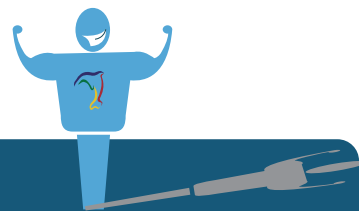


¿Qué ofrece Autentia Real Business Solutions S.L?

Somos su empresa de **Soporte a Desarrollo Informático**.
 Ese apoyo que siempre quiso tener...

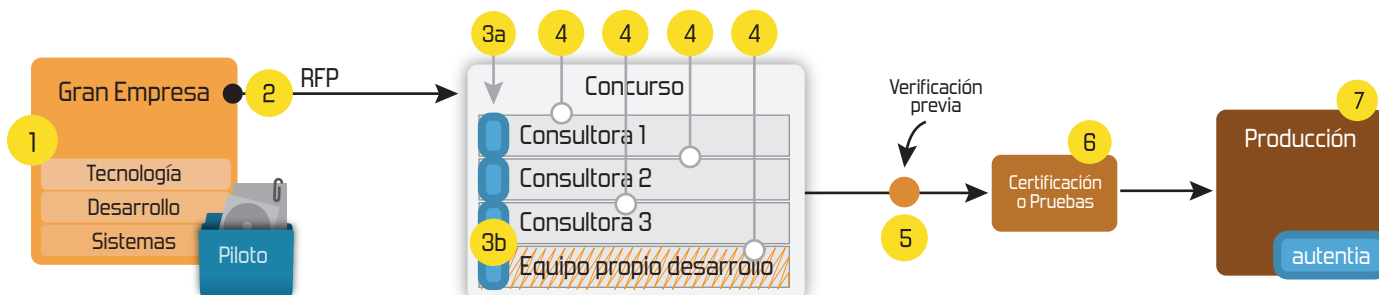
1. Desarrollo de componentes y proyectos a medida



2. Auditoría de código y recomendaciones de mejora

3. Arranque de proyectos basados en nuevas tecnologías

1. Definición de frameworks corporativos.
2. Transferencia de conocimiento de nuevas arquitecturas.
3. Soporte al arranque de proyectos.
4. Auditoría preventiva periódica de calidad.
5. Revisión previa a la certificación de proyectos.
6. Extensión de capacidad de equipos de calidad.
7. Identificación de problemas en producción.



4. Cursos de formación (impartidos por desarrolladores en activo)

Spring MVC, JSF-PrimeFaces /RichFaces,
 HTML5, CSS3, JavaScript-jQuery

Gestor portales (Liferay)
 Gestor de contenidos (Alfresco)
 Aplicaciones híbridas

Tareas programadas (Quartz)
 Gestor documental (Alfresco)
 Inversión de control (Spring)

Control de autenticación y
 acceso (Spring Security)
 UDDI
 Web Services
 Rest Services
 Social SSO
 SSO (Cas)

JPA-Hibernate, MyBatis
 Motor de búsqueda empresarial (Solr)
 ETL (Talend)

Dirección de Proyectos Informáticos.
 Metodologías ágiles
 Patrones de diseño
 TDD

BPM (jBPM o Bonita)
 Generación de informes (JasperReport)
 ESB (Open ESB)

**adictos** al trabajo**Autentia**
business solutions
patrocinado por
datosE-mail: Contraseña: [Deseo registrarme](#) [Entrar](#)
He olvidado mis datos de acceso[Inicio](#) [Quiénes somos](#) [Tutoriales](#) [Formación](#) [Comparador de salarios](#) [Nuestro libro](#) [Charlas](#) [Más](#)

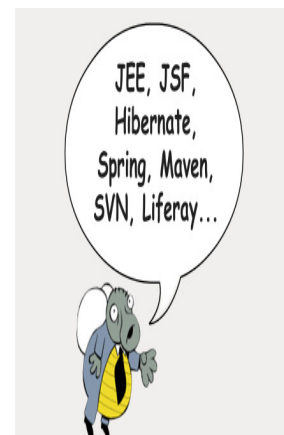
❖ Estás en:

[Inicio](#) [Tutoriales](#) [Ejemplo de arquitectura propuesta por Autentia](#) DESARROLLADO POR: [Alejandro Pérez García](#)

Alejandro es socio fundador de Autentia y nuestro experto en J2EE, Linux y optimización de aplicaciones empresariales.

