Escola Universitària Politècnica de Mataró

TecnoCampus Mataró-Maresme

Desarrollo de la v3 de javaHispano

Alumno: Xavier Lluch i Urrutia
Ponente: Enric Sesa Nogueras

Primavera 2010

Dedicatoria

Este trabajo lo dedico a mis padres y hermanos, a toda mi familia y a mis amigos, que durante mi vida, me han dado soporte para llegar donde estoy ahora. También a aquellas personas que me han acompañado durante el tránsito de mi carrera.

Agradecimientos

Quiero expresar mis agradecimientos a las personas que han sido una fuente de ayuda y motivación para mí. Yolanda, Sergi, Sergio, etc. Gràcies.

Me gustaría dar las gracias especialmente a mi tutor Enric Sesa Nogueras y a aquellos que sin su presencia no estaría presentando este trabajo.

Resumen

La web javaHispano es un portal social de temática tecnológica centrada en todo aquello que esté relacionado con el lenguaje de programación Java. javaHispano es una comunidad de habla hispana y todos los recursos que allí se encuentran, están en este idioma.

Debido a limitaciones en el framework utilizado actualmente, se ha iniciado un nuevo proyecto de reingeniería del sistema gestor de contenidos. Pretende ser un proyecto de código libre aplicando algunas de las tecnologías más conocidas y actuales.

Algunos de los objetivos marcados en el proyecto inicial son implementar el sistema de seguridad básico, la autenticación y gestión del usuario, maximizar la usabilidad y la interacción del usuario con el sistema.

El resultado es una aplicación potente, robusta, reutilizable y escalable. Además, pretende ser una herramienta instructiva y un proyecto de referencia para aquel que esté interesado en ver aplicada alguna de las tecnologías más modernas.

Resum

La web javaHispano és un portal social de temàtica tecnològica centrada en tot allò que estigui relacionat amb el llenguatge de programació Java. javaHispano és una comunitat de parla hispana i tots els recursos que s'hi troben, estan en aquest idioma.

A causa de limitacions en el framework utilitzat actualment, s'ha iniciat un nou projecte de re enginyeria del sistema gestor de continguts. Pretén ser un projecte de codi lliure aplicant algunes de les tecnologies més conegudes i actuals.

Alguns dels objectius marcats en el projecte inicial són implementar el sistema de seguretat bàsic, l'autenticació i gestió de l'usuari, maximitzar la usabilitat i la interacció de l'usuari amb el sistema.

El resultat és una aplicació potent, robusta, reutilitzable i escalable. A més, pretén ser una eina instructiva i un projecte de referència per a aquell que estigui interessat en veure aplicada alguna de les tecnologies més modernes.

Abstract

The web javaHispano is a social portal about technology focused on everything that is related to the Java programming language. javaHispano is a hispanic community and all resources found there are in this language.

Due to limitations in the framework used today, a new reengineering project of the content management system has begun. It aims to be an open source project using some of the best known and most current technologies.

Some of the targets set in the initial project are implementing the basic security system, authentication and user management, maximize usability and user interaction with the system.

The result is a powerful, robust, reusable and scalable application. It also aims to be a teaching tool and a reference project for those interested in seeing implemented some of the latest technology.

Índice

Índice de figuras

1 Introducción

1.1 Planteamiento

La web javaHispano[1] (jH) es un portal creado en 2001, por la asociación de aficionados a la programación en lenguaje Java. El objetivo de este proyecto, fue crear una comunidad de habla hispana donde se pudiera reunir la máxima información posible sobre Java en español. jH fue fundada con la finalidad de contener todas aquellas noticias de actualidad, artículos, eventos, tutoriales, foros de debate y consultas relacionadas con el lenguaje de programación Java creado por Sun Microsystems.

jH es una asociación sin ánimo de lucro. Se caracteriza por ser una comunidad libre, es decir, que no es propiedad de ninguna persona ni empresa.

Actualmente jH es una web "meritocrática". Esto significa que aquellos usuarios interesados en desarrollar, administrar y/o aportar sus conocimientos para mejorar y enriquecer el contenido de alguna parte de esta comunidad, deben ganarse la confianza del equipo de administradores. Siendo los méritos propios los que permitan, a los que lo merezcan, realizar tareas de mayor responsabilidad.

El desarrollo de la nueva versión del portal jH es un proyecto de código libre. Un usuario puede desarrollar cualquier tipo de funcionalidad y, si es aprobado por los administradores de desarrollo, será implantada en producción.

Existen varios motivos por los que se ha decidido migrar a la nueva versión. Uno de ellos es que el portal se está quedando atrás en el tema de colaboración e interacción social. Por ese motivo se ha decidido hacer una re ingeniería y tener en cuenta estos aspectos a la hora de diseñar la nueva arquitectura del sistema.

Otra de las razones principales por la que se ha decidido migrar el portal a la versión 3 es la exclusividad de este framework, es decir, al ser un proyecto privado creado por

una empresa, contiene partes de código perteneciente a esta por lo que imposibilita la apertura del código por completo.

También la carencia de documentación y evolución, ha hecho que sea más difícil que algún otro usuario pueda participar desarrollando nuevas funcionalidades. Esto hace que los usuarios tengan que especializarse en una herramienta que no se utiliza en ningún otro sitio.

Es por eso que se ha decidido desarrollar toda la arquitectura de la nueva versión utilizando el framework Spring v3.0 de SpringSource, y de esta manera, al ser un framework tan extendido, es más fácil que el equipo de desarrollo sea más grande y deje de depender de una sola persona.

Este trabajo de final de carrera, tiene como objetivo reflejar el trabajo realizado como miembro del equipo de desarrollo, para la creación de una nueva versión de código libre del portal javaHispano.

En este trabajo se refleja la definición y desarrollo del registro y gestión del usuario, la seguridad de acceso del usuario a los recursos de la aplicación, la integración del uso de algunos recursos externos cómo el sistema de verificación de usuarios OpenId, imágenes de perfil de Gravatar o perfil profesional de LinkedIn. También se quiere mejorar los aspectos de interacción social entre usuarios y de la motivación para que estos aporten mas contenidos a la comunidad jH y con mejor calidad.

1.2 Evolución histórica de jH

Desde la creación de la comunidad jH, en el año 2001, se ha animado a los usuarios a aportar sus conocimientos y dudas para formar parte del grupo de programadores que lo han hecho crecer hasta el día de hoy. Actualmente no queda ningún de los miembros del grupo original que creó el portal.

La primera versión de jH fue un grupo de Yahoo[2] muy simple a principios del año 2001. En agosto ya disponían de 500 subscripciones a las noticias y a finales de año el portal es programado en Java.

En el año 2002 se añadió la posibilidad de hacer una búsqueda en el site mediante el buscador Google. También se añadió las noticias mediante RSS y el mapa web.

Figura I: Aspecto de la v1 de jH el año 2003

El primer congreso vino a finales noviembre del 2003 y desde entonces la comunidad empezó a crecer y a añadir entrevistas a profesionales del mundo Java. En diciembre del año siguiente se organizó un segundo congreso de jH.

En el año 2007 se escogió un framework para albergar portales y se puso en marcha la versión 2 de la comunidad. En la actualidad se puede añadir podcasts sobre noticias, entrevistas y opiniones.

Figura II: Aspecto de la v2 de jH (En la actualidad)

1.3 Proyecto Cáñamo

El proyecto Cáñamo[3] es un portal y un Sistema de Gestión de Contenidos (CMS) creado por la empresa New High Technologies of Norwick S.L. (Bilbao, España). Posteriormente fue utilizado para elaborar la segunda versión de jH. Cáñamo está

desarrollado en lenguaje Java y es similar a Nuke (un framework de PHP), con soporte para cubrir las necesidades de un portal como:

- Gestión de usuarios.
- Encuestas.
- Soporte RSS.
- Foros.
- Listado de elementos (Noticies, etc.).
- Artículos.
- Edición de contenidos online.

Algunas de las características más destacables del proyecto Cáñamo son:

- Sigue el patrón Modelo Vista Controlador
- Soporta diferentes tecnologías para generar la presentación: plantillas HTML, XSLT o JSP.
- Dispone de multitud de servicios integrados como la gestión de persistencia, mailing, etc.

2 Objetivos

Este trabajo tiene varios objetivos, tanto a nivel del propio proyecto de desarrollo cómo a nivel personal.

El portal jHv3 además de ser un nuevo sistema de gestión de contenidos con una interacción con el usuario más ágil e intuitiva, pretende ser un proyecto de referencia sobre tecnologías modernas y buenas prácticas para que otros desarrolladores puedan usarlo como ejemplo práctico de aplicación web en Java. Más adelante se describirán las tecnologías utilizadas.

Partiendo de las ideas aportadas por el grupo de desarrolladores que participan en el proyecto jHv3, se han obtenido una serie de objetivos mediante una serie de reuniones con el arquitecto principal.

Algunos de estos objetivos son:

- Realizar un análisis detallado del portal para un mejor entendimiento del funcionamiento actual de la comunidad, posibilitando así un mejor diseño. Interesa que la nueva versión del portal mantenga muchas de las funcionalidades actuales.

- Dividir el diseño por funcionalidades modulares, priorizar estos módulos y repartir las tareas va a realizar cada miembro en la creación de la versión inicial.

- Realizar la formación necesaria para manejar algunas las tecnologías que se manejan, tales como la configuración del proyecto con Maven, Spring MVC, motor de plantillas FreeMarker, Spring Security o REST.

- Generar los casos de uso de las funcionalidades que afectan principalmente al usuario.

- Crear el esquema de la base de datos y, sirviéndonos del framework iBatis 2, desarrollar aquellas partes que permitan a la aplicación manejar los usuarios y comunicarse con la base de datos.

- Establecer y configurar las directivas de seguridad de la aplicación utilizando el framework Spring Security 3.

- Implementar la lógica de negocio que dará servicio y los controladores con el framework Spring MVC 3.

- De forma paralela al punto anterior se realizarán las unidades de test para verificar y garantizar el correcto funcionamiento de dichas implementaciones.

- También se tienen que implementar con FreeMarker unas sencillas plantillas encargadas de generar el código HTML y así poder ir interactuando con el sistema de forma gráfica.

- La codificación tiene que tiene que ser limpia, usando buenas formas de programación. Todos lo métodos y clases tienen que estar correctamente comentados ya que también se trata de un proyecto de código libre.

A nivel personal, espero ganar experiencia profesional dentro del mundo del código libre mediante el trabajo en equipo, el uso de una metodología de trabajo y un conjunto de tecnologías punteras en el sector. Todo esto se convertirá en una sólida base de conocimientos de las técnicas de desarrollo web.

3 Análisis de situación actual

Para poder hacer una buena especificación de requerimientos para la nueva versión de la comunidad, es necesario tener bien claros los apartados que la forman actualmente. Además se debe tener bien claro cuál es el funcionamiento y los protocolos que sigue la aplicación actualmente.

Desde los inicios de la versión 1, el diseño del sistema ha permitido grabar todos los movimientos (clics) que los usuarios hacen al navegar por la comunidad. Esto ayudó a diseñar la v2 con aquellos apartados que eran más relevantes o aquellos que eran más visitados.

Los administradores vieron que algunos aspectos de la web eran limitados:

- Muchas secciones estaban muertas por el desuso.
- Subir un artículo o tutorial era una tarea muy engorrosa.
- Al ser un portal hecho por y para programadores, había ciertas limitaciones, por ejemplo, en la clasificación de noticias.

Para solucionar todo esto se redujeron las secciones, se añadió un editor de texto para facilitar y mejorar la redacción de noticias y artículos. También se añadió un sistema de moderación y reporting para eliminar aquellos artículos/tutoriales que no interesa que estén publicados.

