Trabajo de Final de Carrera

Desarrollo de una aplicación web multiplataforma según el patrón Modelo Vista Controlador

Memoria

Jorge Puig de la Bellacasa Alcaide
Ingeniería Técnica en Informática de Sistemas
Universitat Oberta de Catalunya

Consultor: Jordi Ceballos Villach

5 de Enero de 2014

A Margarita, mi madre, por su apoyo incondicional e imprescindible.
A Marta, mi vida, por apoyarme todo este tiempo sin quejas.
A Carlos y Marta, mi padre y mi hermana, por creer en mí contra todo pronóstico.
Y a Jorge, por su inesperada, sacrificada y esencial guía, no podría haberlo hecho sin ti.

Gracias a todos de corazón, os quiero.

Índice de contenidos

1 Definición general del proyecto

1.1 Objetivo del proyecto

El fin principal de este proyecto es poner en práctica los conocimientos adquiridos durante la carrera mediante el desarrollo de una aplicación completamente funcional.

Para ello se proponen los siguientes objetivos:

- Poner en práctica las técnicas de desarrollo del software estudiadas durante la carrera en el desarrollo de una aplicación, siguiendo el ciclo de vida desde el análisis previo hasta la finalización.
- Introducirse en las tecnologías multiplataforma, con un énfasis especial en su uso en dispositivos móviles.
- Profundizar en el diseño y administración de bases de datos relacionales.
- Poner en práctica la comunicación entre elementos de las aplicaciones a través de redes informáticas públicas y privadas.
- Introducirse en el uso del patrón *MVC* (Modelo Vista Controlador).

1.2 Descripción del proyecto

El proyecto tiene por objeto realizar el análisis, diseño e implementación de un recetario para múltiples dispositivos, en el que los usuarios puedan introducir, buscar y comentar recetas de cocina. Para ello se desarrollará un sistema cliente/servidor, utilizando el patrón *MVC* (Modelo Vista Controlador).
El cliente accederá al sistema de gestión de bases de datos a través de una interfaz de usuario, con una presentación intuitiva y sencilla proporcionada por la vista mediante acceso *HTML*.

1.3 Listado de funcionalidades principales

- Permitir el registro e identificación de los usuarios
- Almacenar en el servidor y acceder a recetas de platos desde dispositivos cliente.
- Tomar fotografías de los platos desde el dispositivo móvil y compartirlas con otros usuarios de la aplicación.
- Introducir comentarios en las recetas.
- Permitir la gestión de las recetas, fotografías y comentarios propios.
- Permitir a usuarios administradores la gestión de usuarios y moderación de comentarios y fotografías.

1.4 El diseño arquitectónico

Para diseñar un proyecto como este es indispensable ser consciente del punto de partida: la tecnología *HTTP* no fue diseñada para la implementación de aplicaciones complejas

donde los usuarios interaccionan con el sistema para crear su propio contenido.

En sus inicios la *web* no dejaba de ser principalmente un medio de acceso a contenidos estáticos, sólo alterables por sus creadores en un entorno confinado y fácilmente controlable: el usuario solicita un artículo almacenado en un servidor y lo recibe, y este artículo es el mismo para cualquier usuario que lo solicite.

Sin embargo, en la actualidad se espera de las aplicaciones que permitan a los usuarios interactuar con el sistema y crear contenidos que se servirán a otros usuarios. Y para ello resulta esencial acotar eficientemente el acceso de los usuarios a las partes vitales del sistema al mismo tiempo que se les permite acceder y modificar los contenidos del mismo.

Por lo tanto, es evidente que separar la vista del modelo de datos, e implementar un módulo que se encargue de facilitar y controlar la comunicación entre ambos es una excelente idea. Esto es el patrón Modelo Vista Controlador.

Gracias a este tipo de implementación podemos delimitar al detalle las funcionalidades que están al alcance de los usuarios, al mismo tiempo que se separa la aplicación en módulos independientes, fácilmente mantenibles y ampliables.

En los siguientes sub apartados se analizan las tres partes esenciales de este patrón de diseño, y las conclusiones del autor derivadas de su implementación.

1.4.1 El modelo

Las bases de datos relacionales son un medio excelente para almacenar los datos de una aplicación de estas características, pero implementar la lógica de negocio directamente sobre la base de datos es una tarea muy complicada y muy propensa a errores y a fallos de seguridad. Por no hablar de la dificultad de mantener y ampliar un sistema desarrollado de esta manera.

Por lo tanto es lógico delegar el tratamiento de los datos en un módulo implementado en un lenguaje de alto nivel, más apropiado para esta tarea, y limitar la función del sistema de gestión de bases de datos al almacenamiento y recuperación de los datos exclusivamente.

Ese es el objetivo del patrón del diseño *Data Access Object*, que propone implementar un módulo independiente cuyo único objetivo sea realizar la traducción de las entradas de la base de datos en objetos simples del lenguaje de alto nivel, con los que éste pueda trabajar cómodamente.

Por encima de esta capa se encuentra la lógica de negocio, que trata con esos objetos simples según los requerimientos de la aplicación, y limita el acceso y la interacción con esos datos al nivel que se desee. Definiendo por tanto qué se puede hacer y de qué manera.

Todo esto hace que el diseño del modelo sea comprensible y modular, y sus módulos fácilmente reutilizables y ampliables.

1.4.2 El controlador

Una vez definido el modelo, es necesario diseñar la interacción del mismo con la vista, y para ello se desarrolla el controlador.

Sus funciones serán todas aquellas derivadas de las necesidades de tratamiento de esta interacción: realizar la traducción entre el lenguaje que implementa la vista y el lenguaje de alto nivel que implementa el modelo, gestión de la sesión y de los permisos de acceso, validación de los datos y seguridad del sistema.

Hay que tener en cuenta que la tecnología *HTTP* solo trabaja con cadenas de texto, mientras que los lenguajes de alto nivel tratan con objetos complejos, y por lo tanto poblar los objetos del modelo con los datos obtenidos de la vista y viceversa no es una tarea trivial. Afortunadamente las librerías que siguen el patrón *MVC* tienen este problema muy en cuenta y proporcionan herramientas que realizan la transformación de los datos de manera transparente al programador.

En realidad, todas las funciones que realiza el controlador están contempladas en estas librerías, y su uso facilita enormemente la tarea.

1.4.3 La vista

Siguiendo con el modelo *MVC*, la vista tiene como tarea recoger los datos y solicitudes del usuario y presentarle los resultados correspondientes de una manera funcional y visualmente atractiva. De esta forma se evita el problema de implementar lógica de negocio en *JavaScript*, directamente en la vista, lo que permitiría que el usuario modificase el código a su antojo, creando un agujero de seguridad de gran importancia.

Sin embargo, *JavaScript* y sus librerías (como *jQuery*) permiten desarrollar una vista interactiva que presente los datos de manera dinámica y, gracias al uso de *CSS*, también atractiva.

Las librerías como *jQuery Mobile* además simplifican esta tarea hasta el punto de poder implementar una vista con un aspecto y funcionalidad profesionales sin necesidad de tener amplios conocimientos de *JavaScript* y *jQuery*. Resulta sencillo añadir menús desplegables, formularios interactivos, barras de herramientas y animaciones atractivas, entre otras cosas.

1.5 Tecnologías escogidas

Como sistema de gestión de bases de datos se ha escogido *MySQL* por su extensa implantación en el ámbito profesional, siendo por tanto una excelente adición a los conocimientos de este estudiante.

Para la implementación del acceso a datos, el modelo y el controlador se ha escogido el lenguaje *Java2EE* debido a la familiaridad adquirida durante la carrera con este lenguaje. Con este objetivo se seguirá un patrón *DAO* (*Data Access Object*), que permitirá el cambio de sistema de gestión de bases de datos con el mínimo esfuerzo posible, si así se desea en un futuro. Esta capa junto con el sistema de gestión de bases de datos formará el modelo.

El controlador se implementará mediante la librería de *Java Struts2*, orientada

específicamente al patrón *MVC*. Esta capa será la encargada de permitir la comunicación entre la vista y el modelo y de encapsular las funciones inherentes a esta comunicación: transferencia y validación de datos, identificación de usuarios y seguridad.

Para implementar la vista se utilizará *HTML5* junto con *CSS3* y *JavaScript* con el *framework jQuery Mobile*, permitiendo así el acceso a la aplicación desde multitud de sistemas con diferentes configuraciones de pantalla.

Finalmente, utilizaremos las librerías de Java: *ESAPI*, para evitar ataques de inyección de código y *jasypt* para encriptar las contraseñas.

2 Calendario del proyecto

Para la realización del proyecto se seguirá el ciclo de vida clásico, adaptado a las entregas marcadas por el consultor, que se detallan a continuación.

2.1 Entregas establecidas

Fecha de entrega	Tarea	Descripción de la entrega
1 de Octubre	PEC1	Plan de trabajo
29 de Octubre	PEC2	Análisis funcional, diseño técnico y prototipo de la vista
26 de Noviembre	Beta	Implementación beta e instalables
10 de Diciembre	PEC3	Implementación final e instalables
8 de Enero	Final	Memoria y video de presentación del proyecto

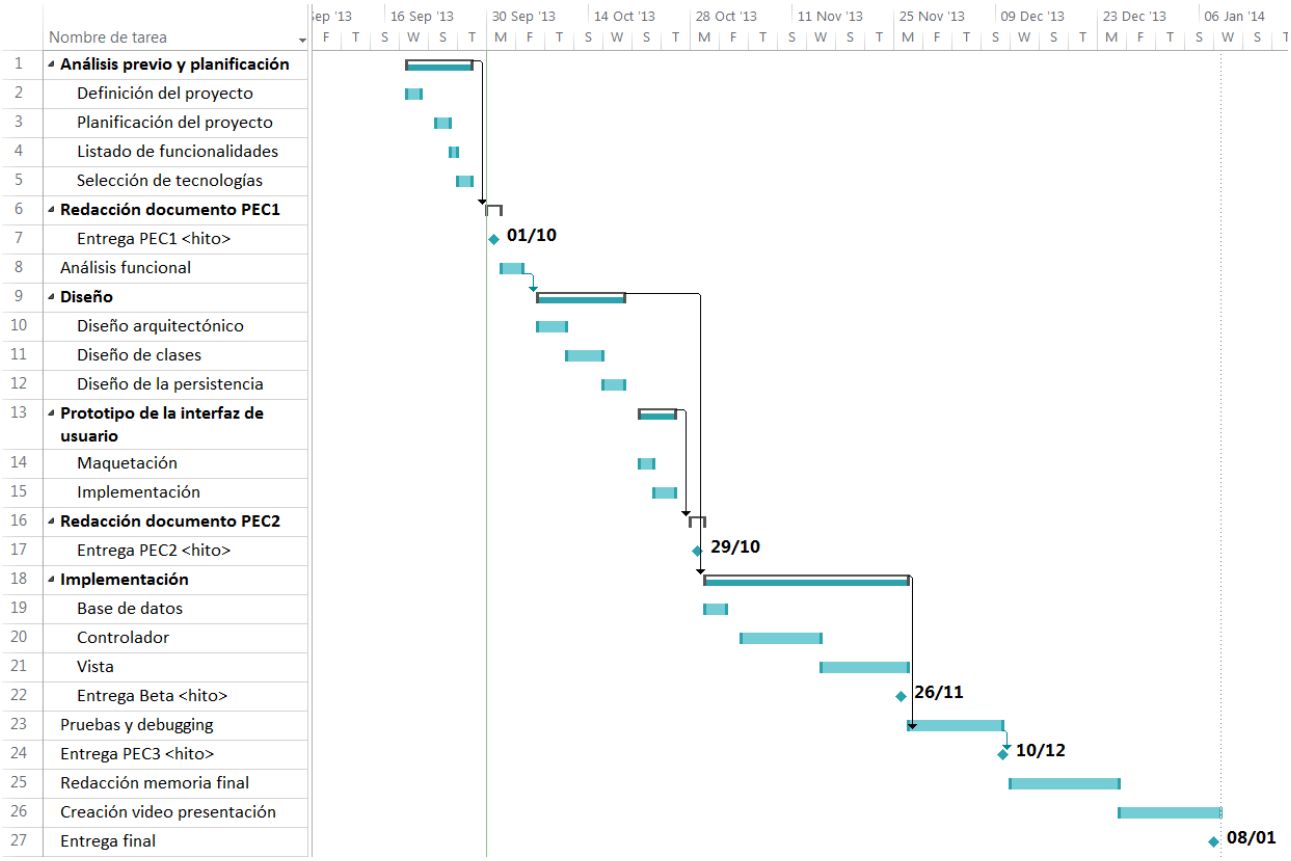
2.2 Planificación temporal

El proyecto comienza el día 18 de Septiembre y debe ser entregado el día 8 de Enero. Por lo tanto, teniendo en cuenta que se trabajará de lunes a viernes, se dispone en total de 81 días para su realización.

A continuación se muestra la planificación detallada de las tareas necesarias para la consecución de los objetivos del proyecto, así como las entregas establecidas por el consultor, señaladas como hitos.

	Nombre de tarea	Duration	Start	Finish
1	**Análisis previo y planificación**	**7 days**	**Thu 19/09/13**	**Fri 27/09/13**
2	Definición del proyecto	2 days	Thu 19/09/13	Fri 20/09/13
3	Planificación del proyecto	2 days	Mon 23/09/13	Tue 24/09/13
4	Listado de funcionalidades	1 day	Wed 25/09/13	Wed 25/09/13
5	Selección de tecnologías	2 days	Thu 26/09/13	Fri 27/09/13
6	**Redacción documento PEC1**	**2 days**	**Mon 30/09/13**	**Tue 01/10/13**
7	Entrega PEC1 <hito>	0 days	Tue 01/10/13	Tue 01/10/13
8	Análisis funcional	3 days	Wed 02/10/13	Fri 04/10/13
9	**Diseño**	**10 days**	**Mon 07/10/13**	**Fri 18/10/13**
10	Diseño arquitectónico	4 days	Mon 07/10/13	Thu 10/10/13
11	Diseño de clases	3 days	Fri 11/10/13	Tue 15/10/13
12	Diseño de la persistencia	3 days	Wed 16/10/13	Fri 18/10/13
13	**Prototipo de la interfaz de usuario**	**5 days**	**Mon 21/10/13**	**Fri 25/10/13**
14	Maquetación	2 days	Mon 21/10/13	Tue 22/10/13
15	Implementación	3 days	Wed 23/10/13	Fri 25/10/13
16	**Redacción documento PEC2**	**2 days**	**Mon 28/10/13**	**Tue 29/10/13**
17	Entrega PEC2 <hito>	0 days	Tue 29/10/13	Tue 29/10/13
18	**Implementación**	**20 days**	**Wed 30/10/13**	**Tue 26/11/13**
19	Base de datos	3 days	Wed 30/10/13	Fri 01/11/13
20	Controlador	9 days	Mon 04/11/13	Thu 14/11/13
21	Vista	8 days	Fri 15/11/13	Tue 26/11/13
22	Entrega Beta <hito>	0 days	Tue 26/11/13	Tue 26/11/13
23	Pruebas y debugging	9 days	Wed 27/11/13	Mon 09/12/13
24	Entrega PEC3 <hito>	0 days	Tue 10/12/13	Tue 10/12/13
25	Redacción memoria final	11 days	Wed 11/12/13	Wed 25/12/13
26	Creación video presentación	10 days	Thu 26/12/13	Wed 08/01/14
27	Entrega final	0 days	Wed 08/01/14	Wed 08/01/14

Diagrama de Gantt asociado a la planificación:



2.3 Breve detalle de las tareas a realizar

nº tarea	Nombre de la tarea	Breve descripción de la tarea
1	**Análisis previo y planificación**	
2	-Definición del proyecto	Propuesta al consultor del proyecto a realizar
3	-Planificación del proyecto	Establecimiento de las tareas a realizar y su planificación temporal
4	-Listado de funcionalidades	Establecimiento de las funcionalidades básicas del proyecto
5	-Selección de tecnologías	Selección de las tecnologías a utilizar en función de las necesidades establecidas
6	**Redacción documento PEC1**	
7	-Entrega PEC1 <hito>	
8	**Análisis funcional**	Análisis de los usuarios a considerar, requisitos funcionales y no funcionales, y análisis del sistema (diagramas de casos de uso y su descripción textual)
9	**Diseño**	
10	-Diseño arquitectónico	Diseño de la arquitectura global
11	-Diseño de clases	Diseño del modelo y diseño estático de clases del modelo
12	-Diseño de la persistencia	Diseño de la persistencia, modelo relacional de la base de datos y diagrama del modelo relacional
13	**Prototipo de la interfaz de usuario**	
14	-Maquetación	Diseño previo en papel de la interfaz de usuario
15	-Implementación	Realización de la maqueta del diseño obtenido
16	**Redacción documento PEC2**	
17	-Entrega PEC2 <hito>	
18	**Implementación**	
19	Base de datos	Implementación de la base de datos
20	Controlador	Implementación del controlador
21	Vista	Implementación de la vista
22	Entrega Beta <hito>	
23	Pruebas y *debugging*	Pruebas en busca de fallos con diferentes dispositivos y corrección de los mismos
24	Entrega PEC3 <hito>	
25	Redacción memoria final	
26	Creación video presentación	
27	Entrega final	

3 Análisis de riesgos

Las características del proyecto incluyen una plantilla de una sola persona, y una preparación escasa respecto a la mayoría de las tecnologías a utilizar. Teniendo esto en cuenta se han previsto los siguientes riesgos, ordenados por impacto en orden descendente:

Riesgo	Descripción	Estimación de la probabilidad de aparición	Impacto	Acciones preventivas
Incapacidad de alcanzar los objetivos de la planificación temporal	Retrasos en la marcha del proyecto debidos a una mala planificación o a factores externos como enfermedad o similares.	Media	Crítico	-Adelantar trabajo siempre que sea posible, sin confiar excesivamente en la estimación de tiempo para completar las tareas restantes. -Reprogramar las tareas restantes en caso de presentarse retrasos, o en caso de disponer de tiempo extra si hay tareas que se terminen antes del tiempo estimado.
Pérdida de datos	Pérdida del trabajo realizado debido a fallos en los discos duros, virus o problemas similares.	Baja	Crítico	-Establecer un sistema de copias de seguridad con volcado de datos diario, tanto en discos duros secundarios como en servidores externos. -Disponer de un sistema antivirus actualizado.
Falta de conocimientos	Retrasos debidos a la inexperiencia con las tecnologías escogidas.	Media	Alto	-Disponer de documentación adecuada sobre las tecnologías escogidas. -Realizar un esfuerzo extraordinario en estudiar y familiarizarse con las tecnologías a utilizar. -Buscar ayuda externa para consultas (consultor de la UOC, foros especializados, amigos expertos…).
Fallo del sistema (hardware)	Fallo catastrófico del ordenador personal utilizado para el proyecto, imposibilitando su uso hasta encontrar el problema y sustituir el/los componente/s afectado/s.	Baja	Medio	-Disponer de un ordenador secundario con el software del proyecto instalado y listo para usar.

