# TFC-J2EE: assign2

## Memoria

**Argimiro Sánchez Aguilera**
ETIS

**José Juan Rodríguez**

17 de Junio de 2013

# Resumen

Como estudiante de Ingeniería Técnica de Sistemas en la UOC realizo este proyecto como trabajo final de carrera, intentando poner en práctica todos los conocimientos transversales adquiridos de las diferentes materias.

El área que me ha correspondido es J2EE o JEE, plataforma de programación empresarial basada en el lenguaje de programación Java y conocida como la más portable, compatible y escalable que existe actualmente en el mercado.

Compuesto por tres partes principales: el análisis, el diseño y la implementación, con este proyecto pretendo desarrollar una aplicación web que permita un intercambio de peticiones y/o mensajes entre diferentes usuarios de un mismo grupo.

El análisis implica determinar los actores y casos de uso para los que la aplicación debe dar soporte, deben especificarse de forma adecuada y sin ambigüedades.

Por otra parte el diseño pretende abstraer la realidad de los actores y casos de uso, trasladándolos a clases y objetos utilizables en programación orientada a objetos. También determina que objetos deben tener persistencia, mediante un modelo ER.

La parte fundamental de la implementación es la programación de la aplicación, en este caso en lenguaje Java. Siguiendo las directrices marcadas en el en el diseño debe culminar en un software que se satisfaga los requerimientos del análisis.

**Área del TFC:** Tecnología J2EE

**Palabras claves:** J2EE; JEE; Java; aplicación web; análisis; diseño; implementación;

# Índice

# 1. Introducción

Cuando trabajas como único informático en una pequeña / mediana empresa terminas siendo responsable de multitud de temas. Te esfuerzas por ser eficiente y puedes llegar a convertirte en una persona multitarea. Apuntas las cosas rápido sobre papelitos, para no olvidarte. Atiendes peticiones telefónicas lo mejor que puedes, mientras tu subconsciente está pensando en otro problema al que intenta buscar solución. Hay momentos en que te parece que lo tienes todo controlado y te sientes satisfecho, pero en algunas ocasiones pretender cumplir con todo rápidamente puede hacerte estresar.

Un estrés controlado puede ser motivante, pero un estrés prolongado puede hacer que te agobies y que tu eficacia se vaya reduciendo. Es un peligro que entres en ese estado, porque si el agobio permanece puedes llegar a tener bloqueos mentales que incurran en un círculo vicioso. Hay que hacer algo para salir de ese estado lo antes posible, ¡un descanso!, pues sí, pero y si como dice el dicho *más vale prevenir que curar*, intentamos proveernos de formas para no entrar en él.

## 1.1 Justificación del proyecto

Hace tiempo que le doy vueltas a como continuar mejorando y ser más organizado, por eso he escogido como temática de mi proyecto: hacer una herramienta que "atienda" por nosotros las peticiones que otros nos asignen, que permita un buen *feeling* entre las partes implicadas, y que contribuya a confirmar que las cosas se están haciendo bien. Así que he bautizado mi proyecto con el nombre de *assign2* (asignar a).

Pensado como un herramienta disponible en internet para todo el mundo, he intentado abstraer al máximo las dependencias y requerimientos, tanto conceptuales, para que sea útil en cualquier área, como de hardware, básicamente que sea adaptable a todo tipo de pantalla (ordenadores, móviles, tablets, etc.), simple de manejar y con un diseño de pantalla poco cargado que dé una sensación confortable y que aporte una buena experiencia de usuario.

Después de maximizar la abstracción, y conseguir romper con el rol de administrador / usuario, al que estamos tan enganchados los informáticos, mi visión es la siguiente:

Comienzas registrándote en el sistema como usuario. Un usuario adquiere el rol de peticionario cuando envía peticiones a otro y de responsable cuando las atiende. Para que un usuario pueda enviar peticiones a otro debe existir un vínculo entre ambos,

esto se lleva a cabo mediante la gestión de los grupos, lo cual permite establecer relaciones entre los diferentes usuarios.

Al crear una petición puedes asignarle una fecha de caducidad, en caso de no hacerlo la opción por defecto es sin fecha de caducidad. La petición será gestionada por el usuario responsable al añadirle acciones. A una petición también se le pueden añadir comentarios, tanto por parte del responsable como del peticionario.

Las peticiones pueden tener relaciones de uno a uno, o de uno a muchos, es decir enviadas de un usuario a otro usuario de forma exclusiva, o de un usuario a un grupo, o a varios grupos. A pesar de que las peticiones uno a uno son privadas, si ambos lo acuerdan pueden hacerlas públicas y son visibles para todo el/los grupo/s a los que una petición puede estar vinculada, permitiéndoles o no hacer comentarios al resto.

Una petición pasará por diferentes estados. Se inicializa en estado de pendiente y tras añadirle la primera acción, por parte del responsable, pasa al estado de en curso. Es el peticionario quien pondrá la petición en estado de resuelta cuando lo considere oportuno y entonces le asignará un nivel de satisfacción.

Existe la posibilidad de que el responsable no quiera tomar ninguna acción sobre una petición y la lleve directamente a estado de desestimada, en este caso el peticionario podrá asignarle un nivel de inconformidad. También podría darse el caso que pase tiempo y el responsable no se pronuncie, en este caso cuando el peticionario lo considere oportuno puede pasar la petición a estado de no escuchada.

La aplicación podrá enviar de forma automática emails de aviso, de recuerdo y de confirmación. En el cuerpo de un email puede existir un link con una serie de parámetros encriptados que faciliten la interacción instantánea con la aplicación.

Se facilitará la incorporación de nuevos idiomas de forma fácil, con simples ficheros de traducción de términos. También se determinarán el tipo de consultas, resúmenes y estadísticas, que los usuarios podrán realizar por pantalla y que podrán ser exportables a formato PDF y/o Excel.

## 1.2 Objetivos del TFC

El objetivo principal de la práctica es superar el reto de ser el único responsable del desarrollo por completo de una aplicación partiendo de cero. Comenzando con el papel de cliente, pues yo mismo me planteé el objetivo a resolver, siguiendo con los roles de analista, diseñador, programador y testeador, con toda la responsabilidad y matices que cada uno de los cometidos conlleva.

## 1.3 Enfoque y método seguido

He seguido las pautas marcadas en el *Plan Docente* de la asignatura y que se corresponden con el enfoque clásico del desarrollo de software: análisis, diseño e implementación. Para conseguirlo he procedido con una buena planificación que identifique bien cada uno esos pasos:

**Plan de trabajo**

- Determinar las fases del proyecto
- Preparar un calendario de trabajo
- Redacción del Plan de trabajo

**Análisis**

- Definición funcional
- Especificación detallada

**Diseño**

- Diagramas UML
- Definir las clases que forman la lógica de negocio
- Esbozo de la Interfaz Gráfica de Usuario
- Diseño conceptual y modelo ER de la DB

**Implementación**

- Elección de tecnologías y componentes
- Implementación iterativa
    - Iteración 1: Desarrollo escueto de un caso de uso
    - Iteración 2: Añadirle complejidad: validaciones, etc.
    - Iteración 3: Completarlo para su uso final
    - Iteración 4: Implementar el resto de casos de uso.
- Implementación de la base de datos
- Implementación de las clases DAO
- Implementación de las clases de la lógica de negocio

**Documentación**

- Realizar la memoria del proyecto
- Preparar el manual de usuario
- Elaborar una presentación en PowerPoint
- Guía de instalación del entorno de desarrollo

## 1.4 Planificación del proyecto

El tiempo es escaso y pasa muy rápido, por lo que sin una buena planificación el fracaso es más que probable. Así que a continuación estudio las fechas con las que debo cumplir, mi tiempo disponible y la forma de organizarlo.

### 1.4.1 Fechas comprometidas

Todo el proyecto está condicionado y comprometido a las siguientes fechas clave, cosa que he tenido en cuenta al confeccionar mi calendario, intentando asegurar que el trabajo esté siempre listo el día de antes.

| | Fecha de entrega |
|---|---|
| **PAC 1** Plan de trabajo | 11/03/2013 |
| **PAC 2** Análisis y Diseño | 15/04/2013 |
| **PAC 3** Implementación | 03/06/2013 |
| **PAC 4** Entrega final | 17/06/2013 |

### 1.4.2 Sesiones de trabajo

Mis obligaciones laborales no me permiten dedicar mucho tiempo entre semana, así que la parte fundamental de mi dedicación estará en los fines de semana. Dispongo de una semana de vacaciones en el mes de Abril que utilizaré para avanzar en la medida de lo posible.

| Día | Horas |
|---|---|
| Lunes | 0 |
| Martes | 1 |
| Miércoles | 0 |
| Jueves | 1 |
| Viernes | 1 |
| Sábado | 5 |
| Domingo | 5 |
| **Total** | **13** |

### 1.4.3 Calendario de trabajo

Después de introducir mi horario en Microsoft Projects y hacer el reparto de tiempos de cada fase veo que el tiempo de que dispongo para realizar la totalidad del proyecto son 207 horas.

| Nombre de tarea | Duración | Comienzo | Fin |
|---|---|---|---|
| **TFC-J2EE: assign2** | **207 horas** | **jue 28/02/13** | **dom 16/06/13** |
| **Plan de Trabajo ( PAC1 )** | **25 horas** | **jue 28/02/13** | **dom 10/03/13** |
| Redacción del Plan de Trabajo | 7 horas | | |
| Desglose de Tareas | 6 horas | | |
| Calendario de Trabajo | 6 horas | | |
| **Memoria del Proyecto** | **6 horas** | | |
| Diseño de la documentación | 3 horas | | |
| "Peleas" con Word y Projects | 3 horas | | |
| **Análisis y Diseño ( PAC2 )** | **65 horas** | **mar 12/03/13** | **dom 14/04/13** |
| **Análisis** | **20 horas** | **mar 12/03/13** | **sáb 23/03/13** |
| Definir funcionalidad principal | 5 horas | | |
| Especificación detallada | 15 horas | | |
| **Diseño** | **30 horas** | **dom 24/03/13** | **dom 07/04/13** |
| Diagramas UML | 10 horas | | |
| Definición Clases lógica de negocio | 10 horas | | |
| Esbozo de la Interfaz Gráfica de Usuario | 5 horas | | |
| Diseño conceptual y modelo ER de la DB | 5 horas | | |
| **Memoria del Proyecto** | **5 horas** | | |
| Avanzar en su desarrollo | 5 horas | | |
| **Preparaciones varias** | **10 horas** | **lun 08/04/13** | **dom 14/04/13** |
| Instalación entorno desarrollo | 5 horas | | |
| HowTos & HelloWorlds | 5 horas | | |
| **Implementación ( PAC3 )** | **91 horas** | **mar 16/04/13** | **dom 02/06/13** |
| **Implementación** | **80 horas** | **mar 16/04/13** | **vie 31/05/13** |
| Desarrollo de un Caso de Uso sencillo | 5 horas | | |
| Creación detallada de la BD | 5 horas | | |
| Diseño detallado de la GUI | 15 horas | | |
| Implementación de todos los Casos | 35 horas | | |
| Pruebas y Tests | 20 horas | | |
| **Memoria del Proyecto** | **11 horas** | | |
| Avanzar en su desarrollo | 11 horas | | |
| **Entrega Final ( PAC4 )** | **26 horas** | **mar 04/06/13** | **dom 16/06/13** |
| Revisión de la Memoria del Proyecto | 10 horas | | |
| Preparar Manual de usuario | 10 horas | | |
| Confeccionar presentación PowerPoint | 6 horas | | |

## 1.5 Productos obtenidos

Los productos son el resultado tangible de las diferentes etapas del proyecto. Así los principales han sido esta memoria y una aplicación software, pero también se han realizado otros complementarios:

- Memoria del proyecto
    - Presentación resumen en PowerPoint
- Aplicación en formato *servlet* que corre en un servidor de aplicaciones sobre la plataforma JEE.
    - Guía de instalación del entorno de desarrollo
    - Manual de usuario de la aplicación

## 1.6 Descripción de los capítulos siguientes

Los capítulos sucesivos desarrollan cada una de las etapas del proyecto que, a grandes rasgos, son las siguientes:

El capítulo de análisis se identifican los actores que manipulan la aplicación y sus casos de uso, realizando una descripción detalla de cada caso.

En el capítulo de diseño se establecen las clases, se presentan sus relaciones mediante diagramas UML y se determina que clases van a tener persistencia en la base de datos. Se muestra como ha sido diseñada la interfaz gráfica de usuario y se exponen los componentes que componen la arquitectura del sistema.

En el capítulo de implementación explico la configuración de mi entorno de desarrollo y destaco algunas formas de buenas prácticas.

Para finalizar se intenta dar una valoración económica al producto final y resumo algunas de las conclusiones más importantes a las que he llegado tras realizar la totalidad del proyecto.

# 2. Análisis

Comenzaré recordando que el proyecto sobre el que va a hacerse el análisis es básicamente un sistema de comunicación asíncrona y que como en todo sistema de comunicación no existe el concepto de borrado o modificación de un mensaje ya enviado, por lo que para retractarse o hacer aclaraciones es necesario transmitir un nuevo mensaje.

## 2.1 Actores

Por la naturaleza de la aplicación debes identificarte antes de entrar en el sistema, y para eso antes debes haberte registrado. Así pues el actor principal es el usuario registrado y no se considera usuario a nadie que no esté registrado.

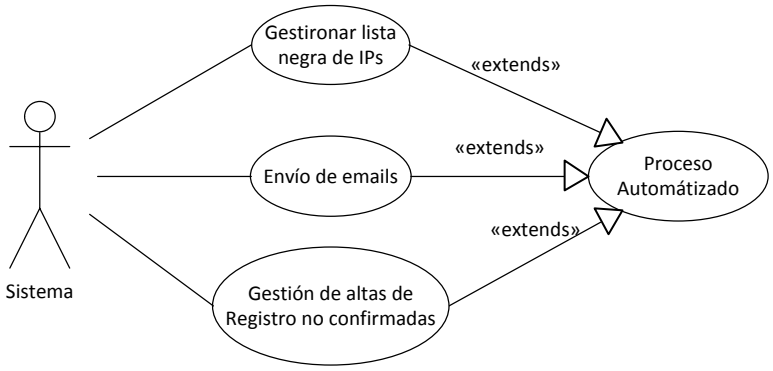
Todos los usuarios son iguales en posibilidades con respecto a los casos de uso disponibles en el sistema. Como en todo sistema de comunicación irán alternando los roles de emisor (Peticionario) y receptor (Responsable).

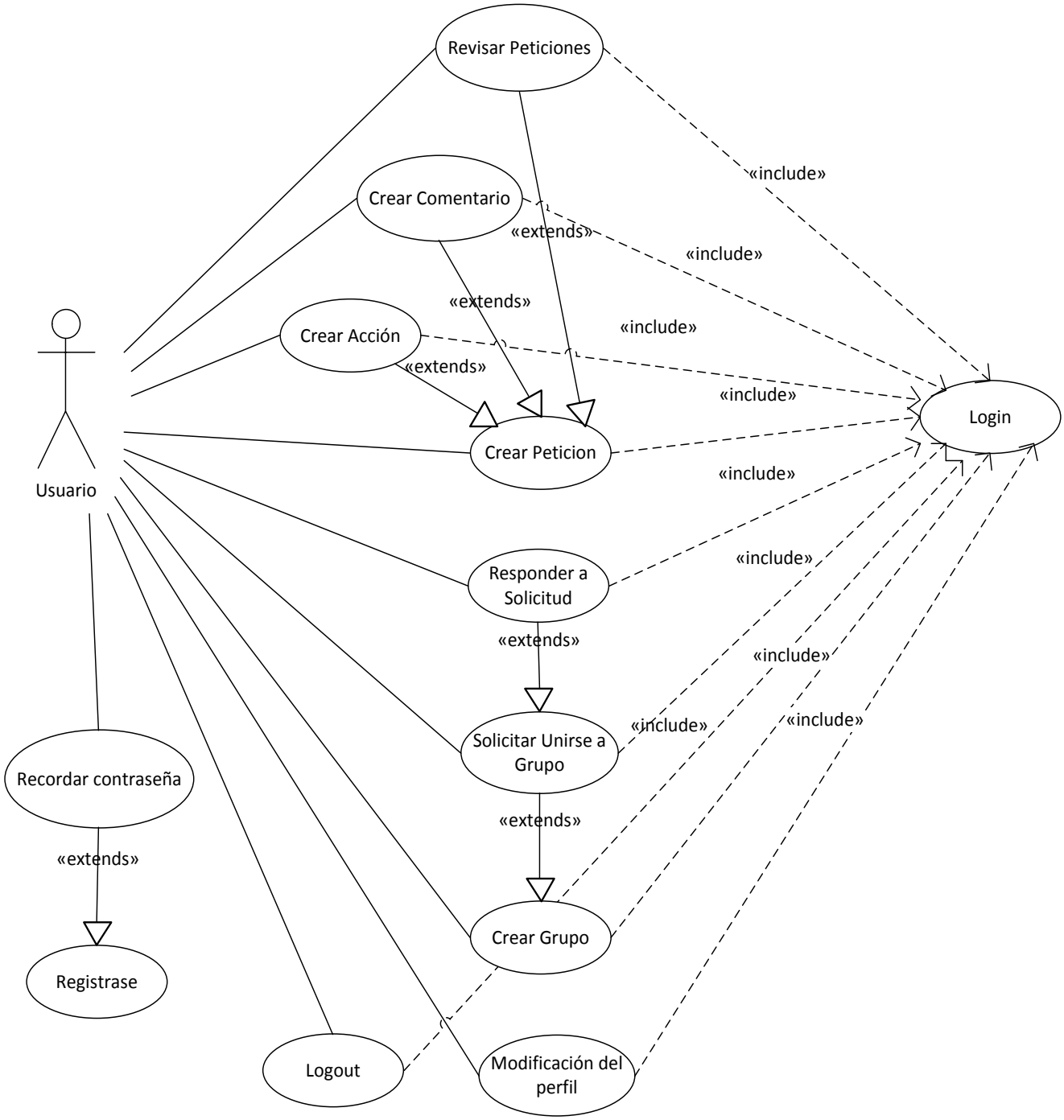
También creo adecuado considerar que el propio Sistema sea otro actor, pues tiene tareas de control, aviso y mantenimiento de formas automatizadas.