Se realizaron cambios en el diseño visual haciendo que éste fuera más agradable para el usuario. Para ello se utilizaron estándares actuales como XHTML y CSS 2 que permiten una mejor accesibilidad desde todo tipo de dispositivos.

La estructura básica de navegación de los contenidos del portal quedó de la siguiente manera:

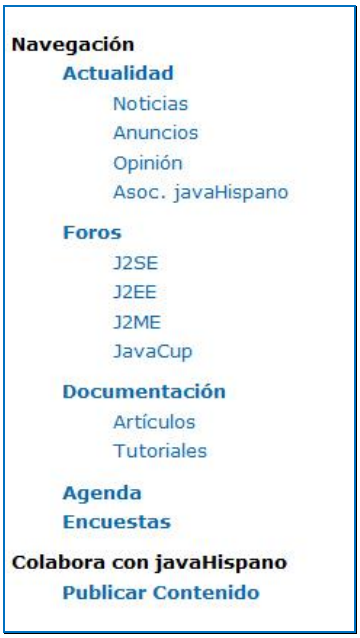


Figura III: Mapa web de la v2 jH

3.1 Página de inicio (Home)

La home es la página de inicio que se muestra cuando un usuario se conecta al portal. Esta se puede dividir en cuatro partes:

- **Destacados**: En esta sección se muestran las noticias más destacadas como pueden ser un acontecimiento especial o un artículo que ha recibido muchos votos por los usuarios de la comunidad.
- **Últimas noticias**: En esta sección aparecen las últimas noticias publicadas por los usuarios en la comunidad.
- **Foros jH**: En esta sección se muestran las últimas entradas introducidas en los foros.
- **Planeta jH**: Muestra las últimas entradas del blog de jH. Además es un repositorio que se alimenta de entradas de otros blogs.

3.2 Actualidad

3.2.1 Noticias

En el apartado de noticias se muestran aquellas noticias sobre sucesos, eventos, etc., que son de interés para los usuarios.

3.2.2 Anuncios

Esta sección está dedicada principalmente a eventos que tengan lugar próximamente, como puede ser talleres de programación, conferencias y charlas (coloquios/jornadas) tecnológicas y encuentros sociales.

3.2.3 Opinión

Cuando a algún usuario le interesa expresar la visión que tiene sobre algún tema en particular relacionada con el mundo Java puede hacerlo en esta sección.

También se ofrece la posibilidad a los usuarios de abrir debates sobre cualquier otro tema que le pueda interesar para que la gente pueda expresar su opinión al respecto.

3.2.4 Asociación jH

Cuando la comunidad jH, ya sea como asociación en general o uno de sus miembros importantes en particular, crea algún podcast, organiza o patrocina algún acontecimiento, éste se publica dentro de esta sección.

3.3 Foros

El foro de jH tiene el funcionamiento típico de un foro, dividido en algunos subtemas concretos.

Dispone de un buscador donde el usuario introduce lo que quiere buscar dentro del contenido de cualquier tema. Actualmente el buscador presenta algunos defectos, como por ejemplo, que no ordena con ningún tipo de criterio los resultados, ni dispone de un sistema de paginación tampoco, cosa que dificulta encontrar aquello que el usuario busca.

Un punto a favor son los feeds, fuentes de información web, que se generan cada vez que se contribuye en alguno de los temas. De esta forma el usuario puede recibir información actualizada en alguno de los lectores de noticias que hay en el mercado sin necesidad que se conecte al portal jH.

Los temas que componen el foro de jH son fijos y son:

- **J2SE**: Foro dedicado a Java 2 Standard Edition.
- **J2EE**: Foro para preguntas sobre Java 2 Enterprise Edition.
- **J2ME**: Todo sobre Java 2 Micro Edition, es decir, Java en dispositivos móviles.
- **XML**: Foro dedicado en el que se encuentran preguntas sobre el tratamiento de XML con Java.
- **Certificación**: Foro dedicado a todos los términos de certificación de la plataforma Java en todas sus variantes.
- **javaHispano**: Aquí están las preguntas, quejas, sugerencias, etc., que tienen los usuarios sobre jH.
- **Bugs en javaHispano**: Si el usuario encuentra algún fallo en el sistema de jH lo comenta en este foro.
- **Foro antiguo sobre java**: Aquí hay almacenados todos los mensajes que habían en el antiguo foro de jH. Está sólo para consultar y no admite nuevos mensajes.
- **Desarrollo de juegos**: Foro dedicado al desarrollo de juegos en Java.
- **Congreso javaHispano**: En este foro los usuarios pueden dar su opinión sobre los congresos que organiza la comunidad, sugerencias, temas a tratar, recomendaciones, preguntas y respuestas, etc.

- **Apoyo al curso J2SE:** Foro de soporte para hacer preguntas referentes al curso de J2SE alojado en jH (Curso J2SE).
- **Persistencia:** Foro para todo aquello relacionado con la persistencia de objetos en Java. Motores de persistencia, bases de datos relacionales y orientadas a objetos, etc.
- **IDEs**: Foro específico para comentarios sobre los diferentes IDEs del mercado (Eclipse, Netbeans, etc.).
- **JavaCup:** Foro sobre la javaCup. Un torneo de programación de tácticas de futbol en lenguaje Java.
- **Empleo:** Foro moderado para ofertas de trabajo. Es obvio que todas las ofertas tienen que ser relacionadas con el mundo Java.

3.4 Documentación

Accediendo al menú Documentación, un usuario puede hacer una aportación con la publicación de un curso, tutorial, artículo o explicación de algún tema.

Dentro del detalle de cada artículo o aportación, el usuario dispone de un apartado donde puede descargar los archivos (documento, código fuente, transparencias, etc.) adjuntados por el creador de éste. Todo usuario puede hacer comentarios sobre el curso o tutorial publicado.

3.5 Agenda

Los usuarios disponen de una agenda en la que figuran aquellos eventos que organiza la comunidad o que son de interés general. Para cada uno de ellos se informa de la descripción del evento, la fecha y la ubicación donde tendrá lugar. Las entradas en la agenda son introducidas por los administradores del portal.

3.6 Encuestas

Periódicamente la comunidad realiza una pequeña encuesta a los usuarios con la finalidad de recoger información general de la comunidad y, posteriormente, se publica una noticia o podcast sobre un tema relacionado con ésta.

La encuesta está compuesta de una serie de preguntas y los usuarios pueden escoger entre varias respuestas.

Las encuestas están disponibles dentro de la sección de actualidad en una de las columnas laterales. Haciendo clic sobre el titulo “Encuestas” se puede acceder a todas las encuestas que se están realizando y las que están ya cerradas. De esta manera el usuario puede consultar los resultados de éstas.

3.7 Colaboración en jH

Con la finalidad de agilizar la publicación de anuncios, opiniones, noticias o podcasts, el sistema incluye un editor de contenidos (CMS). Este editor permite escoger uno de los tags principales JavaSE, JavaEE, JavaME, XML o Otros, asignar también unas etiquetas para facilitar la búsqueda posterior y un pequeño, pero útil, editor de texto con un conjunto de funciones que permite dar al texto un formato más atractivo.

3.7.1 Nube de etiquetas

La *nube* de etiquetas permite al usuario filtrar aquellos elementos que más le interesa de todos los que se están mostrando en aquel momento. Estas etiquetas se asignan al documento en cuestión en el momento de editarlo.

Cuando se muestran los diferentes elementos del portal (artículos, noticias, etc) dentro del cuerpo de la web aparecen los nombres de las etiquetas en cuestión y el número de veces que éstas aparecen en los elementos en cuestión.

3.7.2 Votación de elementos

El sistema de votaciones permite promocionar noticias y artículos para que se puedan mostrar en el apartado de "Destacados". Este sistema de votaciones sólo está disponible para los usuarios destacados, que pueden asignar, según los privilegios que tengan, una determinada puntuación a un elemento.

3.7.3 Podcasts

El sistema de podcasts permite divulgar entrevistas, debates y entrevistas en formato de audio.

Desde la puesta en marcha del sistema de podcasts, la comunidad ha publicado más de 70 entrevistas, debates y noticias en este formado. Aunque a este sistema le falta una buena herramienta de acceso, ha tenido bastante éxito entre los usuarios.

3.8 Contactar

Cuando un usuario, esté registrado o no, quiere ponerse en contacto con la administración de jH, puede hacerlo entrando dentro del apartado "Contactar" situado en la cabecera de la web.

Si el usuario esta registrado y ha iniciado sesión, su nombre y dirección de correo electrónico se informan automáticamente.

Los destinatarios del mensaje se pueden escoger entre tres posibles: "Ayuda/Soporte & Webmaster", "Marketing/Publicidad" o "Acuerdos y colaboraciones".

3.9 Buscador de la comunidad

El buscador se encuentra dentro de la cabecera del portal y permite la búsqueda del concepto que el usuario introduce dentro del contenido de toda la comunidad, exceptuando el contenido de los foros.

Aunque el buscador realiza la función para la que está diseñado, mostrando los resultados que coinciden con los criterios introducidos, no muestra los resultados de una forma clara para el usuario. Por cada resultado se muestra en que sección se puede encontrar y el título del elemento en cuestión.

La falta de información adicional dificulta al usuario poder discriminar entre los resultados obtenidos cuales son de su interés y esto hace que sea una funcionalidad poco accesible.

4 Análisis

4.1 Especificación de requisitos

Como ya se ha comentado en la introducción, se quiere crear un nuevo portal social tecnológico. Respecto a la versión anterior se quiere mejorar el acceso del usuario a los recursos de la comunidad y, por eso, se le quiere dar mucho valor a como va a interactuar éste con el sistema.

Teniendo en cuenta esto se quiere mejorar el sistema de autenticación de usuarios con algún sistema estandarizado que les permita utilizar un tipo de identificación global. Si observamos algunas de las competencias como www.stackoverflow.com[4], ya hace tiempo que utilizan este tipo de autenticación para identificar a sus usuarios mediante el sistema OpenId.

Pero jH no se conforma con esto, sino que quiere ir un paso más allá en cuestión de usabilidad. En el momento en que un usuario no registrado quiere ingresar en el sistema, este lo redirecciona al formulario de registro con sus datos ya reflejados.

De esta forma el usuario no tiene que rellenar ningún campo, sólo verificar que los datos recuperados por el sistema son correctos. Así se consigue que el proceso de registro sea más ágil.

Otro inconveniente de una comunidad donde todo el mundo es libre de dar su opinión sobre un tema, son los famosos usuarios anónimos ya que en ocasiones estos pueden crear malestar con aportaciones malintencionadas o de carácter ofensivo.

Por esto motivo se quiere definir un tipo de usuario nuevo al que se le llamará “Usuario-Cookie”. Estos usuarios serán poseedores de una cuenta que tendrá un tiempo de vida limitado y, para poder utilizarla, tendrán que suministrar un nombre de usuario y una dirección de correo electrónico.

Los usuarios que ya están registrados actualmente tienen el derecho de mantener todos sus datos y contenidos sin perder ninguna información. Pero como el sistema de gestión de usuarios va a ser diferente, tendrán que registrarse de nuevo con una cuenta OpenId. Una vez hecho esto pueden importar todos los datos del usuario que poseían en jHv2 identificándose.

Con este planteamiento el sistema contemplará cuatro tipos de usuarios:

- Usuario no registrado
- Usuario registrado con OpenId
- Usuario registrado con Cookie
- Usuario registrado jHv2. Este último tipo de usuario no interactuará con el sistema, sino que su única misión será mantener los datos de los usuarios que actualmente existen en jH a la espera de que un usuario registrado con OpenId importe sus datos.

