# GENERACIÓN DE NUEVAS ARQUITECTURAS BASADAS EN EJB E INTEGRACIÓN CON LA HERRAMIENTA DE ADMINISTRACIÓN DE PROYETOS MAVEN PARA EL GENERADOR DE COMPONENTES DE SOFTWARE ZATHURACODE

**ANDRÉS FELIPE PUERTA SIMBAQUEBA**

**STIVEN ORLANDO GARCÍA MORENO**

**UNIVERSIDAD DE SAN BUENAVENTURA**

**FACULTAD DE INGENIERÍA**

**PROGRAMA DE INGENIERÍA DE SISTEMAS**

**SANTIAGO DE CALI, 2014**

# GENERACIÓN DE NUEVAS ARQUITECTURAS BASADAS EN EJB E INTEGRACIÓN CON LA HERRAMIENTA DE ADMINISTRACIÓN DE PROYETOS MAVEN PARA EL GENERADOR DE COMPONENTES DE SOFTWARE ZATHURACODE

**ANDRÉS FELIPE PUERTA SIMBAQUEBA**

**STIVEN ORLANDO GARCÍA MORENO**

Informe de Investigación presentado

Para optar por el Título de

Ingeniero de Sistemas

Director

DIEGO ARMANDO GÓMEZ MOSQUERA

Ingeniero de Sistemas

**UNIVERSIDAD DE SAN BUENAVENTURA**

**FACULTAD DE INGENIERÍA**

**PROGRAMA DE INGENIERÍA DE SISTEMAS**

**SANTIAGO DE CALI, 2014**

# AGRADECIMIENTOS

Agradecemos a DIOS primero que todo por darnos la oportunidad de realizar estudios profesionales, por permitirnos llegar con éxito a este punto donde se culmina una etapa más de nuestras vidas.

A nuestras familias que con su apoyo, esfuerzo, dedicación y amor nos guiaron, aportaron y contribuyeron durante nuestro proceso para poder llegar a esta realidad que comenzó como un sueño.

A los profesores que nos colaboraron en el transcurso de la carrera, a nuestro director el Ing. Diego Armando Gómez alguien fundamental para el desarrollo de esta tesis, ya que gracias a su alto conocimiento en el ámbito del desarrollo de software, nos apoyó y guio para la finalización de este trabajo y por último a todos aquellos compañeros que de una u otra forma participaron durante todo el proceso universitario.

## CONTENIDO

## ILUSTRACIONES



## DOCUMENTOS

# 1. INTRODUCCIÓN

En el proceso de construcción de aplicaciones Web, las variables más representativas que se tienen en cuenta cuando se está desarrollando un contrato son calidad, tiempo y costo.

Para cumplir con estas expectativas fue desarrollada una herramienta generadora de componentes de software llamada Zathuracode. Actualmente el plugin es usado por varias empresas o compañías de Santiago de Cali, empresas que han conocido el producto y aprovechado sus características y beneficios que este ofrece, algunas de ellas son: CIAT (Centro Internacional de Agricultura Tropical), Codesa, Geniar S.A, MAC-Johnson Controls Colombia S.A.S entre muchas otras[1].

Zathuracode es un generador de código para apoyar el desarrollo de aplicaciones empresariales en JavaEE[2], a partir de un modelo de base de datos existente. La idea de construir este plugin surge del Ingeniero Diego Armando Gómez Mosquera, que junto a un grupo de Ingenieros inicio la implementación del mecanismo de generación de componentes de software Java.

El grupo administrador y desarrollador del plugin se ha interesado cada día en mejorar y hacer de este una herramienta robusta que supla las necesidades de muchas de las empresas y casas desarrolladoras de software que se dedican a la generación de aplicaciones web basadas en la plataforma Java, por tal razón en los últimos años se han venido generando versiones con nuevas arquitecturas, solución de bug`s, presentadas en las versiones anteriores, inclusión de nuevos framework`s, actualización de versiones de cada uno de los componentes visuales y de los motores de persistencia, tales como Hibernate y JPA.

A finales del segundo semestre del año 2008 surge la primera versión beta de Zathuracode 0.1. A partir de este primer lanzamiento durante el año siguiente se desarrollaron versiones beta para la implementación en el IDE[3] Eclipse.

---

[1] Más información de empresas que lo utilizan http://zathuracode.org/quien-lo-utiliza/

[2] JavaEE: Java Enterprise Edition, plataforma para el desarrollo y ejecución de programas Java.

[3] Entorno de desarrollo integrado, por sus siglas en inglés: Integrated Development Environment.

El 31 de Agosto de 2009 se lanza la versión oficial de Zathura para Eclipse 3.5 JEE Galileo. Esta versión comprendió la solución de 10 errores, integración del componente visual Icefaces 1.8.1 y la compatibilidad con Facelets, siendo estos tres puntos lo más destacado de la versión.

En el cuarto trimestre de 2010 nace Zathuracode 2.1.0, versión que tiene como principales características la compatibilidad con el IDE MyEclipse 8.5, además de la herramienta de mapeo de base de datos**.**

En los inicios del año 2011 se publica la versión de Zathuracode 2.1.1, versión que contaba con tres nuevas arquitecturas, *JavaEE Spring Hibernate Core Web Centry, JavaEE GWT JPA Web Centry* y *JavaEE HibernateCore Web Centry,* además de las nuevas arquitecturas, la nueva versión es compatible con el IDE Eclipse Indigo 3.7.

Para el segundo trimestre del año 2012 se lanza la versión Zathuracode 3.0.0 en la cual se implementan nueve arquitecturas en las que para cada una de ellas su principal característica es la compatibilidad con el componente visual Primefaces.

En su versión actual Zathuracode 3.0.0, la cual cuenta con varias arquitecturas y tecnologías que han permitido a las casas de desarrollo de software ahorrar tiempo en la implementación de código fuente. Las arquitecturas, los Framework[4], las tecnologías y componentes con las que cuenta esta herramienta, hacen que este sea un software de gran beneficio para las empresas que ofrecen soluciones con el desarrollo de aplicaciones para las distintas problemáticas que se presentan al momento de elegir una arquitectura de software.

Zathuracode 3.0 aunque es una herramienta robusta y con pocos errores debe continuar en su evolución para seguir supliendo las necesidades de calidad, tiempo y costo pero apoyado en recientes versiones de los framework`s y la inclusión de las nuevas tendencias tecnológicas que ayudan al desarrollo de aplicaciones web.

En esta oportunidad se presentará una nueva versión de Zathuracode, versión en la cual se incluye una arquitectura basada en la API[5] EJB[6] con las anotaciones que ella requiere, es una arquitectura orientada a transacciones la cual permite simplificar gran parte del

[4] Framework: Es un marco de trabajo para el desarrollo y/o la implementación de una aplicación.

[5] Application Programming Interface.

[6] Enterprise Java Beans. Para más información ver capítulo 7.

desarrollo logrando que los artefactos encargados de la gestión de la aplicación controlen dichas transacciones y los parámetros de seguridad, de igual manera la inclusión del administrador de proyectos software MAVEN**,** con todas sus características, por ultimo aplicara para este proyecto la implementación de un DAO[7] genérico para la gestión de mantenimiento de las tablas de una base de datos, que como prioridad es hacer uso de la codificación implementada para una tabla y hacer uso de ella para todas las tablas contenidas en el desarrollo. Esperamos que esta nueva versión Zathuracode cumpla con las expectativas y sea una herramienta mucho más completa para la generación de diversas aplicaciones de software a la medida.

[7] Data Access Object (DAO, Objeto de Acceso a Datos).

# 2. DEFINICIÓN DEL PROBLEMA

La evolución en las buenas practicas que se ejercen al momento de generar código son bastante exigentes y requieren un nivel elevado de análisis y conocimiento por parte de los arquitectos de software, que cada día se concentran más en lo que se llama reglas de negocio, es decir las definiciones específicas que hacen diferente la idea de este, las culés son vitales para alcanzar el objetivo del negocio, ya que es ahí donde se tiene el valor agregado de cada una de las aplicaciones que surgen con la finalidad de brindar una solución.

Partiendo del hecho que Zathuracode es una herramienta desarrollada bajo la plataforma Java y su distribución se realiza bajo la licencia Apache 2.0, licencia que nos garantiza que esto nunca tendrá costo alguno. El hecho que la herramienta sea libre permite que se encuentre al alcance de muchos desarrolladores y su uso se haga extensivo, logrando que cada uno de los individuos interactúe, conozcan, aprendan y realicen sus aportes acerca del funcionamiento, identificando beneficios, características, falencias y oportunidades de mejora para el plugin.

Las oportunidades de mejora se basan específicamente en los cambios tan repentinos y ligeros que se presentan en el ámbito tecnológico en un periodo de tiempo muy corto. Estos cambios se pueden evidenciar en ramas tales como: Los aparatos inteligentes, la telefonía celular y sus sistemas operativos, dispositivos móviles entre muchas alternativas que surgen cada día, las cuales exigen que los sistemas de información y aplicaciones se encuentren a la vanguardia.

Reconociendo que Zathuracode es un generador de código en crecimiento y bien fundamentado sobre una arquitectura MVC[8], esta debe actualizar sus componentes y tecnologías que la constituyen con el fin de mantenerse vigente y seguir siendo una alternativa para los diferentes grupos desarrolladores de software que generan sus aplicaciones en JavaEE, además de poder ser una herramienta que aplica buenas practicas del desarrollo formal de software, permitiendo dar un manejo a nivel de código fuente bastante entendible y claro para todo aquel desarrollador que requiera interpretar la codificación de un proyecto generado por medio del plugin Zathuracode.

[8] MVC: Patrón de diseño por sus siglas (Modelo Vista Control).

Es importante saber que en la versión más reciente es la del año 2012 y desde el momento en que surge su primera versión no se ha hecho una actualización general del plugin que entre otras cosas llegue a tocar su aspecto visual, aun así el software se ha mantenido de buena manera cumpliendo con cada uno de los requisitos planteados al momento de generar código con base a la arquitectura y la fuente de datos seleccionada por el usuario final limitándolo a una serie de alternativas que no cuentan con la versión más reciente de las tecnologías aplicadas por Zathuracode.

Recopilando información de los principales consumidores y creadores de la herramienta coinciden en que los Framework`s, tecnologías y componentes actualmente, se encuentran en versiones superiores en comparación a las que se encuentran en el generador de componentes de software hoy en día. De igual manera Zathuracode en una herramienta consolidada, es así como surge nuestro interés por aportar al generador de código y brindar la posibilidad de ampliar el catálogo de arquitecturas disponibles en el plugin y consolidar cada vez más un producto software que se encuentra disponible para toda la comunidad desarrolladora de software.

# 3. JUSTIFICACIÓN

Dada la importancia que tiene el generador de componentes de software Zathuracode en el mercado del desarrollo de software, permitiendo suplir las necesidades de calidad, tiempo y costo al momento de la construcción de aplicaciones, haciendo que la creación de los componentes encargados del CRUD[9] sea de manera más dinámica y que el desarrollo de la lógica de negocio sea el enfoque principal de las empresas que lo utilizan como CIAT, Codesa, Vortexbird S.A.S., Geniar S.A.S., etc…

Es de vital importancia la integración de nuevas versiones de framework's, de API´s, de componentes visuales, de herramientas para la generación del mapeo objeto relacional, incluso la generación de nuevas arquitecturas con características que cumplan con las necesidades que estas empresas requieren para seguir con la continua evolución de sus aplicaciones JavaEE.

A medida que las aplicaciones JavaEE evolucionan implementando nuevas tecnologías que ofrecen ventajas y beneficios en el entorno de programación y otorgan un continuo mejoramiento en el desarrollo de proyectos de software, Es relevante que Zathuracode incluya estas tecnologías para seguir siendo una herramienta que facilite la construcción de estas aplicaciones. Por lo cual para la versión 4.0 integra una herramienta administradora de proyectos como MAVEN, una API que se encarga de manejar las transacciones, concurrencia, seguridad entre otros beneficios como son los EJB y la actualización de versiones de Spring framework y la suite de componentes visuales Primefaces.