Ingeniero en Informática y Certified ScrumMaster

Si te gusta lo que ves, puedes contratarle para darte ayuda con soporte experto, impartir **cursos presenciales** en tu empresa o para que **realicemos tus proyectos como factoría** (Madrid). Puedes encontrarme en [Autentia](#): Ofrecemos servicios de soporte a desarrollo, factoría y formación

[Catálogo de servicios Autentia](#)[Últimas Noticias](#) [Corto sobre Metodologías Ágiles](#) [Comentando el Libro: Piensa, es gratis de Joaquín Lorente.](#) [Problemas de aspecto en AdictosAlTrabajo.com: Refresco de la hoja de estilo.](#) [Comentando el libro: Lucro sucio de Joseph Heath](#) [Nuevo formato de tutoriales podcast con bolígrafo LiveScribe](#) [Histórico de NOTICIAS](#)**Fecha de publicación del tutorial: 2010-07-22**[Share](#) |[Regístrate para votar](#)

Ejemplo de arquitectura propuesta por Autentia

Creación: 22-07-2010

Índice de contenidos

1. Introducción
2. Arquitectura propuesta
3. Entorno y metodología del equipo de desarrollo
4. Conclusiones
5. Sobre el autor

1. Introducción

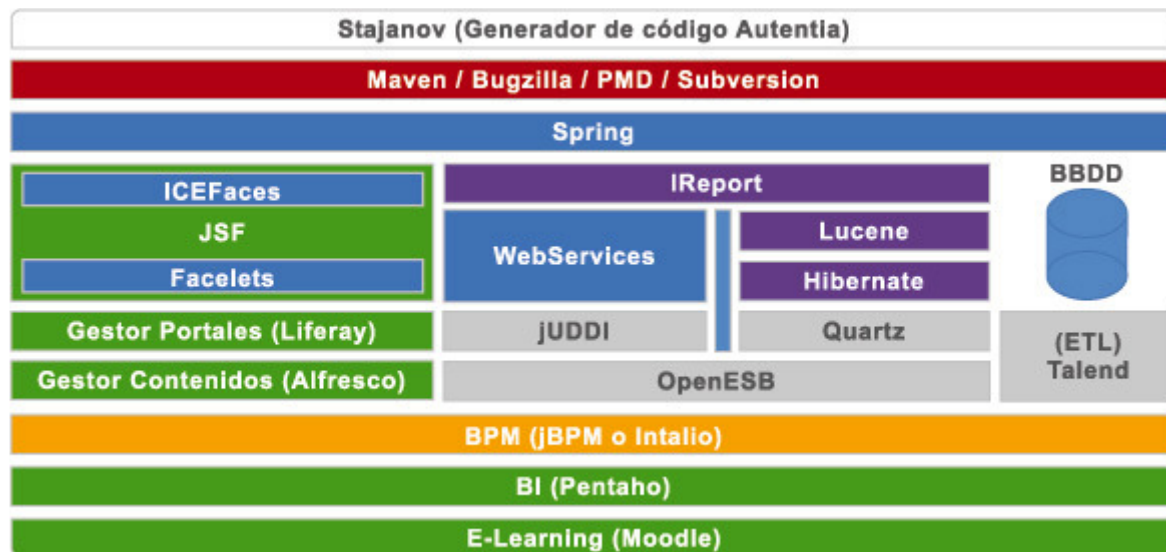
En este documento se describe la arquitectura de proyecto que propone Autentia para llevar a cabo la implementación técnica de sus soluciones de negocio.

La elección de las tecnologías que forman parte de dicha esta arquitectura está basada en estándares, de fábrica y de facto, ampliamente extendidos y aceptados en el mercado, así como en

[Últimos Tutoriales](#) [Creación de servicios web RESTful con el soporte](#)

la experiencia de hacer uso de estas y muchas otras tecnologías en proyectos reales, con la experiencia que supone haber experimentado las ventajas e inconvenientes de cada una de ellas.

No hay que olvidar que esto no es más que un ejemplo, y al final la elección va a depender mucho de las necesidades e infraestructura del cliente. De hecho hemos trabajado con éxito con JBoss Seam, Struts, AppFuse, ... Además no hay que olvidar que hay muchas formas de hacer las cosas y que también tenemos a nuestra disposición Gestores de Contenidos, BPM, Gestores de Portales, ... A continuación podéis ver un pequeño gráfico de como podríamos estructurar todas estas capas y tecnologías.



Quiero aprovechar para agradecer a mi compañero

Jose Manuel Sánchez Suarez su colaboración en este artículo, colaboración sin la cual esto no habría sido posible.

2. Arquitectura propuesta

La arquitectura propuesta se basa en las siguientes tecnologías: JSF con jQuery y Primefaces, integración de servicios con Spring, persistencia con JPA con el soporte de Hibernate.

En la siguiente figura se presenta un esquema donde se puede ver donde aplica cada una de las tecnologías mencionadas. Identificando si se ejecuta en cliente o en el servidor, así como la capa a la que pertenece: presentación y control, negocio o persistencia.

de RESTeasy de Jboss Seam.