De momento no se ha definido un actor que actúe como administrador y por tanto no existen herramientas de consulta y mantenimiento global.

## 2.2 Diagrama de casos de uso

A continuación se muestran los gráficos UML de casos de uso de los Actores.






## 2.3 Descripción detallada de los casos de uso

En primer lugar detallaré el único caso de uso que no pertenece a un usuario, sino a un potencial usuario, que es el caso de uso Registrarse:

| Caso de uso: **Registrarse** | |
|---|---|
| *Objetivo:* Permite registrarse para posteriormente poder hacer Login en el sistema. | |
| *Actor* | Futuro usuario, persona NO registrada en el sistema. |
| *Precondición* | El usuario no existe en el sistema (el email proporcionado no debe existir en otra cuenta). |
| *Postcondición* | El usuario queda registrado en el sistema y puede hacer Login. |
| *Flujo de eventos* | |
| 1. Se abre un formulario.<br>2. Se rellenan correctamente los datos.<br>3. Se graban en el sistema.<br>4. Envío de email para confirmación de alta.<br>5. Pinchar en el link del email enviado y el sistema activa la cuenta. | |
| *Alternativas de proceso y excepciones* | |
| Debe recibirse la confirmación de alta a través del link del email en un periodo inferior a 24 horas, sino el sistema eliminará la petición de registro. De esta forma se confirma la validez del email proporcionado. Además en el formulario de registro existirá un captcha u operación matemática para descartar intentos de fuerza bruta por bots. | |

| Caso de uso: **Login** | |
|---|---|
| *Objetivo:* Permite entrar el en sistema | |
| *Actor* | Usuario. |
| *Precondición* | El usuario se registró y activó la cuenta. |
| *Postcondición* | El usuario accede al menú del sistema. |
| *Flujo de eventos* | |
| 1. Se solicitan los datos de acceso: email y contraseña.<br>2. El usuario tiene acceso al menú del sistema. | |
| *Alternativas de proceso y excepciones* | |
| El email introducido no existe: se avisa que es incorrecto. Si en un periodo inferior a 15 minutos se producen más de tres fallos, de este tipo, la IP de origen pasa a la lista negra durante 5 horas (estar en la lista negra representa no tener acceso a la pantalla de login desde la IP de origen).<br><br>El email existe pero la contraseña es incorrecta: se avisa la posibilidad de recordar la contraseña enviando un correo a la cuenta de email. Si en un periodo inferior a 5 minutos se producen más de tres fallos la IP de origen pasa a la lista negra durante 1 hora y se envía un correo informando de intento de acceso a la cuenta de email. | |

| Caso de uso: **Recordar contraseña** | |
|---|---|
| *Objetivo:* Permite recordar la contraseña enviandola a la cuenta de email. | |
| *Actor* | Usuario. |
| *Precondición* | El usuario se registró y activó la cuenta. |
| *Postcondición* | El usuario recibe los datos de acceso en su cuenta de email. |
| *Flujo de eventos* | |
| 1. Se introduce la cuenta de email.<br>2. El sistema envía los datos de acceso a esa cuenta.<br>3. Muestra un mensaje por pantalla. | |
| *Alternativas de proceso y excepciones* | |
| El email introducido no existe: se avisa que es incorrecto. Si en un periodo inferior a 15 minutos se producen más de tres fallos, de este tipo, la IP de origen pasa a la lista negra durante 5 horas. | |

| Caso de uso: **Logout** | |
|---|---|
| *Objetivo:* Permite desconectarse del sistema. | |
| *Actor* | Usuario. |
| *Precondición* | El usuario hizo login y está actualmente dentro del sistema. |
| *Postcondición* | El usuario queda en la pantalla de Login. |
| *Flujo de eventos* | |
| 1. El usuario pincha la opción de “salir”.<br>2. El sistema vuelve a posicionarse en la pantalla de Login. | |
| *Alternativas de proceso y excepciones* | |
| | |

| Caso de uso: **Crear Petición** | |
|---|---|
| *Objetivo:* Crear en el sistema una entrada de tipo Petición, de forma que sea informada al destinatario o destinatarios correspondientes. | |
| *Actor* | Usuario. En este caso el usuario de origen adquiere el rol de Peticionario |
| *Precondición* | Haber hecho Login. (En los siguientes casos de uso esta precondición se da por entendida.) |
| *Postcondición* | Una petición es creada e informada a él o los usuarios de destino. Estos cogen el rol de Responsables. |
| *Flujo de eventos* | |
| 1. Rellenar los datos de la Petición.<br>2. Establecer a quien, o quienes, debe ser enviada.<br>3. Confirmar el envío. | |
| *Alternativas de proceso y excepciones* | |
| | |

| Caso de uso: **Crear Acción** | |
|---|---|
| *Objetivo:* Crear una acción en respuesta a una petición. | |
| *Actor* | Usuario. |
| *Precondición* | Una petición fue creada por otro usuario y el Actor es destinatario y Responsable. La acción debe estar en estado: Pendiente o En Curso. |
| *Postcondición* | La acción es informada al Peticionario y es visible en el sistema por él y por todos los que tengan permiso de lectura . |
| *Flujo de eventos* | |
| 1. Tener activa la Petición correspondiente.<br>2. Escoger del menú la opción de Crear Acción.<br>3. Rellenar los datos del formulario.<br>4. Confirmar el envío. | |
| *Alternativas de proceso y excepciones* | |
| La petición no está en estado Pendiente o En Curso: en el menú la opción de Crear Acción debe quedar inhabilitada.<br>El destinatario no es Responsable: solo puede seguirla en modo lectura, o hacer comentarios si tiene habilitado el permiso para ello. | |

| Caso de uso: **Crear Comentario** | |
|---|---|
| *Objetivo:* Crear un comentario sobre una Petición o una Acción | |
| *Actor* | Usuario. |
| *Precondición* | El Actor debe ser responsable o en caso contrario tener permiso de lectura y de hacer comentarios. |
| *Postcondición* | El comentario es informado a usuario que creó la Petición o Acción, según corresponda. Además es visible por todos los que tengan permiso de lectura. |
| *Flujo de eventos* | |
| 1. Seleccionar una Petición o Acción sobre la que se desea realizar un comentario.<br>2. Rellenar el comentario.<br>3. Confirmar el envío. | |
| *Alternativas de proceso y excepciones* | |
| | |

| Caso de uso: **Revisar Peticiones** | |
|---|---|
| *Objetivo:* Es la pantalla principal que se muestra tras haber hecho Login y permite la navegación entre las peticiones. Existe un seleccionador de acceso rápido que filtra las peticiones en 4 clases: Recibidas y pendientes, Recibidas y finalizadas, Enviadas y pendientes, Enviadas y finalizadas. | |
| *Actor* | Usuario. |
| *Precondición* | Tener Peticiones en alguna de sus “bandejas” |
| *Postcondición* | |

| *Flujo de eventos* |
|---|
| 1. Acceso a la navegación de Peticiones. |
| *Alternativas de proceso y excepciones* |
| |

| Caso de uso: **Crear grupo** | |
|---|---|
| *Objetivo:* Crear un nuevo grupo, actúa como una lista de usuarios vinculados, de esta forma pueden clasificarse por ejemplo en: familia, amigos, empresa, etc. | |
| *Actor* | Usuario. |
| *Precondición* | El nombre de grupo no debe existir dentro de los grupos ya creados por este usuario. |
| *Postcondición* | Un nuevo grupo queda creado y listo para ser contenedor de usuarios vinculados. |
| *Flujo de eventos* | |
| 1. Acceder a la opción de Crear Grupo.<br>2. Indicar el nombre del nuevo grupo.<br>3. Aceptar. | |
| *Alternativas de proceso y excepciones* | |
| | |

| Caso de uso: **Crear solicitud de unirse a grupo** | |
|---|---|
| *Objetivo:* Solicitar a otro usuario que quieres crear un vínculo con el añadiéndolo a uno de tus grupos. La petición se realiza conociendo su dirección de email. | |
| *Actor* | Usuario. |
| *Precondición* | Debe existir el email al cual se solicita vincularse. |
| *Postcondición* | El usuario de destino es informado en su pantalla de solicitudes pendientes de resolver. (Adicionalmente también puede ser informado a su email). |
| *Flujo de eventos* | |
| 1. Estar dentro del grupo al que se desea vincular otro usuario.<br>2. Presionar la opción Crear solicitud.<br>3. Indicar el email.<br>4. Aceptar. | |
| *Alternativas de proceso y excepciones* | |
| El email no existe en el sistema: Se informa de que no existe ningún usuario con ese email.<br><br>Nada impide tener un usuario vinculado en más de un grupo, cuando esto se produce no se envía una solicitud de unión sino que solo se hace en el primer grupo al que se le vinculó. | |

| Caso de uso: **Responder a Solicitud** | |
|---|---|
| *Objetivo:* Responder a una solicitud de vinculación con otro usuario. Esto puede hacerse desde la pantalla de Solicitudes pendientes de resolver, o en el caso del que el origen activase la casilla de informar por email desde el link aceptar o rechazar. | |
| *Actor* | Usuario. |
| *Precondición* | Tener una solicitud de vinculación pendiente de resolver. |
| *Postcondición* | La solicitud queda resulta. |
| *Flujo de eventos* | |
| 1. Acceder a la pantalla de solicitudes pendientes de resolver.<br>2. Presionar el botón Aceptar o el de Rechazar la solicitud.<br>3. Se informa al origen con otro email de la resolución tomada en destino. | |
| *Alternativas de proceso y excepciones* | |
| Haber recibido un email de solicitud y responder desde los link de aceptar o rechazar. | |

| Caso de uso: **Modificación del perfil** | |
|---|---|
| *Objetivo:* Poder modificar datos en su ficha de usuario. | |
| *Actor* | Usuario. |
| *Precondición* | Estar registrado en el sistema |
| *Postcondición* | La ficha de usuario queda actualizada |
| *Flujo de eventos* | |
| 1. Puede alterar cualquier dato en su ficha.<br>2. Al aceptar las modificaciones estas quedarán actualizadas en el sistema. | |
| *Alternativas de proceso y excepciones* | |
| Si modifica el email, este debe ser validado por el sistema para lo cual enviará un email de comprobación con un link de confirmación, si en 24 horas no se ha recibido la confirmación el sistema continúa utilizando el email anterior. | |

| Caso de uso: **Gestionar lista negra de IPs** | |
|---|---|
| *Objetivo:* No permitir intentos de suplantación de identidad. | |
| *Actor* | Sistema. |
| *Precondición* | Fallos reiterativos a la hora de hacer Login. |
| *Postcondición* | La IP de origen no puede acceder a la pantalla de Login. |
| *Flujo de eventos* | |
| 1. Cada vez que se produce un fallo a la hora de hacer un Login un contador de tiempo y reintentos se pone en marcha.<br>2. Se comparan las condiciones de tiempo, numero de reintentos, y datos de acceso con las políticas de seguridad.<br>3. Se determina la acción a tomar. | |
| *Alternativas de proceso y excepciones* | |
| | |

| Caso de uso: **Envío de emails** | |
|---|---|
| *Objetivo:* Gestionar el envío de cualquier tipo de email. | |
| *Actor* | Sistema. |
| *Precondición* | |
| *Postcondición* | Un email es enviado. |
| *Flujo de eventos* | |
| 1. El sistema genera y controla la entrega de cualquier tipo de email. | |
| *Alternativas de proceso y excepciones* | |
| | |

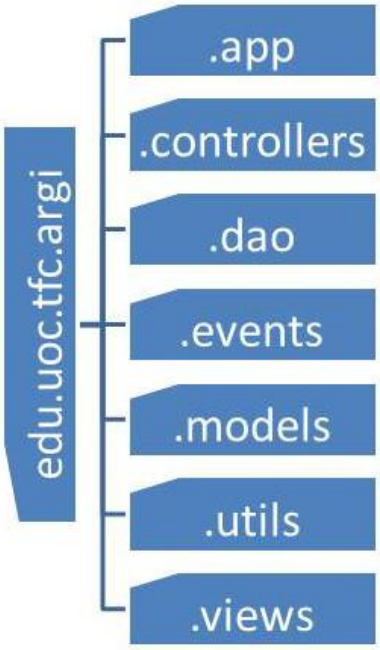
| Caso de uso: **Gestionar altas de Registro no confirmadas** | |
|---|---|
| *Objetivo:* Eliminar altas de Registro no confirmadas en un período inferior a 24 horas. | |
| *Actor* | Sistema. |
| *Precondición* | Se realizó un alta de Registro en el sistema que está pendiente de confirmar. |
| *Postcondición* | El registro de alta es eliminado. |
| *Flujo de eventos* | |
| 1. Cada hora son comprobadas las altas de Registro que están pendientes de confirmar.<br>2. Las que llevan más de 24 horas son eliminadas. | |
| *Alternativas de proceso y excepciones* | |
| | |

# 3. Diseño

Esta es la fase más técnica donde se desarrolla y especifica cómo se pretende resolver el proyecto tras haber realizado el análisis. Con un diseño basado en programación orientada a objetos la tarea principal es identificar las clases que compondrán la aplicación, la forma en que se hará la persistencia y la interacción con el usuario final mediante la interfaz gráfica.

## 3.1 Los Paquetes y sus clases

Las clases están agrupadas por funcionalidad en paquetes Java y la estructura de la aplicación ha quedado distribuida de la siguiente forma:



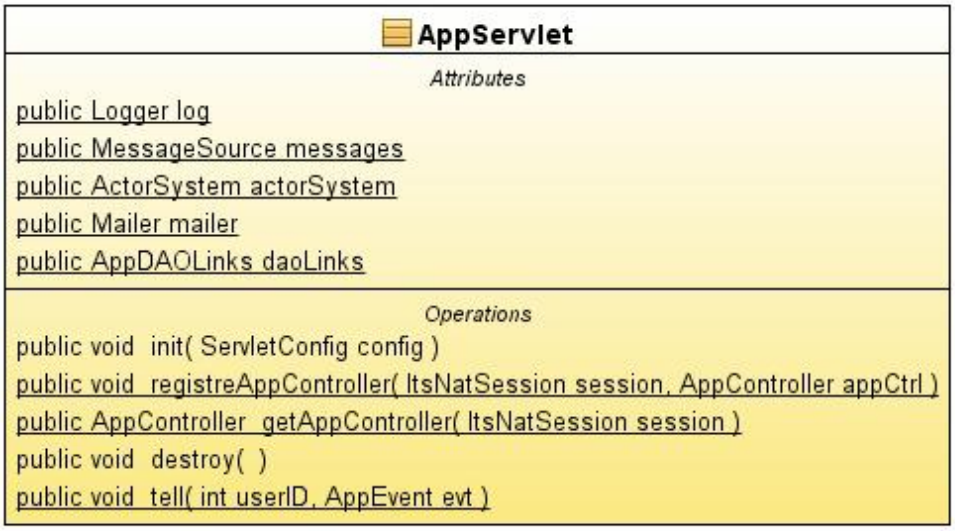

A continuación se detallan las clases de cada paquete y sus funcionalidades principales.

