ACTUALIZACIÓN DEL GENERADOR DE CÓDIGO ZATHURACODE 5.0.1

JOSE LUIS RODRIGUEZ PARRA

UNIVERSIDAD DE SAN BUENAVENTURA

FACULTAD DE INGENIERÍA

PROGRAMA DE INGENIERÍA DE SISTEMAS

SANTIAGO DE CALI, 2015

ACTUALIZACIÓN DEL GENERADOR DE CODIGO ZATHURACODE 5.0.1

JOSE LUIS RODRIGUEZ PARRA

Informe de Investigación presentado

Para optar por el Título de

Ingeniero de Sistemas

Director

M.Sc. DIEGO ARMANDO GÓMEZ MOSQUERA

Ingeniero de Sistemas

UNIVERSIDAD DE SAN BUENAVENTURA

FACULTAD DE INGENIERÍA

PROGRAMA DE INGENIERÍA DE SISTEMAS

SANTIAGO DE CALI, 201

AGRADECIMIENTOS

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

A mi familia 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 me 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, me apoyó y guió 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.

Tabla de contenido

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.

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

---

[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.

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 la versión Zathuracode 4.0.1 se incluye el soporte para generación de código bajo proyectos administrados con la herramienta MAVEN, se incluyen dos nuevas arquitecturas, las cuales implementan Spring Framework 3.2.3, Primefaces 4.0, Hibernate 4.2.3, Spring Security 3.2.2, además de una Pantalla de autenticación y corrección de Bugs.

En la versión Zathuracode 5.0.0 se realiza una actualización de los Frameworks Spring Security 4.1.1, Primefaces 5.1, Hibernate 4.3.6, Spring Security 3.2.5, Jamon 2.79, JSF 2.2. En esta versión se soportan solamente tres arquitecturas , las cuales son:

JENDER.
SKYJET.
WALL-J.

Se incluye en la versión 5 un mejor soporte a errores en generación, mejor soporte en compilación de código con Java SE8, Logs en consola para verificar la generación de código, validación de instalación correcta de JDK.

En su ultima versión Zathuracode 5.0.1 se agrega el soporte para IDEs variados como MyEclipse 2014, MyEclipse 2015, Eclipse Indigo, Luna, Kepler, Spring Tools Suite, se hace actualización de los Frameworks implicados, Spring Framework 4.1.5, Hibernate

4.1.5, Spring Security 3.2.6, se incluyen mejoras en el código generado a partir de la declaración de slf4j en todas las clases.

Zathuracode 5.0.1 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, con la cual se reemplazara la capa de presentación de Primefaces manejada por la capa de Backing Beans por una capa de servicios REST conectados a una nueva capa de presentación en HTML5 y Angular JS, además incluirá un soporte para generación de código en proyectos administrados por la herramienta GRADLE. Se espera 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.

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, conozca, aprenda y realice 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 áreas 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[4], 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.

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

Es importante saber que la versión más reciente es la del año 2015 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 en 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 con las que se encuentran en el generador de componentes de software hoy en día. De igual manera Zathuracode es una herramienta consolidada, es así como surge el 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 de los DAO y la presentación de la aplicación sean 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 6.0 integra una herramienta administradora de proyectos como GRADLE.

Incluir un administrador de proyectos de desarrollo de software como GRADLE 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 el cambio en la presentación bajo las tecnologías de HTML5 y Angular JS se logrará una gran fluidez entre la capa de servicios REST y el navegador, además de un control mas amplio en la capa de presentación con todo lo que provee Angular JS.

4. OBJETIVO GENERAL

Actualizar la versión del generador de componentes de software Zathuracode actualmente en su versión 5.0.1 a su versión 6.0 adicionando a las 2 arquitecturas Jender y SkyJet las capas de presentación y servicios REST además de la integración con la herramienta GRADLE en cada una de ellas.

5. OBJETIVOS ESPECÍFICOS

- Investigar sobre las tecnologías implicadas y como lograr la integración con el plugin en las dos arquitecturas Jender y SkyJet.
- Diseñar los componentes que permitirán la integración de las nuevas tecnologías con las dos arquitecturas del plugin Jender y SkyJet,
- Construir los componentes necesarios para la integración de las nuevas tecnologías con las dos arquitecturas del plugin Jender y SkyJet.
- Implementar la integración de las tecnologías con las dos arquitecturas seleccionadas del plugin.
- Probar las dos arquitecturas del plugin con la integración de estas nuevas tecnologías en la generación de código para un proyecto.

6. ALCANCE DEL PROYECTO

Los siguientes son los ítems que el proyecto contemplará:

1. Generar los artefactos de software desde la herramienta Zathuracode que se integren con la herramienta de gestión de proyectos GRADLE.
2. Generar los artefactos de software desde la herramienta Zathuracode que se integren con la tecnología Angular JS.
3. Generar los artefactos de software desde la herramienta Zathuracode que se integren con HTML5.
4. Refinar e implementar mecanismos, que permitan corregir bug´s encontrados en Zathuracode.
5. Entregar un manual de instrucción bajo el cual las pantallas en la capa de presentación generadas puedan ser usadas para crear una aplicación con el Framework, Apache Cordova.

Los siguientes son los ítems que el proyecto no contemplará

1. Creación de un proyecto bajo la tecnología Apache Cordova

7. 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[5] .

Así como se menciona en un artículo publicado por la casa de desarrollo Ssquare-SA[6] 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.

Algunos generadores de código que también se encargar de facilitar el trabajo de las empresas para su optimización de código son:

SCULPTOR

Es una herramienta de código abierto para aumentar la productividad en la codificación, que aplica conceptos del diseño guiado por el dominio y lenguaje especifico del dominio para la generación de código de alta calidad y una configuración Java desde una especificación textual.

---

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

[6] 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.

BITE BUDDY

Es una biblioteca de código para la generación de las clases Java durante el tiempo de ejecución y sin la ayuda de un compilador.

SPRING FUSE

Es un generador de código de origen Francés, el cual se encuentra al servicio desde el año 2005. Spring Fuse 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 proyectos por medio de esta herramienta se realiza desde la web, retornándole al cliente el paquete del proyecto codificado.

ZATHURACODE 5.0.1

Zathuracode es una herramienta desarrollada en Cali, Colombia por un grupo de desarrollo patrocinado por la compañía Vortexbird, el plugin ya cuenta con 7 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 qué arquitectura ejecutar el proceso de implementación o la fase de desarrollo del software.

Retomando la investigación realizada bajo Zathuracode. Sabemos que 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 Frameworks 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:

- WallJ: implementando Maven como gestor de dependencias e involucrando los Frameworks Primefaces 5.1, EJB 3.1, JPA 2.1.
- SkyJet: implementando Maven como gestor de dependencias e involucrando los Frameworks Primefaces 5.1, Spring 4.1, Spring Security 3.2.6, JPA 2.1.
- Jender: implementando Maven como gestor de dependencias e involucrando los Frameworks Primefaces 5.1,Spring 4.1.5, Spring Security 3.2.6, Hibernate 4.3.8.

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:

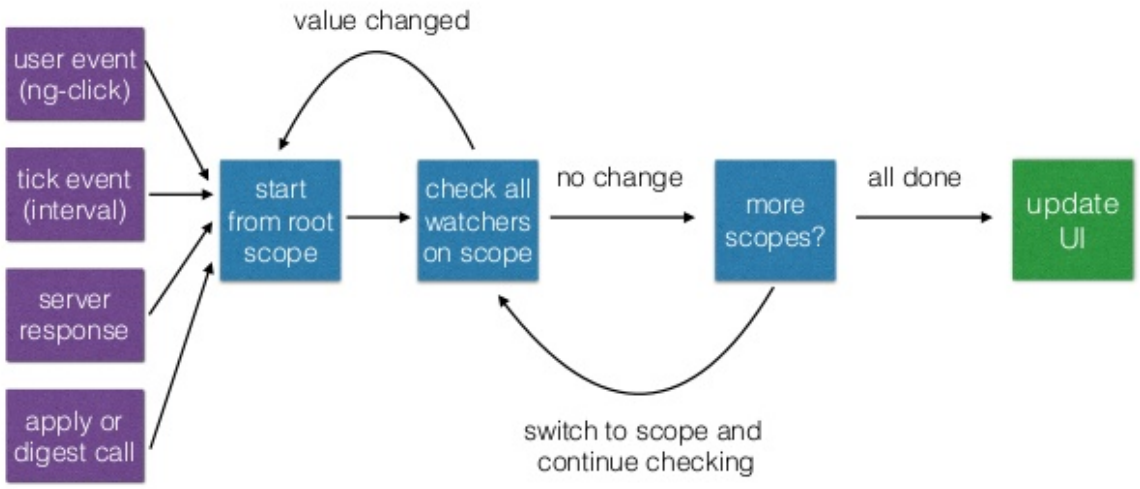
- Contenedor de aplicaciones Tomcat
- Servidor de aplicaciones GlassFish
- Servidor de aplicaciones JBoss

8. MARCO TEÓRICO

Angular JS

Angular JS es un Framework de JavaScript de código abierto, mantenido por Google, que ayuda con de lo que se conoce como aplicaciones de una sola página, Su objetivo es aumentar aplicaciones basadas en navegador con capacidad de Modelo Vista Controlador(MVC), en un esfuerzo para hacer que el desarrollo y las pruebas sean más fáciles.