Independientemente del tipo de usuario que se utiliza para entrar al sistema, todos ellos tendrían que tener las siguientes funcionalidades:

- **Lectura de contenidos:** El usuario podrá visualizar los contenidos añadidos por los usuarios.
- **Visualizar perfiles de otros usuarios:** El usuario podrá ver los datos personales de un usuario. Desde el perfil de estos usuarios también podrá navegar a sus perfiles públicos de LinkedIn y Twitter.

Los usuarios no registrados, tendrían que tener, además de las funcionalidades comunes comentadas anteriormente, las siguientes funcionalidades:

- **Registrarse como Usuario-Cookie:** Un usuario puede obtener una cuenta temporal para poder realizar aportaciones con sólo un nombre de usuario y una dirección de correo electrónico.
- **Registrarse como Usuario-OpenId:** Con sólo tener una cuenta con algún proveedor de OpenId un usuario podrá realizar el registro.

Los usuarios registrados tanto por Cookie como por OpenId tendrían que tener, además de las funcionalidades comunes comentadas anteriormente, la funcionalidad:

- **Editar perfil de usuario:** El usuario podrá modificar sus datos personales, entre otros su identificador de Twitter y la URL de LinkedIn.

Los usuarios registrados con OpenId que tuvieran una cuenta jHv2 tendrían que tener, además de las funcionalidades comunes comentadas anteriormente, las siguientes funcionalidades:

- **Vincular usuario jHv2 con usuario jHv3:** Un usuario OpenId puede importar sus datos personales, contenidos etc. que tenia como usuario jHv2. Para hacerlo, simplemente tendrá que verificarse como tal en el sistema.

En una segunda fase de análisis, se han buscado formas de motivar al usuario a interactuar más con el sistema. Estudiando algunos de los sistemas que existen se ha optado por estudiar la posibilidad de implantar un sistema de medallas (en inglés badge) como hace http://www.osqa.net/[5].

Este sistema de medallas consiste en gratificar al usuario por realizar ciertas acciones que enriquezcan el contenido de la comunidad (aportaciones de contenido, comentar contenidos de otros usuarios, etc.).

Estas medallas son premios simbólicos que aparecen reflejados en el perfil del usuario y que hacen que este se sienta motivado a conseguir más para destacar sobre el resto de usuarios.

Para establecer los diferentes tipos de medallas que se pueden entregar a los usuarios del sistema y cuales son los requisitos para ganarlas se ha realizado una sesión de lluvia de ideas (brainstorming).

Para la versión inicial de jHv3 se ha decidido establecer unos cuantos tipos de medallas a otorgar a los usuarios a las que se irán añadiendo algunos tipos nuevos en próximas iteraciones.

Los diferentes tipos de medallas que se contemplan en la versión inicial son:

- **Perfil Completado:** Cuando un usuario completa todos los datos de su perfil.
- **Primera aportación de contenido:** Cuando un usuario colabora aportando algún contenido.
- **Realizar 50 aportaciones:** Cuando un usuario ha colaborado con 50 aportaciones de contenido.
- **Realizar 250 aportaciones:** Cuando un usuario colaborado con 250 aportaciones de contenido.
- **Primer comentario en una aportación**: Cuando un usuario deja su primer comentario en algún contenido.
- **Realizar 50 comentarios en aportaciones:** Cuando un usuario ha comentado 50 veces.
- **Realizar 250 comentarios en aportaciones**: Cuando un usuario deja su primer comentario en algún contenido.

El número de aportaciones y comentarios necesarios para recibir dichas medallas son temporales ya que este sistema aún esta en desarrollo y no es definitivo.

4.2 Diagrama de casos de uso

El siguiente diagrama UML muestra los casos de uso han surgido del análisis realizado. También podemos ver que tipo de actores interactuarán con el sistema jH y que casos de uso pueden acceder estos.

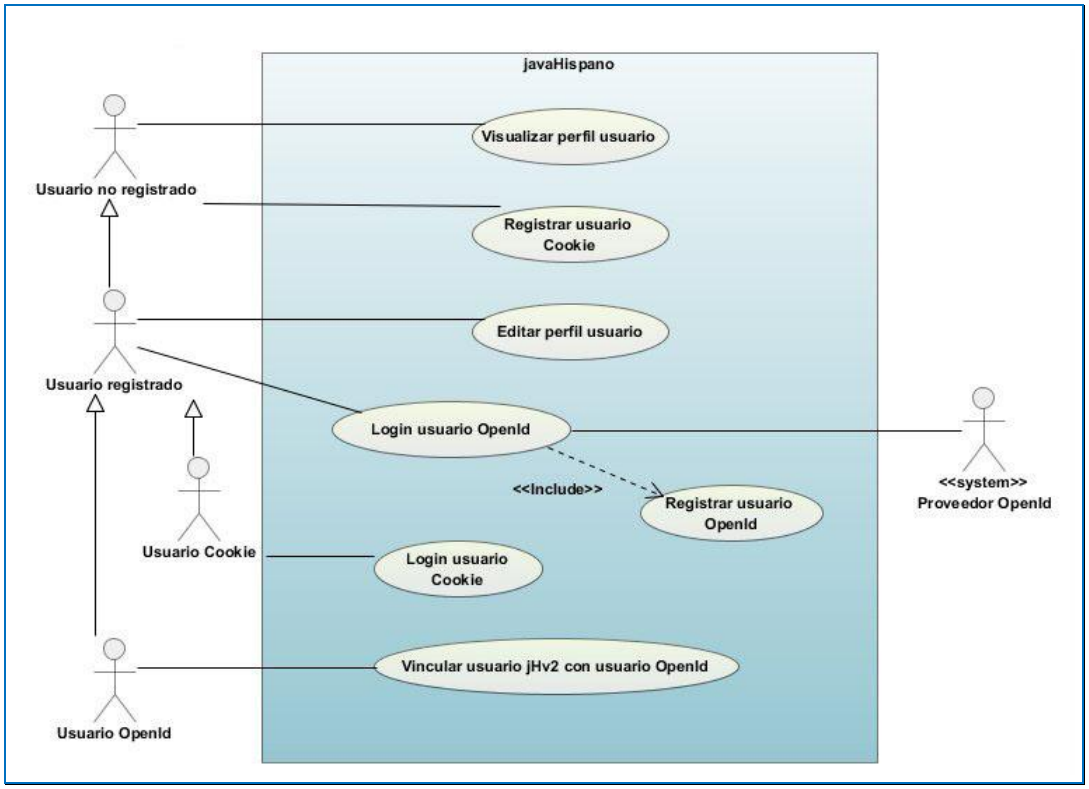


Figura IV: Diagrama de casos de uso

4.3 Especificación de casos de uso

Los casos de uso hallados en el diagrama anterior, los podemos especificar de la siguiente manera:

- **Visualizar perfil de usuario:** Cualquier usuario puede consultar el perfil de otro usuario. Si este está registrado puede consultar el suyo propio.

- **Registrar usuario-Cookie:** Un usuario no registrado puede obtener este tipo de cuenta para interactuar con el sistema.

- **Editar perfil de usuario:** Cualquier usuario registrado puede editar los datos de su propio perfil.

- **Login de usuario con OpenId:** Un usuario-OpenId puede ingresar en el sistema.

- **Registrar usuario con OpenId:** Cualquier usuario que tenga una cuenta en un proveedor de OpenId puede registrarse en el sistema.

- **Login usuario con Cookie:** Mientras el usuario tenga en su sistema la cookie generada por jH o no haya expirado puede ingresar en el sistema.

- **Vincular de usuario jHv2 con usuario OpenId:** Los usuarios-OpenId que tuvieran una cuenta en jHv2 pueden importar su antigua cuenta a la nueva.

4.3.1 Registrar usuario con OpenId

Camino directo.

El usuario no registrado o usuario-Cookie decide que quiere registrarse e inicia el proceso.

1. El sistema le muestra al usuario los proveedores más comunes de OpenId que existen.
2. El usuario escoge el proveedor que quiere utilizar para verificarse (Ej.: Google, OpenId URL o Abrir-se una cuenta en OpenId)
3. El sistema redirecciona la petición al sistema remoto de OpenId escogido por el usuario y este le pide que inicie sesión.
4. El usuario inicia sesión en el sistema remoto y le da permiso para dar información al sistema jH.
5. El sistema de OpenId devuelve una URL de verificación del usuario junto con algunos datos del usuario que el sistema jH le ha requerido.
6. El sistema verifica que no hay un usuario registrado con esa misma URL de OpenId y muestra un formulario de registro básico con los datos devueltos por el

proveedor de OpenId el nombre de usuario, nombre de pila, apellidos y dirección de correo electrónico.
7. El usuario rellena aquellos campos obligatorios que el sistema no haya informado y acepta el registro.
8. El sistema hace la asignación de rol básica, genera la URL de Gravatar a partir de la dirección de correo electrónico y almacena el nuevo usuario.
9. El sistema inicia la sesión del usuario automáticamente y este es redireccionado a la pantalla de inicio.

Alternativas.

* Del paso 1 al 7 el usuario puede cancelar el proceso de registro.

6b. El sistema detecta que el usuario ya existe en el sistema e inicia sesión automáticamente.

4.3.2 Vincular de usuario jHv2 con usuario jHv3

Camino directo.

El usuario ya estaba registrado antes de implantar OpenId y quiere vincular su registro OpenId con su cuenta de la versión 2 de jH.

1. El usuario selecciona la opción de recuperación de datos de jHv2.
2. El sistema le pide el nombre de usuario y contraseña de su cuenta jHv2.
3. El sistema verifica que existe esa cuenta y que las credenciales proporcionadas son correctas.
4. El sistema realiza todos los procesos necesarios para enlazar todos los datos de la cuenta jHv2 del usuario con su nueva cuenta jHv3.

Alternativas.

* En cualquier momento el usuario puede cancelar el proceso de migración.

3b. El sistema detecta que el nombre de usuario o la contraseña proporcionados por el usuario no existen en el sistema e informa al usuario de dicha excepción y vuelve al paso 2.

4.3.3 Registrar usuario-Cookie

Camino directo.

El usuario quiere poder aportar contenidos sin tener que registrarse en el sistema.

1. El sistema pide al usuario que introduzca una dirección de correo electrónico y un nombre de usuario.
2. El usuario introduce su dirección de correo electrónico y un nombre de usuario válidos.
3. El sistema verifica que no existe otro usuario con esa misma dirección de correo electrónico.
4. El sistema almacena el nuevo usuario-Cookie.
5. El sistema hace la asignación de rol básica, genera la URL de Gravatar a partir de la dirección de correo electrónico proporcionada por el usuario y almacena el nuevo usuario
6. El sistema genera una cookie con un identificador único que se almacena en la máquina del usuario-Cookie, inicia la sesión automáticamente y continúa con el proceso de aportación de contenido.

Alternativas.

* En cualquier momento el usuario puede cancelar el proceso de registro en los pasos 1 y 2.

3b. El sistema detecta que el usuario ya existe ya existe en el sistema e inicia sesión automáticamente.

4.3.4 Login de usuario con OpenId

Camino directo.

El usuario registrado con OpenId quiere iniciar sesión en el sistema.

1. El sistema le muestra al usuario los proveedores más comunes de OpenId que existen.
2. El usuario escoge el proveedor que quiere utilizar para verificarse (Ej.: Google, OpenID URL o Abrir-se una cuenta en OpenId)
3. El sistema redirecciona la petición al sistema remoto de OpenId escogido por el usuario y este le pide que inicie sesión.
4. El usuario inicia sesión en el sistema remoto y le da permiso para proporcionar información al sistema jH.
5. El sistema de OpenId devuelve una URL de verificación del usuario junto con algunos datos del usuario que el sistema jH le ha requerido.