- Facebook Social Plugins
- EGit un plugin de Eclipse para el sistema de control de versiones distribuido Git
- Crear una estructura compleja del tipo Array en un servicio web Axis2
- Apache TCPMON : El Sniffer de los Servicios Web

Últimos Tutoriales del Autor

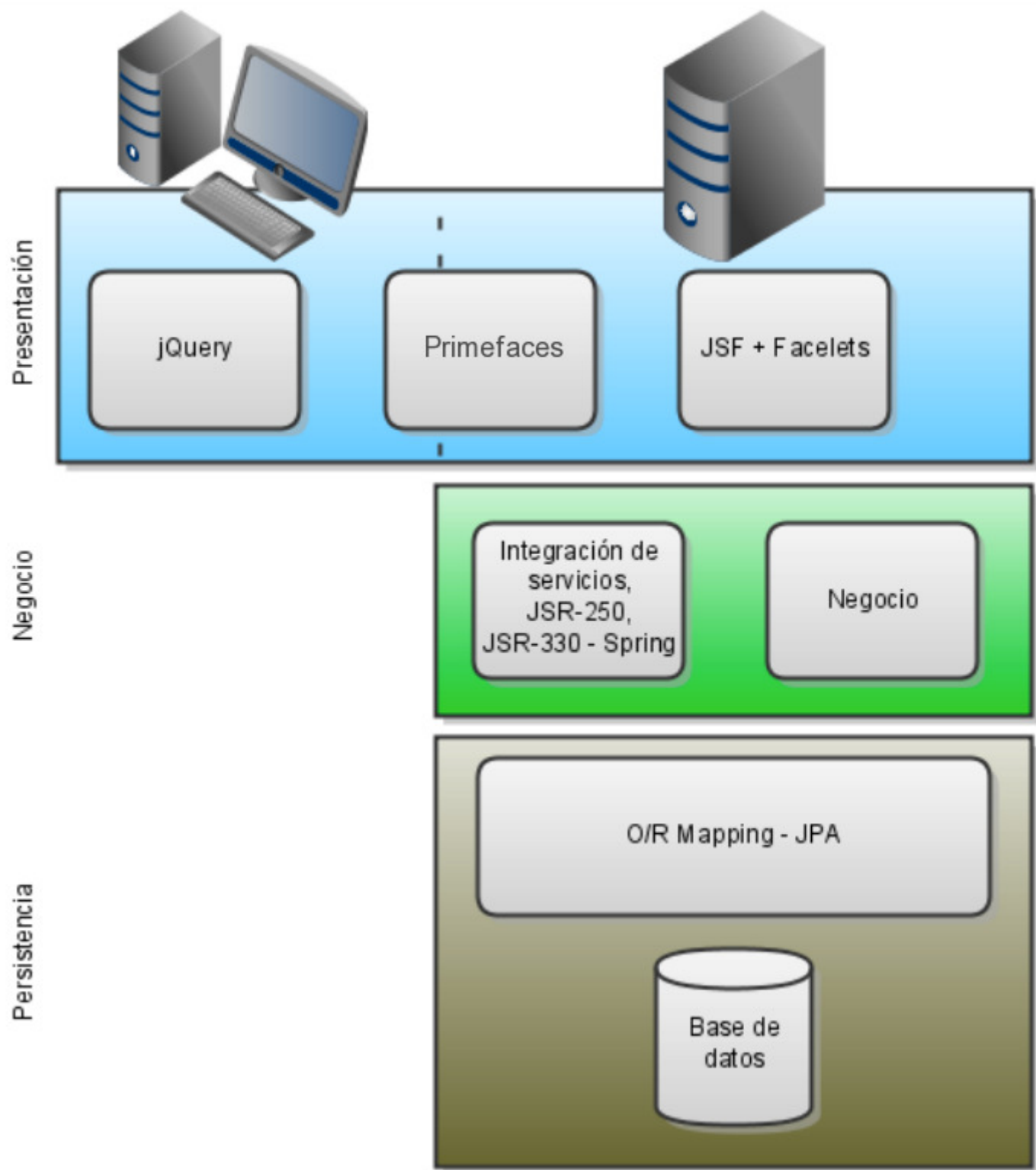
- EGit un plugin de Eclipse para el sistema de control de versiones distribuido Git
- Git y cómo trabajar con un repositorio de código distribuido
- Gestión de proyectos ágiles con Pivotal Tracker
- Eclipse Helios, la nueva versión 3.6 de Eclipse
- Mac Automator y como renombrar múltiples ficheros

Síguenos a través de:



Últimas ofertas de empleo

2010-06-25
T. Información - Analista / Programador - BARCELONA.



A continuación se lleva a cabo un breve detalle de cada una de las tecnologías incluidas en el esquema.

2.1 JSF + Facelets

Java Server Faces es un estándar dentro de la pila JEE (Java Enterprise Edition) que permite el desarrollo de la capa de presentación y control para aplicaciones Web.

JSF permite el desarrollo de aplicaciones Web en base a componentes y eventos. Aproximación típica para el desarrollo de aplicaciones interactivas en entornos de escritorio. De esta manera abandona la aproximación tradicional y más limitada de "petición/respuesta", en la que se trabaja básicamente con documentos y apuesta por el trabajo con pantallas de interacción.