Riesgo	Descripción	Estimación de la probabilidad de aparición	Impacto	Acciones preventivas
Fallo prolongado en el tiempo del servicio de conexión a Internet	Avería o fallo en la infraestructura del proveedor de Internet que impida el acceso a recursos externos durante un período prolongado de tiempo.	Baja	Bajo	-Disponer de una conexión secundaria, como el acceso compartido a la red 3G desde el teléfono móvil.

4 Requisitos iniciales

4.1 Usuarios a considerar

Los tipos de usuarios que deberán ser tenidos en cuenta en el sistema son los siguientes:

- Usuario Anónimo: únicamente tendrá acceso a las funcionalidades de entrada al sistema, así como al registro en el mismo.

- Usuario Registrado: podrá consultar el catálogo de recetas así como añadir recetas al mismo e introducir comentarios en las mismas, además de cerrar su sesión en el sistema y cambiar su contraseña y su dirección de correo electrónico.

- Usuario Administrador: podrá realizar las mismas acciones que el usuario registrado, así como gestionar el sistema eliminando recetas, comentarios, fotografías y usuarios registrados. Además podrá promocionar nuevos usuarios a administradores para ayudar en las labores de gestión del sistema.

4.2 Requisitos funcionales

Los requisitos se han agrupado en 5 bloques principales:

- Funcionalidades de seguridad: donde se recogen las necesidades relativas al mantenimiento de usuarios y control de acceso al sistema.
- Funcionalidades del recetario: donde se reflejan los requisitos relativos a la consulta y creación de nuevos contenidos en el recetario.
- Funcionalidades de la gestión de usuarios: donde se recogen las necesidades relativas al control de usuarios por parte de los usuarios administradores del sistema.
- Funcionalidades de la gestión del recetario: donde se reflejan los requisitos relativos a la moderación y mantenimiento del recetario por parte de los usuarios administradores.

4.2.1 Funcionalidades de seguridad

Iniciar sesión
Deberá permitir a los usuarios anónimos identificarse en el sistema para así poder acceder al resto de funcionalidades del mismo. Para identificarse, el usuario anónimo deberá proporcionar su identificador de usuario y su contraseña.

Finalizar sesión
Deberá permitir a un usuario previamente identificado en el sistema cerrar la sesión actual.

Cambiar contraseña
Permitirá cambiar la contraseña del usuario previamente identificado. Para ello el usuario deberá proporcionar su contraseña antigua y la nueva contraseña por duplicado.

Alta de usuario
Deberá permitir el registro en el sistema de un usuario anónimo (no identificado en el sistema).

4.2.2 Funcionalidades del recetario

Añadir receta
Permite añadir una nueva receta al sistema.

Buscar receta
Permite buscar recetas por nombre de receta o bien por identificador de usuario.

Visualizar receta
Permite a un usuario visualizar los detalles de una receta, junto con los comentarios que puedan existir en la misma.

Modificar receta
Permite a un usuario modificar una receta propia o bien eliminarla del sistema.

Añadir comentario
Permite añadir un nuevo comentario a una receta existente.

Eliminar comentario
Permite a un usuario eliminar un comentario propio en una receta.

4.2.3 Funcionalidades de moderación del sistema

Buscar receta
Permite buscar recetas por nombre (o bien mostrar todas las recetas) para su posterior gestión.

Eliminar receta
Permite que el usuario administrador elimine una receta junto con sus fotografías y comentarios.

Gestionar comentarios
Permite que un usuario administrador elimine comentarios de una receta.

4.2.4 Funcionalidades de administración de usuarios

Buscar usuarios
Permite que un usuario administrador busque un usuario en el sistema, ya sea por identificador de usuario o bien acceder a un listado de todos los usuarios existentes.

Eliminar usuario
Permite que un usuario administrador elimine un usuario del sistema junto con todas las aportaciones de ese usuario (recetas y comentarios).

Promocionar usuario
Permite que un usuario administrador promocione a un usuario registrado a usuario administrador.

4.3 Requisitos no funcionales

4.3.1 Requisitos de interfaz

El sistema contará con un único interfaz para todos los usuarios, destinado a abarcar un amplio número de dispositivos: ordenadores personales, *Tablet PC* y *Smartphones*. Sin embargo, el interfaz se diseñará con prioridad hacia los dispositivos portátiles teniendo en cuenta sus limitaciones.

4.3.2 Requisitos de información

La información que deberá considerarse es:

Recetas:

- Referencia de la receta: un código único para cada receta asignado automáticamente por la base de datos en el momento de añadir la misma.
- Nombre: el nombre de la receta.
- Comensales: el número de comensales.
- Método de preparación: un texto explicando la forma de preparar la receta.
- Ingredientes: un texto con el listado de ingredientes necesario para cocinar la receta.
- Fotografía: una fotografía de la receta preparada (opcional).

Usuarios:

- Identificador de usuario: un nombre elegido por el usuario que deberá ser único en el sistema.
- Contraseña: la encriptación de la contraseña que el usuario elija.
- e-mail: dirección de correo electrónico del usuario.
- Tipo de usuario: una cadena que indique si el usuario es administrador o un usuario regular (*admin* o *user*).

Comentarios:

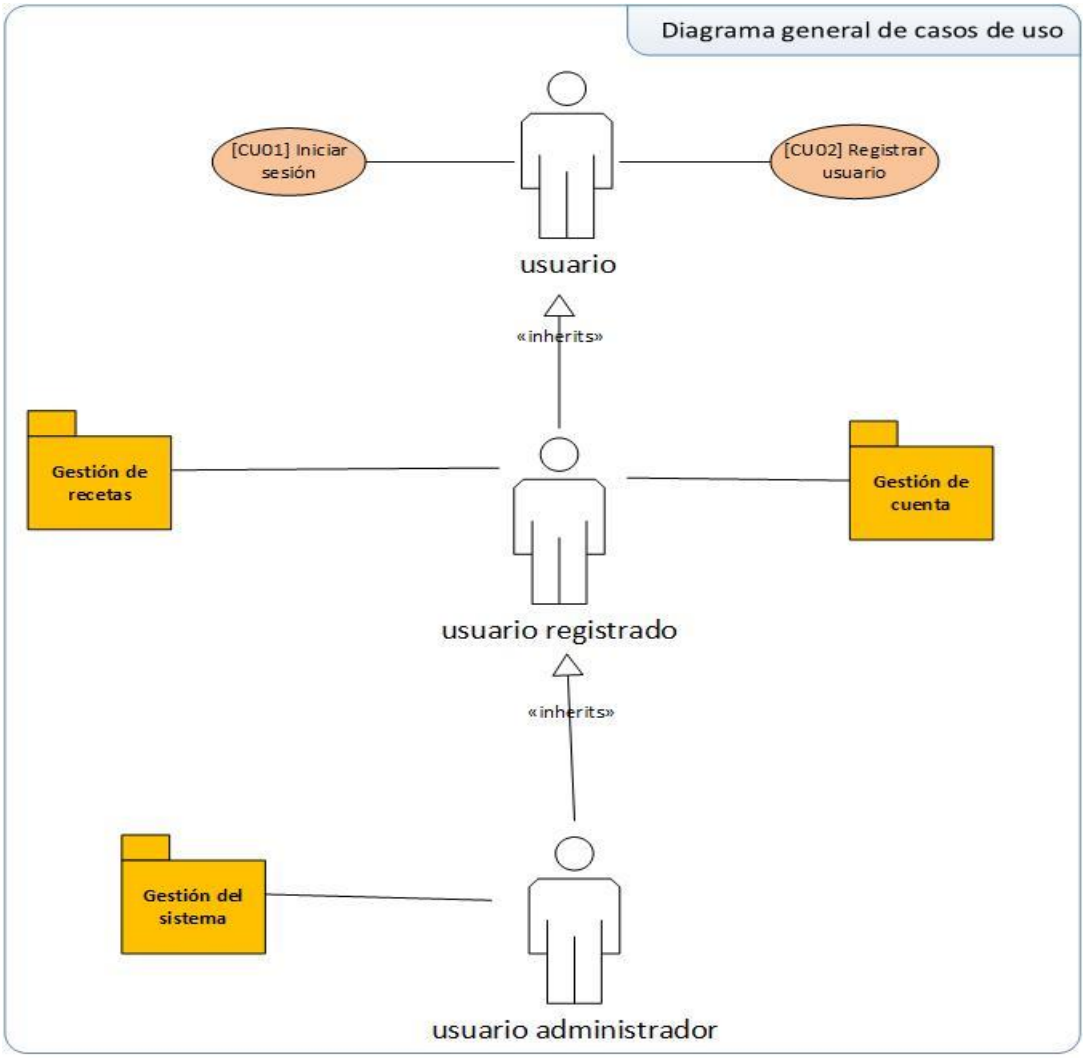
- Identificador de comentario: un identificador único asignado automáticamente por la base de datos en el momento de añadir el mismo.
- Identificador de usuario: el identificador único del usuario que introdujo el comentario.
- Identificador de receta: el identificador único de la receta en la que se añadió el comentario.

Si resulta necesario crear más entidades podrán ser diseñadas a conveniencia.

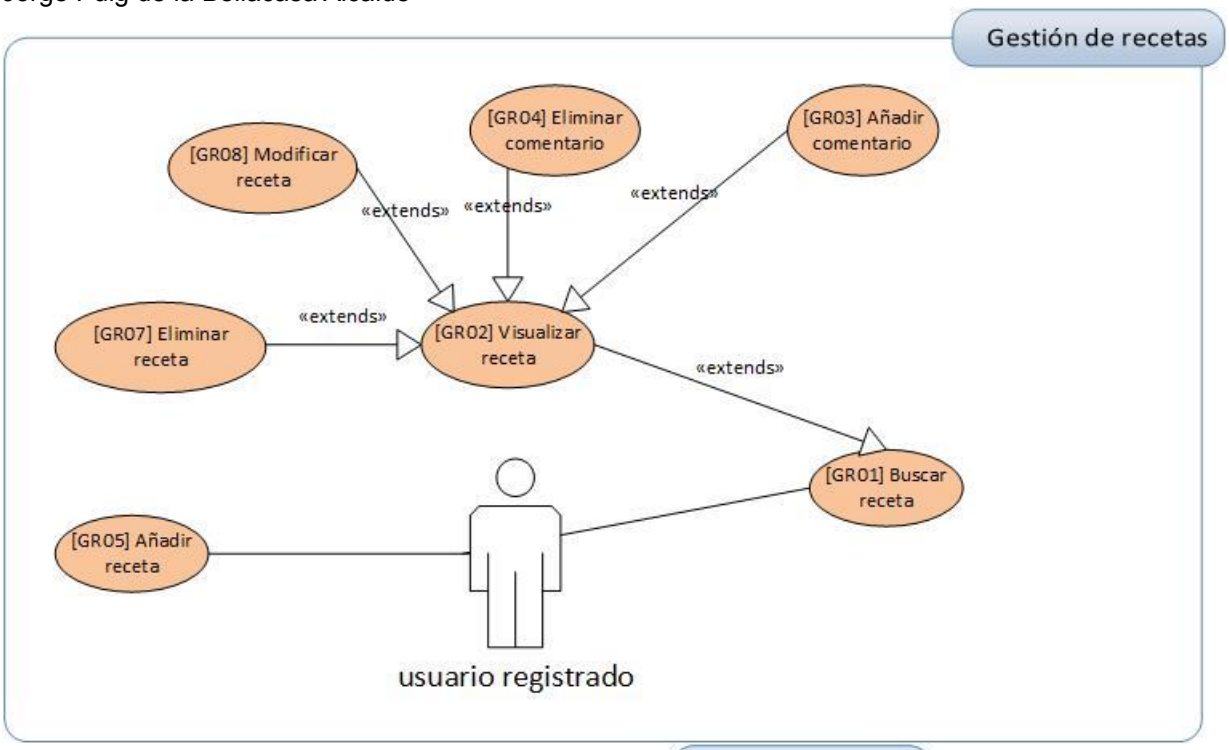
5 Análisis del sistema

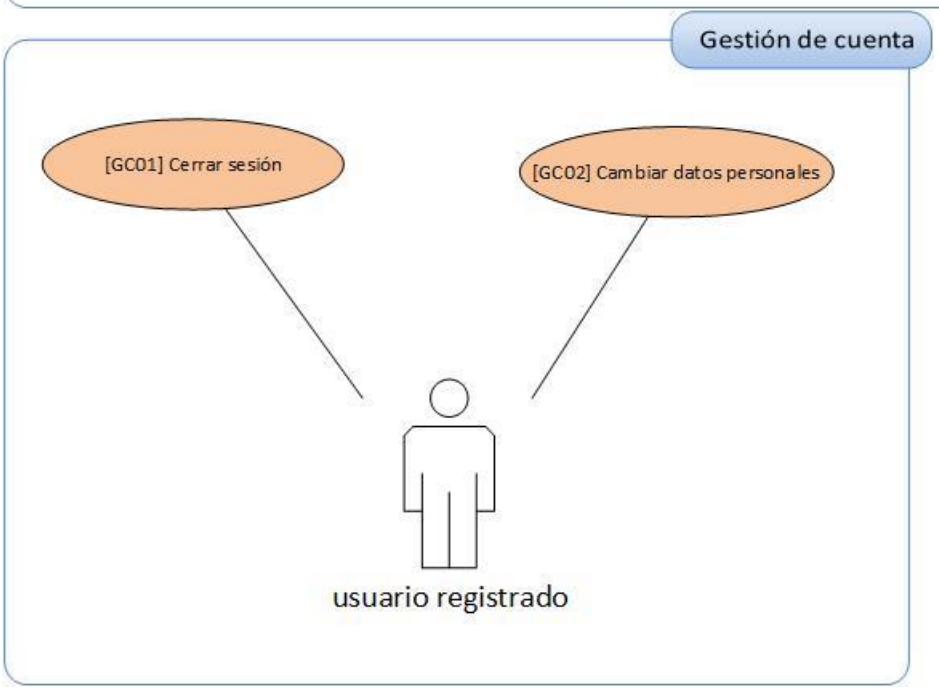
5.1 Diagrama de casos de uso

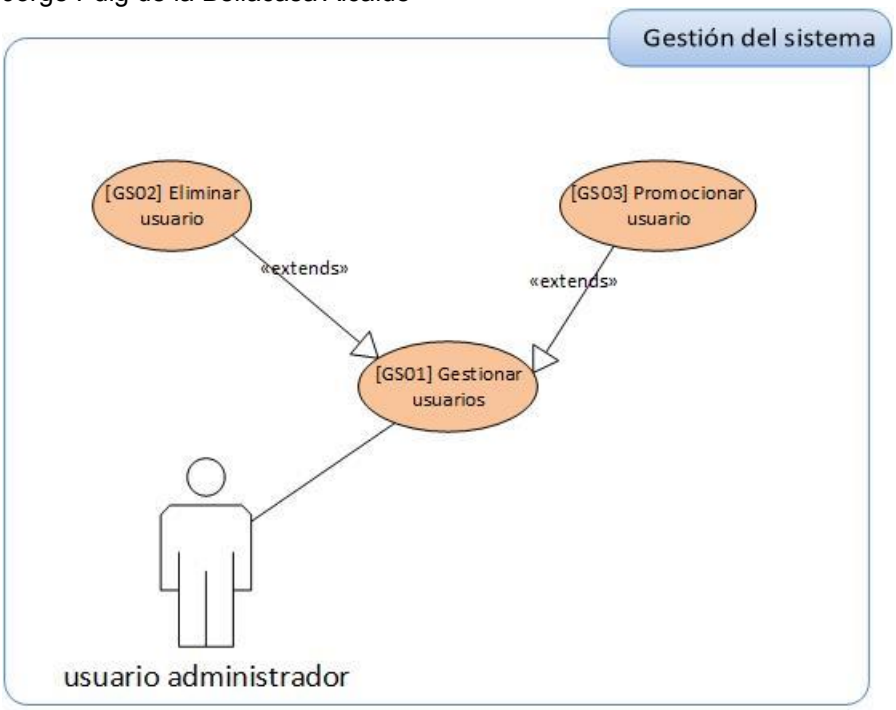
El siguiente diagrama recoge una vista global de los actores y casos de uso descritos en los requisitos funcionales. Se han agrupado la mayoría de casos de uso en grupos funcionales (carpetas en el diagrama) para evitar mostrar un esquema demasiado complicado. Además, estos grupos funcionales serán los mismos que se utilizarán en el diseño y programación del sistema, como se podrá apreciar más adelante en este documento.
Hay que tener en cuenta que los casos de uso CU01 y CU02 no pertenecen a un grupo funcional ya que tienen requisitos distintos entre sí y no tiene sentido crear un grupo funcional para un único caso de uso.