### 3.1.1 Paquete App

Contiene la clase principal *AppServlet.class* que es el punto de entrada a la aplicación. Hereda de la clase *HttpServlet* y en ella se configuran los parámetros básicos del framework ItsNat, se especifica donde se encuentran las plantillas HTML y se registran los listeneters pertinentes que atienden la petición inicial del navegador web.
También lanza el ApplicationContext de Spring, si, no me he equivocado, estoy utilizando parte de Spring como soporte y no como base MVC para la aplicación y es el framework ItsNat quien controla la aplicación.

Así que se registran las siguientes clases a nivel estático de aplicación:

- Log : logger de la aplicación.
- MessageSource : sistema de traducciones i18n.
- Mailer : gestión de envío de emails Spring + JavaMail.
- AppDAOLinks: enlace a las clases DAO.
- ActorSystem: actor principal del framework Akka que controla todos los demás actores Akka creados por la aplicación.





### 3.1.2 Paquete Controllers

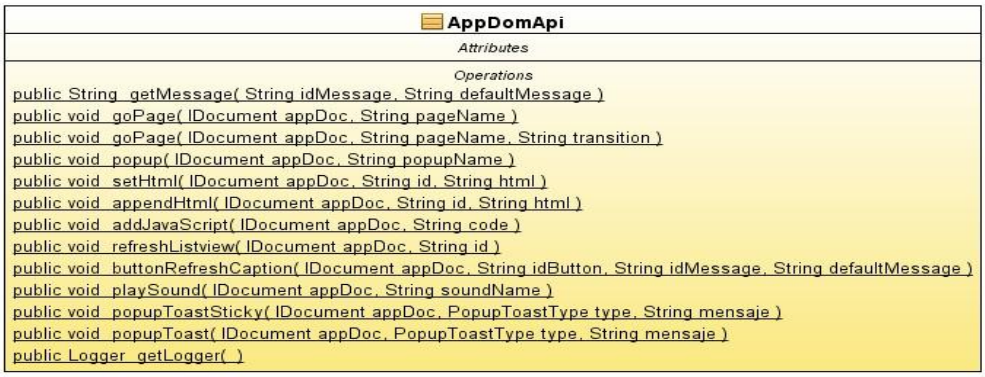
Aquí se encuentran unas de las clases más importantes del programa. Son las que controlan el comportamiento del programa cuando el usuario interactúa con la interfaz gráfica.

En primer lugar acceden al DOM para engancharse a los controles HTML necesarios, reconociéndolos por su tag *id*. También registran los listeners adecuados para responder a los eventos del usuario en el navegador. Así cuando el usuario introduce un valor en un campo de texto o pincha un botón estas acciones son informadas al servlet que actúa en el DOM con la respuesta pertinente.

He creado una clase estática llamada *AppDomAPI.class* que contiene métodos genéricos y que facilita el acceso y manipulación del DOM.

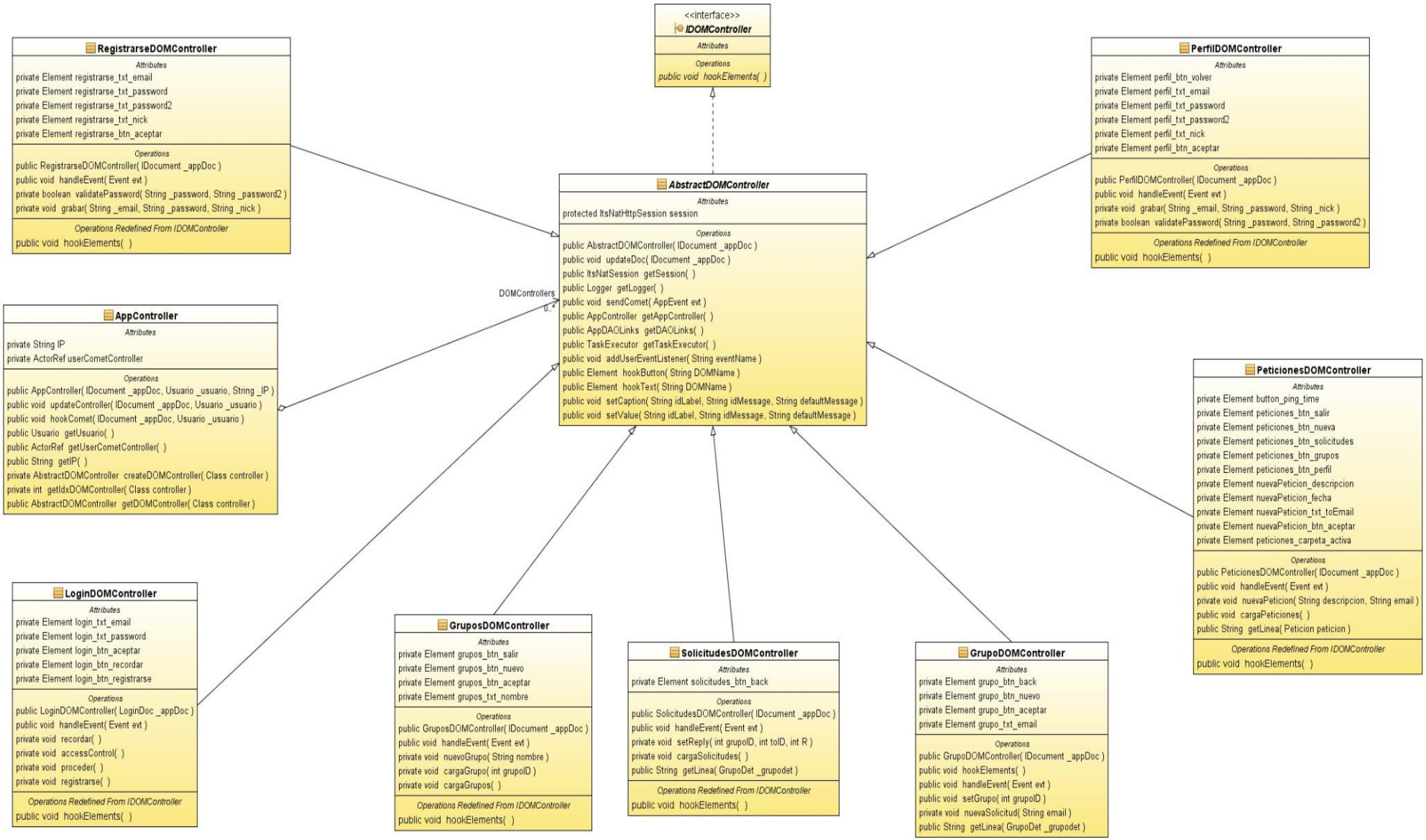
La navegación web fue creada teniendo en mente la acción petición -> respuesta, el usuario pide algo -> el servidor responde. Pero la evolución de la web ha requerido ingeniárselas para enviar información sin que esta sea solicitada. Se trata de COMET o Long Polling, se engaña al navegador sin enviarle el código de final de respuesta dejándolo de esta forma enganchados a la escucha por si es necesario transmitirle más información. No parece una manera muy eficiente de trabajar, pero es lo que había hasta ahora, por eso actualmente se está evolucionando hacia los WebSockets.

La clase *UserCometController.class* utiliza la tecnología COMET para modificar el DOM sin que el usuario lo solicite.



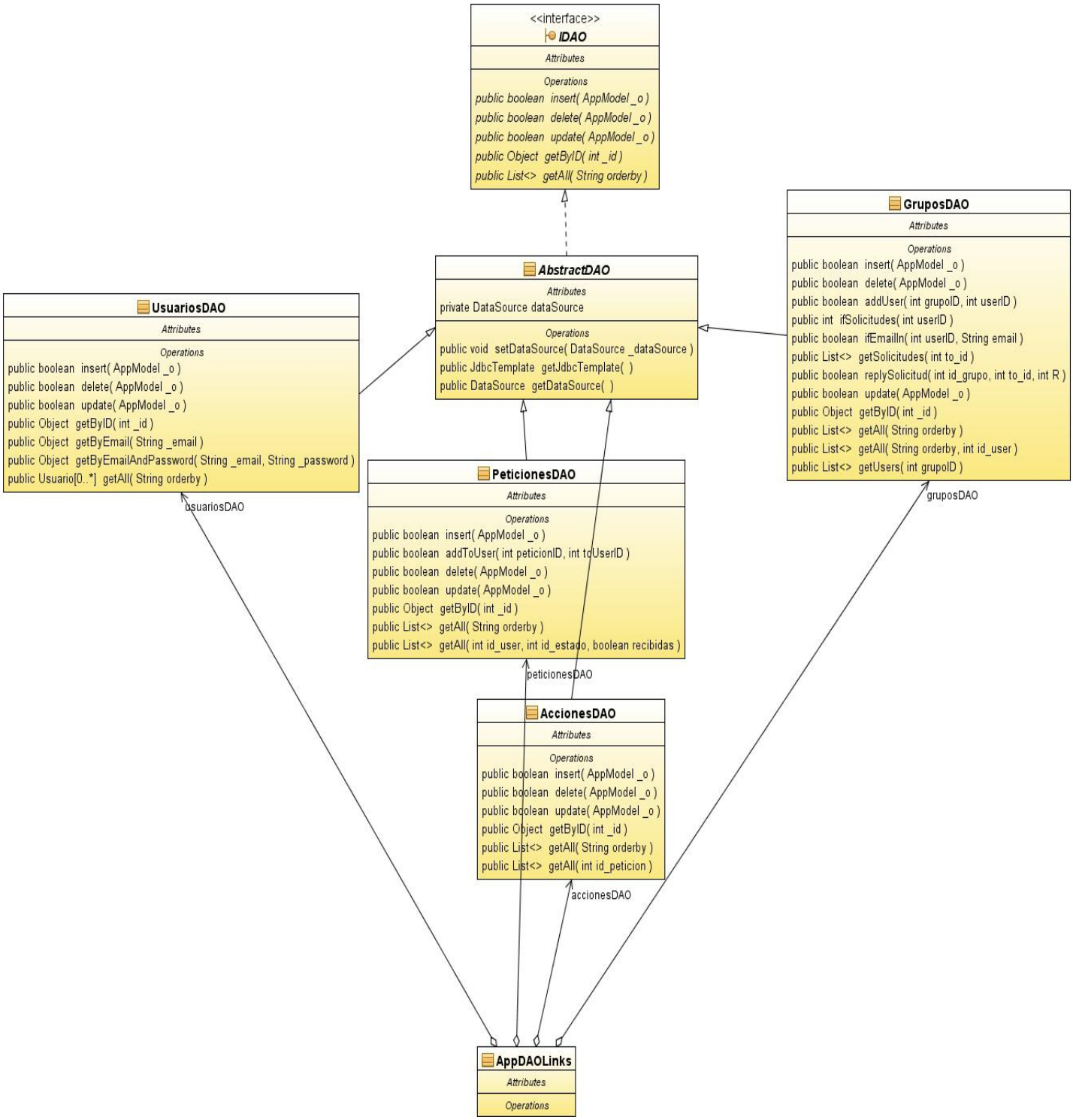


Merece mención especial la clase *AppController.class* pues encapsula todas las clases controller e identifica al usuario. Esta clase es registrada en la variable *session* con la que el servidor de aplicaciones identifica de forma unívoca al navegador cliente y se hace en la aplicación justo después de que usuario haya realizado el login.


<<interface>>
IDOMController
Attributes
Operations
public void hookElements( )
RegistrarseDOMController
Attributes
private Element registrarse_txt_email
private Element registrarse_txt_password
private Element registrarse_txt_password2
private Element registrarse_txt_nick
private Element registrarse_btn_aceptar
Operations
public RegistrarseDOMController( IDocument _appDoc )
public void handleEvent( Event evt )
private boolean validatePassword( String _password, String _password2 )
private void grabar( String _email, String _password, String _nick )
Operations Redefined From IDOMController
public void hookElements( )
PerfilDOMController
Attributes
private Element perfil_btn_volver
private Element perfil_txt_email
private Element perfil_txt_password
private Element perfil_txt_password2
private Element perfil_txt_nick
private Element perfil_btn_aceptar
Operations
public PerfilDOMController( IDocument _appDoc )
public void handleEvent( Event evt )
private void grabar( String _email, String _password, String _nick )
private boolean validatePassword( String _password, String _password2 )
Operations Redefined From IDOMController
public void hookElements( )
AbstractDOMController
Attributes
protected ItsNatHttpSession session
Operations
public AbstractDOMController( IDocument _appDoc )
public void updateDoc( IDocument _appDoc )
public ItsNatSession getSession( )
public Logger getLogger( )
public void sendComet( AppEvent evt )
public AppController getAppController( )
public AppDAOLinks getDAOLinks( )
public TaskExecutor getTaskExecutor( )
public void addUserEventListener( String eventName )
public Element hookButton( String DOMName )
public Element hookText( String DOMName )
public void setCaption( String idLabel, String idMessage, String defaultMessage )
public void setValue( String idLabel, String idMessage, String defaultMessage )
DOMControllers
0..*
AppController
Attributes
private String IP
private ActorRef userCometController
Operations
public AppController( IDocument _appDoc, Usuario _usuario, String _IP )
public void updateController( IDocument _appDoc, Usuario _usuario )
public void hookComet( IDocument _appDoc, Usuario _usuario )
public Usuario getUsuario( )
public ActorRef getUserCometController( )
public String getIP( )
private AbstractDOMController createDOMController( Class controller )
private int getIdxDOMController( Class controller )
public AbstractDOMController getDOMController( Class controller )
PeticionesDOMController
Attributes
private Element button_ping_time
private Element peticiones_btn_salir
private Element peticiones_btn_nueva
private Element peticiones_btn_solicitudes
private Element peticiones_btn_grupos
private Element peticiones_btn_perfil
private Element nuevaPeticion_descripcion
private Element nuevaPeticion_fecha
private Element nuevaPeticion_txt_toEmail
private Element nuevaPeticion_btn_aceptar
private Element peticiones_carpeta_activa
Operations
public PeticionesDOMController( IDocument _appDoc )
public void handleEvent( Event evt )
private void nuevaPeticion( String descripcion, String email )
public void cargaPeticiones( )
public String getLinea( Peticion peticion )
Operations Redefined From IDOMController
public void hookElements( )
LoginDOMController
Attributes
private Element login_txt_email
private Element login_txt_password
private Element login_btn_aceptar
private Element login_btn_recordar
private Element login_btn_registrarse
Operations
public LoginDOMController( LoginDoc _appDoc )
public void handleEvent( Event evt )
private void recordar( )
private void accessControl( )
private void proceder( )
private void registrarse( )
Operations Redefined From IDOMController
public void hookElements( )
GruposDOMController
Attributes
private Element grupos_btn_salir
private Element grupos_btn_nuevo
private Element grupos_btn_aceptar
private Element grupos_txt_nombre
Operations
public GruposDOMController( IDocument _appDoc )
public void handleEvent( Event evt )
private void nuevoGrupo( String nombre )
private void cargaGrupo( int grupoID )
private void cargaGrupos( )
Operations Redefined From IDOMController
public void hookElements( )
SolicitudesDOMController
Attributes
private Element solicitudes_btn_back
Operations
public SolicitudesDOMController( IDocument _appDoc )
public void handleEvent( Event evt )
private void setReply( int grupoID, int toID, int R )
private void cargaSolicitudes( )
public String getLinea( GrupoDet _grupodet )
Operations Redefined From IDOMController
public void hookElements( )
GrupoDOMController
Attributes
private Element grupo_btn_back
private Element grupo_btn_nuevo
private Element grupo_btn_aceptar
private Element grupo_txt_email
Operations
public GrupoDOMController( IDocument _appDoc )
public void hookElements( )
public void handleEvent( Event evt )
public void setGrupo( int grupoID )
private void nuevaSolicitud( String email )
public String getLinea( GrupoDet _grupodet )