Gracias a los componentes de JSF y Facelets es mucho más sencillo reutilizar piezas visuales (listas, controles, campos de entrada, ...) ya que estos se pueden "paquetizar" en forma de componentes por composición, que resultarán muy fáciles de usar en distintas partes de la aplicación, cómo si se tratara de un componente estándar.

Además, la idea de gestión de la interacción entre el usuario y la máquina en base a eventos, permite realizar interfaces de usuario mucho más ricas desde el punto de vista de la usabilidad, y simplifica el desarrollo de las mismas. Acercándose cada vez más las aplicaciones Web a las funcionalidades visuales típicamente ofrecidas por las aplicaciones de escritorio.

JSF se encuentra actualmente en su versión 2.x. Este nueva versión da soporte directamente a tecnología AJAX (Asynchronous JavaScript and XML), lo que la hace aún más atractiva, ya que gracias a AJAX se pueden actualizar partes de la página sin necesidad de hacer una petición completa al servidor. De esta forma se mejora el ancho de banda ya que por la red viajan menos datos, y se mejora la experiencia del usuario ya que se evitan incómodos parpadeos (al recargar la página completa) y se mejora la usabilidad haciendo las páginas más dinámicas.

Otras alternativas descartadas

- **Servlets + JSP:** También pertenecen al estándar JEE, pero estaríamos hablando de una capa de abstracción varios niveles por debajo de JSF. Esto quiere decir que hay que hacer muchas más cosas a mano, siendo los desarrollos más complejos (todo depende del desarrollador) y necesitando más tiempo para intentar conseguir los mismos resultados.
- **Struts:** la versión 1.2.x ha sido durante algunos años el framework más conocido y demandado en el mundo JEE; cuando el estándar no daba más de sí, Struts cubría la carencia de una separación entre capas y el soporte de ciertas piezas para generar la vista y controlar el mecanismo petición/respuesta entre cliente y servidor. En la actualidad existe una versión 2 que no ha dado continuidad a la versión 1.2.x, puesto que se basa en un framework distinto (Apache WebWork); ahora Struts se parece bastante a JSF en cuanto a la gestión de eventos. Teniendo el estándar y sus componentes visuales nos decantamos por la línea oficial de JSF.
- **GWT:** Google Web Toolkit. Es una tecnología pensada para desarrollar aplicaciones Web usando la misma idea de componentes y eventos. El problema de esta tecnología es que es propietaria de Google. Esto no tiene por que ser malo, pero lo que sucede es que en el mercado se encuentra mucho más soporte y muchos más profesionales con experiencia en JSF que en GWT. Además la separación entre el cliente y el servidor no está del todo clara pudiendo, si no se tiene cuidado, llevar al cliente (navegador Web) lógica de negocio.

2.2 Primefaces

Primefaces es una librería de componentes para JSF. Estos componentes aportan, frente a los componentes estándar de JSF, una abstracción para el uso de la tecnología AJAX ya soportada en JSF 2. Es decir, el desarrollador puede centrarse en la funcionalidad ofrecida sin tener que preocuparse del JavaScript que se ejecutará en el cliente o de que partes de la pantalla serán necesarias refrescar en respuesta de un evento en la interfaz de usuario.

No siendo Primefaces parte del estándar JEE, ahora es la única librería de componentes visuales que podemos decir que soporta de manera estable la versión 2 de JSF.

Otras alternativas descartadas

Podemos encontrar otras librerías de componentes como **MyFaces Tomahawk** de Apache, **RichFaces** de Jboss o **ICEfaces** de ICEsoft. En el caso de la primera las funcionalidades ofrecidas se quedan un poco cortas comparadas con las del resto; en el caso de la segunda el modelo que sigue obliga al desarrollador a tener más control sobre lo que quiere refrescar, además de ser un poco más "cerrada" al estar claramente orientado su uso junto con Seam; en el caso de la tercera, ha sido el estándar de facto y forma parte de la solución RIA de Sun (Oracle) para JSF 1.2, si bien, aun no proporciona un soporte estable para JSF2.

En cualquier caso, podemos decir que tenemos proyectos con todas las librerías de componentes JSF comentadas, puesto que en función de las necesidades o preferencias de nuestros clientes nos hemos tenido que mover entre todas ellas.

2.3 jQuery

jQuery es una librería JavaScript para conseguir una interfaz rica de usuario en aplicaciones Web (RIA - Rich Internet Application). Esto quiere decir que esta librería JavaScript nos permitirá dotar a las páginas Web de efectos visuales para asemejar su uso al de aplicaciones tradicionales de escritorio, mejorando en gran medida la usabilidad de las mismas.

Primefaces se apoya en el uso extensivo de jQuery para la comunicación Ajax y el uso de componentes visuales, aunque también hace uso de la librería Javascript YUI de Yahoo.

Si bien es verdad que con JSF + Facelets + Primefaces es raro tener que llegar a escribir una sola línea de JavaScript, ya que estas piezas nos ofrecen gran cantidad de componentes pensados para RIA, si quisiéramos crear un nuevo componente con efectos visuales sin duda la elección sería jQuery para eliminar en gran medida el JavaScript propio, y así minimizar los posibles errores, por ejemplo, producidos por la ejecución de la página en distintos navegadores (Explorer, Firefox, Chrome, Safari, ...).

