**Documento de Arquitectura**

**1. Visión del sistema**

A definir

**2. Requerimientos que afectan a la arquitectura**

A definir

**2.1 Funcionalidades típicas de la aplicación**

A definir

**2.2 Priorización de requerimientos no funcionales**

A definir

**3. Arquitectura de aplicación**

La arquitectura a utilizar será orientada a tiers. El modelo de dominio representará un tier, la persistencia otro, también los servicios, y así.

**3.1 Módulos del sistema (dependencias)**

A definir

**3.2 Vista de alto nivel de la arquitectura**

A definir

**3.3 Patrones de diseño implementados dentro del proyecto**

A definir

**4. Tecnologías y criterios de diseño utilizados**

**4.1 Capa de presentación**

La tecnología elegida para la presentación será Wicket en su versión 1.4.17.

**¿Que es Wicket?**

Wicket es un Framework para el desarrollo de aplicaciones Web en

Java y se basa únicamente en Java y HTML. Básicamente está orientado

a componentes y el manejo de eventos dentro de una aplicación Web. Es

una forma de desarrollo web muy similar a crear una aplicación de escritorio,

es decir, como si se usara AWT o SWING. Apache Wicket es un

Frameworks de desarrollo Web muy prometedor ya que separa claramente

vista y lógica. Todo el comportamiento de la página estará programado en

Java

**¿Por qué Wicket?**

Si se busca un lenguaje para desarrollar Web en java, existen muchas posi-

bles elecciones hoy en dıa, existen muchos Framework Web de java. Cuantos

Frameworks de java existen, la respuesta se presenta a continuación:

Echo

Cocoon

Millstone

Struts

SOFIA

Tapestry

RIFE

Spring MVC

Canyamo

JPublish

JATO

Folium

Verge

Niggle

Bishop

Action

Framework Shocks

TeaServlet

Expresso

Bento

jStatemachine

OpenEmcee

Turbine

Scope

JWAA

Jaffa

Jacquard

Smile

MyFaces

Chiba

Jeenius

JWarp

Genie

Dovetail

Cameleon

JFormular

Japple

Helma

Dinamica

Nacho

Cassandra

Baritus

Click

GWT

OXF

WebWork

Maverick

Jucas

Barracuda

wingS

jZonic

Warfare

Macaw

JBanana

Melati

Xoplon

WebOnSwing

Stripes

Viendo todas estas soluciones, ¿De qué sirve otra solución java? Wicket

surge para mejorar los Framework de java que ya existen, y ofrecer mucho

más que de lo que pueden ofrecer otros Framework. Los Frameworks más

cercanos a Wicket son probablemente Tapestry y Echo, pero incluso ahı la

semejanza es muy poco profunda:

Como Tapestry, Wicket utiliza un atributo HTML especial para

designar los componentes, lo que permite una fácil edición ordinaria

con editores de HTML.

Al igual que Echo, Wicket tiene un modelo de componentes de

primera clase.

Aun ası, las aplicaciones Wicket no son como aplicaciones escritas en Tapestry o Echo, Wicket incorpora lo mejor de ambos mundos. Se obtienen los beneficios de un modelo de componentes de primera clase y un enfoque no intrusivo en HTML. En muchas situaciones, esta combinación puede resultar ser una importante ventaja en el desarrollo.

**Motivación**

Para entender por qué Wicket es diferente, se intentarán explicar las motivaciones que llevaron a su desarrollo.

1. La mayoría de los Framework prevén débilmente o inexistente

un soporte para la gestión en el lado del servidor.

Esto normalmente significa gran cantidad de código ad-hoc en las aplicaciones Web repartiendo la administración del servidor. Mientras Wicket no permitirá dejar de pensar en el estado del servidor.

En Wicket, todo el estado del servidor es gestionado de forma automática. Nunca se utilizará directamente el objeto HttpSession u objetos similares de estado. Cada componente página tiene una jerarquıa de componentes con estado, donde el modelo de cada componente es, en definitiva, un POJO (Plain Old Java Object). Wicket mantiene un mapa de estas páginas en la sesión de cada usuario.

El programador trata con objetos simples, familiares Java y Wicket lidia con cosas como URL, ID de sesión y peticiones GET / POST, facilitando su uso.

