Trabajo de Fin de Carrera - **Java EE**

Inmuebles

Portal de anuncios inmobiliarios

Ricardo de los Ríos Sánchez

Consultor: Oscar Escudero Sánchez

ETIS – UOC
20 de Junio de 2011

La presente memoria explica detalladamente el proceso de definición, análisis, diseño e implementación de un portal basado en la arquitectura Java EE que sintetiza los conocimientos adquiridos en la carrera de Ingeniería Técnica de Sistemas cursada en la UOC.

ÍNDICE

1. Introducción

La característica fundamental que presenta el panorama tecnológico actual es la complejidad, y el mundo de las aplicaciones informáticas es un ejemplo manifiesto. Basta echar un vistazo a cualquier librería comercial para observar la gran variedad de sistemas operativos, lenguajes y plataformas disponibles para realizar aplicaciones que van desde una sencilla calculadora a un complejo sitio de Internet, con bases de datos y arquitecturas cliente-servidor subyacentes, que son capaces de procesar ingentes cantidades de información con eficacia ofreciendo servicios a millones de usuarios.

La plataforma *Java EE* es una respuesta eficaz ante el desafío de construir, mantener y evolucionar pequeñas y grandes aplicaciones para Internet con un rendimiento y control óptimos heredando las prestaciones de portabilidad y reusabilidad que ofrece el lenguaje Java.

1.1. Justificación del TFC

El presente trabajo de fin de carrera pretende alcanzar dos objetivos primordiales:

- Realizar una **aplicación web** basada en tecnología Java EE que sintetice los conocimientos adquiridos en las asignaturas de la carrera de Ingeniería Técnica de Sistemas.

- Ganar experiencia y conocimiento de las tecnologías subyacentes mediante la realización de un **caso práctico avanzado** que cumpla con los criterios de calidad requeridos para una aplicación de esta índole. Se debe realizar un recorrido documentado de todas las fases del ciclo de desarrollo habitual: análisis, diseño e implementación. Además, se construirá, probará y entregará el código completo de dicha aplicación en las fechas indicadas al efecto.

Estos objetivos se concretan en la realización de un portal web para **la gestión de anuncios de una empresa inmobiliaria**.

1.2. Objetivos del TFC

A continuación se presentan los objetivos perseguidos en este trabajo clasificados en funcionales y técnicos. Los objetivos funcionales se refieren a "qué hace el portal", especificando los usuarios y roles asociados, así como los itinerarios, procesos y tareas que la aplicación realiza, sin entrar en detalles técnicos de plataforma o implementación. Los objetivos técnicos hacen hincapié en las plataformas y marcos de trabajo utilizados y el detalle de uso de los mismos.

1.2.1. Objetivos funcionales

El objetivo funcional de la aplicación será la de **proporcionar una herramienta de registro y consulta de anuncios inmobiliarios**, para alquiler o venta, accesible por múltiples usuarios vía Internet.

Un usuario registrado podrá tanto consultar como publicar sus anuncios y dispondrá de formularios en la interfaz para introducir la información del inmueble y sus características de forma cómoda, segura y amigable.

La información que consultará un usuario invitado referente a un inmueble se facilitará mediante formularios de consulta con múltiples filtros con el fin de afinar la búsqueda según criterios de zona y características de los pisos.

Opcionalmente se posibilitará la comunicación mediante mensajes de texto en el mismo portal entre anunciantes e interesados. Otra posibilidad sería incluir un sistema de alarmas por correo electrónico que avisara de cambios en los precios.

1.2.2. Objetivos técnicos

La construcción de la aplicación de este trabajo se basará en el lenguaje de programación Java sobre una arquitectura Java EE, lo cual garantiza la robustez, escalabilidad, extensibilidad y reusabilidad intrínseca perseguida por esta tecnología.

Al tratarse de un aplicación web, se utilizará como base el patrón de diseño MVC que proporciona Struts 2, un framework de reciente aparición diseñado para crear aplicaciones de empresa Java EE y que proporciona conceptos, prácticas y criterios metodológicos que cubren de forma probada el ciclo de desarrollo completo de este tipo de aplicación: construcción, desarrollo, mantenimiento y evolución posterior.

El uso del patrón MVC permite el desacople de las tres capas fundamentales: capa de presentación del portal, capa de acceso a los datos y lógica de negocio y capa controladora. De esta manera, el código se fragmenta ganando en reusabilidad y autonomía.

El entorno de desarrollo elegido es *Netbeans IDE 7.0* de la recientemente adquirida por *Oracle, Sun Microsystems*, que ofrece editores de código *Java*, páginas *JSP* con *CSS*, refactorización de código e integración con servidores *Java EE* como *Apache Tomcat*. Este entorno facilita la codificación controlada de la aplicación y la realización de las pruebas necesarias para su validación.

Como sistema de gestión de base de datos, se elige *Oracle*, en particular la edición express 10g, estudiada en profundidad en la asignatura Sistemas de Gestión de Bases de Datos y que ofrece excelentes prestaciones para el diseño y mantenimiento de las bases de datos y consultas que se realicen sobre ellos.

1.4. Planificación inicial del proyecto

La elaboración de este trabajo se ha dividido en 4 etapas principales que han dado como resultado un entregable por cada etapa. La ilustración 1 muestra un desglose de cada fase. Cada una de estas fases corresponde a un hito fundamental del ciclo de desarrollo de software habitual, a saber: planificación, análisis, implementación y pruebas.

Entregable	Fase	Tareas	Duración estimada
PEC1	**Planificación**	• Definición de requisitos • Objetivos • Plan de trabajo	8 días
PEC2	**Análisis y diseño**	• Arquitectura • Requerimientos funcionales • Casos de uso • Diagramas de clase • Diagramas de colaboración • Diseño de BD	20 días
PEC3	**Implementación**	• Instalación herramientas • Diseño físico/Creación BD • Implementación Acciones • Implementación modelo y clases entidad • Implementación presentación • Pruebas • Manual de instalación	30 días
Producto final	**Memoria y presentación virtual.**	• Correcciones • Redacción de la memoria • Elaboración de la presentación	20 días

Ilustración 1. Tabla de entregables y fases del proyecto

La ilustración 2 muestra un desglose de las tareas que se han llevado a cabo en forma de Diagrama de Gantt. Las tareas están ordenadas por etapas y se remarcan especialmente las fechas de entrega oficiales.

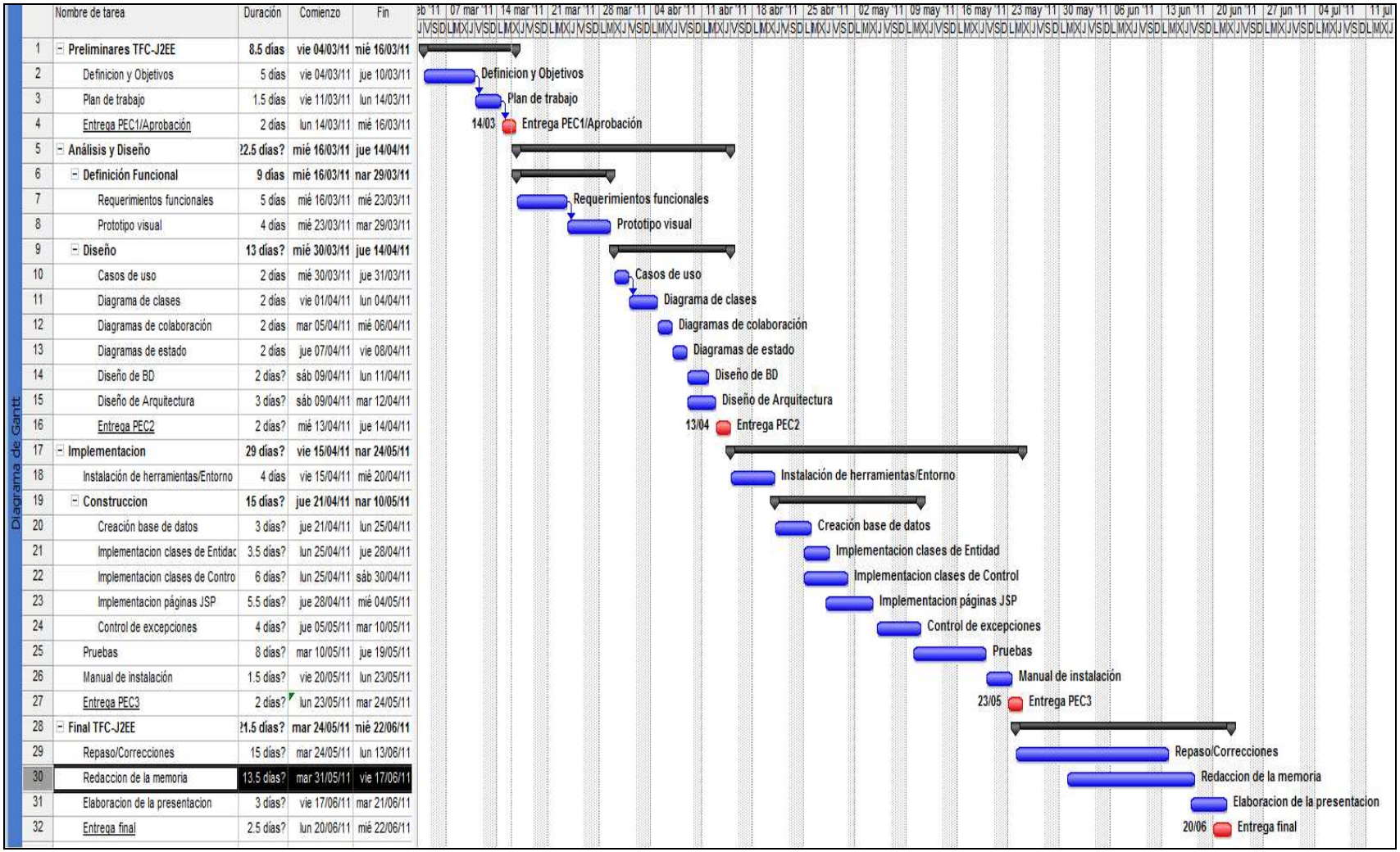


Ilustración 2. Diagrama de Gantt de las tareas a realizar en el presente trabajo.

1.5. Productos obtenidos

El producto final son 3 archivos *".zip"* comprimidos que contienen los siguientes ficheros:

- Memoria (*rdelo_memoria.zip*):
  - *rdelo_memoria.doc.* Es el presente documento que contiene la **memoria del TFC**.

- Presentación (*rdelo_presentacion.zip*):
  - *rdelo_presentacion.ppt*. Presentación mediante diapositivas donde se realiza la **exposición del trabajo** mostrando de forma visual las distintas fases que dieron lugar al producto final del proyecto.

- Producto (*rdelo_producto.zip*):
  - Inmuebles.war – Contiene la **aplicación Java EE** lista para desplegar.
  - Inmuebles.xml – Fichero de **configuración** de la aplicación.
  - src.zip - Contiene los **ficheros fuente** de la aplicación.
  - bd.zip –Contiene los **scripts** de creación, borrado y poblado de la base de datos.
  - javadoc.zip. Contiene la **documentación técnica** detallada de la aplicación.

2. Análisis y diseño.

En esta fase se realiza la distinción de los distintos elementos que configuran la arquitectura de la aplicación y los diagramas UML que representan visualmente los elementos, flujos de proceso y estructuras que se deben respetar en la futura implementación.

2.1. Arquitectura.

La arquitectura general de la aplicación depende en gran medida del medio al que está destinada, en este caso, Internet y de su naturaleza cliente-servidor. El cliente final accederá a la funcionalidad del portal mediante un navegador web, lo cual significa que el contenido de la aplicación correrá en un servidor web que estará activo y preparado para dar soporte a las múltiples peticiones que los múltiples usuarios conectados realicen.

El servidor de aplicaciones elegido para desplegar la aplicación debe ser de tipo Java EE, como lo es *Tomcat* de *Apache Foundation*, que permite desplegar aplicaciones de empresa de una manera sencilla y controlada y que goza de reconocimiento y soporte a nivel internacional. Este contenedor generará una instancia de la aplicación a la cual adosamos las librerías externas requeridas.

Como se indicó en los objetivos de este trabajo, la aplicación Java EE Inmuebles está basada en el patrón Modelo-Vista-Controlador que sustenta el framework *Struts 2*. Es por tanto indispensable conocer la arquitectura de este framework para adaptar el diseño del portal. Partimos de un estudio de la arquitectura de este framework que adaptaremos y ampliaremos de forma personalizada.

A estos componentes básicos, hemos de añadir una base de datos que asegura la persistencia de la información manejada por los objetos de la aplicación. En la ilustración 3 podemos ver integrados todos estos elementos.

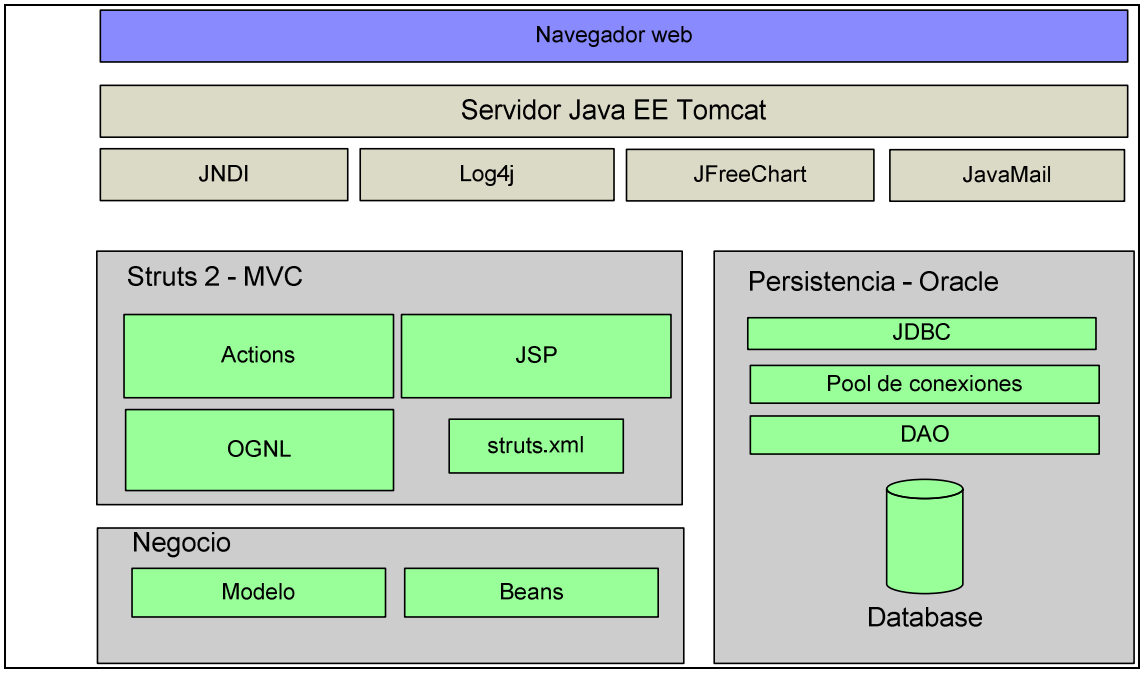


Ilustración 3. Arquitectura general de la aplicación.

2.1.1. Ciclo de vida de la petición HTTP en Struts 2

Cada vez que una petición HTTP es recibida desde el cliente, es procesada por el controlador de Struts 2, en particular por el filtro inicial *Dispatcher Filter,* que decide qué acción tomar. Este componente es declarado en el fichero web.xml – fichero de configuración de la aplicación web- por el desarrollador para delegar a *Struts 2* la toma de decisiones del flujo de lado del servidor. Elegida la acción a tomar, se ejecutan unos procesos intermedios de preparación llamados interceptores y a continuación se lanzan los métodos de negocio. Finalmente, la salida de resultados se renderiza en las paginas de servidor (jsp) asociadas y finalmente todo esta información es traducida a HTML y presentada de vuelta al cliente.