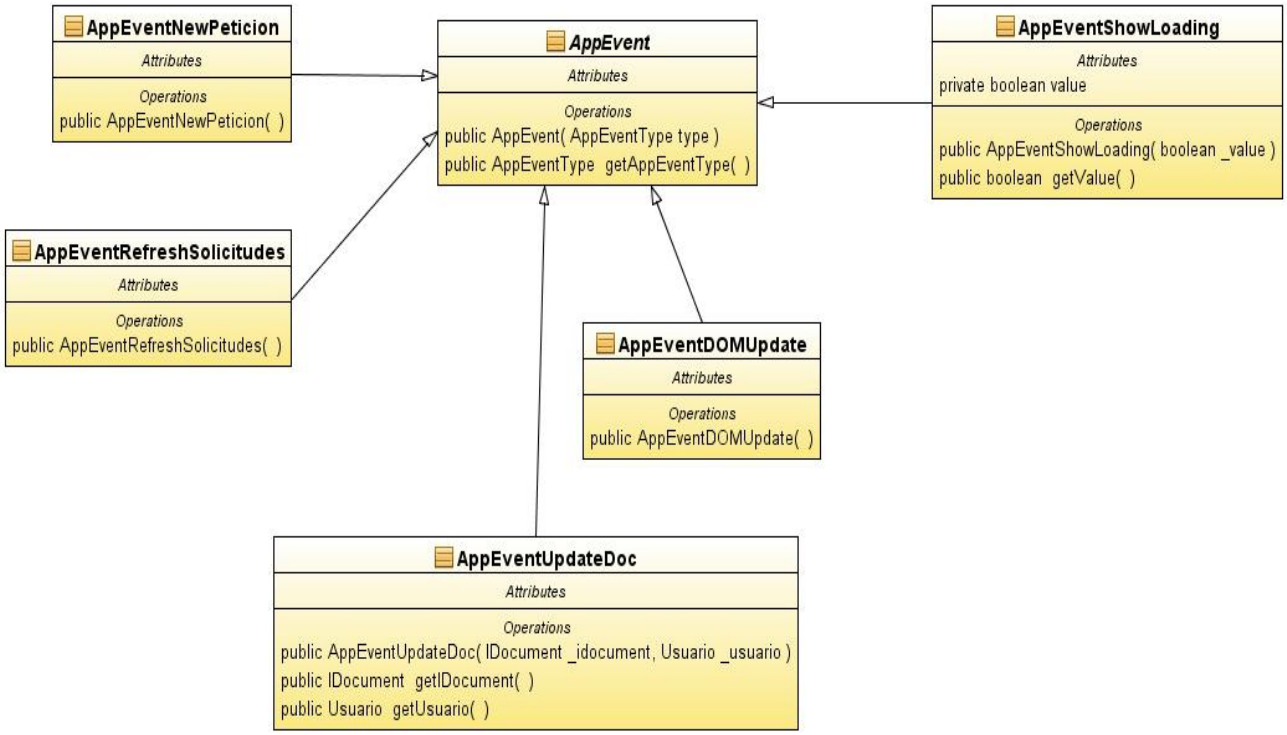
### 3.1.3 Paquete DAO

Los DAO u Objetos de Acceso a Datos son los encargados de gestionar la persistencia de las clases que lo requieren dentro de la aplicación. Están implementadas utilizando la clase JDBCTemplate de Spring Framework.



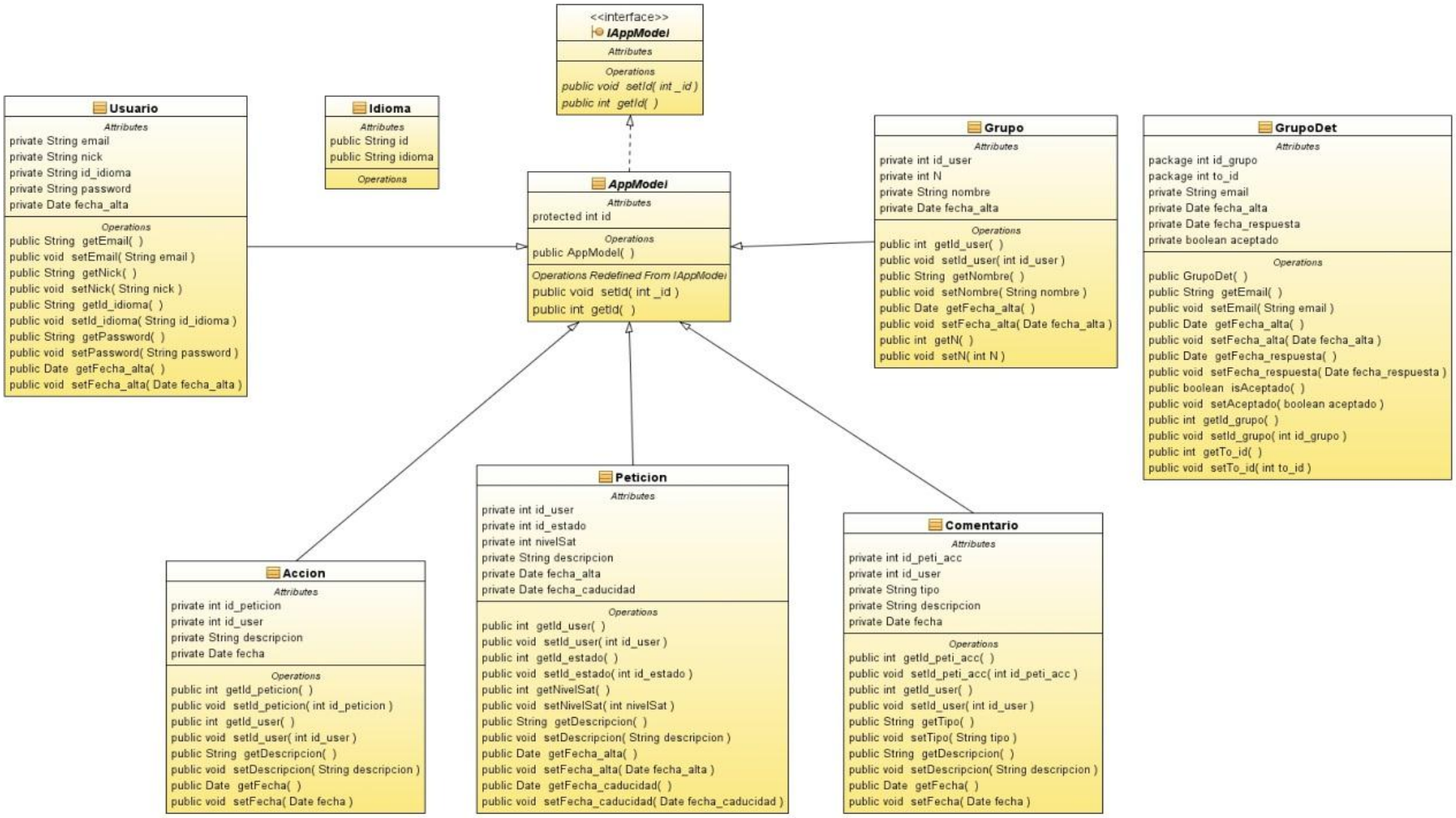
### 3.1.4 Paquete Events

Las clases descendientes de AppEvent son utilizadas para disparar eventos COMET, así que siempre son gestionadas por la clase *UserCometController.class* y pueden ser lanzadas desde cualquier punto de la aplicación una vez el usuario haya realizado login.



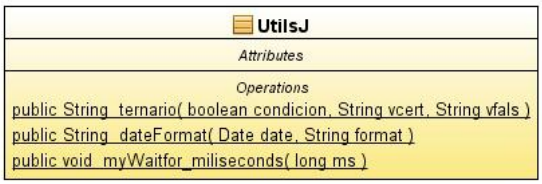

### 3.1.5 Paquete Models

Las clases del paquete Models son los POJOS, podríamos decir que son las clases básicas que implementan la lógica de negocio. Estas son manejadas por las clases DAO, y por tanto cualquier petición de selección y persistencia son solicitadas a estas.


<<interface>>
IAppModel
Attributes
Operations
public void setId( int _id )
public int getId( )
Usuario
Attributes
private String email
private String nick
private String id_idioma
private String password
private Date fecha_alta
Operations
public String getEmail( )
public void setEmail( String email )
public String getNick( )
public void setNick( String nick )
public String getId_idioma( )
public void setId_idioma( String id_idioma )
public String getPassword( )
public void setPassword( String password )
public Date getFecha_alta( )
public void setFecha_alta( Date fecha_alta )
Idioma
Attributes
public String id
public String idioma
Operations
AppModel
Attributes
protected int id
Operations
public AppModel( )
Operations Redefined From IAppModel
public void setId( int _id )
public int getId( )
Grupo
Attributes
private int id_user
private int N
private String nombre
private Date fecha_alta
Operations
public int getId_user( )
public void setId_user( int id_user )
public String getNombre( )
public void setNombre( String nombre )
public Date getFecha_alta( )
public void setFecha_alta( Date fecha_alta )
public int getN( )
public void setN( int N )
GrupoDet
Attributes
package int id_grupo
package int to_id
private String email
private Date fecha_alta
private Date fecha_respuesta
private boolean aceptado
Operations
public GrupoDet( )
public String getEmail( )
public void setEmail( String email )
public Date getFecha_alta( )
public void setFecha_alta( Date fecha_alta )
public Date getFecha_respuesta( )
public void setFecha_respuesta( Date fecha_respuesta )
public boolean isAceptado( )
public void setAceptado( boolean aceptado )
public int getId_grupo( )
public void setId_grupo( int id_grupo )
public int getTo_id( )
public void setTo_id( int to_id )
Accion
Attributes
private int id_peticion
private int id_user
private String descripcion
private Date fecha
Operations
public int getId_peticion( )
public void setId_peticion( int id_peticion )
public int getId_user( )
public void setId_user( int id_user )
public String getDescripcion( )
public void setDescripcion( String descripcion )
public Date getFecha( )
public void setFecha( Date fecha )
Peticion
Attributes
private int id_user
private int id_estado
private int nivelSat
private String descripcion
private Date fecha_alta
private Date fecha_caducidad
Operations
public int getId_user( )
public void setId_user( int id_user )
public int getId_estado( )
public void setId_estado( int id_estado )
public int getNivelSat( )
public void setNivelSat( int nivelSat )
public String getDescripcion( )
public void setDescripcion( String descripcion )
public Date getFecha_alta( )
public void setFecha_alta( Date fecha_alta )
public Date getFecha_caducidad( )
public void setFecha_caducidad( Date fecha_caducidad )
Comentario
Attributes
private int id_peti_acc
private int id_user
private String tipo
private String descripcion
private Date fecha
Operations
public int getId_peti_acc( )
public void setId_peti_acc( int id_peti_acc )
public int getId_user( )
public void setId_user( int id_user )
public String getTipo( )
public void setTipo( String tipo )
public String getDescripcion( )
public void setDescripcion( String descripcion )
public Date getFecha( )
public void setFecha( Date fecha )

### 3.1.6 Paquete Utils

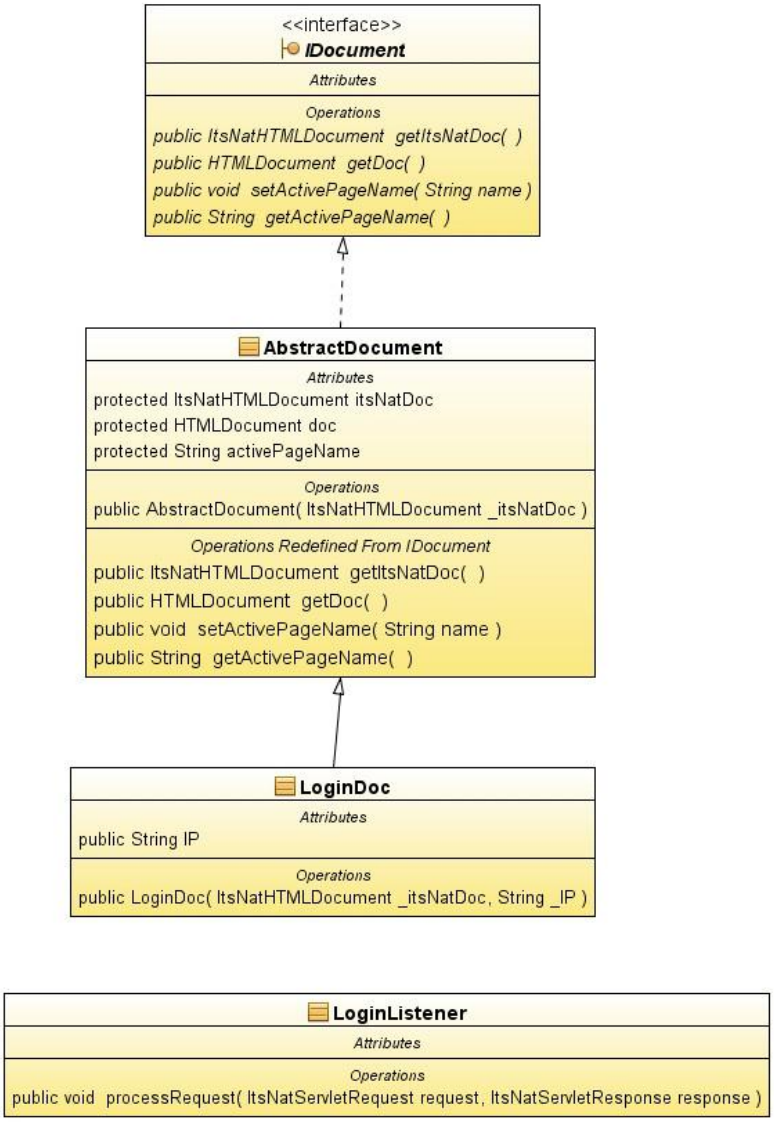
En el paquete Utils se localizan las clases de propósito general.





### 3.1.7 Paquete Views

La clase *LoginListener.class*, registrada en *AppServlet.class*, actúa como punto de entrada a la aplicación. Devuelve un *LoginDoc,* que lo que hace es cargar el código HTML que hay en *WEB-INF/docs/app.xhtml* e inicia el DOM con *LoginDOMController* que sitúa a el usuario en la pantalla de Login. A partir de este momento todo el código HTML está residente ya en el navegador del cliente y son las clases DOMControllers las que se encargan de modificarlo dinámicamente.

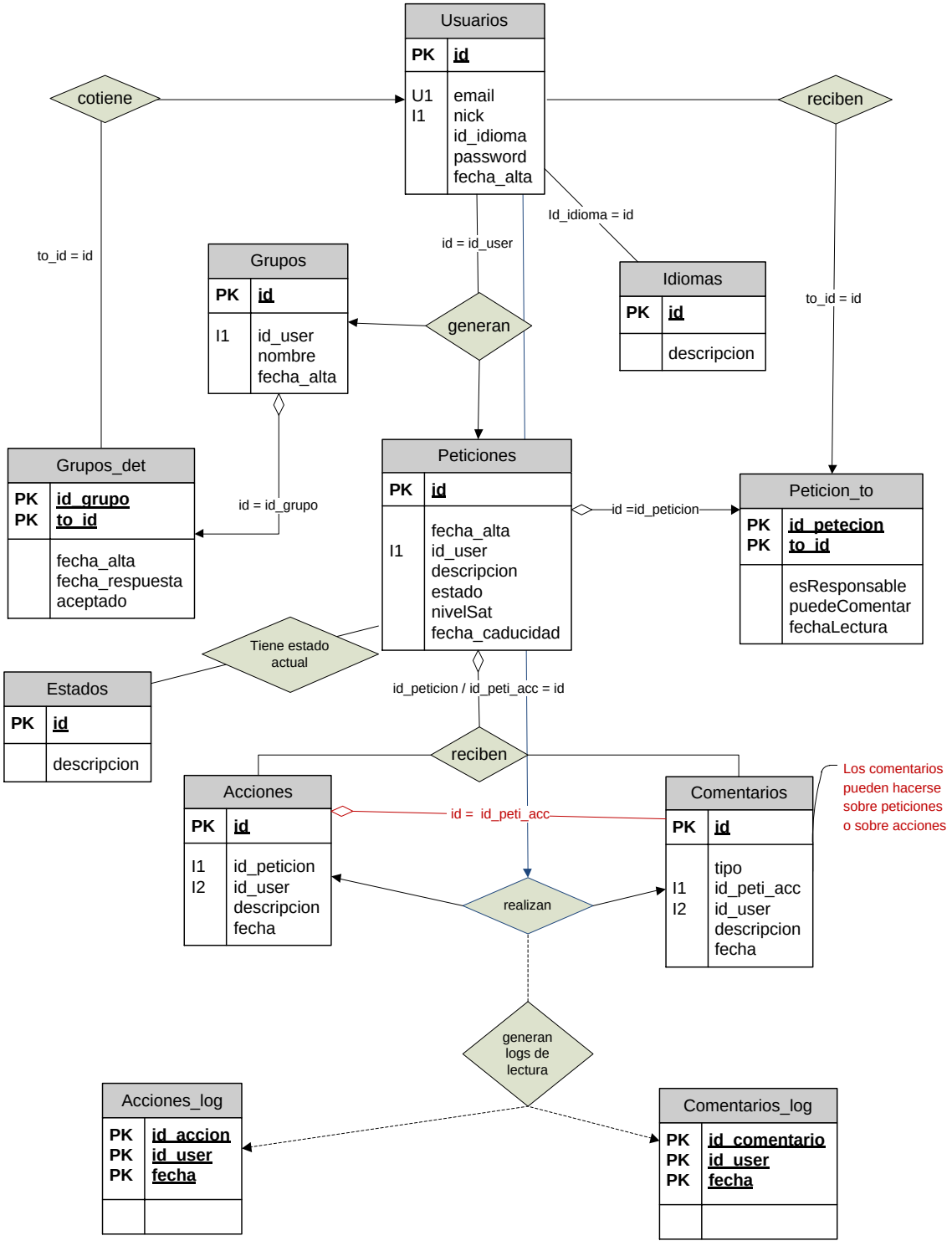



Existe también un paquete resources que contiene ficheros properties utilizados para facilitar la traducción de la aplicación siguiendo el estándar i18n.