A continuación se muestran los diagramas de los diferentes grupos funcionales introducidos en la anterior figura:


Gestión de recetas
[GR08] Modificar receta
[GR04] Eliminar comentario
[GR03] Añadir comentario
«extends»
«extends»
«extends»
[GR07] Eliminar receta
«extends»
[GR02] Visualizar receta
«extends»
[GR01] Buscar receta
[GR05] Añadir receta
usuario registrado


Gestión de cuenta
[GC01] Cerrar sesión
[GC02] Cambiar datos personales
usuario registrado



5.2 Descripción textual de los casos de uso

En esta sección se pueden consultar las descripciones en detalle de los casos de uso mostrados en los diagramas anteriores.

5.2.1 CU01-Iniciar sesión

Identificador	CU01
Nombre	Iniciar sesión
Autor	Jorge Puig de la Bellacasa
Resumen	Este caso de uso muestra cómo un usuario se identifica en el sistema
Actor(es)	Usuario
Precondiciones	No hay ninguna sesión activa
Postcondiciones	El actor ha iniciado la sesión o bien ha finalizado la aplicación
Flujo normal	1. El caso de uso se inicia cuando el actor inicia la aplicación, o bien cuando se ha finalizado una sesión anterior. 2. El sistema solicita al actor que introduzca su identificador y su contraseña, o bien que se dé de alta en el sistema (caso de uso CU02). 3. El actor introduce los datos solicitados y valida el formulario. 4. Si los datos son correctos se inicia la sesión y se muestra el menú principal de la aplicación. El caso de uso finaliza.
Flujos alternativos	4a. En caso de que los datos suministrados no sean correctos, se muestra un error y el flujo vuelve al paso 2.
Inclusiones	Ninguna
Extensiones	Ninguna

5.2.2 CU02-Registrar usuario

Identificador	CU02
Nombre	Registrar usuario
Autor	Jorge Puig de la Bellacasa
Resumen	Este caso de uso muestra cómo un usuario se da de alta en el sistema
Actor(es)	Usuario
Precondiciones	No hay ninguna sesión activa
Postcondiciones	Ninguna
Flujo normal	1. El caso de uso se inicia cuando el actor inicia la aplicación, o bien cuando se ha finalizado una sesión anterior. 2. El sistema solicita al actor que introduzca su identificador y su contraseña o bien que se dé de alta en el sistema. 3. El actor pulsa el botón de alta en el sistema. 4. El sistema muestra un formulario solicitando el nuevo nombre de usuario, la contraseña y la dirección de correo electrónico por duplicado, y dos botones, uno para aceptar y otro para cancelar. 5. El actor introduce los datos requeridos y valida el formulario. 6. El sistema muestra un mensaje de confirmación e inicia una nueva sesión, mostrando el menú principal de la aplicación. El caso de uso finaliza.
Flujos alternativos	6a. En caso de que el identificador de usuario proporcionado ya exista en el sistema se informa al actor mediante un mensaje de error, y vuelve al paso 4. 6b. En caso de que algún dato proporcionado sea erróneo, se muestra un error informando al actor y vuelve al paso 4. Los errores posibles son: -la contraseña tiene menos de 6 caracteres o contiene caracteres distintos de números y letras. -La dirección de e-mail proporcionada no es una dirección de e-mail. -La contraseña o el e-mail proporcionados no coinciden en sus dos casillas correspondientes.
Inclusiones	Ninguna
Extensiones	Ninguna

5.2.3 GC01-Cerrar sesión

Identificador	GC01
Nombre	Cerrar sesión
Autor	Jorge Puig de la Bellacasa
Resumen	Este caso de uso muestra cómo un usuario finaliza la sesión en el sistema.
Actor(es)	Usuario registrado
Precondiciones	Existe una sesión activa

Postcondiciones	No existe ninguna sesión activa o bien continúa la sesión anterior
Flujo normal	1. El caso de uso se inicia cuando el actor pulsa el botón de finalizar sesión en el menú de configuración. 2. El sistema muestra una pantalla solicitando confirmación. 3. El actor confirma que quiere finalizar la sesión. 4. El sistema finaliza la sesión y muestra la pantalla de inicio de sesión (caso de uso CU01). El caso de uso finaliza.
Flujos alternativos	3a. Si el actor cancela (no confirma) la solicitud, el sistema muestra el menú principal de la aplicación y el caso de uso finaliza.
Inclusiones	Ninguna
Extensiones	Ninguna

5.2.4 GC02-Cambiar datos personales

Identificador	GC02
Nombre	Cambiar datos personales
Autor	Jorge Puig de la Bellacasa
Resumen	Este caso de uso muestra cómo un usuario cambia sus datos personales en el sistema (contraseña y/o dirección de correo electrónico).
Actor(es)	Usuario registrado
Precondiciones	Ninguna
Postcondiciones	Ninguna
Flujo normal	1. El caso de uso se inicia cuando el actor presiona el botón de configuración en el menú principal. 2. El sistema solicita al actor que introduzca su contraseña actual, así como la nueva contraseña y/o nuevo e-mail por duplicado. 3. El actor introduce los datos solicitados y valida el formulario. 4. Si los datos son correctos se cambian los datos del actor por los nuevos datos proporcionados y se muestra un mensaje informativo confirmando el cambio. El caso de uso finaliza.
Flujos alternativos	4a. En caso de que la contraseña actual no sea correcta, el sistema muestra un error y el flujo vuelve al paso 2. 4b. En caso de que los nuevos datos introducidos sean incorrectos, el sistema muestra un error y el flujo vuelve al paso 2. Los posibles errores en los datos son análogos al caso de uso CU02 - Registrar usuario, flujo alternativo 6b.
Inclusiones	Ninguna
Extensiones	Ninguna

5.2.5 GR01-Buscar receta

Identificador	GR01

Nombre	Buscar receta
Autor	Jorge Puig de la Bellacasa
Resumen	Este caso de uso muestra cómo un usuario busca una receta
Actor(es)	Usuario registrado
Precondiciones	Ninguna
Postcondiciones	Ninguna
Flujo normal	1. El caso de uso se inicia cuando el actor presiona el botón de búsqueda en el menú principal. 2. El sistema muestra un formulario con los diferentes criterios posibles de búsqueda (nombre de receta o todas las recetas de un usuario). 3. El actor introduce los datos solicitados y valida el formulario. 4. El sistema muestra un listado de recetas con los resultados de la búsqueda. El caso de uso finaliza.
Flujos alternativos	Ninguno
Inclusiones	Ninguna
Extensiones	GR02 (Visualizar receta)

5.2.6 GR02-Visualizar receta

Identificador	GR02
Nombre	Visualizar receta
Autor	Jorge Puig de la Bellacasa
Resumen	Este caso de uso muestra cómo un usuario accede a la vista de una receta
Actor(es)	Usuario registrado
Precondiciones	Ninguna
Postcondiciones	Ninguna
Flujo normal	1. El caso de uso se inicia cuando el actor selecciona una receta del listado proporcionado por el sistema en el caso de uso GR01. 2. El sistema muestra la vista de la receta seleccionada. El caso de uso finaliza.
Flujos alternativos	Ninguno
Inclusiones	Ninguna
Extensiones	GR03 (Añadir comentario) GR04 (Eliminar comentario) GR07 (Eliminar receta) GR08 (Modificar receta)

5.2.7 GR03-Añadir comentario

Identificador	GR03
Nombre	Añadir comentario
Autor	Jorge Puig de la Bellacasa
Resumen	Este caso de uso muestra cómo un usuario añade un comentario a una receta
Actor(es)	Usuario registrado
Precondiciones	Ninguna
Postcondiciones	Ninguna
Flujo normal	1. El caso de uso se inicia cuando el actor se encuentra visualizando una receta e introduce texto en el formulario de nuevo comentario, presionando el botón de enviar comentario después. 2. El sistema muestra un mensaje solicitando confirmación. 3. El actor confirma la acción. 5. El sistema añade el nuevo comentario a la receta y refresca la vista de la receta. El caso de uso finaliza.
Flujos alternativos	3a1. El actor pulsa el botón de cancelación. 3a2. El sistema muestra un mensaje confirmando la cancelación y vuelve a la vista de la receta. El caso de uso finaliza.
Inclusiones	Ninguna
Extensiones	Ninguna

5.2.8 GR04-Eliminar comentario

Identificador	GR04
Nombre	Eliminar comentario
Autor	Jorge Puig de la Bellacasa
Resumen	Este caso de uso muestra cómo un usuario elimina un comentario de una receta
Actor(es)	Usuario registrado, usuario administrador
Precondiciones	Ninguna
Postcondiciones	Ninguna
Flujo normal	1. El caso de uso se inicia cuando el actor pulsa el botón de eliminar de un comentario, dentro de la vista de una receta. 2. El sistema muestra un mensaje solicitando confirmación. 3. El actor confirma la acción. 4. El sistema muestra un mensaje confirmando la eliminación del comentario y refresca la vista de la receta. El caso de uso finaliza.
Flujos alternativos	2a. En caso de que el actor no sea usuario administrador, si además no es el autor original del comentario, el sistema muestra un mensaje de error y el caso de uso finaliza. 3a1. El actor cancela la acción. 3a2. El caso de uso finaliza.
Inclusiones	Ninguna

Extensiones	Ninguna

5.2.9 GR05-Añadir receta

Identificador	GR05
Nombre	Añadir receta
Autor	Jorge Puig de la Bellacasa
Resumen	Este caso de uso muestra cómo un usuario añade una receta al recetario
Actor(es)	Usuario registrado
Precondiciones	Ninguna
Postcondiciones	Ninguna
Flujo normal	1. El caso de uso se inicia cuando el actor pulsa el botón de añadir receta del menú principal. 2. El sistema muestra un formulario con los datos a introducir. 3. El actor introduce los datos solicitados (nombre de la receta, número de comensales, ingredientes y preparación). 4. El actor presiona el botón de enviar receta. 5. El sistema muestra un mensaje solicitando confirmación. 6. El autor confirma la acción. 7. El sistema añade los datos de la nueva receta a la base de datos y la muestra al actor. El caso de uso finaliza.
Flujos alternativos	4a1. El actor selecciona el botón añadir fotografía. 4a2. El sistema solicita al sistema operativo del dispositivo que lance la aplicación de fotografía predeterminada. La obtención de la fotografía a retornar dependerá del funcionamiento de la aplicación en cuestión. 4a3. El sistema operativo retorna la fotografía. El sistema guarda la fotografía a añadir a la receta y el flujo vuelve al paso 4.
Inclusiones	Ninguna
Extensiones	Ninguna

5.2.10 GR07-Eliminar receta

Identificador	GR07
Nombre	Eliminar receta
Autor	Jorge Puig de la Bellacasa
Resumen	Este caso de uso muestra cómo un usuario elimina una receta propia
Actor(es)	Usuario registrado
Precondiciones	Ninguna
Postcondiciones	Ninguna
Flujo normal	1. El caso de uso se inicia cuando el actor selecciona “eliminar

	receta” en la vista de una receta. 2. El sistema muestra un mensaje solicitando confirmación. 3. El usuario confirma la acción. 4. El sistema elimina la receta de la base de datos si el usuario es de tipo administrador o bien si la receta pertenece al usuario, y refresca la vista de gestión de recetas. El caso de uso finaliza.
Flujos alternativos	3a1. El usuario cancela la acción. 3a2. El caso de uso finaliza. 4a. Si el usuario no es administrador o bien la receta no pertenece al usuario el sistema informa al usuario de que no tiene permisos para realizar la acción y el caso de uso finaliza.
Inclusiones	Ninguna
Extensiones	Ninguna

5.2.11 GR08-Modificar receta

Identificador	GR08
Nombre	Modificar receta
Autor	Jorge Puig de la Bellacasa
Resumen	Este caso de uso muestra cómo un usuario modifica una receta propia
Actor(es)	Usuario registrado
Precondiciones	El actor ha seleccionado una receta en el menú de gestión de recetas
Postcondiciones	Ninguna
Flujo normal	1. El caso de uso se inicia cuando el actor selecciona “modificar receta” en la vista de una receta. 2. El sistema muestra una pantalla similar a la del caso de uso GR05 (añadir receta), con todos los datos existentes de la receta ya introducidos. 3. El actor modifica los datos que desee y presiona el botón “confirmar modificaciones”. 4. El sistema muestra una pantalla solicitando confirmación. 5. El actor confirma la acción. 6. El sistema modifica la receta y la muestra. El caso de uso finaliza.
Flujos alternativos	4a. Si el actor cancela la acción, el flujo vuelve al paso 2. 6a. Si el usuario no es administrador o el dueño de la receta el sistema informa al usuario de que no tiene permisos para realizar la acción y el caso de uso finaliza.
Inclusiones	Ninguna
Extensiones	Ninguna

5.2.12 GS01-Gestionar usuarios

Identificador	GS01

Nombre	Gestionar usuarios
Autor	Jorge Puig de la Bellacasa
Resumen	Este caso de uso muestra cómo un usuario administrador gestiona los usuarios registrados en el sistema
Actor(es)	Usuario administrador
Precondiciones	Ninguna
Postcondiciones	Ninguna
Flujo normal	1. El caso de uso se inicia cuando el actor accede al menú de gestión del sistema e introduce un criterio de búsqueda de usuario, presionando después el botón “administrar usuario”. 2. El sistema muestra una página con el identificador y tipo de usuario del usuario encontrado y las opciones disponibles (eliminar usuario o promocionar usuario). El caso de uso finaliza
Flujos alternativos	Ninguno
Inclusiones	Ninguna
Extensiones	GS02 (Eliminar usuario) GS03 (Promocionar usuario)

5.2.13 GS02-Eliminar usuario

Identificador	GS02
Nombre	Eliminar usuario
Autor	Jorge Puig de la Bellacasa
Resumen	Este caso de uso muestra cómo un usuario administrador elimina un usuario del sistema.
Actor(es)	Usuario administrador
Precondiciones	Ninguna
Postcondiciones	Ninguna
Flujo normal	1. El caso de uso se inicia cuando el actor selecciona “eliminar usuario” en la vista obtenida en el caso de uso GS01. 2. El sistema muestra un mensaje de advertencia y solicita confirmación del actor. 3. El actor confirma la acción. 4. El sistema elimina el usuario y todas sus aportaciones (recetas y comentarios) de la base de datos, y muestra un mensaje de confirmación. 5. El actor presiona “aceptar”. 6. El sistema muestra el menú de gestión del sistema y el caso de uso finaliza.
Flujos alternativos	3a. El actor cancela la acción. El flujo vuelve al paso 1.
Inclusiones	Ninguna
Extensiones	Ninguna

5.2.14 GS03-Promocionar usuario

Identificador	GS03
Nombre	Promocionar usuario
Autor	Jorge Puig de la Bellacasa
Resumen	Este caso de uso muestra cómo un usuario administrador promociona a un usuario registrado a usuario administrador
Actor(es)	Usuario administrador
Precondiciones	Ninguna
Postcondiciones	Ninguna
Flujo normal	1. El caso de uso se inicia cuando el actor selecciona “promocionar usuario” en la vista obtenida en el caso de uso GS01. 2. El sistema muestra un mensaje de advertencia y solicita confirmación del actor. 3. El actor confirma la acción. 4. El sistema promociona al usuario seleccionado a usuario administrador, y muestra un mensaje de confirmación. 5. El actor presiona “aceptar”. 6. El sistema muestra el menú de gestión del sistema y el caso de uso finaliza.
Flujos alternativos	3a. El actor cancela la acción. El flujo vuelve al paso 1.
Inclusiones	Ninguna
Extensiones	Ninguna

6 Diseño

6.1 Arquitectura global

La arquitectura global de la solución se basa en un sistema cliente/servidor, siguiendo el patrón Modelo Vista Controlador. De tal manera que el modelo se programará en *Java2SE* siguiendo el patrón *DAO* para facilitar el acceso a la base de datos, el controlador se implementará mediante el uso de *struts2*, y la vista se implementará mediante *HTML5* y *CSS3*, apoyándose en su caso en el uso de *JavaScript* y *jQuery Mobile*.