La relación de este proceso con el patrón MVC se ilustra en la Ilustración 4, donde se muestra la capa controladora donde se encuentra el filtro inicial, que mapea las acciones y ejecuta los interceptores; la capa Modelo donde las acciones acceden a la lógica de negocio (y a la base de datos si es preciso) y finalmente el renderizado a HTML en la capa de la Vista. Aunque las acciones son propias de la capa controlador, se han situado en la capa modelo para hacer mas visible el intercambio de datos con el controlador y la vista.

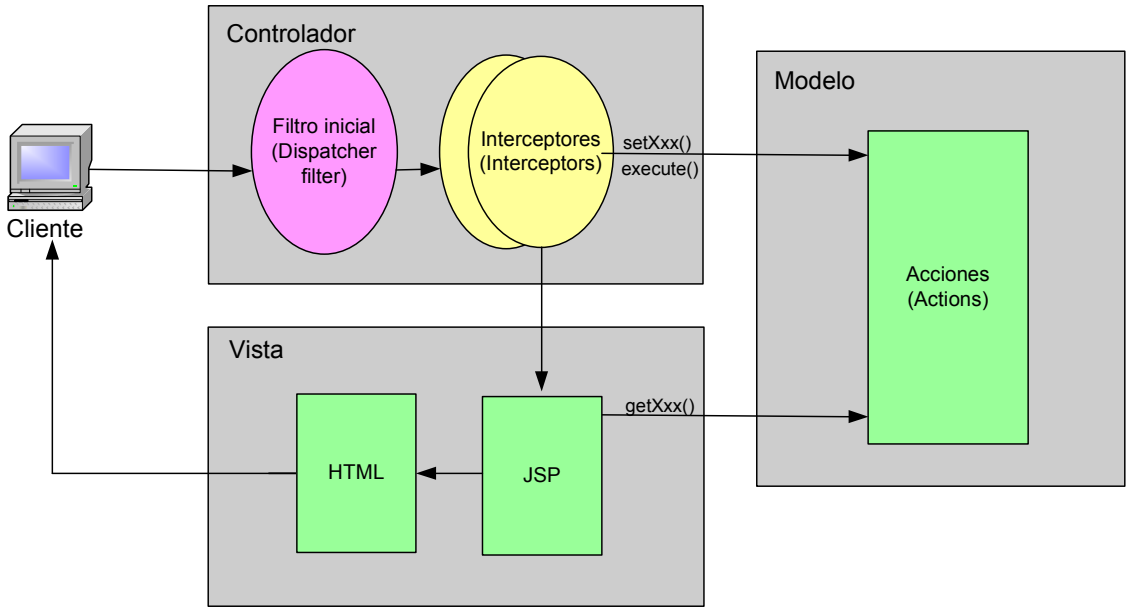


Ilustración 4. Patrón MVC y elementos principales de Struts 2 asociados.

Los elementos Action del controlador se comunican con los formularios alojados en las páginas jsp para recoger la información introducida por los usuarios. Estos objetos se encargan de invocar la lógica de negocio. Cuando se acceda a base de datos se realiza una llamada sobre un objeto de acceso a datos (capa DAO) cuya misión es invocar operaciones sobre los datos.

Este ordenamiento de la codificación permite realizar cambios en cualquiera de las capas del patrón de forma independiente y conectada. Facilita así incorporar separadamente ampliaciones y extensiones sin solapamientos de código y sin necesidad de modificar forzadamente o mezclar las tareas de las distintas capas.

Para introducir una nueva acción, bastará con declarar una entrada en el fichero de declaración de acciones (Struts.xml) y añadir al código fuente de la aplicación las clases de tipo acción y las páginas de servidor referenciadas. El objeto de arquitectura *ActionProxy* se encarga de crear los objetos de invocación, que implementan los métodos de negocio, basándose en el fichero de inicialización de struts mencionado (Struts.xml).

Los interceptores son elementos configurados para aplicar acciones comunes automáticamente, tales como validaciones, subida de ficheros y flujo de trabajo, antes o después y no durante la realización de las acciones (Ilustración 5).

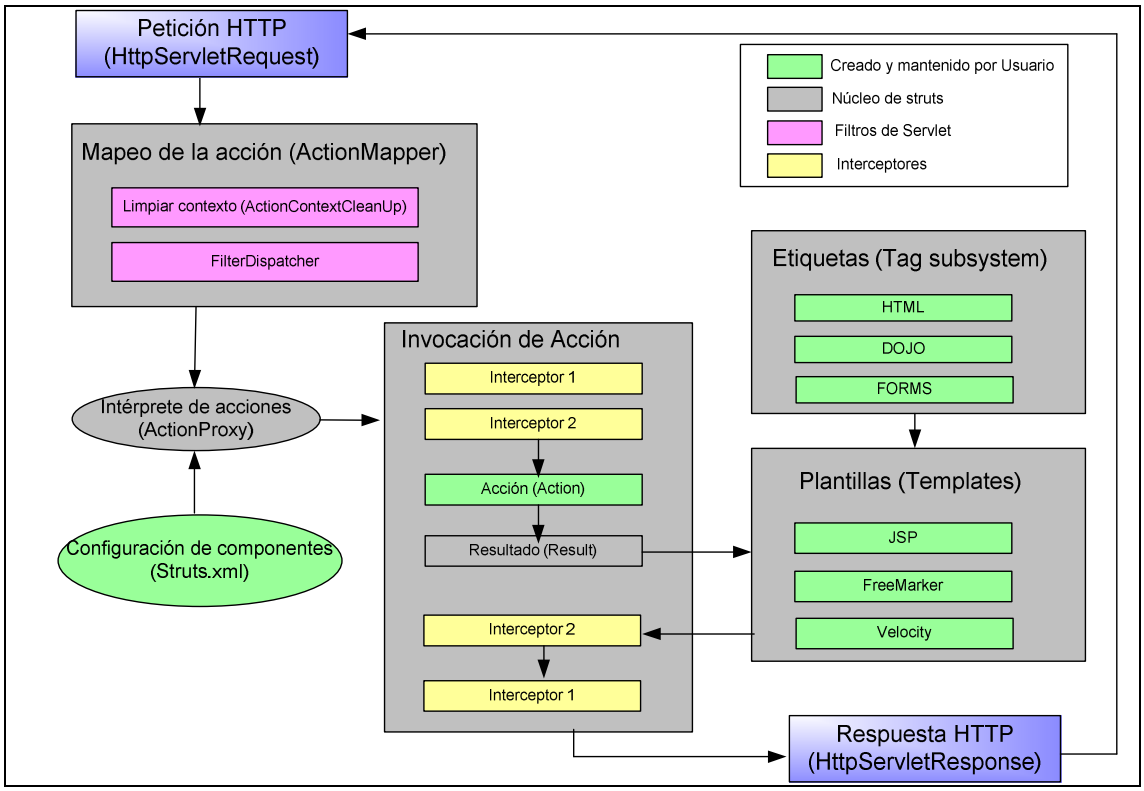


Ilustración 5. Ciclo detallado de petición HTTP en Struts 2.

La salida final se renderiza en el objeto Result, al cual se vuelven a aplicar los interceptores que realizan tareas de limpieza y procesado adicional. Una vez generado el código HTML, este es retornado al contenedor servlet que envía la salida al navegador.

2.2. *Patrones empleados*

Los patrones de diseño son soluciones contrastadas a problemas de software orientado a objetos que han ido moldeándose y fijando según la experiencia. Su aprendizaje y uso favorecen la aplicación de diseños ejemplares y facilitan y agilizan el diseño de componentes complejos y extensibles. Para la aplicación de este proyecto, a parte del patrón MVC que sirve de base a Struts 2, se han empleado otros de probada eficacia como Singleton y DAO.

2.2.1. MVC (Modelo-vista-controlador)

El componente MVC es de tipo arquitectural y genérico y se emplea para desacoplar las capas de presentación y negocio asignando una capa controladora que rige el flujo entre ambas. Esta estructura flexibiliza y hace adaptativos los elementos de la aplicación. Es preciso realizar un análisis cuidadoso para que las restricciones específicas de la aplicación no impidan su implementación.

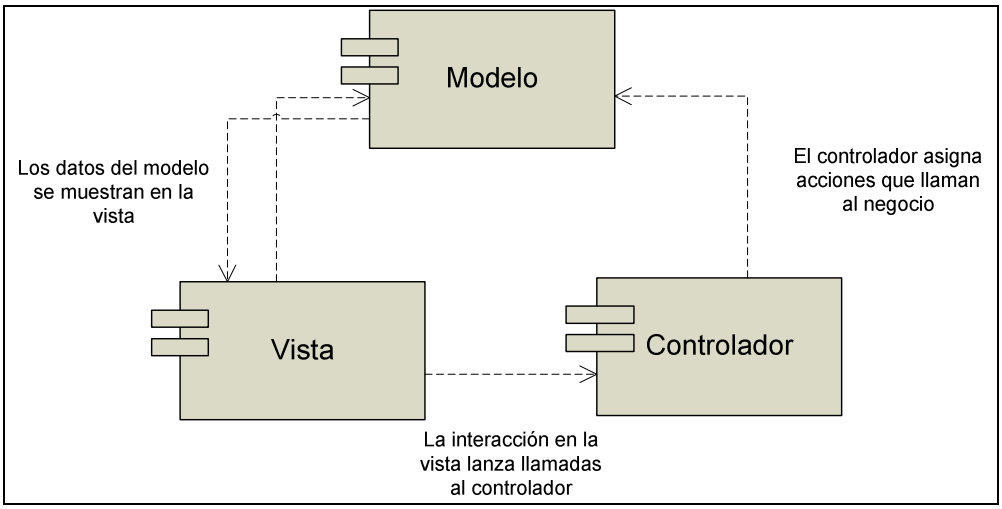


Ilustración 6. Diagrama de componentes del patrón MVC.

2.2.2. Singleton (único)

Este patrón permite guardar instancias únicas de conjuntos de datos que permanecerán prácticamente inalterados en las sucesivas ejecuciones de la aplicación, por ejemplo, el registro de comandos de la aplicación.

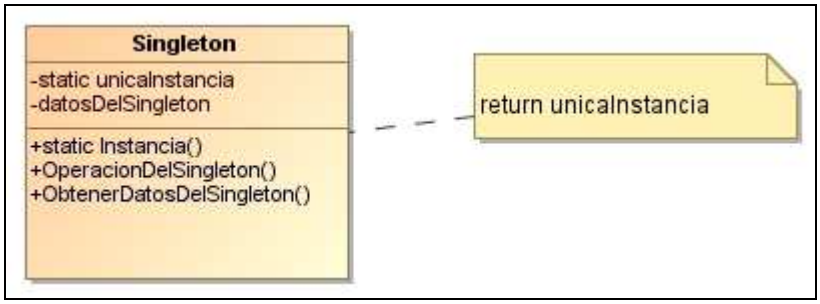


Ilustración 7. Estructura de clases del patrón Singleton.

Todo Singleton proporciona un punto de acceso global a su instancia. Además se encargará de interceptar cualquier petición de crear nuevos objetos derivados.

La instancia debe ser extensible mediante herencia y cualquier cliente debería poder usar estas extensiones sin modificar su código.

En el portal Inmuebles será empleado para mantener una instancia única de datos constantes para la aplicación: listas de información fija para selección en la interfaz (etiquetas de tipos de acciones, tipos de inmuebles etc) y log de la aplicación.

2.2.3. DAO (Objeto de acceso a datos)

Para facilita la persistencia de objetos de la aplicación empleamos el patron DAO. Este patrón gestiona la conexión a la base de datos de las clases de negocio que requieran persistencia. Basta con que cada clase de negocio herede la clase principal de este patrón para que disponga de acceso a base de datos, eximiéndola de responsabilidades de gestión de conexiones.

Cada clase del modelo debe poseer su propia clase derivada DAO, por ejemplo, una clase bean Cliente.java, debe tener su ClienteModeloDAO.java que contiene los métodos de acceso a base de datos que pueden ser de consulta (obtener listados), actualización, borrado y creación de registros.

La clase principal del patrón es ModeloDAO que implementa el método de la interfaz de conexión (getConnection()). En el código de la aplicación se va a utilizar un pool de conexiones (DataSource) que será utilizado por esta clase para gestionar múltiples conexiones entrantes optimizando el uso de los objetos de conexión. Además permite la transmisión de la conexión mediante el setter setConnection.

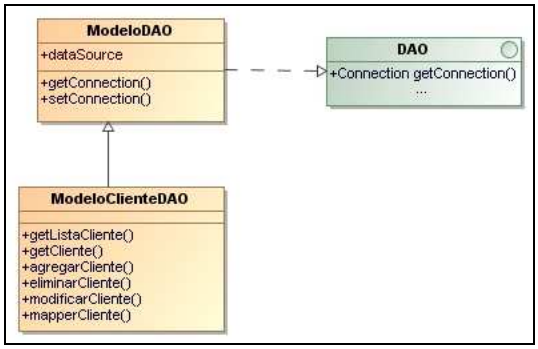


Ilustración 8. Patrón de diseño DAO.

2.3. Actores y revisión de los casos de uso.

Se distinguen tres perfiles que representan a los distintos tipos de usuario que accederán al sitio web: usuario invitado, usuario registrado y director. El diagrama de casos de uso facilita la visualización directa de las acciones que realizará cada perfil de forma clara e interrelacionada. Cada perfil se representa por un actor en el diagrama con el mismo nombre.

La elaboración del diagrama de casos de uso y su especificación nos permite adelantarnos a los posibles problemas funcionales que aparecerán en la interacción del usuario con el sistema. También permite hacernos una idea de los elementos que conformarán la interfaz visual.

2.3.1. Actor Invitado

Representa a un usuario cualquiera de Internet que accede al sitio para realizar búsquedas de anuncios inmobiliarios o bien para curiosear. Una vez acceda al portal elegirá la opción "entrar como invitado", sin registrarse ni estar registrado.

Este usuario puede utilizar los filtros de búsqueda para listar inmuebles que cumplan los criterios que le interesen. Estos criterios incluyen la localización geográfica del inmueble, tipo de vivienda (piso, garaje, oficina, etc), tipo de acción (alquiler, compra o compartir) y diversas características de la vivienda (alquiler garantizado, garaje, ascensor etc.). El portal le permitirá navegar por los resultados facilitando los datos de los propietarios de los inmuebles y de los pisos que cumplan los criterios. Existe la opción de denunciar un anuncio fraudulento, esto es, que no cumpla realmente con lo publicado.

En todo momento tendrá disponible la opción de registrarse para acceder a las ventajas que esto supone: guardar búsquedas favoritas, activar alertas, etc.

Para registrarse, el sistema solicitará al usuario datos de contacto y datos personales que almacenará acorde con la política de privacidad del portal.

2.3.2. Actor Registrado

Se trata de usuarios que desean, además de realizar búsquedas de inmuebles, publicar anuncios utilizando el sitio. Podrán llevar una gestión personal de los mismos y opcionalmente, mediante el pago de una cuota, obtener ventajas sobre otros anunciantes.

Tendrán a su disposición unos formularios de entrada de datos donde introducir la información referente al inmueble. Estos quedarán automáticamente publicados una vez solicitado en el portal.

En la selección de filtros pueden guardar búsquedas favoritas, activar alertas por email de inmuebles que cumplan con los filtros y comprar ventajas adicionales.

2.3.3. Actor Director

Este usuario puede consultar estadísticas internas que genera el portal además de gestionar las denuncias que pudiera haber por anuncios fraudulentos. Estas estadísticas mostrarán cuantos clientes están registrados, aquellos cuya compra/alquiler se haya realizado exitosamente, así como información estadística geográfica de anuncios, ordenados por tipo acción y otros parámetros.

Puede consultar contadores del portal, como el número de búsquedas realizadas por usuarios (incluyendo invitados). Además tiene la potestad de eliminar un anuncio que acumule demasiadas denuncias.

No tendrá acceso a búsquedas de inmuebles, ya que su función es de seguimiento de operaciones.