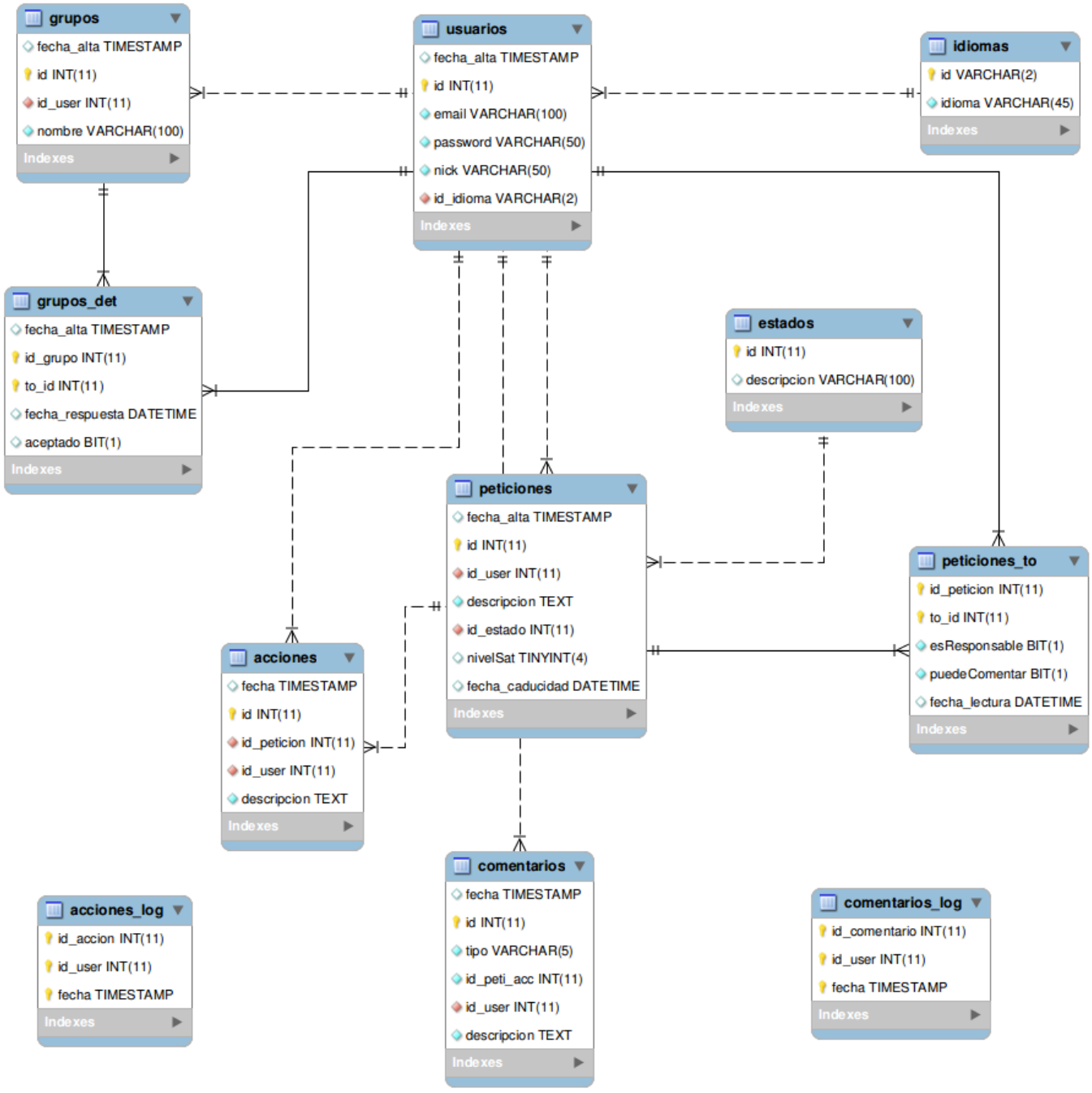
## 3.2 Modelo ER

El modelo entidad-relación que viene a continuación pretende representar las entidades del sistema, sus propiedades y como se interrelacionan.

Estas entidades son las que exigen persistencia en la base de datos y sus relaciones nos ayudarán en la definición de reglas de claves primarias y foráneas.



### 3.2.1 Diseño lógico y script SQL




```
CREATE TABLE estados (
  id int(11) NOT NULL,
  descripcion varchar(100) COLLATE utf8_unicode_ci DEFAULT NULL,
  PRIMARY KEY (id)
);

CREATE TABLE idiomas (
  id varchar(2) COLLATE utf8_unicode_ci NOT NULL,
  idioma varchar(45) COLLATE utf8_unicode_ci NOT NULL,
  PRIMARY KEY (id)
);
```

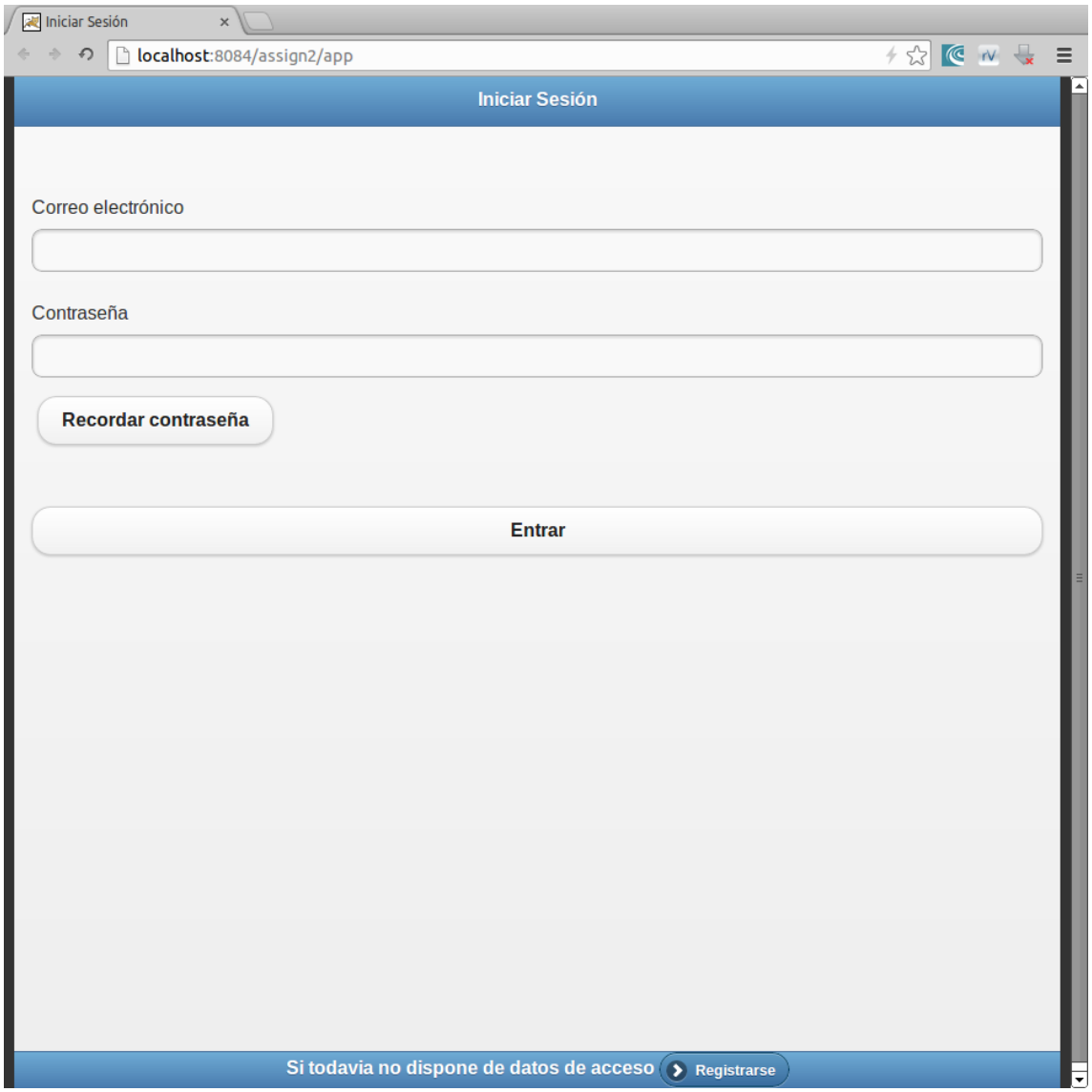
```
CREATE TABLE usuarios (
  fecha_alta timestamp NULL DEFAULT CURRENT_TIMESTAMP,
  id int(11) NOT NULL AUTO_INCREMENT,
  email varchar(100) COLLATE utf8_unicode_ci NOT NULL,
  password varchar(50) COLLATE utf8_unicode_ci NOT NULL,
  nick varchar(50) COLLATE utf8_unicode_ci NOT NULL,
  id_idioma varchar(2) COLLATE utf8_unicode_ci NOT NULL,
  PRIMARY KEY (id),
  UNIQUE KEY idx_email (email),
  KEY idx_nick (nick),
  KEY fk_idioma_idx (id_idioma),
  CONSTRAINT fk_idioma FOREIGN KEY (id_idioma) REFERENCES idiomas (id) ON DELETE NO ACTION ON
UPDATE NO ACTION
);

CREATE TABLE grupos (
  fecha_alta timestamp NULL DEFAULT CURRENT_TIMESTAMP,
  id int(11) NOT NULL AUTO_INCREMENT,
  id_user int(11) NOT NULL,
  nombre varchar(100) COLLATE utf8_unicode_ci NOT NULL,
  PRIMARY KEY (id),
  UNIQUE KEY fk_grupos_1_idx (id_user,nombre),
  CONSTRAINT fk_grupos_1 FOREIGN KEY (id_user) REFERENCES usuarios (id) ON DELETE NO ACTION ON
UPDATE NO ACTION
);

CREATE TABLE grupos_det (
  fecha_alta timestamp NULL DEFAULT CURRENT_TIMESTAMP,
  id_grupo int(11) NOT NULL,
  to_id int(11) NOT NULL,
  fecha_respuesta datetime DEFAULT NULL,
  aceptado bit(1) DEFAULT NULL,
  PRIMARY KEY (id_grupo,to_id),
  KEY fk_grupos_det_1_idx (id_grupo),
  KEY fk_grupos_det_2_idx (to_id),
  CONSTRAINT fk_grupos_det_1 FOREIGN KEY (id_grupo) REFERENCES grupos (id) ON DELETE NO ACTION
ON UPDATE NO ACTION,
  CONSTRAINT fk_grupos_det_2 FOREIGN KEY (to_id) REFERENCES usuarios (id) ON DELETE NO ACTION ON
UPDATE NO ACTION
);

CREATE TABLE peticiones (
  fecha_alta timestamp NULL DEFAULT CURRENT_TIMESTAMP,
  id int(11) NOT NULL AUTO_INCREMENT,
  id_user int(11) NOT NULL,
  descripcion text COLLATE utf8_unicode_ci NOT NULL,
  id_estado int(11) NOT NULL,
  nivelSat tinyint(4) DEFAULT NULL,
  fecha_caducidad datetime DEFAULT NULL,
  PRIMARY KEY (id),
  KEY fk_peticiones_1_idx (id_user),
  KEY fk_peticiones_2_idx (id_estado),
  CONSTRAINT fk_peticiones_1 FOREIGN KEY (id_user) REFERENCES usuarios (id) ON DELETE NO ACTION
ON UPDATE NO ACTION,
  CONSTRAINT fk_peticiones_2 FOREIGN KEY (id_estado) REFERENCES estados (id) ON DELETE NO ACTION
ON UPDATE NO ACTION
);
```

```
CREATE TABLE peticiones_to (
  id_peticion int(11) NOT NULL,
  to_id int(11) NOT NULL,
  esResponsable bit(1) NOT NULL,
  puedeComentar bit(1) NOT NULL,
  fecha_lectura datetime DEFAULT NULL,
  PRIMARY KEY (id_peticion,to_id),
  KEY fk_peticiones_to_1_idx (id_peticion),
  KEY fk_peticiones_to_2_idx (to_id),
  CONSTRAINT fk_peticiones_to_1 FOREIGN KEY (id_peticion) REFERENCES peticiones (id) ON DELETE
NO ACTION ON UPDATE NO ACTION,
  CONSTRAINT fk_peticiones_to_2 FOREIGN KEY (to_id) REFERENCES usuarios (id) ON DELETE NO ACTION
ON UPDATE NO ACTION
);

CREATE TABLE acciones (
  fecha timestamp NULL DEFAULT CURRENT_TIMESTAMP,
  id int(11) NOT NULL AUTO_INCREMENT,
  id_peticion int(11) NOT NULL,
  id_user int(11) NOT NULL,
  descripcion text COLLATE utf8_unicode_ci NOT NULL,
  PRIMARY KEY (id),
  KEY fk_acciones_1_idx (id_user),
  KEY fk_acciones_2_idx (id_peticion),
  CONSTRAINT fk_acciones_1 FOREIGN KEY (id_user) REFERENCES usuarios (id) ON DELETE NO ACTION ON
UPDATE NO ACTION,
  CONSTRAINT fk_acciones_2 FOREIGN KEY (id_peticion) REFERENCES peticiones (id) ON DELETE NO
ACTION ON UPDATE NO ACTION
);

CREATE TABLE acciones_log (
  id_accion int(11) NOT NULL,
  id_user int(11) NOT NULL,
  fecha timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (id_accion,id_user,fecha)
);

CREATE TABLE comentarios (
  fecha timestamp NULL DEFAULT CURRENT_TIMESTAMP,
  id int(11) NOT NULL AUTO_INCREMENT,
  tipo varchar(5) COLLATE utf8_unicode_ci NOT NULL,
  id_peti_acc int(11) NOT NULL,
  id_user int(11) NOT NULL,
  descripcion text COLLATE utf8_unicode_ci NOT NULL,
  PRIMARY KEY (id),
  KEY fk_comentarios_1_idx (id_user),
  CONSTRAINT fk_comentarios_1 FOREIGN KEY (id_user) REFERENCES usuarios (id) ON DELETE NO ACTION
ON UPDATE NO ACTION
);

CREATE TABLE comentarios_log (
  id_comentario int(11) NOT NULL,
  id_user int(11) NOT NULL,
  fecha timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (id_comentario,id_user,fecha)
);
```

