COMPARATIVA ENTRE EL DESARROLLO WEB USANDO EL FRAMEWORK JBOSS SEAM Y EL DESARROLLO TRADICIONAL

Jorge Ruiz-Robles

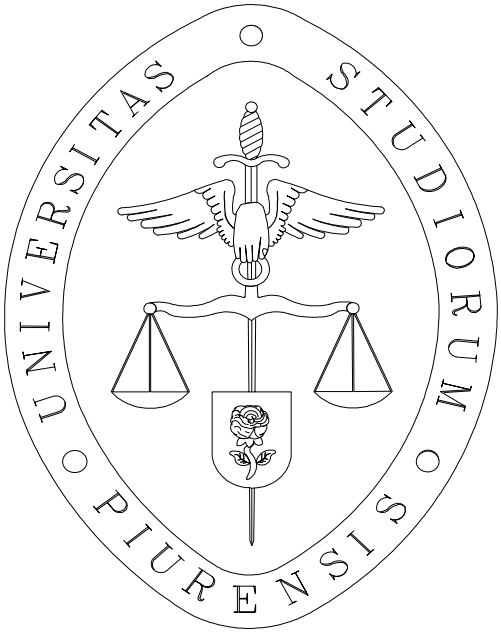
Piura, febrero de 2011

FACULTAD DE INGENIERÍA

Área Departamental de Ingeniería Industrial y Sistemas

Ruiz, J. (2011). *Comparativa entre el desarrollo Web usando el Framework Jboss Seam y el desarrollo tradicional*. Tesis de pregrado no publicado en Ingeniería Industrial y de Sistemas. Universidad de Piura. Facultad de Ingeniería. Programa Académico de Ingeniería Industrial y de Sistemas. Piura, Perú.

UNIVERSIDAD DE PIURA
FACULTAD DE INGENIERIA



COMPARATIVA ENTRE EL DESARROLLO WEB USANDO EL FRAMEWORK JBOSS SEAM Y EL DESARROLLO TRADICIONAL

Tesis para optar el Título de Ingeniero Industrial y de Sistemas

JORGE ROBERTO RUIZ ROBLES

Asesor: Omar Hurtado Jara

Piura, Febrero 2011

A mis padres Marly y Ronald por enseñarme a buscar cada día ser mejor persona y mejor profesional; y a mi hermano Alejandro, por ser siempre un gran ejemplo y un gran apoyo a lo largo de mi formación académica y profesional.

Prólogo

Hoy en día, el desarrollo de aplicaciones Web requiere el uso de diferentes tecnologías que garanticen alta disponibilidad, robustez, calidad y escalabilidad. Con la llegada de la Web 2.0, el diseño de interfaces gráficas de gran usabilidad, altamente amigables y de mucha interacción con el usuario, se ha vuelto imprescindible para cualquier portal o aplicación Web. La naturaleza misma de estos sistemas se encuentra en constante cambio y surgen cada vez nuevos requerimientos. Debido a ello, es necesario optar por una tecnología de desarrollo que se encuentre en la capacidad de satisfacer este entorno cambiante.

JBoss Seam es un entorno de desarrollo (*Framework*) hecho en Java 2 Enterprise Edition (J2EE), que abarca componentes que van desde las interfaces gráficas hasta un motor de persistencia para la gestión de la base de datos. Además, ofrece un desarrollo ágil gracias a sus herramientas para generación automática de código y la reutilización de componentes especializados de Java.

El desarrollo de aplicaciones Web, recurriendo a las nuevas tecnologías para poder cubrir los requerimientos de la actualidad, se ha vuelto sumamente complicado y demandaría periodos mayores de desarrollo. *JBoss Seam* reduce notablemente los tiempos y costos gracias a todas las herramientas que integra; sin embargo, es necesario tener en cuenta que como todo *framework*, *Seam* tiene sus propias particularidades, lo que exige un periodo de aprendizaje antes de comenzar a usarlo.

Resumen

Dado que *Seam* se presenta como una buena alternativa para el desarrollo de aplicaciones Web, el presente trabajo busca comparar los costos y los tiempos del desarrollo de manera tradicional y del desarrollo con el *framework*, así como definir las consideraciones a tener en cuenta antes de optar por usarlo para el desarrollo de una aplicación.

Para efectuar dicha comparación se estudió la evolución de la Web y cómo se convirtió en Web 2.0, el funcionamiento del entorno de desarrollo empresarial de Java, y la arquitectura de las aplicaciones hechas en *Seam*. Luego se evaluó el caso concreto de desarrollo con el *framework* para compararlo con el desarrollo tradicional de la misma aplicación. Finalmente se encontraron las ventajas y casos particulares en que conviene inclinarse por *Seam*.

De todo esto se pudo concluir que *Seam* hace que el desarrollo y mantenimiento de las aplicaciones sea más rápido y fácil. Pero antes de comenzar con un proyecto en *Seam*, es importante tener en cuenta que la infraestructura necesaria siempre será más cara que la que se usa normalmente para desarrollo tradicional.

Índice General

Introducción

En la actualidad, existen dos estudios importantes acerca del framework[1] *JBoss Seam*. El primero de ellos se titula "*JBoss Seam: A Deep Integration Framework*" (Yuan, 2007) y el segundo, "*Which is the Hottest Java Web Framework? Or maybe not Java?*" (Kalla, 2008). Estos estudios no son muy actuales, y dado que *Seam* se encuentra bajo cambios y actualizaciones constantes, las versiones a las que hacen referencia ya están descontinuadas.

"Comparativa entre el Desarrollo Web usando el *Framework JBoss Seam* y el Desarrollo Tradicional" es un trabajo que fue hecho sobre una versión más reciente de *Seam*, con el propósito de determinar costos y tiempos de desarrollo, y casos en los que conviene utilizar el *framework*; teniendo en cuenta las tecnologías más modernas que existen en la actualidad. A continuación se explican los contenidos de cada capítulo.

En el capítulo 1 se busca ilustrar al lector con respecto a lo que está detrás del origen de *Seam*, cuáles son las tendencias actuales de las aplicaciones y las tecnologías Web.

En el capítulo 2 se hace una introducción al *framework JBoss Seam*, dando a conocer sus características y cómo está estructurado un proyecto hecho en *Seam*.

En el capítulo 3 se presenta un caso de estudio, el cual será evaluado desarrollándose tanto de manera tradicional como utilizando el *framework*, de esta forma se tendrá los elementos necesarios para llevar a cabo la comparación.

En el capítulo 4 se definen los factores e indicadores que se utilizarán para las pruebas de calidad y luego se presentan los resultados de las pruebas de calidad.

En el capítulo 5 se presenta un resumen de las características de *Seam*, así como las ventajas, consideraciones y recomendaciones a tener en cuenta al abordar un proyecto de desarrollo Web utilizando el *framework*.

[1] La definición de *framework* se puede encontrar en la página 12

Capítulo I
Marco Teórico

1. Aplicaciones Web

Seam permite la creación de aplicaciones Web generadas automáticamente a partir de una base de datos. Una aplicación Web es un sistema que funciona haciendo uso de las tecnologías Web que operan sobre Internet, esto significa que las interfaces de usuario se presentan en forma de páginas Web. Este tipo de aplicaciones se basan en la arquitectura Cliente-Servidor, y específicamente en una arquitectura de tres capas, tal como se puede apreciar en la Figura 1.

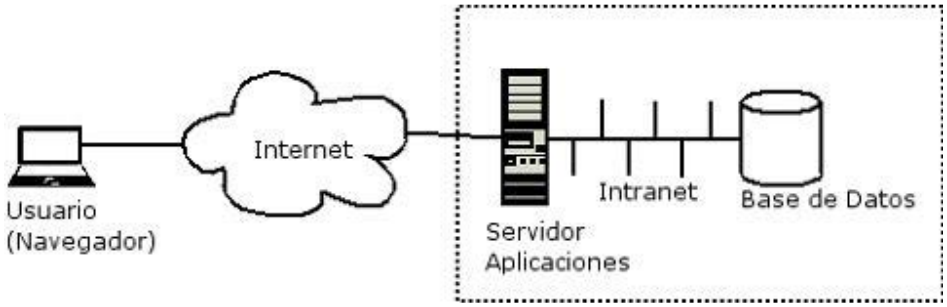


Figura 1. Arquitectura de Tres Capas a través de Internet

El usuario (capa cliente) accede a una interfaz conformada por páginas Web, que es visualizada a través de un navegador Web, los clientes se comunican con el servidor (capa lógica) sobre el cual funciona la aplicación Web, haciendo uso del protocolo HTTP (protocolo de transferencia de hipertexto).
En el servidor de aplicaciones se ejecuta el sistema responsable de:

- generar las interfaces de usuario (páginas Web) que son enviadas a los clientes
- ejecutar la lógica del negocio
- comunicarse con la base de datos (capa de persistencia) para almacenar o recuperar información.

En la capa de persistencia normalmente se encuentra en funcionamiento un servidor de bases de datos responsable de almacenar toda la información perteneciente al sistema.
Las aplicaciones Web pueden funcionar en redes locales (como por ejemplo Intranets corporativas o sistemas internos de gestión) y ser accesibles sólo desde la red de la

organización, o pueden funcionar en Internet (como por ejemplo el email o las redes sociales) permitiendo que se conecten clientes de todo el mundo a determinados servidores para interactuar con dichas aplicaciones, eliminando de esta forma cualquier tipo de barrera geográfica.

Características de una aplicación Web

Una aplicación Web presenta muchas características, pero hay cuatro que se pueden considerar como las más importantes:

- **Capacidad de gestionar información:** como toda aplicación, las aplicaciones Web permiten consultar, agregar, modificar y eliminar información.
- **Multiplataforma:** Dado que las interfaces de usuario son creadas mediante páginas Web, no hay restricciones para los clientes que quieran interactuar con el sistema, sólo necesitan utilizar un navegador Web como Internet Explorer, Firefox, Chrome, etc. funcionando sobre cualquier sistema operativo.
- **Disponibilidad permanente:** Dado que las aplicaciones se encuentran funcionando en servidores locales o en Internet, están disponibles para ser utilizadas las 24 horas, sin restricciones de tiempo.
- **Privacidad de la información**: Existe la opción para que los usuarios provean una contraseña de acceso para poder interactuar con el sistema, de esta forma se garantiza su identidad y sólo podrán acceder a la información para la que están autorizados.

2. JAVA 2

Seam está hecho en lenguaje Java y genera aplicaciones también escritas en Java. Java es una plataforma de desarrollo multiplataforma orientado a objetos, que ha recibido una gran aceptación y soporte por parte de empresas y fundaciones sobresalientes en la industria del software como IBM, Oracle, Apple, Apache, etc.
Existen diferentes ediciones de Java, cada una permite abordar soluciones de diferente índole:

- Java 2 Standard Edition (J2SE), plataforma estándar, es la edición más básica de Java, sobre la cual están basadas las otras ediciones
- Java 2 Enteprise Edition (J2EE), plataforma de desarrollo de aplicaciones empresariales (incluyendo las aplicaciones Web), se explicará más acerca de esta edición posteriormente.
- Java 2 Micro Edition (J2ME), plataforma de desarrollo para dispositivos móviles.

2.1. Entornos de desarrollo de Aplicaciones Empresariales

Los entornos empresariales están pensados para dar prestaciones a sistemas con un amplio rango de funcionalidades y de alta concurrencia. Presentan las siguientes características:

- Heterogeneidad: permiten múltiples plataformas, sistemas operativos y lenguajes de programación.
- Fiabilidad: tienen que dar la confianza de que cumplirá su cometido.
- Seguridad: debe controlar el acceso, autorización y transporte de la información.
- Robustez: debido a la criticidad del sistema, es indispensable que sea tolerante a fallos y que estos no ocasionen caídas del sistema.
- Escalabilidad: tiene que facilitar la ampliación y modificación necesarias.
- Alta disponibilidad: debe asegurar un cierto grado de continuidad en su funcionamiento a lo largo del tiempo.
- Fácil mantenimiento: debe ser fácil mantener el sistema mediante la actualización de sus componentes.

Entre las alternativas más difundidas para entornos empresariales en la actualidad se encuentran:

- .NET (dot NET, punto NET), desarrollado y mantenido por Microsoft.
- Java 2 Enterprise Edition (J2EE), desarrollado por Sun Microsystems, pero mantenido actualmente por Oracle debido a la adquisición por parte de éste, de Sun Microsystems. (Dpto. Ing. Electrónica, Sist. Informáticos y Automática, Universidad de Huelva, 2006)

2.2. Java 2 Enterprise Edition (J2EE)

J2EE es un entorno para desarrollo, construcción y despliegue en línea de aplicaciones empresariales, es independiente de la plataforma y está centrado en Java, incluye muchos componentes del J2SE y consiste en un conjunto de servicios, APIs, y protocolos que proveen la funcionalidad para desarrollar aplicaciones Web de múltiples capas.

J2EE simplifica el desarrollo de aplicaciones y disminuye las necesidades de programar y formar programadores al crear componentes modulares estandarizados, reusables y al permitir que se gestionen muchos aspectos de la programación automáticamente.

Para el desarrollo empresarial se necesita J2EE porque escribir aplicaciones de negocio distribuidas no es fácil, y se necesita una solución confiable de alta productividad, que permita concentrarse en escribir la lógica de negocio y tener un amplio rango de servicios basados en clases empresariales. (Sun Microsystems, Inc.)

2.3.Tecnologías

Estas son algunas de las tecnologías que están disponibles y se pueden implementar dentro de cualquier aplicación J2EE:

- *Enterprise Java Bean* (EJB), componentes que agrupan funcionalidades para aplicaciones empresariales
- *Java Server Page Standard Tag Library* (JSTL), permite la inclusión de bibliotecas con etiquetas dinámicas para la generación de contenido Web

- *Java Message Service* (JMS), para la comunicación entre componentes y/o aplicaciones de software
- *Java Transaction API* (JTA), especifica interfaces entre un gestor de transacciones, un servidor de aplicaciones y una aplicación transaccional
- *Java Mail API*, para el envío de correos
- *Java Beans Activation Framework* (JAF), para la identificación de datos binarios
- Procesamiento de XML : JAXP, JAX-RPC, JAXB, SAAJ, JAXR
- Procesamiento de XML de Web Services: JAX-WS
- *Java Database Connectivity* (JDBC) API, para la conexión a bases de datos
- *Java Naming and Directory Interface* (JNDI), para servicios de nombrado y directorio, que sirven para buscar objetos a través de nombres identificadores
- *Java Authentication and Authorization Service* (JAAS), para la gestión de autenticación y autorización sobre cualquier aplicación en Java
- Java Servlets
- *Java Server Pages* (JSP)
- *Java Server Faces* (JSF)

Sobre estas tres últimas se explica más a continuación. Se puede ver en la Figura 2 cómo están distribuidas algunas de estas tecnologías. (Dpto. Ing. Electrónica, Sist. Informáticos y Automática, Universidad de Huelva, 2006)

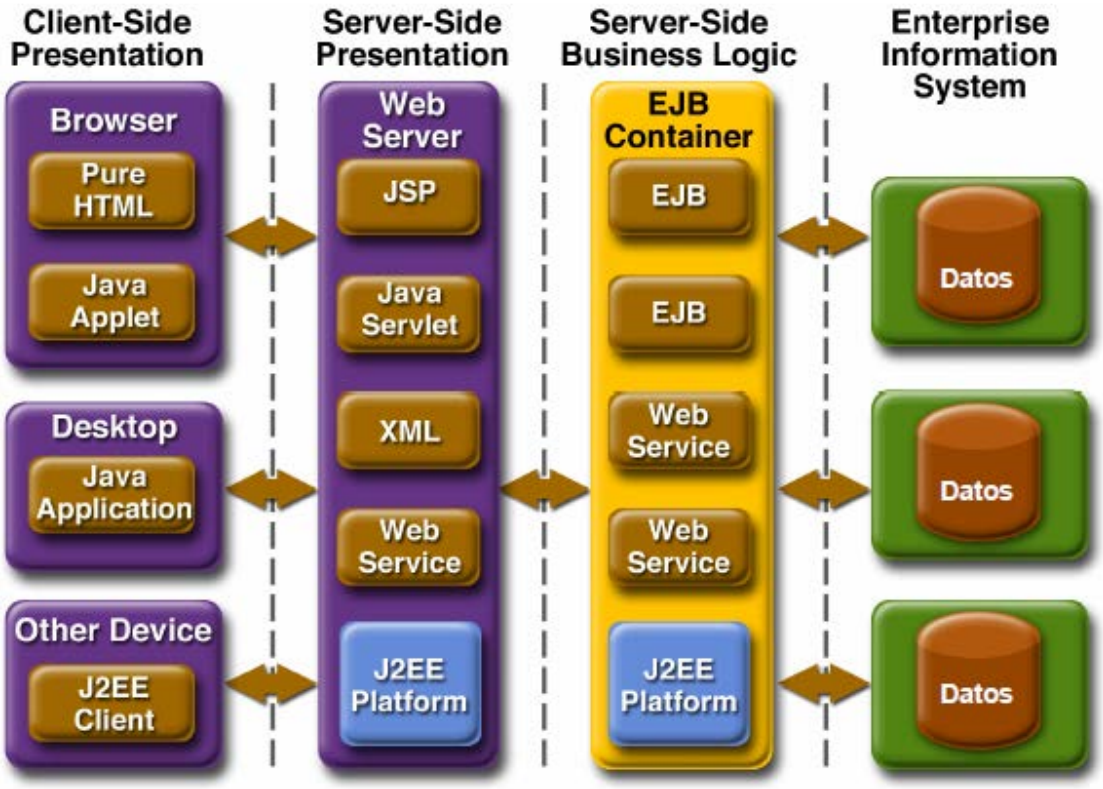


Figura 2. Modelo de la distribución de tecnologías Java EE

2.4.Aplicaciones Web y las Tecnologías Java

J2EE maneja distintas tecnologías dependiendo del contexto en el que va a ser usado, entre ellas tenemos las tecnologías Web, que permiten la ejecución de aplicaciones Web sobre servidores de aplicaciones. Las tecnologías Web más usadas son Java Servlets, *Java Server Pages* (JSP) y *Java Server Faces* (JSF).