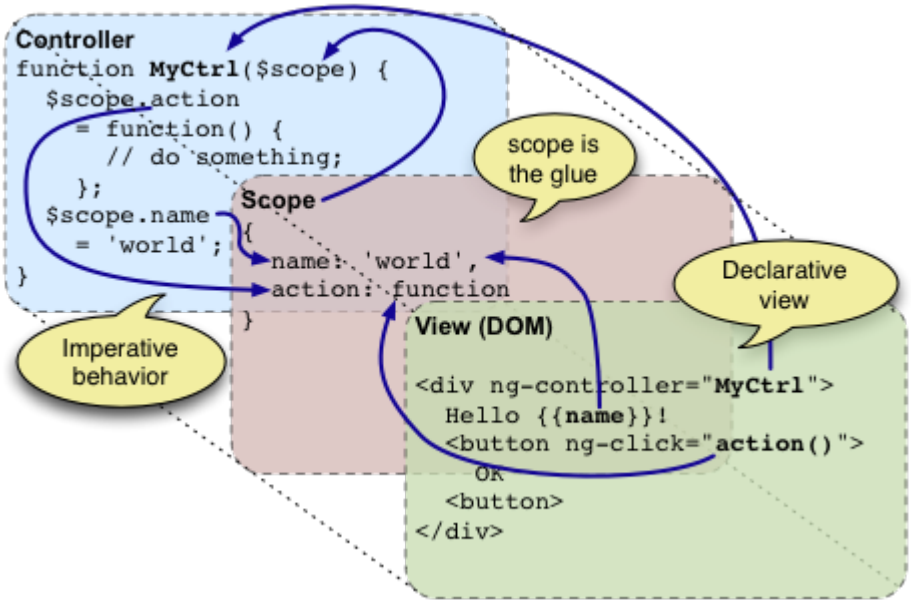
La biblioteca lee el HTML que contiene atributos de las etiquetas personalizadas adicionales, entonces obedece a las directivas de los atributos personalizados, y une las piezas de entrada o salida de la página a un modelo representado por las variables estándar de JavaScript. Los valores de las variables de JavaScript se pueden configurar manualmente, o recuperados de los recursos JSON estáticos o dinámicos.



Está construido en torno a la creencia de que la programación declarativa es la que debe utilizarse para generar interfaces de usuario y enlazar componentes de software, mientras que la programación imperativa es excelente para expresar la lógica de negocio.

Lo beneficioso de este Framework es que adapta y amplía el HTML tradicional para servir mejor contenido dinámico a través de un “data-binding” bidireccional que permite la sincronización automática de modelos y vistas como resultado, Angular JS pone menos énfasis en la manipulación del DOM y mejora la testeabilidad y el rendimiento

Angular JS sigue el patrón MVC de ingeniería de software y alienta la articulación flexible entre la presentación, datos y componentes lógicos. Con el uso de la inyección de dependencias, Angular JS lleva servicios Tradicionales del lado del servidor, tales como controladores dependientes de la vista, a las aplicaciones Web del lado del cliente. En consecuencia, gran parte de la carga en el Backend se reduce , lo que conlleva a aplicaciones web mucho mas ligeras.



HTML5

Es la quinta revisión importante del lenguaje básico de la World Wide Web, HTML. HTML5 especifica dos variantes de sintaxis para HTML: una clásica HTML(text/html), conocida como HTML5, y una variante XHTML conocida como sintaxis XHTML5 que deberá servirse con sintaxis (application/xhtml+xml), Esta es la primera vez que HTML y XHTML se han desarrollado en paralelo. La versión definitiva de la quinta revisión del estándar se publico en octubre del 2014.

Al no ser reconocido en viejas versiones de navegadores por sus nuevas etiquetas, se recomienda al usuario común actualizar su navegador a la versión mas nueva para poder disfrutar de todo el potencial que provee HTML5.

HTML5 establece una serie de nuevos elementos y atributos que reflejan el uso típico de los sitios web moderno. Algunos de ellos son técnicamente similares a las etiquetas y , pero tienen un significado semántico.

Entre las novedades que incluye este lenguaje incorpora etiquetas (canvas 2D y 3D, audio , video) con codecs para mostrar los contenidos multimedia, etiquetas para manejar grandes conjuntos de datos: Datagrid, Details, Menú y Command que permiten generar tablas dinámicas que pueden filtrar, ordenar y ocultar contenido en cliente, mejoras en los formularios, nuevos tipos de datos (eMail, number, url, datetime).

REST

Transferencia de estrado representacional (Representational State Transfer) o REST es un estilo de arquitectura software para sistemas hipermedia distribuidos como la World Wide Web. El termino se origino en el año 2000, en una tesis doctoral sobre la web escrita por Roy Fielding, uno de los principales autores de la especificación del protocolo http y ha pasado a ser ampliamente utilizado por la comunidad de desarrollo.

Inicialmente el termino REST se refería originalmente a un conjunto de principios de arquitectura, actualmente se usa en el sentido mas amplio para describir cualquier interfaz entre sistemas que utilice directamente HTTP para obtener datos o indicar la ejecución de operaciones sobre los datos, en cualquier formato (XML, JSON, etc.).

Los sistemas que siguen los principios REST se llaman con frecuencia RESTful.

REST afirma que la web ha disfrutado de escalabilidad como resultado de una serie de diseños fundamentalmente clave:

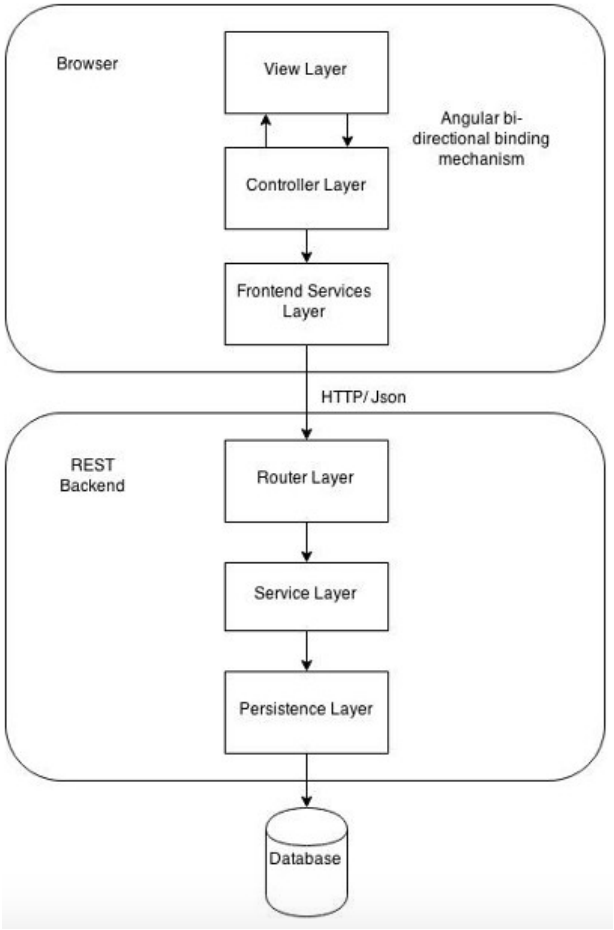
- Un protocolo cliente/servidor sin estado: cada mensaje http contiene toda la información necesaria para comprender la petición. Como resultado, ni el cliente ni el servidor necesitan recordar ningún estado de las comunicaciones entre mensajes, sin embargo, en la practica muchas aplicaciones basadas en HTTP utilizan cookies y otros mecanismos para mantener el estado de la sesión.

- Un conjunto de operaciones bien definidas que se aplican a todos los recursos de información: HTTP en si define en conjunto pequeño de operaciones, las mas importantes son POST, GET, PUT, DELETE con frecuencia estas operaciones se equiparan a las operaciones CRUD en bases de datos.

- Una sintaxis universal para identificar los recursos. En un sistema REST, cada recurso es direccionable únicamente a través de su URI.

- El uso de hipermedios, tanto para la información de la aplicación como para las transiciones de estado de la aplicación: la representación de este estado en un sistema REST son típicamente HTML o XML como resultado de esto, es posible navegar de un recurso REST a muchos otros, simplemente siguiendo enlaces sin requerir el uso de registros u otra infraestructura adicional.

La petición puede ser transmitida por cualquier numero de conectores (por ejemplo clientes, servidores, caches, túneles, etc.) pero cada uno lo hace sin “ver mas allá” de su propia petición (lo que se conoce como Stateless(sin estado)).



GRADLE

Es una herramienta de automatización usando los conceptos de Apache Ant y Apache Maven y introduce un lenguaje específico de dominio basado en Groove en lugar del tradicional formulario en CML para declarar la configuración del proyecto. Gradle usa un grafo dirigido a cíclico para determinar el orden en que las tareas se pueden ejecutar.

Gradle fue diseñado para proyectos los cuales pueden ir creciendo, y soporta incrementalmente los cambios determinando cuales partes del grafo deben ser actualizadas para que ninguna tarea independiente tenga que encargarse de re-ejecutar la actualización del grafo.

Los plugins iniciales fueron enfocados para desarrollo y despliegue bajo los lenguajes Java, Groovy y Scala, sin embargo, en el camino se ha añadido soporte a mas lenguajes.