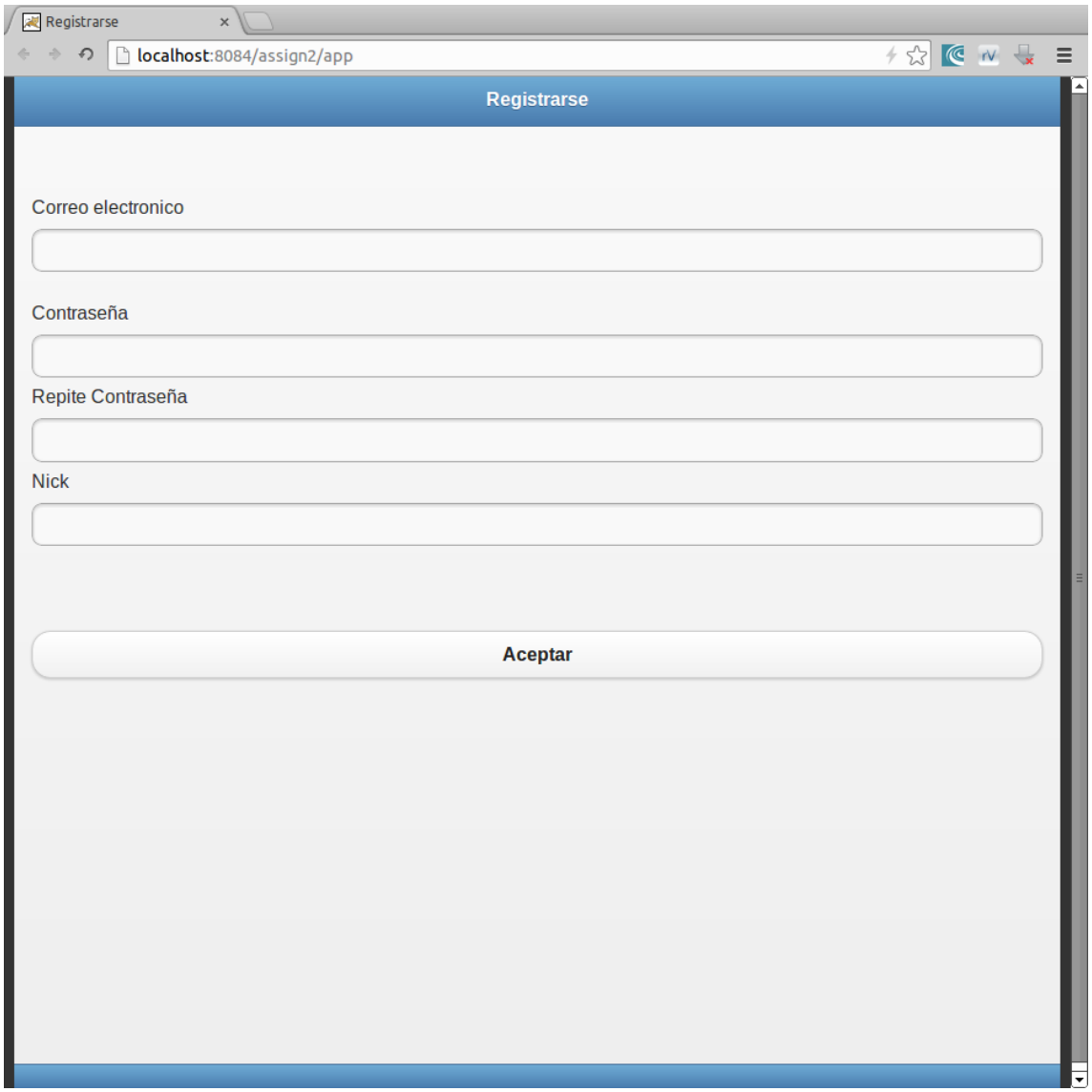
## 3.3 Interfaz gráfica de usuario

Existen muchos y buenos frameworks CSS, como por ejemplo Bootstrap Twitter, pero buscando un poco más he conocido jQueryMobile, que sin duda se acerca más a lo que estaba buscando para este proyecto.

Al conectarnos a la aplicación lo primero que nos encontramos es la pantalla de login.




Simplemente requiere los datos de acceso y pulsar el botón entrar. Si no recordamos la contraseña podemos pulsar el link *Recordar contraseña* que nos enviará una contraseña provisional, siempre y cuando hayamos introducido un correo electrónico valido para el sistema. Si no tenemos datos de acceso podemos pulsar el botón *Registrarse* el cual nos conducirá al formulario de alta de registro.

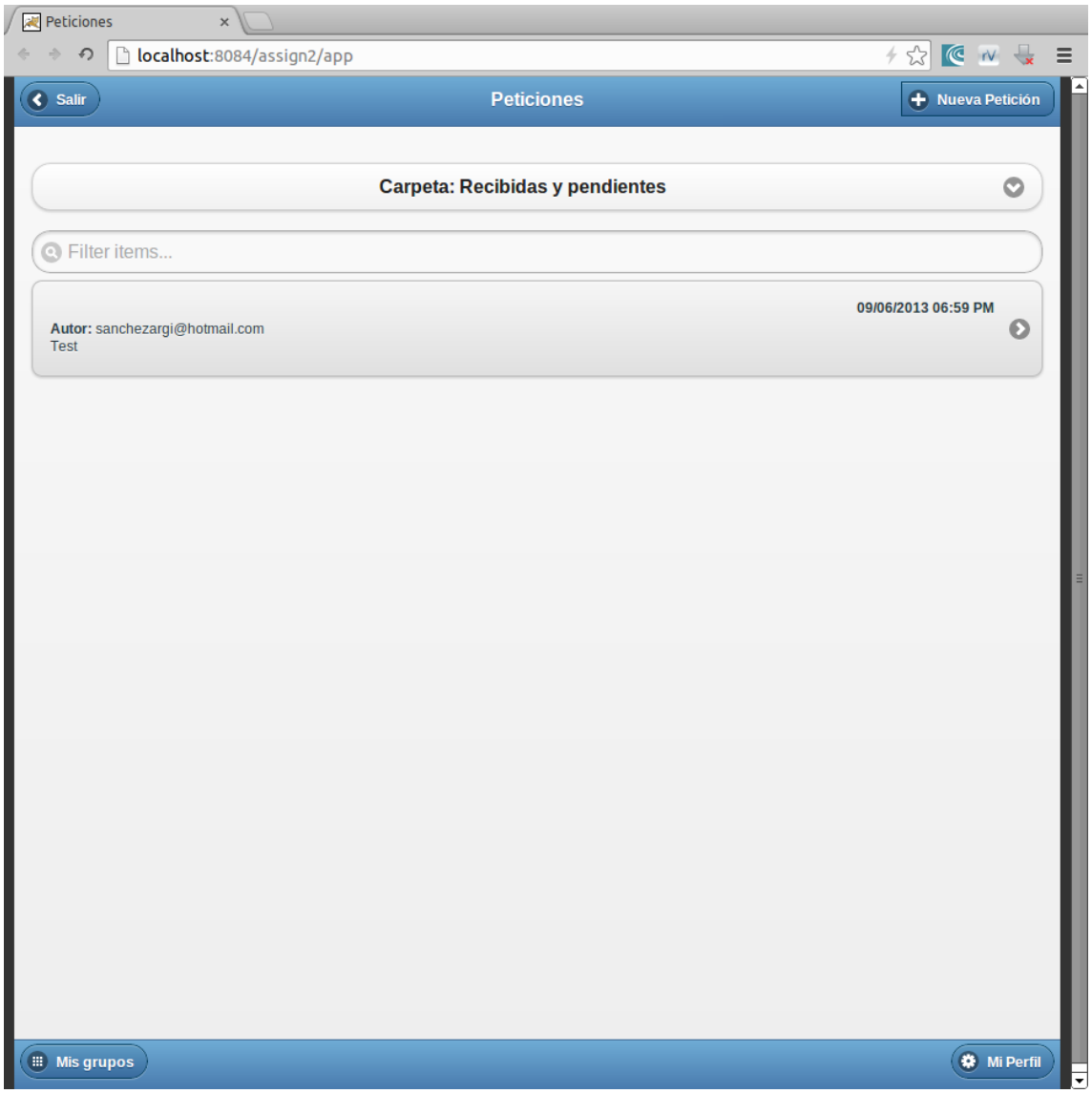



En este punto me gustaría hacer un inciso para hacer unas puntualizaciones respecto al tema de seguridad:

Durante todo el desarrollo la aplicación se ha estado ejecutando sobre http, pero como una aplicación pensada para correr en internet y con información confidencial esta debería correr sobre https, por lo que deberíamos activar SSL en Tomcat mediante la adquisición de un certificado digital a una entidad certificadora. De esta forma todos los datos viajaran encriptados por la red.

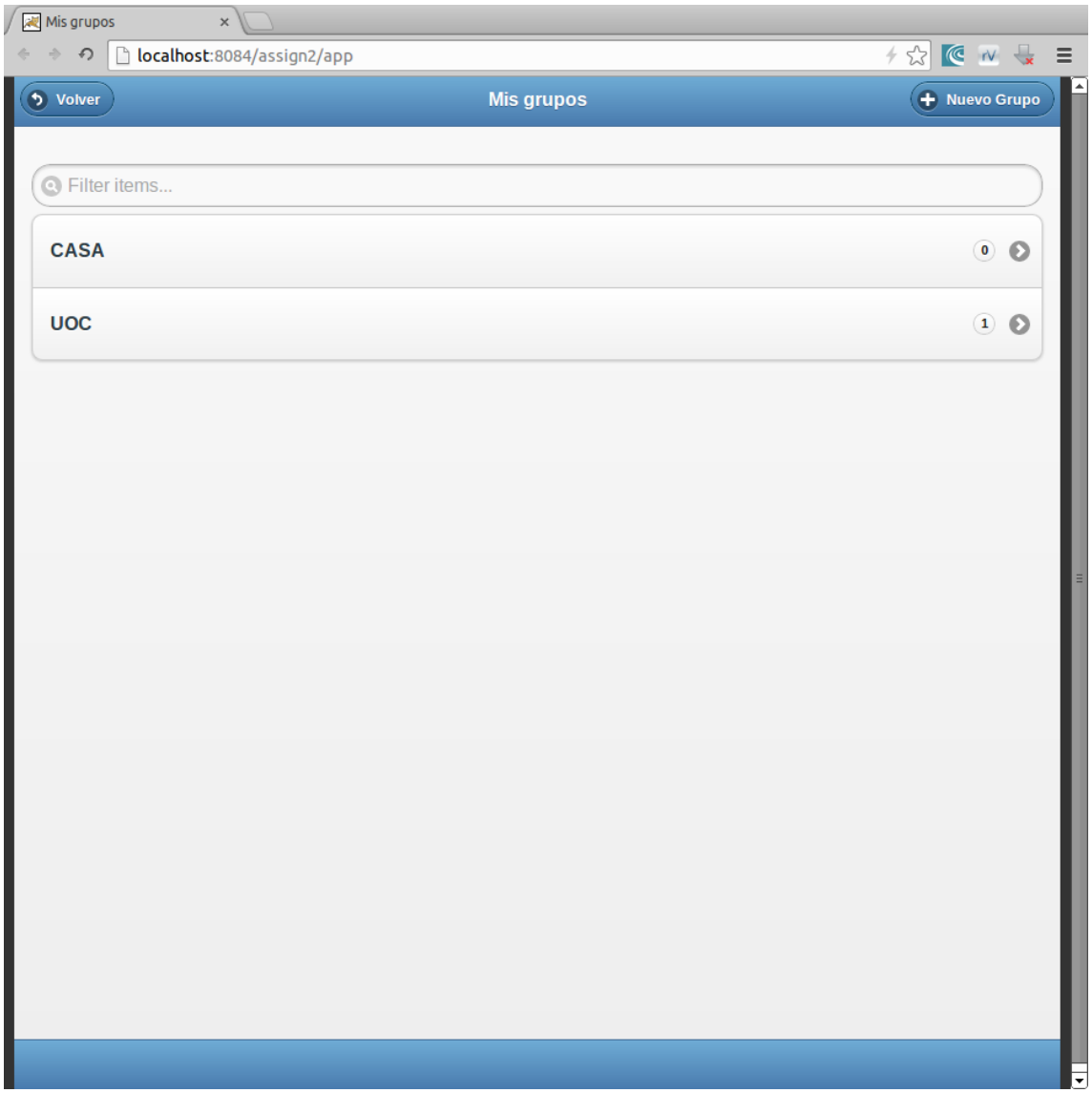
Otra observación importante es que todas las contraseñas son grabadas en la DB mediante una función Hash. Las funciones Hash son conocidas como funciones de solo ida, pues no tienen ningún algoritmo de retorno que permita hacer una desencriptación, de esta forma solo el usuario sabe realmente su contraseña y esta no podrá ser deducida ni siquiera por los administradores del sistema con acceso directo a la base de datos.

Tras hacer el login automáticamente somos reconducidos a la pantalla principal, que por defecto es la carpeta de peticiones recibidas y pendientes:



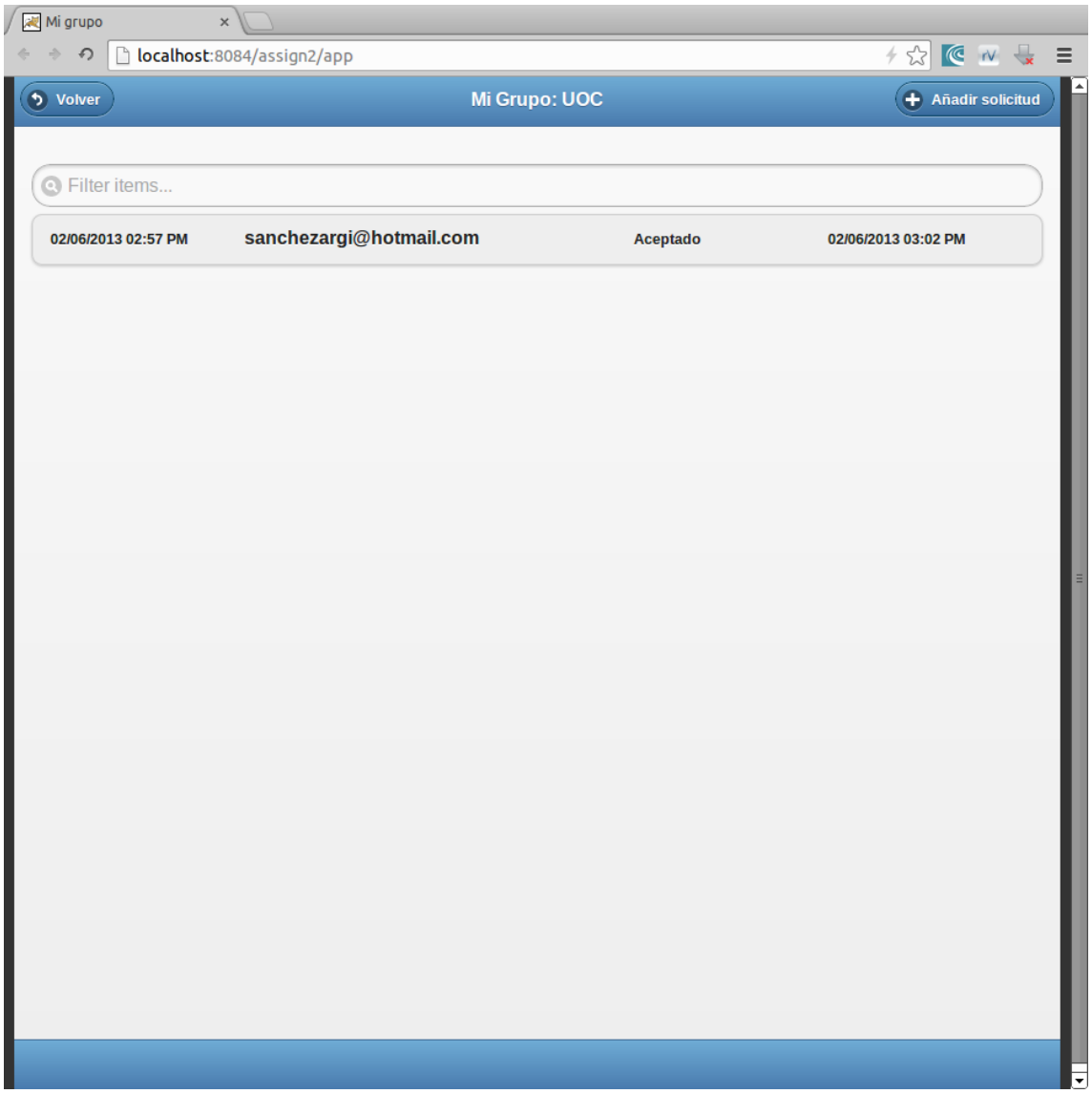

Tenemos un desplegable que nos permite cambiar de carpeta. Como en todas las pantallas con líneas de detalle disponemos de un campo de filtrado rápido que nos facilita las búsquedas por cualquier trozo de cadena que aparezca en la línea.

Desde esta pantalla accedemos al resto de opciones y pulsando el botón [Mis grupos] accedemos a la gestión de grupos donde se muestra todas nuestras agrupaciones en orden alfabético y el número de usuarios que hay dentro de cada grupo.