- Java Servlet provee mecanismos simples para manejar las peticiones de usuario y las respuestas que se le enviarán, se podría considerar que un Servlet es un *applet*[2] que corre en la capa servidor sin tener una interfaz gráfica definida. Están pensados para las funciones de control de las aplicaciones Web, como el procesamiento de peticiones y la interacción con la capa de persistencia.
- JSP permite crear contenido Web que tiene componentes tanto estáticos como dinámicos, dispone de todas las capacidades dinámicas de Java Servlet pero provee un enfoque más natural para crear contenido estático. Genera *markup*[3] basado en texto como por ejemplo: HTML, SVG, WML y XML.

Desde la introducción de las tecnologías Java Servlet y JSP, se han desarrollado tecnologías adicionales y *framework*s para construir aplicaciones Web interactivas. Estas tecnologías y sus relaciones están ilustradas en la Figura 3.

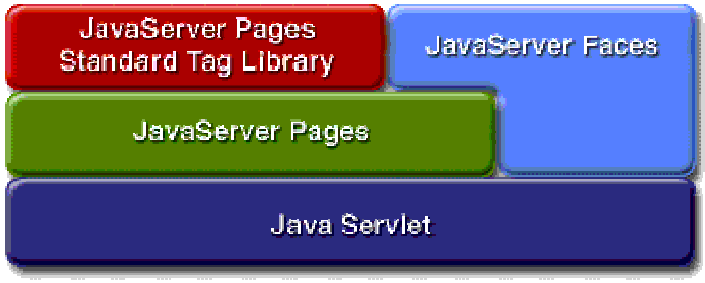


Figura 3. Tecnologías de Aplicaciones Web en Java

- JSF establece un estándar para la construcción de interfaces de usuario que corren en la capa servidor, es un lenguaje muy similar al JSP sólo que incorpora la capacidad de incluir bibliotecas (gracias a la tecnología JSTL) para un manejo mucho más simple de los contenidos dinámicos.

Tal como se aprecia en la Figura 3, la tecnología Java Servlet es la base de todas las tecnologías de aplicaciones Web. Cada tecnología agrega un nivel de abstracción que hace que el desarrollo de aplicaciones Web sea más rápido y que las aplicaciones desarrolladas sean más fáciles de mantener, más escalables y robustas. (Oracle)

[2] Un applet es un componente de una aplicación que se ejecuta en el contexto de otro programa, por ejemplo en un navegador web. Los applets de Java solicitan una autorización al usuario antes de ejecutarse en el navegador.
[3] Un lenguaje markup es un sistema para anotar texto mediante sintaxis, ejemplos de esto son XML y HTML.

3. Componentes y Reutilización

Seam incluye componentes propios y externos, y los reutiliza a lo largo de las aplicaciones que genera. En la Figura 4 tenemos dos representaciones distintas de un componente, la primera en UML[4] y la segunda en objetos COM[5] de Microsoft. Los conectores salientes de los componentes son sus interfaces.

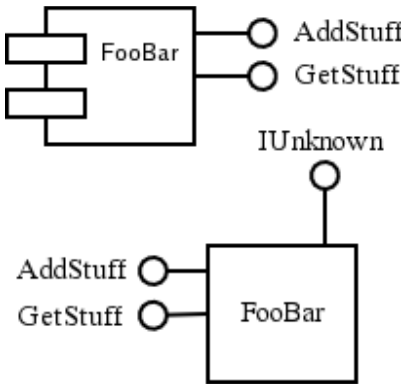


Figura 4. Representaciones de un componente

Los componentes de software son un campo de estudio de la ingeniería de software. Por definición estos deberían ser, al igual que los componentes de hardware, hechos para ser intercambiables y confiables.
Un componente es "una parte casi totalmente independiente y reemplazable de un sistema que cumple una función específica en el contexto de una arquitectura bien definida" (Brown & Wallnau, 1996). Encapsula funcionalidades y tiene las siguientes características:

- Múltiples usos
- No específico para un contexto
- Integrable con otros componentes
- Encapsulado, no se sabe cómo funciona al interior de sus interfaces
- Una única unidad de despliegue y de versiones independientes (Szyperski, 2008)

Cada componente es diseñado para encajar en un estilo específico de arquitectura, por lo tanto sus interacciones con los sistemas están estandarizadas, un protocolo de comunicación ha sido preestablecido. (wordIQ.com)

3.1. Ingeniería de Software basada en Componentes (CBSE, por sus siglas en inglés)

El propósito de la CBSE es incrementar la productividad en la creación de software, pues permite construir sistemas a partir de componentes estandarizados

[4] UML: *Unified Modeling Language* o Lenguaje Unificado de Modelo es un lenguaje gráfico utilizado para modelar sistemas de software.
[5] COM: Component Object Model o Modelo de Objecto Componente, es un estándar para composición de software creado por Microsoft.

en vez de “reinventar la rueda” una y otra vez. La CBSE propone la composición de sistemas de software en vez de programarlos.
La CBSE permite que los sistemas de software sean más fácilmente ensamblados, y menos costosos de construir. Si desarrollamos sistemas de esta forma, no sólo son más simples y baratos sino que además tienden a ser más robustos, adaptables y actualizables.
Un ingeniero de software que va a trabajar con un componente solo recibirá una interface externa bien definida, con la cual tendrá que trabajar. Por lo tanto de los componentes solamente se conoce la interface y la funcionalidad que tiene más no la forma en que logra esa funcionalidad. (Siddiqui, 2000)

3.2. Reutilización de Software

El propósito de la reutilización es que las partes comunes (ya sean clases o funciones) en una aplicación de software sólo necesitan ser escritas una vez, y reutilizadas en vez de ser reescritas cada vez que una nueva aplicación es desarrollada.

3.2.1. Requerimientos

- Ocultar el código, para evitar que la reutilización lleve a la reescritura de código.
- Lo anterior provee abstracción, un proceso que reduce la información oculta y da una idea sobre los requisitos de un componente para llevar a cabo una tarea.
- Interfaces, que deberán ser simples y trabajar con datos lo más simples posibles.
- Documentación, que permita obviar la necesidad de examinar el código fuente.
- Calidad, porque los componentes reusables deben tener la mejor calidad si van a ser usados por terceros. Esto provee la confiablidad necesaria para alentar más reutilización de software y continuar con las mejoras.

3.2.2. Dificultades asociadas con la Reutilización de Software

- Funciones: no todas las funciones incluidas en un componente concuerdan totalmente con las funciones requeridas por la aplicación que las usará.
- Programación: hay un problema grave cuando la aplicación está escrita en un lenguaje distinto que el componente.
- Entorno: un componente que fue desarrollado en un entorno determinado (de hardware y software) podría no ser transferible a otro entorno.
- Estándares: un componente configurado para conectarse usando algún estándar de comunicaciones podría no tener la capacidad para usar otros estándares.
- Formatos: un claro ejemplo está en las bases de datos, que usan diferentes formatos para almacenar data de tipo fecha, hora, moneda, etc.

- Estructura: diferencias estructurales entre la aplicación y los componentes podrían tener consecuencias sobre los requerimientos como la búsqueda, replicación, ordenación de datos, etc. Esto ocurre debido a la variedad de algoritmos usados para implementar tareas clave. (Booth, 2006)

4. El patrón de diseño Modelo Vista Controlador (MVC)

La flexibilidad de los sistemas basados en componentes grandes ha hecho surgir preguntas sobre cómo organizar un proyecto para desarrollo y mantenimiento fáciles, protegiendo a la vez datos y confiabilidad. La respuesta está en usar el patrón de diseño MVC. Este patrón de diseño describe un problema frecuente y su solución, donde la solución no siempre es exactamente la misma para cada ocurrencia.
Para usar el MVC con efectividad, es necesario entender la división de las tareas entre Modelo, Vista y Controlador, y cómo interactúan entre sí.
Las aplicaciones generadas por *Seam* siguen la lógica del MVC.

4.1.¿Por qué MVC?

Cuando se desarrolla una aplicación para soportar un solo tipo de cliente, conviene entrelazar a veces el acceso a datos y las reglas de la lógica de negocio con lógica propia de la interfaz para la presentación y control. Pero ese enfoque, sin embargo, es inadecuado cuando se aplica a sistemas empresariales que necesitan soportar múltiples tipos de clientes. Se necesita desarrollar distintas aplicaciones, para soportar cada tipo de interfaz de cliente. El código no específico de la interfaz es duplicado en cada aplicación, resultando en esfuerzos duplicados de implementación (usualmente de la variedad de los copia-y-pega fragmentos de código), así como de pruebas y mantenimiento. La tarea de determinar qué duplicar es pesada ya de por sí, ya que están entrelazados los códigos específicos y los no específicos de la interfaz. Los esfuerzos duplicados son inevitablemente imperfectos. Lenta pero definitivamente, las aplicaciones, que se supone proveerían la misma funcionalidad en el núcleo, evolucionan en sistemas diferentes.

En la Figura 5 tenemos un claro ejemplo, en el que un cliente Web clásico accede por una vista HTML, un usuario con dispositivo móvil accede a una vista WML (wireless markup language), un administrador accede desde un cliente de escritorio JSF/Swing y un proveedor accede a través de un Web service. A pesar de los clientes distintos, el sistema de información empresarial es único.

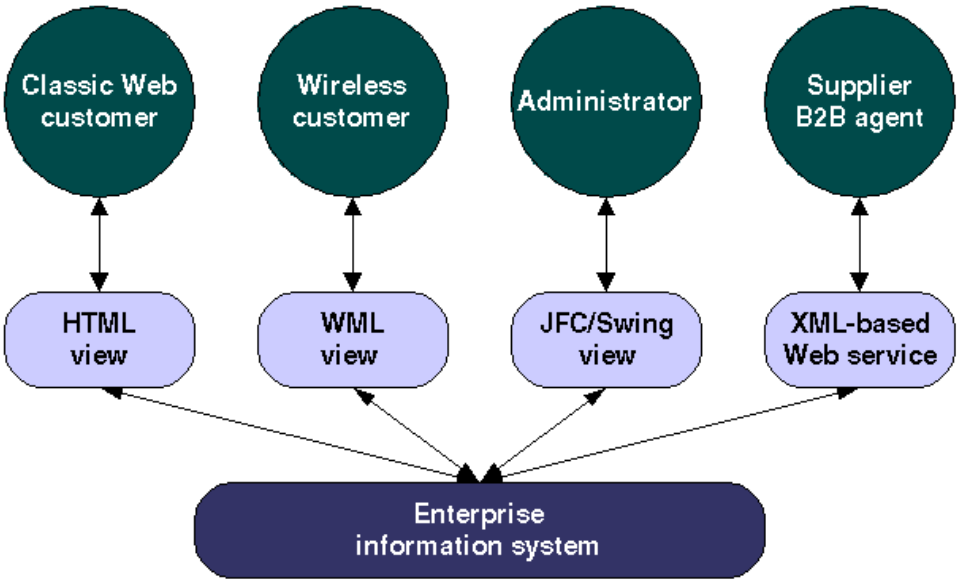


Figura 5. Sistema empresarial con múltiples tipos de clientes

Aplicando la arquitectura MVC al sistema empresarial, se separa la funcionalidad del núcleo del modelo de negocio de la presentación y la lógica de control que usa esta funcionalidad. Dicha separación permite que múltiples vistas puedan compartir la misma data empresarial, lo que hace que el soporte de múltiples clientes sea más fácil de implementar, probar y mantener.

El soporte para múltiples vistas permite generar además, bajo una misma estructura de datos, diferentes presentaciones del mismo documento. Por ejemplo, un conjunto de tablas estadísticas, podría presentarse en forma de página Web, pero también como Hoja de Excel o Documento PDF.

4.1.1. Participantes y responsabilidades

La arquitectura MVC tiene sus raíces en Smalltalk[6], donde fue aplicada originalmente para mapear la entrada, proceso y salida de tareas tradicionales al modelo de interacción gráfica del usuario. Sin embargo, es sencillo mapear estos conceptos al dominio de aplicaciones empresariales multi-capa. El patrón MVC se puede apreciar en la Figura 6 y se explica a continuación.

[6] Lenguaje de programación en el cual los objetos pueden comunicarse entre sí mediante el envío de mensajes.

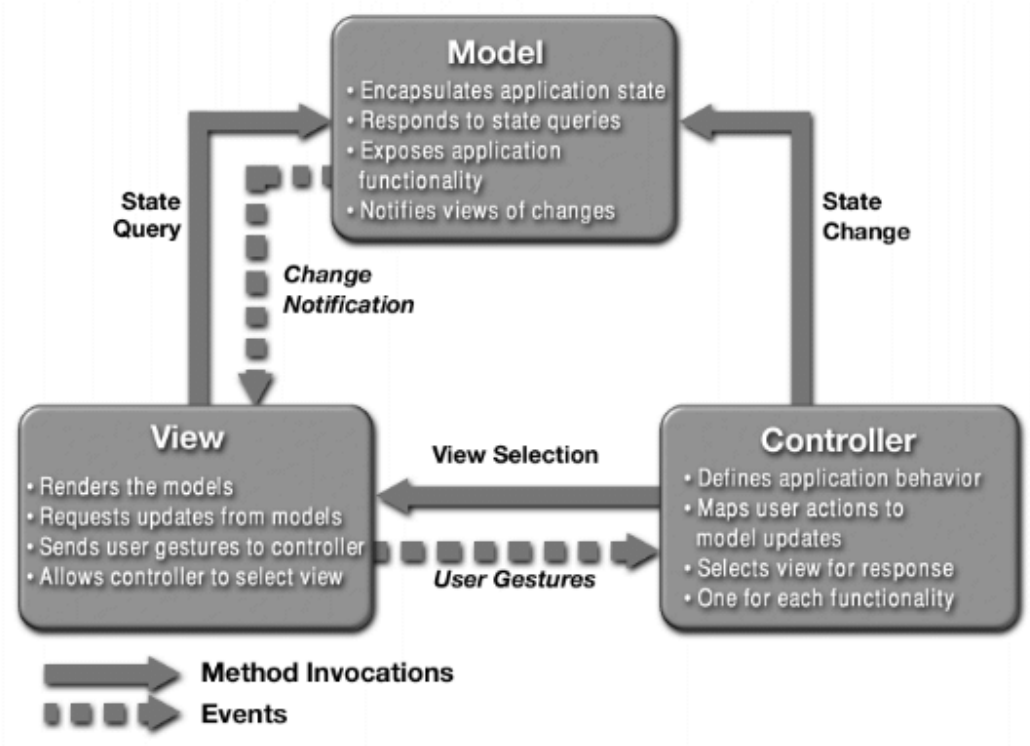


Figura 6. Patrón MVC

- **Modelo** – Representa los datos empresariales y las reglas de negocio que gobiernan el acceso y las actualizaciones de los datos. A menudo el modelo sirve como una aproximación de software al proceso en el mundo real, así que las técnicas para modelar el mundo real se aplican al definir el modelo. El modelo es el responsable de la persistencia de los datos.
- **Vista** – Muestra los contenidos del modelo. Accede a los datos empresariales a través del modelo y especifica cómo se deberán presentar esos datos. Es responsabilidad de la vista mantener consistencia en su presentación cuando el modelo cambia. La vista es la parte del sistema con la que interactúa el usuario.
- **Controlador** – Traduce interacciones con la vista en acciones que serán realizadas por el modelo. En un cliente individual de tipo GUI, las interacciones del usuario podrían ser clics en botones o selecciones de menú, mientras que en una aplicación Web, aparecen como peticiones HTTP GET y POST (uso de links y formularios). Basándose en las interacciones del usuario y el resultado de las acciones del modelo, el controlador responde seleccionando la vista apropiada.