Incluir un administrador de proyectos de desarrollo de software como MAVEN facilita la compilación, el empaquetamiento y la generación de componentes de despliegue de la aplicación, así como también las librerías a usar por parte de la aplicación, ya que no estarán integradas en el proyecto, sino que se basa en un archivo de configuración de dependencias de librerías.

Con la especificación EJB se otorga al contenedor de aplicaciones la administración de la apertura y cierre de la transacción, la seguridad de accesos a los métodos del bean, la

[9] CRUD (Create Read Update Delete) Acrónimo adoptado por la industria de software para referirse a las operaciones básicas de toda aplicación (crear, consultar, actualizar y borrar).

llamada simultánea al bean desde el cliente, sincronización entre la información de la tabla en base de datos y el bean de la aplicación.

Actualizar la versión de Spring Framework permite la administración de seguridad configurando el acceso a las páginas y la autenticación en el logueo.

Se añaden mejoras a los componentes para la interfaz gráfica, con la versión 3.5 de Primefaces.

## 4. OBJETIVO GENERAL

Actualizar la versión del generador de componentes de software Zathuracode actualmente en su versión 3.0 a su versión 4.0 adicionando 3 nuevas arquitecturas y la integración con la herramienta MAVEN de cada una de ellas.

## OBJETIVOS ESPECÍFICOS

- Diseñar y construir los componentes necesarios, para que las aplicaciones construidas por Zathuracode, permitan el uso de la herramienta MAVEN 3.1 como el administrador de dependencias y constructor de artefactos para el despliegue.
- Diseñar y construir un componente de acceso a datos genérico para que las aplicaciones generadas por Zathuracode, en su capa de acceso a datos realicen los métodos CRUD a través de una clase genérica.
- Diseñar y construir los componentes de software necesarios, para que las aplicaciones construidas por Zathuracode, en su capa de presentación use la suite de componentes visuales Primefaces 3.5.
- Diseñar y construir los componentes de software necesarios, para que las aplicaciones construidas por Zathuracode, integren el framework Spring 3.2.3 con la suite de componentes visuales Primefaces 3.5.
- Diseñar y construir los componentes de software necesarios, para que las aplicaciones construidas por Zathuracode, integren el API Enterprise Java Beans (EJB) 3.1 con la suite de componentes visuales Primefaces 3.5.

# 5. ALCANCE DEL PROYECTO

Los siguientes son los ítems que el proyecto contemplará:

1. Generar artefactos de software desde la herramienta Zathuracode que se integren con el Framework SPRING en la versión 3.2.3.
2. Generar artefactos de software desde la herramienta Zathuracode que se integren con la especificación EJB de JAVA EE en la versión 3.1.
3. Generar artefactos de software desde la herramienta Zathuracode que se integren con la herramienta de administración de proyectos MAVEN en la versión 3.1.0.
4. Generar artefactos de software desde la herramienta Zathuracode que usen la suite de componentes visuales PRIMEFACES en la versión 3.5.
5. Refinar e implementar mecanismos, que permitan corregir bug´s encontrados en Zathuracode.

Los siguientes son los ítems que el proyecto no contemplará:

1. Construir un ambiente integrado de desarrollo (IDE) para la construcción de aplicaciones.
2. Generar un proyecto web con un arquetipo basado en MAVEN para el desarrollo de aplicaciones.

# 6. ESTADO DEL ARTE

Las fábricas de desarrollo de software hoy día buscan la manera de optimizar el tiempo y esfuerzo en codificación que indiferente de su entorno siempre se ocupan de una tarea fundamental, que consta de la interacción con un modelo de base de datos y su gestión sobre la misma, aquella gestión permite la inserción, captura, actualización y eliminación de datos con el fin de dar soluciones a problemas o administrar información para una idea de negocio, lo que ha permitido que los desarrolladores de software intenten organizar el código de sus programas, dando lugar así a que se generen los paradigmas de programación, que en nuestro mundo el más usado es la POO[10] .

Así como se menciona en un artículo publicado por la casa de desarrollo Ssquare-SA[11] en la que afirma, *Un generador de código fuente al interior de una constructora de software permite generar automáticamente aquellos fragmentos de código que pueden repetirse más de una vez en las iteraciones de construcción de un sistema y en proyectos diferentes de software.*

En este capítulo se describen importantes herramientas de generación de código existentes, abordaremos con mayor profundidad el plugin Zathuracode Generator.

## OPENXAVA

Es un marco de trabajo para desarrollo rápido de aplicaciones de gestión con Java, el código de la aplicación se estructura desde un punto de vista orientado a objetos puro. El enfoque de OpenXava es dirigido por el modelo (model-driven), donde el desarrollo o la generación del código se centran en las clases Java de los modelos descritos por el desarrollador.

[10] POO: Programación Orientada a Objetos. Véase http://www.desarrolloweb.com/articulos/499.php

[11] Es una empresa del sector de las TICs, especializada en el desarrollo de software a la medida y en la fabricación y comercialización de productos que apoyan al personal directivo en el cumplimiento de la gestión estratégica de las organizaciones.

**SPRINGFUSE**

Al igual que el anterior este es un generador de código de origen francés, el cual se encuentra al servicio desde el año 2005. Springfuse se construye con base al lenguaje de programación Java y permite la generación de proyectos a partir de un modelo de datos seleccionado por el desarrollador o gestor de la nueva aplicación. La generación de los proyectos por medio de esta herramienta se realiza desde la web, retornándole al cliente el paquete del proyecto codificado.

## SQL2JAVA

Esta es una herramienta generadora de código fuente gratuita basada en Java, la cual permite a los desarrolladores asignar un esquema de base de datos relacional a un conjunto de clases.

Lenguaje de generación de código:

- Java

Motores de bases de datos:

- Oracle
- MYSQL
- HSQL
- Cualquier motor de base de datos que soporte JDBC

Plataforma de ejecución:

- Compatible con JVM

Creador:

- Desarrollo conjunto de código abierto.

## ZATHURACODE 3.0.0

Zathruacode es una herramienta desarrollada en Cali, Colombia por un grupo de desarrollo patrocinado por la compañía Vortexbird, el plugin ya cuenta con 6 años de vigencia disponible en la web para los desarrolladores de software que se basan en la plataforma JavaEE; su objetivo se centra en el ahorro de tiempo y costo y aporte de calidad en el desarrollo de los productos software. El paradigma de Zathuracode consiste en omitir al desarrollador la codificación de los casos de uso CRUD, el mapeo objeto-relacional y la estructura de las capas de la arquitectura seleccionada.

La herramienta Zathuracode genera el código fuente de aquellos fragmentos de código que llegan a repetirse en cada proyecto o aplicación que se decide construir, además permitiendo al desarrollador elegir en que arquitectura ejecutar el proceso de implementación o la fase de desarrollo del software.

Retomando la investigación realizada y del desarrollo de la versión 3.0.0 de Zathuracode descrita en el documento *Generación de nuevas arquitecturas basadas en Primefaces y Spring Framework para el generador de componentes de software Zathuracode*. Se constituye a partir de Templates, los cuales contienen la estructura básica para cada arquitectura que se implementara a partir de un modelo de base datos proporcionado por el usuario. Es preciso mencionar que el generador de código se apoya en diferentes framwork y tecnologías, las cuales se encargan de una u otra manera ofrecer al generador útiles herramientas las cuales permiten que se realice un buen procesamiento y escritura del código fuente resultante.

## Tecnologías Involucradas en Zathuracode

Actualmente existe una gran variedad de arquitecturas que se pueden implementar con Zathuracode para la generación de aplicaciones JavaEE; a continuación se enumeran las arquitecturas y tecnologías que se implementan por medio de esta herramienta.

Las arquitecturas generadas por Zathuracode son:

- Arquitectura de Zathuracode, Generador de componentes de software para aplicaciones JavaEE Web Centric.
- Arquitectura de Zathuracode para aplicaciones JavaEE SmartGWT Centric.
- Arquitectura de Zathuracode para aplicaciones JavaEE Primefaces Web Centric.
- Arquitectura de Zathuracode para aplicaciones JavaEE Primefaces Hibernate Centric.
- Arquitectura de Zathuracode para aplicaciones JavaEE Primefaces Jpa Spring.
- Arquitectura de Zathuracode para aplicaciones JavaEE Primefaces Hibernate Spring.
- Arquitectura de Zathuracode para aplicaciones JavaEE Icefaces JPA Spring.

Los motores de bases de datos soportados son:

- Oracle
- MySql
- Postgresql

El código se genera en el lenguaje de desarrollo:

- Java

Las aplicaciones web pueden ejecutarse en:

- Contenedor de aplicaciones Tomcat
- Servidor de aplicaciones GlassFish
- Servidor de aplicaciones JBoss

# 7. MARCO TEÓRICO

## ESTRUCTURA DE LOS ENTERPRISE JAVA BEANS (EJB)

Los EJB aparecen por primera vez en 1998, con la versión 1.0 la cual es la especificación original, desde sus inicio los EJB basan su configuración en ficheros con extensión XML, sin embargo para aquella oportunidad los EJB no hacían parte del estándar J2EE, sino que solo hasta su segunda versión se incluyó la versión 1.1 de los EJB al estándar, durante diez años los EJB han ido evolucionando hasta el año 2009 en la que surge la versión 3.1, de igual manera la especificación conserva sus bases y fundamentos.

Como se menciona previamente el manejo y configuración de los EJB se basa en ficheros XML[12], sin embargo la última versión completa de los EJB es la 3.1, esta versión se caracteriza porque permite ser controlada por medio de **anotaciones.**

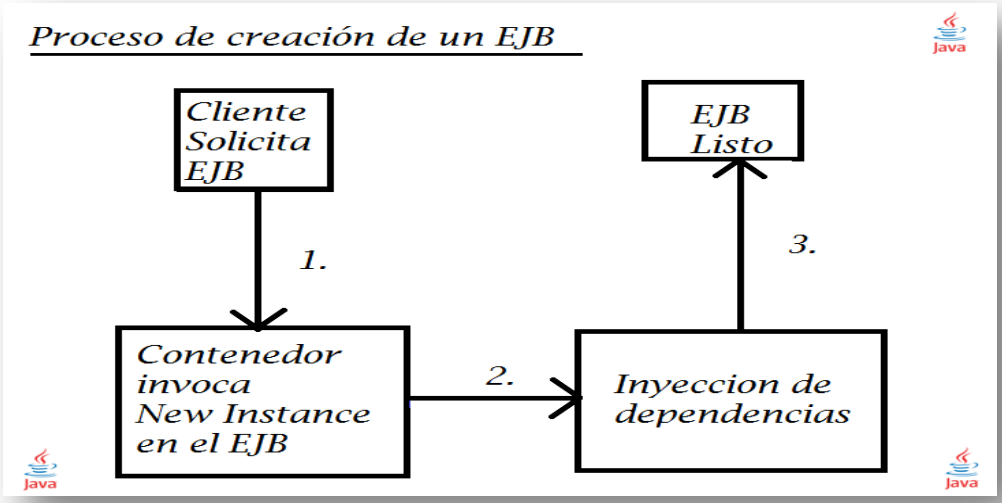


**Ilustración 1 Gráfica de proceso de creación de un EJB**

[12] Extensible Markup Language

Existen 3 tipos de EJB, cada uno posee distintas características, a continuación mencionaremos cada uno de ellos y una breve descripción para cada uno:

**EJB de Entidad**

Un bean de entidad representa un objeto de negocio en un mecanismo de almacenamiento persistente. Algunos ejemplos de objetos de negocio son los clientes, pedidos y productos. En el servidor de aplicaciones, el mecanismo de almacenamiento persistente es una base de datos relacional. Normalmente, cada bean de entidad tiene una tabla subyacente en una base de datos relacional, y cada instancia del bean corresponde a una fila de esa tabla.

**EJB de Sesión**

Los bean de sesión gestionan el flujo de la información en el servidor, estos sirven a los **clientes** como una fachada de los servicios proporcionados por otros componentes disponibles en el servidor, su principal característica es encapsular los procesos de negocio, los bean de sesión se clasifican en dos tipos los cuales son:

- **STATEFUL:** Son los objetos distribuidos con estado, es decir, el estado no es persistente, pero el acceso al bean se limita a un solo cliente o a una única instancia. Los Stateful se declaran por medio de la anotación **@Stateful** a nivel de la clase.