Otras alternativas descartadas

Dentro del mundo Web y el HTML hay gran cantidad de librerías para manejo de JavaScript. La otra gran opción a jQuery sería **Prototype y Scriptaculo**. Si bien jQuery ha venido a suceder como

estándar de facto a Prototype, por el éxito en su difusión, la sencillez de uso, el hecho de ser plugable y disponer de cientos de plugins gratuitos.

De todas formas, igual que decíamos con las alternativas de Primefaces, estas piezas no dejan de ser librerías de funciones JavaScript, por lo que se podría llegar a combinar jQuery con otras librerías. Es decir no habría por que usar única y exclusivamente jQuery.

2.4 Spring

Spring son muchas cosas, pero lo que nos interesa en este punto es usar Spring como contenedor de Inversión de Control (IoC - Inversion Of Control). Este patrón, también conocido como Inyección de Dependencias o incluso Principio de Hollywood, aporta a los desarrollos la facilidad para integrar distintas piezas y servicios, bajando mucho el acoplamiento entre estas piezas, y de esta manera garantizando un mejor mantenimiento y evolución de los diseños.

Como efecto secundario de este bajo acoplamiento, conseguimos facilitar la tarea de pruebas unitarias o de integración, ya que es muy fácil cambiar un servicio por un Mock (servicio de broma, servicio dummy, servicio que simula el comportamiento del real pero en un entorno controlado). Adicionalmente estos Mocks permiten a grupos de desarrollo avanzar sin necesidad de esperar para que otro grupo termine otra pieza de la que depende el primero. Simplemente simulan la pieza y cuando el segundo grupo termina el desarrollo se da el cambiao de una por otra.

Otras de las grandes ventajas que aporta Spring es que soporta los estándares de inyección de dependencias JSR-250 y JSR-330, por lo que su uso no ata al proveedor. Además Spring es un estándar de facto ampliamente extendido y aceptado en el mercado (tanto es así que inspiró en gran medida la nueva especificación de EJB3).

Adicionalmente Spring proporciona integración y facilidades para trabajar con otra gran cantidad de tecnologías, como puede ser gestión de seguridad, transaccionalidad, JSF, JPA, correo, Servicios Web, EJB, tareas programadas, ... Esto va a facilitar mucho el trabajo del desarrollador, ya que se encontrará con multitud de clases y servicios para realizar las tareas comunes en cualquier proyecto.

Otras alternativas descartadas

Existen otros contenedores de IoC como **PicoContainer** y **Google Guice**, ambos cumplen los mismos estándares que Spring por lo que se podría llegar a cambiar uno por otro, pero se elige Spring sobre los demás por su madurez (está mucho más extendido y aceptado en la comunidad) y por la gran cantidad de utilidades que ofrece para integrar nuestro negocio con otras tecnologías.

Otra opción sería **EJB3** o **Jboss Seam** (Seam se basa en EJB3). Se descartan estas tecnologías porque son mucho más pesadas, sobre todo a la hora del desarrollo, aumentando el tiempo de desarrollo y la frustración del desarrollador cada vez que tiene que hacer un despliegue en el servidor, ya que se necesitan servidores de aplicaciones tipo JBoss, WebSphere, WebLogic o Glassfish. Todas las tecnologías anteriormente mencionadas pueden desplegarse tanto en estos servidores como en servidores mucho más ligeros estilo Apache Tomcat o Jetty.

En cualquier caso hay que entender que EJB3 y Spring no son competidores, sino compañeros. Para entender esto hay que entender la principal motivación de los EJB (Enterprise Java Bean): Un EJB es un objeto Java remoto, esto quiere decir que el desarrollador puede trabajar con un objeto como si este estuviera en la misma máquina, aunque en realidad el objeto se puede encontrar en cualquier servidor de la red. Este nivel de abstracción requiere una serie de capas de comunicaciones que tiene su complejidad y penalización.

Otra característica de los EJB es que permite hacer de forma sencilla transacciones distribuidas por distintas fuentes de datos. Esto que suena relativamente fácil, luego no lo es tanto, siendo necesario en muchos casos implementar mecanismos de compensación.

Una vez comprendidas las ventajas pueden aportar los EJB, sería factible hacer un desarrollo basado en Spring, que es mucho más ligero, y usar EJB para aquellos servicios donde realmente les saquemos partido. Spring se integrará perfectamente con estos EJB.

2.5 JPA (Hibernate)

JPA (Java Persistence API) es un estándar de Java (JSR-220) que permite a los desarrolladores trabajar de forma sencilla con bases de datos relacionales. JPA permite la abstracción de la fuente de datos, permitiendo que el código sea portable entre distintos sistemas gestores de bases de datos: Oracle, DB2, MySQL, PostgreSQL, ...