4.1.2. Consecuencias del uso de MVC

- Reutilización de componentes del Modelo. La separación de modelo y vista permite que múltiples vistas usen el mismo modelo

empresarial. Del mismo modo, los componentes del modelo de una aplicación empresarial son más fáciles de implementar, probar y mantener, ya que todo acceso al modelo va a través de estos componentes.

- Soporte más fácil para nuevos tipos de clientes. Para soportar un nuevo tipo de cliente, simplemente hay que escribir las vistas y un poco de lógica de controlador y enlazarlo a la aplicación empresarial (modelo - controlador) ya existente.
- Incremento en la complejidad del diseño. El uso de MVC hace más complejo el diseño debido a la separación modelo, vista y controlador. (Sun Microsystems, Inc., 2002)

5. Framework

Un *framework*, es un conjunto de componentes de software que los programadores pueden usar, extender, o personalizar para una determinada aplicación. Como metodología, es un mecanismo de reutilización orientado a objetos que permite al desarrollador descomponer una aplicación en un conjunto de objetos que interactúan entre sí. Describe las interfaces implementadas por los componentes del *framework*, el flujo de control entre los componentes, y la interacción entre los componentes y el sistema. De esta forma un *framework* es un diseño reusable. Las interfaces e interacciones estandarizadas hacen posible mezclar componentes ya existentes y crear una amplia variedad de sistemas a partir de un conjunto base de componentes.
Con los *framework*s, los desarrolladores no tienen que empezar desde cero cada vez que construyen una aplicación. La flexibilidad inherente permite la creación rápida y el desarrollo de soluciones de software en un entorno de negocio que evoluciona constantemente.

Un *framework* ofrece las siguientes posibilidades:

- Facilita el trabajo de los desarrolladores cuando es necesario usar tecnologías complejas
- Permite la agrupación de muchos objetos/componentes discretos en algo más útil
- Fuerza al equipo de desarrollo a implementar código de una forma que promueve la programación consistente, menos errores, y aplicaciones más flexibles, generalmente siguiendo patrones de diseño (como el MVC).
- Facilita la ejecución de pruebas y la depuración de código (Clifton, 2003)

6. Web 2.0

“Lejos de estrellarse, la Web es más importante que nunca, con apasionantes nuevas aplicaciones y con nuevos sitios Web apareciendo con sorprendente regularidad” (O'Reilly, 2005).
Entre 2001 y 2003, el término 'Web 2.0' arraigó claramente, con más de 9.5 millones de menciones en Google. Pero todavía existe un enorme desacuerdo sobre qué significa Web 2.0, existiendo algunas críticas que afirman que se trata simplemente de

una palabra de moda, fruto del marketing, y sin sentido, en tanto que otros la aceptan como un nuevo paradigma.
Seam considera Web 2.0 como un nuevo paradigma, por el que las aplicaciones deben ser más amigables con los usuarios gracias a componentes visuales y herramientas que faciliten la interacción.

6.1.Aprovechando el aporte de los usuarios

Web 2.0 hace referencia a la evolución de la Web a lo que es actualmente, los contenidos presentados ya no están a cargo de un solo equipo reducido de responsables sino que ahora los usuarios son quienes aportan estos contenidos. Las páginas Web ahora necesitan de los usuarios para la generación de contenidos, y para poder lograr el aporte de ellos, es necesario que las páginas estén diseñadas con alta interacción con sus usuarios, y con interfaces cada vez más amigables, "ricas", que los inviten a enviar contenidos constantemente.
A continuación se presenta algunos portales Web de éxito que operan bajo el paradigma Web 2.0.

- Wikipedia, quizás uno de los pioneros en el tema, una enciclopedia en línea basada en la antes inverosímil idea de que una entrada puede ser agregada por cualquier usuario de la Web, y corregida por cualquier otro, es un experimento radical de confianza, aplicando la máxima de que "con ojos suficientes, todos los fallos son superficiales" (Raymond, 1997) para la generación de contenido. Wikipedia está ya entre las 100 Webs más visitadas, y muchos piensan que llegará a estar entre las 10 de la cima en poco tiempo. Esto sí que es un cambio profundo en la dinámica de la creación de contenidos.
- Del.icio.us y Flickr, dos compañías que han recibido mucha atención últimamente, han promovido un concepto que alguna gente llama 'folksonomy' (en contraste con la taxonomía), un estilo de clasificación colaborativa de sitios usando palabras clave libremente elegidas, a menudo denominadas etiquetas (tags). El marcado con etiquetas permite la clase de asociaciones múltiples, y solapadas que el propio cerebro humano utiliza, en lugar de categorías rígidas. Por ejemplo, una foto de Flickr de un cachorro puede ser marcada con las etiquetas tanto 'cachorro' como 'lindo' - permitiendo encontrar la foto siguiendo los mismos ejes naturales generados por la actividad del usuario.
- Youtube, un portal que permite el alojamiento de videos enviados por los usuarios, cuenta con sistemas para calificación y comentarios sobre cada video, y suscripciones sobre las publicaciones de otros usuarios. Youtube permite, al igual que Flickr, el uso de etiquetas para clasificar sus contenidos.
- Facebook, que permite agregar a otros usuarios como amigos e interactuar con ellos a través de la publicación de contenido multimedia como fotos y videos, enviar comentarios y mensajes privados

- Blogspot y Wordpress, son sistemas gestores de blogs, bitácoras personales donde un usuario puede publicar información sobre temas de su interés y sus lectores pueden enviar comentarios. Wordpress ha tenido tanto éxito que inclusive ha liberado su código fuente para que sus usuarios puedan descargar e instalar su blog en cualquier servidor Web.

De los casos de éxito mencionados, podemos afirmar que las contribuciones del usuario son la clave para el dominio del mercado en la era de la Web 2.0.

6.2.AJAX

Ajax es la abreviatura de "Asynchronous JavaScript + XML" (JavaScript asíncrono + XML), no es una sola tecnología, sino un conjunto que reúne:

- Presentación basada en estándares (XHTML y CSS[7])
- Dibujado dinámico e interacción con las páginas Web (Document Object Model o Modelo Objeto Documento)
- Intercambio y manipulación de datos (XML y XSLT)
- Recuperación asíncrona de datos (XMLHttpRequest)
- Javascript para unir todos los anteriores

Como se puede apreciar en la Figura 7, bajo el enfoque clásico, el servidor hace el trabajo mientras hace esperar al usuario en cada tarea que lleva a cabo. Ajax es diferente porque elimina las interrupciones al introducir un intermediario, conocido como motor Ajax, entre el usuario y el servidor.

[7] CSS: Cascading Style Sheets (Hojas de estilo en cascada), es un lenguaje usado para describir semánticas de presentación de un documento escrito en lenguaje markup como HTML

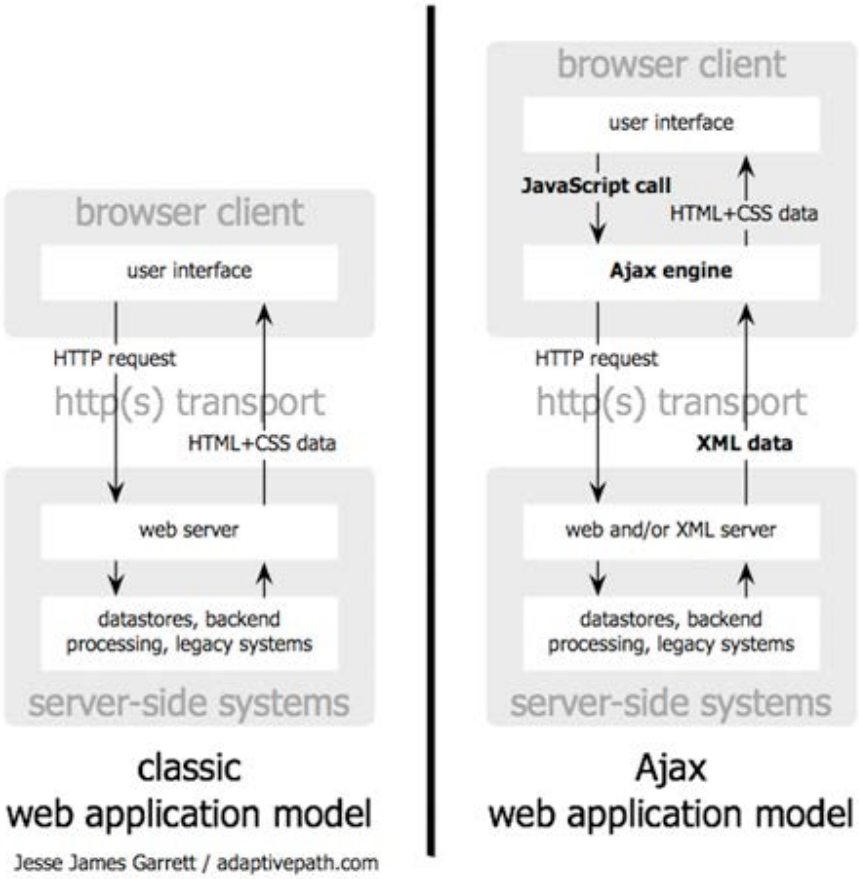


Figura 7. Modelo Clásico vs Modelo Ajax

En lugar de cargar una página Web, al inicio de la sesión, el navegador carga un motor Ajax, escrito en Javascript, el cual es responsable tanto de dibujar la interface que el usuario ve como de comunicarse con el servidor en cada petición del usuario. El motor por lo tanto funciona en el cliente (a través del navegador) y permite la interacción del usuario con la aplicación ocurra asíncronamente, es decir, independientemente de la comunicación directa con el servidor. De esta forma el usuario nunca se queda esperando a que cargue una página Web para poder continuar con sus interacciones. (Garrett, 2005)

Capítulo II
Seam

1. *JBoss Seam*

JBoss Seam es un *framework* de desarrollo para la construcción de aplicaciones de nueva generación Web 2.0, unificando e integrando tecnologías tales como Ajax, JSF, EJB3, Java Portlets y Gestión del Proceso de Negocio (BPM).
Seam ha sido diseñado desde cero para eliminar la complejidad en los niveles de arquitectura y API. Le permite a los desarrolladores ensamblar aplicaciones Web complejas usando *Plain Old Java Objects* (POJOs), clases Java compuestas simplemente por atributos privados y métodos públicos GET y SET; y mediante el uso de anotaciones, widgets de componentes UI y muy poco XML. Tiene soporte para un nuevo contexto de variables conocido como Conversación y la posibilidad de gestionar estados (states) en la aplicación declarándolos en la programación, sobre esto se profundizará en el apartado siguiente. *Seam* presenta una experiencia de usuario más sofisticada, y al mismo tiempo elimina errores comunes encontrados en aplicaciones Web tradicionales.

1.1.Características y Capacidades de *Seam*

- La tecnología EJB, que se caracterizaba por estar conformada por componentes pesados, ha mejorado en su nueva versión 3.0, los componentes que antes eran de poca granularidad han sido convertidos en POJOs livianos de granularidad más fina. En *Seam*, cualquier clase Java puede ser un EJB gracias a las anotaciones. *Seam* elimina la distinción entre los componentes de la capa de presentación y los de la lógica de negocio y trae un modelo de componentes uniformes a la plataforma J2EE.
- Compatibles con versiones anteriores de J2EE: *Seam* no está limitado a entornos que sólo soportan EJB 3.0 y puede ser usado en cualquier entorno J2EE, inclusive en un servidor no empresarial como Apache Tomcat.
- Uso fácil de Ajax: *Seam* integra soluciones Ajax, basadas en JSF como ICEfaces y Ajax4JSF, con el motor de gestión de estados y concurrencia de *Seam*. Se puede agregar Ajax a las aplicaciones con facilidad, sin necesidad de aprender JavaScript, y al mismo tiempo estar protegido de potenciales errores y problemas de desempeño asociados a la implementación de Ajax.

- Un nuevo enfoque sobre la gestión de estados: antes de *Seam*, la sesión HTTP era la única forma de gestionar el estado de la aplicación Web. *Seam* provee múltiples contextos con estados de diferentes granularidades desde el alcance de la Conversación hasta el del Proceso de negocio, liberando a los desarrolladores de las limitaciones de las sesiones HTTP. Por ejemplo, los desarrolladores pueden escribir aplicaciones Web con múltiples espacios de trabajo que se comportan como un cliente de escritorio de varias ventanas.
- Gestión del "flujo": *Seam* integra una gestión transparente del proceso de negocio vía *JBoss* jBPM, simplificando el modelado, la implementación y la optimización de procesos complejos (flujo de trabajo) y las interacciones de usuario (flujo de página).
- Fácil integración de pruebas: los componentes *Seam* son también unidades de pruebas, esto es, unidades de código que se pueden poner a prueba para determinar si están listas para ser implementadas en producción. Para aplicaciones complejas, las unidades de prueba no son suficiente, por lo que *Seam* provee además soporte para ejecución de pruebas. Se puede escribir pruebas en JUnit o TestNG (motores de ejecución de pruebas), que reproduzcan toda la interacción con el usuario, poniendo a prueba así todos los componentes del sistema, y luego ejecutarlas en el software de desarrollo.
- Debido a todas las tecnologías que *Seam* integra y abstrae, el equipo de desarrollo puede concentrarse en la lógica del negocio, mejorando así su productividad.
- Como se puede apreciar en la Figura 8 en la capa de presentación encontramos las tecnologías JSP, Facelets, Portal, en la capa controlador la tecnología JSF, en la gestión de estados se encuentran las tecnologías EJB, jBPM y Hibernate. Seam está situado en la gestión de contextos, un nuevo nivel creado por el mismo *framework*. (JBoss)

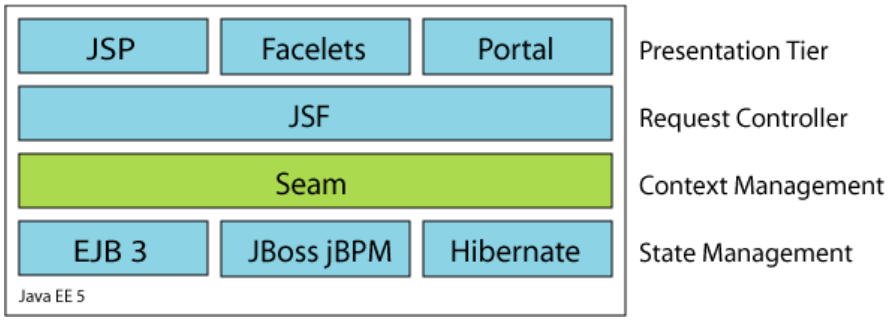


Figura 8. Arquitectura Seam

2. Arquitectura y Componentes en *Seam*

2.1. Un *Framework* de Integración

Los *framework*s de software son herramientas para los desarrolladores. Son usados ampliamente para empaquetar componentes reusables de software y servicios.

Cada *framework* provee un conjunto de patrones de diseño, APIs, y modelos de componentes, que pueden ser usados en el desarrollo de sus aplicaciones. Ejemplos de los populares *framework*s de J2EE incluyen tanto herramientas de código abierto tales como Hibernate, Spring, Struts, etc. como productos basados en estándares tales como muchas implementaciones para Servlet/JSP, JSF, EJB, JMS, Web Services, etc. Una aplicación J2EE típica puede usar más de una docena de *framework*s al mismo tiempo. Por lo tanto, una competencia clave para los desarrolladores es la habilidad de usar esos *framework*s.
Sin embargo, el problema con muchos *framework*s es que cada uno provee un modelo de programación (o modelo de componentes) diferente. Para hacerlos trabajar juntos en una sola aplicación, el desarrollador usualmente necesita escribir mucho código con el propósito de unirlos (por ejemplo, objetos para transferir datos, gestión de componentes, etc.) y gran cantidad de archivos de "unión" de configuración, sólo para mantener a los *framework*s integrados y funcionando. Está claro que esto es contraproducente, y es lo que ha dado salida a un tipo especial de *framework*s conocidos como "*framework*s de integración", los cuales están diseñados para reducir el código de unión y hacer que varios *framework*s trabajen juntos bajo un modelo de programación consistente.