Wicket también proporciona una solución al problema de “Botón de Volver”, tiene una solución genérica y robusta que permite identificar y hacer expirar las páginas que se han vuelto obsoletas debido a los cambios estructurales del modelo de un componente en la página.

Por último, Wicket ha sido diseñado para trabajar con Frameworks de persistencia de Objetos como JDO o Hibernate. Esto puede hacer que las aplicaciones Web con bases de datos sean muy fáciles de implementar.

En términos de eficiencia frente a la productividad, tal vez es Wicket para JSP como Java es para C. Se puede lograr cualquier cosa usando Wicket, incluso hacer que sea más eficiente en términos de memoria o consumo del procesador. Sin embargo, el tiempo de desarrollo es más largo

1. Existen más Frameworks que requieren código HTML especial

JSP es el que peor oferta ofrece en cuanto a inserción de código especial en HTML, pero hasta cierto punto casi todos los Frameworks anteriormente mencionados (a excepción de Tapestry) introducen algún tipo de sintaxis especial al código HTML. La sintaxis añadida es poco recomendable, ya que cambia la naturaleza del HTML, lo que hace más complicado el mantenimiento.

Wicket no añade ninguna sintaxis a HTML. En su lugar, se extiende HTML de una manera compatible con los estándares a través de un namespace que incorpora Wicket, totalmente compatible con el estándar XHTML. Esto significa que se puede usar Macromedia Dreamweaver, Microsoft Front Page, Word, Adobe Live Id, o cualquier otro editor HTML existente para trabajar en páginas Web y componentes Wicket.

Para lograr esto, Wicket utiliza un atributo, un id único, wicket:id, para marcar las etiquetas HTML que deben recibir un trato especial por parte del kit de herramientas de Wicket.

No existe nada adicional en el código HTML, lo que significa que los diseñadores pueden maquetar páginas que se pueden utilizar directamente en el desarrollo. Añadir componentes Java en el código HTML es tan sencillo como establecer el atributo nombre del componente. Del mismo modo, los programadores pueden trabajar en los componentes

de Java que se conectan al HTML sin preocuparse por el diseño que ellos puedan añadir a la Web, se pueden centrar en la funcionalidad

1. Existen frameworks complejos

La mayorıa de las herramientas existentes tienen pobres definiciones o inexistentes del modelo de datos. En algunos casos, el modelo se define utilizando sintaxis XML especial. La sintaxis puede ser tan complicada que se requieren herramientas especiales para manipular toda la información de configuración. Dado que estas herramientas no son simples bibliotecas Java, pueden no ser utilizas en las herramientas de desarrollos que normalmente se utilizan.

Wicket es todo simplicidad. No hay archivos de configuración para aprender en Wicket. Wicket incorpora una biblioteca de clases simples con un enfoque coherente a la estructura de sus componentes. En Wicket, las aplicaciones Web se parecerán más a una aplicación Swing que una aplicación JSP. Si se conoce Java (y especialmente Swing), ya se conoce mucho acerca de Wicket.

Tapestry y JSF tienen modelos de componentes que permiten la reutilización, pero no es particularmente trivial su uso, al menos en comparación con Wicket. Wicket ha sido expresamente diseñado para que sea muy fácil la creación de componentes reutilizables. Es muy simple extender los componentes existentes y crear componentes nuevos.

**4.2 Capa de negocio**

Se utilizarán objetos Java planos para modelar el dominio.

<Diagrama>

**4.3 Capa de acceso a datos**

La siguiente capa en nuestra arquitectura es la capa de mapeo objeto – relacional

Esta capa surge de la necesidad de tener un nivel de mayor abstracción del manejo de los datos. Teniendo esta capa, se podrá encapsular al usuario del diseño, implementación y uso directo de la base de datos, lo que hace que se puedan simplificar muchas etapas del proceso de desarrollo.

Además al utilizar una herramienta que nos permita realizar mapeos de este tipo,

se podrá establecer una clara separación del paradigma relacional (bases de datos) y el paradigma orientado a objetos (clases), lo cual nos permitirá tener un mayor grado de definición para cada una de las capas de la arquitectura.

Para nuestra aplicación, hemos decidido utilizar Hibernate, en su versión 3.6.4, para representar el acceso a la persistencia.