Gracias a JPA el desarrollador se puede centrar en la lógica de negocio, olvidando los detalles de implementación la capa de acceso a datos. Esto permite realizar desarrollos más rápidos y más seguros.

Hibernate es un estándar de facto ampliamente aceptado y extendido en el mercado. Tanto que la implementación de referencia de JPA es el EntityManager de Hibernate.

En la medida de lo posible, tratamos de trabajar con la especificación, de modo que la implementación que usemos por entorno o servidor de aplicaciones debería ser transparente. Una aplicación basada en JPA, aún usando Hibernate, sin salirse del estándar podría hacerse correr bajo el soporte de OpenJPA, por ejemplo.

No obstante lo anterior, si el proyecto lo requiere, podemos hacer uso de ciertas facilidades de Hibernate que no forman parte del estándar, pero imprimen una riqueza del lado de la capa de persistencia como el soporte para una caché de segundo nivel o la integración con un motor de indexación y recuperación de contenidos textuales como es Apache Lucene.

Otras alternativas descartadas

- **EJB3** dispone de un tipo de EJB, los de Entidad, que permiten gestionar la persistencia de la información. Precisamente estos EJB de Entidad se basan en JPA para realizar su labor. Esto quiere decir que cualquier desarrollo con JPA y sin usar EJB sería fácilmente portable a un entorno de EJB si fuera necesario. Y la ventaja de trabajar directamente con JPA en lugar de con EJB es, como ya se ha mencionado anteriormente, que los EJB son muy pesados sobre todo en tiempo de desarrollo, alargando los periodos de construcción del sistema.

3. Entorno y metodología del equipo de desarrollo

En este punto describimos el entorno de trabajo recomendado, tanto a nivel de herramientas como de procedimientos.

En primer lugar indicar la versión de la máquina virtual Java a utilizar. La recomendación es el uso de la versión 6 debido a sus nuevas características, y especialmente por las mejoras de rendimiento realizadas en esta última versión.

Como mínimo sería necesario una versión 5 de la JVM. Este requisito es muy importante ya que a partir de la versión 5 es cuando el lenguaje Java incorpora la idea de Anotaciones. Estas anotaciones son metainformación que se indica en el código mediante el símbolo @. Y gracias a estas anotaciones se va a conseguir simplificar enormemente el desarrollo con todas las tecnologías anteriormente indicadas ya que los ficheros XML de configuración se van a reducir a una mínima expresión.

De esta forma no sólo se desarrolla más rápido porque no hay necesidad de escribir las cosas dos veces (muchas veces lo mismo que se pone en la clase se acaba poniendo también en un fichero de configuración XML), sino que además se evitarán los típicos errores de "impedancia" en el código (se dice que hay errores de "impedancia" en el código cuando existe una desincronización entre lo que pone en la clase y lo que pone en el ficheros XML de configuración, causando errores sólo detectables en tiempo de ejecución).

3.1 Scrum y ciclos de desarrollo de 3 semanas (15 jornadas)

Scrum es una metodología ágil que pretende ayudarnos a realizar la gestión del proyecto.

Hay mucha literatura sobre Scrum, pero en estas líneas simplemente podríamos quedarnos con la idea del desarrollo iterativo incremental. Es decir, el desarrollo se divide en iteraciones de duración constante, 15 jornadas laborales. Al principio de cada iteración el cliente fija las prioridades de los requisitos, es decir, indica que es lo que quiere que se haga primero porque aporta más valor al negocio. Una vez definidas las prioridades, el equipo realiza una estimación de las tareas y adquiere un compromiso con el cliente sobre que funcionalidades estarán disponibles al final de la iteración. Al final de la iteración se hace una entrega y una demostración al cliente de las funcionalidades acordadas.

Las ventajas de esta forma de desarrollo son:

- El cliente tiene un control total sobre el desarrollo ya que si hay cualquier desviación o cambio en los requisitos, estos se podrán detectar en un plazo máximo de una iteración (15 jornadas). Esto permite una adaptación a las necesidades del mercado muy importante, permitiendo posicionarnos por delante de nuestros competidores porque somos más rápidos cubriendo las nuevas necesidades de los usuarios.
- El cliente no tiene que esperar largos periodos de tiempo para ver el sistema funcionando. Esto se consigue gracias al desarrollo vertical de funcionalidades. Es decir, no se hace primero la capa de base de datos, luego la de negocio, luego la de pantallas, ... Sino que se hace un desarrollo por funcionalidades de negocio, primero el listado de vuelos, luego la compra, luego el listado de hoteles, ... Así, al final de cada iteración de 3 semanas el cliente es capaz de probar y verificar las nuevas funcionalidades de negocio desarrolladas durante esas 15 jornadas.
- Se puede hacer una pronta salida a producción o preproducción. Esta ventaja sería una consecuencia de la anterior. Al ir desarrollando funcionalidades de negocio completas, sería factible hacer puestas en preproducción o incluso producción para que los usuarios finales empiecen a utilizar el sistema. De esta forma se tiene un feedback muy bueno sobre como el sistema se ajusta a las necesidades de los usuarios finales.
- Se puede parar el proyecto en cualquier momento. Esta ventaja sería una consecuencia de las dos anteriores. En multitud de casos al principio del proyecto se piensa en un sistema ideal. Al ir desarrollando funcionalidades por orden de prioridad (valor de negocio) y al ir haciendo subidas a producción, puede llegar un momento en el que, sin haber terminado de desarrollar todas las funcionalidades pensadas idealmente, se ve que el sistema cubre todas las