J2EE puede ser considerado por sí mismo como un *framework* de integración. Especifica cómo varios *framework*s, tales como Servlet/JSP, JSF, EJB, JMS, JTA, JCA, etc. trabajan juntos en una aplicación Enterprise. Sin embargo, como especificación estándar, J2EE evoluciona muy lentamente, es muy lento solucionar problemas de diseño, y nuevas ideas y tecnologías de punta a menudo emergen como proyectos de código abierto antes de convertirse en parte del estándar. Otro *framework* de integración ampliamente usado es Spring. Spring provee una envoltura fina alrededor de otros *framework*s y permite a los desarrolladores de aplicaciones usar archivos de configuración XML para gestionar los componentes a lo largo de toda la aplicación. Sin embargo, con Spring, el desarrollador aun necesita mezclar y emparejar diferentes modelos de programación de componentes de los *framework*s integrados. Adicionalmente, el enfoque pesado XML en Spring produce aplicaciones con grandes cantidades de código XML.
JBoss Seam es un *framework* de alta integración y código abierto que trata de tener lo mejor de los mundos de J2EE y Spring. *JBoss Seam* está firmemente atado a los estándares de J2EE. Todo empezó con el propósito de hacerle frente a las fallas de diseño entre dos *framework*s clave de J2EE: JSF y EJB. Algunas de sus características principales están siendo adoptadas como estándares futuros oficiales de J2EE, tales como JSF 2.0 y WebBeans. Mientras más y más usuarios empiezan a adoptar *Seam,* éste se ha expandido mucho más allá del alcance de J2EE.
Seam integra de una manera muy diferente respecto a *framework*s como Spring, provee un modelo unificado de componentes para todos los *framework*s que integra. Los desarrolladores trabajarán con componentes *Seam,* en lugar de usar el API de gestión de componentes de cada *framework* individual; el API de *Seam* representa una mejora significativa debido a todo lo que integra. El API de anotaciones de *Seam* es muy similar a J2EE v5.
Los *framework*s que *Seam* integra quedan entonces por debajo de la aplicación en forma transparente para el usuario. En algunos casos, es posible intercambiar entre

los *framework*s involucrados sin cambiar la aplicación misma, de la misma forma en que J2EE permite múltiples implementaciones de un mismo API.

2.2.El Único Modelo de Programación que Gobierna a Todos

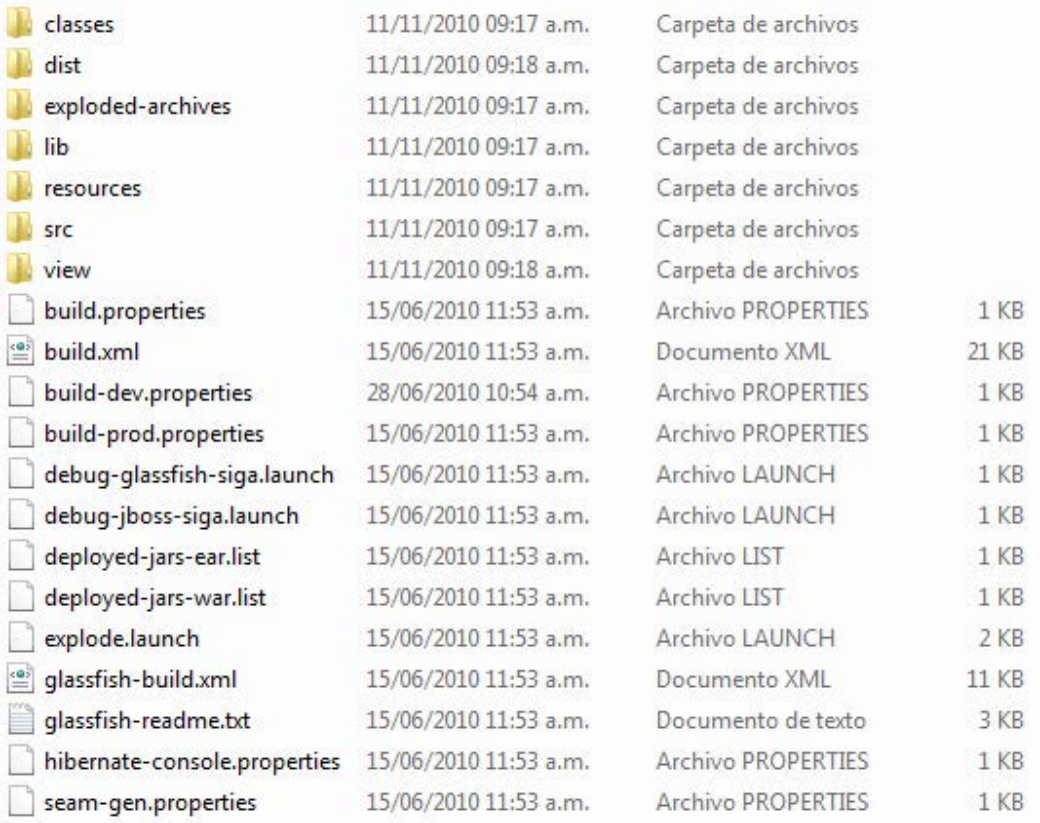
Seam integra varios *framework*s bajo un modelo de programación consistente. Hay muchos otros *framework*s que *Seam* integra además de los mencionados. La clave del modelo de programación de *Seam* está en tres artefactos:

- POJOs anotados: Todos los componentes Java en una aplicación *Seam* están en clases POJO anotadas. Sus interacciones son manejadas por *Seam*. No hay otros modelos de componentes en *Seam*.
- Vistas de página XHTML: todas las páginas de vistas (o UI) son escritas en archivos XHTML. Adicionalmente a las etiquetas JSF estándar, *Seam* define muchas de sus propias etiquetas UI incluyendo las etiquetas PDF UI para la generación de documentos. *Seam* también incorpora bibliotecas de componentes Ajax JSF tales como Ajax4jsf, RichFaces, y IceFaces.
- Lenguaje de expresiones: El Lenguaje de Expresiones JSF (EL) es usado para referenciar componentes Java de *Seam* desde páginas XHTML. *Seam* ha mejorado el estándar de la sintaxis EL para soportar parámetros en los métodos, etc. También ha expandido el uso de EL a todos los archivos de configuración XML y scripts de pruebas.

Con todas las características de *Seam*, el modelo de programación es bastante simple. Tiene una curva de aprendizaje aceptable luego de haber entendido las bases del JSF, y se puede aprender la integración parte por parte. (Yuan, 2007)

3. Esquema de un proyecto hecho en *Seam*

La descarga de *Seam* incluye una herramienta de generación de proyectos llamada seam-gen. Esta herramienta permite configurar los parámetros del proyecto, desde el nombre y el directorio de trabajo hasta los nombres de los paquetes de clases Java y parámetros de conexión a la base de datos.
Luego de los pasos iniciales (configuración y creación del proyecto, y mapeado de entidades a partir de las tablas de la base de datos), seam-gen genera los archivos y directorios presentados en la Figura 9.



classes	11/11/2010 09:17 a.m.	Carpeta de archivos	
dist	11/11/2010 09:18 a.m.	Carpeta de archivos	
exploded-archives	11/11/2010 09:17 a.m.	Carpeta de archivos	
lib	11/11/2010 09:17 a.m.	Carpeta de archivos	
resources	11/11/2010 09:17 a.m.	Carpeta de archivos	
src	11/11/2010 09:17 a.m.	Carpeta de archivos	
view	11/11/2010 09:18 a.m.	Carpeta de archivos	
build.properties	15/06/2010 11:53 a.m.	Archivo PROPERTIES	1 KB
build.xml	15/06/2010 11:53 a.m.	Documento XML	21 KB
build-dev.properties	28/06/2010 10:54 a.m.	Archivo PROPERTIES	1 KB
build-prod.properties	15/06/2010 11:53 a.m.	Archivo PROPERTIES	1 KB
debug-glassfish-siga.launch	15/06/2010 11:53 a.m.	Archivo LAUNCH	1 KB
debug-jboss-siga.launch	15/06/2010 11:53 a.m.	Archivo LAUNCH	1 KB
deployed-jars-ear.list	15/06/2010 11:53 a.m.	Archivo LIST	1 KB
deployed-jars-war.list	15/06/2010 11:53 a.m.	Archivo LIST	1 KB
explode.launch	15/06/2010 11:53 a.m.	Archivo LAUNCH	2 KB
glassfish-build.xml	15/06/2010 11:53 a.m.	Documento XML	11 KB
glassfish-readme.txt	15/06/2010 11:53 a.m.	Documento de texto	3 KB
hibernate-console.properties	15/06/2010 11:53 a.m.	Archivo PROPERTIES	1 KB
seam-gen.properties	15/06/2010 11:53 a.m.	Archivo PROPERTIES	1 KB

Figura 9. Lista de archivos de un proyecto *Seam*

A continuación se describen los elementos más importantes de este directorio:

- **classes:** Clases Java compiladas en formato CLASS
- **dist:** Compilados empaquetados en formatos EAR, JAR y WAR
- **exploded-archives:** Contiene en un directorio los archivos compilados del paquete EAR
- **lib:** Bibliotecas externas requeridas por *Seam*
- **resources:** Archivos de configuración XML
- **src:** Códigos fuente de las clases Java
- **view:** Páginas de vista XHTML
- **build.properties:** Configuración de los servidores de aplicaciones para el proyecto
- **build.xml:** Archivo de configuración del compilador ANT (interno al *Seam*)
- **hibernate-console.properties:** Guarda los parámetros de conexión a la base de datos establecidos desde seam-gen
- **seam-gen.properties:** Guarda todos los parámetros establecidos desde el seam-gen, incluyendo los de hibernate

Capítulo III
Caso de Estudio, Sistema de Gestión de una Asignatura

1. Funcionalidad del sistema

El presente caso de estudio es un sistema que se ha desarrollado por completo dos veces, una haciendo uso del *framework* de desarrollo *Seam* y otra sin hacer uso de ningún *framework* ni componentes, programando la aplicación desde cero del modo tradicional usando Servlets y patrones de diseño para entidades y clases de control.
El sistema es una pequeña intranet que permite la gestión de una asignatura, esto involucra: iniciar sesión, listar alumnos, gestión del material de la asignatura, publicación de avisos, gestión de clases y evaluaciones. Una vez que el usuario inicia sesión, el sistema le asigna automáticamente el rol de alumno o profesor de acuerdo a su información en la base de datos.
En el CD adjunto al presente documento se encuentra una copia de ambos proyectos junto con el script de creación de la base de datos.
A continuación se presenta un breve análisis y diseño del sistema a modo general.

2. Casos de uso del sistema

Los actores involucrados son Profesor y Alumno.
El profesor puede listar alumnos, registrar clases, crear y eliminar evaluaciones, publicar y eliminar material, y publicar y eliminar avisos (Figura 10).
El alumno puede consultar clases, ver evaluaciones, consultar material y avisos de sus asignaturas (Figura 11).

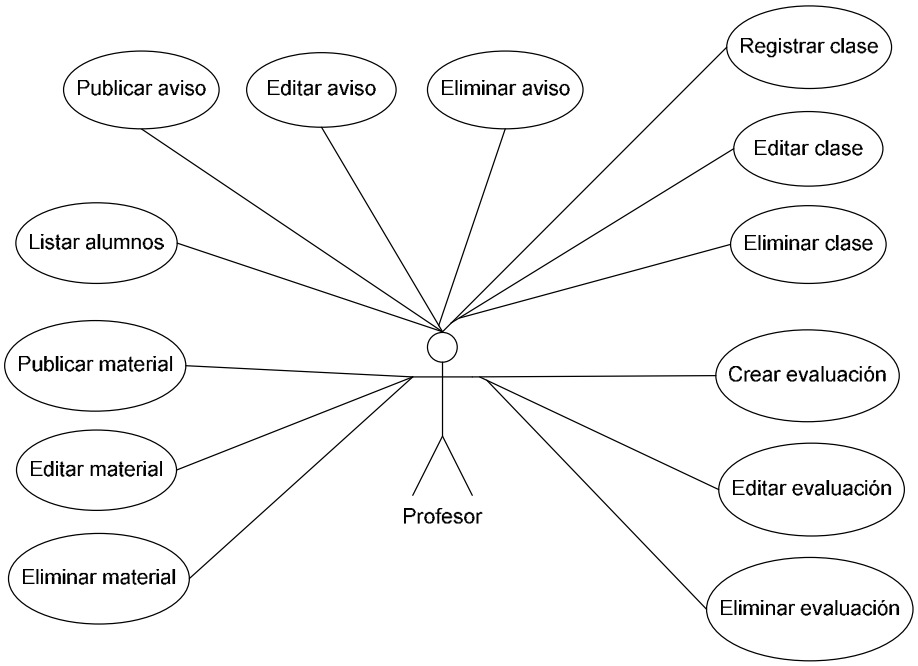


Figura 10. Casos de Uso de Profesor

Listar alumnos	
Actores	Profesor
Descripción	Muestra una lista con los alumnos que se encuentran matriculados en el curso, por cada alumno se muestran carné, apellidos, nombres y su programa académico.

Publicar material	
Actores	Profesor
Descripción	El profesor publica material en una asignatura que dicta, al publicar debe indicar el título, la descripción y la ruta al archivo. Al hacer la publicación se guardan automáticamente la fecha en que se hace la publicación y el profesor que publicó.

Editar material	
Actores	Profesor
Descripción	El profesor selecciona un material de los que tiene publicados en la asignatura y edita su información en el sistema.

Eliminar material	
Actores	Profesor
Descripción	El profesor selecciona un material de los que tiene publicados en la asignatura y lo elimina del sistema.

Registrar clase	
Actores	Profesor
Descripción	El profesor registra una clase en una asignatura que dicta, indicando la fecha

	en que se dictará.

Editar clase	
Actores	Profesor
Descripción	El profesor selecciona una clase de las que ha registrado en su asignatura y edita su información en el sistema.

Eliminar clase	
Actores	Profesor
Descripción	El profesor selecciona una clase de las que ha registrado en su asignatura y la elimina del sistema.

Crear evaluación	
Actores	Profesor
Descripción	El profesor registra una evaluación dentro de una de las asignaturas que dicta, debe indicar nombre, descripción, fecha en que se tomará y peso.

Editar evaluación	
Actores	Profesor
Descripción	El profesor selecciona una evaluación de las que ha registrado en su asignatura y edita su información.

Eliminar evaluación	
Actores	Profesor
Descripción	El profesor selecciona una evaluación de las que ha registrado en su asignatura y la elimina. Sólo puede eliminar evaluaciones que aun no se han tomado, es decir, programadas para fechas futuras.

Publicar aviso	
Actores	Profesor
Descripción	El profesor publica un aviso dentro de una asignatura, especificando título y contenido. Al publicar se guarda automáticamente la fecha en que se publica el aviso y el profesor que publicó.

Editar aviso	
Actores	Profesor
Descripción	El profesor selecciona un aviso en una de las asignaturas que dicta y edita su información en el sistema.

Eliminar aviso	
Actores	Profesor
Descripción	El profesor selecciona un aviso en una de las asignaturas que dicta y lo elimina del sistema.

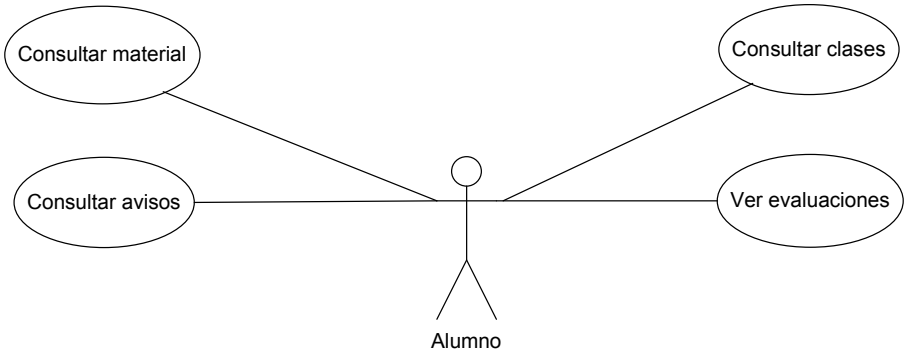


Figura 11. Casos de Uso de Alumno

Consultar material	
Actores	Alumno
Descripción	El alumno lista y descarga el material publicado por el profesor del curso dentro de una asignatura que lleva.

Consultar avisos	
Actores	Alumno
Descripción	El alumno lee los avisos publicados por el profesor del curso en una asignatura.

Consultar clases	
Actores	Alumno
Descripción	El alumno lista las fechas en que se ha dictado o se ha programado clases en una asignatura en la que se encuentra matriculado.

Ver evaluaciones	
Actores	Alumno
Descripción	El alumno lista las evaluaciones de un curso que esté llevando.

3. Modelo relacional

En la Figura 12 se puede apreciar el modelo relacional de la base de datos del sistema. El diccionario de datos se puede encontrar en el Anexo A: Diccionario de Datos.