6.2 Arquitectura de componentes del servidor

La arquitectura del servidor se dividirá en tres capas:

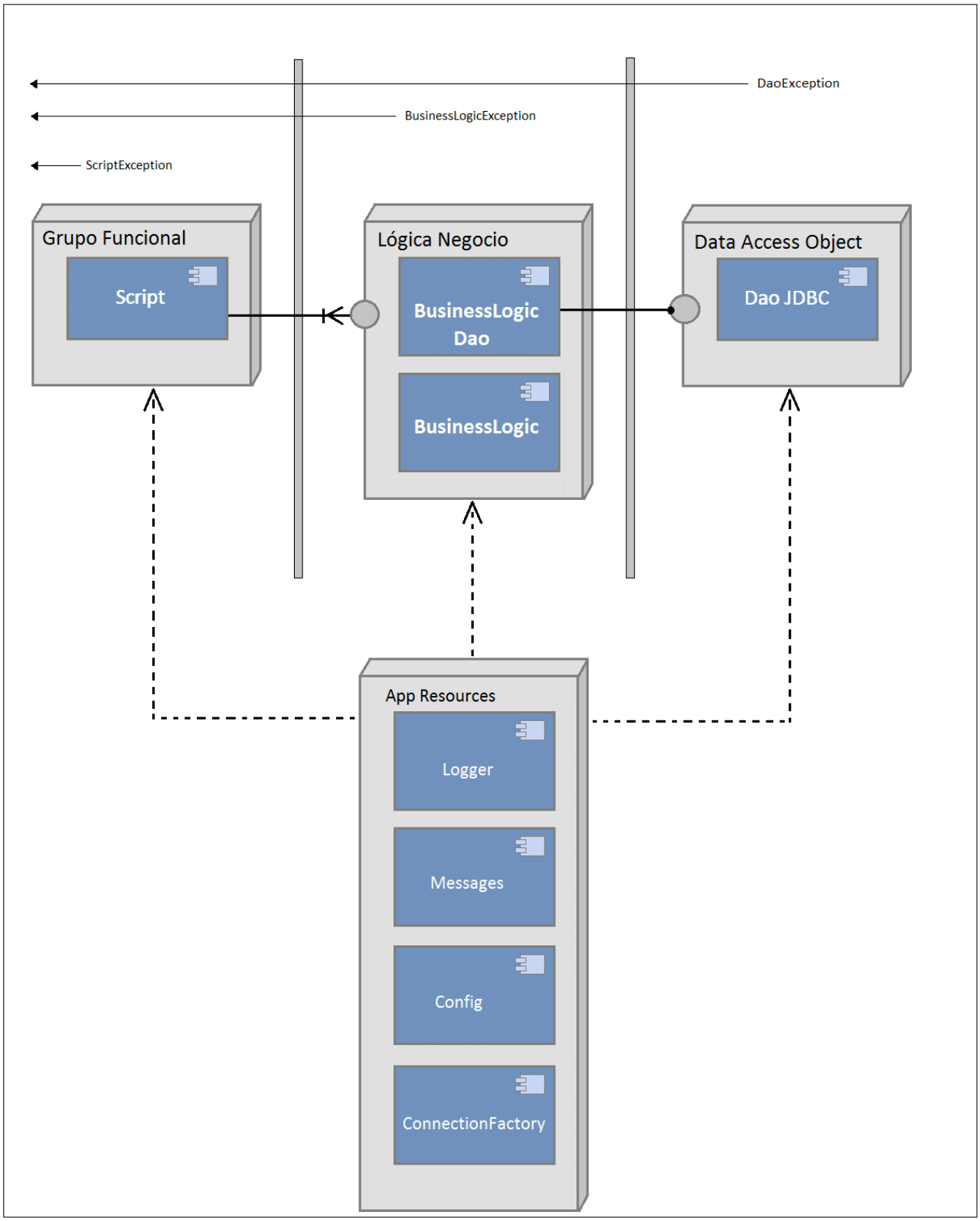
- La capa de acceso a datos desarrollada mediante el patrón *Data Access Object*, que consistirá en clases *DAO* (una por entidad de la base de datos) que se encargarán de traducir los datos recuperados de la base de datos en objetos Java (*POJO*) equivalentes a esas mismas entidades, permitiendo así un manejo intuitivo de los datos en Java, e independizando el funcionamiento del resto de capas del sistema del *SGBD* escogido.

- La capa de la lógica de negocio, que implementará el tratamiento de los datos según las reglas de negocio con una clase *BusinessLogic* para cada clase *DAO*.

- La capa de los grupos funcionales, que utilizará la capa de la lógica de negocio para implementar las funciones de los grupos funcionales especificados en los casos de uso explicados más arriba (identificación y registro, gestión de cuenta, gestión de recetas y gestión del sistema). Será por tanto esta la capa encargada de llevar el peso de la realización de una operación de usuarios en el servidor, a partir de los datos obtenidos en la parte de la vista.

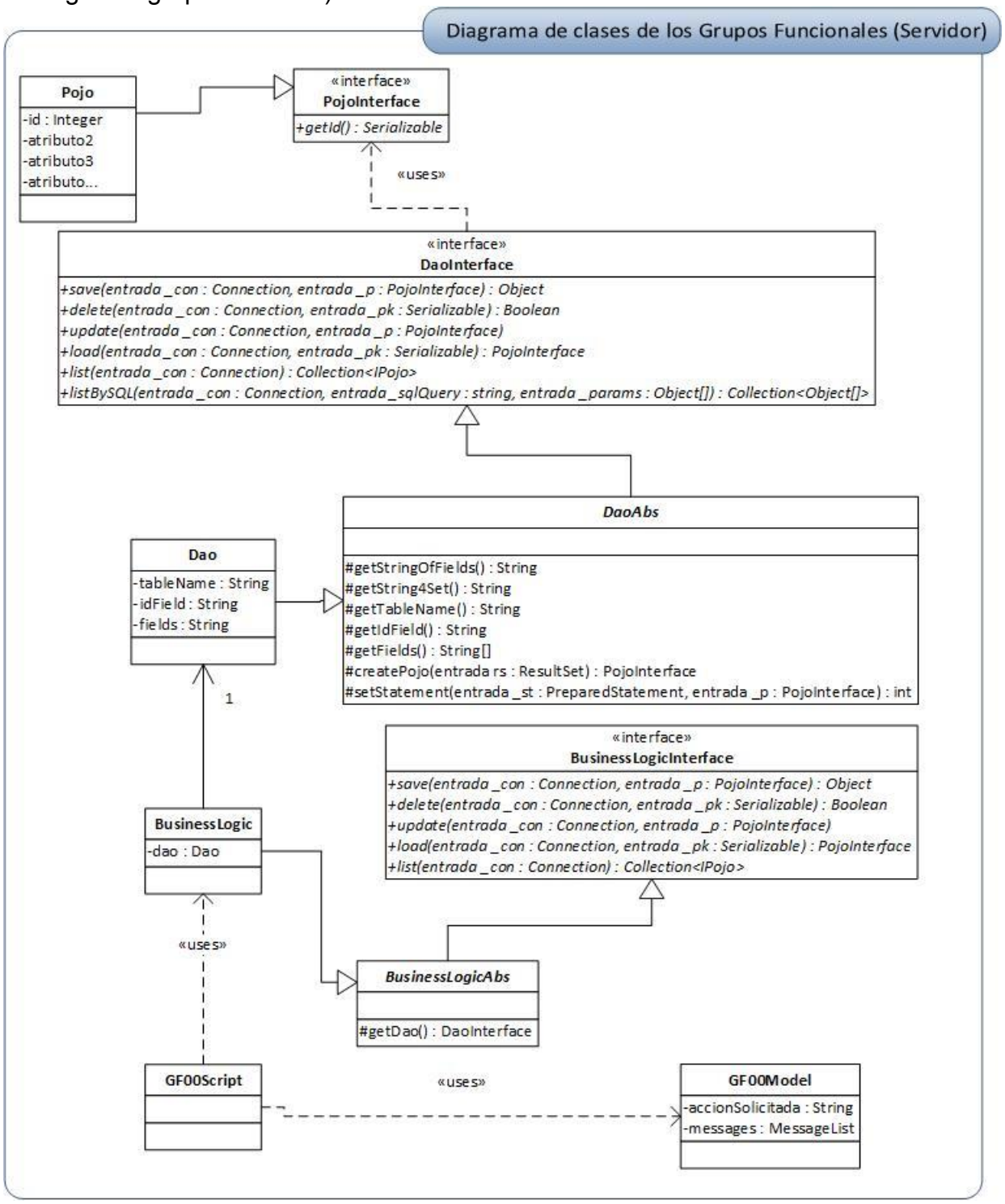
Además se dispondrá de una serie de utilidades externas como el archivo de configuración de conexión al *SGBD*, el *logger* para mantener un registro de los eventos, los mensajes que necesite lanzar el sistema y una factoría de conexiones para gestionar las conexiones al *SGBD*, así como cualquier otra utilidad externa que se necesite implementar durante el desarrollo del sistema.

El diagrama de componentes será por lo tanto el siguiente:



6.2.1 Diseño estático de clases de la parte servidor

A continuación se muestra el diagrama del diseño estático de clases de la parte servidor. Hay que tener en cuenta que por razones de legibilidad y espacio se han representado clases generales en lugar de las específicas de cada entidad de la base de datos. Es decir, donde se ve una clase *Pojo*, esta representa todas las clases *Pojo* existentes que corresponden a cada entidad en la base de datos (*UsuarioPojo*, *RecetaPojo*, etc.).
Lo mismo es aplicable a las clases *Dao*, *BusinessLogic*, *GF00Script* y *GF00Model* (donde *GF* significa grupo funcional).

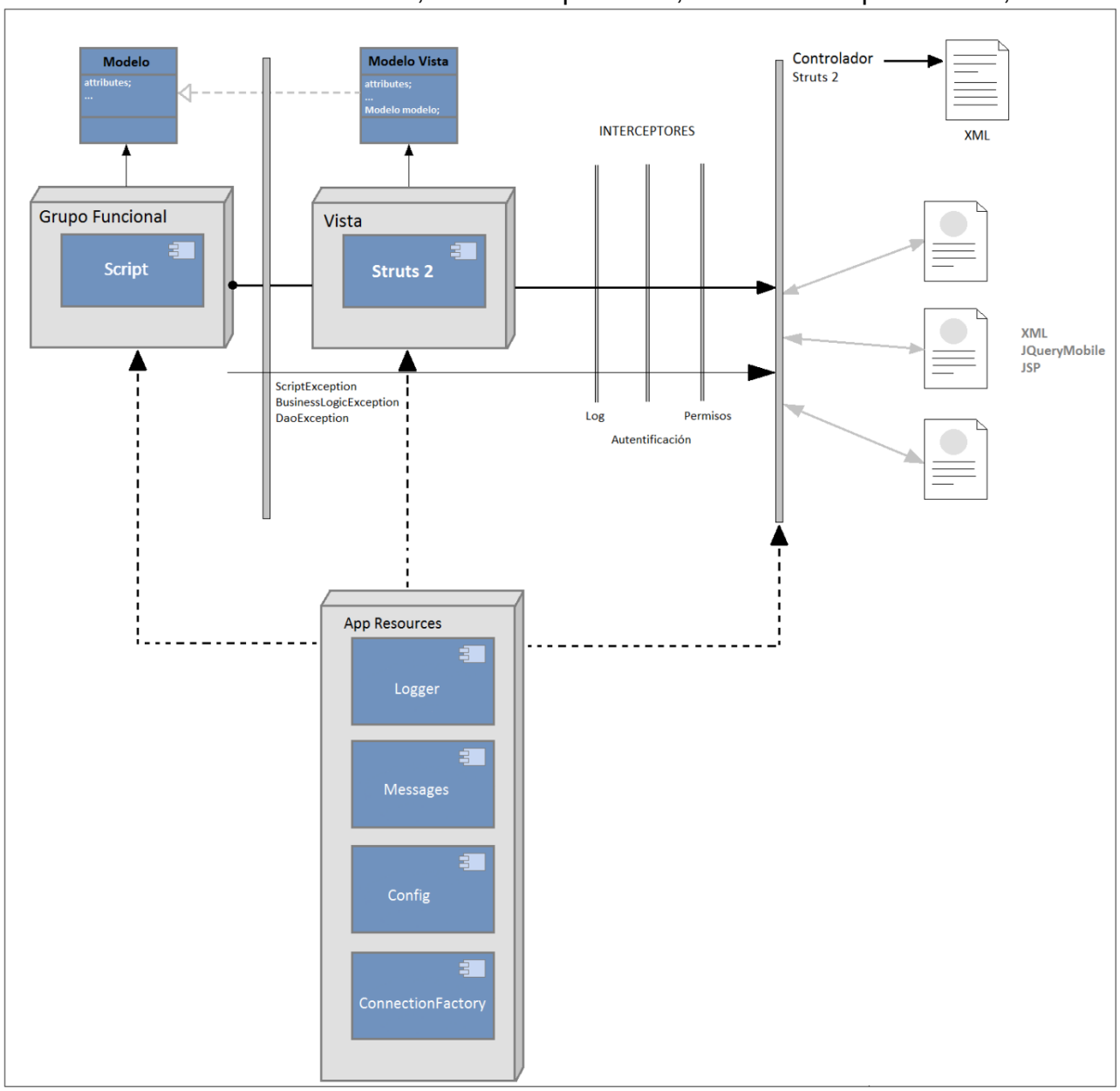


6.3 Arquitectura de componentes de la vista

La vista se comunicará con la capa de servidor mediante los *Script* de cada grupo funcional únicamente. Los datos necesarios para lanzar las operaciones del *Script* se intercambiarán mediante un *bean* (*Modelo* en el gráfico que se muestra a continuación), que a su vez será un subconjunto más pequeño de otro *bean* creado en la capa de presentación (*Modelo Vista* en el gráfico) que contendrá atributos necesarios para la correcta representación de resultados en la pantalla.

La implementación del Modelo Vista Controlador se realizará mediante el uso del *framework struts2*.

Para enlazar las validaciones transversales (aquellas que se aplican a varios grupos funcionales) se diseñará un sistema basado en Interceptores *struts2*. La bondad de este diseño es que se especifican en el mismo *XML* de definición de *struts* y se puede retornar al flujo del usuario antes de llegar a la capa Vista. Ejemplos comunes de interceptores serían módulos de autentificación, control de permisos, validación de parámetros, etc.



6.4 Prototipo de la vista

En el prototipo se ha optado por una vista orientada verticalmente, con un menú de navegación estático en la parte baja de la pantalla que será visible en prácticamente toda la aplicación. Si la página a mostrar es más larga que la pantalla del dispositivo, al hacer *scroll* el menú se mantendrá en la misma posición, siendo el resto de la página el que avance o retroceda.
Aunque se pretende llegar a todo tipo de dispositivos, el objetivo principal de este proyecto son los dispositivos móviles, y por lo tanto se ha tenido en cuenta el posible pequeño tamaño de las pantallas a la hora de realizar el diseño. Con botones grandes y un manejo que depende únicamente de pulsaciones (o *clicks* en dispositivos con ratón), *page scroll* y entrada de texto.

6.4.1 Identificación y registro

En la pantalla de la izquierda se realizará la identificación del usuario. Ésta será la pantalla que aparecerá al iniciar la aplicación.
Por otra parte, si no se dispone de cuenta de usuario se puede acceder a la pantalla de registro tocando el botón correspondiente, con lo que se pasaría a la pantalla de la derecha.