necesidades reales de los usuarios finales. En este punto podríamos plantear para el desarrollo sin necesidad de realizar el resto de funcionalidades ideales.

3.2 Pruebas unitarias

Uno de los grandes problemas a la hora de desarrollar software es que al introducir nuevas funcionalidades "se rompen" otras que ya estaban funcionando correctamente, incluso que ya estaban en producción.

Para evitar este problema el mejor método es la realización de pruebas automáticas, ya sean unitarias, de integración, o incluso funcionales.

La idea es conseguir una batería de pruebas automáticas que, dando a un "botón", y en cuestión de segundos, sea capaz de probar las funcionalidades principales del sistema.

De esta forma podemos hacer cambios en el sistema con mucha garantía de que lo que ya estaba hecho no deja de funcionar.

Esta forma de trabajar facilita las refactorizaciones para adaptar el sistema a los nuevos requisitos, y facilita la implementación del patrón de diseño "Cerrado a modificaciones, abierto a ampliaciones". Es decir, siempre se debería intentar que para añadir una nueva funcionalidad no sea necesario cambiar el código ya escrito, pero para conseguir esto es inevitable ir refactorizando el código y los diseños iniciales. Sin pruebas automáticas, esto se convierte en una tarea titánica, y sin ninguna garantía de éxito.

3.3 Maven + Subversion

Subversion es un repositorio de código. Este tipo de servicios nos permiten guardar nuestro código y poder ver el histórico de cambios, hacer etiquetas (versiones) sobre un punto determinado, abrir nuevas ramas de desarrollo, ...

A día de hoy no se concibe un desarrollo sin usar este tipo de herramientas, ya que facilitan la colaboración entre desarrolladores (mientras más grande sea el equipo más se justifica la herramienta), sirve como backup (al igual que una base de datos, todo el código se almacena en un punto central, además de en las máquinas de cada desarrollador), permite identificar errores (cuando los test dejan de funcionar es muy sencillo ver todos los cambios que se hicieron entre el momento actual y la última subida al repositorio, ...

Otra alternativa a Subversion puede ser CVS. Este es otro servidor de control de versiones ampliamente extendido, pero el subversion aporta ciertas ventajas, como es el versionado de metadatos y directorios, y el ser capaz de identificar el estado de todo el repositorio en un punto determinado del tiempo con un solo identificador de revisión (de esta manera, sabiendo el número de revisión podemos obtener una foto completa de todo el repositorio en ese momento del tiempo).

Maven es una herramienta que permite automatizar el proceso de construcción y empaquetado del software.

Las herramientas del estilo del Maven son el complemento perfecto para el Subversion, ya que de no usarlas el proceso de compilación y empaquetado tendría que ser manual, y por lo tanto muy propenso a errores (qué pasa si la persona que normalmente lo hace no ha venido, o qué pasa si se olvida un paso, o ...).

Todos estos errores van a causar que hagamos instalaciones en preproducción, o incluso en producción, de sistemas para los que no somos capaces de identificar el código fuente. El gran problema de esta situación vendrá cuando alguno de nuestros clientes nos reporte una incidencia, y seamos incapaces de reproducirla en nuestro entorno debido a que no estamos trabajando con los mismos fuentes.

En Java, otra alternativa típica a Maven sería Ant. Se recomienda encarecidamente el Maven frente al Ant por las innumerables ventajas que aporta el primero. Por ejemplo, Maven da un ciclo de vida estándar para la construcción, pruebas, empaquetado, ... Maven trae de serie un sistema para la gestión de dependencias tanto directas como indirectas (esto es fundamental para saber contra que librerías estamos compilando nuestro código). También genera documentación, se integra con multitud de herramientas, ... y todo sin escribir una sola línea de script (el principal problema de Ant es que es un "lenguaje" de script y al final todo el mundo hace lo mismo pero de formas distintas, es decir, no hay dos scripts de Ant iguales).

3.4 Hudson + Sonar

Con estas herramientas pretendemos ayudar a la calidad del código.

Hudson es un servidor de integración continua. Hudson se encarga de, cada vez que hay un cambio en el repositorio de código (en el Subversion) compilar todo el proyecto, lanzar las pruebas, compilar otros proyectos dependientes y, en general, lanzar cualquier otra tarea que creamos conveniente (por ejemplo lanzar Sonar).