6. El sistema verifica que existe un usuario registrado con ese misma URL de OpenId e inicia la sesión del usuario automáticamente y este es redireccionado a la pantalla de inicio.

Alternativas.

* Del paso 1 al 4 el usuario puede cancelar el proceso de login.

6b. El sistema detecta que el usuario no existe en el sistema y lo redirecciona al proceso de registro.

4.3.5 Login usuario con Cookie

Camino directo.

El usuario quiere iniciar sesión en el sistema.

1. El usuario tiene todavía en su sistema la cookie de autenticación generada por jH, con lo que tan solo cargar la página del portal éste se loguea de forma automática.

Alternativas.

1b. No existe cookie de autenticación y el proceso no se completa.

1c. La cookie de autenticación ha expirado y el proceso no se completa.

4.3.6 Visualizar perfil de usuario

Camino directo.

Un usuario quiere visualizar su perfil o el de otro usuario y hace la petición al sistema.

1. El sistema recupera el perfil almacenado del usuario y lo muestra por pantalla.

Alternativas.

1b. El Sistema verifica que el perfil de usuario pedido no existe e informa al Usuario.

4.3.7 Editar perfil de usuario

Camino directo.

El Usuario quiere editar su perfil e inicia el proceso.

1. El sistema muestra un formulario con los datos del Usuario para que este pueda añadir datos o modificarlos.
2. El Usuario edita los datos del formulario que quiere cambiar y acepta los cambios.
3. El Sistema verifica los datos introducidos en el formulario, genera de nuevo el código de Gravatar (para la imagen del perfil) y almacena el perfil.
4. El Sistema redirecciona al Usuario a la visualización de su perfil.

Alternativas.

* En los pasos 1 y 2 el usuario puede cancelar el proceso de editar su perfil.

3b. El Sistema redirecciona al Usuario al paso 1 y le muestra los errores encontrados.

5 Diseño

5.1 Diseño de la base de datos

Dentro del submodelo Usuario de la Base de Datos se encuentra toda la información referente a los usuarios, tanto información personal de éstos como de los diferentes roles de los que este dispone.

Toda la información referente a la cuenta de usuario se almacenará dentro de la tabla account:

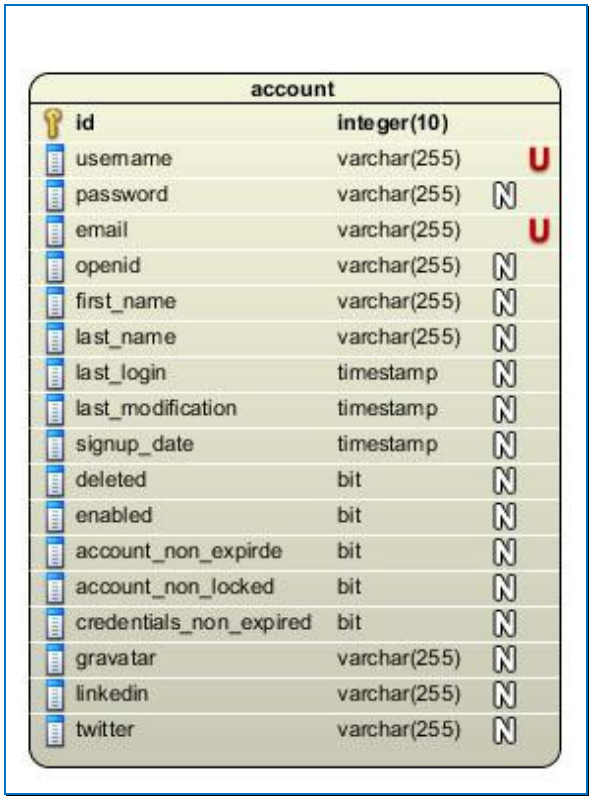


Figura V: Tabla account (usuario)

Cada usuario tendrá asignado unos determinados roles que le permitirán tener acceso a los recursos de jH. La información referente a estos roles se almacenará dentro de la tabla role:

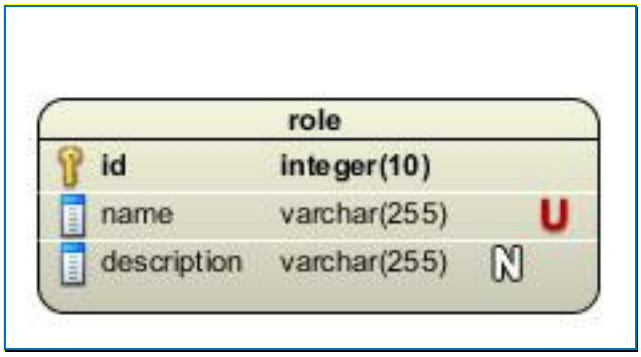


Figura VI: Tabla role (rol de usuario)

El diseño del submodelo Usuario de la Base de Datos es el siguiente:

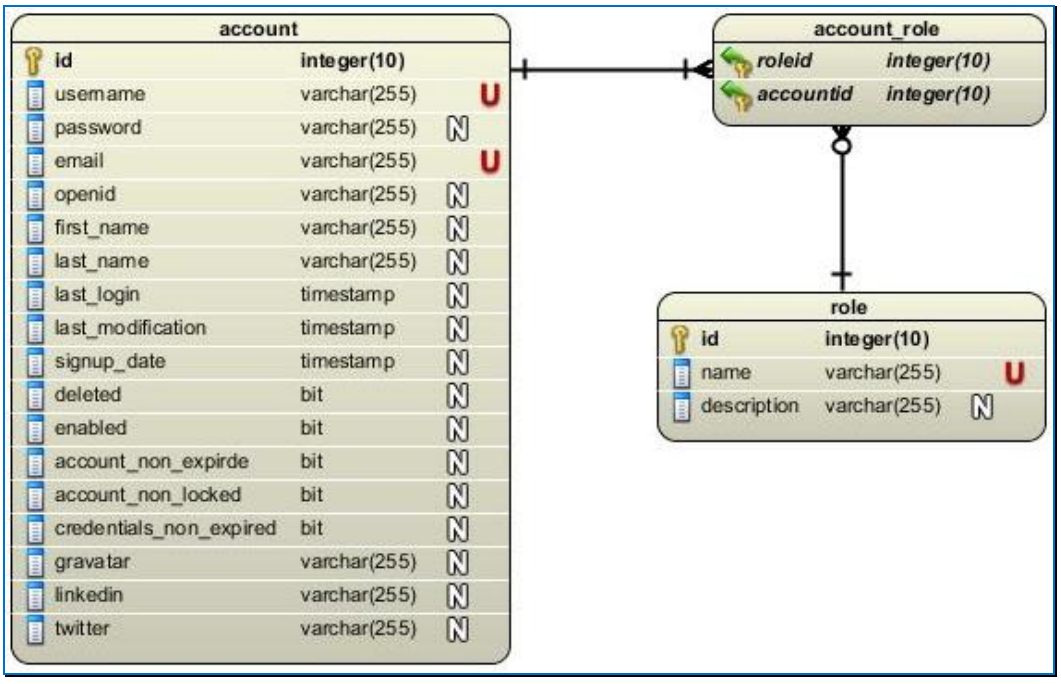


Figura VII: Diseño de la base de datos de Usuario

5.2 Capas de abstracción

El proyecto está dividido en capas bien definidas que caracterizan a las aplicaciones Spring MVC. Como capa entendemos a una parte de la arquitectura con unas responsabilidades exclusivas e independientes del resto.

Las capas son abstracciones dentro de una aplicación que cubren una necesidad arquitectónica y las interfaces definen como pueden interactuar estas entre sí. Cada capa es accesible solo desde la capa inmediatamente superior, a excepción del dominio que se extiende verticalmente y es accesible desde todas las otras.

Figura VIII: Capas de abstracción de la aplicación

5.2.1 Capa de interface de usuario

Esta capa se encarga de la generación de la interface o vista de usuario y también de la transformación de los resultados obtenidos, a partir de la petición de un usuario, a código HTML de forma que este pueda visualizarlos.

A esta capa podemos acoplar cualquier tecnología compatible de representación de datos. Spring, por defecto, permite utilizar Freemarker, JSP, Velocity, XSTL, JasperReports, Excel y PDF.

En esta capa también reside el *ViewResolver*, cuya finalidad es traducir las referencias físicas en referencias lógicas. Por ejemplo, la referencia física de la vista de login es */WEB-INF/templates/login.ftl*, la podemos referenciar simplemente con *login*. De esta forma desacoplamos la vista, plantillas Freemarker, del código que les hace referencia.

Aunque no es necesario, esta capa tiene dependencias con la capa de dominio para hacer más sencillo el trabajo a la hora de representar los datos de las páginas web.

5.2.2 Capa Web

En esta capa residen los controladores que se encargan de conducir al usuario por el flujo de navegación correcto. A su vez es responsable de derivar las peticiones de los usuarios a la capa de servicios y encaminar los resultados a la vista de la capa de interface adecuada para su representación.

Esta capa solo tendrá dependencias con las capas de servicio y dominio.

5.2.3 Capa de servicio

Todas las reglas definidas por los casos de uso se definen aquí, donde residirá la lógica de negocio. Esta capa recibe las peticiones de la capa web y las procesa de forma transaccional, es decir, que todas las operaciones se ejecutan en una sola transacción.

La capa de servicio tiene dependencias con la capa de dominio para trabajar con los objetos del modelo de negocio y con la capa de persistencia que es la encargada de gestionar el almacenamiento de los objetos del dominio.

5.2.4 Capa de modelo del dominio

Esta capa es, sin duda alguna, la capa más importante de todas ya que contiene modelo de negocio. Este modelo se compone de un conjunto de entidades que definen el negocio llamadas POJO (Plain Old Java Object) o beans anémicos.

Estos POJOs no deben contener funcionalidad alguna del sistema, tan solo los atributos y los métodos necesarios para la encapsulación de estos (getters y setters).

A esta capa se le acoplan todas las demás pero no se acopla con ninguna otra.

5.2.5 Capa de persistencia

Esta capa es la encargada de ser la intermediaria entre el sistema y los sistemas gestores de almacenamiento como bases de datos, ficheros, sistemas externos, etc. Esto permite desacoplar la gestión de persistencia del resto de la aplicación.

Ninguna capa situada por encima de esta no tiene porqué saber como se almacenan los datos. Esto hace que no se vean afectadas por los cambios realizados en alguno de los sistemas de persistencia.

La capa de persistencia ofrece operaciones de tipo CRUD (Crear, Leer, Actualizar y Eliminar) y su finalidad es extraer y organizar los datos de los objetos del dominio y después almacenarlos mediante instrucciones que estos puedan interpretar y ejecutar.

6 Implementación

6.1 Tecnologías

El punto fuerte del proyecto son las tecnologías que se han utilizado para llevar a cabo el proyecto. Al tratarse de un proyecto de código abierto, todas las tecnologías utilizadas también lo son.

De entre la gran variedad de tecnologías existentes, se ha intentado escoger aquellas que se utilizan frecuentemente y que tienen una gran demanda en el mercado.

6.1.1 Maven

Maven[6] es una herramienta de comprensión y gestión de proyectos software creada por Jason van Zyl, de la empresa Sonatype, en 2002.