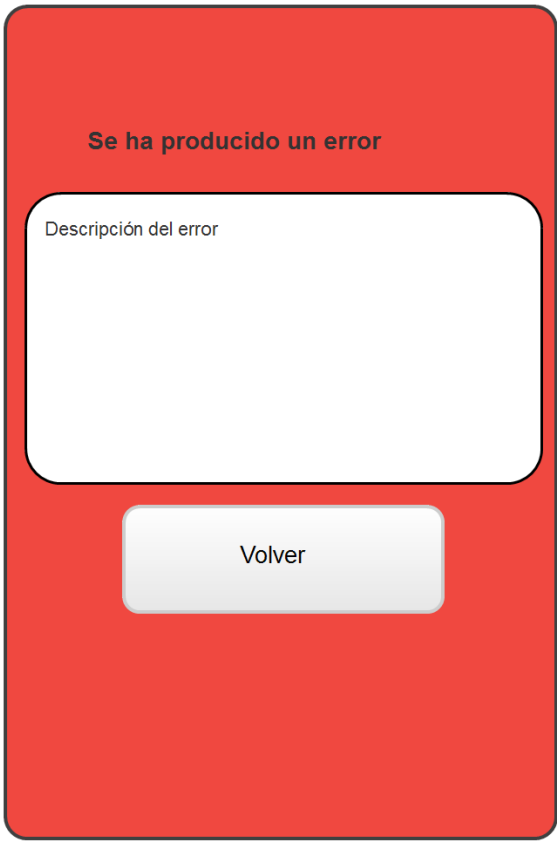
6.4.2 Búsqueda de recetas, menú de navegación y pantalla de error

Una vez identificado el usuario, se mostrará la pantalla de búsqueda de recetas. En esta pantalla aparece por primera vez el menú de navegación, que consta de cuatro botones para acceder a las diferentes secciones de la aplicación. La lupa llevará a esta misma

pantalla, el lápiz a la pantalla de introducir nueva receta, la rueda dentada al menú de configuración (*logout* y cambio de datos personales) y el gorro de policía al menú de moderación.

La pantalla en sí es el punto de partida para encontrar una receta, o bien para administrar las recetas del propio usuario. Se pueden buscar recetas por nombre de receta o por nombre del usuario que creó la receta.
Para buscar una receta se introducirá un criterio de búsqueda y se pulsará el botón situado a la derecha de ese mismo criterio. Una vez hecho esto, el sistema mostrará la pantalla que se muestra a la derecha en este documento, donde el usuario deberá escoger una receta y pulsar “mostrar receta seleccionada”.

Una vez seleccionada una receta de la lista y pulsado el botón de mostrar, se mostrará la vista de la receta en cuestión, como se puede ver en la siguiente figura.
Esta vista muestra la receta con su fotografía (si la hubiere), además de los comentarios que otros usuarios hayan añadido a la misma. Desde esta misma pantalla se puede añadir un nuevo comentario y eliminar un comentario anterior, aunque únicamente los usuarios administradores podrán borrar cualquier comentario, mientras que los usuarios registrados únicamente podrán eliminar sus propios comentarios.
En caso de que un usuario no administrador intente borrar un comentario que no le pertenezca, se mostrará una pantalla de error con el mensaje pertinente. Aprovecharemos esta oportunidad para enseñar la pantalla de notificación de error estándar, a la derecha de la pantalla de receta.



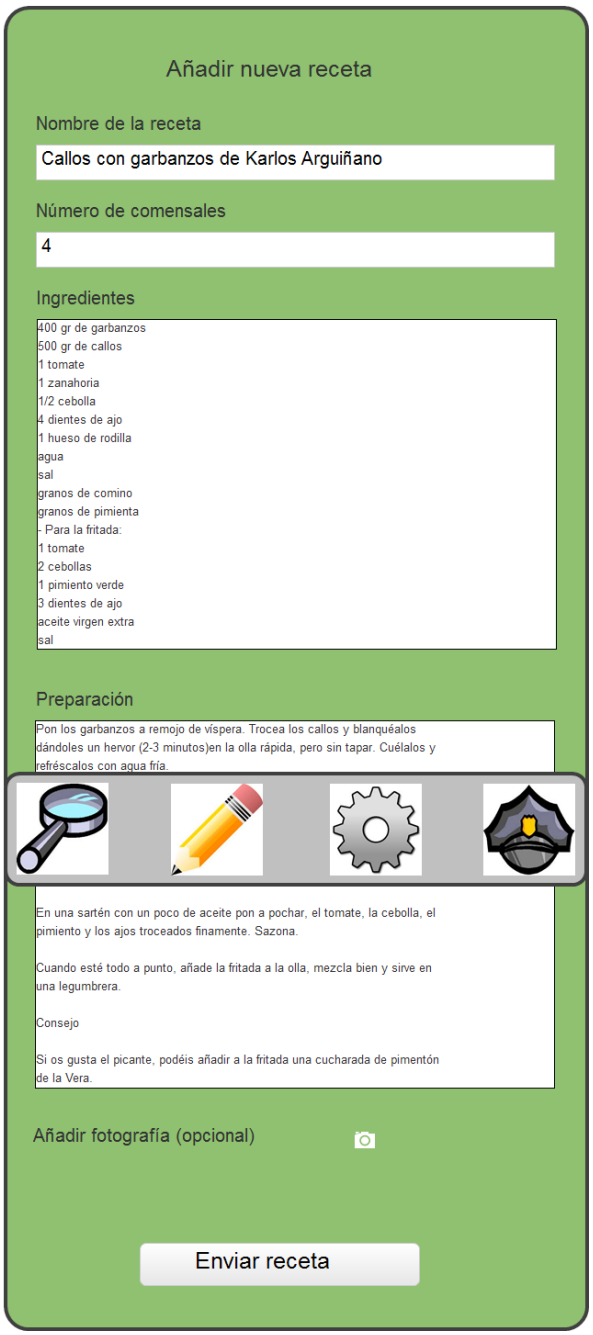
Nótese cómo se muestra en la pantalla de la izquierda lo que ocurre cuando la página es demasiado larga para la pantalla del dispositivo. El menú se muestra flotando porque se encuentra en la parte más baja de la pantalla. Si se hace *scroll down* el menú seguirá en la parte baja de la pantalla, mientras el contenido oculto surge desde abajo.

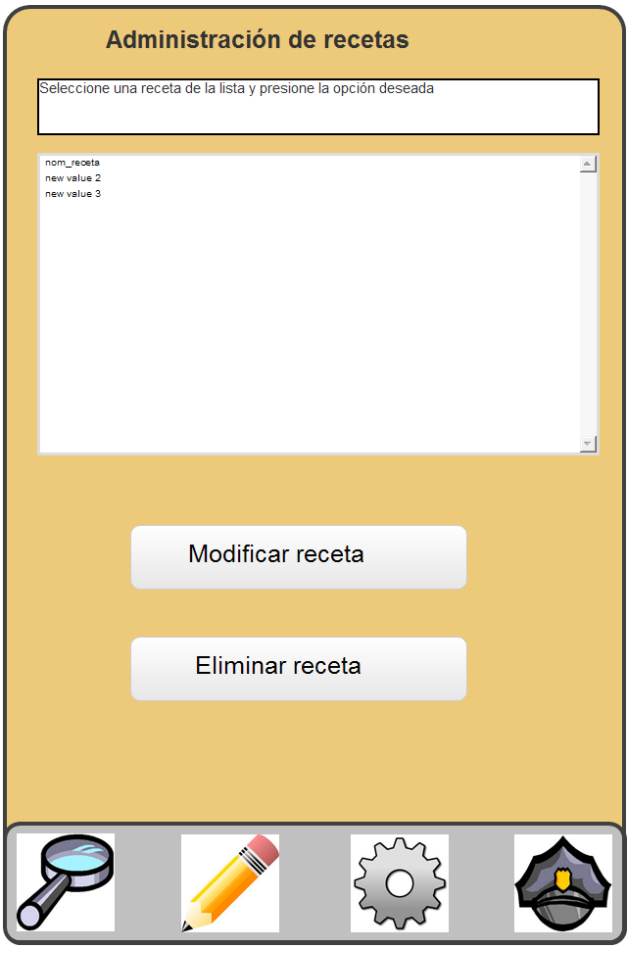
La pantalla de error mostrará un mensaje describiendo el error, y al pulsar el botón volver se volverá a la pantalla anterior.

6.4.3 Nueva receta y administración de recetas propias

Cuando se presione el botón del lápiz que se encuentra en el menú de navegación, el sistema mostrará la pantalla de crear nueva receta, que se muestra a continuación. En ella el usuario deberá introducir los datos solicitados, y opcionalmente podrá añadir una fotografía que podrá obtener de su dispositivo, ya sea mediante la cámara integrada o la

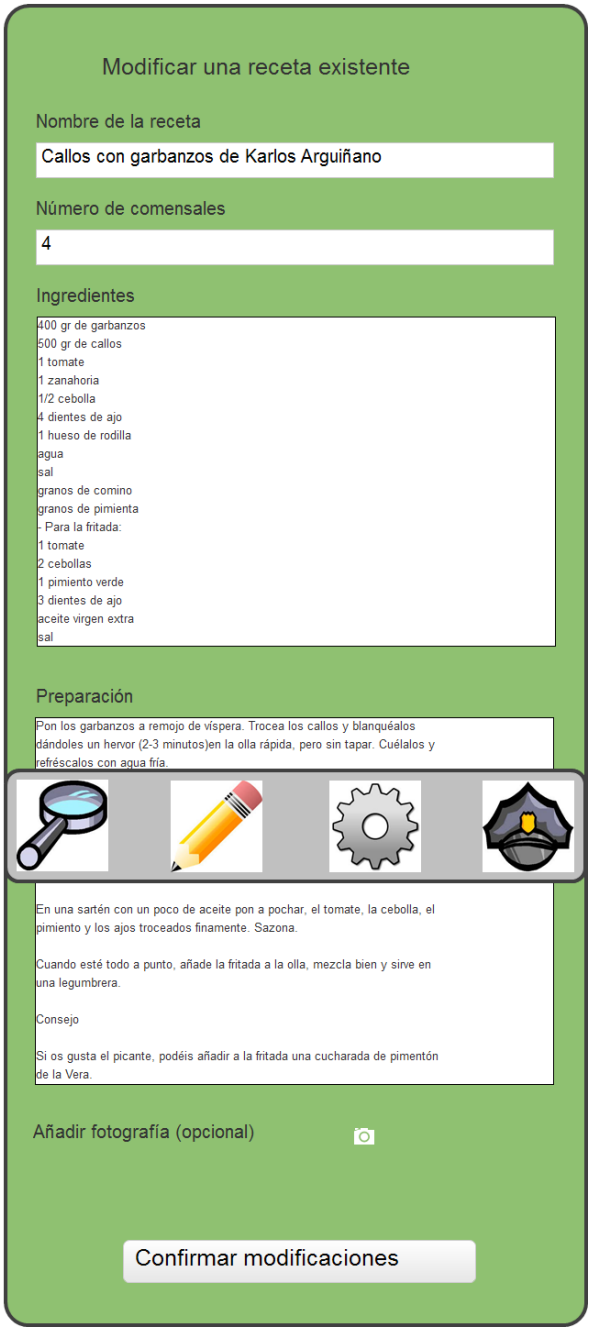
búsqueda de una fotografía ya existente en la memoria del dispositivo. La obtención de esta fotografía dependerá del sistema operativo en cuestión.
Una vez se presione el botón “enviar receta”, y si no se produce ningún error, se creará la nueva receta y el sistema la mostrará mediante la pantalla vista en el apartado anterior.





Si en la pantalla de búsqueda de recetas del apartado anterior se presiona el botón “mostrar y administrar mis recetas” se llegará a la página que se muestra a la derecha. En ella se muestra un listado con todas las recetas del usuario, y seleccionando una se puede modificar o eliminar dependiendo del botón escogido. Si se quiere eliminar una receta el sistema mostrará un mensaje solicitando confirmación.

Si por el contrario se presiona modificar receta, el sistema mostrará la siguiente pantalla, que como se puede observar es prácticamente igual a la pantalla de añadir nueva receta. Cuando se muestre esta pantalla, los apartados del formulario se encontrarán ya preparados con los datos existentes de la receta.



6.4.4 Menú de configuración

Cuando se presione la rueda dentada del menú de navegación se mostrará la pantalla de configuración.
En ésta, un usuario puede cerrar la sesión en curso y cambiar sus datos personales (excepto su identificador de usuario).
A continuación se muestra esta pantalla.

6.4.5 Menú de administración del sistema

Cuando se presiona el gorro de policía del menú de navegación el sistema muestra la pantalla que se ve a continuación. Únicamente los usuarios administradores podrán acceder a este menú, mientras que cualquier usuario no administrador que lo presione se encontrará con un mensaje de error.
Desde esta pantalla se puede acceder a la administración de usuarios y de recetas, introduciendo un criterio de búsqueda en el formulario adecuado y presionando el botón correspondiente.

La pantalla que se muestra a la derecha corresponde a aquella a la que se llegaría en caso de presionar “administrar usuarios” en la pantalla de la izquierda.
Esta pantalla mostrará un listado con los usuarios cuyo identificador contiene la cadena de caracteres y números introducida en la pantalla de la izquierda.
El administrador deberá entonces seleccionar un resultado y podrá promocionar (dar permisos de administrador) o eliminar el usuario seleccionado.
En esta figura se puede observar, además, un ejemplo de una pantalla de confirmación.

En caso de seleccionar *administrar recetas* en la pantalla de la izquierda, el sistema mostrará la misma pantalla de administración de recetas del apartado 6.4.3, y el funcionamiento será análogo, excepto por el hecho de que se mostrarán en el listado las recetas cuyo nombre contenga la cadena de caracteres introducido antes de presionar “administrar recetas”.

Un usuario administrador podrá, por tanto, eliminar o modificar cualquier receta.

6.5 Diseño de la persistencia

La persistencia de este proyecto es muy simple. Únicamente ha de representarse a los usuarios, las recetas, y los comentarios. Siendo las fotografías un atributo de la receta, ya que únicamente se permite una fotografía por receta.

6.5.1 Modelo relacional de la base de datos

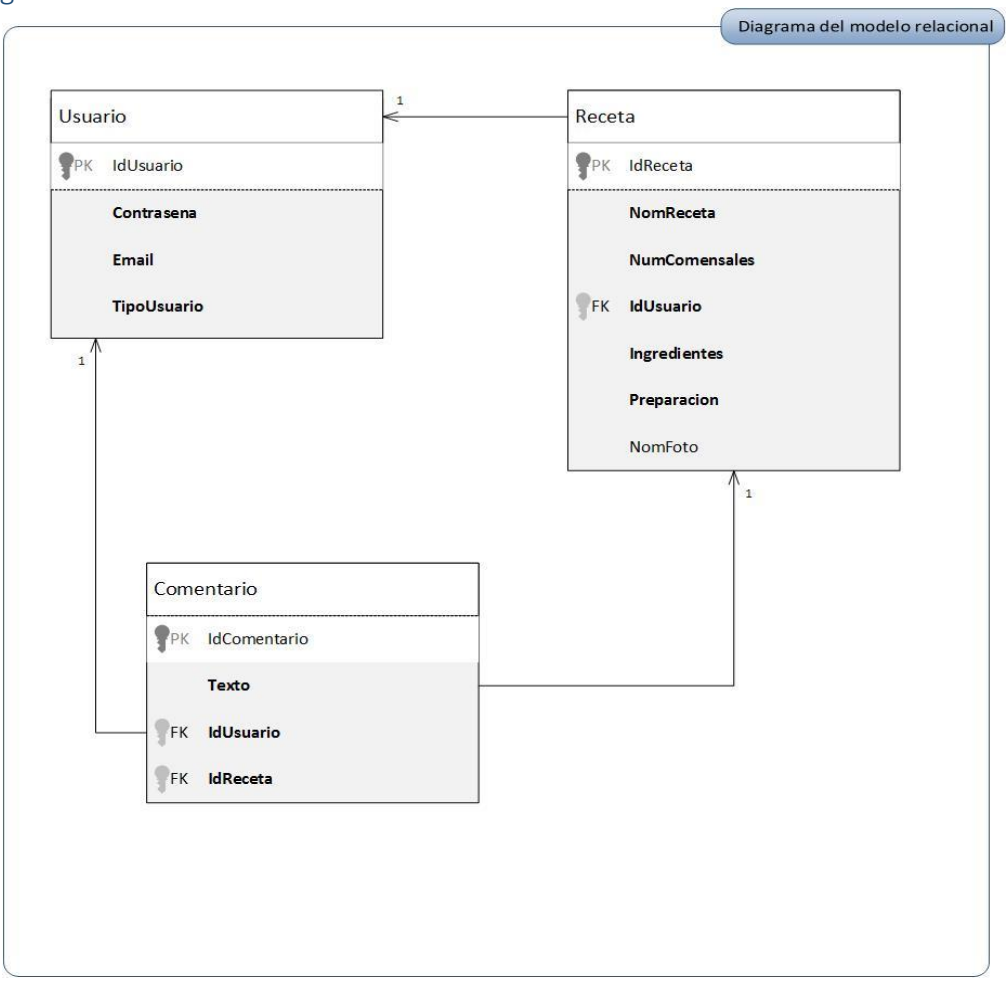
Siguiendo los requisitos de información especificados en su correspondiente apartado, se ha diseñado el siguiente modelo relacional para la base de datos:

Usuario (id_usuario, contrasena, email, iSadmin)

Receta (id_receta, nom_receta, num_comensales, id_usuario, ingredientes, preparacion, foto)
{id_usuario} clave foránea hacia Usuario (id_usuario)
foto puede tomar valores nulos

Comentario (id_comentario, texto, id_usuario, id_receta)
{id_usuario} clave foránea hacia Usuario (id_usuario)
{id_receta} clave foránea hacia Receta (id_receta)

6.5.2 Diagrama del modelo relacional



7 Implementación

En los siguientes apartados se describe la fase de implementación de cada uno de los componentes, y se explica su funcionamiento y principales características.