En las siguientes imágenes se ilustra la simplicidad de GRADLE frente a MAVEN:

- MAVEN

[pom.xml]

```xml
<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>com.technologyconversations</groupId>
    <artifactId>java-build-tools</artifactId>
    <packaging>jar</packaging>
    <version>1.0</version>

    <dependencies>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.11</version>
        </dependency>
        <dependency>
            <groupId>org.hamcrest</groupId>
            <artifactId>hamcrest-all</artifactId>
            <version>1.3</version>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>2.3.2</version>
            </plugin>
        </plugins>
    </build>

</project>
```

[pom.xml]

```
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-checkstyle-plugin</artifactId>
    <version>2.12.1</version>
    <executions>
        <execution>
            <configuration>
                <configLocation>config/checkstyle/checkstyle.xml</configLo
                <consoleOutput>true</consoleOutput>
                <failsOnError>true</failsOnError>
            </configuration>
            <goals>
                <goal>check</goal>
            </goals>
        </execution>
    </executions>
</plugin>
<plugin>
    <groupId>org.codehaus.mojo</groupId>
    <artifactId>findbugs-maven-plugin</artifactId>
    <version>2.5.4</version>
    <executions>
        <execution>
            <goals>
                <goal>check</goal>
            </goals>
        </execution>
    </executions>
</plugin>
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-pmd-plugin</artifactId>
    <version>3.1</version>
    <executions>
        <execution>
            <goals>
                <goal>check</goal>
            </goals>
        </execution>
    </executions>
</plugin>
```

GRADLE

[build.gradle]

```
apply plugin: 'java'
apply plugin: 'checkstyle'
apply plugin: 'findbugs'
apply plugin: 'pmd'

version = '1.0'

repositories {
    mavenCentral()
}

dependencies {
    testCompile group: 'junit', name: 'junit', version: '4.11'
    testCompile group: 'org.hamcrest', name: 'hamcrest-all', version: '1.3'
}
```

en ambos casos se presenta exactamente la misma configuración, es notable la diferencia ya que la simplicidad de GRADLE permite una mayor productividad, dejando más tiempo al desarrollador para ocuparse de lo realmente importante.

9. 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[7]'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, Componente que contiene el modelo de referencia de la tabla, y el diccionario de datos que contiene información de atributos y referencias mapeadas.
- **MetaDataReader:** Componente que usa el patrón de diseño **Factory**[8], para administrar las instancias del componente **MetaDataReader** al momento de realizar el mapeo objeto relacional, 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.
- **MetaDataEngine:** Componente que contiene los métodos orientados a la carga de las entidades con JPA.

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.

- **GeneratorFactory:** Componente encargado de crear el Factory que a su vez crea una instancia del componente generador del código.
- **GeneratorModel:** Componente encargado de hacer la definición del generador.
- **RobotJender:** Componente encargado de generar código bajo las tecnologías que aplican para la arquitectura Jender.

---

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

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

- **RobotSkyJet:** Componente encargado de generar código bajo las tecnologías que aplican para la arquitectura Jender.
- **RobotWallJ:** Componente encargado de generar código bajo las tecnologías que aplican para la arquitectura Jender.
- **Utilities:** Componente de utilidades que contiene variables necesarias para el generador.

3. REVERSE

Es la capa encargada de hacer los POJOS según el sistema gestor de bases de datos que se vaya a manejar para el proyecto que se quiera generar código.

- **IZathuraReverseEngineering:** es la interfaz encargada de realizar los POJOS teniendo en cuenta las propiedades de conexión y el numero de tablas que tenga la base de datos.

10. INTEGRACIÓN DE LA HERRAMIENTA ADMINISTRADORA DE PROYECTOS GRADLE A ZATHURACODE

Para la integración de GRADLE a Zathuracode, se modifico la capa Generator.

La capa de Generator Robot contiene los generadores de templates para cada una de las arquitecturas, 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 dominio, del DAO, de la lógica, de los delegados y de la vista, tienen sus respectivos templates.

Para integrar GRADLE es necesario crear un template llamado build.gradle, 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 GRADLE, el cual es utilizado en la capa Engine.

En la capa Engine se realiza la validación de qué si el proyecto es GRADLE 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 GRADLE con la configuración del proyecto y la dependencia de librerías.

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

1. Creación del template build.gradle.
2. Adicionar el build.gradle 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 GRADLE.
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 GRADLE se usa crea el archivo build.gradle.

11. CAMBIOS EN LOS COMPONENTES DE ZATHURACODE PARA LA GENERACIÓN DE LAS NUEVAS ARQUITECTURAS.

La generación de código bajo las dos nuevas arquitecturas Jerder JS y SkyJet JS que implican nuevos frameworks tanto en la parte del Front-End como el BackEnd los cuales han sido descritos con anterioridad, generan cambios en el proyecto Zathuracode en ciertos niveles.

Zathuracode en su arquitectura se divide en 3 Capas las cuales son:

- Generator
- Metadata
- Reverse

En el numeral 9 de este documento se explica qué papel cumple cada uno de los paquetes en la generación de código, sin embargo, no se muestra en qué paquetes se aplican los cambios para que las nuevas arquitecturas puedan ser integradas al plugin, por lo cual a continuación se describen los cambios que implican en la arquitectura las nuevas arquitecturas del plugin.

Jender JS

Jender JS toma su nombre de su homólogo Jender, aplicando los nuevos Frameworks en la capa de presentación y administración de dependencias en la generación de código, los cambios que toma ZathuraCode en su Arquitectura son:

- Creación de un nuevo paquete que contiene al robot Jender JS (org.zcode.generator.robot.jenderJS), el cual contiene las siguientes clases Java
    - IStringBuilder: Interfaz que contiene la firma de los métodos para la clase StringBuilder.
    - IStringBuilderForId: Interfaz que contiene la firma de los métodos para la clase StringBuilderForId.
    - IZathuraJenderJsTemplate: Interfaz que contiene la firma de los métodos para la clase JenderJs.
    - JenderJs: Es la clase principal del paquete que se encarga de la generación de código bajo los frameworks por los cuales está definida.

    - StringBuilder: Es la clase dedicada a analizar la metadata de las entidades que vienen de la base de datos para hacer luego la generación de sus clases Java.
    - StringBuilderForid: Es la clase dedicada a analizar las llaves primarias de las entidades para que puedan generarse los métodos que permiten el encapsulamiento de dichas clases
    - Utilities:Clase que proporciona utilidades específicas a las StringBuilder, StringBuilderForId y JenderJs.
- Agregación de la descripción del nuevo robot generador de código al archivo xml (zathura-generator-factory-config.xml), el cual contiene la descripción de cada uno de los robots. Este archivo contiene la siguiente estructura para la descripción del robot:
    - <name>: Nombre del robot
    - <gui-name>: Nombre el cual aparecerá en la pantalla cuando el usuario quiera generar con el determinado robot.
    - <class>: Ruta de la clase Java la cual contiene el robot.
    - <persistence>: el tipo de persistencia sobre la cual el robot genera el código
    - <zathuraVersion>: la versión de zathura en la cual fue creado
    - <description>: contiene la descripción las tecnologías las cuales usa el robot además de las imágenes de cada una de las tecnologías implicadas en la generación del código.
- Agregación de la carpeta Jender JS en la ubicación (/templates), la cual contiene todos los templates desarrollados con la ayuda de la tecnología Apache Velocity necesarios para la generación de código. Es válido aclarar que Jender JS en algunos de sus templates tiene cierta similitud con los de su robot homólogo Jender por lo cual a continuación se describen algunos de los templates específicos que fueron agregados:
    - mvc-dispatcher-servlet.xml.vm: archivo el cual proporciona al código generado la capacidad de tomar y entender las peticiones que se hacen desde las páginas HTML a los controllers administrados con Spring.
    - HTMLSpringHibernateAngular.vm: es el template el cual permite generar cada una de las páginas HTML.

Parte de estos cambios son evidenciados en el documento anexo (JenderJSDiagramClass.png) en el cual se ilustra el diagrama de clases del robot y además de esto se compara con el diagrama de clases del robot Jender.

Skyjet JS

Skyjet JS toma su nombre de su homólogo Skyjet , aplicando los nuevos Frameworks en la capa de presentación y administración de dependencias en la generación de código, los cambios que toma ZathuraCode en su Arquitectura son:

- Creación de un nuevo paquete que contiene al robot Skyjet JS (org.zcode.generator.robot.skyjetJS), el cual contiene las siguientes clases Java
    - IStringBuilder: Interfaz que contiene la firma de los métodos para la clase StringBuilder.
    - IStringBuilderForId: Interfaz que contiene la firma de los métodos para la clase StringBuilderForId.
    - IZathuraSkyJetJsTemplate: Interfaz que contiene la firma de los métodos para la clase JenderJs.
    - SkyJetJs: Es la clase principal del paquete que se encarga de la generación de código bajo los frameworks por los cuales está definida.
    - StringBuilder: Es la clase dedicada a analizar la metadata de las entidades que vienen de la base de datos para hacer luego la generación de sus clases Java.
    - StringBuilderForid: Es la clase dedicada a analizar las llaves primarias de las entidades para que puedan generarse los métodos que permiten el encapsulamiento de dichas clases
    - Utilities:Clase que proporciona utilidades específicas a las StringBuilder, StringBuilderForId y SkyJetJs.
- Agregación de la descripción del nuevo robot generador de código al archivo xml (zathura-generator-factory-config.xml), el cual contiene la descripción de cada uno de los robots. Este archivo contiene la siguiente estructura para la descripción del robot:
    - <name>: Nombre del robot
    - <gui-name>: Nombre el cual aparecerá en la pantalla cuando el usuario quiera generar con el determinado robot.
    - <class>: Ruta de la clase Java la cual contiene el robot.