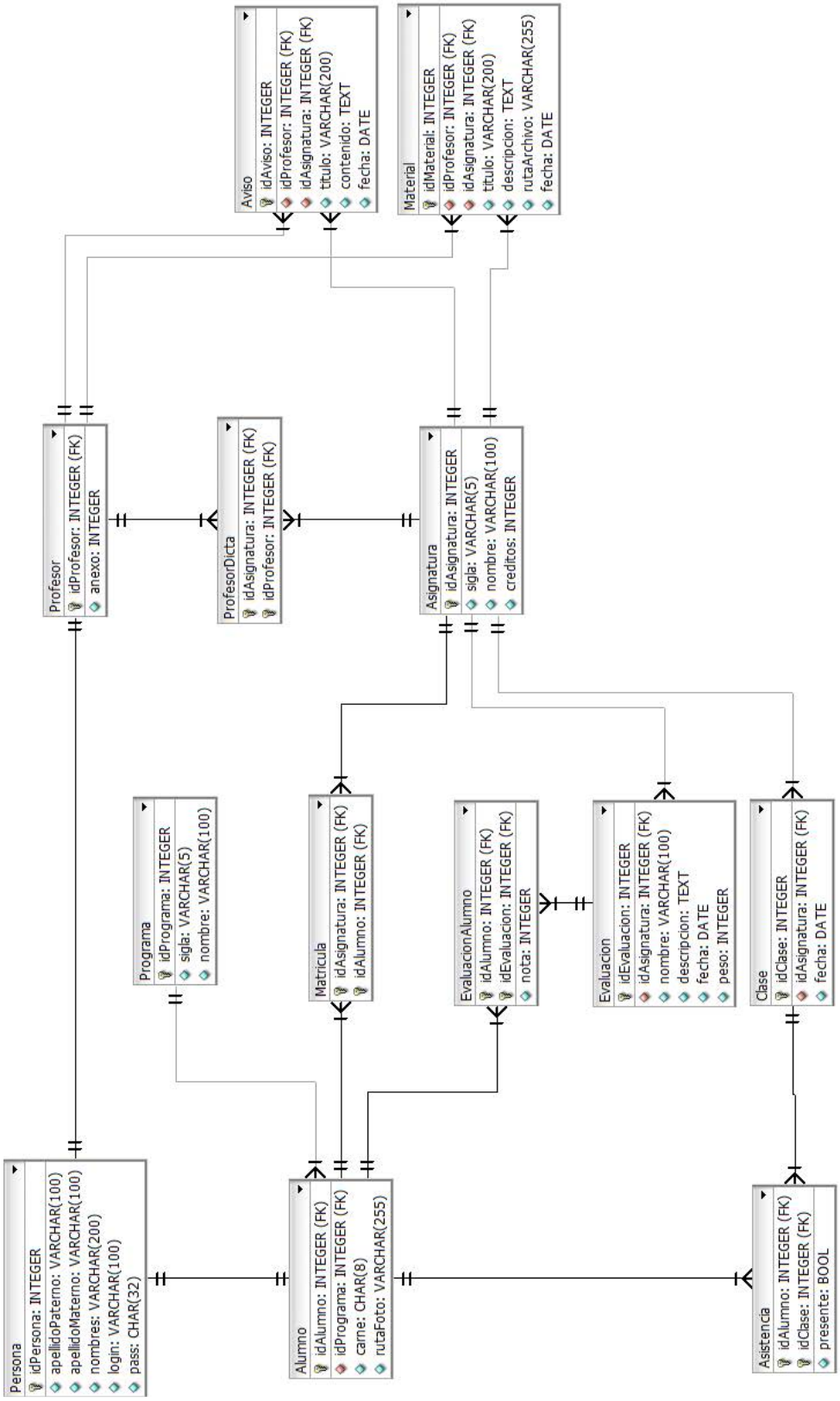


Figura 12. Modelo Relacional

4. Diseño de pantallas

A continuación se presenta el diseño de pantallas del sistema. Las opciones a las que tendrá acceso el usuario dependerán de su rol: alumno o profesor.

4.1.Inicio de sesión

Antes de poder acceder al sistema el usuario debe ingresar su nombre usuario y su contraseña para iniciar sesión (Figura 13).

Figura 13. Inicio de sesión

4.2.Acceso como alumno

El sistema valida si tiene acceso como alumno, si es así le asigna el rol de alumno y le da acceso a sus asignaturas matriculadas. A continuación tenemos las pantallas del alumno.

4.2.1. Pantalla de inicio

El usuario accede a esta pantalla luego de que inicia sesión y el sistema le da acceso de alumno. Desde esta pantalla el alumno accede a las asignaturas que tiene matriculadas. En este ejemplo el alumno de usuario jruizr está matriculado en A2 y ADS (Figura 14).

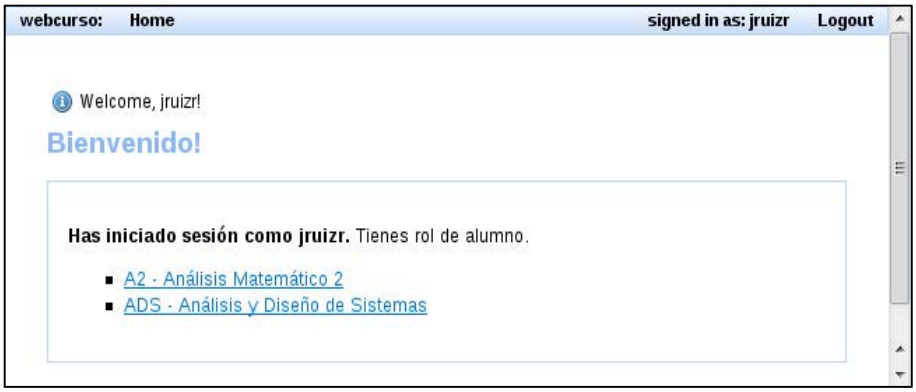


Figura 14. Pantalla de inicio (alumno)

Luego de elegir una asignatura, el alumno tiene acceso a la vista de asistencias, avisos, evaluaciones y material.

4.2.2. Ver clases y asistencias

En esta pantalla el alumno verá la lista de clases registradas por el profesor y un detalle que le indica si estuvo presente o no en la clase (Figura 15).

Figura 15. Ver clases y asistencias

4.2.3. Ver avisos de asignatura

Los avisos de asignatura son publicados por el profesor y se publican inmediatamente para los alumnos de la asignatura. El alumno verá la lista de avisos incluyendo el detalle completo de cada aviso y el profesor que publicó (Figura 16).

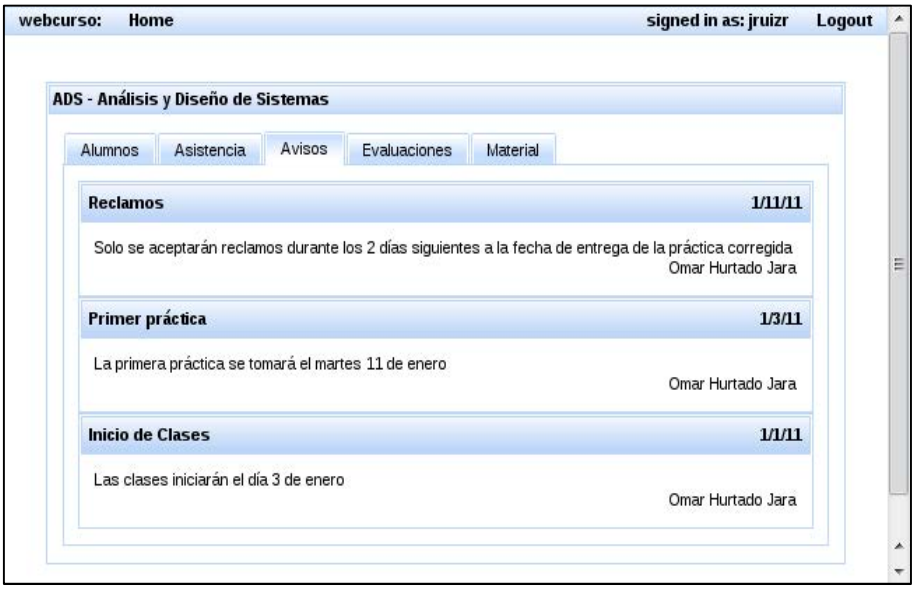


Figura 16. Ver avisos de asignatura

4.2.4. Ver evaluaciones y notas

El alumno puede ver también sus evaluaciones, así como el detalle de la fecha en que se tomó (o tomará), el peso de la evaluación y la nota que tuvo si es que ya fue calificada (Figura 17).

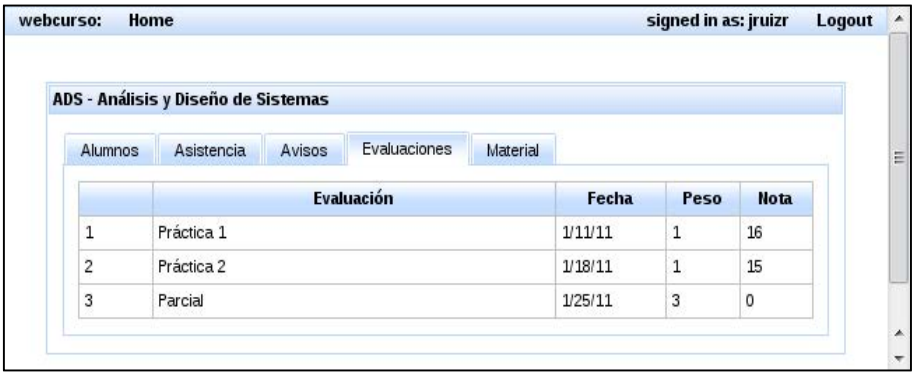


Figura 17. Ver evaluaciones y notas

4.2.5. Ver material de la asignatura

El profesor del curso publica material que los alumnos pueden descargar. El material estará disponible permanentemente hasta que el profesor decida quitarlo del sistema (Figura 18).

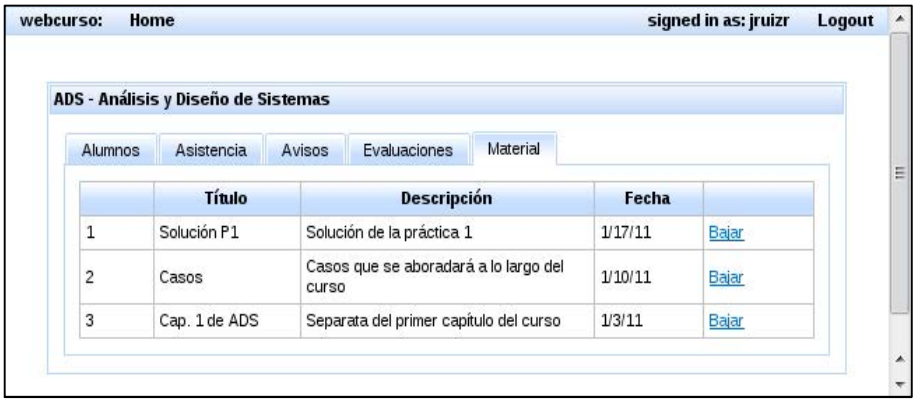


Figura 18. Ver material de la asignatura

4.3.Acceso como profesor

4.3.1. Pantalla de inicio

El usuario accede a esta pantalla luego de que inicia sesión y el sistema le da acceso de profesor, esto se puede dar sólo si el sistema validó previamente que el usuario no es alumno. Desde esta pantalla el profesor accede a las asignaturas que dicta. En este ejemplo el profesor de usuario ohurtado dicta ADS y BD (Figura 19).

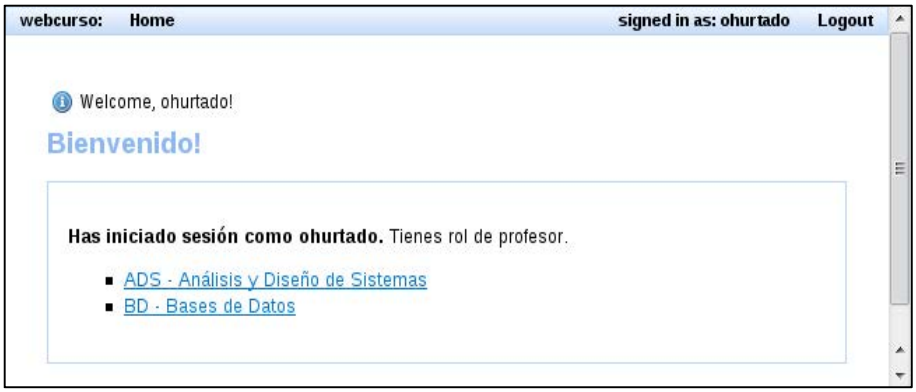


Figura 19. Pantalla de inicio (profesor)

Luego de elegir una asignatura, el profesor tiene acceso a la vista de alumnos, asistencias, avisos, evaluaciones y material.

4.3.2. Ver alumnos de la asignatura

El profesor puede consultar qué alumnos están matriculados en su asignatura, el sistema le muestra además de los apellidos y nombres, el carné y el programa académico (P.A.) de cada alumno (Figura 20).

Figura 20. Ver alumnos de la asignatura

4.3.3. Gestión de clases

4.3.3.1.Listar clases

La lista de clases le muestra las fechas de las sesiones de clase y la cantidad de asistencias que tuvo en cada clase (Figura 21).

Figura 21. Listar clases

4.3.3.2.Registrar clase

Al usar la opción “Registrar clase” el sistema lleva al profesor a una pantalla donde puede seleccionar una fecha y guardar la nueva clase en el sistema (Figura 22).

Figura 22. Registrar clase

4.3.3.3.Editar clase

Cuando el profesor da clic en Editar, accede a una pantalla desde la que puede cambiar la fecha de una clase. Adicionalmente, desde esta pantalla tiene la opción para Eliminar la clase (Figura 23).

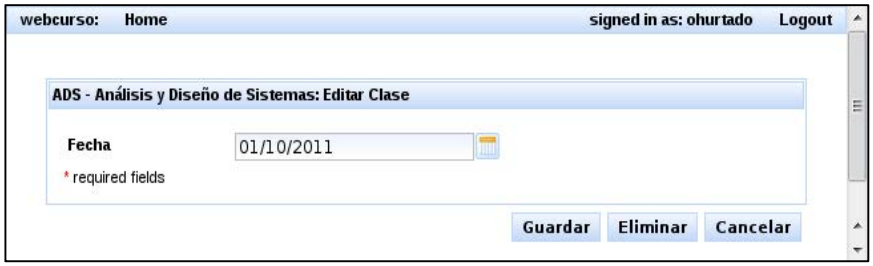


Figura 23. Editar clase

4.3.4. Gestión de avisos

4.3.4.1.Listar avisos

El profesor tiene una lista con el detalle de los avisos, similar al que ve el alumno, pero tiene además las opciones para publicar y editar avisos (Figura 24).

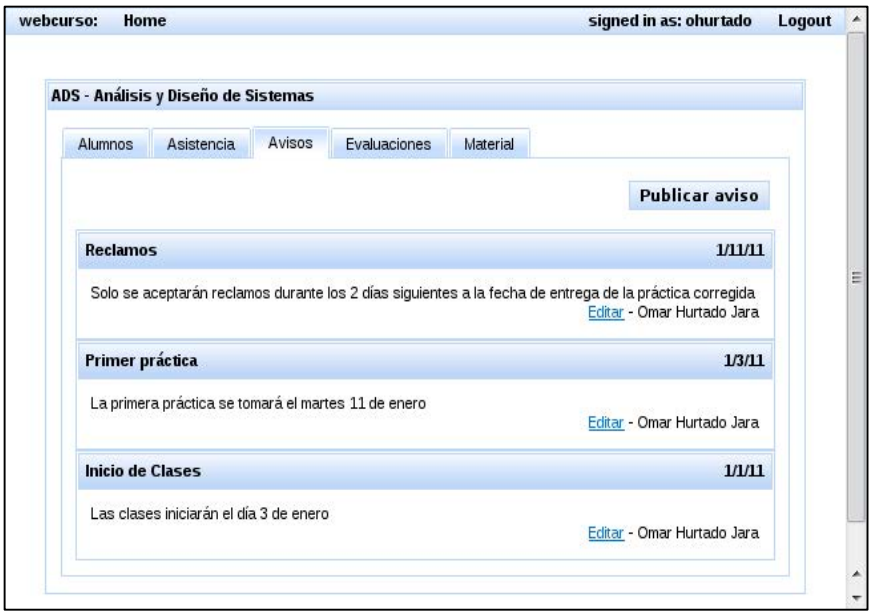


Figura 24. Listar avisos

4.3.4.2.Publicar aviso

Desde la opción Publicar aviso, el profesor podrá definir un título y un contenido para el aviso y a continuación publicarlo para que los alumnos puedan leerlo (Figura 25).

Figura 25. Publicar aviso

4.3.4.3.Editar aviso

Al editar un aviso el profesor puede cambiar el título o el contenido. También tiene la opción de eliminar del sistema el aviso que se encuentra editando (Figura 26).