7.1 Modelo

7.1.1 Base de datos

La base de datos únicamente almacena la información y no implementa reglas de negocio por haber considerado lo contrario un método poco elegante y difícil de mantener. Por lo tanto el esquema es realmente simple: se han creado las entidades identificando sus claves primarias y foráneas únicamente, además claro está del resto de atributos.

Esto implica que es necesaria cierta cautela si se modifican los datos de la base de datos directamente, sin pasar por los componentes de Java que implementan las reglas de negocio.

Las fotografías se almacenan directamente en disco en el servidor, separadas en carpetas nombradas según el identificador único de cada receta en la base de datos. Mientras que en la propia base de datos únicamente se almacena el nombre del archivo para su posterior recuperación. Este método se ha escogido una vez más por eficiencia y elegancia, ya que lo contrario implicaría transmisiones de datos mucho mayores entre la base de datos y la aplicación *Java*, y aunque en general se instalarán ambos componentes en el mismo servidor, nada impide (dada la implementación) que se instalen cada uno en un servidor, y cada servidor en diferentes localizaciones.

7.1.2 Acceso a Datos

El acceso a datos se realiza mediante el patrón *DAO*, de manera que se ha creado una clase *POJO* (*Plain Old Java Object*) por cada entidad de la base de datos. Estas clases implementan el interfaz *IPojo*, que únicamente obliga a las clases que lo implementan a incluir un método que retorne el identificador único del objeto *POJO*, cuyo tipo de datos únicamente ha de implementar el interfaz *Serializable*.

Estas clases *POJO* tienen como único objetivo representar a la entidad de la base de datos correspondiente, y por lo tanto implementan los mismos atributos que la entidad con tipos de datos equivalentes, además de los métodos *getters* y *setters* y un método *toString()* personalizado para su uso en pruebas.

Las clases *DAO* (*Data Access Object*) serán las encargadas de realizar el acceso a datos y transformar los datos obtenidos de la base de datos en objetos *POJO* o viceversa.

Para la implementación de este elemento se ha creado un interfaz *IDao* que indica los métodos mínimos que una clase *DAO* debe implementar: guardar, borrar, actualizar y cargar una entrada por su identificador; devolver todas las entradas de su entidad correspondiente y finalmente listar por consulta *SQL* (para consultas más complejas).

Seguidamente se ha creado una clase abstracta *DaoAbs* que implementa todos estos

métodos y define otra serie de métodos auxiliares que deberán implementar las clases *DAO*, de los que se valdrá para obtener datos específicos necesarios para realizar las consultas, como el nombre de la tabla, el listado de sus atributos etc. Así como dos métodos: *createPojo* y *setStatement* que deberán crear el objeto *POJO* correspondiente a un *ResultSet* dado, e introducir los datos de un *POJO* dado en un *PreparedStatement* respectivamente. Estos métodos serán por lo tanto los encargados de traducir los tipos de datos entre base de datos y aplicación *Java*, y viceversa.

Finalmente, se ha creado una clase *DAO* por cada entidad de la base de datos, heredando todas ellas de *DaoAbs*. Estas clases contienen los atributos que indican el nombre de la tabla, el nombre de la clave primaria y un *Array* con los nombres de los diferentes atributos de la tabla, e implementan los métodos *createPojo* y *setStatement* que acabamos de comentar.

El acceso a la base de datos se ha implementado mediante la librería *SQL* estándar de *Java*, de manera que se pueda cambiar de Sistema de Gestión de Bases de Datos si se desea únicamente indicándole a la aplicación *Java* el driver *JDBC* correspondiente en el archivo de configuración *db.properties*. Será la clase ConexionFactory la factoría encargada de crear los objetos *Connection* a utilizar según los datos de conexión indicados en este archivo.

7.1.3 Lógica de negocio

La capa de lógica de negocio se vale de la capa *DAO* para realizar funciones más complejas derivadas de las relaciones entre entidades, como por ejemplo obtener un listado de los comentarios pertenecientes a una receta, o eliminar un autor de la base de datos (que requiere eliminar primero sus comentarios y seguidamente sus recetas para no entrar en conflicto con las reglas de integridad de la base de datos).

Para su implementación se ha seguido el mismo método que con las clases *DAO*. Es decir, se ha creado un interfaz *IBusinessLogic* que determina los métodos mínimos que toda clase *BusinessLogic* debe implementar. Estos métodos son equivalentes a algunos de los métodos definidos en el interfaz *IDao* (*save, load, delete, update, list*), y son implementados por una clase abstracta *BusinessLogicAbs* que simplemente llama a los métodos de la clase *DAO* correspondiente. Esta clase abstracta define además un método *getter* abstracto que deberán implementar las clases que hereden de ella, para recuperar el *DAO* a utilizar.

Finalmente se ha implementado una clase *BusinessLogic* por cada entidad de la base de datos, que hereda de *BusinessLogicAbs* y contiene un atributo con el *DAO* correspondiente, e implementa los métodos más complejos de la lógica de negocio.

Será de los métodos definidos en estas clases de los que se valgan las clases *Script* para realizar las acciones correspondientes a los grupos funcionales, descritos anteriormente en la arquitectura del proyecto.

Existe una clase *Script* por cada grupo funcional, y cada una de ellas se vale de una o varias clases *Model* que almacenarán los datos a transferir entre el módulo controlador y el módulo modelo y viceversa. Estas clases *Script* se encargan de implementar cada uno de los casos de uso de su grupo funcional y además tienen los siguientes cometidos:

- Crear y administrar las conexiones a utilizar mediante el uso de la clase

ConexionFactory.

- *Logear* en el sistema las excepciones que puedan producirse en las capas anteriores mediante el uso de la clase *Logger*.
- Encriptar y comprobar contraseñas mediante el uso de la librería *jasypt*.

7.2 Controlador

Como se ha comentado anteriormente, el controlador será el encargado de comunicar el modelo con la vista, y realizar funciones de validación de datos, seguridad y control de sesión. Y ha sido implementado mediante el uso de la librería de *Java struts2*.

El funcionamiento de *struts2* es complejo y no es el objetivo de este proyecto el realizar una explicación en profundidad del mismo, pero en términos generales es el siguiente:

En el archivo *web.xml* del contenedor web se le indica a éste que será *struts2* el encargado de gestionar las peticiones *HTTP* que sigan un patrón determinado, en nuestro caso todas las peticiones.

A partir de este momento, toda petición será resuelta por *struts2* de la siguiente manera:

1. El usuario realiza una petición *HTTP*.
2. *Struts2* determina la acción encargada de procesar la petición según el mapeado determinado en su archivo *struts.xml*, así como los interceptores que será necesario utilizar, también especificados en el archivo *struts.xml*.
3. Los interceptores saltan en el orden indicado de manera recursiva, de forma que un interceptor realiza sus tareas (como la validación de datos) y llama al siguiente interceptor, que realiza sus tareas y llama al siguiente, y así sucesivamente hasta llegar al último interceptor que llama al método correspondiente de la acción. En este punto todos los datos que la acción necesitará para sus tareas se encuentran poblados en los atributos de la acción, en nuestro caso una clase *Model*.
4. La acción realiza sus tareas y modifica los datos de la clase *Model* para su uso en la creación dinámica de la página a presentar (en nuestro caso mediante el uso de *JSPs*), y retorna una cadena que será utilizada para determinar el archivo *JSP* a utilizar, según el mapeado del archivo *struts.xml*.
5. El flujo pasa al último interceptor, que realiza otra serie de acciones si es necesario y al terminar el flujo pasa al interceptor anterior, y así recursivamente hasta el final del método recursivo.
6. *Struts2* sirve a la vista la página *HTML* creada mediante el uso del *JSP* correspondiente.

A continuación se realiza una descripción más detallada de las clases que componen este módulo.

7.2.1 Interceptores

Struts2 consta de multitud de interceptores listos para utilizar en las tareas más comunes de cualquier aplicación *web*. Y aunque permite crear nuestros propios interceptores el paquete ofrecido por *struts2* cubre con creces las necesidades de este proyecto, por lo que no se ha desarrollado ningún interceptor nuevo para su uso en el mismo.

Entre las tareas de estos interceptores se encuentran las de poblar los atributos de la clase *Model* correspondiente (realizando para ello la conversión de datos adecuada, dado

que todos los datos que se obtienen de una consulta *HTTP* son cadenas de texto), gestionar la subida de archivos al servidor, crear la sesión, generar los archivos *HTML* a partir de los datos del modelo y de los archivos *JSP*, y realizar las acciones necesarias si los datos no son validados por la acción correspondiente.

7.2.2 Modelos

Los modelos son clases creadas para almacenar y transferir los datos entre el módulo de la vista y el del modelo. Dado que los datos a utilizar en cada caso son distintos existe al menos una clase *Model* por cada grupo funcional especificado en la arquitectura del proyecto, que contiene atributos como el identificador de usuario de la sesión actual, el identificador de la receta a utilizar, la acción solicitada, colecciones de recetas o comentarios etc.

Serán estas clases las que los interceptores pueblen con los datos obtenidos de la vista y también las que las acciones pueblen con datos para su uso en la generación dinámica de los archivos *HTML* a presentar.

7.2.3 Acciones

Las acciones son clases que heredan de la clase de *struts2 ActionSupport*, y son las encargadas de las siguientes tareas:

- Validación de datos: La acción debe implementar un método *validate* que se encargue de comprobar que los datos con los que se ha poblado su clase *Model* son correctos. De lo contrario la acción deberá añadir un error de campo mediante su método *addFieldError*, pasando como parámetro el campo del formulario donde se ha producido el error, y un mensaje explicativo para el usuario.
  Si se ha añadido algún error de campo, el interceptor de validación se encargará de mostrar al usuario la página del formulario en el que se produjo el error de validación, con el mensaje explicativo situado encima del campo donde se produjo el error.
- Control de permisos: En su método principal (*execute* por defecto), la acción será la encargada de comprobar que el usuario almacenado en la sesión tiene permisos para realizar la acción que ha solicitado. Para acceder a esta sesión, la acción debe implementar el interfaz *SessionAware*.
- Protección contra inyección de código: Cuando se vayan a mostrar datos introducidos por usuarios en el archivo *HTML* generado, la acción ha de escapar estos datos para evitar ataques de inyección de código, y lo hace mediante el uso de la librería de seguridad *ESAPI*.
- Comprobar que la petición *HTTP* incluye todos los datos necesarios para la ejecución del caso de uso en cuestión, ya que la comprobación realizada por la vista es fácilmente sorteable.
- Llamar a los métodos adecuados de las clases *Script* para realizar las tareas del caso de uso correspondiente, pasándoles la clase *Model* ya poblada con los datos necesarios, y actualizando esta misma clase con los resultados obtenidos de la clase *Script* para su posterior uso en la generación dinámica de los archivos *HTML* a partir del archivo *JSP* correspondiente.

Las acciones pueden almacenar los datos en *beans* o, como es nuestro caso, en una clase modelo, para lo que la acción debe implementar el interfaz *ModelDriven<Modelo>*.

Asimismo, aunque el método principal de la acción por defecto es el método *execute*, se pueden implementar otros métodos de manera que una acción pueda encargarse de varios o todos los casos de uso de un grupo funcional. Será en el archivo *struts.xml* donde se especifique el método a llamar según sea necesario.

7.3 Vista

Como se ha comentado anteriormente, la vista se compone de los archivos *HTML* generados a partir de los archivos *JSP* correspondientes.

Gracias a las *tags* de *struts2* se pueden inyectar los datos del modelo en estos archivos *HTML*, y asignar los datos recuperados de los formularios a los atributos correspondientes de la clase *Model*. Sin embargo, *struts2* no está diseñado con *HTML5* en mente y ha sido necesario evitar los nuevos elementos introducidos por éste.

En cuanto al uso de *JavaScript* y *jQuery*, éste se ha limitado al uso de la librería *jQuery mobile*, que permite ofrecer un diseño elegante y dinámico sin tener necesariamente que conocer el funcionamiento de *JavaScript* o su librería *jQuery*.
Además *jQuery mobile* se encarga de ajustar los datos a la pantalla del dispositivo que se use para acceder a la aplicación, de manera que se han probado con éxito multitud de dispositivos que se detallan a continuación:

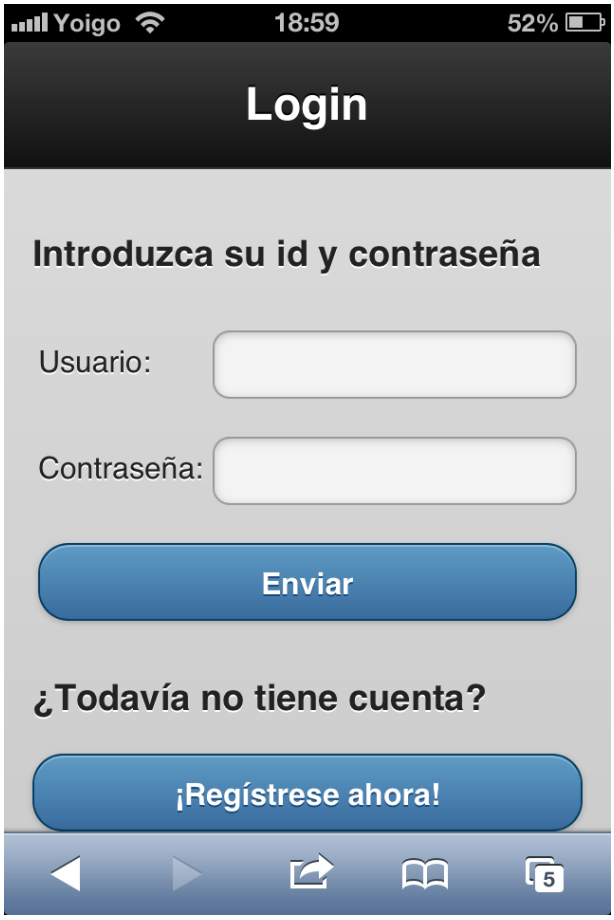
- *IPhone 4* con *iOS 6*, tanto con el navegador *Safari* como con *Chrome*.
- *IPad 2* con *iOS 6*, con los mismos navegadores: *Safari* y *Chrome*.
- *Samsung Galaxy Nexus*, con *Android 4.3* y el navegador de *Android*, así como *Chrome*.
- *HTC Desire*, con *Android 2.3* y el navegador de *Android*.
- *Samsung Galaxy Note 10.1*, con *Android 4.1.2* y los navegadores de *Android* y *Chrome*.
- *PC* con navegadores *Firefox* y *Chrome*.

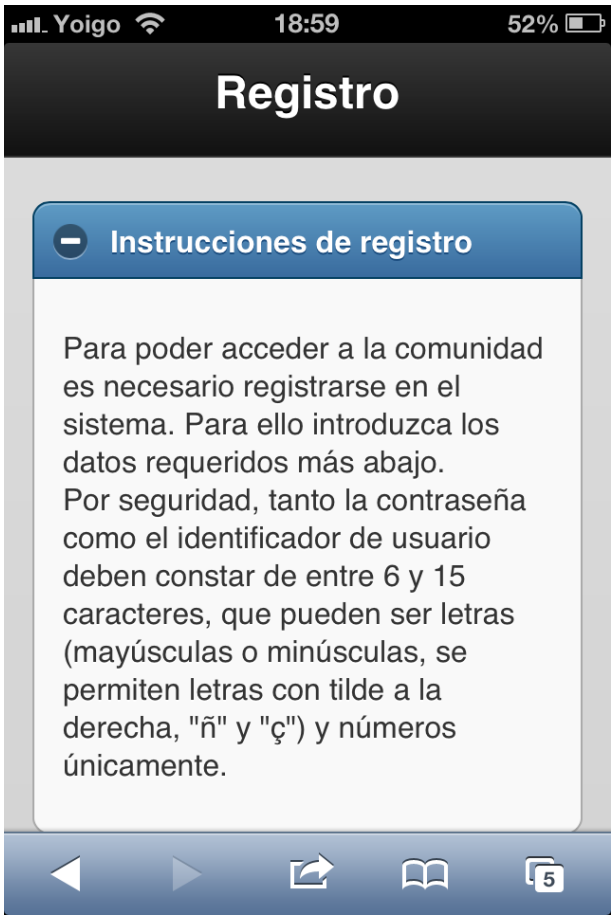
El único fallo encontrado en estos dispositivos consiste en que algunos navegadores no permiten el uso del botón “atrás”, pero la aplicación no requiere en absoluto del uso de este botón.
Además, la vista en *PC* muestra los elementos quizá excesivamente grandes, pero de forma funcional. Sin embargo como ya se ha comentado en este documento, el objetivo del proyecto eran los dispositivos móviles, y no entra dentro de la planificación o los objetivos implementar una vista exclusiva para *PC*.