    - <persistence>: el tipo de persistencia sobre la cual el robot genera el código
    - <zathuraVersion>: la versión de zathura en la cual fue creado
    - <description>: contiene la descripción las tecnologías las cuales usa el robot además de las imágenes de cada una de las tecnologías implicadas en la generación del código.
- Agregación de la carpeta SkyJetJs en la ubicación (/templates), la cual contiene todos los templates desarrollados con la ayuda de la tecnología Apache Velocity necesarios para la generación de código. Es válido aclarar que Skyjet JS en algunos de sus templates tiene cierta similitud con los de su robot homólogo Skyjet por lo cual a continuación se describen algunos de los templates específicos que fueron agregados:
    - mvc-dispatcher-servlet.xml.vm: archivo el cual proporciona al código generado la capacidad de tomar y entender las peticiones que se hacen desde las páginas HTML a los controllers administrados con Spring.
    - HTMLSpringHibernateAngular.vm: es el template el cual permite generar cada una de las páginas HTML.

Parte de estos cambios son evidenciados en el documento anexo (SkyjetJSDiagramClass.png) en el cual se ilustra el diagrama de clases del robot y además de esto se compara con el diagrama de clases del robot Skyjet.

Al agregar estos dos nuevos robots a la arquitectura del plugin no se ve afectado ninguno de los anteriores robots ya que la modularidad del plugin permite crear múltiples robots sin que ninguno de los anteriormente creados tenga conflictos con los nuevos.

12. DIFERENCIAS ENTRE LOS FRAMEWORKS ANTERIORES Y LOS NUEVOS FRAMEWORKS IMPLEMENTADOS EN LAS VIEJAS Y NUEVAS ARQUITECTURAS

Como se ha dejado claro a lo largo del documento las anteriores arquitecturas Jender y SkyJet generan código en la capa de presentación bajo el framework Primefaces , y ahora Jender JS y SkyJet JS generan código en la capa de presentación bajo el framework Angular JS con HTML5.

Por lo cual es de interés recalcar sus diferencias.

Primefaces

JSF llegó más o menos al mismo tiempo que el Ajax explotó en la escena de desarrollo web hace una década. La versión inicial de JSF no fue diseñado con el Ajax en mente, pero en cambio se entiende como un modelo completo de solicitud / respuesta por parte de la página.

En este modelo, existe un árbol DOM-como de los componentes que representan la interfaz de usuario en la memoria, pero existe este árbol sólo en el lado servidor.

El servidor de Vista luego se convirtió de ida y vuelta a HTML, CSS y Javascript, tratando el navegador sobre todo como una plataforma de representación sin control de estado y limitado sobre lo que está pasando.

Las páginas se generan mediante la conversión de la representación del servidor para HTML, CSS y Javascript a través de un conjunto de clases especiales llamados representadores, antes de enviar la página al usuario.

¿Cómo funciona JSF?

El usuario luego interactuar con la página y enviar de vuelta una acción típicamente a través de un HTTP POST, y luego un ciclo de vida del lado del servidor se activa con el controlador JSF, que restaura el árbol de vista, se aplica a los nuevos valores a la vista y los valida, actualizaciones el modelo de dominio, invoca la lógica de negocio y hace volver una nueva vista.

A continuación, el marco se desarrolló en JSF 2 para el soporte Ajax nativo y desarrollo web sin estado, pero el enfoque principal de generar el código HTML en el navegador de un modelo de servidor se mantuvo.

¿Cómo se compara angular de JSF ?

La diferencia principal es que en el diseño de Angular JS están el Modelo, la Vista y el Controlador que fueron trasladados del servidor al navegador.

En angular, las tecnologías de navegación no son vistas como algo que debe evitarse o ocultarse, sino algo que se utilizará en toda la extensión de sus capacidades, para construir algo que es mucho más similar a un cliente pesado swing en lugar de una página web.

Angular no es mandatorio en el sentido de mantener un estado, sin embargo puede mantenerlo a travez de servicios REST con JSON, lo cual el servidor no podría hacer con primefaces.

¿Por qué Angular JS?

En un sencillo ejemplo podremos demostrar su punto de productividad y simplicidad en desarrollo, el ejemplo es una calculadora simple la cual está descrita en el siguiente código

```
<div ng-app="Calculator" ng-controller="CalculatorCtrl">
    <input type="text" ng-model="model.left"> *
    <input type="text" ng-model="model.right"> =
    <span>{{multiply()}}</span>
</div>

angular.module('Calculator', [])
    .controller('CalculatorCtrl', function($scope) {
        $scope.model = {
            left: 10,
            right: 10
        };

        $scope.multiply = function() {
            return $scope.model.left *
                    $scope.model.right;
        }
    });
```

como podemos ver son simplemente 18 líneas de código en las cuales tenemos una calculadora básica y también tenemos el modelo-vista-control descrito de la siguiente manera:

El modelo

```
$scope.model = {
    left: 10,
    right: 10
};
```

El modelo en Angular JS es simplemente un objeto descrito en Javascript este modelo ha sido inyectado en el ámbito(“scope”), los cambios realizados al modelo son instantáneamente identificados por el ámbito e inmediatamente son visibles en la pantalla al usuario.

La Vista

Como se puede apreciar en el código inicial de ejemplo en una de las líneas tenemos la expresión {{multiply()}} la cual es la que se conecta con los controladores para dar solución al mtodo que se ejecuta cuando el usuario ingresa los dos números que le son solicitados.

El Controlador

En el código Javascript que esta abajo de la presentación en HTML5 podemos ver el nombre de un controller ‘CalculatorCtrl’ el cual es el encargado de soportar tanto el modelo como las funciones que interactúan con el modelo.

13. COMPONENTES DE LAS ARQUITECTURAS JAVAEE GENERADAS POR ZATHURACODE

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[9] 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.

En la versión 6.0 los componentes BUSSINESDELEGATE, VIEW Y XHTML podrán ser opcionales ya que esta versión integrará el uso de servicios REST en el componente de CONTROL, y los componentes de VIEW y XHTML podrán ser reemplazados por las paginas en HTML5 con la integración de Angular JS.

[9] XHTML (Extensible Hipertext Markup Languaje)

14. CARACTERÍSTICAS DE LA VERSIÓN 6.0

A continuación se mencionan los aspectos relevantes de la versión 5.0.1 de Zathuracode, y los aspectos relevantes que se adicionaron en la versión 6.0.

ZATHURACODE 5.0.1

- WallJ: implementando Maven como gestor de dependencias e involucrando los Frameworks Primefaces 5.1, EJB 3.1, JPA 2.1.
- SkyJet: implementando Maven como gestor de dependencias e involucrando los Frameworks Primefaces 5.1, Spring 4.1, Spring Security 3.2.6, JPA 2.1.
- Jender: implementando Maven como gestor de dependencias e involucrando los Frameworks Primefaces 5.1,Spring 4.1.5, Spring Security 3.2.6, Hibernate 4.3.8.

ZATHURACODE 6.0

- SkyJet JS: implementando Maven o GRADLE como gestor de dependencias e involucrando los Frameworks Angular JS, HTML5 , Spring 4.1, JPA 2.1.
- Jender JS: implementando Maven o GRADLE como gestor de dependencias e involucrando los Frameworks Angular JS, HTML5, Spring 4.1.5, Hibernate 4.3.8.

En los siguientes diagramas de componentes podemos ver la comparativa entre las dos arquitecturas pues se tiene como base que Jender es el homólogo de Jender JS, y SkyJet es el homólogo de Skyjet JS:

JENDER

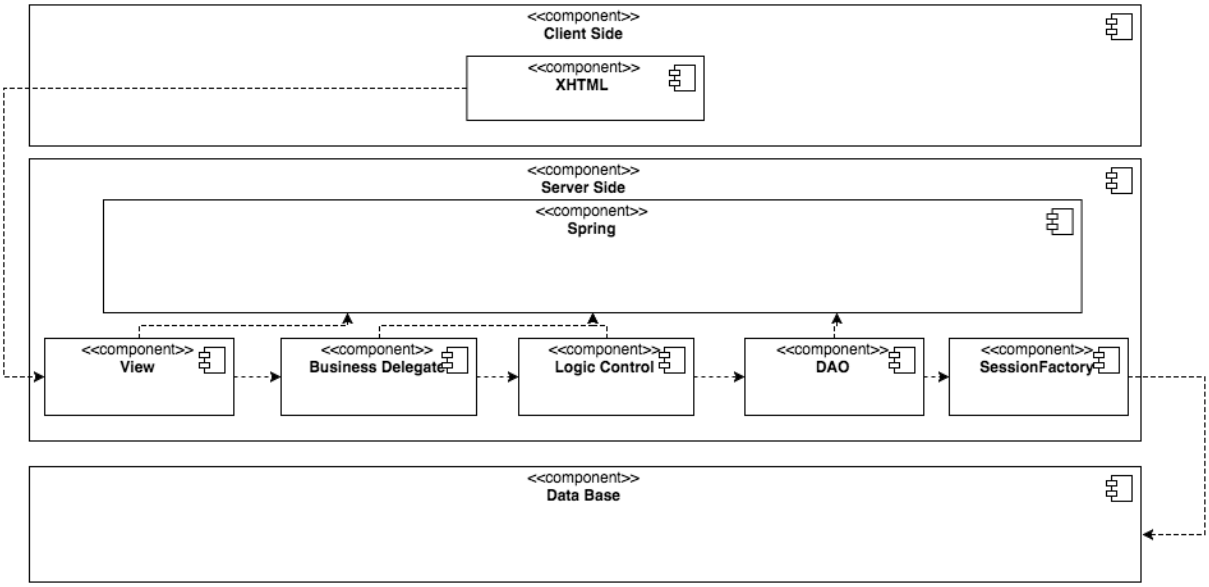


Diagrama de componentes que ilustra la forma en cómo se ejecuta el software generado.

