obtener un compromiso entre creación de objetos y uso de memoria.

Otro aspecto de la aplicación, es el uso de la linea de comandos a través de JAVA mediante las herramientas de psTools para Windows, se realizó simplemente un ejemplo mediante el comando psInfo, pero pueden utilizarse cualquiera de ellos y queda como aspecto de posible ampliación la implementación de un mayor número de funcionalidades respecto a esta herramienta.

En conclusión, durante el desarrollo del trabajo fin de carrera se ha podido ver de forma general como es el desarrollo de una aplicación con la arquitectura J2EE y las principales tecnologías en las que se apoya para dar como resultado un software fiable.

Bibliografía.

La utilización de patrones para cada una

Gestión de expedientes para ayuntamientos
J2EE MVC
PFC : Diseño de un Framework de presentación
Diseño e implementación de un marco de trabajo de presentación para aplicación J2EE
http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=IntroduccionJSFJava
http://facestutorials.icefaces.org/tutorial/facelets-tutorial.html
http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=faceletsEclipse
http://www.danilat.com/weblog/2007/07/06/empezando-con-facelets/
http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=migrateJSF2Facelets
http://refcardz.dzone.com/refcardz/core-java?oid=ban00025-0
http://www.icefaces.org