7.3.1 Resultado

A continuación se muestra una serie de capturas de pantalla de la aplicación en un *iPhone 4*, con comentarios relativos al resultado obtenido con esta implementación y a los cambios que se han realizado sobre el prototipo inicial de la vista.

Como se puede observar en estas capturas, se han generado elementos elegantes como botones, cuadros de texto desplegables y cabeceras mediante el uso de *jQuery mobile*.

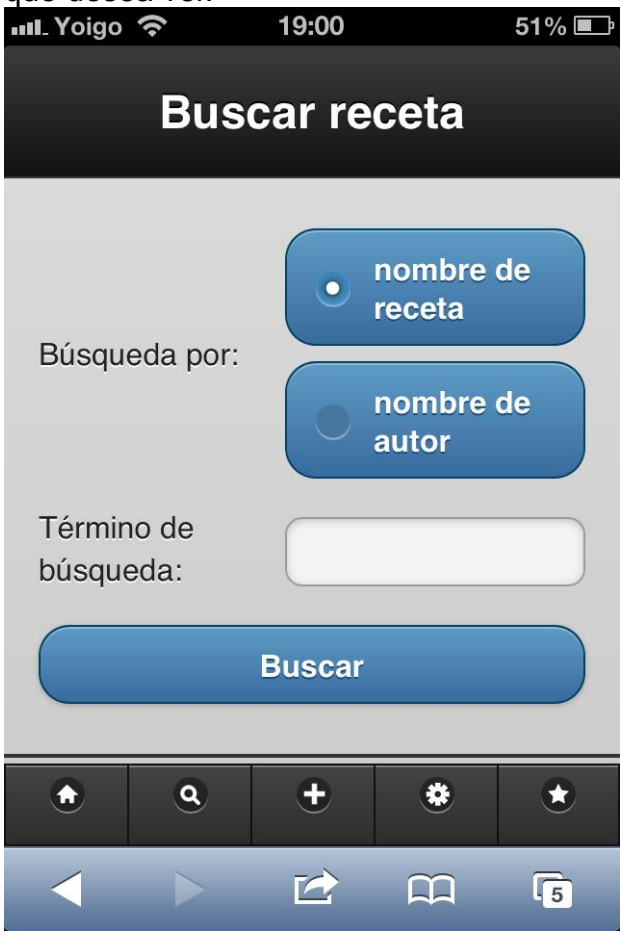




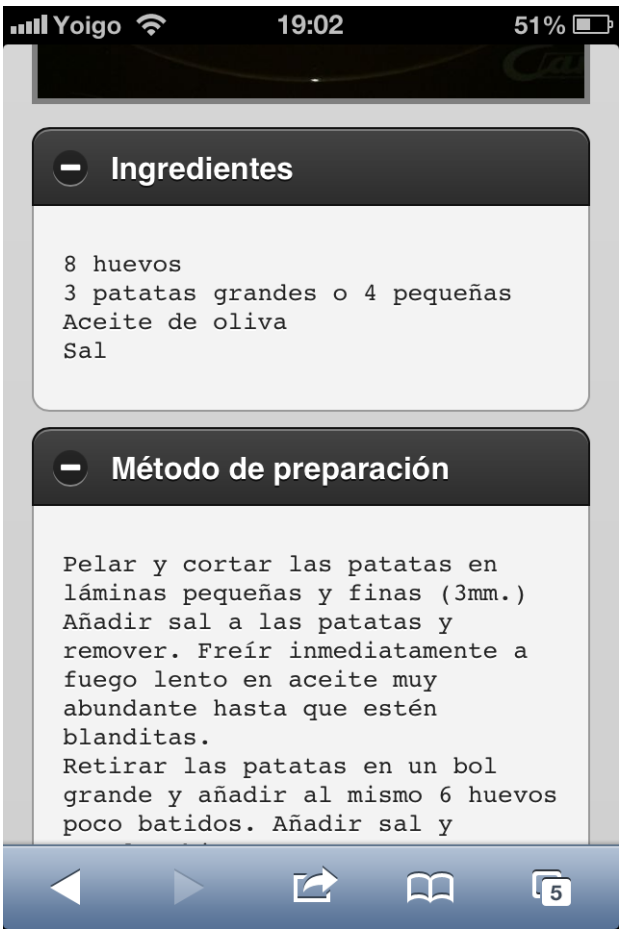
En la captura siguiente situada a la derecha, se puede ver como aparece el menú de navegación una vez el usuario se registra en el sistema.
Al contrario que en el prototipo conceptual, se han utilizado los iconos que proporciona *jQuery mobile*, mucho más adecuados en su diseño, y se ha añadido un quinto botón, el botón de *home* (primero por la izquierda), que lleva a la página de bienvenida que se muestra en la misma captura (también una nueva adición).
El resto de botones guarda el mismo orden que se mostraba en el prototipo: buscar receta, añadir receta, gestión de cuenta y gestión del sistema, de izquierda a derecha.

En las siguientes capturas se puede apreciar un menú radial y cómo se ha resuelto el problema de mostrar el listado de recetas correspondiente de manera que sean legibles y fácilmente seleccionables en dispositivos pequeños.

En este listado cada elemento es un botón que al ser pulsado dirige directamente a la vista de la receta en cuestión. Y esta solución permite listar grandes cantidades de recetas permitiendo al usuario simplemente bajar en la página para encontrar la receta que desea ver.



Las siguientes capturas muestran la vista de una receta.
Como se puede observar la presentación de los datos difiere de aquella anticipada en el prototipo, con cuadros de texto desplegables para los ingredientes, el método de preparación y los comentarios de los usuarios. Y además en la cabecera se puede apreciar un cambio significativo: un botón que lleva a un menú de opciones que se muestra en el siguiente par de capturas.



Aquí podemos observar cómo se presentan los comentarios de los usuarios. El botón a la derecha de cada comentario sirve para borrar el comentario correspondiente. Recordemos que únicamente los usuarios administradores pueden borrar todos los comentarios, y los usuarios regulares pueden borrar sus propios comentarios.

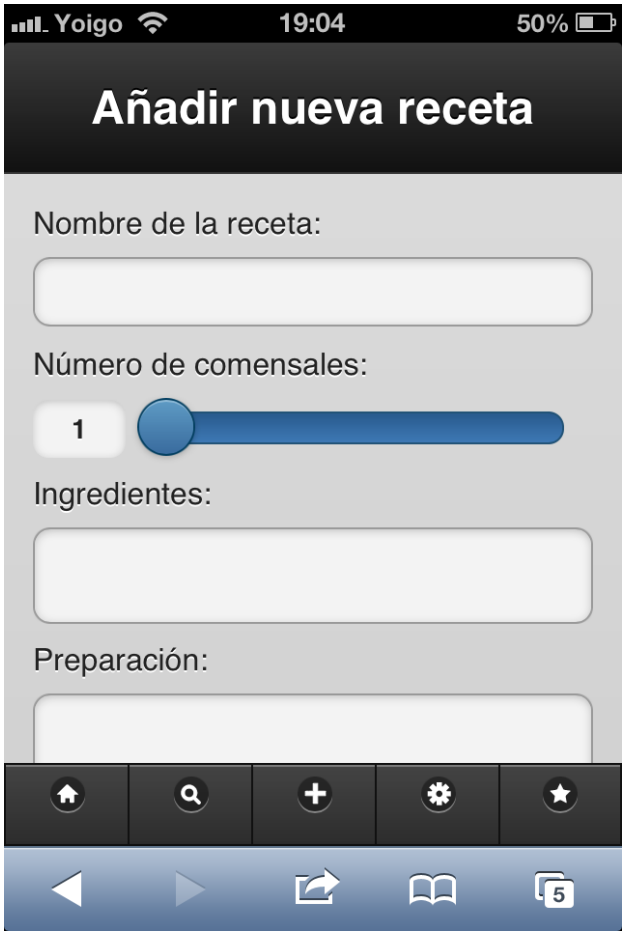
En la captura de la derecha se muestra el menú que surge por la parte derecha del dispositivo si se aprieta el menú de opciones situado en el título (mostrado en la captura anterior de la izquierda). Y como se puede observar se han agrupado aquí los casos de uso relativos a la gestión de las recetas existentes, de manera que cualquier usuario administrador (o el usuario creador de la receta) pueden modificar o eliminar la receta, así como modificar la fotografía, sin necesidad de crear un menú de búsqueda de recetas únicamente para este cometido. Resulta un método más elegante y eficiente.

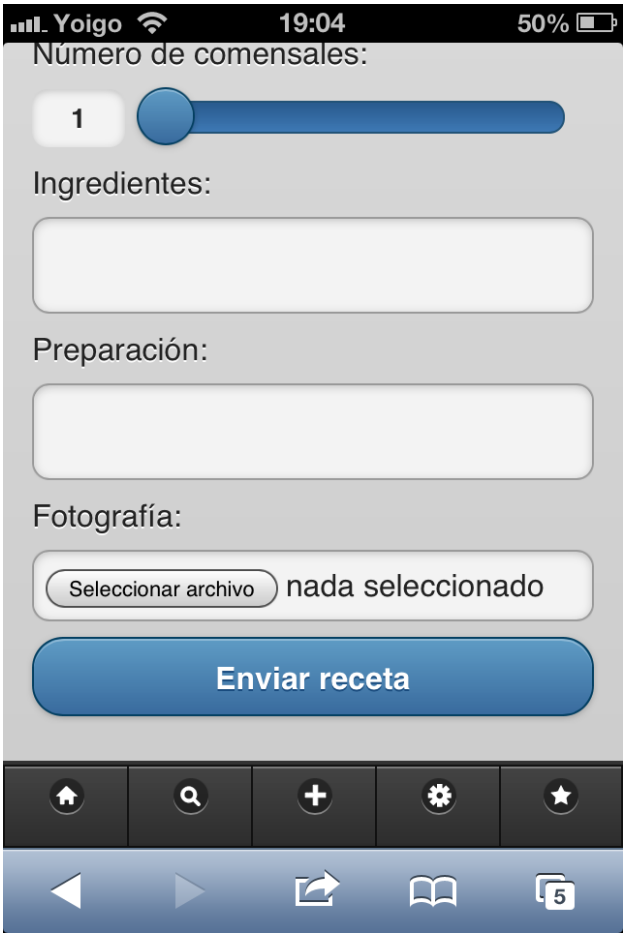

Yoigo
19:04
50%
Comentarios de los usuarios
Manolo dijo:
¡Mejor con cebolla!
Es broma, tu tortilla
está muy buena ;)
administrador dijo:
¡Pues échale cebolla!
:P


Yoigo
19:04
50%
Mostrar receta
192.168.1.5:8080/receta
Buscar
Opciones
Eliminar receta
Modificar receta
Eliminar fotografía
Ocultar panel

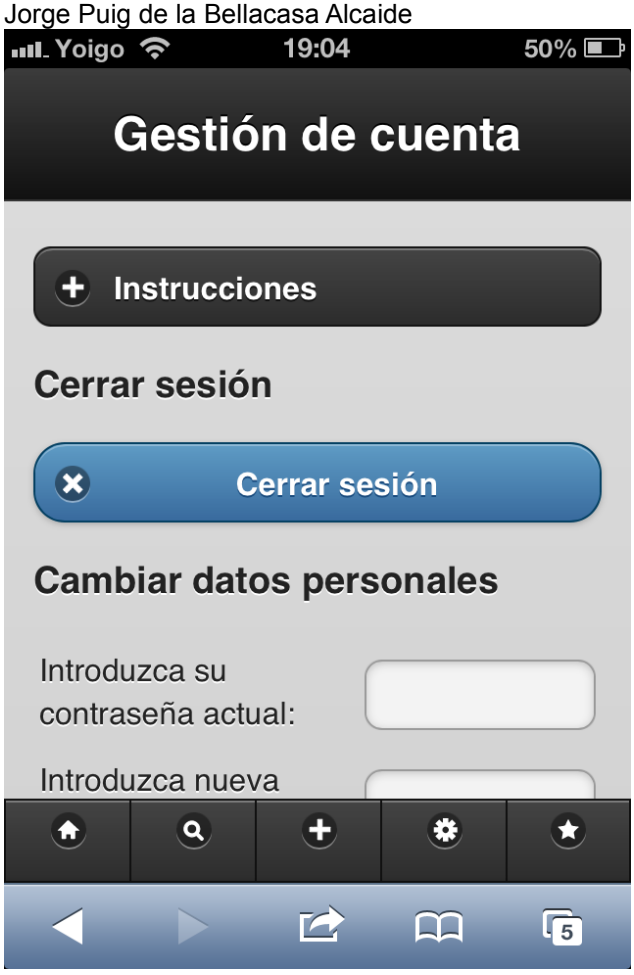
Seguidamente se muestra el formulario para la creación de una nueva receta.
Se puede observar un interesante elemento de *jQuery mobile*: un selector de número de comensales formado por un deslizador de rango que ahorra explicarle al usuario el tipo de datos que puede utilizar en esta entrada del formulario, además de ser una adición vistosa al mismo.

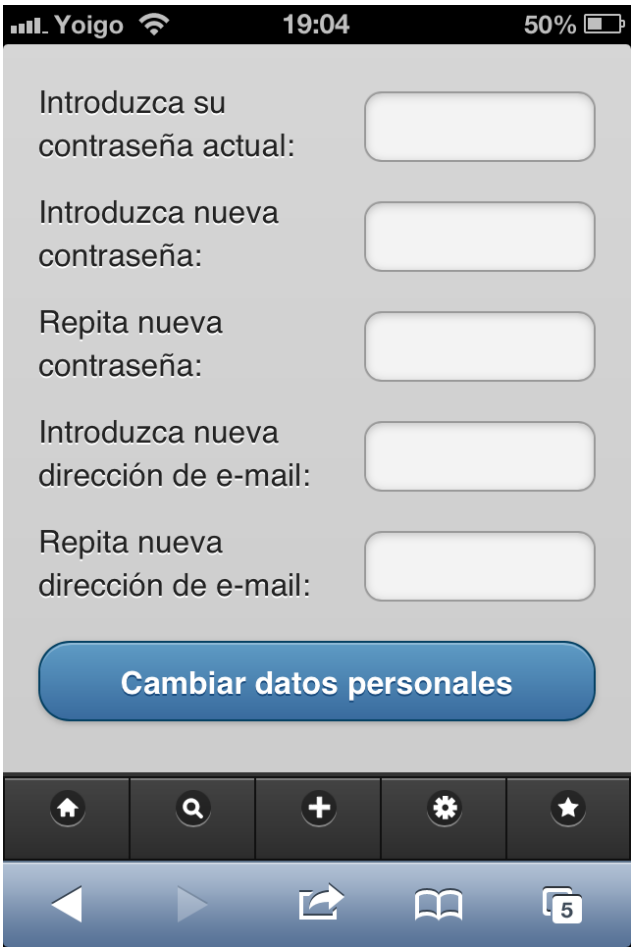
En la captura de la derecha se puede observar que el propio selector de archivo de *jQuery mobile* indica si se ha seleccionado un archivo o no, e incluso muestra un minúsculo *thumbnail* de la foto seleccionada en su caso (aunque en esta captura no se aprecia al no haber ninguna fotografía seleccionada).