**¿Por qué Hibernate?**

1. Es open source.
2. Es un framework maduro, ya que es uno de los mas utilizados actualmente con muy buenos resultados.
3. Hibernate da un completo soporte al modelo de programación orientado a objetos, lo cual es una ventaja en el desarrollo de este proyecto ya que este se hará sobre JAVA.
4. Ofrece un lenguaje natural para la búsquedas en la base de datos (HSQL) que es muy similar al que hemos manejado(SQL).
5. Maneja XML para los mapeos, lo que hace que estos sean de fácil entendimiento por la estructura que este maneja.

**4.3.1 Soporte de transacciones**

El sistema soporta transacciones vía Spring combinado con programación orientada a aspectos (AOP).

**4.5 Servicios**

Los Servicios fueron implementados con Spring.

**Spring**

**Introducción**

Spring Framework (también conocido como Spring) es un Framework de código abierto de desarrollo de aplicaciones para la plataforma Java.

A pesar de que Spring Framework no obliga a usar un modelo de programación en particular, se ha popularizado usarlo junto a Java al considerarse una alternativa y sustituto del modelo de Enterprise JavaBean.

**¿Qué es Spring?**

La motivación inicial era facilitar el desarrollo de aplicaciones J2EE, promoviendo buenas prácticas de diceño y programación. Se enfoca al manejo de objetos de negocio, dentro de una arquitectura en capas. Además una de sus mayores ventajas es su modularidad, pudiendo utilizar algunos de sus módulos sin comprometerse con el uso del resto.

Uno de los principales objetivos de Spring es no ser intrusivo, es decir, las aplicaciones están configuradas para utilizar Beans mediante Spring, no necesitan depender de interfaces o clases de Spring, obteniendo su configuración a través de las propiedades de sus Beans. Este concepto puede ser aplicado a cualquier entorno, desde una aplicación J2EE hasta un applet.

Una caracterıstica de Spring es que puede conseguir la integración entre diferentes APIs (JDBC, JNDI, etc.) y Frameworks (por ejemplo entre Structs e iBatis).

Spring está compuesto por un conjunto caracterısticas, las cuales están agrupadas en seis módulos principales:

1. El módulo Core o “Núcleo” es la parte fundamental del Framework ya que provee toda la funcionalidad de Inyección de Dependencias permitiéndo administrar la funcionalidad del contenedor de beans. Esto significa que la creación de los objetos lo lleva a cabo un contenedor externo inyectándolos a otros objetos que dependan de los primeros.
2. Encima del módulo core se encuentra el módulo Context (Contexto), el cual provee de herramientas para acceder a los beans de una manera elegante, similar a un registro JNDI.
3. El paquete DAO provee una capa de abstracción de JDBC que elimina la necesidad de teclear código JDBC. También provee un mecanismo de administración de transacciones tanto declarativas como programáticas, no solo para clases que implementen interfaces especiales, sino para todos los POJOs.
4. El paquete ORM provee capas de integración para APIs de mapeo objeto relacional, incluyendo JDO, Hibernate e iBatis.
5. El paquete AOP provee una implementación de programación orientada a aspectos compatible con AOP Alliance. Con ello se quiere desacoplar el código de una manera limpia implementando funcionalidad que por lógica y claridad deberıa estar separada.
6. El paquete Web provee caracterısticas básicas de integración orientadas a la Web. El paquete Web MVC provee de una implementación Modelo-Vista-Controlador para las aplicaciones Web. La implementación de Spring MVC permite una separación entre código y la interfaz Web, permitiendo además el uso de otras caracterısticas de Spring Framework como lo es la validación

**4.5 Seguridad**

A definir

**4.6 Estrategia de manejo de errores**

A definir

**4.7 Tratamiento de transacciones**

A definir

**4.8 Logging general**

A definir

**5. Estrategias de test**

Se realizarán pruebas unitarias del modelo de dominio, usando JUnit.

**6. Solución para las funcionalidades típicas del sistema**

A definir

**6.1 Funcionalidad típica 1: ????**

A definir

**6.2 Funcionalidad típica 2: ????**

A definir

**7. Diseño del modelo**

A definir

**8. Lineamientos y estandares de codificación**

A definir