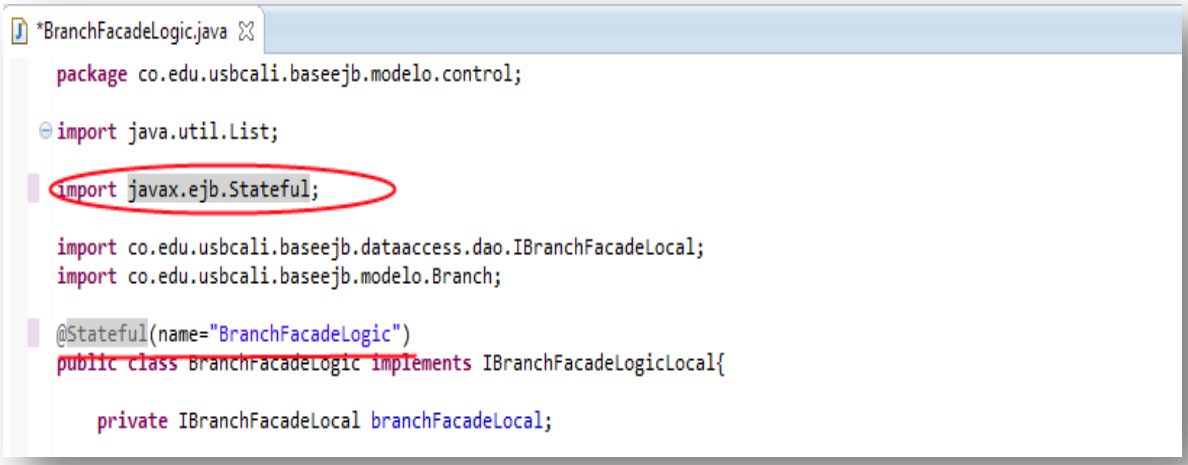


**Ilustración 2 Vista de desarrollo de un componente Stateful**

La ilustración previa nos muestra la declaración de un Stateful a nivel de código fuente, donde simplemente importamos la librería de EJB y firmamos la clase con su respectiva anotación.

- **STATELESS:** Los bean de sesión sin estado son objetos distribuidos que carecen de estado, permitiendo que se accedan concurrentemente, no es garantía que los contenidos de las variables de instancia se conserven entre llamadas al método, es decir, el componente no está asociado al usuario. Los Stateless se declaran por medio de la anotación **@Stateless** a nivel de clase.

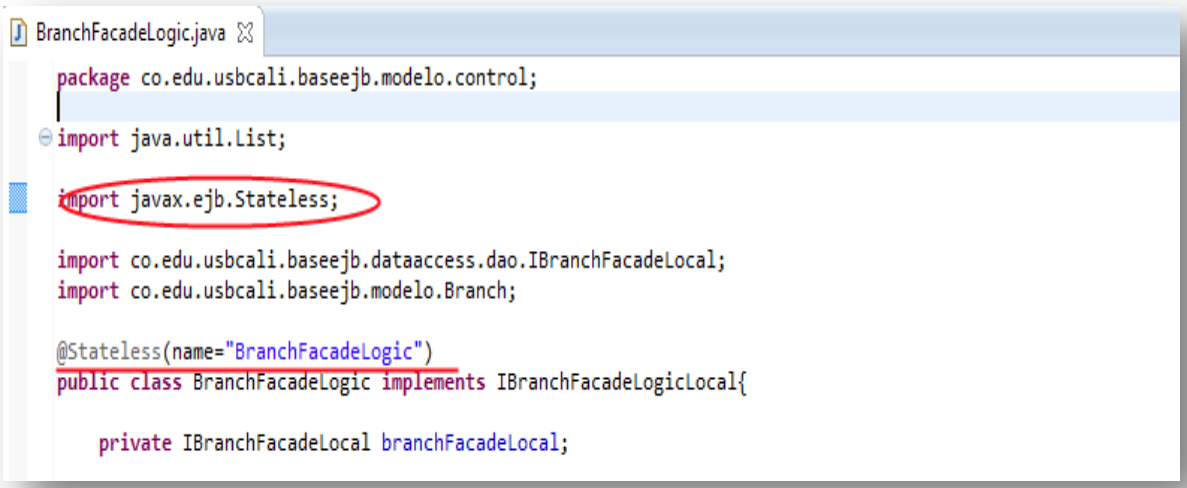


**Ilustración 3 Vista de desarrollo de un componente Stateless**

- **SINGLETON:** Es un nuevo tipo de bean de sesión introducido en la especificación EJB 3.1. Este componente se basa en el patrón de diseño del mismo nombre. Este patrón[13] de diseño garantiza que de una clase dada solamente pueda crearse una instancia, con un punto de acceso global para acceder a dicha instancia. La naturaleza única de un componente Singleton conlleva un alto rendimiento dentro del contenedor para este tipo de bean de Sesión.

[13] Una solución a un problema de diseño.

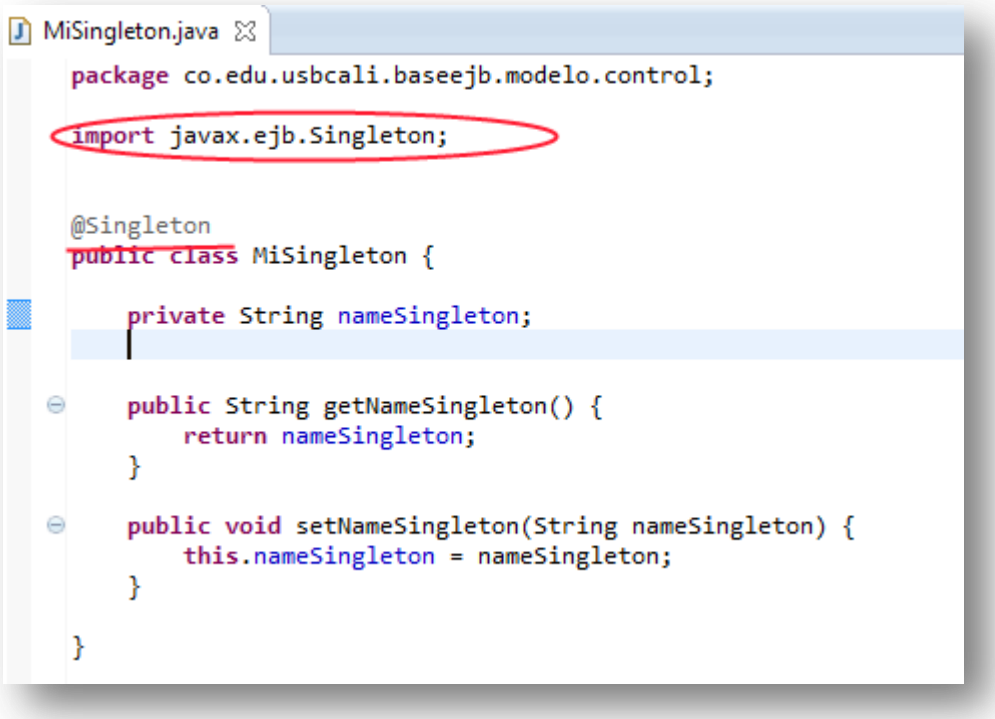


**Ilustración 4 Vista de desarrollo de un componente Singleton**

### EJB Dirigidos por mensajes

Un MDB[14] no es más que un componente que espera a que se le envie un mensaje, y realiza cierta acción cuando finalmente recibe dicho mensaje (el MDB escucha por si alguien le llama, de ahí su nombre). Algunas propiedades de los componentes MDB son:

- ✓ No mantienen estado.
- ✓ Son gestionados por el contenedor (transacciones, seguridad, concurrencia, etc).
- ✓ Son clases puras que no implementan interfaces de negocio (puesto que son invocados por un cliente).

[14] Message-driven EJBs, Beans manejados por mensajes.

## MAVEN 3.1.0

**MAVEN** este framework que nos permite la gestión de proyectos de software, basados en JavaEE, manejo de las dependencias de librerías en los proyectos y el empaquetamiento de los componentes de despliegue.

Esta herramienta es un sistema que plantea un estándar, una configuración para acceder a los múltiples repositorios de las empresas de tecnologías, usado para administrar proyectos, definiendo un ciclo de vida para su construcción, prueba, despliegue de cada uno de los componentes que conforman el proyecto, además proporciona un marco para una facilidad para reutilizar la lógica común de la estructura para todos los proyectos.

Existen tres aspectos muy relevantes por los cuales MAVEN es una herramienta muy usada en la administración de proyectos de desarrollo de software:

1. **POM.xml**

Llamado así por sus siglas en ingles *Project Object Model*, es un archivo fundamental en la funcionalidad de MAVEN, el cual contiene la configuración sobre el proyecto, como las librerías que usa, el empaquetamiento y su distribución.

A continuación se muestra un pom.xml configurado de manera básica

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/maven-v4_0_0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>org.miOrganizacion</groupId>
    <artifactId>miProyecto</artifactId>
    <version>1.0-SNAPSHOT</version>
    <packaging>jar</packaging>
    <name>Módulo de mi organización</name>
    <name>Este modulo contiene todo el proceso de negocio de mi organización</name>

    <dependencies>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>3.8.1</version>
            <scope>test</scope>
        </dependency>
    </dependencies>
</project>
```

## 2. LIFE-CYCLE

Ciclo de vida de construcción que define de manera explícita el proceso de construcción, pruebas y la distribución del artefacto, mediante las fases de ciclo de vida manejadas por el proyecto.
Entre sus principales fases se encuentran:

- **Validate**: Valida que el proyecto esta correcto y tiene toda la información necesaria para su construcción.
- **Compile:** Compila el código fuente del proyecto.
- **Test:** Lanza los test de la aplicación. Estos test no necesitan que la aplicación este empaquetada ni desplegada.
- **Package:** Toma las clases ya compiladas y crea un paquete con el proyecto, según este configurado en la etiqueta <package> del archivo pom.xml
- **Verify:** Realiza un chequeo general comprobando que el paquete cumple normas de calidad.
- **Install:** instala el paquete en el repositorio local, para ser usado como dependencia por otros proyectos usados localmente.
- **Deploy:** copia el paquete a un repositorio remoto para ser compartido con otros usuarios y otros proyectos.

Estas fases mencionadas, se ejecutan en el orden en el que fueron mencionadas.

## 3. REPOSITORIES

Repositorios en los cuales se encuentran las librerías que usará la aplicación, con el objetivo de garantizar que todas las librerías estén incluidas en el artefacto generado para el despliegue.

Para la configuración de repositorios, MAVEN crea una carpeta .m2 en el directorio HOME del usuario del sistema operativo, dentro de esta carpeta se crea un archivo llamado settings.xml.

Configuración básica de un archivo settings.xml direccionado al repositorio de JBOSS:

```
<?xml version="1.0" encoding="UTF-8"?>

<settings xmlns="http://maven.apache.org/SETTINGS/1.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0
http://maven.apache.org/xsd/settings-1.0.0.xsd">