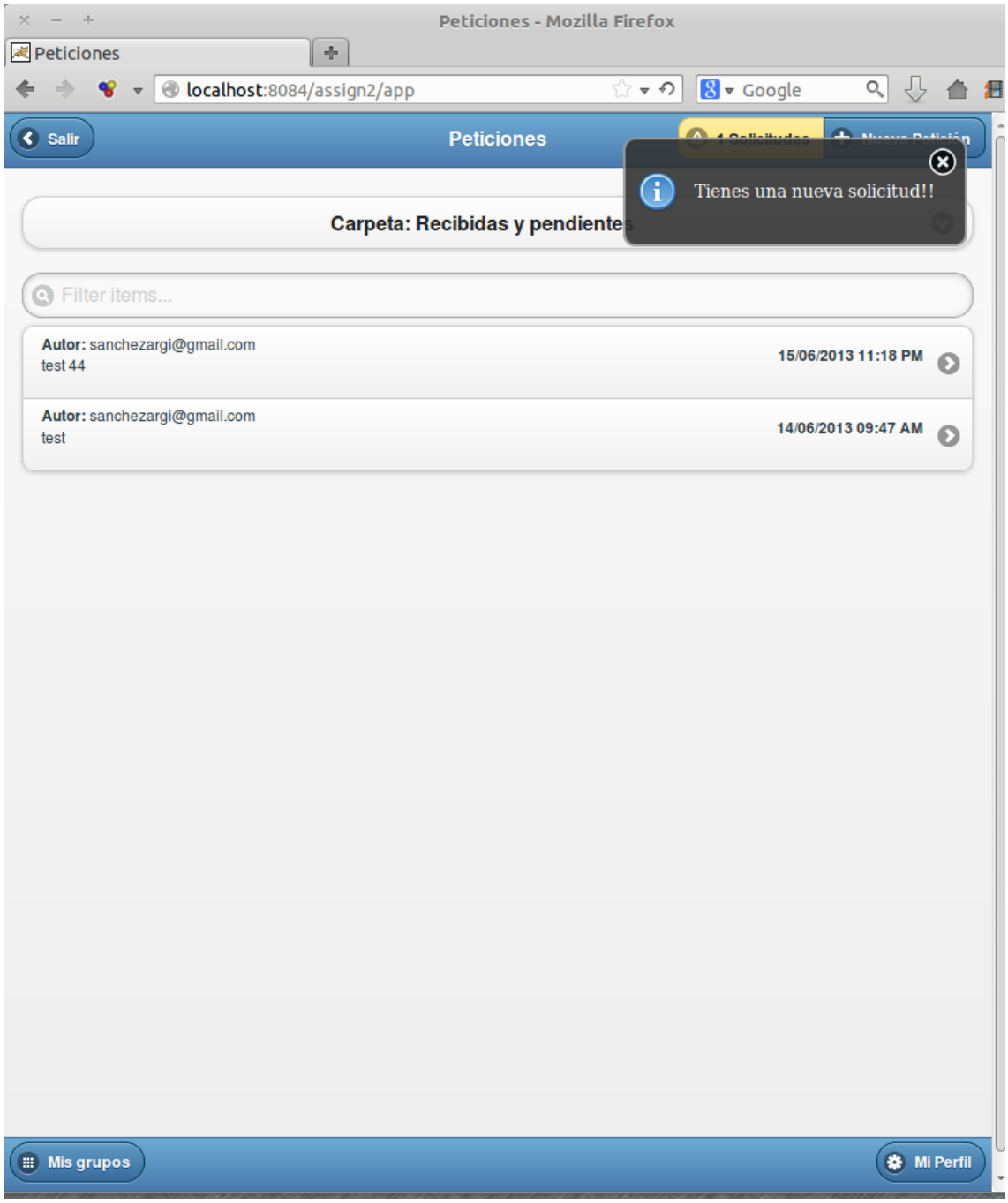



Si pulsamos en el nombre del grupo entramos en la vista detallada desde la que podemos enviar solicitudes a otros usuarios para que nos den el permiso de añadirlos.

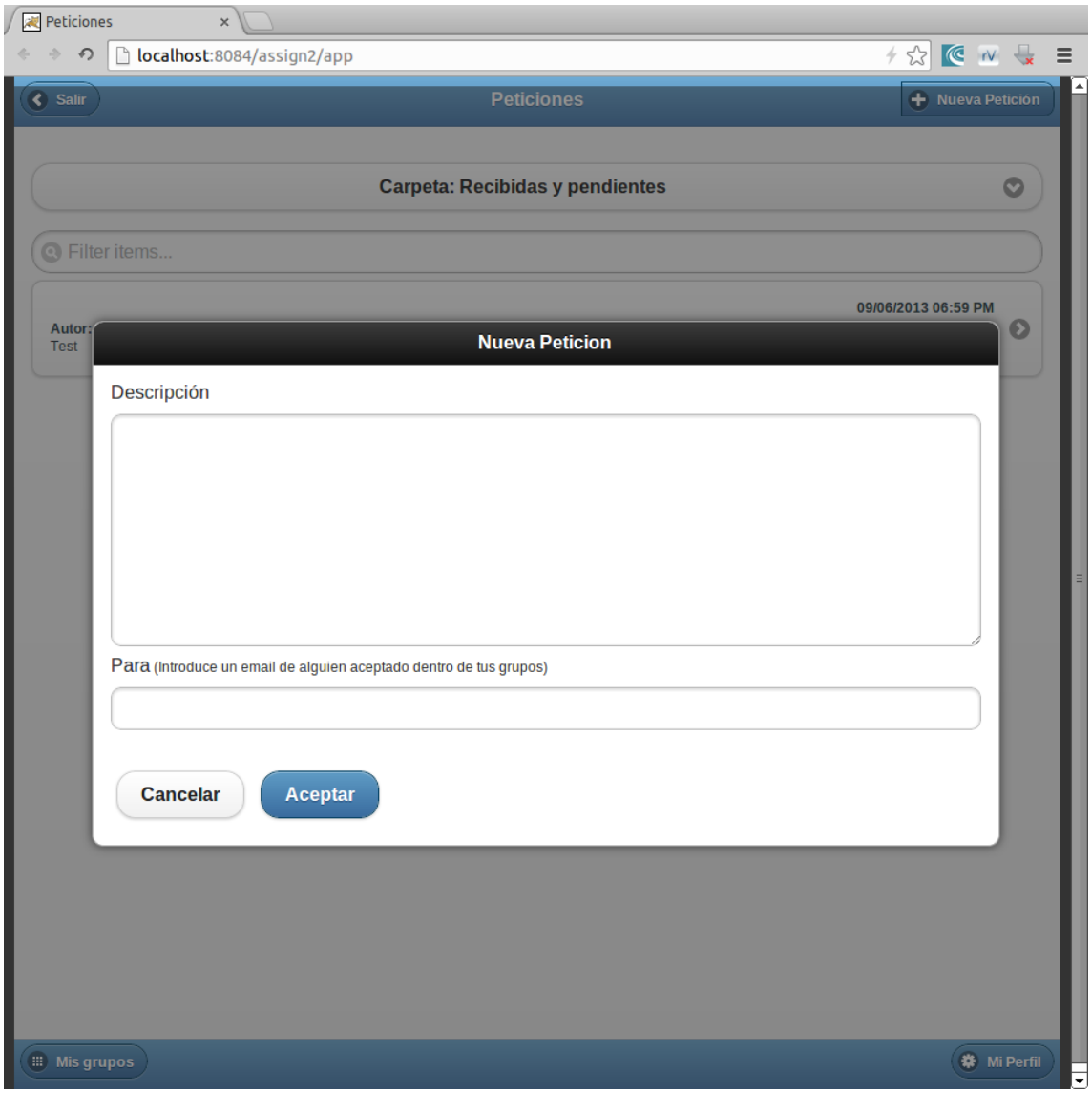
Cuando alguien acepta nuestra invitación de unirse a uno de nuestros grupos podremos ver en el detalle del grupo su respuesta y la fecha y hora en que lo hizo.



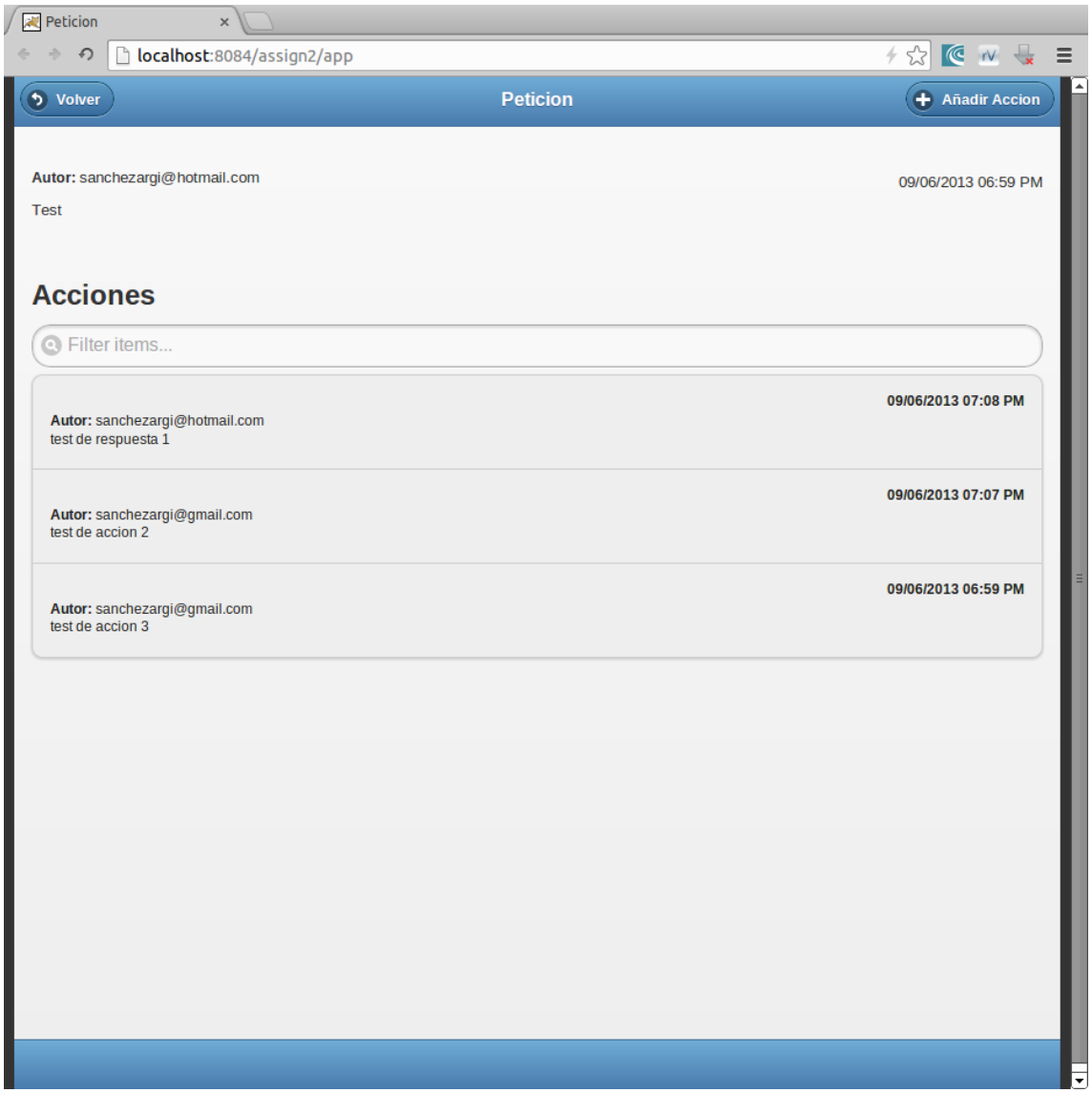
Cuando otro usuario nos envía una solicitud o una petición y estamos online se nos notifica de inmediato con un mensaje popup acompañado de un sonido, aquí es donde estoy utilizando la tecnología COMET al entregar información sin que hay sido solicitada. Si no estamos online lo veremos la próxima vez que entremos en el sistema.



Para enviar una petición a un usuario lo haremos desde la pantalla principal pulsando el botón [+Nueva Petición]. Rellenamos la descripción e introducimos el email de un usuario existente en alguno de nuestros grupos.



Al pinchar sobre una petición accederemos a su detalle y mientras la petición no haya sido dada por finalizada podremos añadir nuevas acciones pulsado en botón [+Añadir Acción]. Es solo el usuario autor de la petición quien puede dar por finalizada una petición.



## 3.4 Arquitectura

Como en toda aplicación web tenemos una parte servidora y una parte cliente. En este caso en la parte servidora tenemos el servidor de aplicaciones, corriendo sobre la JVM, que atenderá las peticiones de los clientes web. En la parte cliente cualquier navegador web, que se encargará de realizar las peticiones necesarias al servidor, interpretando y renderizando el código HTML + JavaScript.

Que nuestro servidor corra sobre J2EE es una ventaja, pues no será por falta de posibilidades, ya que las hay para volverte loco: miles de frameworks, paradigmas y filosofías de programación.

Después de comparar muchos frameworks (Spring, Struts, ZK, Apache Wicket, Apache Click, Vaadín, etc.) y revisar un poco las diferentes filosofías (3 Capas -Presentación, Lógica de Negocio, Controlador-, MVC, MVVC, Inversión de Control, Inyección de dependencias) he llegado a una conclusión: utilizar el principio de programación KISS.

El despliegue del acrónimo KISS puede sonar un poco fuerte: Keep It Simple Stupid (Manténlo sencillo, estúpido). Sin la intención de insultar a nadie este principio defiende que los sistemas funcionan mejor cuando tienen diseños simples. Así que esa ha sido mi idea: partir de un diseño simple.

Los frameworks y componentes escogidos para este proyecto han sido los siguientes:

**ItsNat**: http://itsnat.sourceforge.net
A diferencia de muchos frameworks web que te marcan como hacer las cosas, siendo que a veces cosas sencillas se vuelven algo complejas, ItsNat te permite trabar a tu antojo. Esto es así porque te da un acceso muy amigable al DOM, puedes utilizar Ajax fácilmente y te brinda acceso a la tecnología COMET de forma muy simplificada. De hecho su acrónimo viene a ser en castellano "Es natural o de forma natural". Yo he trabajado con la versión 1.2 pero el día 8 de Junio su autor acaba de sacar la vér. 1.3.

**Akka** : http://akka.io
La filosofía basada en Actores me parece de lo más acertada a la hora de lidiar con threads y Akka tiene una implementación muy buena, te abstrae la parte difícil para que te concentres en la solución a tu problema.

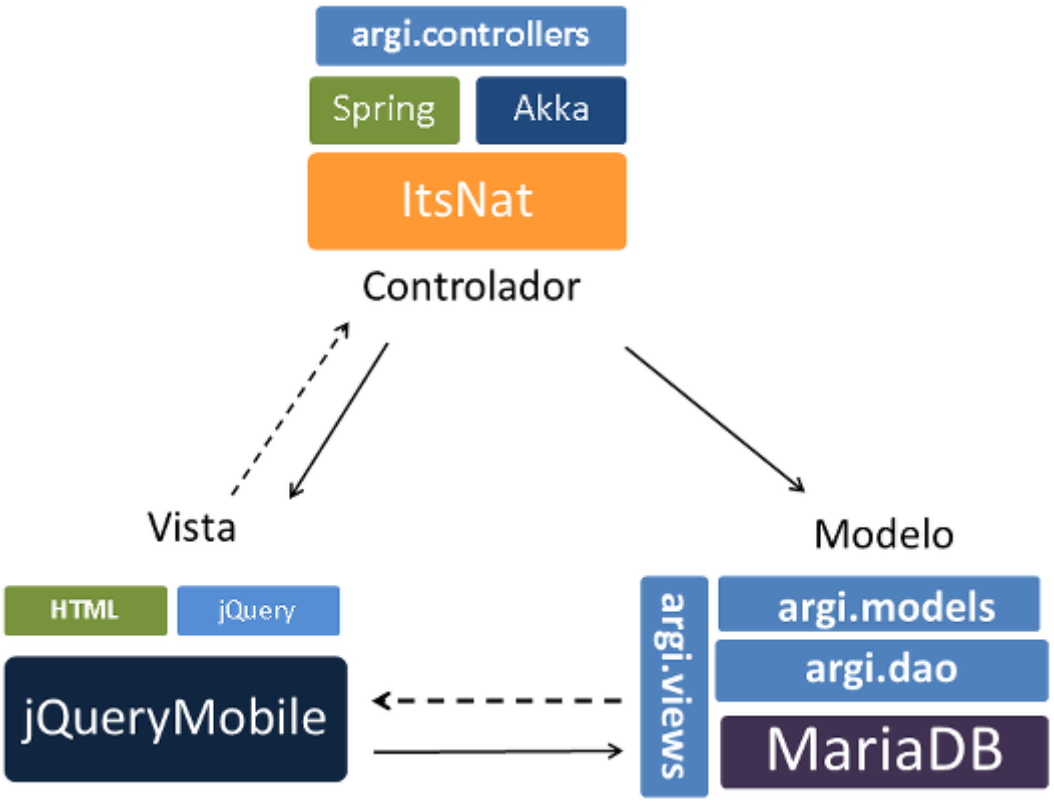
**Spring Framework 3.2**:
http://static.springsource.org/spring/docs/3.2.x/spring-framework-reference/html
Conocido ampliamente como un contenedor para la inversión de control y la inyección de dependencias, en mi proyecto no lo estoy utilizando con esa finalidad pues no uso

la inversión de control y hago pocas inyecciones de dependencia, pero al ser un framework modular te permite utilizar las partes que necesites. Así que utilizo su gestión de persistencia para Jdbc mediante JdbcTemplate, me ha parecido muy buena porque es una capa que te quita lo redundante en este tipo de codificaciones. Aunque la gestión de mail con JavaMail directamente no es complicada este framework te la facilita todavía más. El soporte para traducciones i18n con ficheros properties y MessageSource es realmente genial. La facilidad de utilizar Log4J en toda tu aplicación.