2.3.4. Diagrama de casos de uso

En el diagrama se muestran las relaciones entre los actores y sus casos de uso, que pueden ser

- de inclusión, aquellos necesarios para que se de la funcionalidad referida
- de extensión , aquello opcionales que amplían la funcionalidad referida

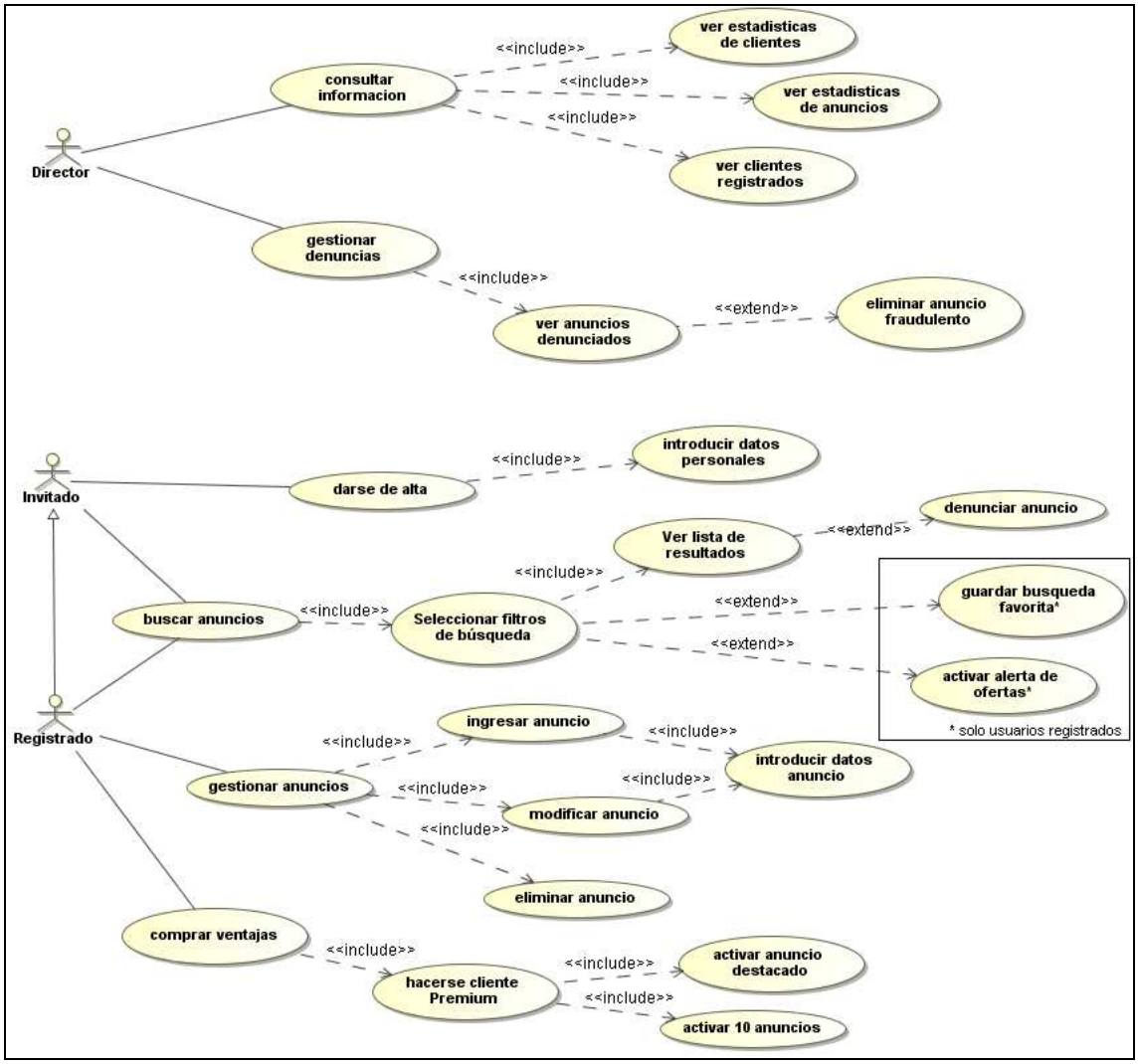


Ilustración 9. Diagrama de casos de uso para el portal Inmuebles.

2.4. Diseño lógico de la base de datos.

A partir del diagrama de clases podemos deducir, mediante una serie de transformaciones estudiadas en la asignatura Ingeniería del Software, un modelo lógico relacional de la base de datos.

Cada clase se transforma en una entidad y cada atributo en un campo de la base de datos. Las claves primarias del nuevo diseño corresponderán con los atributos código que representa de forma unívoca cada dato.

Las asociaciones del tipo "1 a varios" se tratan añadiendo a la tabla que representa la clase del lado "varios" los atributos que forman la clave primaria de la tabla que representa la clase del lado "1". Quedan marcados como claves foráneas. Toda clave foránea obliga a que el atributo sea válido, o sea, que la otra tabla contenga el nuevo valor.

2.4.1. Diagrama ER

Basándonos en el análisis realizado en el apartado anterior obtenemos el diagrama ER de la figura, del cual podemos escribir una descripción textual de cada tabla y relaciones.

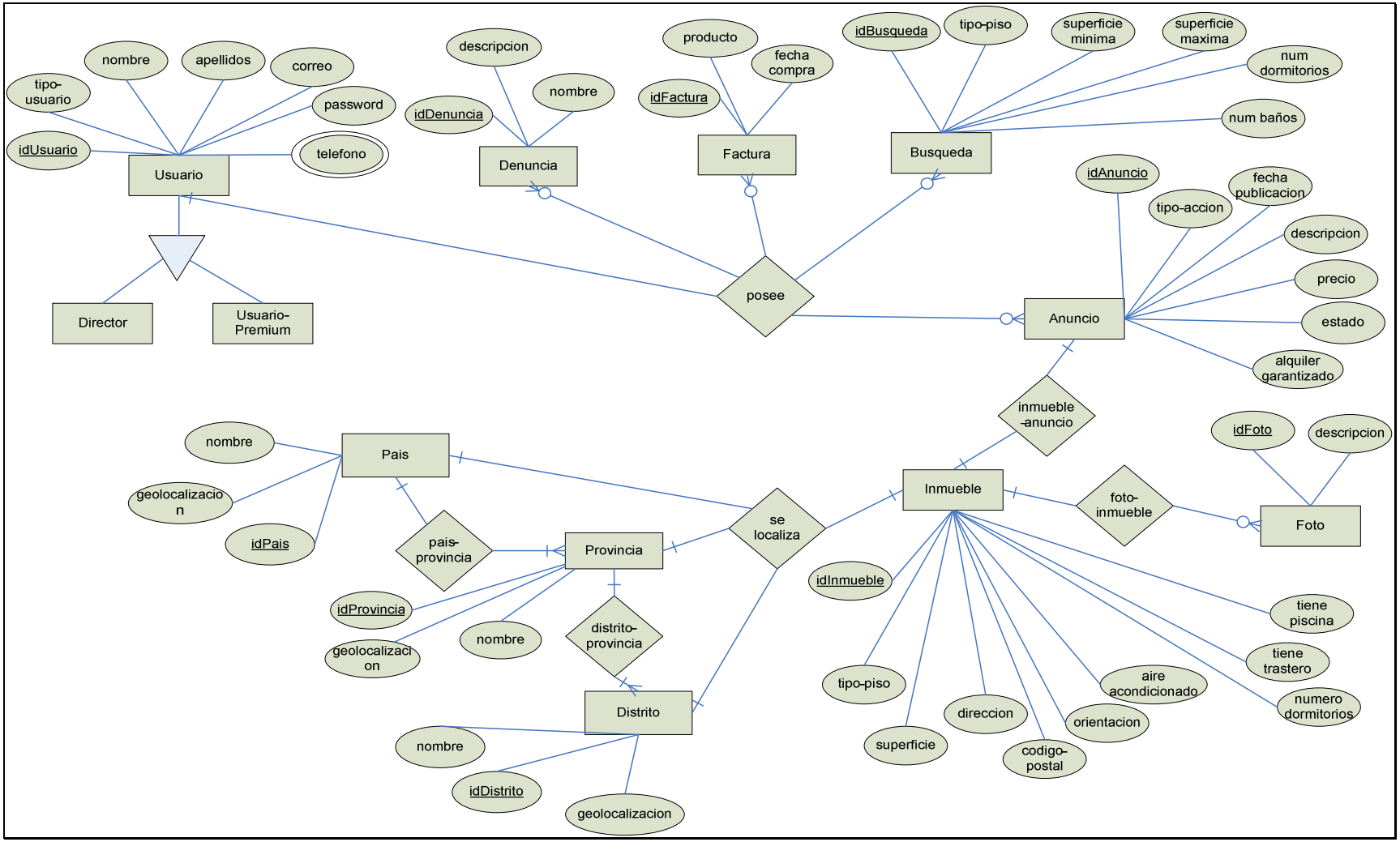


Ilustración 10. Diagrama ER de la base de datos. Se lee de izquierda a derecha y de arriba a abajo.

A continuación se presentan las tablas mas representativas en negrita, las claves primarias subrayadas y las claves foráneas en cursiva.

Usuario (idUsuario, tipoUsuario, correo, nick, password, nombre, apellidos, telefono1, telefono2)

Pais (idPais, nombre, geolocalizacion)

Inmueble (idInmueble, tipoPiso, superficie, direccion, codigoPostal, orientación, geolocalizacion, numDormitorios, numBanios, ascensor, trastero, terraza, piscina, aireAcondicionado, aguaCalienteIndv, calefaccionIndv, exterior, distrito)
{distrito} es clave foránea a la tabla Distrito

Anuncio (idAnuncio, tipoAccion, fechaPublicacion, descripción, precio, estado, alquilerGarantizado, *inmueble*)
{inmueble} es clave foránea a la tabla Inmueble

Factura (idFactura, fechaCompra, periodoValidez, fechaCaducidad, nombreProducto, *usuario*)
{usuario} es clave foránea a la tabla Usuario

Búsqueda (idInmueble, tipoPiso, superficie, direccion, codigoPostal, orientación, geolocalizacion, numDormitorios, numBanios, ascensor, trastero, terraza, piscina, aireAcondicionado, aguaCalienteIndv, calefaccionIndv, exterior, usuario)
{usuario} es clave foránea a la tabla Usuarios

2.4.2. Enumeraciones y definiciones para las reglas de integridad

Se definen las siguientes enumeraciones necesarias para la definición de la base de datos:

- Un inmueble puede ser comprado ("C"), alquilado ("A") o compartido ("S"). Este dato se llama tipo de acción.
- Un usuario puede ser normal ("N"), director ("D") o Premium ("P").
- Cada anuncio puede referirse a Vivienda ("V"), Habitación ("H"), Oficina ("O"), Local ("L"), Garaje ("G") u Otros ("O"). Este dato se define como el "tipo de piso".

- El estado de un anuncio puede ser publicado ("P"), éxito ("E") o retirado ("R").
- El estado de una vivienda puede ser "buen estado" (B), "para reformar" (P) y "obra nueva" (N).

2.4.3. Reglas de negocio

Se deben cumplir una serie de reglas fundamentales para conservar la coherencia de los datos de las tablas. Estas reglas serán controladas por el sistema gestor de base de datos mediante los mecanismos de los que dispone: checks, diparadores etc.

- Solo hay un director. Los directores no pueden poner anuncios.
- Un inmueble es único y no puede tener dos anunciantes.
- No pueden haber dos inmuebles con la misma dirección.
- No pueden haber dos anuncios con la misma descripción.
- Un usuario normal solo puede poner 2 anuncios y guardar 3 búsquedas favoritas.
- Dos búsquedas favoritas del mismo usuario no pueden ser iguales.
- Un usuario Premium puede mantener hasta 5 anuncios y guardar 10 búsquedas favoritas.
- Vencido el plazo, los anuncios y búsquedas de un usuario Premium quedan "congelados" (se mantienen pero no pueden modificarse salvo para cancelar) hasta que se renueve mediante pago.

2.5. Diagramas de clases, secuencia y colaboración

Del análisis inicial realizado se presenta el siguiente diagrama con los atributos tipados, relaciones de cardinalidad, relaciones de herencia y asociaciones de las clases que se utilizarán en la lógica de negocio. Inicialmente surgen estas 12 clases, lo cual no significa que el diagrama sea definitivo y que no puedan hacerse modificaciones por funcionalidad no contemplada, ampliaciones o correcciones a lo largo de ciclo. En el desarrollo de software es habitual realizar cambios aunque en cada versión se pretende alcanzar el máximo número de objetivos posibles.

A partir de este diagrama, mediante una transformación estudiada en detalle en la asignatura Ingeniería del Software, se extraerá un modelo lógico de la base de datos que ofrecerá la persistencia de los datos en servidor.

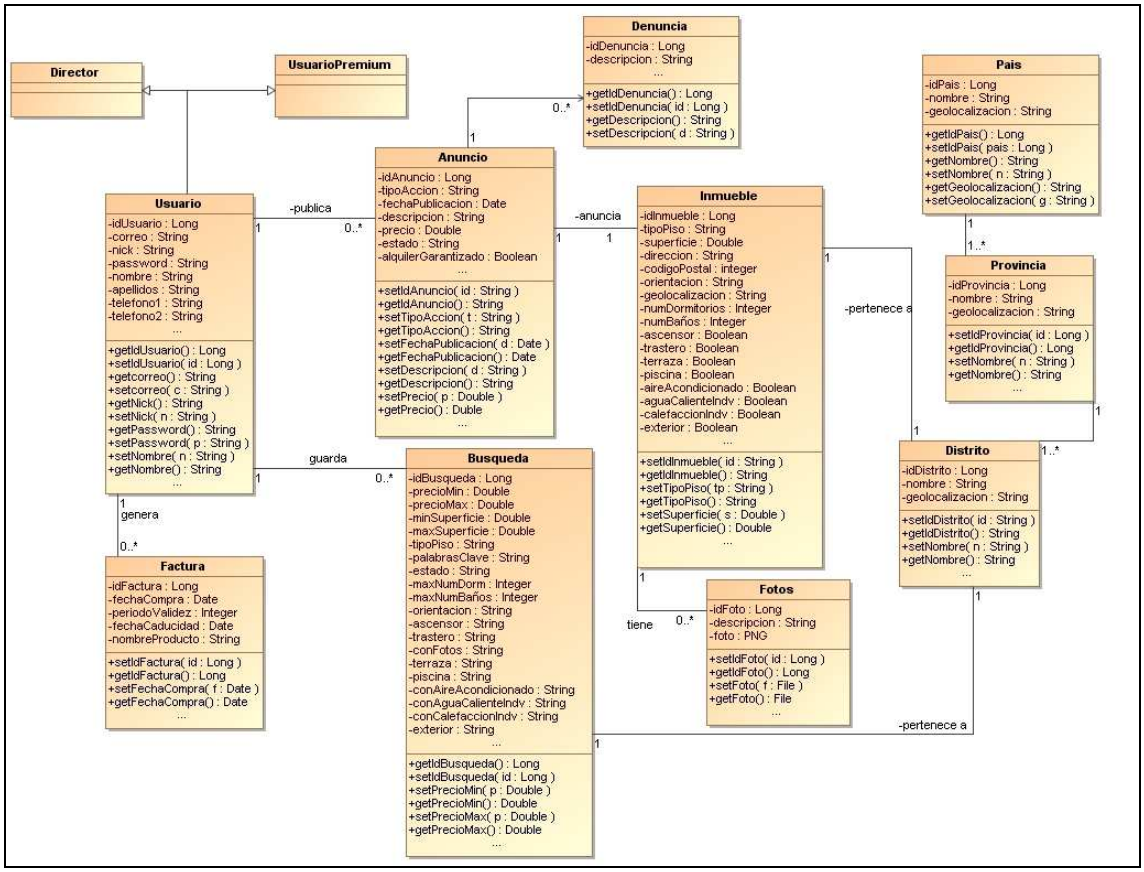


Ilustración 11. Diagrama de clases inicial basado en beans.

La mayoría de estas clases se transformarán en java-beans que serán cargados con datos por la capa DAO para su uso en las pantallas. Por ejemplo, las clases Pais-Provincia-Distrito se cargaran en java-beans tipo lista y se mostrarán en la interfaz gráfica para filtrar los anuncios inmobiliarios según desee el cliente. Estos datos seleccionados se guardarán en la sesión en forma de beans de contexto para realizar los filtrados en páginas posteriores.