Esta tecnología se basa en el concepto anglosajón Project Object Model (POM) que se utiliza para describir en formato XML el proyecto a construir, sus dependencias de otros módulos y componentes externos, y el orden de construcción de los elementos.

Este archivo de configuración de proyecto se llama *pom.xml*.

6.1.2 Spring Framework 3

Spring Framework[7] (SF), también conocido como Spring, es un proyecto de código abierto de desarrollo de aplicaciones para la plataforma Java.

Por su diseño, Spring ofrece mucha libertad a los desarrolladores en Java y soluciones muy bien documentadas y fáciles de usar para las prácticas comunes en la industria. Además, Spring es modular, lo que permite utilizar sólo aquellas partes que los desarrolladores necesitan, sin tener que recurrir al resto.

Se puede utilizar el contenedor IoC (Inversión de control). Este nos permite delegar el control del flujo de ejecución de la aplicación al Framework, indicándole que acciones tiene que llevar a cabo según las respuestas obtenidas de algunos sucesos o peticiones de datos.

6.1.2.1 Spring MVC

El framework Spring ofrece un marco de trabajo Modelo-Vista-Controlador (MVC) con todas las funciones, y que permite integrar de forma transparente AoP (siglas en inglés de Programación Orientada a Aspectos) a un proyecto software.

Spring está diseñado para ser no intrusivo, lo que significa que el código de la lógica de negocio en general no tiene dependencias hacia el propio framework. En alguna capa, pueden existir algunas dependencias de la tecnología de acceso de datos y las bibliotecas de Spring. Sin embargo, debe ser fácil de aislar estas dependencias desde el resto del código base.

Una de las novedades que se presentan en la nueva versión de Spring MVC, es la posibilidad de desarrollar aplicaciones de tipo REST[8] (abreviatura de Representational State Transfer). Esta tecnología ha surgido como alternativa a otras arquitecturas distribuidas como SOAP, WSDL y otras basadas en WebService.

Una ventaja de REST es que utiliza HTTP como protocolo de comunicación. Los recursos son subministrados a través de URI (siglas en inglés de identificador uniforme

de recurso). Al usar este protocolo pueden utilizarse las operaciones GET, PUT, POST, DELETE.

Permite dar los resultados no solo en HTML, sino en XML JSON (una alternativa a XML en AJAX), etc..El protocolo HTTP tiene la ventaja, respecto a otro tipo de arquitecturas, de estar soportado por todas las plataformas y lenguajes.

Pero una desventaja de este, es que, no todos los navegadores web soportan todas las operaciones de HTTP. Por ejemplo, no permite utilizar PUT para ejecutar opraciones de actualización, ni DELETE para las de eliminación.

Por otro lado, se ha elegido este tipo de arquitectura porque cumple perfectamente las necesidades del sistema jH. Ver el apartado "*6.2.2. Indicando a Spring los recursos disponibles*".

Desde la versión 2.5 de Spring MVC, están disponibles las anotaciones[9] para declarar controladores MVC. Para indicar a Spring la existencia de un controlador se anota con el estereotipo *@Controller*. Esta se inserta justo encima de la declaración de la clase en cuestión. De esta forma, queda registrado en el sistema como tal e indica al framework que tiene que escanear el controlador en busca de mapeos de petición (*request mappings* en inglés).

Esos request mappings se expresan con la anotación @RequestMapping. Pueden ponerse a nivel de clase o de método para vincularlos a un path de recurso HTTP en el *DispatcherSerlet*. Por ejemplo, si añadimos la anotación @RequestMaping("/user/profile") en un método, este será ejecutado cuando se especifique la URL "*http://www.javahispano.org/user/profile*".

En la versión 3 de Spring MVC se ha añadido las *URL Templates*. Estas permiten capturar variables incluidas en el path, evitándose así el paso de parámetros. Estas se pueden capturar con la anotación *@PathVariable* como si fuera un parámetro del método.

6.1.2.2 Spring Security

Spring Security[10] es uno de los proyectos SpringSource más maduros y extensamente utilizado. Fue fundado en 2003 y desde entonces ha sido mantenido activamente. Hoy en día es utilizado en numerosos entornos incluyendo agencias gubernamentales, aplicaciones militares y entidades financieras.

Mediante un archivo de configuración XML se pueden aplicar directivas de seguridad a la aplicación, tales como control de acceso a usuarios, gestión de sesiones, restricciones de acceso a recursos mediante roles asignados al usuario, etc.

Cada vez que se recibe una petición entrante, ya sea desde un navegador web, un web service o una aplicación AJAX, se ejecuta lo que se llama cadena de filtros. Cada uno de ellos tiene su responsabilidad.

Los filtros de seguridad se ejecutan, unos u otros, dependiendo de que servicios se requieran. El orden en el que se ejecutan es importante ya que existen dependencias entre ellos como si de capas se tratara.

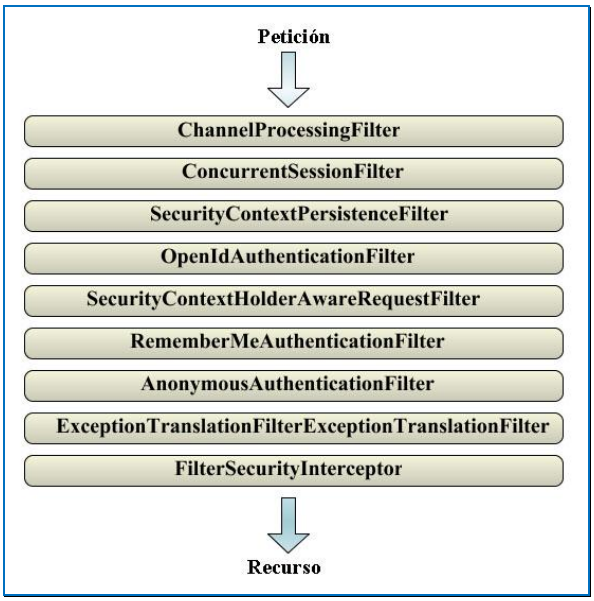


Figura IX: Cadena de filtros de Spring Security

- **ChannelProcessingFilter:** Se encarga de encaminar la petición por el canal correcto.
- **ConcurrentSessionFilter:** Actualiza el registro de sesiones y verifica que no han expirado.
- **SecurityContextPersistenceFilter:** Filtro para configurar el *SecurityContext*.
- **OpenIdAuthenticationFilter:** Filtro encargado de la autenticación del usuario con un proveedor OpenId.
- **SecurityContextHolderAwareRequestFilter:** Un filtro que añade un Nuevo *request wrapper* al Servlet Request.
- **RememberMeAuthenticationFilter:** Si no existe ninguna autenticación en el *SecurityContext* y el cliente tiene una cookie *remember-me*, se inicia sesión automáticamente**.**
- **AnonymousAuthenticationFilter:** *Para el proceso de usuarios anónimos.*
- **ExceptionTranslationFilter:** Este filtro captura todas las excepciones que lanza Spring y genera una respuesta HTTP de error.
- **FilterSecurityInterceptor:** Este filtro protege las URIs y lanza excepciones de acceso denegado.

6.1.3 Persistencia con iBatis 2

Para la persistencia de los datos se ha escogido el framework iBatis 2[11], un framework de código abierto, creado por Clinton Begin, que se ocupa de la capa de persistencia. Se encuentra disponible en los lenguajes de programación Java y .NET.

Este framework se encarga de asociar los objetos del dominio con la base de datos mediante archivos de configuración XML. De esta forma se simplifica mucho el trabajo de almacenamiento.

La capa de **Abstracción** implementa el patrón DAO (Data Acces Object) y hace de fachada entre la aplicación y el resto de la persistencia. Esta parte se encuentra en el archivo de configuración *ibatis-config.xml.*

En la capa de **Framework de Persistencia** residen las clases Java que implementan la interface Dao y los archivos que contienen las sentencias SQL. Cada objeto del dominio

tendrá su correspondiente archivo de tipo SqlMap.xml, por ejemplo, el objeto *Account* con *Account.xml*.

La capa de **Driver** se encarga de la comunicación con la Base de Datos utilizando un Driver específico para la misma.

En un principio se quería usar la versión iBatis 3 pero debido a problemas en las operaciones transaccionales de Spring se optó por usar iBatis 2 de este framework hasta que estos problemas estén solventados.

6.1.4 Sistema gestor de base de datos PostgreSql

Como sistema gestor de base de datos se ha escogido PostgreSQL[12] en su versión 8.4.3. Ha sido desarrollado por el *PostgreSQL Global Development Group*.

Es uno de los mejores sistemas gestores de base de datos de código libre y dispone de infinidad de características interesantes. Para la administración de PostgreSQL se utiliza pgAdmin, que es la aplicación, de código libre también, más conocida para realizar esta tarea.

6.1.5 Interface generada con Freemarker

El encargado de generar el código HTML del interface es, el motor de código libre basado en plantillas, Freemarker[13]. Este está diseñado para ser práctico en la generación de código HTML, particularmente en aplicaciones que siguen el patrón MVC.

La idea de usar Freemarker para generar paginas web dinámicas es separar las tareas de diseño HTML y las de los programadores. Los diseñadores pueden cambiar la apariencia de una página sin que el programador tenga que modificar o recompilar código, debido a que la lógica de la aplicación (código Java) y el diseño de la página están separados.

Aunque FreeMaker tiene algunas características de programación, no es un lenguaje de programación en toda regla como PHP. El código Java prepara los datos para mostrar y

FreeMaker sólo genera páginas textuales que muestran los datos preparados utilizando plantillas.

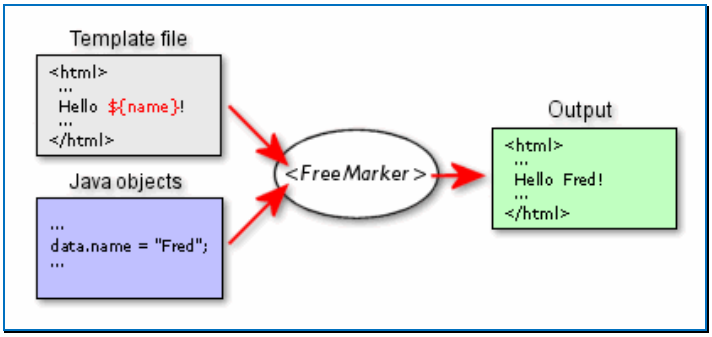


Figura X: Funcionamiento de Freemarker

6.1.6 Entorno de desarrollo SpringSource Tool Suite

Las herramientas de desarrollo han evolucionado mucho en los últimos tiempos y comúnmente son conocidas como IDE (las siglas en ingles de Entorno de Desarrollo Integrado). Algunas de estas IDEs como NetBeans de Sun MicroSystems, Eclipse o SpringSource Tool Suite[14] (STS) son proyectos de código libre.

Para el desarrollo de este proyecto se ha escogido STS, un IDE desarrollado por SpringSource y que utiliza un Eclipse como base. Provee el mejor entorno para construir aplicaciones utilizando Spring.

6.1.7 Servidor de aplicaciones Tomcat

El servidor de aplicaciones web escogido para trabajar de forma local es Apache Tomcat[15] 6.0. Este es un contenedor de servlets que implementa las tecnologías Java Servlet y JavaServer Pages.

Existen otros servidores como Glasfish de SunMicroSystems que tiene un rendimiento parecido al de Apache Tomcat.

6.1.8 Control de versiones Git

Git[16] es un sistema de control de versiones distribuidas de código libre diseñado para soportar cualquier tipo de proyecto, tanto pequeño como de gran extensión, de forma eficiente y rápida.