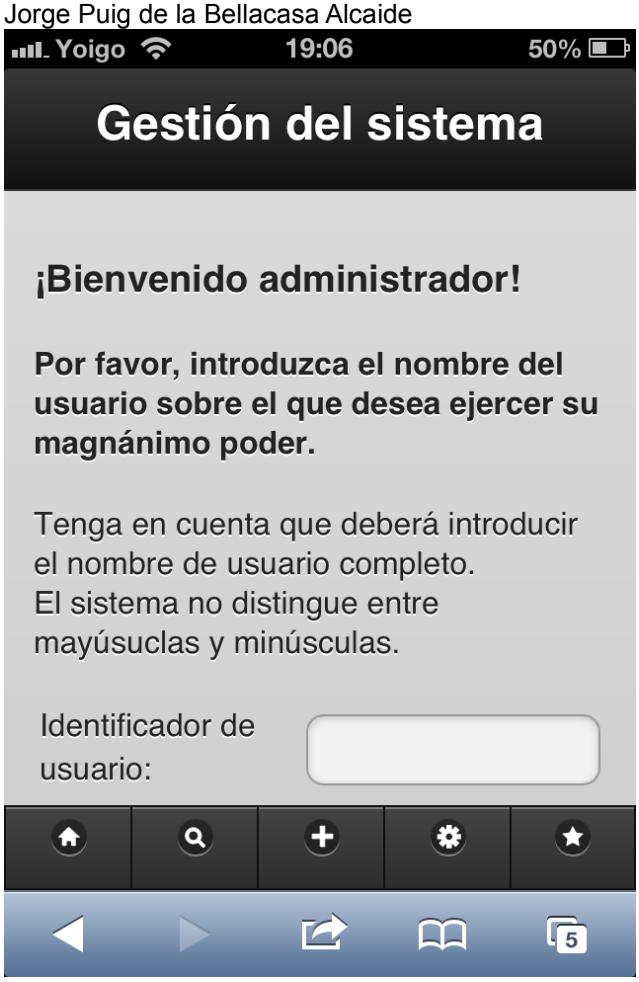
En las siguientes capturas se pueden ver el resto de pantallas de la aplicación. Los únicos elementos nuevos que se aprecian son la ventana de confirmación y la página de información.

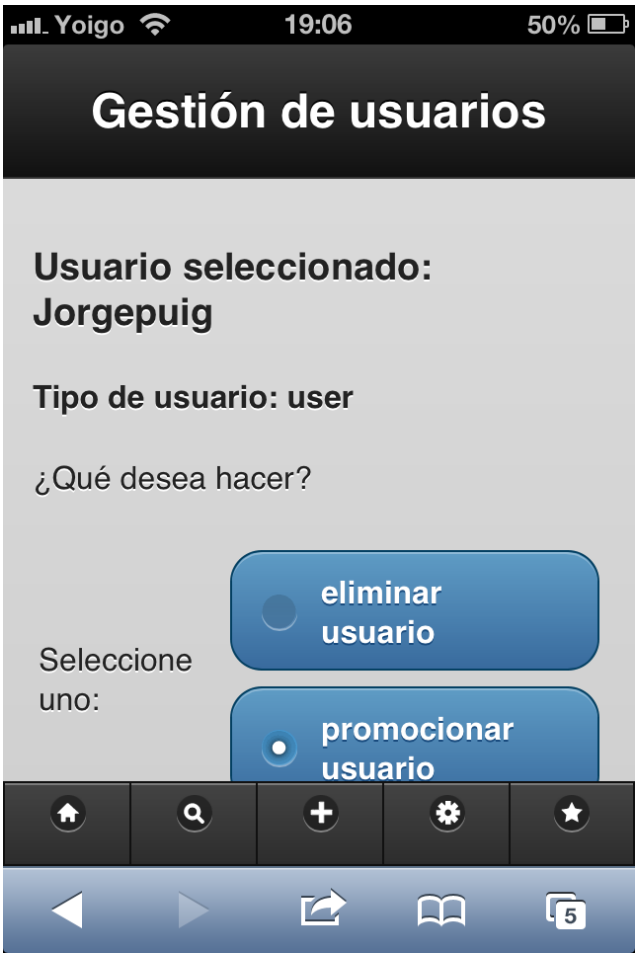

Yoigo
19:04
50%
Gestión de cuenta
Instrucciones
Cerrar sesión
Cerrar sesión
Cambiar datos personales
Introduzca su contraseña actual:
Introduzca nueva

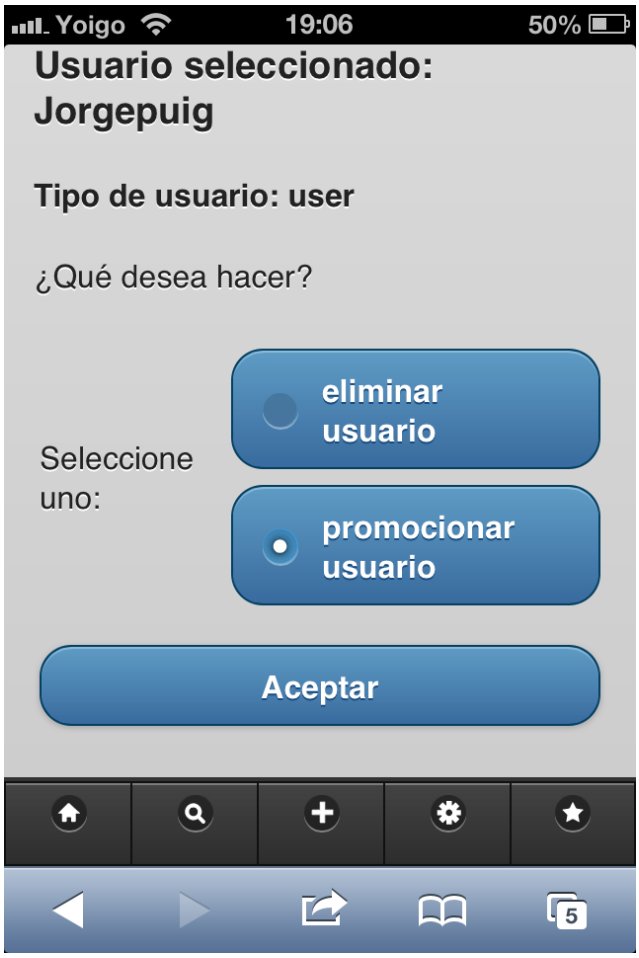

Yoigo
19:04
50%
Introduzca su contraseña actual:
Introduzca nueva contraseña:
Repita nueva contraseña:
Introduzca nueva dirección de e-mail:
Repita nueva dirección de e-mail:
Cambiar datos personales


Yoigo
19:06
50%
Repita nueva dirección de e-mail:
jorgepuig@gm
Cambiar datos personales
http://192.168.1.5
¿Seguro que quiere cambiar los datos personales?
Cancelar
OK
espacio
intro


Yoigo
19:06
50%
INFO
se han cambiado correctamente los datos de usuario


Yoigo
19:06
50%
Gestión del sistema
¡Bienvenido administrador!
Por favor, introduzca el nombre del usuario sobre el que desea ejercer su magnánimo poder.
Tenga en cuenta que deberá introducir el nombre de usuario completo.
El sistema no distingue entre mayúsculas y minúsculas.
Identificador de usuario:


Yoigo
19:06
50%
Gestión de usuarios
Usuario seleccionado: Jorgepuig
Tipo de usuario: user
¿Qué desea hacer?
Seleccione uno:
eliminar usuario
promocionar usuario


Yoigo
19:06
50%
Usuario seleccionado: Jorgepuig
Tipo de usuario: user
¿Qué desea hacer?
Seleccione uno:
eliminar usuario
promocionar usuario
Aceptar


Yoigo
19:06
50%
INFO
el usuario se ha promocionado correctamente. Ahora Jorgepuig es usuario administrador

7.4 Instalación

El programa no requiere de ningún tipo de instalación por parte del usuario. Se accede a él a través del navegador de su dispositivo como con cualquier página *web*.
Si desea probar la aplicación, estará disponible en la dirección http://recetario.no-ip.biz/recetarioFinal durante la evaluación del proyecto.

8 Conclusiones

El desarrollo de esta aplicación ha supuesto un ejercicio de autosuficiencia excelente, y dada la velocidad a la que evolucionan las tecnologías de la información, investigar y utilizar las fuentes que proporciona Internet se convierte en una necesidad esencial. Esta es sin duda la parte más positiva de la realización de este proyecto.

En cuanto a la consecución de los objetivos no puedo sino estar satisfecho. Es evidente que en un período de tiempo tan reducido no se puede entrar en profundidad a conocer tantas tecnologías, pero la realización de este proyecto sin duda ha requerido una fuerte primera toma de contacto con la mayoría de ellas, y ha revelado las dificultades derivadas de la comunicación entre tecnologías, y de la rápida (casi vertiginosa) evolución de las mismas.

El modelo arquitectónico (*MVC*) ha resultado ser una elección excelente para la realización de proyecto: con una separación por componentes y por capas que ha permitido un desarrollo independiente de cada apartado, comprensible y con las funciones de cada componente bien definidas, de manera que permite una rápida comprensión de su funcionamiento, además de una gran facilidad para su ampliación y mantenimiento.

Quedan sin embargo como tareas personales la comprensión en mayor profundidad de varias tecnologías como *servlets*, *JavaScript* y su librería *jQuery* (cuyo uso ha estado oculto bajo la librería *jQuery mobile*) y ante todo *HTML5* y todas sus nuevas funciones (*CSS3*, *canvas*, video, geolocalización, etc.).

9 Posibles mejoras futuras

Teniendo en cuenta el importante factor social de la aplicación, las mejoras futuras deberían estar orientadas hacia la interacción entre usuarios principalmente, sin desmerecer otros apartados como la presentación y la seguridad.

Con esta idea en mente se presentan las siguientes propuestas de mejora:

- Mensajes entre usuarios.
- Sistemas de votación de comentarios y recetas, que permitan detectar los contenidos más populares o aquellos que sean inapropiados.
- Presentar el contenido más popular del momento en la pantalla de bienvenida.
- Permitir buscar y ordenar recetas por popularidad.
- Sistema de avisos que informe a los usuarios de nuevos comentarios en sus recetas.
- Permitir a todos los usuarios añadir sus propias fotografías a las recetas de otros usuarios, de manera que haya un álbum de fotografías por cada receta.

En cuanto a la seguridad, aunque se han tenido en cuenta los ataques de inyección de

código (utilizando para ello la librería de seguridad *ESAPI*), este es un tema de una amplitud enorme que requiere de un estudio serio. Ya que es fundamental asegurar la integridad del sistema y especialmente la de los equipos de los usuarios.

Y en lo referente a la presentación, sería recomendable implementar una presentación más personalizada y atractiva, que diferencie a esta aplicación del resto de aplicaciones desarrolladas con *jQuery mobile*.

10 Glosario

A continuación se presenta un cuadro con los términos y acrónimos utilizados en esta memoria para su mejor comprensión.

Término	Descripción
CSS	*Cascading Style Sheets* (hojas de estilo en cascada): Lenguaje de hojas de estilos usado para describir la presentación semántica (el aspecto y formato) de un documento escrito en lenguaje de marcas[i].
DAO	*Data Access Object* (objeto de acceso a datos): Componente de software que suministra una interfaz común entre la aplicación y uno o más dispositivos de almacenamiento de datos[ii].
ESAPI	*Enterprise Security API* (librería de seguridad empresarial): Librería Java de control de seguridad para aplicaciones *web* gratuita y de código abierto[iii].
HTML	*HyperText Markup Language* (lenguaje de marcas de hipertexto): Lenguaje de marcado para la elaboración de páginas web[iv].
HTTP	*HyperText Transfer Protocol* (protocolo de transferencia de hipertexto): Protocolo estándar usado en las transacciones de la World Wide Web[v].
jasypt	*Java Simplified Encryption* (encriptación simplificada para Java): Librería Java gratuita para la encriptación de datos[vi].
Java	Lenguaje de programación orientado a objetos.
JavaScript	Lenguaje de programación orientado a objetos.
JDBC	*Java Database Connectivity* (conectividad a base de datos de Java): Librería que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java[vii].
jQuery	Librería de JavaScript que permite simplificar la manera de interactuar con los documentos HTML, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web[viii].
jQuery mobile	Librería de JavaScript centrada en la compatibilidad con una amplia variedad de *Smartphones* y *Tablet PC*[ix].
JSP	*JavaServer Pages* (páginas de servidor Java): Tecnología para la creación de páginas web dinámicas basadas en HTML y XML, entre otros tipos de documentos[x].
MVC	*Model View Controller* (modelo vista controlador): Patrón de software para la implementación de interfaces de usuario[xi].
MySQL	Sistema de gestión de bases de datos relacionales de código abierto.
POJO	*Plain Old Java Object* (objeto Java antiguo y simple): Término para referirse a objetos Java que no siguen ninguno de los modelos, convenciones o librerías de objetos Java modernos[xii].
Servlet	Clase del lenguaje Java usada para extender las capacidades de un servidor[xiii].
SGBD	Sistema de Gestión de Bases de Datos: Conjunto de programas que permiten el almacenamiento, modificación y extracción de la información en una base de datos[xiv].
SQL	*Structured Query Language* (lenguaje de consulta estructurado): Lenguaje de programación de propósito especial diseñado para manejar los datos almacenados en un sistema de gestión de bases de datos relacional[xv].
struts2	Librería Java de código abierto para el desarrollo de aplicaciones web basada en el patrón Modelo Vista Controlador[xvi].

11 Bibliografía

Freeman Elisabeth; Freeman Eric (2005). *Head First HTML with CSS and XHTML.* Sebastopol, California: O'Reilly Media, Inc.

Freeman Eric; Robson Elisabeth (2011). *Head First HTML5 Programming.* Sebastopol, California: O'Reilly Media, Inc.

Morrison Michael (2007). *Head First JavaScript*. Sebastopol, California: O'Reilly Media, Inc.

Benedetti Ryan; Cranley Ronan (2011). *Head First jQuery*. Sebastopol, California: O'Reilly Media, Inc.

Brown Donald; Davis Chad Michael; Stanlick Scott (2008). *Struts 2 in Action*. Greenwich, Connecticut: Manning Publications Co.

12 Otras fuentes de información

Wikipedia. [En línea]. http://es.wikipedia.org

Stack Overflow. [En línea]. http://stackoverflow.com/

The Apache Software Foundation. Struts 2. [En línea]. http://struts.apache.org/development/2.x/

Oracle. Java™ EE 7 specification APIs. [En línea]. http://docs.oracle.com/javaee/7/api/

jQuery mobile API. [En línea]. http://api.jquerymobile.com/

Y una miríada de foros, videos y tutoriales que me es imposible recordar, pero cuya guía y ayuda agradezco enormemente.

[i] Wikipedia (2013). Hojas de estilo en cascada. [En línea]. http://es.wikipedia.org/wiki/Hojas_de_estilo_en_cascada [fecha de consulta: 30 de diciembre de 2013].
[ii] Wikipedia (2013). Data Access Object. [En línea]. http://es.wikipedia.org/wiki/Data_Access_Object [fecha de consulta: 30 de diciembre de 2013].
[iii] OWASP (2013). OWASP Enterprise Security API. [En línea]. https://www.owasp.org/index.php/Category:OWASP_Enterprise_Security_API [fecha de consulta: 30 de diciembre de 2013].
[iv] Wikipedia (2013). HTML. [en línea]. http://es.wikipedia.org/wiki/HTML [fecha de consulta: 30 de diciembre de 2013].
[v] Wikipedia (2013). Hypertext Transfer Protocol. [en línea]. http://es.wikipedia.org/wiki/HTTP [fecha de consulta: 30 de diciembre de 2013].
[vi] Jasypt (2013). Java Simplified Encryption. [en línea]. http://www.jasypt.org/ [fecha de consulta: 30 de diciembre de 2013].
[vii] Wikipedia (2013). Java Database Connectivity. [en línea]. http://es.wikipedia.org/wiki/JDBC [fecha de consulta: 30 de diciembre de 2013].
[viii] Wikipedia (2013). jQuery. [en línea]. http://es.wikipedia.org/wiki/Jquery [fecha de consulta: 30 de diciembre de 2013].
[ix] Wikipedia (2013). jQuery Mobile. [en línea]. http://en.wikipedia.org/wiki/JQuery_Mobile [fecha de consulta: 30 de diciembre de 2013].
[x] Wikipedia (2013). JavaServer Pages. [en línea]. http://es.wikipedia.org/wiki/JavaServer_Pages [fecha de consulta: 30 de diciembre de 2013].
[xi] Wikipedia (2013). Model-view-controller. [en línea]. http://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller [fecha de consulta: 30 de diciembre de 2013].
[xii] Wikipedia (2013). Plain Old Java Object. [en línea]. http://en.wikipedia.org/wiki/POJO [fecha de consulta: 30 de diciembre de 2013].
[xiii] Wikipedia (2013). Java Servlet. [en línea]. http://en.wikipedia.org/wiki/Servlet [fecha de consulta: 30 de diciembre de 2013].
[xiv] Wikipedia (2013). Sistema de gestión de bases de datos. [en línea]. http://es.wikipedia.org/wiki/SGBD [fecha de consulta: 30 de diciembre de 2013].
[xv] Wikipedia (2013). SQL. [en línea]. http://en.wikipedia.org/wiki/SQL [fecha de consulta: 30 de diciembre de 2013].
[xvi] Wikipedia (2013). Apache Struts. [en línea]. http://en.wikipedia.org/wiki/Struts2 [fecha de consulta: 30 de diciembre de 2013].