Aunque es posible crear una única clase para la asociación Anuncio-Inmueble donde queden representados todos los atributos de un anuncio, se ha preferido separar en dos clases con vistas a conservar una tabla de base de datos con información estática de inmuebles y otra con información más dinámica (precio, fechas de publicación, etc) dedicada a la información manejada por anunciantes.

La tabla búsqueda se utiliza para guardar los filtros mas habituales que empleen los usuarios. Esta tabla tiene muchos atributos similares a inmueble pero con un significado distinto, puesto que admiten valores dobles y triples: pueden buscarse pisos exteriores, pisos no exteriores o bien que este dato sea indiferente. A esto se debe que los atributos de tipo booleano ahora sea tipo

string o tipo integer, para admitir la posibilidad de "indiferente" u otras posibilidades que sugiera el tipo de dato y que el usuario podrá seleccionar en las listas de selección del portal.

2.5.1. Diseño de las acciones.

Se mencionó en anteriores apartados que el diseño de las clases de este portal deben adaptarse al de Struts 2. Las acciones llaman a los componentes de negocio que a su vez se relacionan con la base de datos. Esto implica que las acciones son el hilo conductor de todo el proceso.

Una acción fundamental como la acción Login estará relacionada con la clase Usuario. Este portal solicita un mail y un password para realizar el login; el bean *LoginMailPwd* encapsula esta información que es introducida en la vista. Para completar el login es necesario acceder a la base de datos, buscando si el mail y el password introducidos existen y son correctos. De esto se encarga la clase ModeloUsuarioDAO y su método getUsuarioMail(), que se encarga de responder a esta consulta (Ilustración 12).

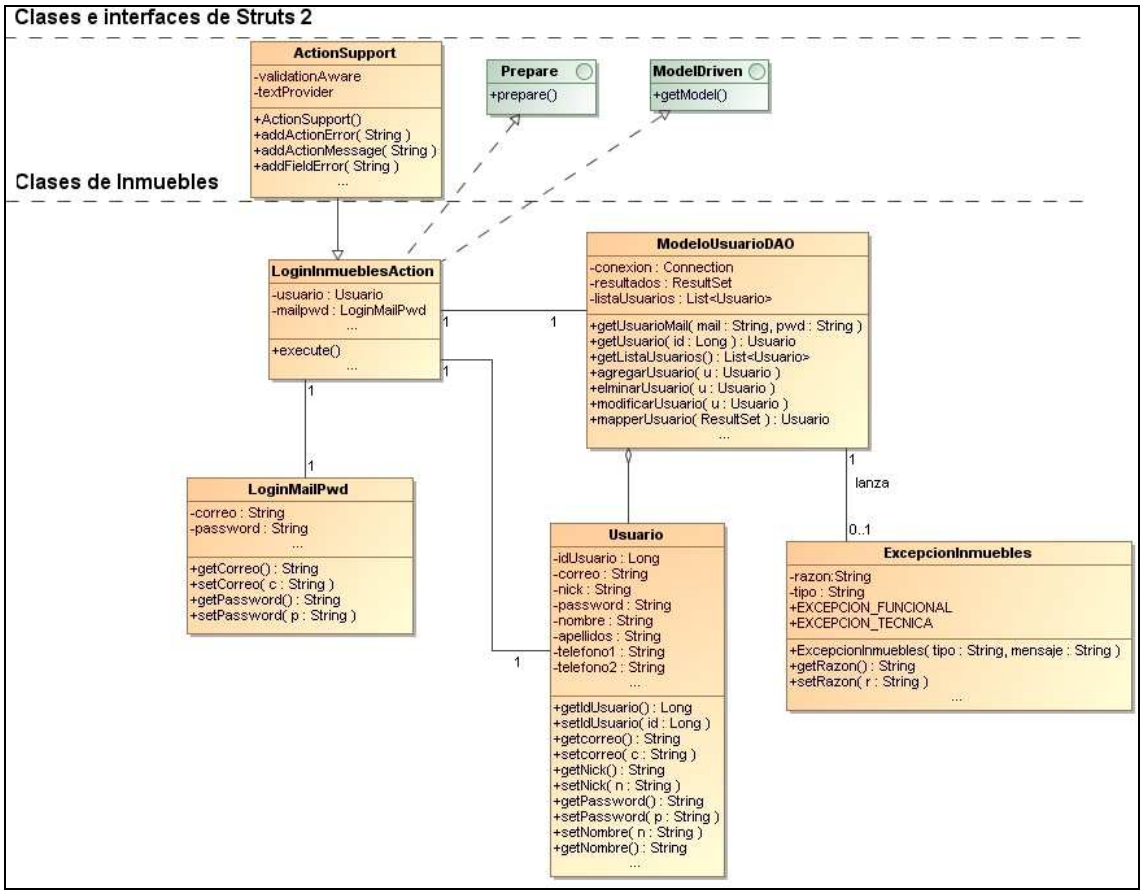


Ilustración 12. Diagrama de clases de la acción LoginInmueblesAction

Las acciones de este portal extienden a la clase de *Struts 2, ActionSupport*, que incluye métodos y campos de soporte para almacenar errores definidos por el usuario. El interface Prepare representa un interceptor asociado a una acción que se ejecuta antes de cualquier método de la acción. El interface ModelDriven relaciona cada elemento identificable de la página jsp llamante con la clase de acción de manera que podemos acceder a ellos con métodos get y set.

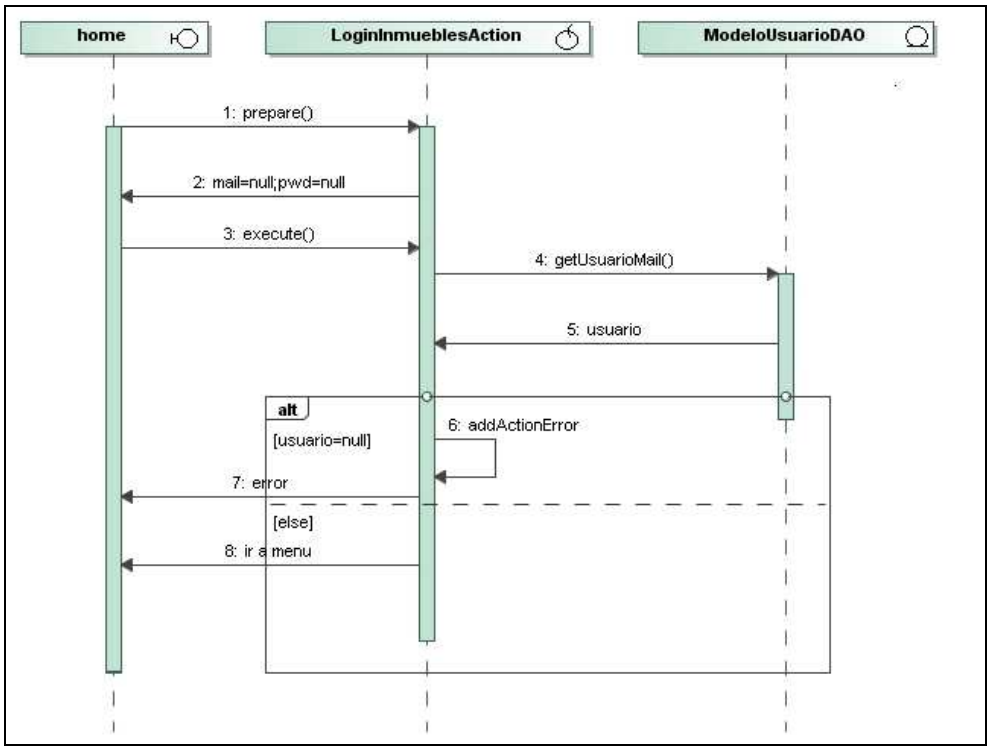


Ilustración 13. Diagrama de secuencia para el login.

Otro ejemplo algo más complejo de acción es el utilizado para el registro de un anuncio: RegistroAnuncioAccion. El interceptor Prepare se encarga de cargar los cuadros de selección para Pais y Distritos, necesarios para que el usuario seleccione este dato de forma controlada.

Una vez que el usuario envíe el formulario se recogen los datos de la página de servidor análogamente con el interfaz ModelDriven y en concreto con el método getModel(). Se ejecuta el método agregarAnuncio de ModeloAnuncioDAO que realiza la operación de inserción de los nuevos datos.

Si todo ha funcionado correctamente el control se devuelve a la acción y finalmente a la página del menu del usuario. En caso de que se produzcan excepciones, se volverá a mostrar la página con información de los errores para que el usuario corrija la entrada de datos.

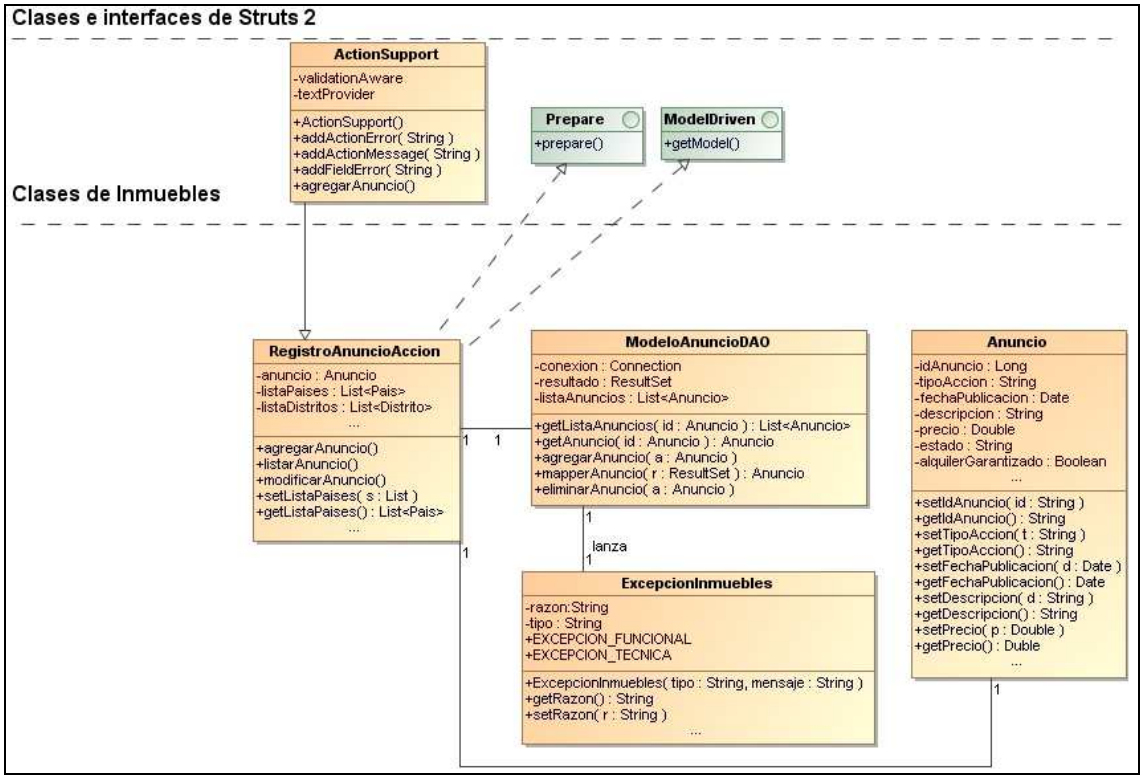


Ilustración 14. Diagrama de clases de la acción de registro de Anuncio.

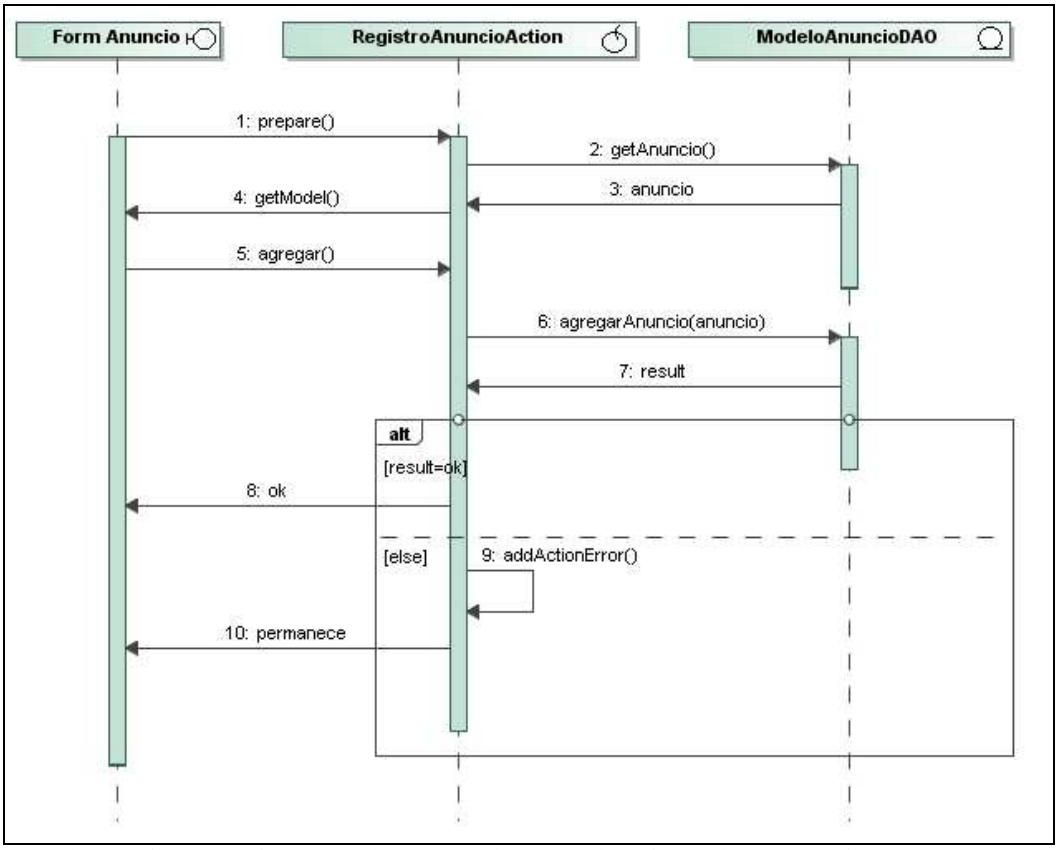


Ilustración 15. Diagrama de secuencia para el registro de un anucio.

Las excepciones son lanzadas por la clase ExcepcionInmuebles y capturadas por la clase action precedente. De esta manera, desde el controlador podemos desviar la ejecución hacia los formularios de entrada con advertencias de error.

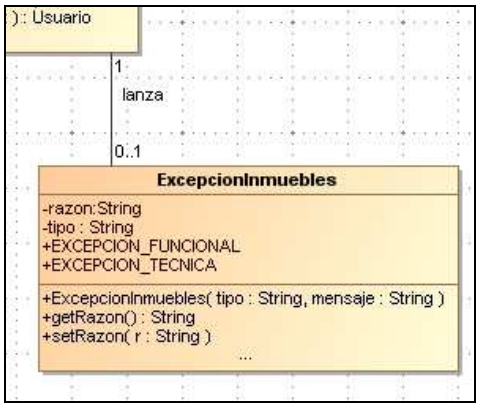


Ilustración 16. Clase excepción.

2.5.2. Diagramas de colaboración

Se presentan a continuación los diagramas de colaboración simplificados dedicados a los 3 actores que participan en este portal. Los elementos de estos diagramas se clasifican en clases frontera, de control y de entidad. Las **clases frontera** representan elementos de la vista, esto es, páginas de servidor con formularios de registro y botones que invocan acciones desde la parte de presentación. Las **clases control** son responsables de gestionar y dirigir las peticiones recibidas desde la vista o el control e invocar la lógica de negocio adecuada o presentación que corresponda en ese momento. Por último, las **clases entidad** encapsulan la persistencia y tienen acceso a la base de datos. Estas últimas son invocadas por los métodos de negocio.