Otros sistemas parecidos son Subversion, CVS o Visual SourceSafe entre otros. Pero una de las características destacables de este sistema es que permite hacer branches y commits de forma local, es decir, se puede trabajar de forma local e ir haciendo varias versiones sin necesidad de conectar-se al repositorio central.

El repositorio Git que se utiliza para la control de versiones de este proyecto es GitHub[17]. Este es un servicio de hosting online para proyectos que utilizan este sistema muy completo y utilizado por proyectos tanto privados como de código libre.

Algunos de los proyectos hospedados en GitHub son jUnit, jQuery, PHP, Perl, Prototype, Ruby on Rails y un largo etc. Eso y que se utilizan encriptaciones de seguridad SSH para poder trabajar con el repositorio, dio a jH la confianza suficiente para hospedar allí el nuevo proyecto.

6.1.9 Otras tecnologías

6.1.9.1 OpenId (proveedor de identificación de usuario)

OpenID[18] es un sistema de autenticación digital descentralizado, con el que un usuario puede identificarse en una página web a través de una URL y puede ser verificado por cualquier servidor que soporte el protocolo.

Los usuarios no tienen que crearse una nueva cuenta de usuario para obtener acceso. En su lugar, sólo necesitan disponer de un identificador creado en un servidor que verifique OpenID, llamado proveedor de identidad o IdP, por ejemplo Google, Yahoo o AOL.

La arquitectura OpenID no especifica el mecanismo de autenticación. Por lo tanto, la seguridad de una conexión OpenID depende de la confianza que tenga el cliente OpenID en el proveedor de identidad. Si no existe confianza en el proveedor, la autenticación no será adecuada para servicios bancarios o transacciones de comercio electrónico, sin embargo el proveedor de identidad puede usar autenticación fuerte pudiendo ser usada para dichos fines.

OpenID está ganando fuerza debido al anuncio de algunos sitios grandes, como Wikipedia y Technorati con la adopción de este sistema.

Spring Security permite la identificación de usuarios mediante proveedores de OpenId. En cuanto al tema de usabilidad jH se ha querido ir un paso más allá. La opción de login y la de registro serán fusionadas, así existirá tan solo un punto de entrada para la verificación de usuarios OpenId.

Cuando un usuario no registrado se verifica con OpenId el sistema redirecciona automáticamente al formulario de registro rellenándolo con los datos extraídos del proveedor de autenticación OpenId.

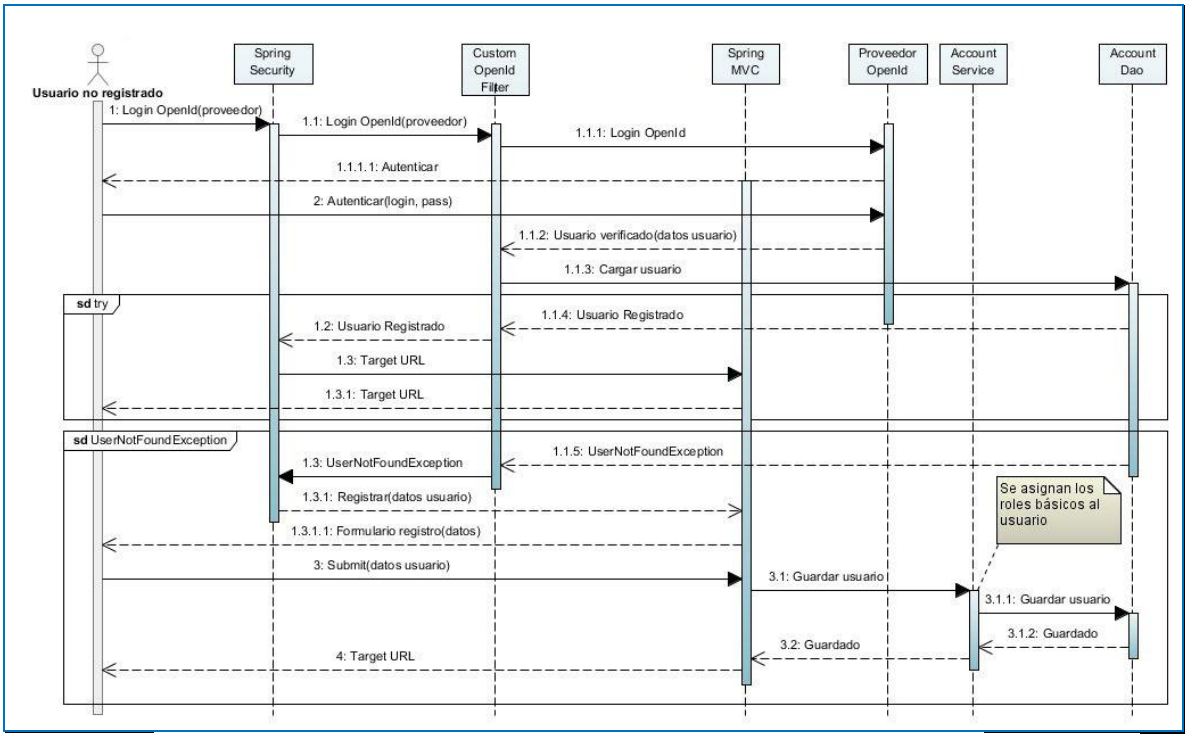


Figura XI: Diagrama de secuencia Login de usuario con OpenID

El problema aparece cuando esos atributos extraídos del proveedor de OpenId no son accesibles en el momento en que Spring Security detecta que ese usuario no esta registrado en el sistema.

Para solucionar esto ha sido necesario redefinir el filtro de seguridad *OPENID_FILTER* de Spring Security. Estudiando a fondo el código fuente del framework de seguridad ha sido posible entender bien el funcionamiento interno de este y, de esta forma se ha conseguido hacer un *bypass* en el framework de seguridad.

Una vez redefinido el filtro de seguridad es necesario indicar en el archivo de configuración de seguridad *security-config.xml* que sustituya el nuevo filtro por el que se ejecuta de forma nativa. Para ver como actúa este filtro mediante un diagrama de secuencia ver ANEXO X - Diagramas UML.

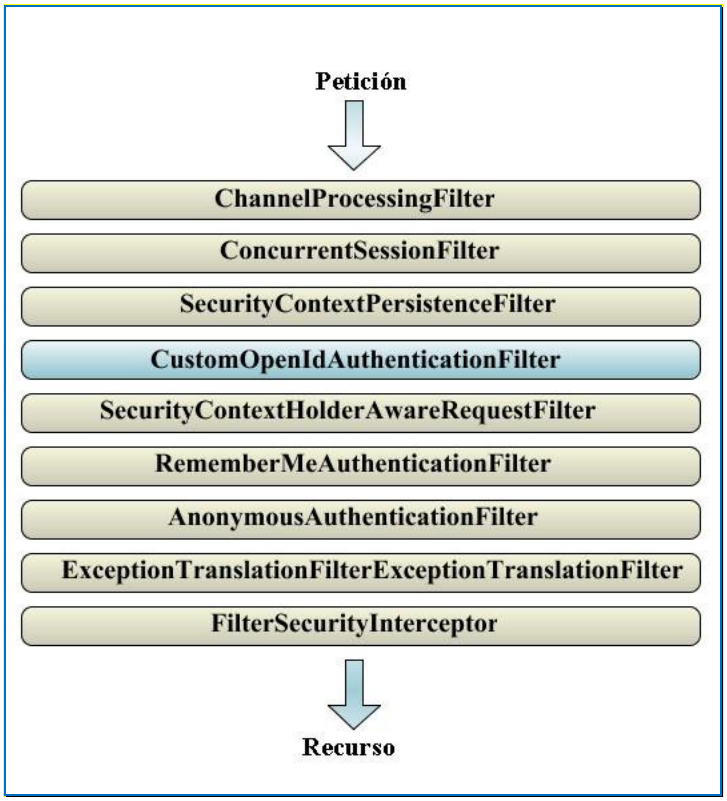


Figura XII: Bypass del filtro de seguridad OpenIDAuthenticationFilter

6.1.9.2 Gravatar

Para representar el avatar del usuario se usa Gravatar[19] (una abreviación para Globally Recognized Avatar en inglés). El avatar de un usuario es la imagen de su perfil.

Gravatar ofrece este servicio para que los usuarios puedan tener su avatar en un único lugar. Los usuarios tienen que aportar su correo electrónico para ser identificados. Este correo se pasa a minúsculas (lowercase) y se transforma en código hash del tipo md5. Ese será el identificador que usara el sistema jH para obtener la imagen avatar del sistema Gravatar.

Si un usuario no dispusiera de una cuenta en Gravatar el sistema muestra un avatar de usuario predefinida de jH.

6.2 Desarrollo de jHv3

Aquí se muestra una descripción más detallada de como se han aplicado algunas de las tecnologías descritas anteriormente al proyecto jHv3.

6.2.1 Patrones destacables

- **Modelo-Vista-Controlador**

El patrón Modelo-Vista-Controlador es una guía para el diseño de arquitecturas de aplicaciones que ofrezcan una fuerte interactividad con usuarios.

Este patrón organiza la aplicación en tres modelos separados, el primero es un modelo que representa los datos de la aplicación y sus reglas de negocio, el segundo es un conjunto de vistas que representa los formularios de entrada y salida de información y el tercero es un conjunto de controladores que procesa las peticiones de los usuarios y controla el flujo de ejecución del sistema.

Spring Framwork implementa este patrón. Esta arquitectura consiste, a grandes rasgos, en la utilización de servlets para procesar las peticiones (controladores), páginas Freemarker para mostrar la interfaz de usuario (vistas) e implementando la parte del modelo mediante unas clases Java llamadas servicios.

- **Bajo acoplamiento**

Los componentes o clases Java deben ser lo mas independientes como sea posible de las otras clases Java. Esto incrementa la posibilidad de reutilizar el código y hacer tests independientemente del resto de la aplicación.

Para conseguir esto, Spring permite la inyección de dependencias (*Dependency Injection*[20]) a otras clases en lugar de que estas creen o busquen los objetos. Por

ejemplo, si tenemos una clase A que tiene dependencia a la clase B, entonces podemos utilizar la inyección de dependencia vía:

- **Contruction Injection:** La clase B es inyectada mediante el constructor de la clase A. Ej: *public AccountServiceImpl(AccountDao accountDao)*.
- **Setter Injection:** La clase B es inyectada utilizando un método *set*. Ej: *setAccountDao(AccountDao accountDao)*.

Su filosofía dice que una clase no debe configurarse ella misma, sino que se debe hacer desde el exterior.

6.2.2 Indicando a Spring los recursos disponibles

Indicar al framework Spring que es un controlador, un servicio o una clase de persistencia es muy sencillo. Para eso, vamos a utilizar las anotaciones que nos ofrece este.

Si queremos indicar que una clase quede registrada como un controlador que recibe las peticiones del usuario, se utiliza la anotación @Controller a nivel de clase.

```
@Controller
public class JavaHispanoController {
```

Figura XIII: Ejemplo de @Controller

Si una clase tiene que ser interpretada como un Servicio se usa la anotación @Service.

```
@Service
public class AccountServiceImpl implements AccountService {
```

Figura XIV: Ejemplo de @Service

Y por ultimo, con la anotación @Repository indicamos al framework que se trata de una clase de Persisténcia.

```
@Repository("accountDao")
public class SqlMapAccountDao extends SqlMapGenericDao<Long, Account> implements
                                AccountDao, UserDetailsService {
```

Figura XV: Ejemplo de @Repository

6.2.3 Peticiones mediante REST.

Para ver como es el funcionamiento de la arquitectura REST veremos como ejemplo el proceso que sigue la edición del perfil de usuario.