  <repositories>

     <repository>

        <id>jboss-public-repository-group</id>

        <name>JBoss - Public JBoss Repository Group</name>

        <url>https://repository.jboss.org/nexus/content/groups/public-jboss/</url>

        <layout>default</layout>

     </repository>

  </repositories>

</settings>
```

# 8. COMPONENTES DE LA ARQUITECTURA DE ZATHURACODE

Zathuracode maneja una arquitectura de 3 capas donde cada capa representa un componente con un proceso específico:

1. **METADATAREADER**

Es la capa encargada de leer los POJO[15]'s con tecnología JPA, realizando un mapeo de las entidades de la base de datos, y convertirla a un modelo genérico con un diccionario de datos que contiene la información de nombre de tabla, columnas y referencias.

- **Model**: Componente genérico que representa una tabla de la base de datos, incluyendo columnas y referencias.
- **MetaDataModel:** Componente que contiene el modelo de referencia de la tabla, y el diccionario de datos que contiene información de atributos y referencias mapeadas.
- **MetaDataReaderFactory:** Componente que usa el patrón de diseño **Factory**[16], para administrar las instancias del componente **MetaDataReader** al momento de realizar el mapeo objeto relacional.
- **MetaDataReader:** Componente encargado de realizar el mapeo objeto relacional de cada entidad de la base de datos.
- **MetaDataUtil:** Componente que contiene métodos a útiles para la búsqueda en el diccionario de datos.

2. **GENERATOR**

Es la capa encargada de generar el tipo de arquitectura que ha sido seleccionada, llamando el tipo de implementación a usar para generar la arquitectura. Esta capa también es la encargada de la interfaz gráfica de usuario.

- **IZathuraGenerator:** Interfaz que contiene los métodos a usar para el generador de arquitecturas.
- **ZathuraGeneratorFactory:** Componente que usa el patrón de diseño **Factory** para administrar las instancias del componente **GeneratorUtil**.
- **GeneratorUtil:** Componente que se encarga de leer las plantillas del motor y volverlas archivos .java, a través de una herramienta llamada **Velocity**[17] quien recibe una plantilla y la traduce a un archivo específico.
- **JalopyCodeFormater:** Componente que se encarga de compilar el archivo generado por **Velocity** y crear el archivo .java.

---

[15] Plain Old Java Object, disponible en: http://www.martinfowler.com/bliki/POJO.html

[16] Catálogo de patrones, disponible en: http://www.corej2eepatterns.com/Patterns2ndEd/index.htm

[17] Velocity es un proyecto de Apache que usa templates basado en Java, disponible en: http://velocity.apache.org/engine/devel/user-guide.html

### 3. ENGINE

Es la capa encargada de administrar las plantillas de la arquitectura que se va a generar, esta capa contiene un componente particular de cada arquitectura que será la implementación y una interfaz que define que métodos son los que se deben usar para crear la arquitectura.

- **IZathuraTemplate:** Interfaz que define los métodos a usar para que una arquitectura quede correctamente generada.

Para el caso de esta capa, mencionaremos las 3 implementaciones que se crearon en esta versión de zathuracode, una por cada arquitectura y que usaran la interfaz **IZathuraTemplate**:

- **ZathuraJavaEE_HibernateCore_Web_Spring_Prime_Centric:** Implementación usada para generar la arquitectura JavaEE Primefaces 3.5 Hibernate 4.2.3 Spring 3.2.3.
- **ZathuraJavaEE_Jpa_Web_Prime_Spring_Centric:** Implementación usada para generar la arquitectura JavaEE Primefaces 3.5 JPA 2.0 Spring 3.2.3.
- **ZathuraJavaEE_Jpa_Web_Prime_EJB_Centric:** Implementación usada para generar la arquitectura JavaEE Primefaces 3.5 JPA 2.0 EJB 3.1

# 9. INTEGRACIÓN DE LA HERRAMIENTA ADMINISTRADORA DE PROYECTOS MAVEN A ZATHURACODE

Para la integración de MAVEN a Zathuracode, se modificaron la capa Engine y la capa Generator.

La capa Generator contiene un paquete llamado generatorTemplates, en el cual se crea un paquete con el nombre de la arquitectura y se añaden los templates de las clases que se van a generar, es decir, las clases del domino, del DAO, de la lógica, de los delegados y de la vista, tienen sus respectivos templates.

Para integrar MAVEN es necesario crear un template llamado POM.vm, el cual es el template que contiene la configuración del proyecto y las librerías a usar, para que al momento de generar el proyecto, este pueda descargar las librerías desde los diferentes repositorios. Así que de esta manera este template se agregó a cada una de las arquitecturas, haciendo que las arquitecturas existentes y las creadas en esta versión puedan soportar esta herramienta.

En la interfaz gráfica de este componente se adiciono una validación que permite saber si el proyecto en el que se va a generar el código es un proyecto WEB o es MAVEN, el cual es utilizado en la capa Engine.

En la capa Engine se realiza la validación de qué si el proyecto es MAVEN o WEB, con base a esto, el motor que es el que genera todos los paquetes y las clases basado en los templates, no adicione las librerías al proyecto, sino que escriba el archivo POM con la configuración del proyecto y la dependencia de librerías.

A continuación se mencionan los pasos realizados para la integración de MAVEN:

1. Creación del template POM.vm el cual se convertirá en el archivo POM.xml.
2. Adicionar el POM.vm a cada arquitectura y configurar la dependencia de librerías para cada arquitectura.
3. Modificar la interfaz gráfica haciendo que valide si el proyecto es WEB o es MAVEN.
4. Adicionar al motor de cada arquitectura la validación del tipo de proyecto, si es WEB entonces se copiaran las librerías que usa la arquitectura y si es MAVEN se usa crea el archivo POM.xml.

# 10. COMPONENTES DE LAS ARQUITECTURAS JAVAEE GENERADAS POR ZATHURACODE

### ENTITY MANAGER HELPER

En Entity Manager Helper para la arquitectura que se genera, permite manejar las entidades y POJOs además de sus estados.

Para este caso el Entity Manager es usado por el Framework Hibernate, el cual se encarga también de realizar todas las transacciones a la base datos.

### DAO

En esta arquitectura estos se encargan de todo lo que tenga que ver con salvar, guardar, modificar y realizar consultas a la base de datos. Son usados solo por la fábrica de JPA y necesitan del entityManager para realizar consultas o para ejecutar querys.

Los DAOs se conectan directamente a él para ejecutar instrucciones SQL (consultas y modificaciones) y sincronizar los POJOs de la aplicación con la base de datos, realizando, si es requerido, commits o rollbacks a las transacciones creadas.

### CONTROL

Es el encargado de redirigir o asignar una aplicación (un modelo) a cada petición; el controlador debe poseer de algún modo, un "mapa" de correspondencias entre peticiones y respuestas (aplicaciones o modelo) que se les asignan.

### BUSINESSDELEGATE

Clase que ofrece una serie de servicios para consumir, todos sus métodos son estáticos.

### VIEW

Es el encargado de mostrar el modelo entregado por el control representándolo por medio de la interfaz de usuario.

### XHTML

XHTML[18] es el formato de páginas WEB que permite a la aplicación interactuar con el usuario capturando y mostrando información a través de los componentes visuales del framework Primefaces. Este formato es soportado por navegadores como Firefox, Chrome, Opera, Internet Explorer.

[18] XHTML (Extensible HyperText Markup Languaje)

**SPRING FRAMEWORK**

El framework de SPRING provee un exhaustivo desarrollo y un modelo de configuración para aplicaciones Enterprise modernas basadas en java en cualquier tipo de plataforma de despliegue, un elemento clave de SPRING es el soporte de infraestructura a nivel de aplicación: Spring se enfoca en “glue code” (Unir código) de aplicaciones Enterprise que se enfocan a nivel de lógica de negocio, sin capas innecesarias para un entorno especifico de despliegue.

SPRING incluye:

- Flexibilidad en la inyección de dependencias con XML y un estilo de configuración basado en anotaciones
- Soporte avanzado para programación orientada a aspectos con base en proxy y variantes basadas en AspectJ[19].
- Soporte para transacciones declarativas, almacenando las declaraciones, validándolas y declarando un formato.
- Potente abstracción con especificaciones comunes JavaEE como JDBC, JPA, JTA y JMS.
- Primera clase que soporta en un framework de código abierto como Hibernate y Quartz[20].
- Es un flexible framework web para construir aplicaciones RESTful MVC y endpoints services.
- Facilidad para realizar pruebas unitarias y pruebas integración.

[19] http://www.eclipse.org/aspectj/

[20] http://quartz-scheduler.org/

# 11. ARQUITECTURA DE ZATHURACODE PARA APLICACIONES JAVAEE PRIMEFACES 3.5 HIBERNATE 4.2.3 SPRING 3.2.3

## APLICACIONES JAVAEE HIBERNATE CENTRIC UTILIZANDO PRIMEFACES Y SPRING

Se plantea la nueva arquitectura para Zathuracode utilizando los componentes de Primefaces en la versión 3.5, Hibernate 4.2.3 y Spring 3.2.3 para la presentación, acceso a datos y la forma de incluir las dependencias respectivamente. A continuación veremos el diagrama de componentes de Zathuracode utilizando Primefaces, Hibernate y Spring.

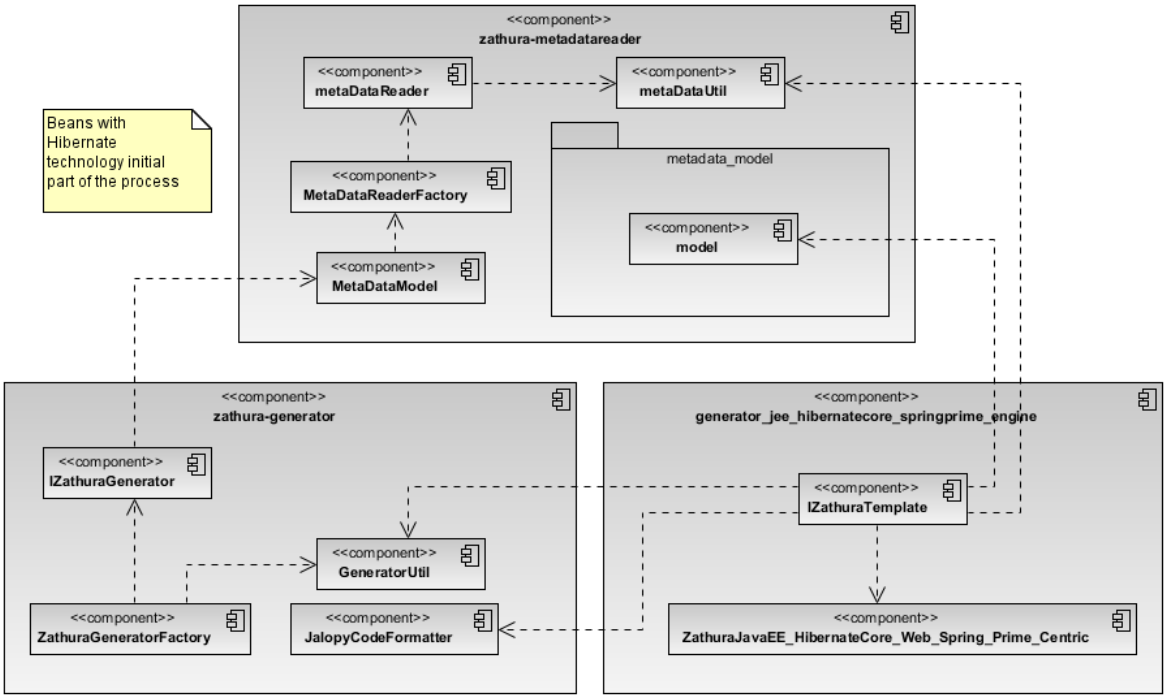


**Ilustración 5 Vista de desarrollo y funcional de Zathuracode utilizando Primefaces 3.5, Hibernate 4.2.3 y Spring 3.2.3**

Para adicionar la arquitectura a Zathuracode se modifica la capa Engine y Generator de la siguiente manera:

1. Se adiciona el paquete de templates de la arquitectura llamado **zathura-JavaEE6-HibernateCore4-Prime35-Spring32-Centric**.
2. Se adiciona el paquete para la capa Engine llamado **org.zathuracode.generator.jee6.hibernatecore4.spring32.prime.engine** encargado de crear los paquetes de la arquitectura y los archivos .java de cada paquete.
3. Se adiciona el paquete para la capa Engine llamado **org.zathuracode.generator.jee6.hibernatecore4.spring32.prime.utils** encargado de comunicarse con la capa MetaReader para usar la información del diccionario de datos.
4. Se agregan las librerías que usa la arquitectura en el paquete generatorLibraries, La capa Engine es la encargada de saber que librerías se deben seleccionar para que la arquitectura no contenga errores de librerías.
5. Se modifica el archivo zathura-generator-factory-config.xml en la capa Generator, para que la arquitectura se muestre en la interfaz gráfica de usuario y así mismo el usuario pueda seleccionarla. La arquitectura se adiciona de la siguiente manera:

```
<generator>
        <name>ZathuraJavaEE_HibernateCore4_Spring32_PrimeFaces35_Web_Centric</name>
        <gui-name>PrimeFaces3.5 + Spring3.2.3 + HibernateCore4.2.3</gui-name>
        <class>org.zathuracode.generator.jee6.hibernatecore4.spring32.prime.engine.ZathuraJavaEE_HibernateCore4_Spring32_Prime</
class>
                <persistence>hibernateCore</persistence>
                <zathuraVersion>4.0</zathuraVersion>
                <description>
                <![CDATA[
                <[CDATA[
                  <DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
                        <HTML>
                        <HEAD>
                        <LINK TYPE="text/css" HREF="${RELATIVE_PATH}css\form.css" REL="stylesheet" />
                            <META HTTP-EQUIV="CONTENT-TYPE" CONTENT="text/html; charset=utf-8">
                              <TITLE></TITLE>
                        </HEAD>
                        <BODY LANG="es-CO" DIR="LTR">
                        <P LANG="en-US" ALIGN=CENTER STYLE="font-style: normal"><FONT SIZE=3><B>The
                                JavaEE HibernateCore Spring PrimeFaces Web Centric </B></FONT>
                        <TABLE>
                            <TR>
                                <TD>
                                  <DIV ALIGN=CENTER STYLE="width:140px;">
                                      <IMG SRC="${RELATIVE_PATH}icons\hibernate.png" ID="hibernateLogo"/>
                                  </DIV>
                                </TD>
                                <TD>
                                  <DIV ALIGN=CENTER STYLE="width:140px;">
                                      <IMG SRC="${RELATIVE_PATH}icons\spring.png"  ID="springLogo"/>
                                  </DIV>
                                 </TD>
                          </TR>
                          <TR>
                              <TD>
                                  <DIV ALIGN=CENTER STYLE="width:140px;">
                                      <IMG SRC="${RELATIVE_PATH}icons\jsf.png" ID="jsfLogo"/>
                                  </DIV>
                             </TD>
                             <TD>
                                 <DIV ALIGN=CENTER STYLE="width:140px;">
                                      <IMG SRC="${RELATIVE_PATH}icons\primefaces.png" ID="primefacesLogo"/>
                                 </DIV>
                             </TD>
                         </TR>
```

```
                        </TABLE>
                      </P>
                        <P LANG="en-US" ALIGN=LEFT STYLE="font-style: normal; font-weight: medium">
                          <FONT SIZE=2>This Generator is appropriate for Web applications that
                                using  the JavaEE Patterns. </BR>
                                Includes:</FONT></P>
                        <TABLE>
                          <TR>
                            <TD>
                              <P LANG="en-US" ALIGN=LEFT STYLE="font-style: normal; font-weight: medium">
                                <FONT SIZE=2>HibernateCore 4.2.3</FONT></P>
                            </TD>
                            <TD>
                              <P LANG="en-US" ALIGN=LEFT STYLE="font-style: normal; font-weight: medium">
                                <FONT SIZE=2>Spring 3.2.3</FONT></P>
                            </TD>
                          </TR>
                          <TR>
                            <TD>
                              <P LANG="en-US" ALIGN=LEFT STYLE="font-style: normal; font-weight: medium">
                                <FONT SIZE=2>JSF 2.1</FONT></P>
                            </TD>
                            <TD>
                              <P LANG="en-US" ALIGN=LEFT STYLE="font-style: normal; font-weight: medium">
                                <FONT SIZE=2>PrimeFaces 3.5</FONT></P>
                            </TD>
                          </TR>
                        </TABLE>
                        <P LANG="en-US" ALIGN=LEFT STYLE="font-style: normal; font-weight: medium">
                          <FONT SIZE=2>The next is a list the generate files.</FONT></P>
                            <UL>
                                <LI><P LANG="en-US" ALIGN=LEFT STYLE="font-style: normal; font-weight: medium">
                                    <FONT SIZE=2>XHTML Views</FONT></P>
                                <LI><P LANG="en-US" ALIGN=LEFT STYLE="font-style: normal; font-weight: medium">
                                    <FONT SIZE=2>ManageBeans</FONT></P>
                                <LI><P LANG="en-US" ALIGN=LEFT STYLE="font-style: normal; font-weight: medium">
                                    <FONT SIZE=2>Faces-config.xml</FONT></P>
                                <LI><P LANG="en-US" ALIGN=LEFT STYLE="font-style: normal; font-weight: medium">
                                    <FONT SIZE=2>Bussines Delegate</FONT></P>
                                <LI><P LANG="en-US" ALIGN=LEFT STYLE="font-style: normal; font-weight: medium">
                                    <FONT SIZE=2>POJO (Logic)</FONT></P>
                                <LI><P LANG="en-US" ALIGN=LEFT STYLE="font-style: normal; font-weight: medium">
                                    <FONT SIZE=2>Generic HibernateDao</FONT></P>
                                <LI><P LANG="en-US" ALIGN=LEFT STYLE="font-style: normal; font-weight: medium">
                                    <FONT SIZE=2>DAO</FONT></P>
                                <LI><P LANG="en-US" ALIGN=LEFT STYLE="font-style: normal; font-weight: medium">
                                    <FONT SIZE=2>IDAO</FONT></P>
                                <LI><P LANG="en-US" ALIGN=LEFT STYLE="font-style: normal; font-weight: medium">
                                    <FONT SIZE=2>hibernate.cfg.xml</FONT></P>
                                <LI><P LANG="en-US" ALIGN=LEFT STYLE="font-style: normal; font-weight: medium">
                                    <FONT SIZE=2>web.xml</FONT></P>
                            </UL>
                        </BODY>
                    </HTML>
                ]]>
                </description>
        </generator>
```

**Documento  1 XML de configuración para Zathuracode Primefaces 3.5, Hibernate 4.2.3 y Spring 3.2.3**

A continuación vemos las capas que componen el proyecto con JavaEE Primefaces Spring:

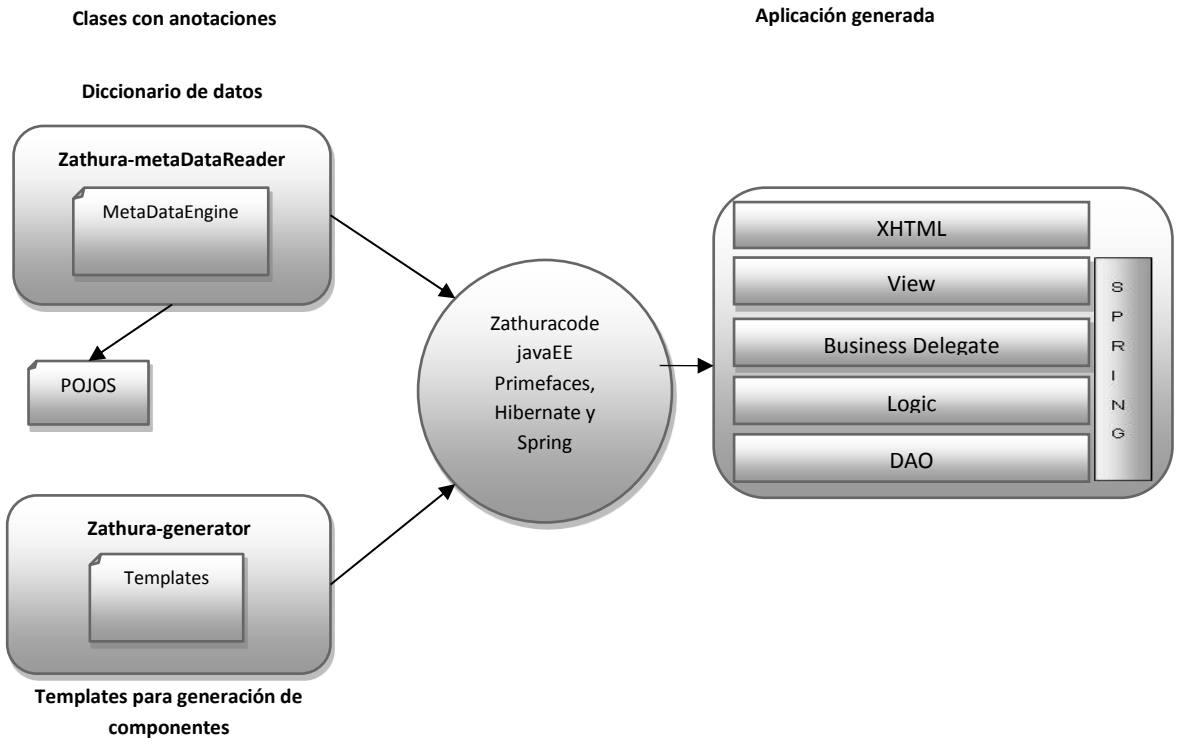


**Ilustración 6 Proceso de generación Zathuracode con capas de Primefaces 3.5, Hibernate 4.2.3 y Spring 3.2.3**

A continuación se presenta el diagrama de los componentes involucrados en la arquitectura JavaEE Centric utilizando la tecnología Primefaces 3.5, Hibernate 4.2.3 y Spring 3.2.3. En la siguiente ilustración se visualiza los componentes del lado del cliente y los componentes del lado del servidor.

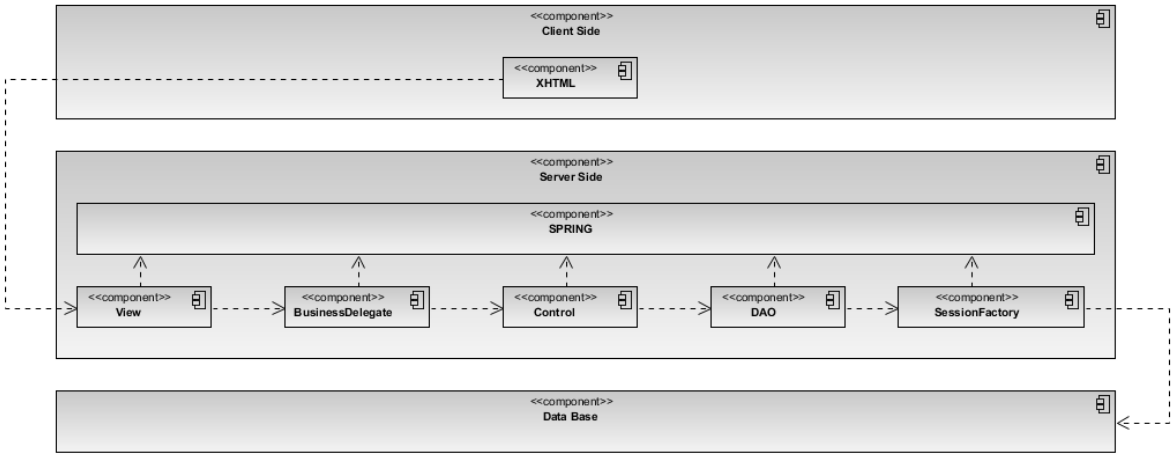


**Ilustración 7 Diagrama de componentes utilizando Primefaces 3.5, Hibernate 4.2.3 y Spring 3.2.3.**

# 12. ARQUITECTURA DE ZATHURACODE PARA APLICACIONES JAVAEE PRIMEFACES 3.5 JPA 2.0 SPRING 3.2.3

## APLICACIONES JAVAEE JPA CENTRIC UTILIZANDO PRIMEFACES Y SPRING

Se plantea la nueva arquitectura para Zathuracode utilizando los componentes de Primefaces en la versión 3.5, JPA 2.0 y Spring 3.2.3 para la presentación, acceso a datos y la forma de incluir las dependencias respectivamente. A continuación veremos el diagrama de componentes de Zathuracode utilizando Primefaces, JPA y Spring.

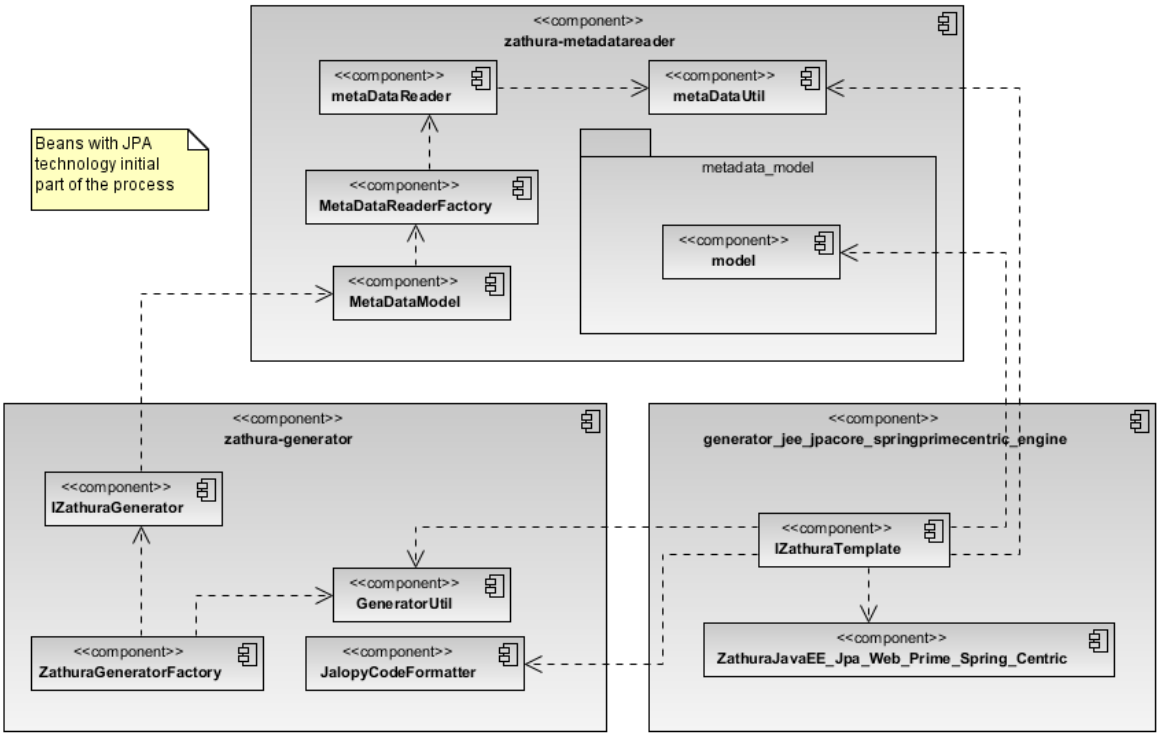


**Ilustración 8 Vista de desarrollo y funcional de Zathuracode utilizando Primefaces 3.5 JPA 2.0 y Spring 3.2.3**

Para adicionar la arquitectura a Zathuracode se modifica la capa Engine y Generator de la siguiente manera:

1. Se adiciona el paquete de templates de la arquitectura llamado **zathura-JavaEE6-jpa-Prime35-Spring32-Centric**.
2. Se adiciona el paquete para la capa Engine llamado **org.zathuracode.generator.jee6.jpa.spring32.prime.engine** encargado de crear los paquetes de la arquitectura y los archivos .java de cada paquete.
3. Se adiciona el paquete para la capa Engine llamado **org.zathuracode.generator.jee6.jpa.spring32.prime.utils** encargado de comunicarse con la capa MetaReader para usar la información del diccionario de datos.
4. Se agregan las librerías que usa la arquitectura en el paquete generatorLibraries, La capa Engine es la encargada de saber que librerías se deben seleccionar para que la arquitectura no contenga errores de librerías.
5. Se modifica el archivo zathura-generator-factory-config.xml en la capa Generator, para que la arquitectura se muestre en la interfaz gráfica de usuario y así mismo el usuario pueda seleccionarla. La arquitectura se adiciona de la siguiente manera:

```
<generator>
        <name>ZathuraJavaEE_JPA_Spring32_Prime35</name>
        <gui-name>PrimeFaces3.5 + Spring3.2.3 + JPA2</gui-name>
        <class>org.zathuracode.generator.jee6.jpa.spring32.prime.engine.ZathuraJavaEE_jpa_Spring32_Prime</class>
        <persistence>jpa</persistence>
        <zathuraVersion>4.0</zathuraVersion>
                <description>
                <![CDATA[
                <![CDATA[
                  <DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
                        <HTML>
                        <HEAD>
                        <LINK TYPE="text/css" HREF="${RELATIVE_PATH}css\form.css" REL="stylesheet" />
                            <META HTTP-EQUIV="CONTENT-TYPE" CONTENT="text/html; charset=utf-8">
                               <TITLE></TITLE>
                        </HEAD>
                        <BODY LANG="es-CO" DIR="LTR">
                        <P LANG="en-US" ALIGN=CENTER STYLE="font-style: normal"><FONT SIZE=3><B> The
                        JavaEE JpaCore Spring PrimeFaces Web Centric </B></FONT>
                        <TABLE>
                            <TR>
                                <TD>
                                  <DIV ALIGN=CENTER STYLE="width:140px;">
                                      <IMG SRC="${RELATIVE_PATH}icons\jpa.png " ID="jpaLogo"/>
                                  </DIV>
                                </TD>
                                <TD>
                                  <DIV ALIGN=CENTER STYLE="width:140px;">
                                      <IMG SRC="${RELATIVE_PATH}icons\spring.png"  ID="springLogo"/>
                                  </DIV>
                                 </TD>
                            </TR>
                            <TR>
                                <TD>
                                    <DIV ALIGN=CENTER STYLE="width:140px;">
                                        <IMG SRC="${RELATIVE_PATH}icons\jsf.png" ID="jsfLogo"/>
                                    </DIV>
                                </TD>
                                <TD>
                                    <DIV ALIGN=CENTER STYLE="width:140px;">
                                        <IMG SRC="${RELATIVE_PATH}icons\primefaces.png" ID="primefacesLogo"/>
                                    </DIV>
                                </TD>
                            </TR>
                        </TABLE>
                        </P>
                            <P LANG="en-US" ALIGN=LEFT STYLE="font-style: normal; font-weight: medium">
```