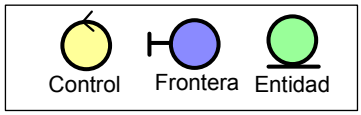


Ilustración 17. Tipos de clase en diagramas de colaboración

El itinerario del actor invitado es sencillo, en la página home selecciona "entrar como invitado" lo cual le da acceso a los filtros de inmuebles. A continuación puede navegar por los resultados según esos filtros y consultar la información concreta de un inmueble. La acción ListadoFiltrosAcc es la encargada de cargar la información necesaria de los inmuebles de la base de datos para mostrarla en la página de resultados en función de los filtros seleccionados.

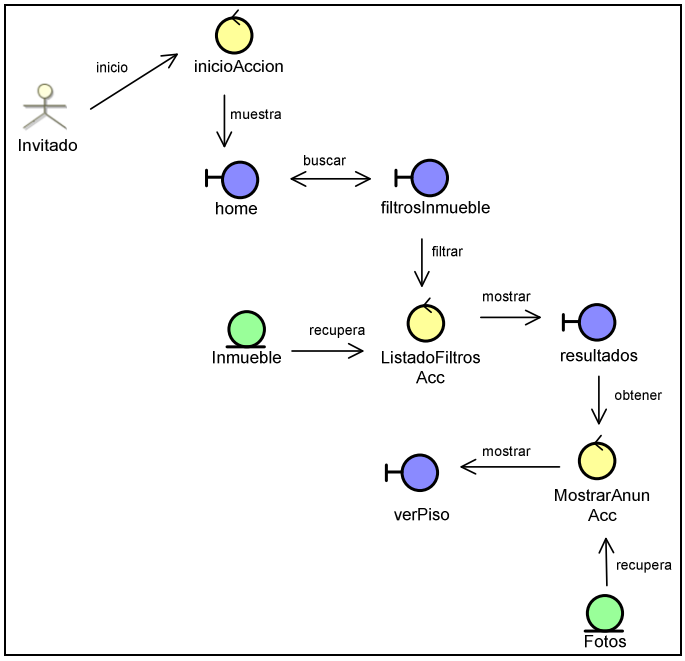


Ilustración 18. Diagrama de colaboración para el actor Invitado.

El director tiene acceso a las páginas de estadísticas y de denuncias, por lo tanto, una vez logado la acción EstadisticasAcc se encargará de lanzar las consultas de estadísticas a la base de datos y mostrarlas en la página verEstadistica.

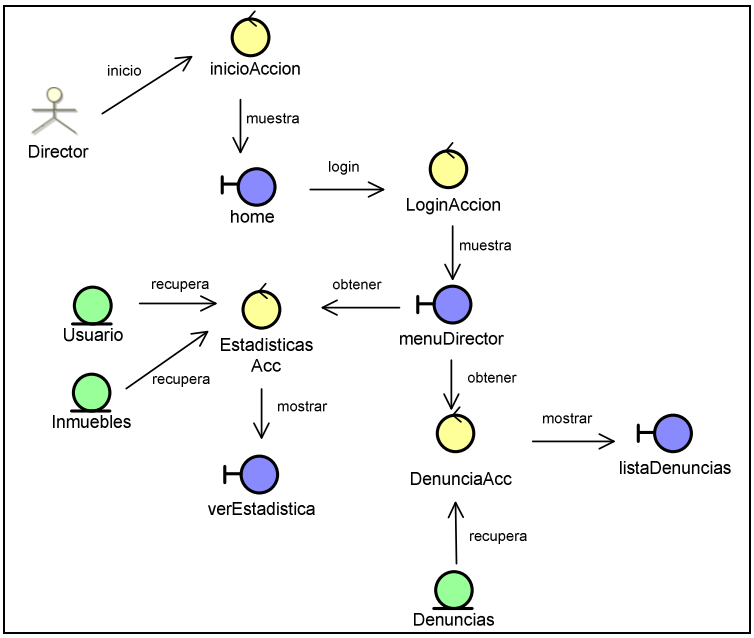


Ilustración 19. Diagrama de colaboración para la consulta del actor Director

Por ultimo, el usuario registrado puede realizar lo mismo que para el actor invitado y además administrar sus anuncios y datos.

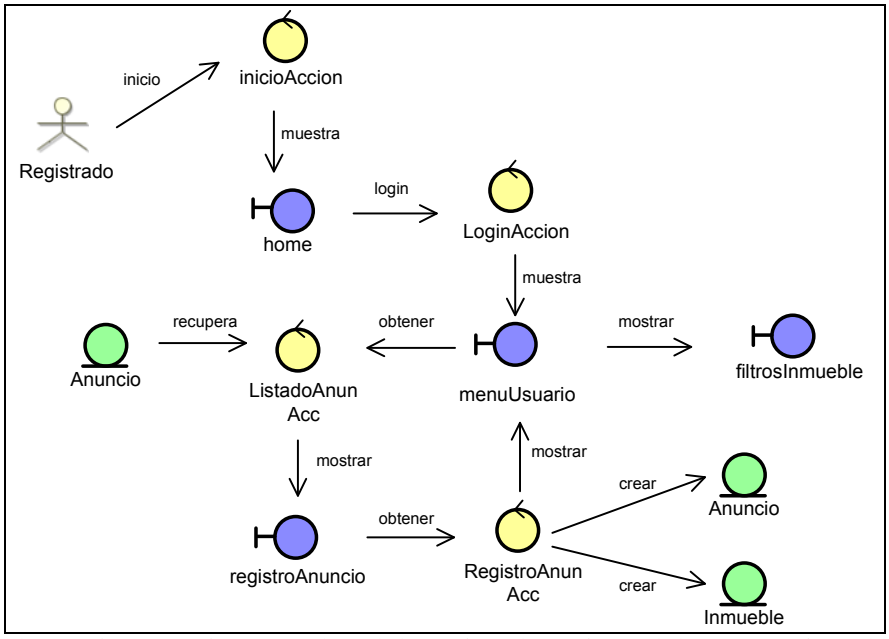


Ilustración 20. Diagrama de colaboración de registro del actor Registrado

3. Implementación.

En esta sección se detalla la plataforma y las tecnologías empleadas en el desarrollo del portal.

3.1. Estructura de la aplicación

La estructura de ficheros de la aplicación inmuebles tiene como base la estructura de una aplicación web. Todos los ficheros de recursos de la aplicación: html, jsp, png, txt, etc cuelgan del directorio raíz de la misma. Se trata de la parte pública de la aplicación.

Por otro lado, la carpeta WEB-INF contiene la parte privada de la aplicación, donde se sitúa el descriptor de despliegue, web.xml. Este fichero describe el contenedor web, sus elementos y el modo de conectar a los mismos. En él, se configuran aspectos de seguridad, ficheros de bienvenida, parámetros iniciales, etc. Un error en la estructura de este archivo imposibilitaría el despliegue de la aplicación.

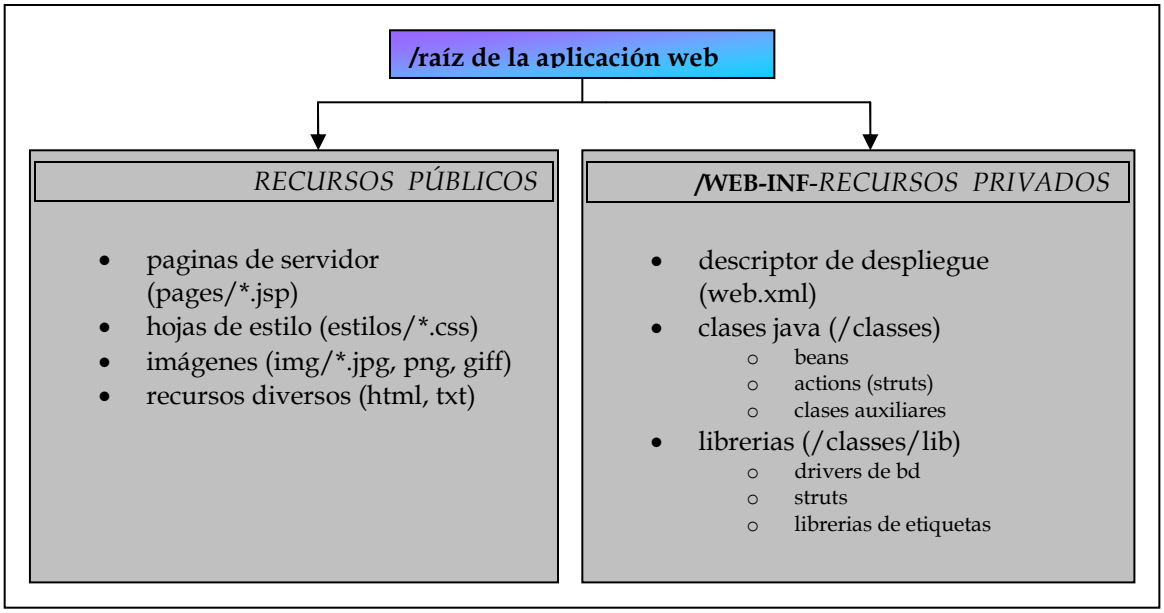
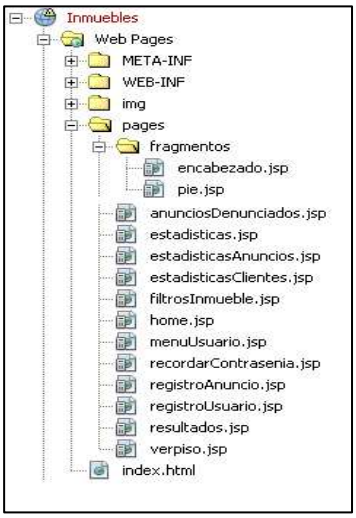


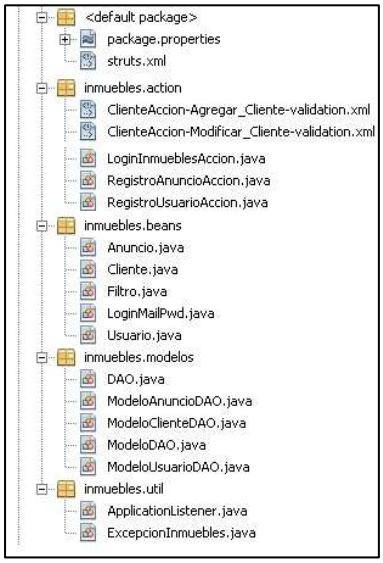
Ilustración 8. Estructura principal de una aplicación web.

3.1.1. Lista de archivos fuente y breve descripción

Centrándonos en los elementos que contienen código jsp consideramos las páginas de servidor que se describen a continuación:



- *home.jsp* - Página de inicio de la aplicación Inmuebles. Permite logarse o acceder como invitado o director.
- *registroUsuario.jsp* - Página de registro de datos personales y de contacto de usuarios.
- *menuUsuario2.jsp* - Página de operaciones de un usuario registrado.
- *filtrosInmueble.jsp* - Página que permite introducir filtros de búsqueda de inmuebles.
- *registroAnuncio.jsp* - Página que permite introducir datos de un nuevo anuncio de inmueble.
- *resultados.jsp* - Página que muestra resultados seleccionables de consulta de inmuebles.
- *verpiso.jsp* - Página que muestra información detallada de un inmueble.
- *estadisticas.jsp* - Página de menu para el director: permite seleccionar estadísticas.
- *estadisticasAnuncios.jsp* - Página que muestra información estadística de anuncios.
- *estadisticasClientes.jsp* - Página que muestra información estadística de clientes.
- *anunciosDenunciados.jsp* - Página que muestra una lista de los anuncios denunciados.



Por otro lado, tenemos el código java ordenado por paquetes:

- *inmuebles.action* – Acciones del controlador de Struts 2
  - *InicioAccion* – Acción encargada de redirigir a home al inicio de la aplicación.
  - *LoginInmueblesAccion* - Acción encargada de gestionar el login de usuarios al portal.

- *RegistroUsuarioAccion* - Acción que maneja la agregación y edición de los usuarios del portal.
- *RegistroAnuncioAccion* - Acción que maneja la agregación y edición de los anuncios de inmuebles.
- *ListadoAnuncioAccion* – Acción que maneja las listas de anuncios de un usuario.
- *ListadoFiltrosAnuncioAccion* – Acción que maneja los listados de anuncios filtrados.
- *LogoutAccion* – Acción que realiza la desconexión del usuario actual.

- *inmuebles.beans* – Beans java que encapsulan las entidades de negocio.
  - *Anuncio.java* - Bean Java que representa el anuncio de un inmueble.
  - *Usuario.java* - Bean Java que representa un usuario del portal.
  - *Filtro.java* - Bean Java que representa un filtro de búsqueda de inmuebles.
  - *LoginMailPwd* - Bean Java que representa un el correo electrónico y la password de un usuario. Sirve para mapear los campos de login.
  - *Alternativa*.java – Bean Java que guarda una respuesta del tipo Si/No/Irrelevante
  - *Pais*.java – Bean que representa la información de un pais.
  - *Provincia*.java – Bean que representa la información de una provincia.
  - *Distrito*.java – Bean que representa el distrito de una provincia.
  - *EstadoInmueble*– Bean que representa estados como "A reformar", "Perfecto estado"
  - *TipoAccion* – Bean que representa estados de "Comprar","Alquilar" etc.
  - *TipoInmueble* – Bean que representa los tipos de inmueble: "Vivienda", "Garaje", etc..
- *inmuebles.modelos* – Clases de acceso a base de datos ordenados por entidad de negocio.
  - *DAO.java* – Conexión genérica a base de datos.
  - *ModeloDAO.java* – Clase principal del patrón DAO, implementa la interfaz DAO.
  - *ModeloUsuarioDAO.java* – Operaciones de recuperación y actualización de base de datos para la entidad usuario.
  - *ModeloAnuncioDAO.java* - Operaciones de recuperación y actualización de base de datos para la entidad anuncio.
  - *ModeloFiltroDAO.java* - Operaciones de recuperación y actualización de base de datos para la entidad filtro.

- *ModeloPaisDAO.java* - Operaciones de recuperación y actualización de base de datos para la entidad pais.
- *EstadisticasDAO.java* - Operaciones de recuperación y actualización de base de datos para la entidad para estadisticas de director.

- *inmuebles.util*
  - *ApplicationListener* - Escuchador que aplica la conexión a la base de datos. Crea el pool de conexiones.
  - *ExcepcionInmuebles* – Clase que representa de forma genérica una excepción del portal Inmuebles.
  - *Singleton* – Clase que carga datos y enumeraciones que permanecen constantes a lo largo de la ejecución.
  - *LogInmuebles* – Singleton con el tratamiento de log de la aplicación.

Nota: Se han descrito los ficheros más representativos.

3.1.2. Documentación de la aplicación (código fuente)

El código se halla documentado con la convención javadoc que permite navegar por la descripción textual de todo el código fuente mediante la generación sencilla de un sitio web asociado. De esta manera se facilita la consulta de la documentación completa de la aplicación en un formato genérico como es html.

Se han documentado clases, métodos y atributos de todas las acciones, clases auxiliares, beans java, métodos específicos de las clases etc que forman parte de la aplicación. Su consulta hace posible una comprensión muy avanzada de la codificación del portal y será absolutamente necesaria para realizar ampliaciones, modificaciones y correcciones futuras.

El formato de la documentación es el siguiente

- Para clases java:
  1. Descripción textual del objetivo de esta clase
  2. Autor (etiqueta @author) de la clase.
  3. Versión java desde la que es operativa (etiqueta @since)
  4. Enlaces a clases relacionadas (etiqueta @see).