Cuando un usuario registrado está visualizando su propio perfil, se le da la opción de poder editarlo.

```
<a href="/portal/user/profile/edit">Editar</a>
```

Figura XVI: HTML visualización de perfil

El usuario inicia el proceso de edición, con la URL "*/portal/user/profile/edit*" Spring busca entre los controladores que tiene registrados y redirecciona la petición hacia el destino correcto.

```
@Controller
@RequestMapping("/user/profile/edit")
public class EditProfileController {
```

Figura XVII: RequestMapping del Controlador EditProfileController

Una vez Spring ha reconocido el controlador pertinente, se ejecuta los métodos anotados con *@ModelAtribute*, encargados de cargar los objetos al modelo para que sean renderizados por la vista.

```
@ModelAttribute("account")
public Account exposeAccount() {
    return accountService.getCurrentUser();
}
```

Figura XVIII: Método del controlador anotado con @ModelAttribute

Y el método anotado con *GET*. Este después redirecciona a la vista "*user/edit*" (formulario de edición de perfil).

```
@RequestMapping(method=RequestMethod.GET)
public String showAccountForm(Model model, HttpServletRequest request) {
    Account signupAccount = (Account) request.getAttribute("signupaccount");
    if (signupAccount != null) {
        model.addAttribute("account", signupAccount);
        return "/signup";
    }
    return "user/edit";
}
```

Figura XIX: Método del controlador anotado con GET

Una vez generada la vista podemos observar dos cosas, el atributo *method* se ha puesto en para que ejecute una operación post. Y la otra característica es que no se ha definido el *action*. Eso indica que la petición se ejecutará sobre el mismo recurso "*/portal/user/profile/edit*".

```
<form method="post" action="">
    <div>
        <label for="username">Nombre de usuario</label>
        <input type="text" id="username" name="username" value="pepito" maxlength="200"    />
    </div>

    <div>
        <label for="email">Email</label>
        <input type="text" id="email" name="email" value="xlluch@javahispano.org" maxlength="200"    />
    </div>

    <div>
        <button type="submit">Guardar</button>
    </div>
</form>
```

Figura XX: HTML de edición de perfil

Una vez se ha editado el perfil y se hace la petición, Spring redirecciona de nuevo al mismo controlador. Pero esta vez se ejecuta el método anotado con *POST*. Si hay errores, este vuelve a la misma vista pero mostrando los errores al usuario.

Una vez almacenados los cambios del usuario, hace una redirección con "*redirect:/user/profile*". Esto, en lugar de cargar una vista, redirecciona al controlador que visualiza el perfil de usuario.

```
@RequestMapping(method = RequestMethod.POST)
public String processContentSubmit(@ModelAttribute Account account,
        BindingResult result) {
    if (result.hasErrors()) {
        if (account.isNew()) {
            return "/signup";
        }
        return "user/edit";
    } else {
        accountService.saveAccount(account);
        return "redirect:/user/profile";
    }
}
```

Figura XXI: Método del controlador anotado con POST

6.2.4 Seguridad de acceso a recursos

En el archivo *security-config.xml* es donde se encuentra toda la configuración de seguridad de la aplicación. Spring Security se encarga de ejecutar las directivas y beans que se describen en este. Toda la configuración que afecta a las peticiones de los usuarios a través del explorador web va incluida dentro del tag **<*html*>**.

Mediante los *url-interceptor*s, se especifica qué roles o *Authorities*, son necesarios para poder acceder a los recursos del sistema. No se puede permitir que un usuario sin el rol *ROLE_ADMIN* acceda a recursos reservados a los administradores, o que un usuario no registrado pueda aportar un contenido sino tiene el rol *ROLE_USER*.

```
<!-- role configuration -->
<intercept-url pattern="/content/new/**"       access="hasAnyRole('ROLE_USER','ROLE_ADMIN')" />
<intercept-url pattern="/admin/**"             access="hasRole('ROLE_ADMIN')" />
<intercept-url pattern="/user/profile/edit/**" access="hasAnyRole('ROLE_USER','ROLE_ADMIN')" />
```

Figura XXII: Configuración de seguridad por roles

Ahora si un usuario quiere acceder, por ejemplo, al recurso "*/content/new*", este ha de tener asignado el rol *ROLE_ADMIN* o el rol *ROLE_USER* para poder hacerlo. De lo contrario el sistema muestra al usuario un mensaje que le indica que no tiene privilegios suficientes.

6.2.5 Autenticación de usuario con OpenId

Cuando un usuario que no ha iniciado sesión en el sistema intenta acceder a algún recurso que requiere algún rol específico, el mecanismo de seguridad lo redirecciona al formulario de login para que se identifique.

Spring Security configura automáticamente la cadena de filtros de seguridad sin necesidad de especificárselo. Pero para poder implementar la funcionalidad de login con OpenId es necesario tener más control sobre esta cadena. Para eso se ha desarrollado una redefinición del filtro de seguridad *OpenIdAuthenticationProcessingFilter*.

```
<beans:bean id="openIDAuthenticationProcessingFilter"
            class="org.javahispano.portal.security.CustomOpenIDAuthenticationProcessingFilter">
    <beans:property name="filterProcessesUrl"           value="/jh_openid_security_check"/>
    <beans:property name="authenticationManager"        ref="authenticationManager"/>
    <beans:property name="authenticationFailureHandler" ref="openIDFilterFailure"/>
    <beans:property name="consumer"                     ref="openID4JavaConsumer"/>
</beans:bean>

<beans:bean id="openIDAuthenticationProvider"
            class="org.javahispano.portal.security.CustomOpenIDAuthenticationProvider">
    <beans:constructor-arg ref="accountDao"/>
</beans:bean>

<beans:bean id="openIDFilterFailure"
            class="org.springframework.security.web.authentication.SimpleUrlAuthenticationFailureHandler">
    <beans:property name="defaultFailureUrl" value="/app/signup"/>
    <beans:property name="useForward" value="true"/>
</beans:bean>

<beans:bean id="openID4JavaConsumer" class="org.javahispano.portal.security.CustomOpenID4JavaConsumer"/>

<authentication-manager alias="authenticationManager">
    <authentication-provider ref="openIDAuthenticationProvider" />
</authentication-manager>
```

Figura XXIII: Declaración del filtro de seguridad OpenIdAuthenticationFilter

Una vez declarados los beans del nuevo filtro, se indica dentro del tag ***<html>*** que tiene que substituirlo por el filtro *OPENID_FILTER.*

```
<custom-filter ref="openIDAuthenticationProcessingFilter" position="OPENID_FILTER"/>
```

Figura XXIV Declaración de CustomOpenIdAuthenticationFilter

7. Conclusiones

Se han cumplido satisfactoriamente los objetivos marcados para la fase inicial del proyecto. Con la creación del nuevo sistema javaHispano, la comunidad obtiene numerosos beneficios.

- Mediante el uso de frameworks muy conocidos y extendidos, se aumenta la posibilidad de que desarrolladores que tengan conocimientos sobre alguno de estos, puedan colaborar en la creación de nuevas funcionalidades,

- Se pone a disposición de los usuarios, el código fuente de un proyecto web que utiliza tecnologías de última generación. Con una metodología de programación limpia y homogénea.

- La reingeniería ha permitido aplicar mecanismos que facilitan la interacción y la accesibilidad del usuario con el sistema.

Una característica a destacar de todo lo que se ha utilizado en el proyecto (tanto tecnologías como herramientas) es que todo el software utilizado es de código libre. De esta forma queda demostrado que crear un portal de código libre es posible y no queda atrás comparado con otras soluciones propietarias.

Una de las ampliaciones previstas es la de implantar un sistema de medallas con el fin de motivar la interacción de los usuarios con el sistema.

Este sistema de medallas consiste en gratificar simbólicamente al usuario por realizar ciertas acciones que enriquezcan el contenido de la comunidad (aportaciones de contenido, comentar contenidos de otros usuarios, etc.).

Aunque el diseño gráfico del interface de usuario no entra en los objetivos de este trabajo, se ha utilizado un diseño provisional para que la presentación de este proyecto sea más agradable.

ANEXO I – Glosario de términos

- **API:** (Application Program Interface). Conjunto de convenciones internacionales que definen cómo debe invocarse una determinada función de un programa desde una aplicación. Cuando se intenta estandarizar una plataforma, se estipulan unos APIs comunes a los que deben ajustarse todos los desarrolladores de aplicaciones.

- **Cáñamo:** Framework para crear portales y comunidades. utilizado para el desarrollo de javaHispano.

- **Cookie:** (galleta) Cuando se visita una página Web, es posible recibir una Cookie. Este es el nombre que se da a un pequeño archivo de texto, que queda almacenado en el disco duro del ordenador. Este archivo sirve para identificar al usuario cuando se conecta de nuevo a dicha página Web.

- **CSS:** (Cascading Style Sheets) Hoja de Estilo en Cascada. Dentro del diseño de páginas de Internet se presenta esta como la vanguardia en cuanto a definición de estilos dentro de las plantillas de diseño. A través de instrucciones en código HTML se definen los estándares del conjunto de páginas que conforman el proyecto. La meta es uniformizar nuestro diseño.

- **CMS:** Del inglés Content Management System, es el sistema de gestión de contenidos de una página web.

- **Feed:** Es un medio de redifusión de contenido web principalmente en formato RSS o Atom. Se utiliza para suministrar información actualizada frecuentemente a sus suscriptores.

- **Framework:** Es un conjunto de herramientas y APIs que liberan al desarrollador de la implementación de gran parte del código de arquitectura.

- **HTTP:** (Hiper Text Transfer Protocol). Protocola de transferencia de HiperTexto. Es el protocolo de Internet que permite que los exploradores del WWW recuperen

información de los servidores. Es un protocolo de aplicación con la sencillez y velocidad necesaria para sistemas de información distribuidos, colaborativos y de diferentes medios.

- **HTML:** (HyperText Markup Language). Lenguaje de marcado de Hipertexto. Es el lenguaje estándar para describir el contenido y la apariencia de las páginas en el WWW.

- **JavaEE:** (Java 2 Enterprise Edition) define un estándar para el desarrollo de aplicaciones empresariales multicapa diseñado por Sun Microsystems. J2EE simplifica las aplicaciones empresariales basándolas en componentes modulares y estandarizados, proveyendo un completo conjunto de servicios a estos componentes, y manejando muchos de las funciones de la aplicación de forma automática, sin necesidad de una programación compleja.

- **JavaME:** (Java 2 Platform Micro Edition) Edición Micro. Una última versión reducida de Java 2 orientada a aplicaciones para dispositivos electrónicos, como móviles, PDAs, etc.

- **JavaSE:** (Java 2 Standard Edition) Edición Estándar. La versión estándar es la más común y cuenta con todo lo necesario para desarrollos de software y acceso a aplicaciones Java.

- **Java:** Lenguaje de programación, desarrollado por Sun Microsystems.

- **Login:** Entrada de identificación de un usuario, conexión.

- **Maven:** Es una herramienta de comprensión y gestión de proyectos software.

- **Open Source:** Software desarrollado bajo la línea del código Abierto, distribuido de manera gratuita, y sin garantías totales de su funcionamiento.

- **Podcast:** Archivo multimedia (normalmente audio o vídeo) distribuido mediante un sistema de sindicación que permita suscribirse y usar un programa que lo descarga para que el usuario lo escuche en el momento que quiera.