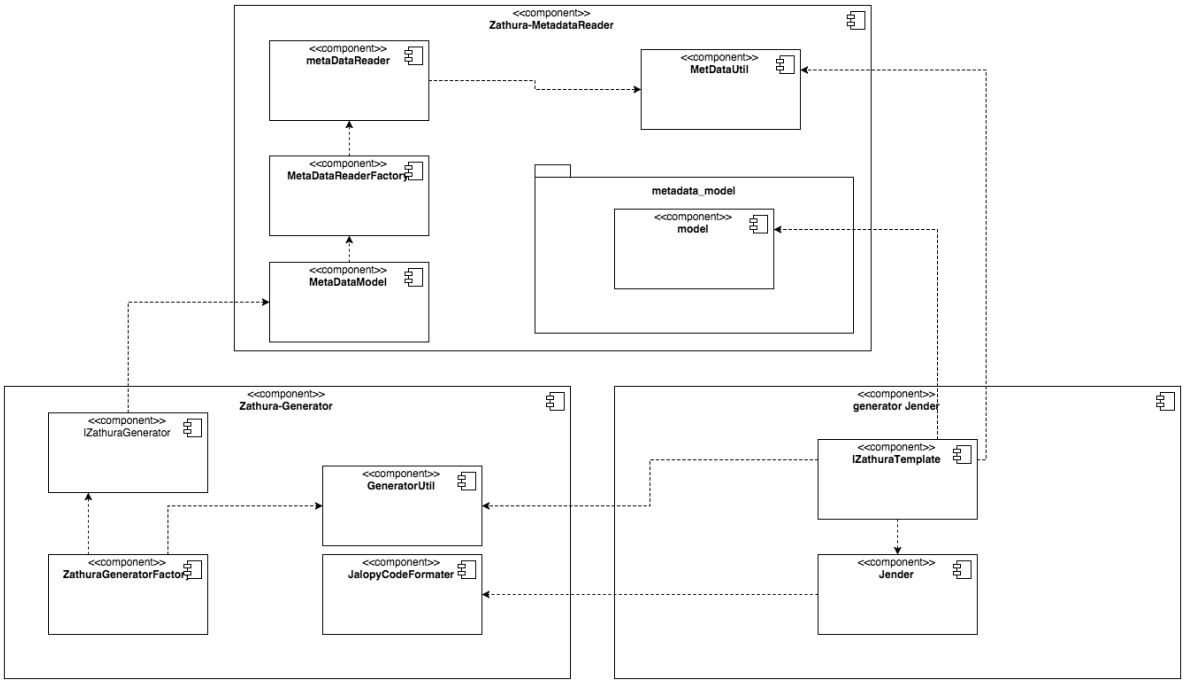


Diagrama de componentes que muestra cómo interactúa el generador cuando se procesa la MetaData que viene de la base de datos para convertirla en la aplicación codificada.

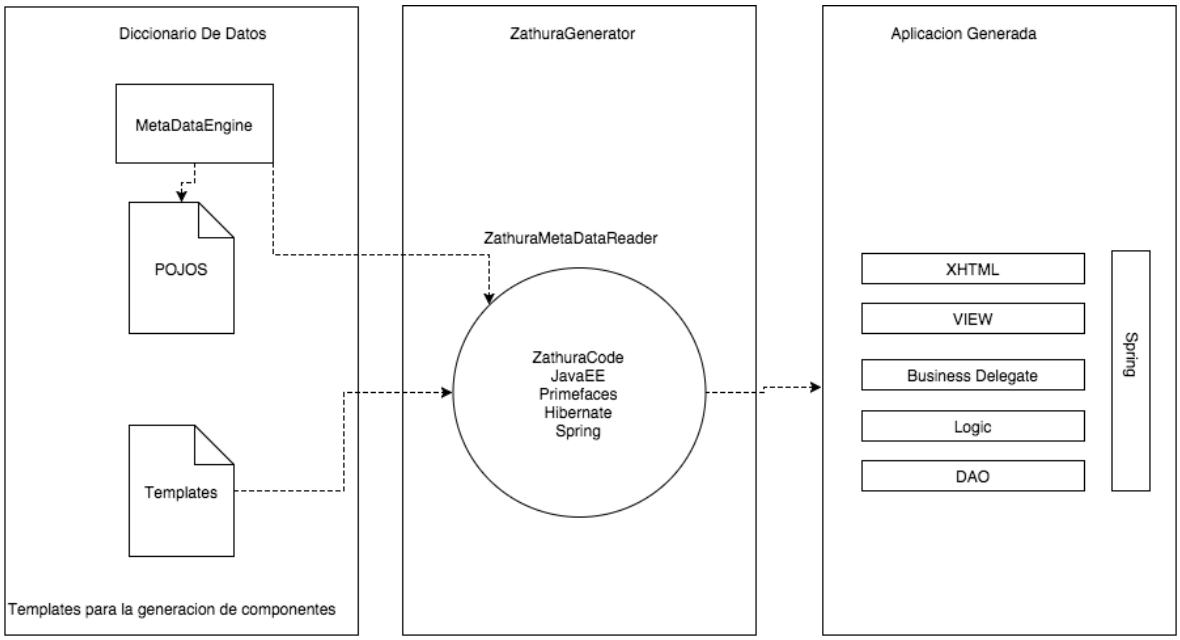


Diagrama que muestra las entradas y las salidas del generador de codigo, ademas muestra sus tecnologías implicadas.

JENDER JS

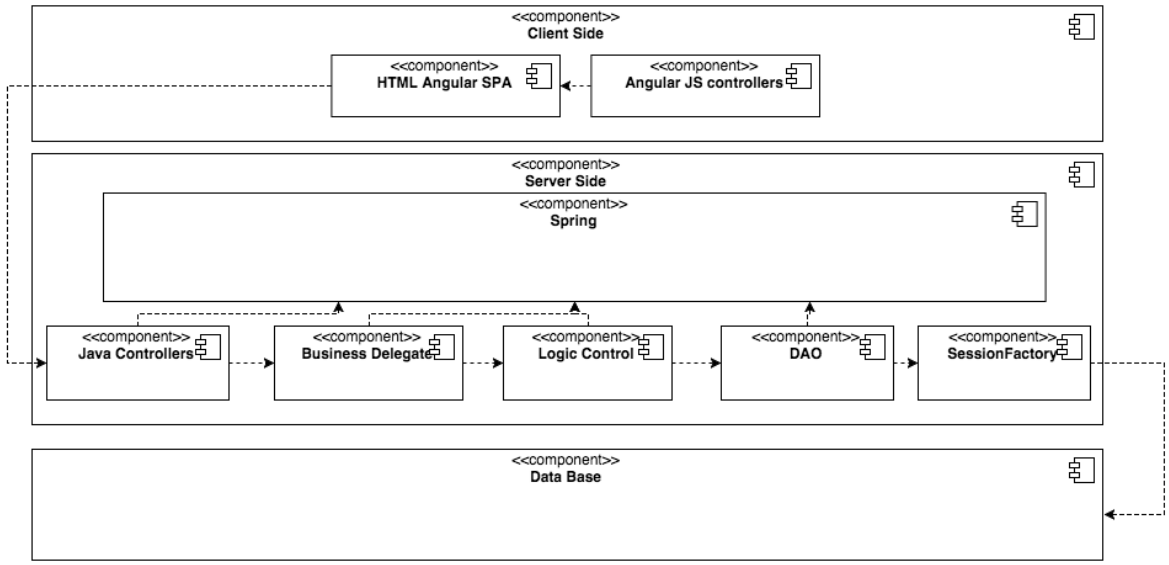


Diagrama de componentes que ilustra la forma en cómo se ejecuta el software generado.