```
                                <FONT SIZE=2>This Generator is appropriate for Web applications that
                                        using  the JavaEE Patterns. </BR>
                                        Includes:</FONT></P>
                                <TABLE>
                                  <TR>
                                    <TD>
                                      <P LANG="en-US" ALIGN=LEFT STYLE="font-style: normal; font-weight: medium">
                                        <FONT SIZE=2> JPA 2.1</FONT></P>
                                    </TD>
                                    <TD>
                                      <P LANG="en-US" ALIGN=LEFT STYLE="font-style: normal; font-weight: medium">
                                        <FONT SIZE=2>Spring 3.2.3</FONT></P>
                                    </TD>
                                  </TR>
                                  <TR>
                                    <TD>
                                      <P LANG="en-US" ALIGN=LEFT STYLE="font-style: normal; font-weight: medium">
                                        <FONT SIZE=2>JSF 2.1</FONT></P>
                                    </TD>
                                    <TD>
                                      <P LANG="en-US" ALIGN=LEFT STYLE="font-style: normal; font-weight: medium">
                                        <FONT SIZE=2>PrimeFaces 3.5</FONT></P>
                                    </TD>
                                  </TR>
                                </TABLE>
                                <P LANG="en-US" ALIGN=LEFT STYLE="font-style: normal; font-weight: medium">
                                  <FONT SIZE=2>The next is a list the generate files.</FONT></P>
                                    <UL>
                                        <LI><P LANG="en-US" ALIGN=LEFT STYLE="font-style: normal; font-weight: medium">
                                        <FONT SIZE=2>XHTML Views</FONT></P>
                                        <LI><P LANG="en-US" ALIGN=LEFT STYLE="font-style: normal; font-weight: medium">
                                        <FONT SIZE=2>ManageBeans</FONT></P>
                                        <LI><P LANG="en-US" ALIGN=LEFT STYLE="font-style: normal; font-weight: medium">
                                        <FONT SIZE=2>Faces-config.xml</FONT></P>
                                        <LI><P LANG="en-US" ALIGN=LEFT STYLE="font-style: normal; font-weight: medium">
                                        <FONT SIZE=2>Bussines Delegate</FONT></P>
                                        <LI><P LANG="en-US" ALIGN=LEFT STYLE="font-style: normal; font-weight: medium">
                                        <FONT SIZE=2>POJO (Logic)</FONT></P>
                                        <LI><P LANG="en-US" ALIGN=LEFT STYLE="font-style: normal; font-weight: medium">
                                        <FONT SIZE=2>Generic JpaDao</FONT></P>
                                        <LI><P LANG="en-US" ALIGN=LEFT STYLE="font-style: normal; font-weight: medium">
                                        <FONT SIZE=2>DAO</FONT></P>
                                        <LI><P LANG="en-US" ALIGN=LEFT STYLE="font-style: normal; font-weight: medium">
                                        <FONT SIZE=2>IDAO</FONT></P>
                                        <LI><P LANG="en-US" ALIGN=LEFT STYLE="font-style: normal; font-weight: medium">
                                        <FONT SIZE=2>web.xml</FONT></P>
                                    </UL>
                            </BODY>
                        </HTML>
                ]]>
                </description>
        </generator>
```

**Documento 2 XML de configuración para Zathuracode Primefaces 3.5 JPA 2.0 y Spring 3.2.3**

A continuación vemos las capas que componen el proyecto con JavaEE Primefaces JPA Spring:

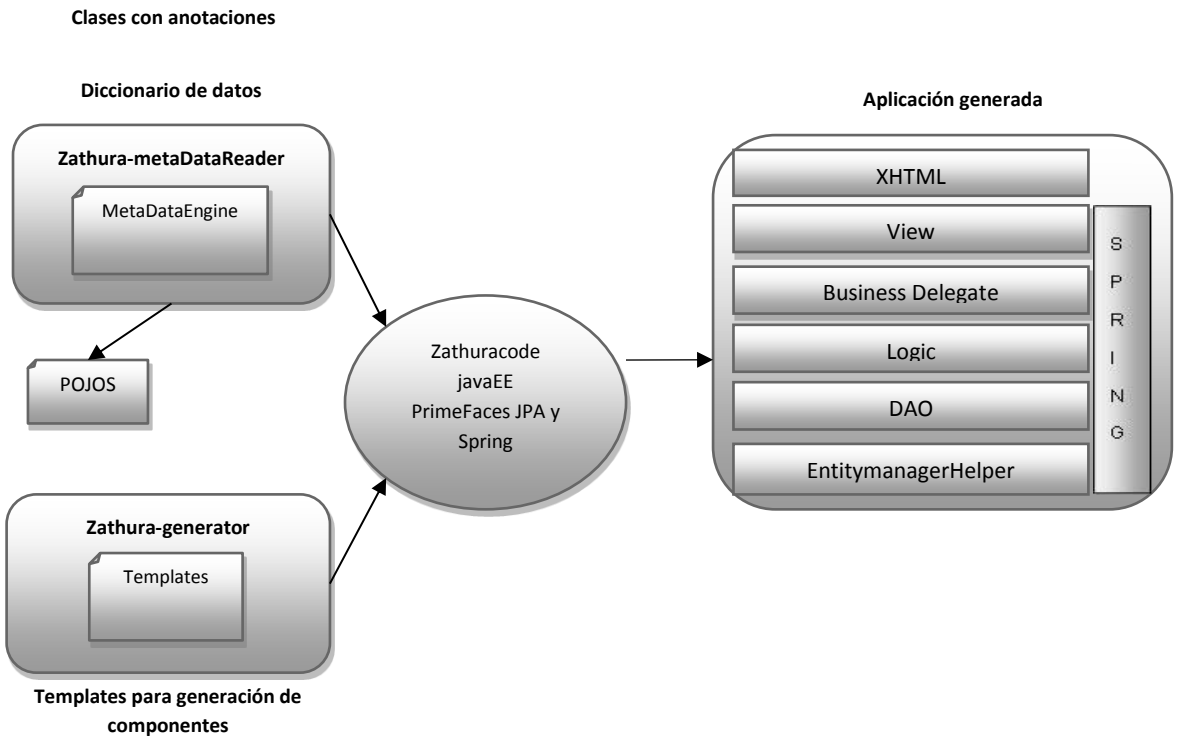


**Ilustración 9 Proceso de generación Zathuracode con capas de Primefaces 3.5 y JPA 2.0 con Spring 3.2.3**

A continuación se presenta el diagrama de los componentes involucrados en la arquitectura JavaEE Centric utilizando la tecnología Primefaces 3.5, JPA 2.0 y Spring 3.2.3. En la siguiente ilustración se visualiza los componentes del lado del cliente y los componentes del lado del servidor.

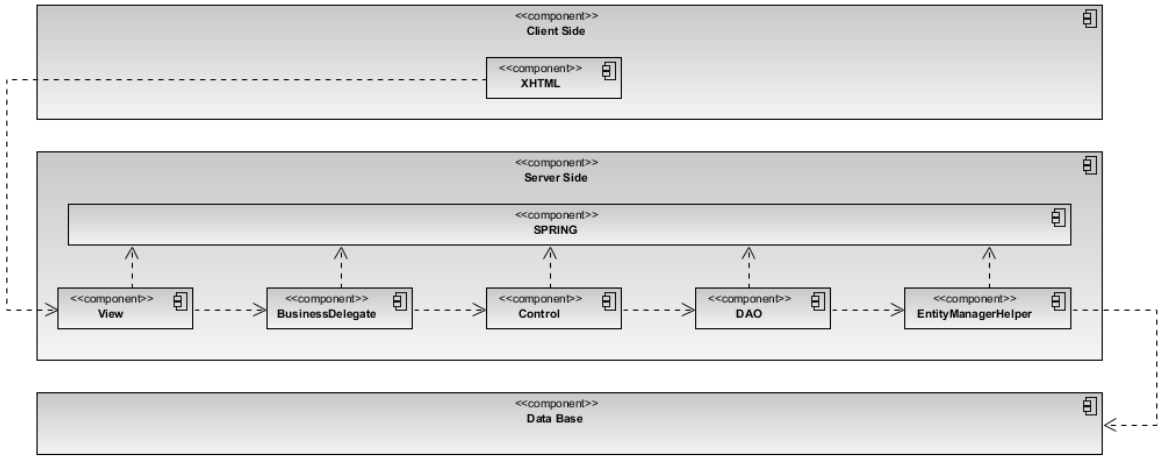


**Ilustración 10 Diagrama de componentes utilizados en la arquitectura JavaEE web centric utilizando Primefaces 3.5, JPA 2.0 y Spring 3.2.3**

# 13. ARQUITECTURA DE ZATHURACODE PARA APLICACIONES JAVAEE PRIMEFACES 3.5 JPA 2.0 EJB 3.1

## APLICACIONES JAVAEE JPA CENTRIC UTILIZANDO PRIMEFACES Y EJB

Se plantea la nueva arquitectura para Zathuracode utilizando los componentes de Primefaces en la versión 3.5, JPA 2.0 y EJB 3.1 para la presentación, acceso a datos y la forma de incluir las dependencias respectivamente. A continuación veremos el diagrama de componentes de Zathuracode utilizando Primefaces, JPA y EJB.

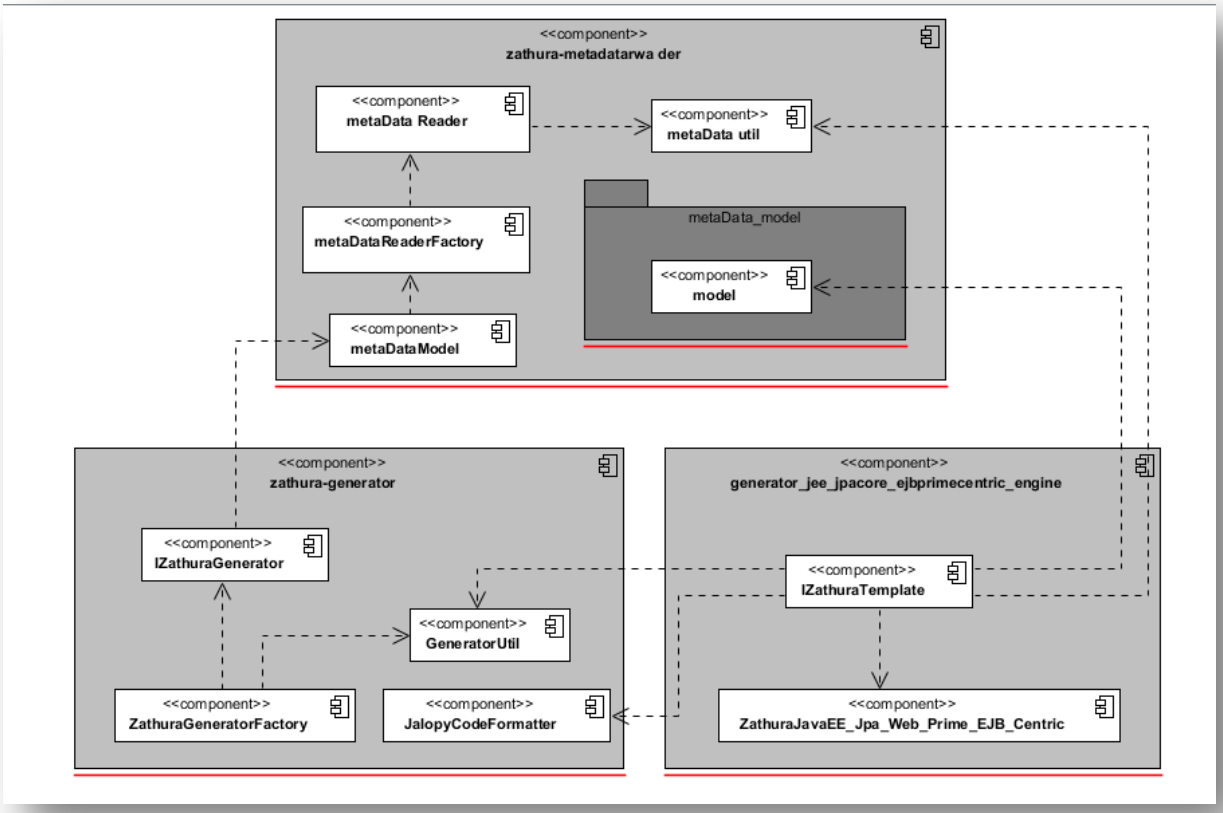


**Ilustración 11 Vista de desarrollo y funcional de Zathuracode utilizando Primefaces 3.5, JPA 2.0 y EJB 3.1**

Para adicionar la arquitectura a Zathuracode se modifica la capa Engine y Generator de la siguiente manera:

1. Se adiciona el paquete de templates de la arquitectura llamado **zathura-JavaEE6-jpa-Prime35-Ejb31-Centric**.
2. Se adiciona el paquete para la capa Engine llamado **org.zathuracode.generator.jee6.jpacore.ejb31.prime35.engine** encargado de crear los paquetes de la arquitectura y los archivos .java de cada paquete.
3. Se adiciona el paquete para la capa Engine llamado **org.zathuracode.generator.jee6.jpacore.ejb31.prime35.utils** encargado de comunicarse con la capa MetaReader para usar la información del diccionario de datos.
4. Se agregan las librerías que usa la arquitectura en el paquete generatorLibraries, La capa Engine es la encargada de saber que librerías se deben seleccionar para que la arquitectura no contenga errores de librerías.
5. Se modifica el archivo zathura-generator-factory-config.xml en la capa Generator, para que la arquitectura se muestre en la interfaz gráfica de usuario y así mismo el usuario pueda seleccionarla. La arquitectura se adiciona de la siguiente manera:

```
<generator>
        <name>ZathuraJavaEE_JPA_EJB31_Prime35</name>
        <gui-name>PrimeFaces3.5 + EJB 3.1 + JPA2</gui-name>
        <class>org.zathuracode.generator.jee6.jpacore.ejb31.prime35.engine.ZathuraJavaEE_JpaCore_Ejb31_Prime35</class>
        <persistence>jpa</persistence>
        <zathuraVersion>4.0</zathuraVersion>
        <description>
        <![CDATA[
                <![CDATA[
                  <DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
                        <HTML>
                        <HEAD>
                        <LINK TYPE="text/css" HREF="${RELATIVE_PATH}css\form.css" REL="stylesheet" />
                            <META HTTP-EQUIV="CONTENT-TYPE" CONTENT="text/html; charset=utf-8">
                               <TITLE></TITLE>
                        </HEAD>
                        <BODY LANG="es-CO" DIR="LTR">
                        <P LANG="en-US" ALIGN=CENTER STYLE="font-style: normal"><FONT SIZE=3><B> The
                                JavaEE JpaCore EJB PrimeFaces Web Centric </B></FONT>
                        <TABLE>
                            <TR>
                                <TD>
                                  <DIV ALIGN=CENTER STYLE="width:140px;">
                                      <IMG SRC="${RELATIVE_PATH}icons\jpa.png " ID="jpaLogo"/>
                                  </DIV>
                                </TD>
                                <TD>
                                  <DIV ALIGN=CENTER STYLE="width:140px;">
                                      <IMG SRC="${RELATIVE_PATH}icons\ejb.png"  ID=" ejbLogo "/>
                                  </DIV>
                                 </TD>
                              </TR>
                              <TR>
                                 <TD>
                                    <DIV ALIGN=CENTER STYLE="width:140px;">
                                       <IMG SRC="${RELATIVE_PATH}icons\jsf.png" ID="jsfLogo"/>
                                    </DIV>
                                 </TD>
                                 <TD>
                                    <DIV ALIGN=CENTER STYLE="width:140px;">
                                       <IMG SRC="${RELATIVE_PATH}icons\primefaces.png" ID="primefacesLogo"/>
                                    </DIV>
                                 </TD>
                              </TR>
                           </TABLE>
                        </P>
                           <P LANG="en-US" ALIGN=LEFT STYLE="font-style: normal; font-weight: medium">
```