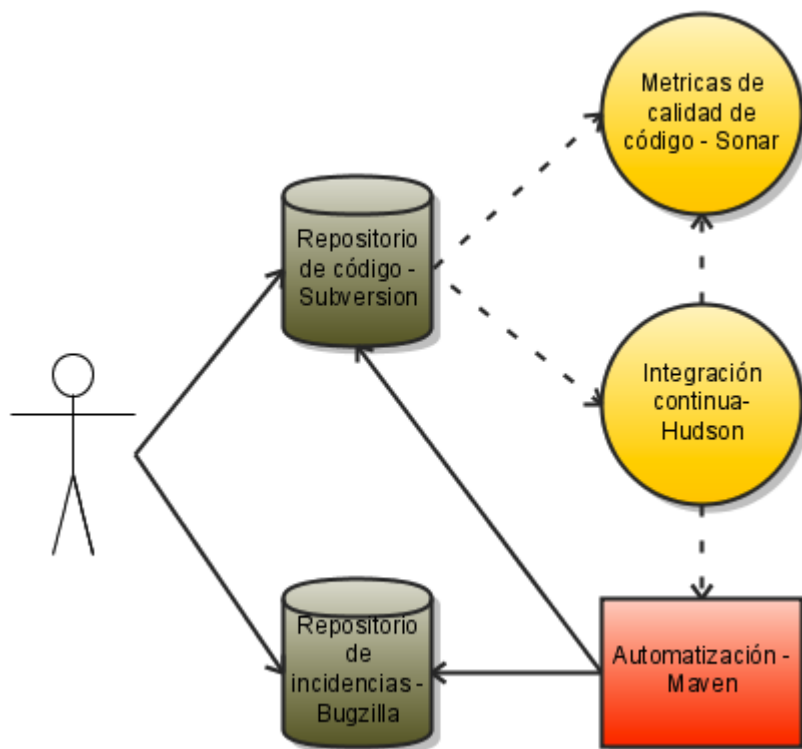
Con esto lo que conseguimos es detectar los posibles errores de integración en cuestión de minutos. Si un desarrollador sube al repositorio código que en su máquina local funcionaba, pero que al integrarlo con el código del resto de compañeros entra en conflicto, Hudson lo detectará en cuestión de minutos y mandará un correo electrónico para avisar del problema.

Además Hudson guarda el histórico, de forma que podemos ver si nuestro código evoluciona adecuadamente, cuantas veces hemos roto al compilación (se sube al repositorio código que no compila), o que test fallan (se sube código que compila pero que no funciona), ...

Sonar es un servidor que, en función de una serie de reglas, es capaz de hacer un análisis estático de nuestro código. Realmente Sonar por debajo se basa en PMD, Checkstyle y FindBugs, de forma que con Sonar podemos ver estas herramientas de una forma unificada, como si se tratara de una sola.

Con Sonar también podemos detectar código duplicado, e incluso medir la cobertura de nuestros test. Es decir, podemos saber que líneas de nuestro código se ejecutan con los test y cuales no.

Al igual que Hudson, Sonar guarda el histórico del proyecto y nos puede mostrar gráficas donde vemos el avance del proyecto.



4. Conclusiones

A día de hoy el mundo del desarrollo es un mar revuelto de plataformas, tecnologías, estándares y procedimientos. Ya pasaron aquellos tiempos "fáciles" en los que todo se reducía a Uno frente a la Máquina. Por eso desde Autentia y www.adictosaltrabajo.com siempre estamos intentando difundir y facilitar el acceso a estas nuevas tecnologías, en contaste evolución. Por eso no dudeís en contactar con nosotros para cualquier problema de arquitectura, desarrollo, auditorías, mentoring, formación, ...; ya que se podría decir que nosotros ya hemos navegado por estas aguas revueltas en numerosas ocasiones y podemos ayudaros a guiaros en la tormenta para no naufragar por culpa de los innumerables escollos.

5. Sobre el autor

Alejandro Pérez García, Ingeniero en Informática (especialidad de Ingeniería del Software) y Certified ScrumMaster

Socio fundador de Autentia (Formación, Consultoría, Desarrollo de sistemas transaccionales)

<mailto:alejandropg@autentia.com>

Autentia Real Business Solutions S.L. - "Soporte a Desarrollo"

<http://www.autentia.com>

Anímate y coméntanos lo que pienses sobre este **TUTORIAL**:

Puedes opinar o comentar cualquier sugerencia que quieras comunicarnos sobre este tutorial; con tu ayuda, podemos ofrecerte un mejor servicio.

Enviar comentario

(Sólo para usuarios registrados)

» **Registrate** y accede a esta y otras ventajas «



SOME RIGHTS RESERVED
2.5

Esta obra está licenciada bajo [licencia Creative Commons de Reconocimiento-No comercial-Sin obras derivadas](#)

Copyright 2003-2010 © All Rights Reserved | [Texto Completo](#) | [Condiciones de uso](#) | [Banners](#) | [Powered by Autentia](#) |