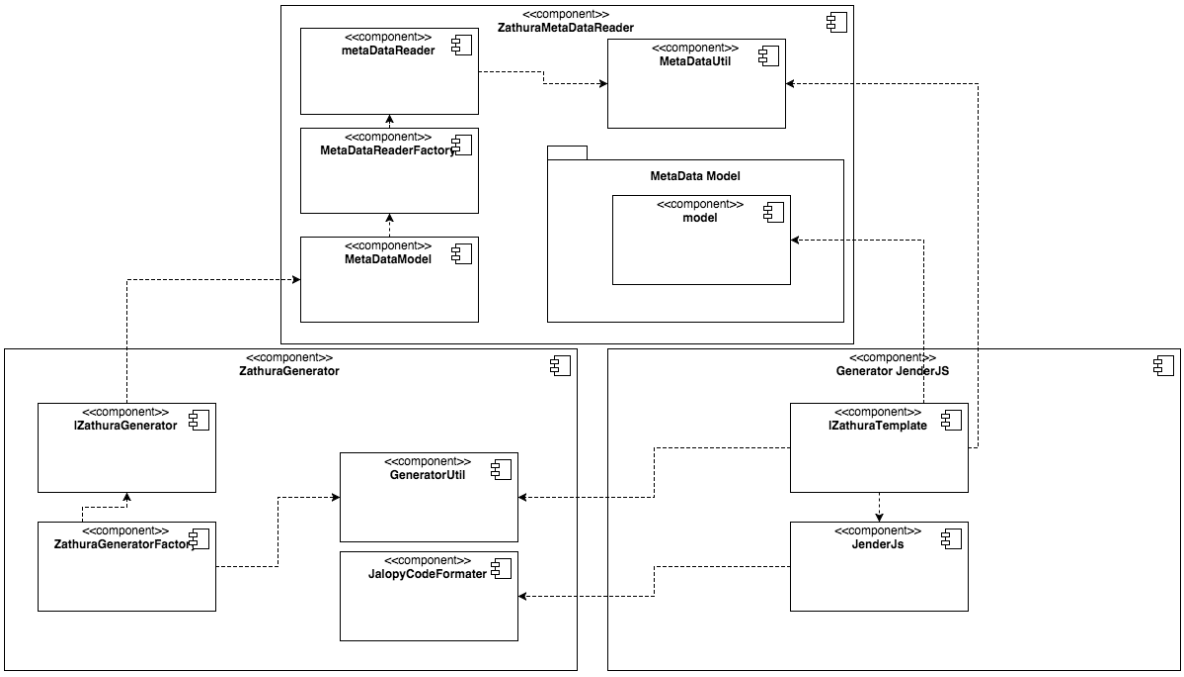


Diagrama de componentes que muestra cómo interactúa el generador cuando se procesa la MetaData que viene de la base de datos para convertirla en la aplicación codificada.

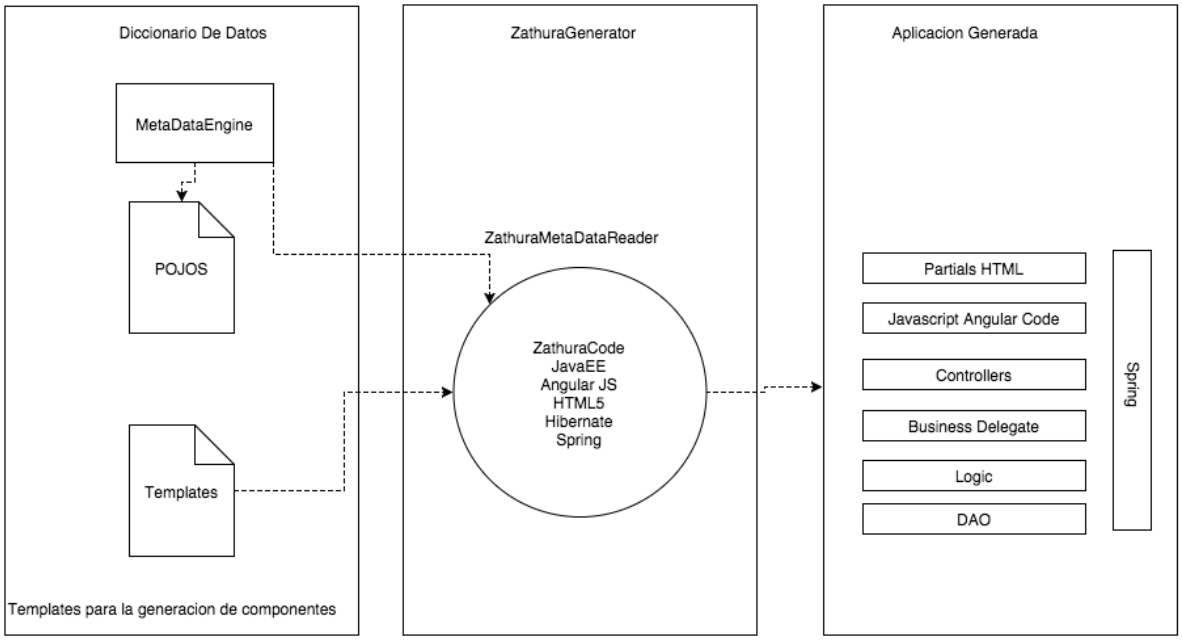


Diagrama que muestra las entradas y las salidas del generador de codigo, ademas muestra sus tecnologías implicadas.

SKYJET

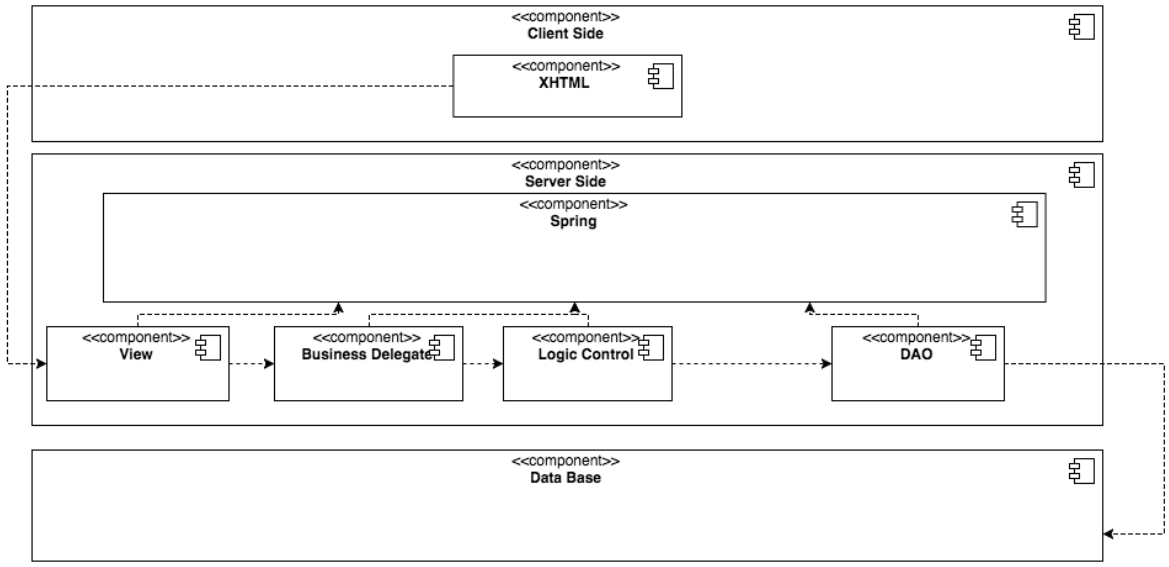


Diagrama de componentes que ilustra la forma en cómo se ejecuta el software generado.

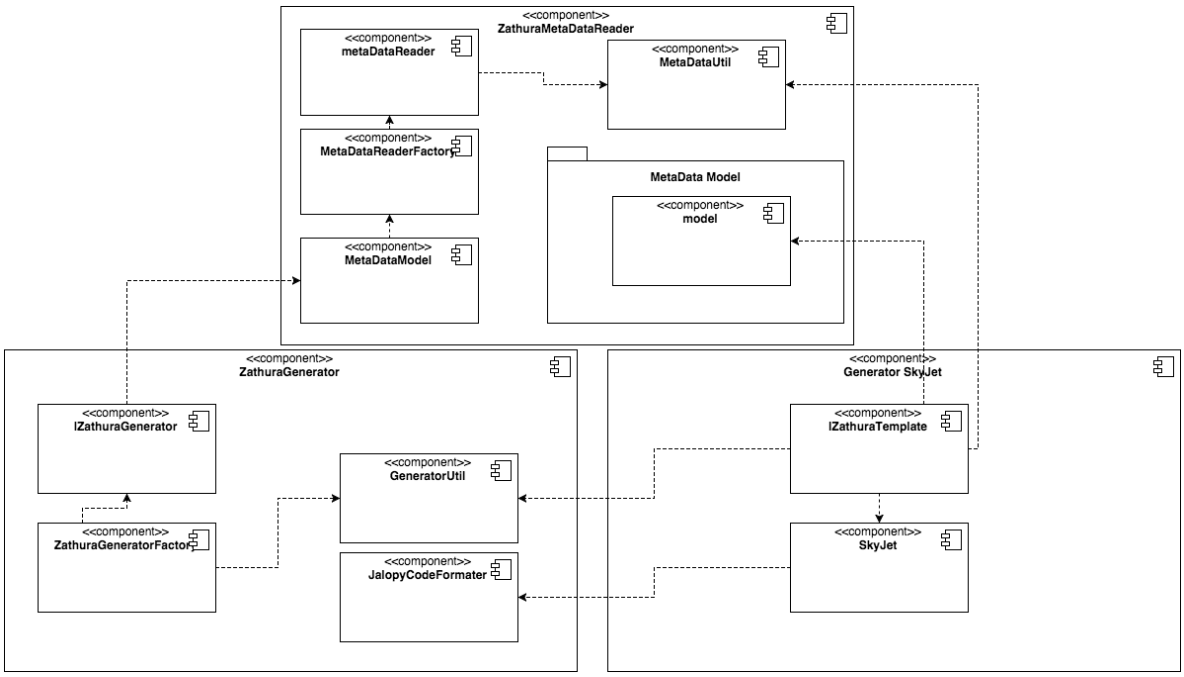


Diagrama de componentes que muestra cómo interactúa el generador cuando se procesa la MetaData que viene de la base de datos para convertirla en la aplicación codificada.