**jQuery :** http://jquery.com
Se trata uno de los Frameworks JavaScript más conocidos y casi un estándar para lidiar con JavaScript en la parte del navegador.

**jQuerymobile**: http://jquerymobile.com
Con jQuery como base este Framework es una extensión muy buena para desarrollar la capa de presentación en dispositivos móviles.

Por tanto la arquitectura final de la aplicación es la siguiente:



# 4. Implementación

## 4.1 Entorno de desarrollo

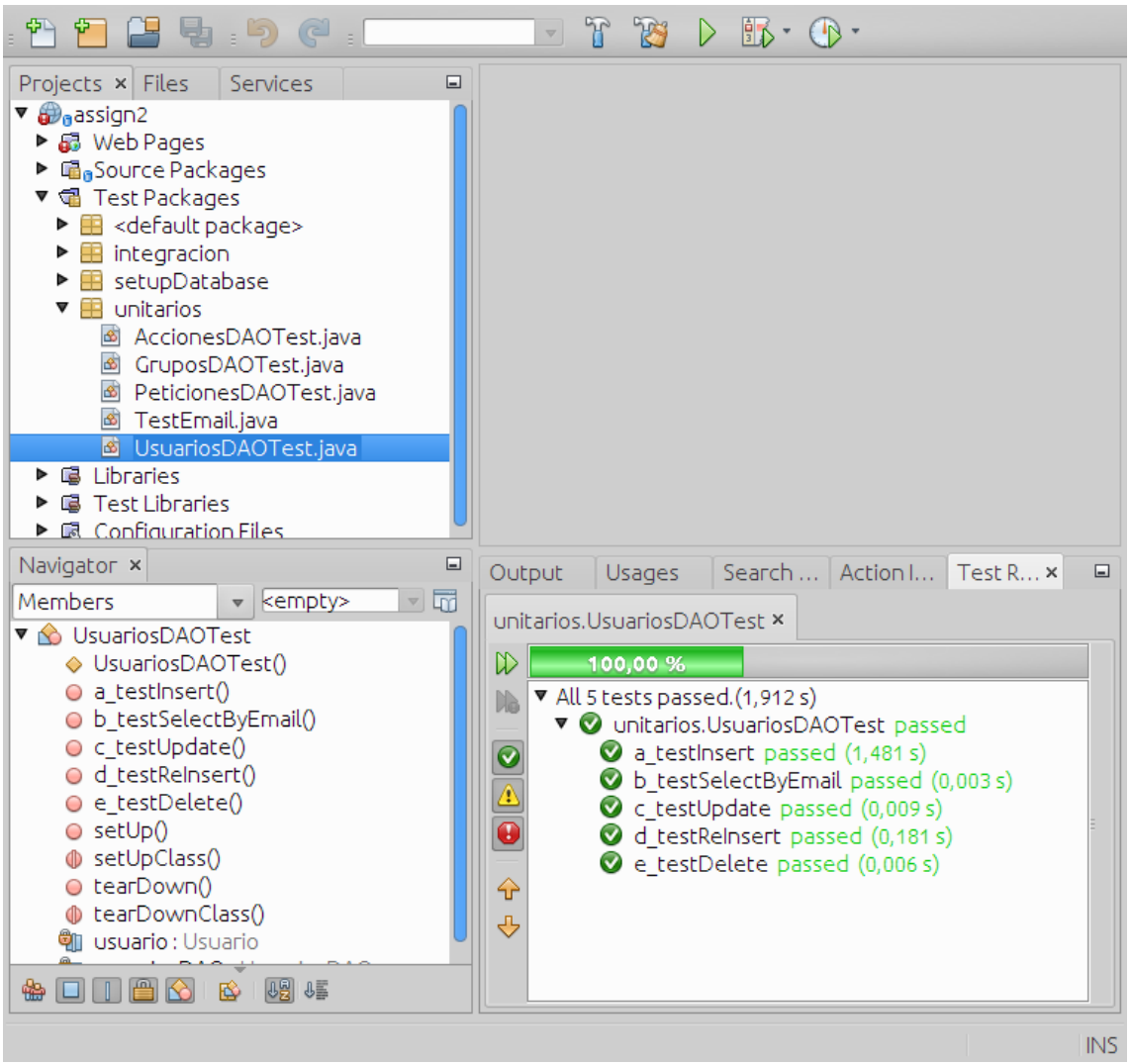
El desarrollo lo he llevado a cabo en mi portátil marca Sony VAIO. Incorpora un procesador Intel Core i5, 8Gb de ram y disco SSD 512Gb OCZ Vertex 4.

- Sistema operativo: Ubuntu 12.04 64 bits, con interfaz Unity.
- Versión java: Oracle JDK 7u17
- IDE de desarrollo: Oracle Netbeans 7.3
- Servidor de aplicaciones: Apache Tomcat 7.0.34 (integrado en Netbeans)
- Motor base de datos: MariaDB 5.5.30
- Gestor base de datos: MySQL Workbench 5.2.47
- Navegadores y extensiones:
    - Google Chrome 26.0
        - Netbeans Connector 1.0
        - responsiView 1.2.2
    - Firefox 20.0
        - Web Developer 1.2.2

## 4.2 Desarrollo enfocado en Pruebas

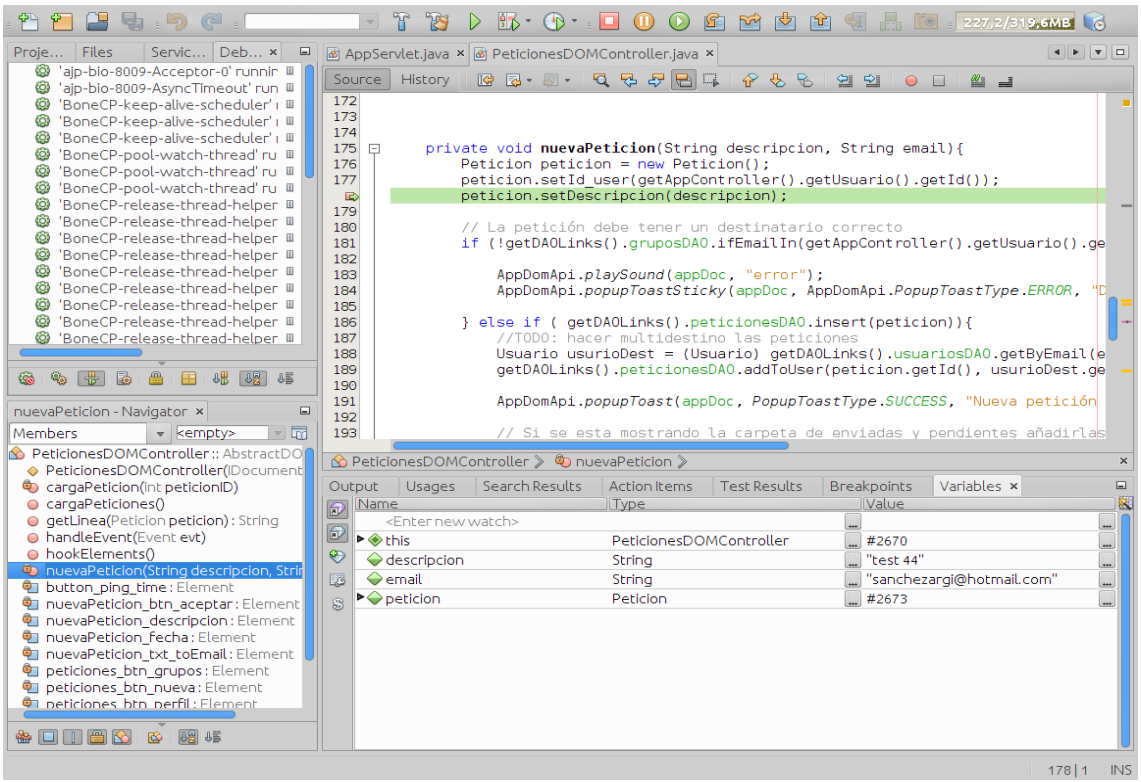
Desde hace tiempo que se enfatiza mucho a los programadores a hacer batería de pruebas a sus programas durante la codificación, hasta el punto que muchos consideran que debe iniciarse la programación por las pruebas. Para mí era un poco chocante porque me preguntaba: ¿pero cómo voy a hacer las pruebas antes que el programa? ¿Qué pruebo?
Pues bien tengo que decir que llevan razón. Por ejemplo antes de desarrollar la pantalla de registro de usuario he comenzado creando el test unitario *UsuariosDAOTest.class* y he avanzado de forma paralela al desarrollo de la clase *UsuariosDAO.class*. Esta forma de proceder me ha ayudado a centrarme primero en los

requerimientos y me ha agilizado la depuración. Es mucho mejor que si tratas de implementar directamente la pantalla de registro a partir de prueba y error.

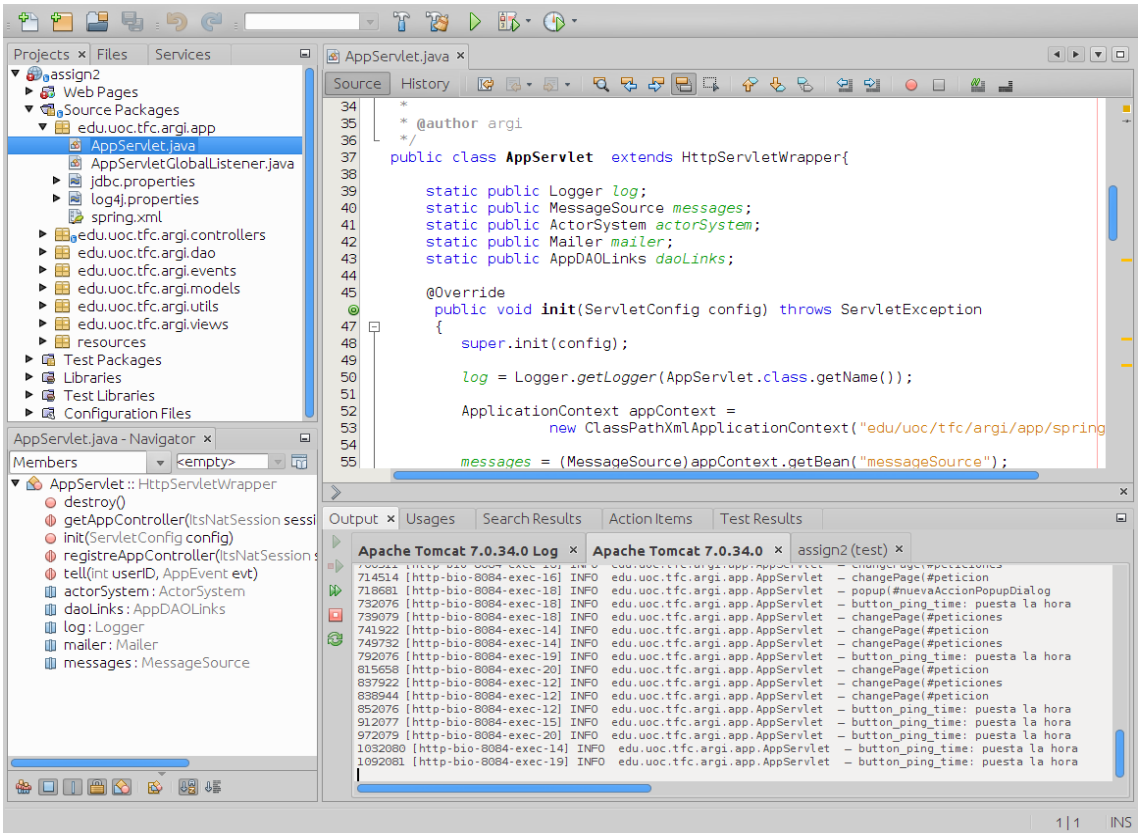



## 4.3 Codificación: depuración activa y pasiva

Queda claro que las pruebas son muy necesarias pero hay ocasiones en que no son suficientes. Cuando existe algún error en la lógica de un procedimiento no queda otra que *tracear* de forma activa línea por línea hasta darnos cuenta a donde nos estamos equivocando.



Utilizar un logger como depuración pasiva también es de gran utilidad pues nos va informando de los puntos de control que previamente hayamos creado y podemos comparar por donde va pasando y si se corresponde o no con lo que esperamos ver en la interfaz de usuario.



# 5. Valoración económica

Trataré de dar una valoración económica como si se tratase de un proyecto real. En primer lugar tengo claro que a día de hoy ninguna empresa te acepta un presupuesto sin cerrar el apartado económico por anticipado, por lo que si te pasas en horas es problema tuyo.

Así que ateniéndome a las horas presentadas en el Plan de Trabajo de las 207 horas totales considero que solo 150 son realmente facturables

| Descripción | Horas | Precio | Importe |
|---|---|---|---|
| Análisis | 20 | 45 | 900 |
| Diseño | 30 | 55 | 1650 |
| Implementación | 80 | 40 | 3200 |
| Testeo | 20 | 35 | 700 |
| | | | |
| | | | |
| | | | |
| **Total** | | | **6450 €** |

Parece un precio anticrisis pero yo ya firmaba por una mensualidad así.

# 6. Conclusiones

El producto final ha sido una versión bastante recortada comparada con todas las cosas que imaginé en un principio. He invertido bastante del tiempo de programación diseñando el cómo hacer las cosas, refactorizando una y otra vez, y creo que más que un producto final ha quedado una base, más o menos buena, que permitirá continuar un desarrollo rápido de nuevas pantallas y funcionalidades.

Me preocupa un fallo aleatorio de ConcurrentModificationException y también el no haber podido hacer un buen JavaDoc, pero he ido estableciendo prioridades y el tiempo se me ha queda muy corto.

Lo que menos me ha gustado es documentar, sinceramente lo odio, también tengo que decir que no estoy muy ducho con el office y que me he pelado mucho con él. Reconozco que es necesario pero para mí ha sido una pesadilla, he necesito invertir más horas con el Microsoft Office que con el Netbeans.

En general ha sido una buena experiencia a pesar de ciertos momentos exasperantes, pues detalles que previamente parecen una tontería me ha llevado más tiempo del previsto, en contraposición cosas a priori más complejas he podido solucionarlas en tiempos mucho más razonables.

# Glosario

**COMET:** Aplicación web que mantiene abierta una petición HTTP y que permite al servidor enviar datos al navegador, sin que este lo solicite.

**DOM:** *Document Object Model*. Proporcina un conjunto estándar de objetos para representar documentos HTML.

**Listener:** En programación orientada a eventos el Listener es el objeto capaz de responder a un determinado evento.

**Login:** Proceso mediante el cual se controla el acceso individual a un sistema.

**POJO:** *Plain Old Java Object*. Clase Java simple, con sus getters y setters, que no extiende a ninguna otra.

**TAG:** Etiquetas HTML que comienzan con < y terminan con /> y que los navegadores leen e interpretan para dar forma las páginas web.

**Websocket:** Tecnología que proporciona una canal de comunicación bidireccional y full-duplex sobre un único socket TCP.

# Bibliografía

La verdad es que aparte de la documentación de los frameworks no he utilizado ninguna bibliografía específica. Eso sí me he apoyado en muchas consultas cortas con google, rae, wordreference y la Wikipedia.

Manual Spring Framework 3.2.x
http://static.springsource.org/spring/docs/3.2.x/spring-framework-reference/pdf/spring-framework-reference.pdf

Manual ItsNat Framework 1.3
http://itsnat.sourceforge.net/php/support/docs/manual.pdf

Manual Akka 2.0.5
http://doc.akka.io/docs/akka/2.0.5/Akka.pdf

BoneCP
http://jolbox.com/configuration-spring.html

Wikipedia
http://es.wikipedia.org/wiki/Wikipedia:Portada

Real Academia Española
http://www.rae.es/rae.html

Wordreference.com
http://www.wordreference.com