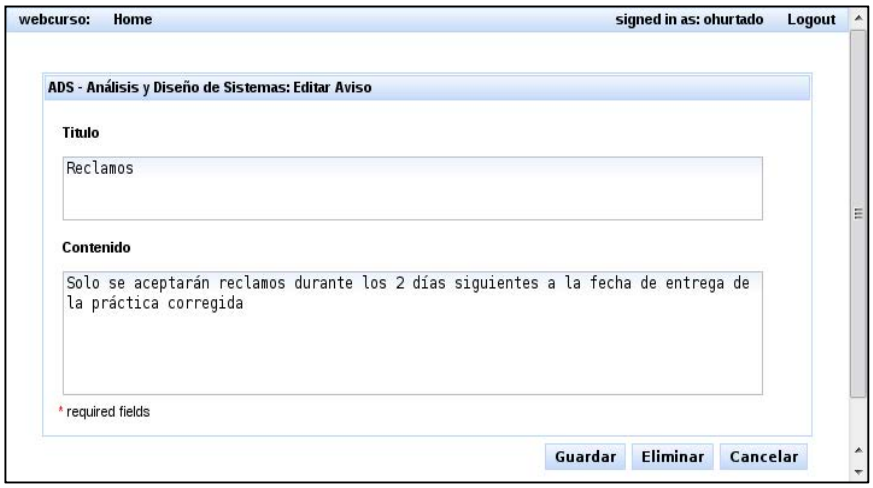


Figura 26. Editar aviso

4.3.5. Gestión de evaluaciones

4.3.5.1. Listar evaluaciones

La lista de evaluaciones muestra el detalle del nombre, fecha y peso de la evaluación. Tiene las opciones para registrar y editar evaluaciones (Figura 27).

Figura 27. Listar evaluaciones

4.3.5.2. Registrar evaluación

El profesor puede registrar una evaluación, especificando nombre, descripción, peso y fecha (Figura 28).

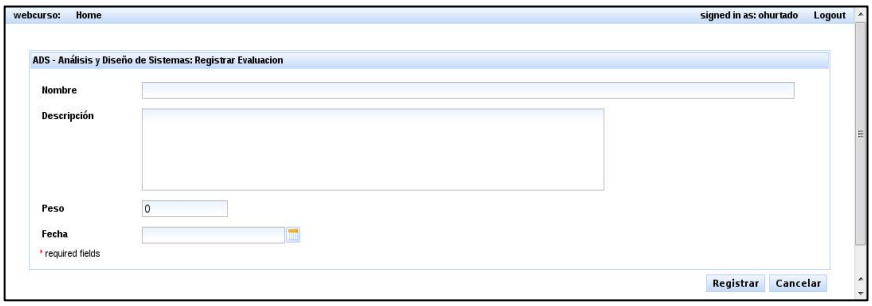


Figura 28. Registrar evaluación

4.3.5.3.Editar evaluación

Al editar una evaluación se puede modificar cualquiera de sus campos, y adicionalmente el profesor tiene la opción para eliminar la evaluación (Figura 29).

Figura 29. Editar evaluación

4.3.6. Gestión de material

4.3.6.1.Listar material de asignatura

El profesor lista el material publicado en la asignatura (ya sea por él o por otros profesores) y tiene la posibilidad de bajar, editar o publicar material nuevo (Figura 30).

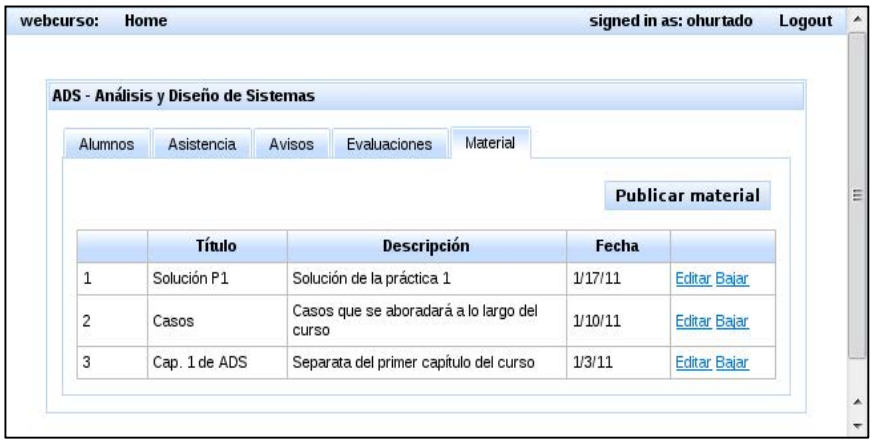


Figura 30. Listar material de asignatura

4.3.6.2.Publicar material

Cuando un profesor publica material debe definir un título y una descripción, y además un archivo que seleccionará de su máquina local para que los alumnos puedan bajarlo (Figura 31).

Figura 31. Publicar material

4.3.6.3.Editar material

Cuando el profesor usa la opción editar, podrá cambiar el título, la descripción o inclusive subir un archivo nuevo que reemplazará al anterior en el material publicado. Adicionalmente cuenta con la opción eliminar que borrará el material del sistema (Figura 32).

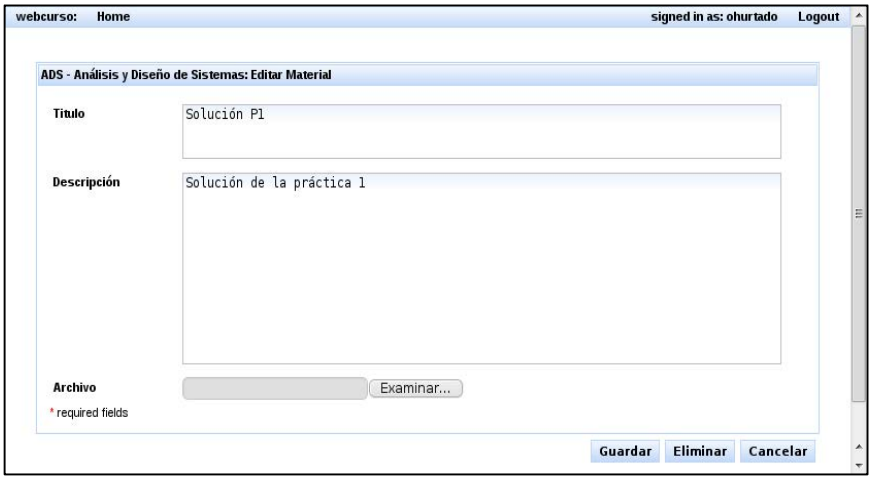


Figura 32. Editar material

Capítulo IV
Pruebas de Calidad Realizadas sobre el Framework

1. Generalidades

El caso de estudio, conformado por los dos sistemas de los que se habló en el capítulo anterior, fue desarrollado por el autor de la presente tesis. El detalle del software utilizado se muestra en la Tabla 1.

Tabla 1. Software utilizado para el desarrollo del caso de estudio

Tipo	Software – Servlets	Software – *Seam*
Sistema operativo	openSUSE 11.2	openSUSE 11.2
Kernel	default 2.6.31.14-0.4	default 2.6.31.14-0.4
Entorno de desarrollo (IDE)	eclipse 3.5.1	*JBoss Developer Studio* 3.0.1.GA
Base de datos	mySQL 5.1.49	mySQL 5.1.49
Servidor de aplicaciones	Apache Tomcat 5.5.28	*JBoss* 4.2.3.GA
Máquina virtual Java	JDK 1.6.0_16	JDK 1.6.0_16

Al llevar a cabo el desarrollo del caso de estudio, se observó que la diferencia más notable se encontraba en el tiempo de desarrollo. El propósito de este capítulo es hacer una revisión de las características comparadas entre los dos sistemas y dar una valoración de las mismas.

Una característica muy importante del *framework Seam* es el uso de Hibernate, un *framework* de persistencia de bases de datos. Basta con configurar los parámetros de conexión a la base de datos para que se generen automáticamente, gracias a Hibernate y la herramienta seam-gen, las clases entidad y las clases de control para manejar la lectura y modificación de información. Se ha estimado que esta parte de la aplicación ahorraría un 30% del tiempo de desarrollo ya que deja las clases listas para ser usadas en cuestión de segundos.

Otra característica muy importante es el uso de JSF y JSTL para las interfaces de usuario de la aplicación, esto permite generar rápidamente código HTML que el usuario visualizará a partir del uso de etiquetas dinámicas. Estas etiquetas permiten escribir rápidamente tablas, listas y otros componentes de iteración de datos.

Existe un IDE desarrollado exclusivamente para el *framework* llamado *JBoss Developer Studio*, está basado en Eclipse y fue desarrollado por RedHat. El IDE entre otras cosas

facilita el uso de los componentes anotados de *Seam* y la escritura de plantillas XHTML con sugerencias de código para agilizar el trabajo. En la Figura 33 se puede apreciar una captura de pantalla del programa.

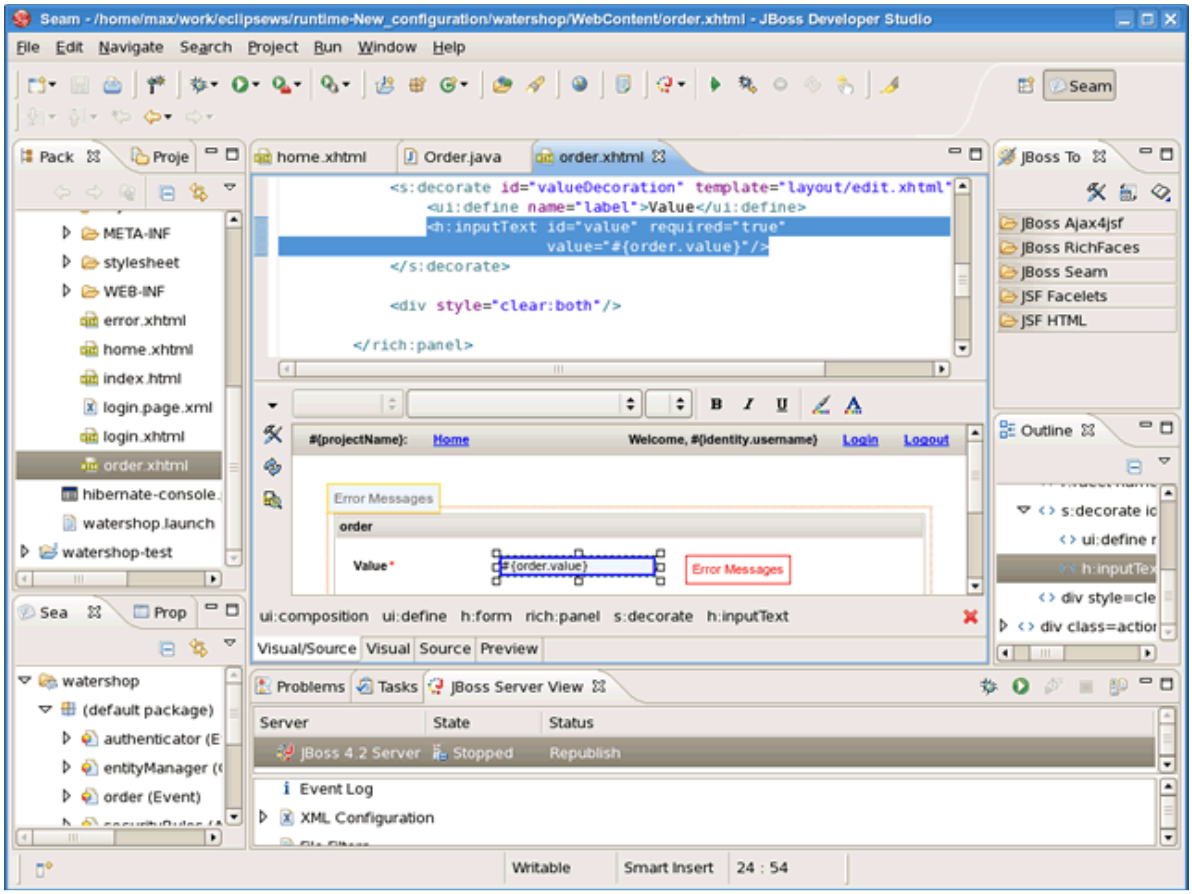


Figura 33. *JBoss Developer Studio*

2. Factores e indicadores

Con el propósito de realizar una comparación imparcial se ha seleccionado un conjunto de criterios con los cuales se analizará el desarrollo de la aplicación con y sin el *framework*. Los criterios utilizados fueron:

2.1. Curva de aprendizaje

La curva de aprendizaje nos permitirá determinar la facilidad con la que se puede empezar a trabajar en el caso desarrollado. Se determinará, tomando como base un conocimiento avanzado del lenguaje Java y la tecnología Servlets, aproximadamente cuánto tiempo toma comprender el funcionamiento del *framework*, sus componentes, cómo se desarrolla y despliega aplicaciones.

2.2. Tiempo de desarrollo

Con respecto al tiempo de desarrollo podemos afirmar que el proyecto seleccionado para el estudio es bastante simple y no requiere operaciones

demasiado complejas de programar. Para medir el tiempo de desarrollo fueron tomados en cuenta solamente 2 factores: el número de líneas de código escritas por el desarrollador una vez que la aplicación estuvo ya puesta en producción; y el tiempo aproximado de desarrollo teniendo en cuenta que la aplicación fue desarrollada por una sola persona.

2.3. Desempeño en producción

Un factor muy importante al momento de evaluar un sistema es su desempeño en producción. Para llevar la evaluación se efectuaron pruebas sobre ambos sistemas utilizando el software *Apache JMeter*. Este software sirve para generar pruebas de estrés simulando peticiones HTTP y ha sido desarrollado por *Apache Software Foundation.*

2.4. Despliegue en producción

Para medir el cumplimiento de este criterio se llevaron ambas aplicaciones a un entorno lo más parecido posible a uno de producción. Para el caso de *Seam* significó colocarlo sobre un servidor *JBoss*, y para el caso de la aplicación tradicional se usó un servidor Apache Tomcat. Se realizaron pruebas por separado en las que se midió el tiempo que toma desplegar la aplicación por primera vez, y el tiempo que toma llevar a producción una actualización sobre la aplicación ya desplegada.

2.5. Costo de alojamiento

Para llevar a cabo este punto de comparación, se estudió el costo promedio de contratar un servicio de hosting con las características de hardware y software necesarias para ejecutar una aplicación *Seam* y para ejecutar una aplicación tradicional en Java Servlets.

2.6. Soporte y mantenimiento de la aplicación

Un criterio de igual importancia que los anteriores es la capacidad de mantenimiento que se tendrá sobre la aplicación luego de ser desarrollada. El equipo responsable de dar soporte y mantenimiento deberá ser capacitado ya sea contratando a una empresa experta que provea el asesoramiento necesario o recurriendo documentación disponible en la Web provista por el *framework* y por los usuarios del mismo.

3. Resultados de las pruebas

A continuación se presentan los resultados obtenidos de las pruebas de calidad que se llevaron a cabo sobre las dos aplicaciones.

3.1. Curva de aprendizaje

El desarrollador ya contaba con experiencia previa en desarrollo en *Seam* gracias a un proyecto abordado anteriormente, del cual se pudo estimar que en promedio el tiempo para tener un dominio avanzado de *Seam* y sus herramientas es de 2 meses, considerando que la persona a capacitarse ya cuenta con un dominio avanzado de desarrollo de aplicaciones Web con tecnologías Java. En otras palabras, el tiempo

de aprendizaje se ha medido partiendo de la premisa de que ya se cuenta con el conocimiento de Servlets.

3.2.Número de líneas de código

Teniendo las aplicaciones ya desarrolladas de la manera tradicional y luego utilizando el *framework Seam*, se procedió a contabilizar el número de líneas que las aplicaciones tuvieron. Los totales de líneas de código contadas se muestran en la Tabla 2.

Tabla 2. Resumen del número de líneas obtenidas

Sección	Servlets	*Seam*	Ahorro
Líneas escritas a mano	4142	375	90.95%
Líneas auto-generadas	0	3270	-
Total	4142	3645	12%

Se ha dividido el número de líneas en dos secciones con el propósito de que se aprecien algunos detalles importantes. *Seam* genera automáticamente código gracias a su herramienta seam-gen, de la que se habló previamente. Una vez que la aplicación ha sido generada, el desarrollador debe revisarla para determinar qué cambios deberá hacer manualmente, de manera que ésta cumpla con sus requerimientos de funcionalidad. En general estos cambios han sido mínimos.
En la Figura 34 se aprecian los porcentajes de líneas de código de la aplicación escrita en *Seam*.