He aquí un ejemplo de documentación de una clase incluyendo los cuatro ítems anteriores resaltados.

```
/**
 * (1)Escuchador que aplica la conexión a la base de datos.
 * Es configurado en web.xml
 *
 * (2)@author Ricardo de los Rios
 * (3)@since 1.6
 * (4)@see GestionBaseDeDatos
 */
```

- Para métodos java:
    1. Descripción textual del objetivo de este método.
    2. Parámetros de entrada (etiqueta @param) que incluye descripción textual e identificador.
    3. Parámetros de retorno (etiqueta @return).
    4. Excepciones que lanza el método (etiqueta @throws)

```
/**
 * (1)Función de cierre del contexto de servlet.
 *
 * (2)@param servletContextEvent Evento de contexto del servlet
 * (3)@return servletContext Contexto del servlet
 * (4)@throws ioexcep Excepción de entrada/salida.
 */
```

Para generar la documentación basta situarse en la raíz de la aplicación y ejecutar mediante consola el siguiente comando:

3.2. Modelo físico de la base de datos

El siguiente diagrama representa el modelo físico de la base de datos de la aplicación. En él se incluyen:

- Para cada tabla las **claves** primaria (PK_ID..), secundaria/s (F) y única/s (UN_...).
- Los **tipos de datos** de todos los atributos (NUMBER, VARCHAR2, DATE etc)
- La **relación entre las tablas** asociando las claves y los campos relacionados.
- El **usuario** al que pertenece cada tabla (ADMIN).

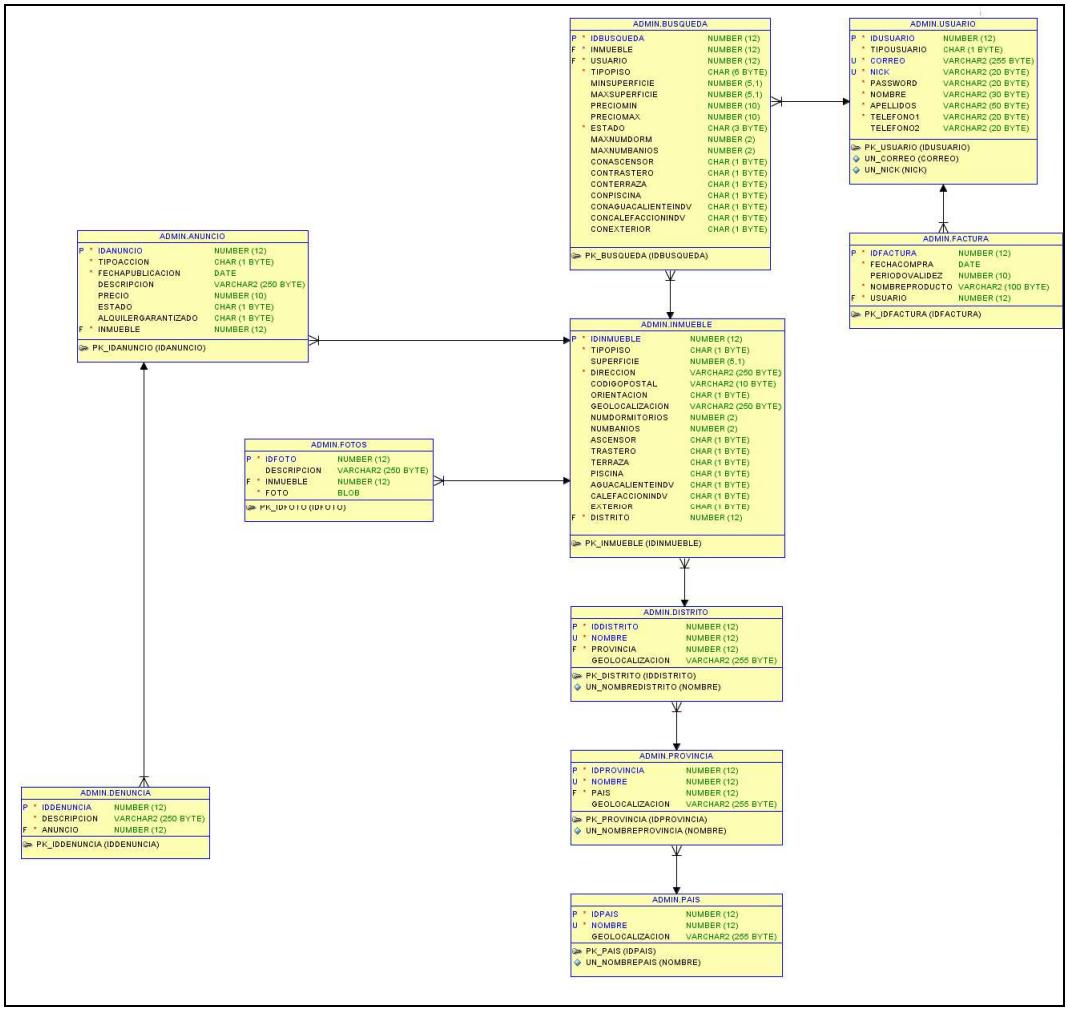


Ilustración 21. Modelo físico de la base de datos de la aplicación.

Se adjuntarán dos ficheros para la creación inicial de la base de datos:

- Definición de tablas y espacios de tabla (catalogo.sql)
    - Definición de campos y tipos
    - Definición de claves primarias y foráneas
    - Secuencias y disparadores para campos auto-incrementales (claves principales de las tablas)

3.3. Construcción de la aplicación

Al ser una aplicación basada en el patrón MVC (Modelo-Vista-Controlador) el código se encuentra desacoplado en consecuencia. La capa del controlador está constituida por las clases action que se encuentran todas ellas en el paquete java *inmuebles.action*. En el descriptor de despliegue de la aplicación (web.xml) se crean los filtros que llaman a estas acciones.

Para la implementación de la lógica de negocio se ha creado el paquete java *inmuebles.modelos* donde están todas las clases que realizan el acceso a datos y gestionan la información de negocio. Cada unidad de negocio tiene un javabean o agrupación de campos con un sentido lógico que se encuentran en el paquete inmuebles.beans.

La mayoría de las clases de la lógica de negocio pertenecen a un patrón DAO (de acceso a datos) que generaliza el acceso a la conexión de base de datos y el control de excepciones en la apertura y cierra de dicha conexión.

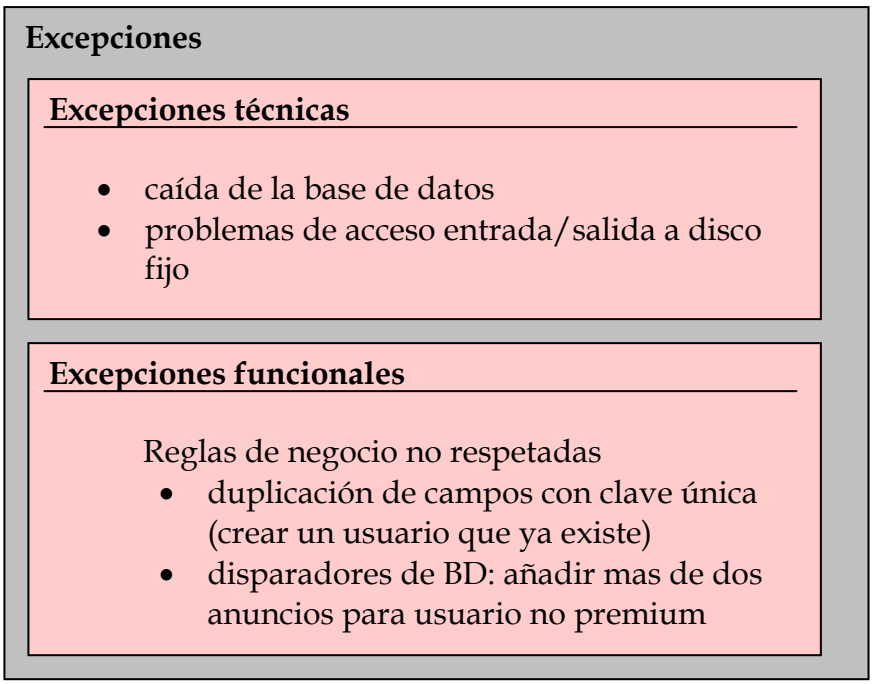
En el apartado de la Vista tenemos las páginas de servidor java (*.jsp) localizadas en la carpeta /pages colgando de la carpeta raíz de la aplicación. La relación entre las acciones del controlador y estas páginas están declaradas en el archivo struts.xml.

3.3.1. Manejo de las excepciones

Analizando el tipo de excepciones a controlar por un portal de este tipo se puede realizar una clasificación de las mismas en dos grandes categorías:

- las provocadas por **no cumplirse las reglas de integridad** de los datos una vez introducidos por el usuario

- las provocadas por **fallos técnicos** como la caída del servidor de base de datos, problemas de entrada/salida de ficheros en el servidor, problemas de pool de conexiones, etc.



Para el tratamiento de las excepciones se ha creado una clase que hereda de java.lang.Exception, *ExcepcionInmuebles* que posibilita clasificar y organizar la información asociada a la excepción una vez se produzca. Para cada clase de excepción se guarda su tipo (técnica o funcional), el mensaje asociado y la información adicional que requiera para el tratamiento posterior.

En cualquier momento se puede lanzar una excepción con el código

```
throw new ExcepcionInmuebles(ExcepcionInmuebles.TIPO_EXCEPCION,mensaje);
```

y ésta puede ser capturada en un clásico bloque try..catch.

Cuando se produce una excepción, se crean trazas en el archivo .log configurado para el API log4j. El nivel de log es configurable , lo que permite obtener monitorización con el nivel de detalle que se desee.

Tanto en el servidor de desarrollo como en el de producción, es indispensable disponer de estos archivos de trazas para realizar estudios del

funcionamiento del portal, especialmente, para estudiar casos de error que destapen un comportamiento incorrecto no controlado.

3.3.2. Pool de conexiones.

Un pool de conexiones centraliza la conexión a la base de datos, en este caso, a Oracle. Se trata de un grupo de conexiones reutilizables que es mantenido en el servidor de la aplicación, concretamente, asociado al contenedor servlet. Cuando una aplicación cierra una conexión, ésta vuelve al pool para quedar vacante.

Las conexiones agrupadas de esta forma reducen el tiempo de transacción en las conexiones a la base de datos compartiendo el objeto de acceso a dicha conexión y evitando crear una nueva conexión física cada vez.

La secuencia de pasos de conexión al pool es:

1. La aplicación **busca en JNDI** (Java Naming and Directory Interface) el nombre del recurso de DataSource JDBC asociado con la base de datos. El API de JNDI permite localizar dicho recurso, que habitualmente se declara en el fichero de configuración xml de la aplicación. Esta búsqueda se realiza mediante el método lookup().

   Cada recurso en JNDI tiene un único nombre y situación. Todos los recursos del contexto de la aplicación están por defecto bajo "java:comp/env". En el caso del recurso JDBC, se enontrarán en "java:comp/env/jdbc".

2. El recurso JDBC especifica qué pool de conexiones utilizar. El pool define los atributos de conexión con la URL, el nombre de usuario y la contraseña.

3. El servidor de la aplicación recupera la conexión física del pool que corresponde a la base de datos. Ahora, la aplicación puede leer, modificar y añadir información a la base de datos. Las aplicaciones acceden a la base de datos mediante el API de JDBC.

4. Una vez finalizado el acceso a la conexión se cierra.

3.3.3. Validación de entradas de formulario

La validación de datos es un pilar fundamental de la construcción de cualquier aplicación que desee mejorar en seguridad y solidez mediante reglas aplicadas a la entrada de datos del usuario. Es un apartado complejo y requiere de un diseño previo que abarque todas las posibilidades de error en la introducción de datos. Algunos de estos errores pueden ser:

- **Campos requeridos** que no son rellenados.
- Campos de tipo entero, fecha, email o url que no son introducidos correctamente o que no tienen el **formato adecuado**.
- Campos con **información incoherente**, por ejemplo, un salario máximo que es menor que un salario mínimo en el mismo formulario.

Struts 2 proporciona un método que facilita esta parte del desarrollo. Salvo casos especiales, la validación en *Struts* se puede hacer sin programación, de forma declarativa en ficheros xml. Hay dos grandes categorías de validaciones que son soportadas por el framework:

- Validadores de tipo campo. Están directamente asociados a los campos de formulario HTML y a unas reglas de validación que se declaran en un fichero xml.
- Validadores de tipo condicional, que realizan las clases para asegurar la que se respeta la funcionalidad de la aplicación. Por ejemplo, un usuario facilita un identificador o nick de nomenclatura correcta pero que ya está en uso en la base de datos. Desde la clase java que maneja la validación se puede lanzar un mensaje de error a visualizar en la página del usuario.

El método de construcción de validadores en Struts se divide en 3 etapas:

1. Definición de la acción (action) que requiere validación de entradas.
2. Creación de un archivo xml que especifica la validación. Debe tener la nomenclatura NombreClaseAccion-validacion.xml. En el caso en que se deseen validar solo métodos aislados de la acción, se utiliza la nomenclatura NombreClaseAccion-metodo-valdacion.xml.
3. Definición de la página a la que se debe navegar cuando ocurra un error. Esto se realiza en el fichero struts.xml.

Los ficheros xml de configuración de las validaciones tienen como etiqueta raíz <validators>, que contiene una o varias etiquetas <field> que se relacionan directamente con campos de formulario. Esta última etiqueta contiene dentro una o varias etiquetas <field-validator> con un atributo *type* que indica el tipo de validación a llevar a cabo.

Tipo de validación	**Atributo *type (field-validator)* de Struts**
Campo requerido.	required
Campo no es null ni vacío.	requiredString
Mínima y máxima longitud aceptada. Comprobación de no vacío.	stringlength
Es posible la conversión a entero. Definición de máximo y mínimo.	int
Requerida fecha.	date
Requerido correo electrónico.	e-mail
Requerida url.	url
Requerida expresión regular.	regex
Combinación de campos . (campo salariomax>campo salario mínimo)	fieldexpression
Se respeta el tipo de campo.	conversión
Vincular validaciones en formularios distintos.	visitor

Por ejemplo, en el siguiente código se declaran validadores para dos campos, identificador y contraseña, de manera que ambos son campos obligatorios y en particular el campo identificador debe contener entre 4 y 10 caracteres:

```
<!DOCTYPE validators PUBLIC
        "-//OpenSymphony Group//XWork Validator 1.0.2//EN"
         "http://www.opensymphony.com/xwork/xwork-validator-1.0.2.dtd">

        <validators>
          <field name="cliente.identificador">
                <field-validator type="requiredstring">
              <message>El campo identificador es obligatorio</message>
            </field-validator>
                <field-validator type="stringlength">
                        <param name="minLength">4</param>
                        <param name="maxLength">10</param>
              <message>El campo identificador debe tener entre 4 y 10
caracteres</message>
            </field-validator>
          </field>
          <field name="cliente.contrasena">
                <field-validator type="requiredstring">
              <message>El campo contraseña es obligatorio</message>
            </field-validator>
          </field>
        </validators>
```

4. Conclusiones.

La realización de un proyecto serio que sintetice y aplique los conocimientos adquiridos a lo largo de esta carrera ha supuesto un importante reto para mí. Este reto, consistía en asumir el riesgo de elegir una tecnología compleja como es Java EE, dentro del universo de sistemas, herramientas y lenguajes existentes actualmente, y conseguir un resultado satisfactorio en un breve periodo de tiempo, con una calidad merecedora de un trabajo de final de carrera. Esta calidad es el resultado de un gran esfuerzo, que implica consulta y estudio de gran cantidad de bibliografía bajo cronómetro y de horas de pruebas para hacer que la implementación funcione.

El uso del framework de Struts 2, marco de trabajo de muy reciente aparición, constituía un desafío añadido que se ha manifestado en forma de problemas de compatibilidad y de madurez de la plataforma que me han restado una gran cantidad de tiempo. Por otro lado, es perfectamente comprensible que en el panorama tecnológico actual se den estos problemas. Los elementos de cualquiera de estas librerías están, así mismo, en continua evolución y un simple cambio de versión en uno de ellos, hace que toda la estructura deje de funcionar.

No hay la menor duda que la elaboración de este proyecto supone un gran paso adelante en mi carrera académica (y profesional), por la experiencia ganada y el sentir que he afrontado un nivel de exigencia alto.

Además, me ha dado la ocasión de demostrarme a mí mismo la capacidad para afrontar y participar en proyectos de esta complejidad y de descubrir mis propias limitaciones y seguir atacando el mito ya clásico en informática, que consiste en estimar como fácil, o planificar de manera demasiado optimista, lo que precisa de una gran cantidad de esfuerzo y tiempo.

5. Bibliografía.

Struts 2

Brown, D.; Davis, C. M.; Stanlick, S. *Struts 2 In Action*. Manning. Greenwich, 2008.