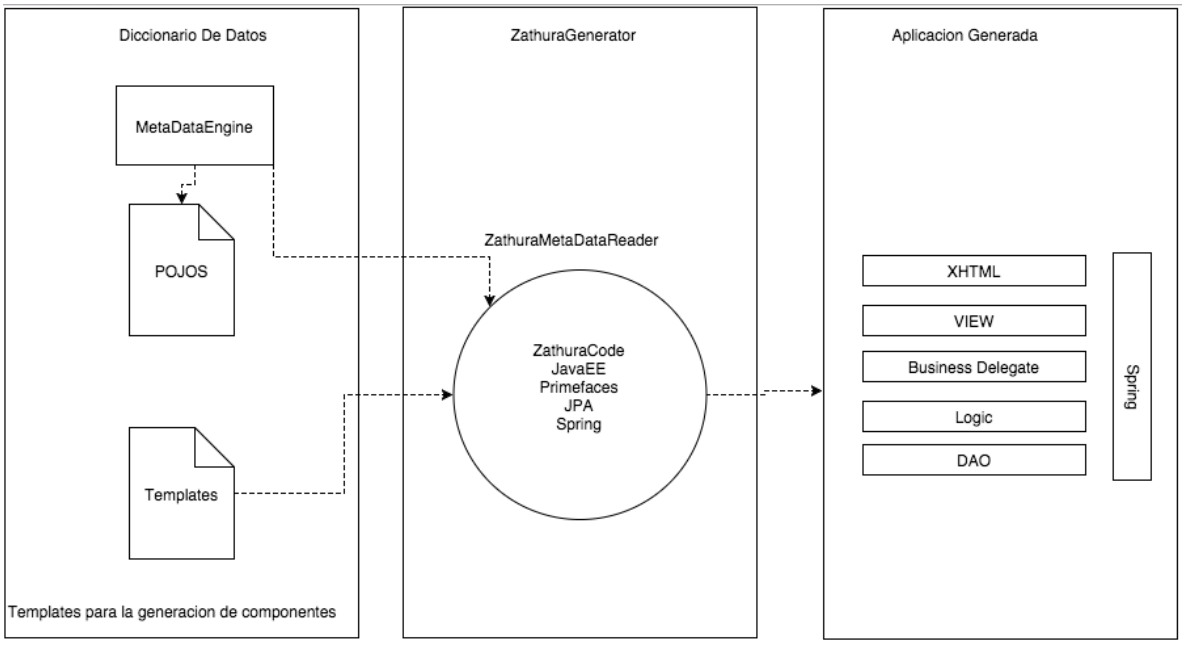


Diagrama que muestra las entradas y las salidas del generador de codigo, ademas muestra sus tecnologías implicadas.

SKYJET JS

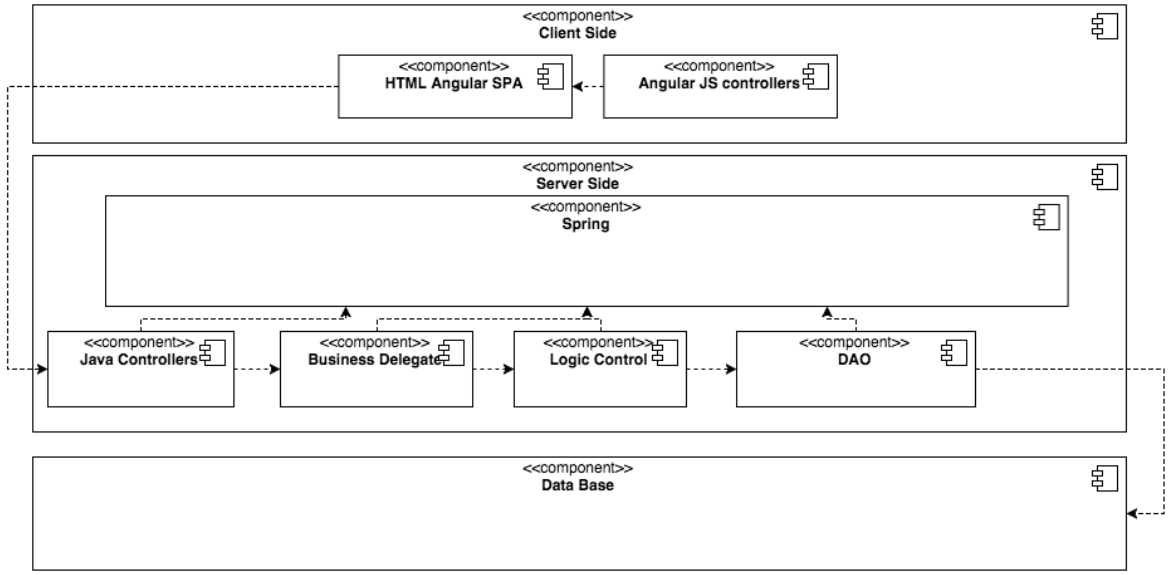


Diagrama de componentes que ilustra la forma en cómo se ejecuta el software generado.

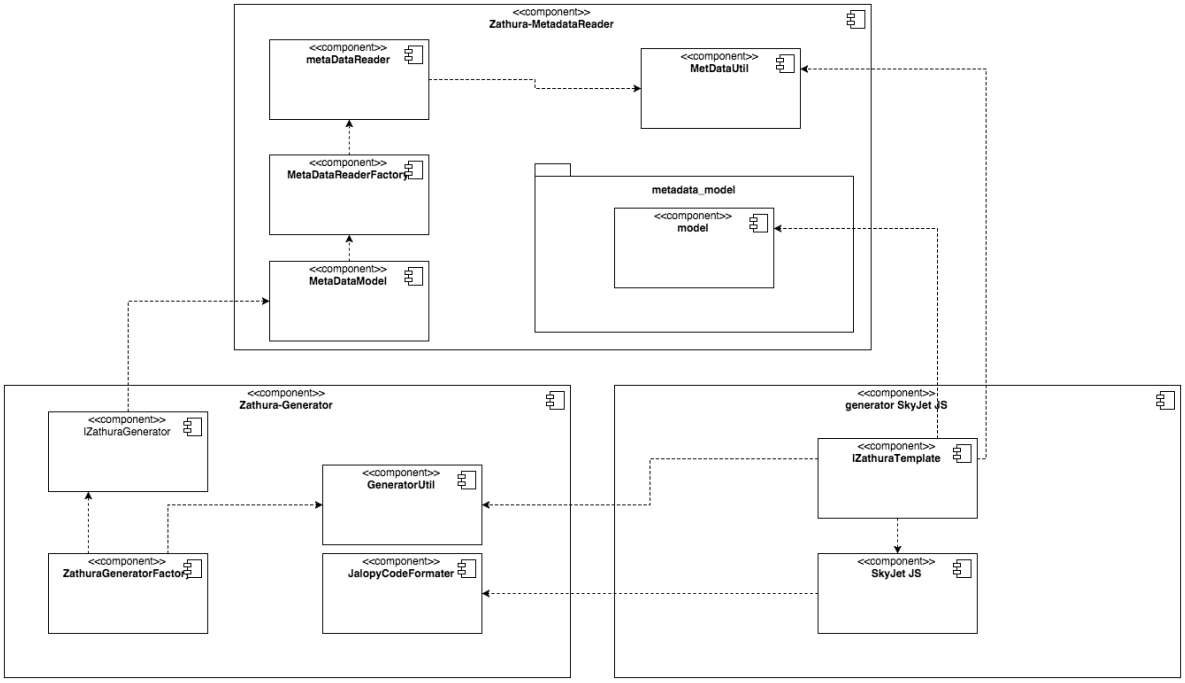


Diagrama de componentes que muestra cómo interactúa el generador cuando se procesa la MetaData que viene de la base de datos para convertirla en la aplicación codificada.