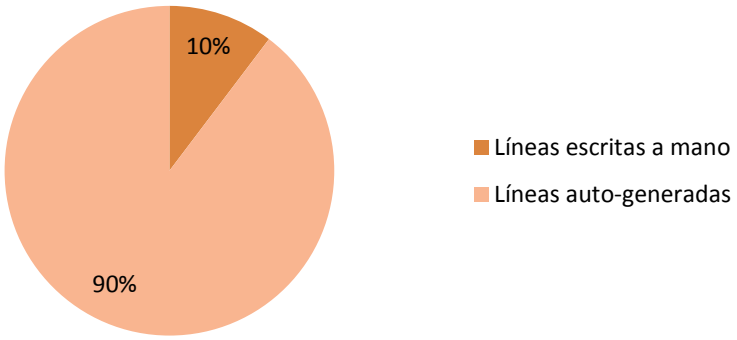


Figura 34. Líneas de código de la aplicación escrita en Seam

Por otro lado, programando del modo tradicional, todo el código debe ser escrito a mano, dando como resultado una cantidad mucho mayor de líneas. Cabe resaltar que a pesar de tener mucho código auto-generado, *Seam* tiene en total 12% menos código que la programación del modo tradicional (ver Figura 35).
En la Figura 35 se muestra una comparativa entre los totales de líneas de código de ambas aplicaciones.

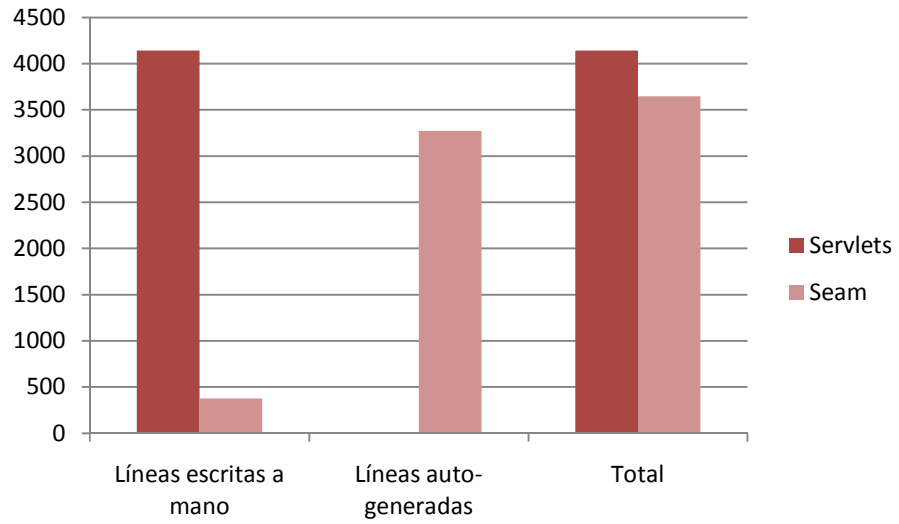


Figura 35. Comparativa de líneas de código entre las aplicaciones en Seam y Servlets

La cantidad de código escrita con *Seam* es aproximadamente el 9% de la cantidad de código que fue necesario escribir de la forma tradicional, tal y como se puede apreciar en la Figura 36.

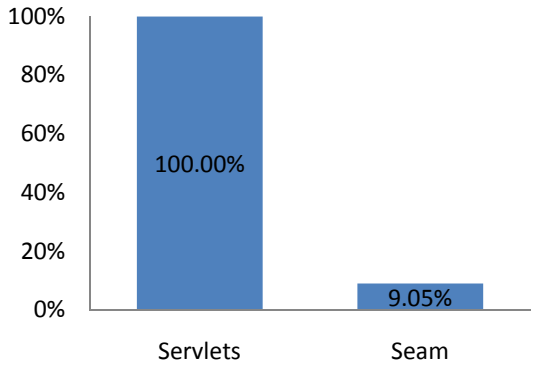


Figura 36. Líneas de código escritas a mano representadas en forma porcentual

3.3.Tiempo de desarrollo

El tiempo utilizado para desarrollar la aplicación con y sin *framework* ha sido significativamente distinto. Considerando que para las dos aplicaciones el desarrollador fue una sola persona, los tiempos utilizados fueron de 2 días para *Seam* y aproximadamente 2 semanas para el modo tradicional. El incremento del tiempo se explica claramente por la cantidad de líneas de código que fue necesario escribir.

Es importante además, tomar en cuenta que el desarrollador ya contaba con experiencia tanto en programación tradicional como en *Seam*, por lo que el tiempo medido no incluye el periodo de aprendizaje.

Tal como se aprecia en la Figura 37 se ahorra poco más del 71% del tiempo de desarrollo.

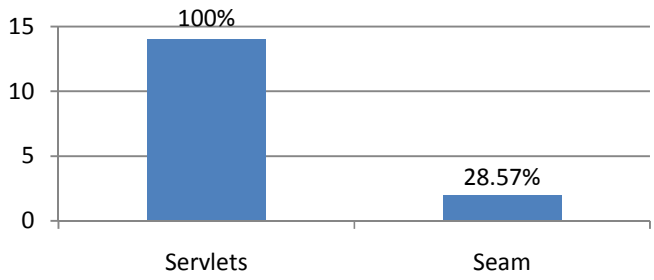


Figura 37. Tiempo de desarrollo en días

3.4.Desempeño en producción

Para efectuar las pruebas de rendimiento, se tomó como base la información de la facultad de Ingeniería de Campus Piura, en la cual se tienen aproximadamente 1000 alumnos y 100 profesores. Se asume que en una hora pico, ingresarán en forma concurrente como máximo menos del 10% del total de usuarios, por lo tanto se hicieron múltiples pruebas, con 10, 50 y 100 usuarios concurrentes en un tiempo de un minuto.

Se midieron los tiempos para dos pruebas sobre cada aplicación: acceso de alumno para consulta de un curso; y acceso de profesor, para consulta de un curso y publicación de un aviso. Los resultados medidos se presentan en la Tabla 3.

Tabla 3. Tiempos de respuesta medidos en las pruebas de estrés

Tiempos de respuesta promedio (ms) para la prueba de Alumno		
	Servlets	**Seam**
10 concurrencias	3,066	518
50 concurrencias	21,274	3,154
100 concurrencias	59,516	5,268
Tiempos de respuesta promedio (ms) para la prueba de Profesor		
10 concurrencias	9,631	3,636
50 concurrencias	77,156	11,572

De los datos presentados, se ha calculado que en promedio Seam presenta un tiempo de respuesta que es solo el 14% del promedio para Servlets, tal como se puede apreciar en la Figura 38.

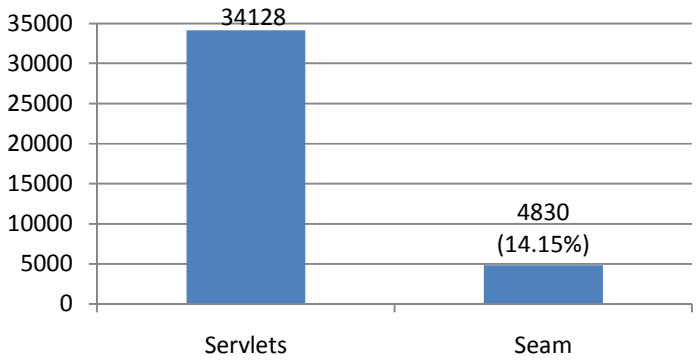


Figura 38. Promedios de los tiempos de respuesta en milisegundos

3.5. Despliegue en producción

En una instalación por defecto el *framework* utiliza un servidor Web que exige una cantidad de recursos bastante alta, *JBoss*; mientras que del modo tradicional se usa Apache Tomcat, un servidor Web bastante liviano. Ambos servidores se ejecutan con base en Java, escuchan por defecto el puerto 8080 para peticiones HTTP, lo cual evita que entre en conflicto si se tiene ya un servidor Web como Apache escuchando por el puerto estándar (80).

No se recomienda usar directamente Tomcat ni *JBoss* para atender las peticiones de los usuarios, sino utilizar Apache como intermediario entre las peticiones de usuario y las respuestas de los servidores Java, mejorando de esta forma tanto el desempeño como la seguridad del sistema.

En ambos casos el despliegue de las aplicaciones fue bastante simple, basta con compilar el proyecto y que un "paquete compilado" (en el caso de Tomcat un archivo WAR, para *JBoss* un archivo EAR) vaya al directorio de despliegue de su respectivo servidor. Ambos servidores tienen la capacidad de desplegar "en caliente", es decir, sin necesidad de reiniciar, se toman en cuenta las nuevas aplicaciones desplegadas.

Una diferencia importante en el despliegue se encuentra en el tiempo que lleva iniciar el servidor, el cual depende del hardware sobre el que se está ejecutando.

A continuación se muestra en la Tabla 4 los datos tomados del despliegue en producción.

Tabla 4. Tiempos de despliegue y hardware utilizado

	Servlets	***Seam***
CPU	2.40 GHz	2.40 GHz
Memoria	4 Gb	4 Gb
Servidor	Tomcat	*JBoss*
Tiempo de despliegue	2 segs.	30 segs.

En la Figura 39 se aprecia que el servidor Tomcat despliega una aplicación un 93% más rápido que *JBoss*.

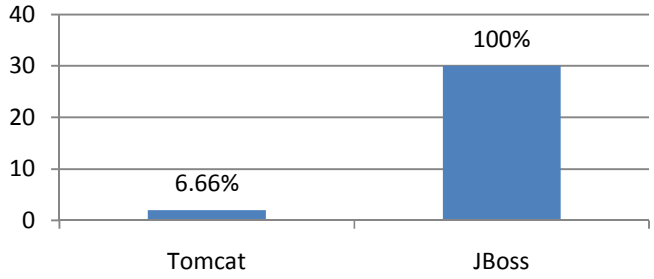


Figura 39. Tiempos de despliegue en segundos

3.6.Costo de alojamiento

Las pruebas que se realizaron consistieron en utilizar Google para encontrar los proveedores de hosting Java más populares, utilizando las palabras clave "hosting tomcat jboss". A continuación las tarifas de los 5 primeros resultados en la Tabla 5.

Tabla 5. Resumen de tarifas de hosting Java en dólares por mes

	Tomcat	***JBoss***
jspzone.net	16.99	19.95
Webhostingjava.net	8, 25	40
astrahosting.com	8, 25	40, 60
dailyrazor.com	9.95 – 32.95	59.95 – 79.95
hostjava.net	7.99 – 14.99	50

Como se puede apreciar, las tarifas varían, esto depende de las características de hardware y restricciones de transferencia de datos que se pueda tener en el servicio, sin embargo lo más destacable es la diferencia en el costo mensual que se da cuando se contrata *JBoss* en vez de Tomcat, esto es debido a que *JBoss* es un servidor que requiere licencia y, como ya se ha explicado, tiene una serie de características especializadas J2EE con las que Tomcat no cuenta.
En el Figura 40 tenemos la comparación entre los costos promedio del hosting por mes, calculados a partir de los datos de la Tabla 5.

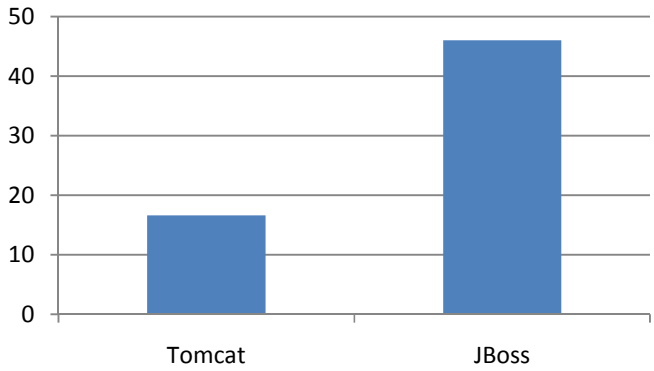


Figura 40. Costo promedio de hosting en dólares por mes

3.7.Soporte y mantenimiento de la aplicación

Es muy importante tener en cuenta antes de decidir qué tecnología se usará, qué capacidad se tiene de encontrar soporte para el mantenimiento posteriormente. Para *Seam* es necesario tomar en cuenta la documentación en línea que provee el portal oficial de *Seam* así como los foros de la comunidad oficial de *Seam*. Por otro lado, podría resultar muy conveniente contar con soporte a nivel local, los cursos de *Seam* son una buena alternativa de capacitación. En el Perú la empresa RedHat tiene a cargo las capacitaciones en *Seam* a través de empresas *partners*.

El soporte a nivel de foros está bastante extendido para *Seam*. Poseen una comunidad creada por los mismos desarrolladores de *Seam* (seamframework.org), donde los usuarios pueden registrarse, descargar tutoriales, manuales, tienen acceso a la documentación de *Seam* disponible hasta el momento (todavía está incompleta en algunos apartados), y plantear sus dudas que serán resueltas por otros expertos en *Seam* y en algunos casos por los mismos desarrolladores del *framework.*

Para el caso de Java Servlets y la programación tradicional de una aplicación Web Java existen decenas de foros de soporte y toda una amplia variedad de componentes con su respectiva documentación para agilizar el proceso de desarrollo. Debido a que es una tecnología tan ampliamente difundida y de poca complejidad, las complicaciones al desarrollar de este modo son mínimas.

Capítulo V
Conclusiones

Seam es un *framework* de desarrollo para la construcción de aplicaciones de nueva generación Web 2.0, unifica tecnologías tales como Ajax, JSF y EJB para que funcionen en conjunto y faciliten el desarrollo ágil de aplicaciones de alta interacción con el usuario.

En cuanto a las características de *Seam* se verificó lo siguiente:

1. **Generación automática de código.** La herramienta seam-gen del *framework* genera automáticamente todas las clases Java de tipo entidad mapeadas directamente de las tablas de la base de datos, y las clases de control para el manejo de registros de las tablas mapeadas. Adicionalmente genera las interfaces de usuario que a través de llamados mediante Lenguaje de Expresiones (EL, por sus siglas en inglés) a las clases generadas, gestiona todo el contenido de la base de datos. En otras palabras, seam-gen permite generar rápida y automáticamente aplicaciones básicas CRUD (siglas en inglés para crear, recuperar, actualizar y eliminar).

2. **Unión de tecnologías.** Las tecnologías EJB y JSF no podían usarse en conjunto sin trabajar con clases intermedias complejas que permitieran la comunicación entre ellas, conocidas como "JSF *Backing Beans*" o *beans* de soporte de JSF. *Seam* utiliza anotaciones para convertir cualquier clase Java en un EJB, y EL para poder hacer llamados a los EJBs desde cualquier interface de usuario e inclusive desde archivos de configuración XML.

3. **Nuevas tecnologías para las interfaces de usuario.** Tomando como base JSF, *Seam* recurre a la tecnología JSTL para crear nuevas etiquetas para la generación de contenidos dinámicos en páginas Web. Las bibliotecas que incluye permiten la creación de componentes visuales "ricos" tales como pestañas, tablas con barra de desplazamiento, arrastrar y soltar objetos, etc.

4. **Uso simple de AJAX.** Los componentes que *Seam* incorpora a JSF permiten programar eventos AJAX sin necesidad de recurrir a la escritura de código JavaScript, pues éste es generado automáticamente a partir del código escrito en JSF con la biblioteca A4J (Ajax4JSF). En otras palabras, se puede usar Ajax sin saber escribir Ajax.

5. **Programación en capas.** *Seam* recurre al patrón MVC para organizar la forma en que está estructurado su código, de tal forma que se puede diferenciar claramente los componentes del modelo (entidades), las clases de control (EJBs) y vista del sistema (interfaces de usuarios escritas en JSF). El equipo de desarrollo puede trabajar con cualquiera de estas capas sin afectar a las otras.

6. **Un *framework* de integración.** Trabajar con múltiples *framework*s en conjunto trae como consecuencia la necesidad de configurar separadamente cada *framework* y prepararlos para funcionar unidos, lo que implica una gran cantidad de archivos de configuración y código, y muchas horas de trabajo. *Seam* se considera un *framework* de integración porque toma las tecnologías de múltiples *framework*s y las unifica, de tal forma que al desarrollar sólo se prepara la configuración de *Seam* y se puede pasar directamente a trabajar con todas las herramientas que proveen los *framework*s que unifica incluyendo las propias de *Seam*.

7. **Entorno de desarrollo propio.** RedHat ha desarrollado un IDE (entorno de desarrollo) basado en Eclipse, llamado *JBoss Developer Studio*, el cual automatiza muchas tareas específicas de desarrollo en *Seam*. El IDE incluye características para trabajar con JSF y las bibliotecas de *Seam* (parte visual) así como soporte para anotaciones y la inclusión de componentes en Java (parte lógica). El uso del IDE agiliza la programación de aplicaciones.

Ventajas encontradas en el desarrollo con *Seam*

1. **Ahorro en el tiempo de desarrollo.** El tiempo que se puede ahorrar programando en *Seam* es muy significativo, a tal punto que para proyectos muy grandes se podría crear aplicaciones hasta **en menos de la cuarta parte del tiempo** que tomaría programar sin recurrir a un *framework*. Es muy importante tener en cuenta que para lograr estos ahorros en el tiempo es necesario contar un equipo de desarrollo debidamente preparado en *Seam*. El ahorro se logra gracias a las características propias de *Seam*, una vez que se tiene el código auto-generado como punto de partida, la unión de tecnologías agiliza todo el trabajo y permite obtener resultados muy rápidamente.