```
                                <FONT SIZE=2>This Generator is appropriate for Web applications that
                                        using  the JavaEE Patterns. </BR>
                                        Includes:</FONT></P>
                                <TABLE>
                                  <TR>
                                    <TD>
                                      <P LANG="en-US" ALIGN=LEFT STYLE="font-style: normal; font-weight: medium">
                                        <FONT SIZE=2> JPA 2.1</FONT></P>
                                    </TD>
                                    <TD>
                                      <P LANG="en-US" ALIGN=LEFT STYLE="font-style: normal; font-weight: medium">
                                        <FONT SIZE=2> EJB 3.1</FONT></P>
                                    </TD>
                                  </TR>
                                  <TR>
                                    <TD>
                                      <P LANG="en-US" ALIGN=LEFT STYLE="font-style: normal; font-weight: medium">
                                        <FONT SIZE=2>JSF 2.1</FONT></P>
                                    </TD>
                                    <TD>
                                      <P LANG="en-US" ALIGN=LEFT STYLE="font-style: normal; font-weight: medium">
                                        <FONT SIZE=2>PrimeFaces 3.5</FONT></P>
                                    </TD>
                                  </TR>
                                </TABLE>
                                <P LANG="en-US" ALIGN=LEFT STYLE="font-style: normal; font-weight: medium">
                                  <FONT SIZE=2>The next is a list the generate files.</FONT></P>
                                    <UL>
                                        <LI><P LANG="en-US" ALIGN=LEFT STYLE="font-style: normal; font-weight: medium">
                                        <FONT SIZE=2>XHTML Views</FONT></P>
                                        <LI><P LANG="en-US" ALIGN=LEFT STYLE="font-style: normal; font-weight: medium">
                                        <FONT SIZE=2>ManageBeans</FONT></P>
                                        <LI><P LANG="en-US" ALIGN=LEFT STYLE="font-style: normal; font-weight: medium">
                                        <FONT SIZE=2>Faces-config.xml</FONT></P>
                                        <LI><P LANG="en-US" ALIGN=LEFT STYLE="font-style: normal; font-weight: medium">
                                        <FONT SIZE=2>Bussines Delegate</FONT></P>
                                        <LI><P LANG="en-US" ALIGN=LEFT STYLE="font-style: normal; font-weight: medium">
                                        <FONT SIZE=2>POJO (Logic)</FONT></P>
                                        <LI><P LANG="en-US" ALIGN=LEFT STYLE="font-style: normal; font-weight: medium">
                                        <FONT SIZE=2>Generic JpaDao</FONT></P>
                                        <LI><P LANG="en-US" ALIGN=LEFT STYLE="font-style: normal; font-weight: medium">
                                        <FONT SIZE=2>DAO</FONT></P>
                                        <LI><P LANG="en-US" ALIGN=LEFT STYLE="font-style: normal; font-weight: medium">
                                        <FONT SIZE=2>IDAO</FONT></P>
                                        <LI><P LANG="en-US" ALIGN=LEFT STYLE="font-style: normal; font-weight: medium">
                                        <FONT SIZE=2>web.xml</FONT></P>
                                    </UL>
                            </BODY>
                        </HTML>
                ]]>
                </description>
        </generator>
```

**Documento 3 XML de configuración para Zathuracode Primefaces 3.5, JPA 2.0 y EJB 3.1**

A continuación vemos las capas que componen el proyecto con JavaEE Primefaces 3.5, JPA 2.0 y EJB 3.1

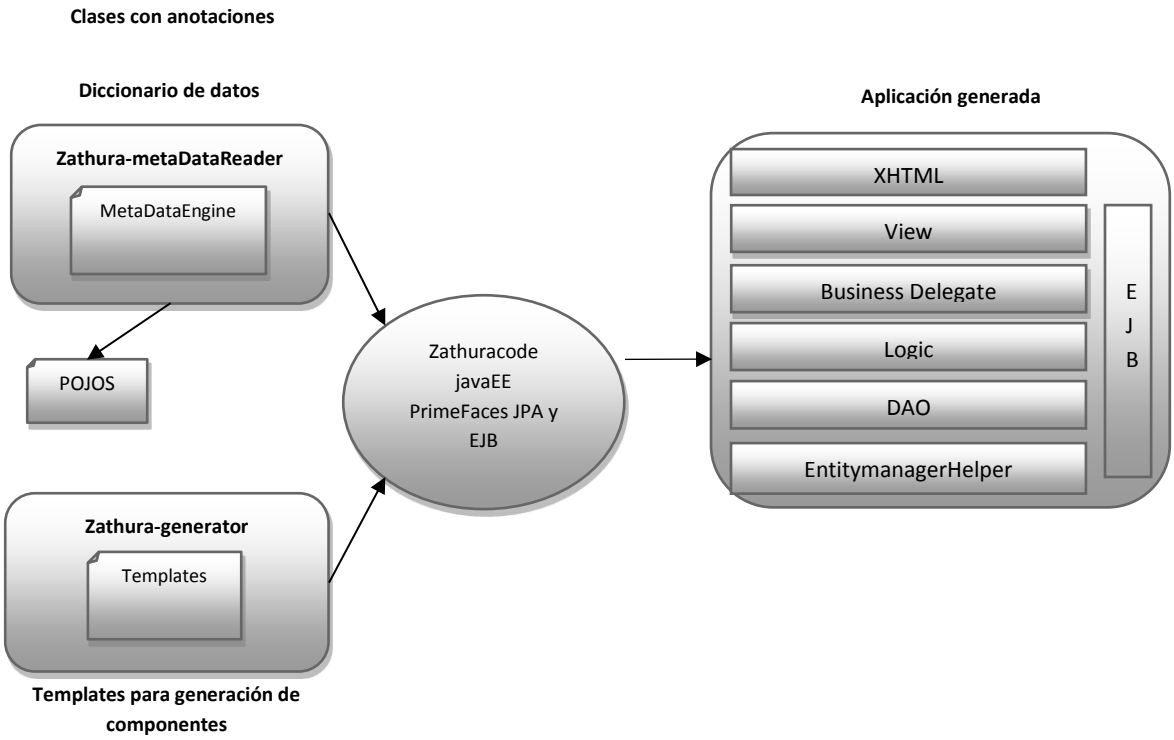


**Ilustración 12 Proceso de generación Zathuracode con capas de Primefaces y JPA con EJB**

A continuación se presenta el diagrama de los componentes involucrados en la arquitectura JavaEE Centric utilizando la tecnología Primefaces, JPA y EJB. En la siguiente ilustración se visualiza los componentes del lado del cliente y los componentes del lado del servidor.

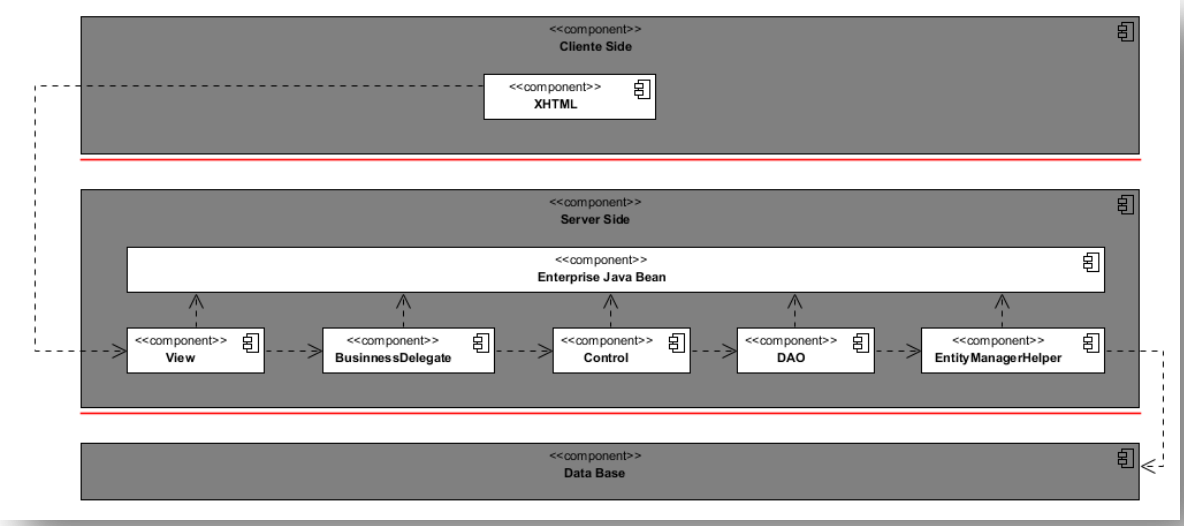


**Ilustración 13 Diagrama de componentes utilizados en la arquitectura JavaEE web centric utilizando Primefaces, JPA y EJB**

# 14. CARACTERÍSTICAS DE LA VERSIÓN 4.0

A continuación se mencionan los aspectos relevantes de la versión 3.0 de Zathuracode, y los aspectos relevantes que se adicionaron en la versión 4.0.

**ZATHURACODE 3.0**

- Arquitectura JAVA EE 5 Web Centric JPA 1.0 Primefaces 3.2
- Arquitectura JAVA EE 5 Web Centric JPA 1.0 Spring 2.1.2 Primefaces 3.2
- Arquitectura JAVA EE 5 Web Centric JPA 1.0 Icefaces 1.8
- Arquitectura JAVA EE 5 Web Centric JPA 1.0 Spring 2.1.2 Icefaces 1.8
- Arquitectura JAVA EE 5 Web Centric HibernateCore Primefaces 3.2
- Arquitectura JAVA EE 5 Web Centric HibernateCore Spring 2.1.2 Primefaces 3.2
- Arquitectura JAVA EE 5 Web Centric HibernateCore Icefaces 1.8
- Arquitectura JAVA EE 5 Web Centric HibernateCore Spring 2.1.2 Icefaces 1.8

**ZATHURACODE 4.0**

- Arquitectura JAVA EE 6 Web Centric JPA 2.0 Spring 3.2.3 Primefaces 3.5
- Arquitectura JAVA EE 6 Web Centric HibernateCore 4.2.3 Spring 3.2.3 Primefaces 3.5
- Arquitectura JAVA EE 6 Web Centric JPA 2.0 EJB 3.1 Primefaces 3.5
- Componente DAO Genérico en la construcción de las arquitecturas de la versión 4.0
- Maven 3.0 incorporado en la generación de las arquitecturas de la versión 3.0 y 4.0

# 15. CONCLUSIONES

- La inclusión de nuevas arquitecturas que permitan aumentar el catálogo de opciones de generación de código continuaran haciendo de Zathuracode mucho más robusta y completa al momento de elegir una herramienta que nos ayude a minimizar tiempo y disminuir costo en los proyectos de desarrollo de aplicaciones.
- Adaptabilidad de nuevas herramientas como MAVEN al plugin, gracias a la arquitectura del generador que consta de 3 capas, las cuales permitieron realizar un trabajo más simple y demostrando que el generador sigue cumpliendo objetivos de mantenibilidad y escalabilidad.
- La selección de una arquitectura que esté acorde a las necesidades, puede determinar el éxito o el fracaso de un proyecto, el cumplimiento de plazos, costos y calidad, por ende, al tener una gama más amplia en la cantidad de arquitecturas y la especificación de dichas necesidades permitirán a los usuarios del plugin un mayor campo de acción para el desarrollo de proyectos Java Web.
- La herramienta MAVEN facilita la gestión de desarrollo del proyecto, haciéndolo portable para cualquier desarrollador, al incluir las dependencias un el POM.xml y no el carpeta de librerías del proyecto.
- El conjunto de funciones y procedimientos que ofrece EJB Lite cuando se piensa en una arquitectura basada en componentes es la reusabilidad, modularidad e interoperabilidad para la construcción de aplicaciones web.
- El framework de Spring es una gran herramienta para la inyección de dependencias y la unificación de código, de esta forma se vuelve parte indispensable de las arquitecturas de Zathuracode, puesto que estas van encaminadas a proyectos empresariales, los cuales tienden a sufrir modificaciones a lo largo de su desarrollo.
- El éxito del desarrollo de cualquier sistema es la evaluación de las necesidades que se deben satisfacer y teniendo esto claro determinar la mejor arquitectura para el desarrollo del mismo.
- Una arquitectura mal definida lleva al reproceso y en algunas ocasiones hasta el fracaso de cualquier aplicación en desarrollo.
- La construcción de aplicaciones mediante Zathuracode reduce el tiempo de desarrollo de aplicaciones Web que usen POJOs con anotaciones del estándar de JPA v2.0 (JSR 220), debido a que todo el trabajo repetitivo como es la creación, modificación y borrado de los datos es hecho por Zathuracode.
- Mediante la creación de componentes de generación de código es posible disminuir los tiempos de desarrollo de software. Al iniciar un proyecto no es necesario construir sus componentes desde cero; esto conlleva a disminuir costos en el desarrollo de software para las empresas.
- La estandarización de arquitecturas para las aplicaciones hace que los costos de mantenimiento sean más bajos, puesto que el grupo de desarrollo de las casas de software conocen la forma en que se encuentran construidas las aplicaciones.

- Mediante la separación en capas y la distribución de responsabilidades en componentes, el diseño y programación de aplicaciones Web, es posible construir software fácil de mantener y modificar debido a que existen componentes especializados en tareas específicas tales como presentación visual, lógica de negocio y persistencia entre otros, lo cual permite identificar fácilmente fallas en los componentes de las aplicaciones.

## 16. REFERENCIAS Y BIBLIOGRAFÍA


- ANTONIO GONCALVES, Beginning JavaEE 6 Platform with GlassFish 3, From Novice to Professional, Editorial APRESS, 2009, pp 167-187.

- Jonathan Wetherbee; Chirag Rathod; Raghu Kodali; Peter Zadrozny, Beginning EJB 3 JavaEE 7 Edtition, Editorial APRESS, 2013.

- Definición de los Singleton Sesion Bean: http://docs.oracle.com/javaee/1.4/tutorial/doc/EJBConcepts4.html [Fecha última consulta Marzo 2014].

- http://maven.apache.org/index.html [Fecha última consulta: 06 de Noviembre de 2013]

- Apache Maven 3 Cookbook - Srirangan_1032.pdf

- Zathura Code 3_0.doc – 2012


## 17. TRABAJOS FUTUROS

- Diseñar y construir los componentes necesarios, para que las aplicaciones construidas por Zathuracode, puedan ser generadas en proyecto Full Enterprise.
- Diseñar y Construir el plugin para el IDE Netbeans debido a que es otro de las herramientas para programar en JavaEE usada en el mercado.