- **RCS:** Es un modo muy elegante de administrar versiones de archivos que permite trabajar fácilmente a más de un desarrollador en el mismo proyecto.
- **RSS (Really Simple Syndication):** Del inglés Sindicación Realmente Simple. Es usado para mandar noticias o contenidos de un sitio web.
- **Servlet:** Aplicación sin interfaz gráfica que se ejecuta en un servidor de Internet, procesando información HTML previamente recogida por un navegador.
- **Sistema de Control de Versiones:** Véase RCS.
- **Spring Framework:** Framework desarrollado por SpringSource para desarrollar aplicaciones web.
- **UML:** (Unifed Modeling Languaje) Es un lenguaje para la especificación, visualización, construcción y documentación de los artefactos y procesos del sistema.
- **URL:** (Universal Resource Locator). Del inglés Localizador Universal de Recursos. Sistema unificado de identificación de recursos en la red. Las direcciones se componen de protocolo ("*http:*", "*ftp:*", "*mailto:*", etc.), autoridad ("*//www.javahispano.org*") y ruta del recurso dentro del servidor ("*/user/profile/*").
- **XHTML (eXtensible Hyper Text Markup Language):** Del inglés Lenguaje eXtensible de Marcado de HiperTexto), XHTML es la versión XML de HTML, por lo que tiene, básicamente, las mismas funcionalidades, pero cumple las especificaciones, más estrictas, de XML.
- **XML (Extensible Markup Language):** Es un meta-lenguaje que permite definir lenguajes de marcado adecuados a usos determinados.

ANEXO II – Articulo "Desarrollo de la v3 de javaHispano"

Xavier Lluch Urrutia
Escola Universitaria Politècnica de Mataró
Ingeniería Técnica en Informática de Gestión

Desarrollo de la v3 de www.javahispano.org

Resumen: La web javaHispano es un portal social de temática tecnológica centrada con todo aquello que esté relacionado con el lenguaje de programación Java. javaHispano es una comunidad de habla hispana y todos los recursos que allí se encuentran, están en este idioma.

Este artículo sintetiza el trabajo realizado como uno de los desarrolladores de la nueva versión del portal javaHispano. Se basa en estudiar el sistema actual, analizar los requisitos del nuevo sistema y diseñar e implementar las funcionalidades básicas referentes al usuario.

El proyecto javaHispan v3 es de código libre. Otro objetivo es que el código sirva para mostrar buenas formas a la hora de programar y que sirva como ejemplo práctico en la aplicación de tecnologías de última generación.

1. Introducción.

La comunidad javaHispano lleva desde el año 2001 informando a los usuarios sobre todo lo que rodea al mundo del lenguaje de programación Java. Es una red social donde la información es aportada por los propios usuarios.

El portal fue creado con el proyecto Cáñamo, un sistema de gestión de contenidos creado por la empresa New High Technologies of Norwick S.L.

Al ser un framework propietario que usa un grupo reducido de usuarios, es más complicado encontrar a gente que quiera desarrollar una funcionalidad nueva. javaHispano

Los Administradores de javaHispano decidieron que tenia que hacerse un proyecto nuevo utilizando tecnologías que estén extendidas y preferiblemente con Java.

2. Planteamiento inicial

Los objetivos son la definición y desarrollo del registro y gestión del usuario, la seguridad de acceso del usuario a los recursos de la aplicación, maximizar la usabilidad y la interacción del usuario con el sistema y la integración del uso de algunos recursos externos cómo el sistema de verificación de usuarios OpenId[1].

Un avance en la usabilidad del portal, respecto a la versión anterior de javaHispano, es la fusión de los conceptos de login y registro de usuario. Quedando solo el concepto de login. Cuando intenta hacer login un usuario que no está registrado, el sistema lo detecta y extrae sus datos del proveedor.

También se quiere mejorar los aspectos de interacción social entre usuarios y de la motivación para que estos aporten más contenidos a la comunidad jH y con mejor calidad.

3. Desarrollo del sistema

Para la definición de la arquitectura de la aplicación se ha escogido Spring Framework 3[1], de SpringSource. Puesto que es uno de los frameworks más conocidos en lenguaje de programación Java.

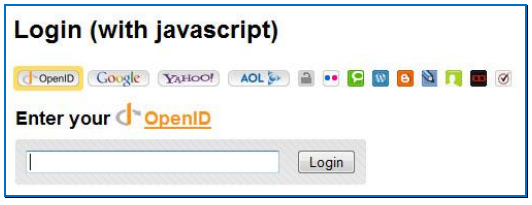
Todo el sistema de login y seguridad es gestionado por el framework Spring Security 3[2], otro proyecto de SpringSource.

Para el control del desarrollo de este proyecto se han seguido las directrices que marca el proceso Scrum[3]. Se definen unas metas de tiempo en función del esfuerzo necesario para desarrollar unas tareas. Periódicamente se hacen reuniones entre el “ScrumMaster” (administrador del proyecto) y los “*Team members*” (desarrolladores) para revisar el trabajo realizado.

El proyecto “javaHispano v3” redefine la forma en que los usuarios ingresan en el portal social. Dispone del sistema de identificación global de usuarios OpenID. El usuario que disponga de una cuenta en un proveedor OpenID podrá registrarse o “loguearse” en el sistema.

La intención del sistema OpenID es permitir a un usuario tener acceso rápido a muchos portales teniendo un único registro, el de OpenID. Eso evita tener que recordar múltiples nombres de usuario y contraseñas. Algo parecido a Microsoft con su “*Windows Live ID*".

Cuando un usuario registrado intenta hacer login, tiene que indicar que proveedor quiere utilizar. Entonces el sistema redirecciona al usuario hacia la página de identificación de su proveedor y este verifica a jH su autenticidad.



Un avance en la usabilidad del portal, respecto a la versión anterior de javaHispano, es la fusión de los conceptos de login y registro de usuario. Quedando solo el concepto de login.

Entonces, cuando intenta hacer login un usuario que no está registrado, el sistema lo detecta y extrae sus datos del proveedor.

Al usuario le aparece el formulario de registro con los campos ya informados. De esta forma el usuario no tiene que rellenar ningún campo, sólo verificar que los datos recuperados por el sistema son correctos. Así se consigue que el proceso de registro sea más ágil.

Los usuarios también disponen de un perfil propio donde se muestran sus datos personales. Para el avatar (imagen del perfil), se utiliza un sistema externo llamado Gravatar[4]. Utilizando el email del usuario, el sistema puede adquirir la foto del usuario del servidor externo, en el caso de que este tuviera una cuenta en Gravatar.

El proyecto "javaHispano v3" tiene definidas unas directivas de seguridad básicas para supervisar el acceso a recursos con la debida autorización.

4. Problemas encontrados

El problema aparece cuando se quiere obtener los atributos del usuario del proveedor de OpenId. No son accesibles en el momento en que Spring Security detecta que ese usuario no esta registrado en el sistema.

Para solucionar esto ha sido necesario redefinir el filtro de seguridad *OPENID_FILTER* de Spring. Estudiando a fondo el código fuente del framework de seguridad ha sido posible entender bien el funcionamiento interno de este y, de esta forma se ha conseguido hacer un *bypass* en el framework de seguridad.

5. Ampliaciones

Una de las ampliaciones previstas es la de implantar un sistema de medallas con el fin de motivar la interacción de los usuarios con el sistema.

Este sistema de medallas consiste en gratificar simbólicamente al usuario por realizar ciertas acciones que enriquezcan el contenido de la comunidad (aportaciones de contenido, comentar contenidos de otros usuarios, etc.).

6. Conclusiones

Se han cumplido satisfactoriamente los objetivos marcados para la fase inicial del proyecto.

El resultado es una aplicación potente, robusta, reutilizable y escalable. Además, pretende ser una herramienta instructiva y un proyecto de referencia para aquel que esté interesado en ver aplicada alguna de las tecnologías más modernas.

Con la creación del nuevo sistema javaHispano, la comunidad obtiene numerosos beneficios.

- Mediante el uso de frameworks muy conocidos y extendidos, se aumenta la posibilidad de que desarrolladores que tengan conocimientos sobre alguno de estos, puedan colaborar en la creación de nuevas funcionalidades.

- Se pone a disposición de los usuarios, el código fuente de un proyecto web que utiliza tecnologías de última generación. Con una metodología de programación limpia y homogénea.

- La reingeniería ha permitido aplicar mecanismos que facilitan la interacción y la accesibilidad del usuario con el sistema.

Una característica a destacar de todo lo que se ha utilizado en el proyecto (tanto tecnologías como herramientas) es que todo el software utilizado es de código libre. De esta forma queda demostrado que crear un portal de código libre es posible y no queda atrás comparado con otras soluciones propietarias.

7. Bibliografía

[1] Spring Framework 3, http://www.springsource.org/about, junio del 2010

[2] Spring Security 3, http://static.springsource.org/spring-security/site/index.html, junio del 2010

[3] Scrum (development), http://en.wikipedia.org/wiki/Scrum_%28development%29, junio del 2010

ANEXO III – Contenido del CD

Carpeta Documentación

- XavierLluchUrrutia – Memoria.pdf
- XavierLluchUrrutia - Article.doc
- XavierLluchUrrutia - Resum.doc

Carpeta Proyecto

- Archivo javaHispano.rar con la aplicación del portal javaHispano v3.

Carpeta Herramientas de desarrollo

- SpringSource Tool Suite (IDE)
- Java Developer Kit (Kit para poder desarrollar aplicaciones Java)
- PostgresQL (Gestor de bases de datos)
- Apache Tomcat 6.0 (Servidor de aplicaciones)
- Git (Sistema de control de versiones)

Bibliografía

[1] www.javahispano.org , Comunidad social de habla hispana centrada en el lenguaje de programación Java.

[2] http://es.groups.yahoo.com/group/javaHispano, Grupo que se uso en los inicios de javaHispano.

[3] http://web.archive.org/web/20031018111329/canyamo.sourceforge.net/, Página oficial del proyecto Cáñamo.

[4] www.stackoverflow.com, Portal de Q&A (preguntas y respuestas) sobre temas de desarrollo de aplicaciones.

[5] http://www.osqa.net/, Proyecto "Open Source Q&A" que implementa un sistema de medallas similar al que se incluirá a jHv3.

[6] http://maven.apache.org/, Página oficial del proyecto maven.

[7] http://static.springsource.org/spring/docs/3.0.2.RELEASE/spring-framework-reference/pdf/spring-framework-reference.pdf, Documento de referencia de Spring Framework 3.0.2. 02-06-2010

[8] http://blog.springsource.com/2009/03/08/rest-in-spring-3-mvc/,

[9] http://blog.springsource.com/2007/11/14/annotated-web-mvc-controllers-in-spring-25/, Arjen Poutsma, REST in Spring 3: @MVC. 14-11-2007

[10] http://static.springsource.org/spring-security/site/index.html, Página oficial de Spring Security.

[11] http://www.mybatis.org, Página oficial de iBatis 2.

[12] http://www.postgresql.org/, Página oficial del sistema gestor de bases de datos PostgreSQL.

[13] http://freemarker.sourceforge.net/, Pagina oficial del motor de platillas Freemarker.

[14] http://www.springsource.com/products/sts, Pagina oficial del IDE SpringSource Tool Suite.

[15] http://tomcat.apache.org/, Pagina oficial del servidor de aplicaciones web Apache Tomcat.

[16] http://git-scm.com/, Pagina oficial del sistema control de versiones Git.

[17] https://github.com/, Pagina oficial del host de repositorios Git GitHub.

[18] http://openid.es/, Pagina oficial del servicio de identificación global OpenID.

[19] http://en.gravatar.com/, Pagina oficial del sistema global de avatares Gravatar.

[20] http://bit.ly/dhDpTN, Lars Vogel, Dependency Injection with Spring Framework. 30-08-2009