2. **Correcciones simplificadas.** Dado que el *framework* delimita claramente el área de acción sobre la que se puede programar, se simplifica la detección de errores. Los errores que se detecten durante la ejecución permitirán solucionar cualquier caso similar que aun no se haya presentado, pero que cumpla las mismas características, a lo largo de toda la aplicación.

3. **Reutilización de código.** Existe un alto nivel de reutilización cuando se programan aplicaciones en *Seam*, lo cual se hace notorio cuando se implementan tareas de gestión de registros de la base de datos (lecturas, listas y actualizaciones). Se logra un desarrollo más veloz pues ya no es necesario volver a escribir código repetitivo.

4. **JAVA: un lenguaje de programación mundialmente difundido.** Dado que *Seam* está hecho en Java, se puede combinar los componentes del *framework* con cualquier

biblioteca disponible en el lenguaje, esto es una fuerte ventaja para desarrollar aplicaciones pues en la mayoría de los casos se tendrá a la mano bibliotecas con las funcionalidades particulares que se pudiera necesitar. Otra ventaja de trabajar en Java está en que la conexión a las bases de datos depende de los drivers JDBC, es decir, cualquier base de datos que cuente con JDBC será compatible con *Seam*.

5. **Soporte para desarrolladores**. La documentación disponible de *Seam* es bastante completa y la comunidad de *Seam* es muy activa, a tal punto que inclusive los creadores del *framework* participan frecuentemente en sus foros, ayudando a resolver problemas cuando éstos tienen una complejidad muy alta, siguiendo de cerca el *framework* para posibles correcciones que sea necesario hacer y actualizándolo constantemente**.** Tanto la documentación como el acceso a los foros es gratuito. Adicionalmente, la compra a RedHat de alguna de las licencias de software relacionados a *JBoss* incluye un foro privado de soporte para sus clientes.

6. **Mejora en el desempeño.** A través de las pruebas de estrés ha quedado demostrado que bajo una carga alta de usuarios, la aplicación desarrollada en Seam se comporta de manera más eficiente que una aplicación básica desarrollada en Servlets. Los componentes que constituyen las aplicaciones de Seam están preparados y optimizados para funcionar en conjunto de manera que su desempeño cumpla con las expectativas que se tiene de un sistema empresarial.

Consideraciones al abordar un proyecto con *Seam*

1. **Tiempo de aprendizaje.** *Seam* es una tecnología relativamente nueva, e incorpora un conjunto de paradigmas de programación a los que el equipo de desarrollo podría no estar acostumbrado. Por lo tanto es necesario someterlos a una capacitación, y sólo cuando estén debidamente preparados, podrán comenzar a desarrollar. Es importante considerar que cuando pasen a desarrollar deberían practicar con aplicaciones pequeñas que les permitan detectar los errores más comunes, como una manera de finalizar el periodo de aprendizaje.

2. **Mantenimiento de las aplicaciones.** Para el mantenimiento se debe tener en cuenta que es necesario contar con un equipo debidamente preparado en *Seam*. Lo recomendable sería contar con al menos una parte del personal que estuvo a cargo del desarrollo, de lo contrario, será necesario pasar nuevamente por un periodo de capacitación al equipo que estará a cargo del mantenimiento.

3. **Consultas SQL.** Dado que *Seam* utiliza el *framework* Hibernate para las operaciones con la base de datos, éste se encarga de generar las consultas automáticamente a partir de las entidades programadas con anotaciones en *Seam*. El hecho de que Hibernate participe como intermediario entre la aplicación y la base de datos simplifica la programación porque se escribe muy poco código SQL, lo que resulta en otro ahorro de tiempo, sin embargo las consultas auto-generadas podrían ser poco eficientes y ocasionar lentitud en la aplicación. En estos casos se puede escribir las consultas manualmente para solucionarlo, Hibernate provee soporte para su propio lenguaje HQL e inclusive para lenguaje SQL nativo.

4. ***JBoss* Application Server.** *JBoss* es la empresa que creó originalmente *Seam*, antes de ser adquirida totalmente por RedHat. Cuenta con su propio servidor de aplicaciones, que lleva el mismo nombre, y es el que usa *Seam* por defecto para desplegar sus aplicaciones. La versión libre (community) contiene únicamente el servidor de aplicaciones mientras que la versión empresarial viene con una solución completa de actualizaciones, soporte para configuración y desarrollo y cursos de certificación. Para desarrollo y pruebas locales basta con trabajar con la versión libre, pero cuando se pase a producción, dado todo el soporte que viene detrás de la solución empresarial, es preferible optar por ésta. Es importante tener en cuenta que *JBoss* está conformado por un amplio rango de componentes, que deben ser inicializados cuando se enciende el servidor de aplicaciones, esto es lo que origina que el periodo de inicialización sea de al menos 30 segundos.

5. **Balanceo de carga.** Para sistemas de alta concurrencia, *JBoss* puede funcionar en forma distribuida, de manera que con el incremento de nodos o instancias, ejecutándose en múltiples servidores, se puede escalar las aplicaciones que se encuentren funcionando sobre *JBoss*.

Recomendaciones

1. **Aplicaciones de baja escala.** Si no se cuenta con conocimientos de *Seam* y la aplicación que se quiere desarrollar es de baja escala, teniendo en cuenta que el periodo de aprendizaje estimado es de 2 meses, es necesario evaluar si el tiempo que tomaría desarrollarla sin un *framework* podría ser menor al tiempo que tomará aprender y dominar *Seam* para luego poder programar la aplicación.

2. **Aplicaciones de gran escala.** En casos en que se sabe que la aplicación que se desarrollará debe ser lanzada a producción rápidamente y que además crecerá constantemente en el tiempo, utilizar *Seam* sería una buena opción. Una vez que el equipo de desarrollo esté suficientemente preparado, podrán lanzar nuevos módulos del sistema rápidamente con modificaciones mínimas sobre lo que ya está programado.

3. **Costos.** Para el despliegue en producción de aplicaciones sobre *JBoss*, ya sea contratando un servicio de hosting o invirtiendo en infraestructura propia, el costo siempre resulta mayor que el requerido para una infraestructura tradicional, por lo tanto es muy importante tener en cuenta, antes de optar por *Seam*, la inversión que conlleva ir por esta opción.

4. **Experiencia previa.** Si el equipo de desarrollo ya cuenta con dominio de *Seam* gracias a experiencias en aplicaciones anteriores, optar por *Seam* siempre será una buena decisión pues podrán sacar a producción rápidamente casi cualquier sistema que se necesite.

Bibliografía

Booth, G. (4 de Sept. de 2006). *Component-based Software Development.* Recuperado el 7 de Sept. de 2010, de myConference: http://active.cput.ac.za/myconference/Gregory%20Booth%20-%20Component%20Software%20Development.pdf

Brown, A., & Wallnau, K. (1996). Engineering of Component-Based Software. *Proceedings of the IEEE International Conference on Engineering of Complex.* Montreal, Canadá.

Clifton, M. (3 de Nov. de 2003). *What is a Framework?* Recuperado el 7 de Sept. de 2010, de The Code Project: Your Development Resource: http://www.codeproject.com/KB/architecture/WhatIsAFramework.aspx

Dpto. Ing. Electrónica, Sist. Informáticos y Automática, Universidad de Huelva. (2006). *Introducción a Java EE.* Recuperado el 8 de Nov. de 2010, de Nuevas Tecnologías de la Programación: http://www.uhu.es/josel_alvarez/NvasTecnProg/recursos/tTema1.pdf

Garrett, J. J. (18 de Feb. de 2005). *Ajax: A New Approach to Web Applications*. Recuperado el 28 de Dic. de 2010, de adaptive path: product experience strategy and design: http://www.adaptivepath.com/ideas/essays/archives/000385.php

JBoss. (s.f.). *JBoss Seam*. Recuperado el 8 de Nov. de 2010, de JBoss: http://www.jboss.com/products/seam/

Kalla, R. (10 de Jun. de 2008). *Which is the Hottest Java Web Framework? Or Maybe Not Java?* Recuperado el 7 de Sept. de 2010, de TheBuzzMedia.com: News, Video Games, Movies, Technology and Humor: http://www.thebuzzmedia.com/which-is-the-hottest-java-Web-framework-or-maybe-not-java/

Oracle. (s.f.). *Getting Started with Web Applications*. Recuperado el 07 de Sept. de 2010, de The J2EE 1.4 Tutorial: http://download.oracle.com/javaee/1.4/tutorial/doc/WebApp.html

O'Reilly, T. (30 de Sept. de 2005). *Design Patterns and Business Models for the Next Generation of Software*. Recuperado el 7 de Sept. de 2010, de O'Reilly Media - Technology Books, Tech Conferences, IT Courses, News: http://oreilly.com/web2/archive/what-is-web-20.html

Raymond, E. S. (1997). *The Cathedral and the Bazaar.* Thyrsus Enterprises.

Siddiqui, F. (10 de Dic. de 2000). *Component Based Software Engineering.* Recuperado el 07 de Sept. de 2010, de uklinux.net - the UK's free Linux ISP: http://www.smb.uklinux.net/reusability

Sun Microsystems, Inc. (2002). *Model-View-Controller*. Recuperado el 07 de Sept. de 2010, de Java Blueprints: http://java.sun.com/blueprints/patterns/MVC-detailed.html

Sun Microsystems, Inc. (s.f.). *Where can I get technical information about Java?* Recuperado el 8 de Nov. de 2010, de java.com: Java y Tú: http://www.java.com/en/download/faq/j2ee.xml

Szyperski, C. (2008). *Component-Based Software Engineering.* Karlsruhe, Alemania.
wordIQ.com. (s.f.). *Software component - Definition.* Recuperado el 7 de Sept. de 2010, de Dictionary, Encyclopedia and Thesaurus - WordIQ Dictionary: http://www.wordiq.com/definition/Software_component
Yuan, M. (1 de Jul. de 2007). *JBoss Seam: A Deep Integration Framework.* Recuperado el 7 de Sept. de 2010, de TheServerSide.com: Your Enterprise Java Community: http://www.theserverside.com/news/1364119/JBoss-Seam-A-Deep-Integration-Framework

Anexos

Anexo A: Diccionario de Datos

Alumno			
Contiene a los alumnos e información específica de ellos.			
Columna	**Tipo de Dato**	**Atributos**	**Descripción**
idAlumno	INTEGER	PK, FK	Llave primaria de la tabla, referencia a la tabla Persona
idPrograma	INTEGER	FK	Llave foránea que referencia al Programa al que pertenece el alumno
carne	CHAR(8)		Campo que guarda el carné del alumno
rutaFoto	VARCHAR(255)		Ruta al archivo que contiene la foto del alumno

Asignatura			
Contiene asignaturas e información de las asignaturas.			
Columna	**Tipo de Dato**	**Atributos**	**Descripción**
idAsignatura	INTEGER	PK	Llave primaria de la tabla
sigla	VARCHAR(5)		Sigla de la asignatura
nombre	VARCHAR(100)		Nombre de la asignatura
créditos	INTEGER		Créditos de la asignatura

Asistencia			
Guarda las asistencias de los alumnos a cada clase.			
Columna	**Tipo de Dato**	**Atributos**	**Descripción**
idAlumno	INTEGER	PK, FK	Es parte de la llave primaria compuesta de la tabla, referencia a la tabla Alumno
idClase	INTEGER	PK, FK	Es la otra parte de la llave primaria compuesta de la tabla, referencia a la tabla Clase
presente	BOOL		Campo que indica si el alumno estuvo presente o no en dicha clase

Aviso			
Contiene los avisos de las asignaturas.			
Columna	**Tipo de Dato**	**Atributos**	**Descripción**
idAviso	INTEGER	PK	Llave primaria de la tabla

idProfesor	INTEGER	FK	Llave foránea que hace referencia al Profesor que publicó el aviso
idAsignatura	INTEGER	FK	Llave foránea que hace referencia a la Asignatura en la que se publicó el aviso
titulo	VARCHAR(200)		Campo que tiene el título del aviso
contenido	TEXT		Campo que tiene el contenido del aviso
fecha	DATE		Fecha en la que se publicó el aviso

Clase			
Contiene las clases registradas dentro de las asignaturas.			
Columna	**Tipo de Dato**	**Atributos**	**Descripción**
idClase	INTEGER	PK	Llave primaria de la tabla
idAsignatura	INTEGER	FK	Llave foránea que hace referencia a la Asignatura a la que pertenece la clase
fecha	DATE		Campo que indica la fecha en que se dictará la clase

Evaluacion			
Contiene las evaluaciones de las asignaturas.			
Columna	**Tipo de Dato**	**Atributos**	**Descripción**
idEvaluacion	INTEGER	PK	Llave primaria de la tabla
idAsignatura	INTEGER	FK	Llave foránea que hace referencia a la Asignatura a la que pertenece la evaluación
nombre	VARCHAR(100)		Campo que contiene el nombre de la evaluación
descripcion	TEXT		Descripción de la evaluación
fecha	DATE		Fecha en que se tomará la evaluación
peso	INTEGER		Peso de la evaluación

EvaluacionAlumno			
Contiene las notas de los alumnos en las evaluaciones que han rendido.			
Columna	**Tipo de Dato**	**Atributos**	**Descripción**
idAlumno	INTEGER	PK, FK	Es parte de la llave primaria compuesta y a la vez es llave foránea, hace referencia al alumno al que pertenece la nota
idEvaluacion	INTEGER	PK, FK	Es la otra parte de la llave primaria compuesta, es también llave foránea y referencia a la Evaluación en la que se ha calificado al alumno
nota	INTEGER		Campo que guarda la nota del alumno en la evaluación

Material			
Contiene los materiales de cada asignatura.			
Columna	**Tipo de Dato**	**Atributos**	**Descripción**
idMaterial	INTEGER	PK	Llave primaria de la tabla
idProfesor	INTEGER	FK	Llave foránea que referencia al Profesor que publicó el material

idAsignatura	INTEGER	FK	Llave foránea que referencia a la Asignatura en la que se publicó
titulo	VARCHAR(200)		Campo que contiene el título del material
descripción	TEXT		Descripción del material
rutaArchivo	VARCHAR(255)		Ruta del archivo que contiene el material publicado
fecha	DATE		Fecha en que se publicó el material

Matricula			
La tabla contiene las matrículas de los alumnos, es decir qué alumno está matriculado en qué cursos.			
Columna	**Tipo de Dato**	**Atributos**	**Descripción**
idAsignatura	INTEGER	PK, FK	Campo que forma parte de la llave primaria compuesta, es también llave foránea y referencia a la Asignatura
idAlumno	INTEGER	PK, FK	Campo que forma parte de la llave primaria compuesta, es también llave foránea y referencia al Alumno

Persona			
Contiene la información de las personas del sistema. Esta tabla debe ser llenada antes de registrar a las personas como alumnos o profesores.			
Columna	**Tipo de Dato**	**Atributos**	**Descripción**
idPersona	INTEGER	PK	Llave primaria de la tabla
apellidoPaterno	VARCHAR(100)		Apellido paterno de la persona
apellidoMaterno	VARCHAR(100)		Apellido materno de la persona
nombres	VARCHAR(200)		Nombres de la persona
login	VARCHAR(100)		Nombre de usuario para ingresar al sistema
pass	CHAR(32)		Contraseña del usuario, encriptada en MD5[8]

Profesor			
Contiene la información de los profesores.			
Columna	**Tipo de Dato**	**Atributos**	**Descripción**
idProfesor	INTEGER	PK, FK	Llave primaria de la tabla, es también llave foránea y hace referencia a la tabla Persona
anexo	INTEGER		Anexo del profesor

ProfesorDicta			
Contiene la carga académica de los profesores, es decir, las asignaturas que dicta cada uno.			
Columna	**Tipo de Dato**	**Atributos**	**Descripción**
idAsignatura	INTEGER	PK, FK	Es parte de la llave primaria compuesta de la tabla, es también llave foránea y referencia a la Asignatura
idProfesor	INTEGER	PK, FK	Es la otra parte de la llave primaria

[8] Algoritmo unidireccional de encriptación utilizado comúnmente para el almacenamiento de contraseñas, una cadena encriptada MD5 siempre tendrá 32 caracteres.

			compuesta de la tabla, es llave foránea y referencia al Profesor

Programa			
Contiene los programas académicos de la universidad			
Columna	**Tipo de Dato**	**Atributos**	**Descripción**
idPrograma	INTEGER	PK	Llave primaria de la tabla
sigla	VARCHAR(5)		Sigla del programa académico
nombre	VARCHAR(100)		Nombre del programa académico