Lafosse Jérôme. *Struts 2, el framework de desarrollo de aplicaciones Java EE*. Ediciones ENI. Barcelona, 2010.

Roughley, I. *Starting Struts 2*. InfoQ series. United States of America, 2006.

Patrones de diseño

Gamma, E.; Helm, R.; Johnson, R.; Vlissides, J. *Patrones de diseño: elementos de software orientado a objetos reutilizable*. Pearson Educación S.A. Madrid, 2003.

Bases de datos

Silberschatz, A; Korth, H; Sudarshan, S. *Fundamentos de bases de datos. Cuarta edición*. McGraw Hill. Madrid, 2002.

Trabajos fin de carrera

Alexandre Vall Mainou, *Desarrollo para Internet con tecnología Java*. ETIG, 2004.

Álvaro Aguado Mariblanca. *UOCasión*. ETIG.

Diego Vilar Gisbert. *Sistema gestor documental de una oficina técnica.* ETIS, 2011.

José Lorenzo Rodríguez Currais. *Análisis, Diseño e implementación de un sistema B2E en una administración pública*. Ingeniería informática, 2008.

Jesús-Miguel Sáenz Morras. *Tienda virtual*. Ingeniería Técnica de Informática de Sistemas, 2005.

Cesar Ruiz Gorrochategi. *Sistema de reservas para hoteles*. ETIS, 2004.

5.1. Webgrafía

Apache Tomcat – The Apache Software Foundation
http://tomcat.apache.org/

Oracle Spain site
http://www.oracle.com/es/index.html

Struts 2 site (*Apache Software Foundation*)
http://struts.apache.org/2.1.6/index.html

NetBeans IDE
http://netbeans.org/

ImagenesGratis
http://www.imagenes-gratis.net

5.2. Índice de ilustraciones

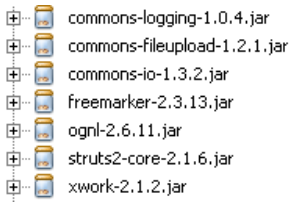
Anexo A. Instalación de la aplicación

El sistema operativo base sobre el que se han realizado pruebas satisfactorias de la aplicación es Windows XP/2000 o superior. Para realizar la instalación es necesario instalar en el sistema una versión Java 1.6 o superior, un servidor de aplicaciones Apache Tomcat versión 7 o superior y un gestor de base de datos Oracle versión 10g o superior. Estas herramientas son la plataforma tecnológica base de la aplicación. En caso de utilizar otra plataforma distinta no se garantiza la ejecución correcta del software.

Aun así, dada la portabilidad de java, sería posible desplegar esta aplicación con éxito en otras plataformas siempre que se respete el mínimo de versión y la compatibilidad de las herramientas.

Las ubicaciones en Internet de los instaladores de las tecnologías base estan indicadas en la tabla 1.

Tabla 1. Ubicación en Internet de las tecnologías base

Tecnología base a instalar	**Versión y ubicación en Internet***
Java	JDK 1.6 Url: http://www.oracle.com/technetwork/java/javase/downloads/jdk-6u25-download-346242.html Tamaño instalador: 76,7 Mb
Servidor de la BD	Oracle Express Edition Universal 10g** Url: http://www.oracle.com/technetwork/database/express-edition/downloads/102xewinsoft-090667.html Tamaño instalador: 207Mb
Servidor de aplicaciones	Apache Tomcat 7.0.11 Url: http://tomcat.apache.org/download-70.cgi Tamaño instalador: 7,7Mb
Asistente visual SQL de BD	SQL Developer 3.0.04 no jre Url: http://www.oracle.com/technetwork/developer-tools/sql-developer/downloads/index.html Tamaño instalador: 147Mb
Librería de Struts 2  	Es fundamental descargar la combinación de versiones indicada puesto que se realizaron pruebas con otras combinaciones (incluidas las oficiales) y se obtuvieron errores aun siendo el código correcto a principios de Mayo 2011. commons-logging-1.0.4.jar Url - http://repo1.maven.org/maven2/commons-logging/commons-logging/1.0.4/commons-logging-1.0.4.jar commons-fileupload-1.2.1.jar Url - http://repo1.maven.org/maven2/commons-fileupload/commons-fileupload/1.2.1/commons-fileupload-1.2.1.jar Nota: Extraer del interior del zip (/lib) commons-io-1.3.2.jar Url - http://www.java2s.com/Code/JarDownload/commons-io-1.3.2.jar.zip Nota: Extraer del interior del zip freemarker-2.3.13.jar Url - http://repo1.maven.org/maven2/org/freemarker/freemarker/2.3.13/freemarker-2.3.13.jar ognl-2.6.11.jar Url - http://mirrors.ibiblio.org/pub/mirrors/maven/opensymphony/jars/ognl-2.6.11.jar struts2-core-2.1.6.jar Url - http://www.java2s.com/Code/JarDownload/struts2-core-2.1.6.jar.zip Nota: Extraer del interior del zip xwork-2.1.2.jar Url - http://www.java2s.com/Code/JarDownload/xwork-2.1.2.jar.zip Nota: Extraer del interior del zip
Driver JDBC-Oracle	ojdbc6.jar Url:http://www.oracle.com/technetwork/database/enterprise-edition/jdbc-112010-090769.html Nota: Aceptar los términos de licencia y elegir el ojdbc6.jar
Log4J	log4j-1.2.16.jar Url: http://apache.rediris.es//logging/log4j/1.2.16/apache-log4j-1.2.16.zip
JFreeChart	JFreeChart-1.0.13 http://sourceforge.net/projects/jfreechart/files/1.%20JFreeChart/1.0.13/jfreechart-1.0.13.zip/download

*Todas las url de los instaladores son de Mayo/2011.

** Se trata de un servidor Oracle con restricciones, por ejemplo, no incluye un asistente de configuración de bases de datos. Utilizamos SQL Developer para facilitar la instalación de la base de datos.

A.1. Manual de instalación

A continuación se detallan los pasos a seguir para una instalación satisfactoria de la aplicación, desde la instalación de las herramientas de soporte básicas (Java, servidor de aplicaciones y gestor de base de datos) hasta la configuración de la aplicación. Así mismo se incluyen diversos juegos de pruebas para probar y observar el funcionamiento "en vivo" del portal.

producto/

Inmuebles.war – Fichero con la aplicación web para ser desplegada en servidor.

Inmuebles.xml – Fichero con la configuración inicial de la aplicación.

bd/

catalogo.sql – Script de **creación** de tablas, relaciones, secuencias y disparadores de la base de datos.

drop.sql – Script de **borrado** de las tablas, relaciones, secuencias y disparadores de la base de datos.

Insert1.sql – Script para **poblar** las tablas Pais, Provincia y Distrito.

Insert2.sql – **Juego de datos** con diversos usuarios, anuncios e inmuebles asociados.

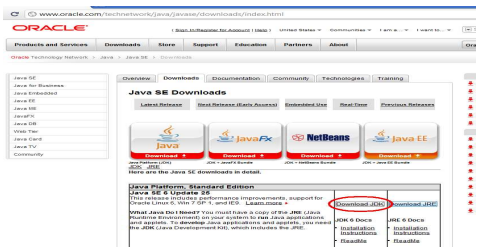
Es necesario realizar todos los pasos en el orden establecido, en caso contrario no se garantiza una ejecución satisfactoria del software.

Pasos previos

1. Se recomienda desinstalar cualquier versión de Oracle y Apache Tomcat que pudiera haber en la máquina donde se despliegue la aplicación así como reconfigurar cualquier aplicación que pueda interferir (por ejemplo, firewalls y otros servicios de bases de datos).

Instalación de Java.

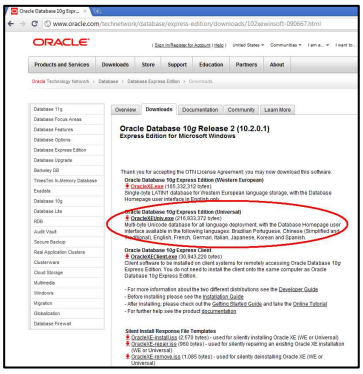
2. Descargar y ejecutar el paquete de instalación Java Development Kit 1.6 (ubicación) sobre Windows XP/2000/Vista o W7.



3. Seleccionar “Instalar”.

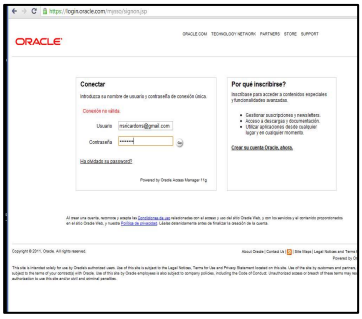
Instalación del servidor Oracle y del asistente visual SQL.

4. Descargar y ejecutar el paquete de instalación Oracle Database 10g Express Edition Universal (ubicación)



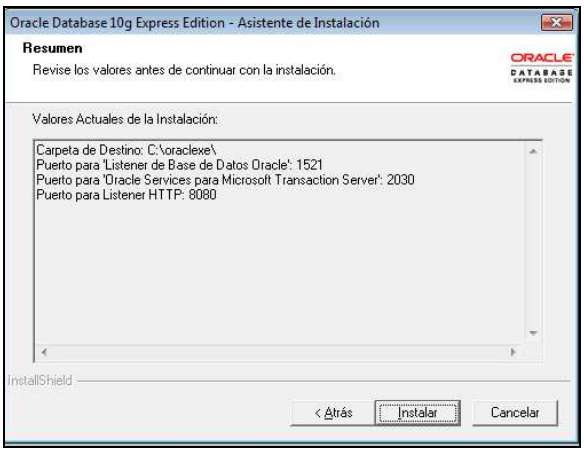
5. Es preciso estar registrado (gratuito) con una cuenta Oracle para descargar el archivo. Es un proceso que lleva unos 5-10 minutos y es necesario un correo electrónico.

6. Seleccionar “Siguiente”.

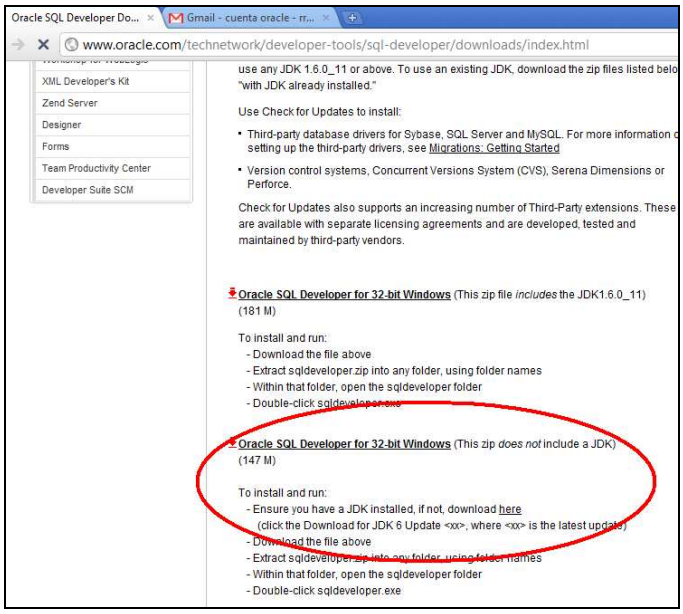


7. Aceptar el contrato de licencia.

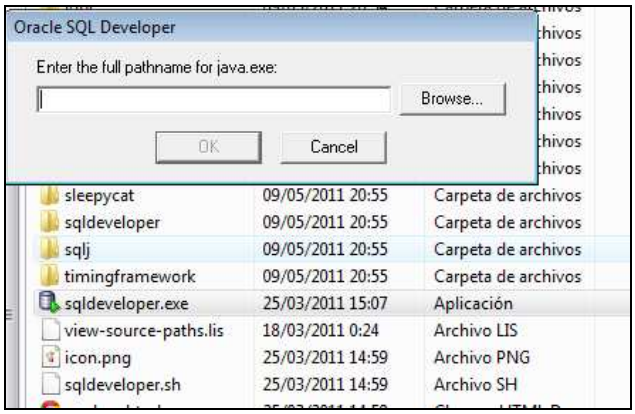
8. Durante el proceso de instalación, introducir un identificador/password para el usuario sistema. Por ejemplo, sys/sys.



9. Descargar el paquete de instalación SQL Developer 3.0.04 (ubicación)



10. Extraer el contenido del archivo .zip en c:\sqldeveloper (ubicación recomendada).
11. Lanzar la aplicación (ubicación recomendada C:\sqldeveloper\sqldeveloper.exe).
12. Seleccionar el directorio bin de la instalación de JDK Java (habitualmente c:\Archivos de programa\Java\jdk1.6.0_25\bin)



Acceso como usuario sistema a Servidor Oracle.

13. En la aplicación SQL Developer seleccionar Archivo->Nuevo->Conexión a base de datos. Seleccionar aceptar.
14. Escribir un nombre de conexión, por ejemplo, "con_sysdba".
15. Introducir el usuario y password del usuario sistema (por ejemplo sys/sys)
16. Elegir como Rol SYSDBA.
17. Para probar la conexión, pulsar el botón "Probar".

Creación del usuario administrador de la base de datos.

18. Ejecutar la siguiente sentencia en la hoja de trabajo SQL para crear el usuario administrador:

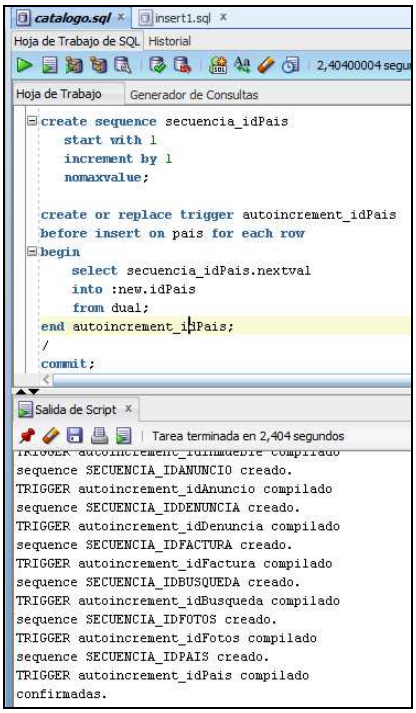
```
CREATE USER admin
IDENTIFIED BY admin
DEFAULT TABLESPACE users
QUOTA UNLIMITED ON users
TEMPORARY TABLESPACE temp;

GRANT CREATE SESSION,
   CREATE TABLE,
   CREATE VIEW,
   CREATE TRIGGER,
   CREATE SEQUENCE
TO admin;
```

19. Crear una nueva conexión para este usuario con rol "Valor por defecto". Llamarla por ejemplo "con_admin_inmuebles".

Ejecución del script de creación de las tablas

20. Seleccionar "Archivo->Abrir" y cargar el archivo "bd/catalogo.sql".
21. Con la conexión del usuario admin ejecutar el script (F5).



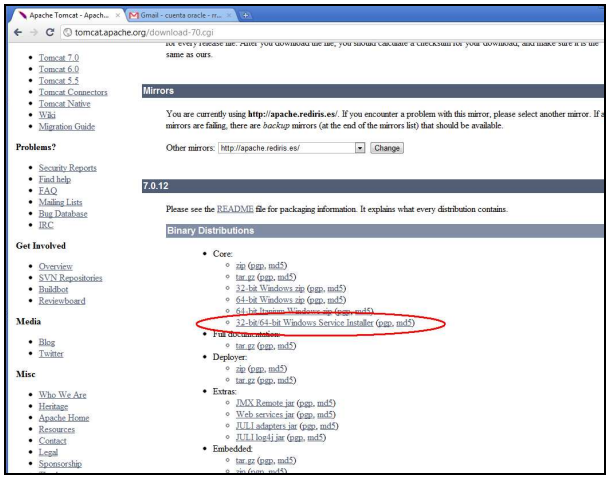
22. Con el mismo procedimiento, ejecutar los scripts *Insert1.sql* e *Insert2.sql* para realizar pruebas de la aplicación con una población de datos de ejemplo.

En cualquier momento se pueden borrar todas las tablas y datos con el script *drop.sql*.

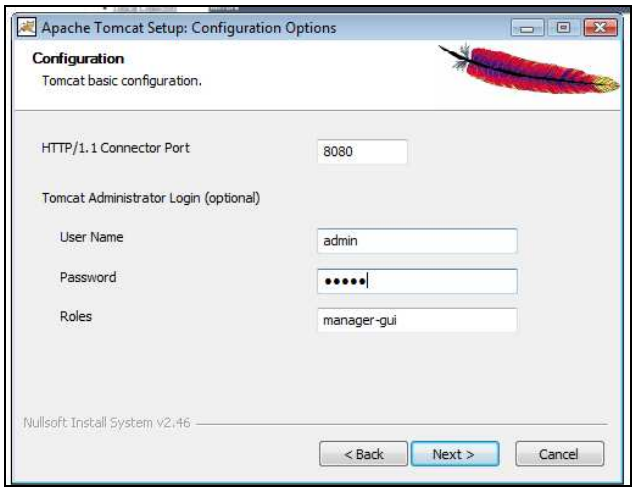
En este punto, la base de datos está lista para poder ser usada por la aplicación. Si los scripts de base de datos han fallado la aplicación no funcionará correctamente.

Instalación del servidor de aplicaciones Apache Tomcat.

23. Ejecutar el instalador de Apache Tomcat 7.0.12 (ubicación).



24. Escribir admin/admin como usuario/password para el login de administrador y mantener el puerto 8080:

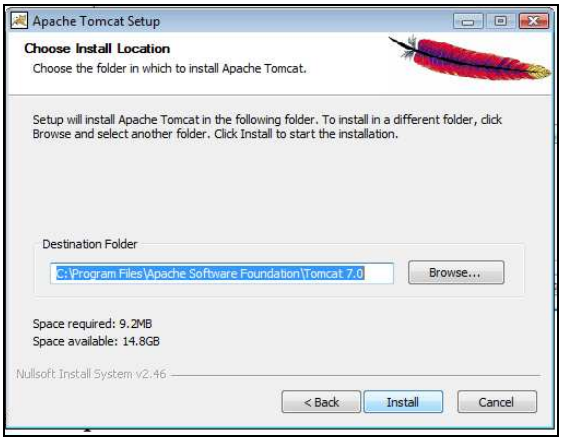


25. Elegir la ruta del Java Runtime, ruta recomendada:

C:\Program Files\Java\jdk1.6.0_25\jre

Nota: En el directorio jre\bin\ext copiaremos las librerías de Struts 2, el driver oracle y demás.

26. Elegir la ubicación de instalación de Tomcat:



27. Mantenemos seleccionado *"Run Apache Tomcat"* que ejecutará el servicio de autodespliegue de la aplicación.

28. Descargar las librerias de Struts 2, el driver JDBC de Oracle, el jar de Log4J y JFreeChart (Ubicaciones).

29. Copiar todos los .jar al directorio

C:\Program Files\Java\jdk1.6.0_25\jre\lib\ext

Despliegue de la aplicación en el servidor.

30. Ubicar el directorio

..\Apache Software Foundation\Tomcat 7.0\conf\Catalina\localhost

Habitualmente:

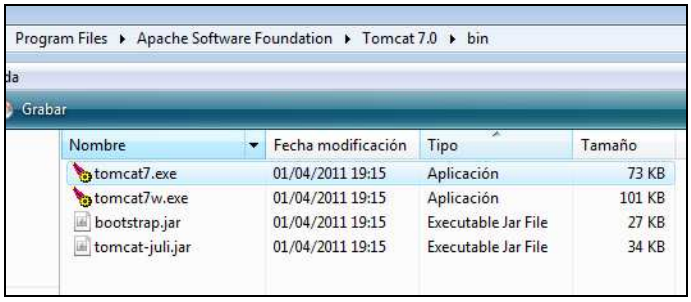
*C:\Program Files\Apache Software Foundation\Tomcat 7.0\conf**Catalina\localhost***

Si no existe, crearlo.

31. Copiar el archivo *Inmuebles.xml* a ese directorio
32. Copiar el archivo de la aplicación *Inmuebles.war* al directorio

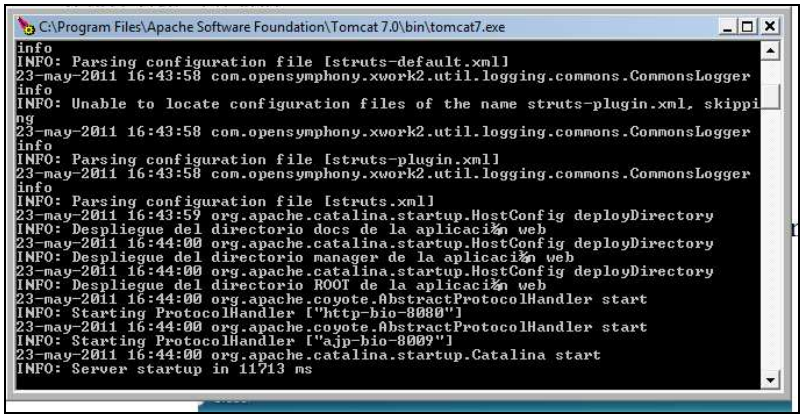
..\Apache Software Foundation\Tomcat 7.0\webapps

33. Tomcat desplegará automáticamente la aplicación. Ejecutamos *tomcat7.exe* del directorio *\bin*.



Nota: En caso de que la aplicación **no funcione en el servidor Tomcat**, empaquetar en el war las librerias externas y desplegar; de esta forma las librerías quedan desplegadas dentro de la instancia que crea el servidor. Es posible que Tomcat no gestione correctamente el orden de carga de las librerías si estas están en otras ubicaciones.

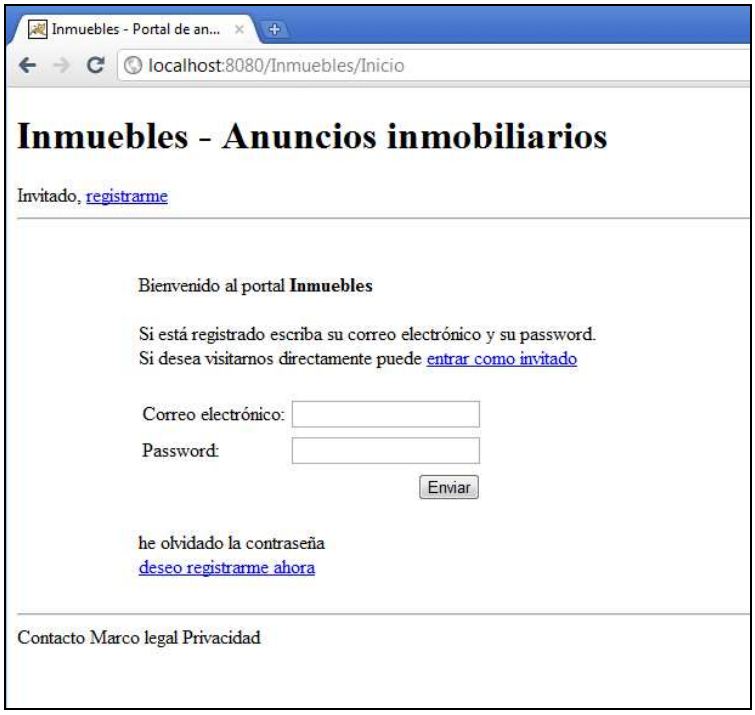
34. Aparecerá la consola del servidor. Esperar hasta que aparezca el mensaje *"Server startup in … ms"*



35. Abrir un explorador web e introducir la url

    http://localhost:8080/Inmuebles/

36. La aplicación debería iniciarse correctamente presentando la página inicial del portal:

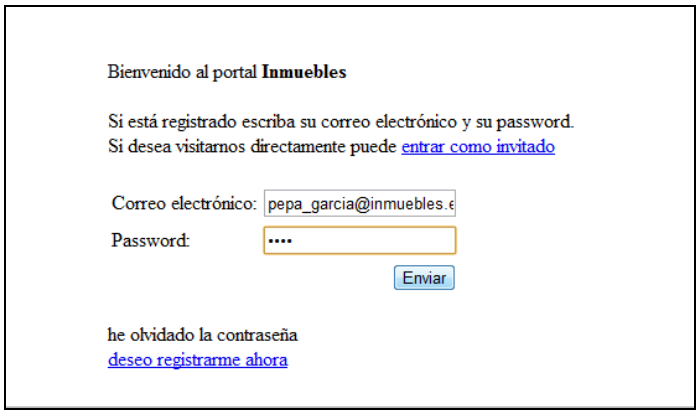


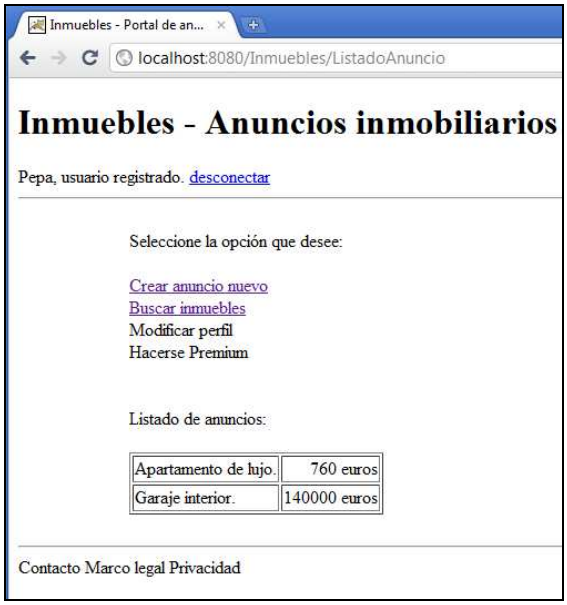
Pruebas navegando por el portal

37. Introdúzcase como correo electrónico y password:

Correo electrónico: pepa_garcia@inmuebles.es
Password: pepa

Nota: Considerando que se ejecutaron los scripts de inserción *Insert1.sql* e *Insert2.sql*.





Si no salen resultados o aparece el mensaje de error

"El correo introducido no existe en el portal"

Ejecutar el comando "*commit;*" en SQLDeveloper o bien cerrar esta aplicación, puede que no se haya ejecutado la última transacción y esté interfiriendo con la aplicación Inmuebles.

38. Puede crear anuncios nuevos con "*Crear anuncio nuevo*" o acceder a la página de filtrado de inmuebles.
39. Pulse "desconectar" para terminar sesión con ese usuario.
40. En el login introducir:

    ***Correo electrónico*: director@inmuebles.es**
    ***Password*: director**

41. Accederá al menú del director.

Anexo B. Especificación de los casos de uso

La especificación detallada de los casos de uso mas representativos mostrada a continuación refleja un primer detalle del itinerario que seguirá cada perfil que acceda a la aplicación, incluyendo información de los bloqueos y excepciones que puedan ocasionarse por problemas técnicos (caídas de red, problemas de acceso a base de datos) o funcionales (por ejemplo por datos inválidos o ausentes). También da una idea de cómo será la interfaz visual pues se hace referencia a botones de acción, campos de texto a rellenar, etc.

Se puede consultar el prototipo web adjunto a esta práctica para una representación visual de las páginas más representativas mencionadas en estos casos de uso.

Caso de uso:	consultar información <Director>
Descripción:	Tareas de seguimiento del portal, estadísticas y control de la información de los clientes.
Actores:	Director
Precondición:	Debe logarse como director
Postcondición	N/A
Inicio	Login de la página de inicio del portal.
Pantalla inicio	Home
Pantalla salida	Estadisticas
Proceso	
Paso	**Acción**
1	El usuario entre al portal
2	Se loga con el usuario/password de director
3	Puede seleccionar cualquiera de las acciones de la pantalla
Excepciones	
Paso	**Acción**
1	El usuario no es válido, debe solicitar contraseña.
2	Elige “he olvidado la contraseña”
3	Recibe correo con nueva contraseña en email
4	Vuelve a logarse

Caso de uso:	ver estadísticas de clientes <Director>
Descripción:	Consultar información de nuevos clientes y de la actividad de los mismos.
Actores:	Director
Precondición:	Debe logarse como director
Postcondición	N/A
Inicio	El usuario se loga en el portal.
Pantalla inicio	Estadisticas
Pantalla salida	Estadisticas
Proceso	
Paso	**Acción**
1	El director selecciona la opción estadisticas de clientes en la pantalla Estadísticas.
2	Consulta los resultados.
3	Pulsa el botón de volver si desea ver otras estadísticas
Excepciones	
Paso	**Acción**
1	Problemas de conexión a base de datos
2	Se muestra mensaje "Problema de conexión a base de datos"
3	Se pide al usuario volver a intentarlo

Caso de uso:	gestionar denuncias <Director>
Descripción:	Consultar las denuncias a anuncios realizadas por clientes e invitados.
Actores:	Director
Precondición:	Debe logarse como director
Postcondición	N/A
Inicio	El usuario se loga en el portal.
Pantalla inicio	Estadisticas
Pantalla salida	Estadisticas
Proceso	
Paso	**Acción**
1	El director selecciona la opción "consultar anuncios denunciados" en la pantalla Estadísticas.
2	Consulta los resultados y puede eliminar anuncios que considere fraudulentos con el botón eliminar anuncio.
3	Pulsa el botón de volver si desea ver otras estadísticas
Excepciones	
Paso	**Acción**
1	Problemas de conexión a base de datos

2	Se muestra mensaje "Problema de conexión a base de datos"
3	Se pide al usuario volver a intentarlo pasado un tiempo

Caso de uso:	darse de alta <Invitado>
Descripción:	Introducir datos de contacto y personales para registrarse en el portal.
Actores:	Invitado
Precondición:	Debe disponer de un correo electrónico.
Postcondición	N/A
Inicio	El usuario entra al portal y elige "deseo registrarme ahora"
Pantalla inicio	Home
Pantalla salida	Filtros de anuncios
Proceso	
Paso	**Acción**
1	El usuario introduce los datos personales y pulsa el botón siguiente.
2	El usuario introduce los datos de contacto y pulsa el siguiente
3	Se muestra mensaje de registro correcto.
Excepciones	
Paso	**Acción**
1	Se introduce algun dato de forma incorrecta (email, password y confirmación distintas, etc)
2	Aparece mensaje de aviso y se marcan los campos que son incorrectos.
3	El usuario corrige los datos incorrectos.

Caso de uso:	buscar anuncio <Invitado, Registrado>
Descripción:	Introducir filtros de selección de anuncios inmobiliarios y consultar los resultados.
Actores:	Invitado, Registrado
Precondición:	N/A
Postcondición	N/A
Inicio	El usuario entra al portal y elige "entrar como invitado" o bien se loga
Pantalla inicio	Home
Pantalla salida	VerPiso
Proceso	
Paso	**Acción**
1	El usuario selecciona los filtros de ubicación, precio,

	superficie y algunas características principales.
2	El usuario pulsa siguiente e introduce filtros relacionados con características mas específicas del piso.
3	Pulsa el botón de "ver resultados". Se muestra listado de resultados.
4	El usuario navega por los resultados y pulsa el botón ver.
5	El usuario elige la opción de ver contacto del anunciante.
Proceso alternativo	
Paso	**Acción**
1	El usuario introduce los filtros de selección.
2	No hay resultados, el usuario puede modificar los filtros.
3	Se pide al usuario volver a intentarlo pasado un tiempo

Caso de uso:	guardar búsqueda favorita <Registrado>
Descripción:	Guardar los filtros de una búsqueda para no tener que volverlos a introducir.
Actores:	Registrado
Precondición:	Debe logarse usuario registrado
Postcondición	N/A
Inicio	El usuario se loga en el portal.
Pantalla inicio	Home
Pantalla salida	VerResultado
Proceso	
Paso	**Acción**
1	El usuario se loga en el sitio.
2	Introduce los filtros principales y específicos.
3	En la pantalla del listado selecciona "guardar filtro".
Excepciones	
Paso	**Acción**
1	Problemas de conexión a base de datos
2	Se muestra mensaje "Problema de conexión a base de datos"
3	Se pide al usuario volver a intentarlo pasado un tiempo