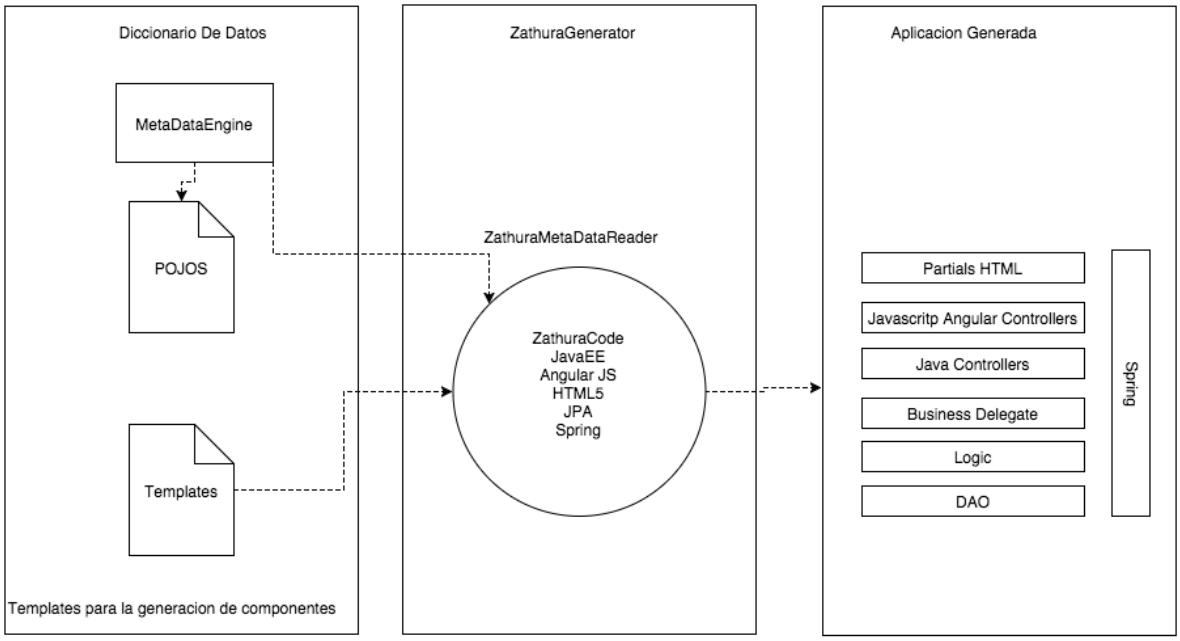
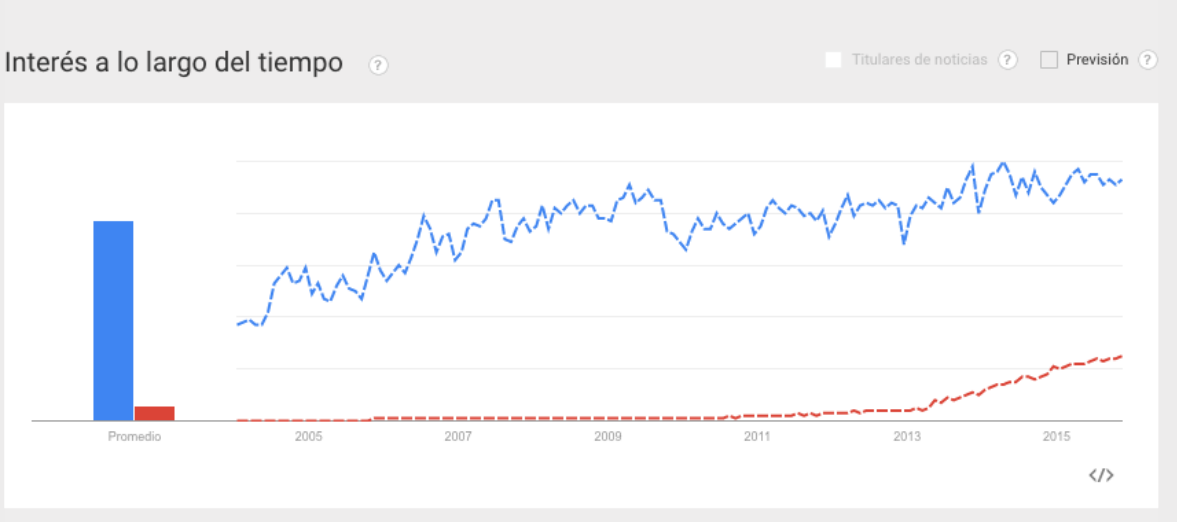
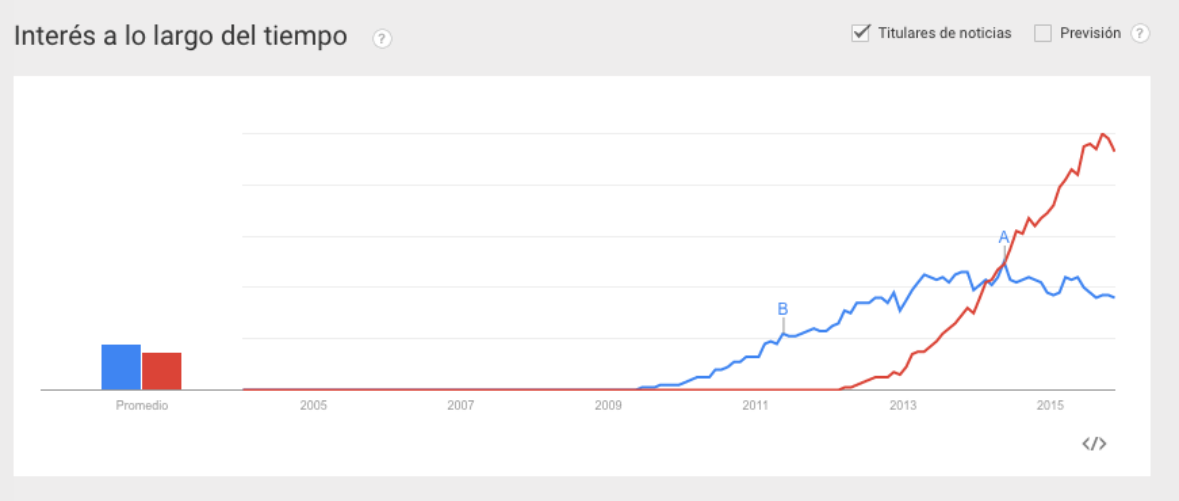


Diagrama que muestra las entradas y las salidas del generador de codigo, ademas muestra sus tecnologías implicadas.

Además de mostrar como quedan las arquitecturas luego de creadas, también es importante porque es necesario la creación de los nuevos robots implicando las nuevas arquitecturas por lo cual recurriremos a google trends para examinar el auge de estas nuevas tecnologías que han sido implementadas.





En el primer gráfico tenemos a Maven en color azul frente a gradle en color rojo , si bien es notable que Maven está por mucho encima en búsquedas, reconocemos también que le lleva bastante tiempo lanzado al mercado y que gradle es relativamente nuevo por lo cual muchos desarrolladores no migran sus proyectos a gradle aun así sabiendo que es mucho mejor y fácil de usar, personalmente reconozco como ingeniero su facilidad y por eso decidí implementarlo en los dos generadores de código nuevos.

En el segundo gráfico tenemos a Primefaces en azul y a Angular Js en rojo, como podemos apreciar aunque Primefaces lleva más tiempo en el mercado y aunque por un año lideró por encima de Angular luego de mediados del 2014 vemos como Angular Js equipara su cantidad de búsquedas, luego de esto en un año casi que duplica su cantidad en búsquedas y vemos como Primefaces sufre un declive, por lo cual tenemos que los desarrolladores están implementando mucho más Angular Js que Primefaces.

15. PRUEBA DE GENERACION DE CODIGO EN ZATHURACODE 6

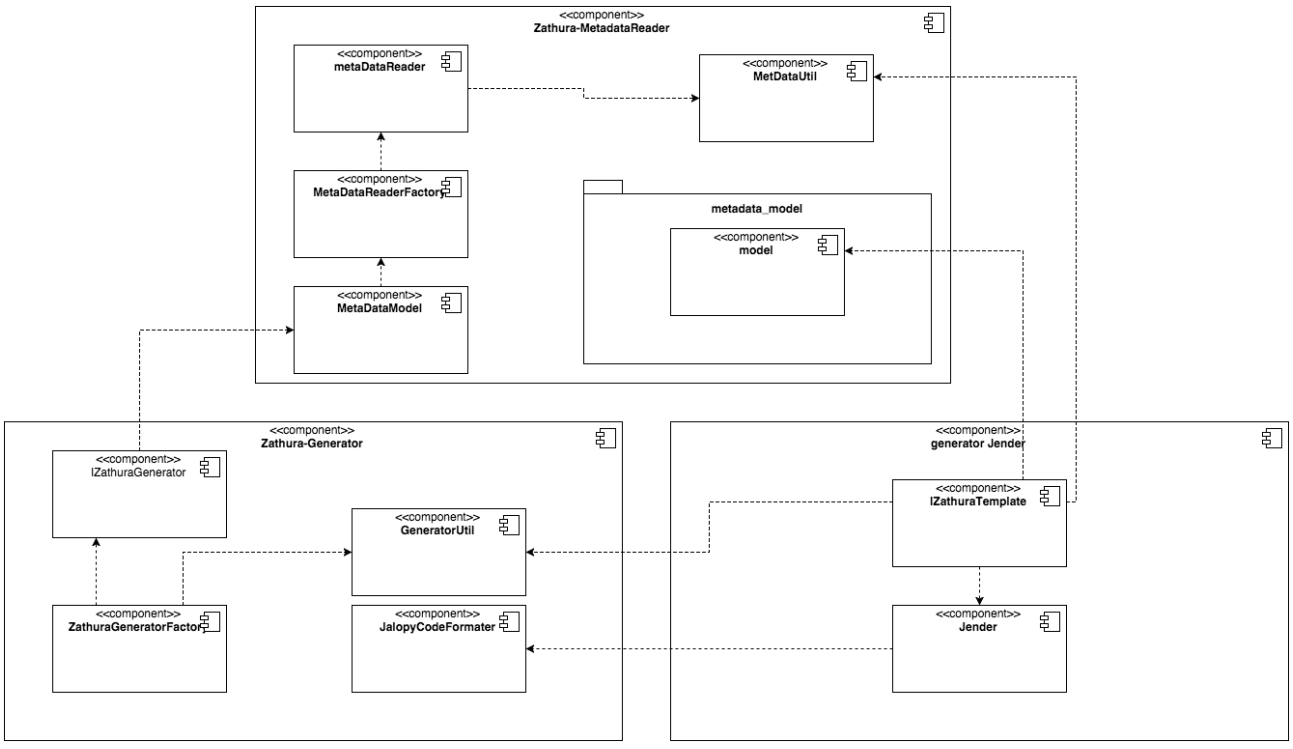
Con la inclusión de las nuevas arquitecturas se propone generar código para algún proyecto de prueba, ZathuraCode solo necesitará la base de datos para iniciar con la generación de código.

Se pidió a dos desarrolladores par hacer la generación del código mediante los nuevas arquitecturas, terminado el proceso de generación se les pidió llenar una encuesta que se encuentra como adjunta al documento de tesis para ser analizada.

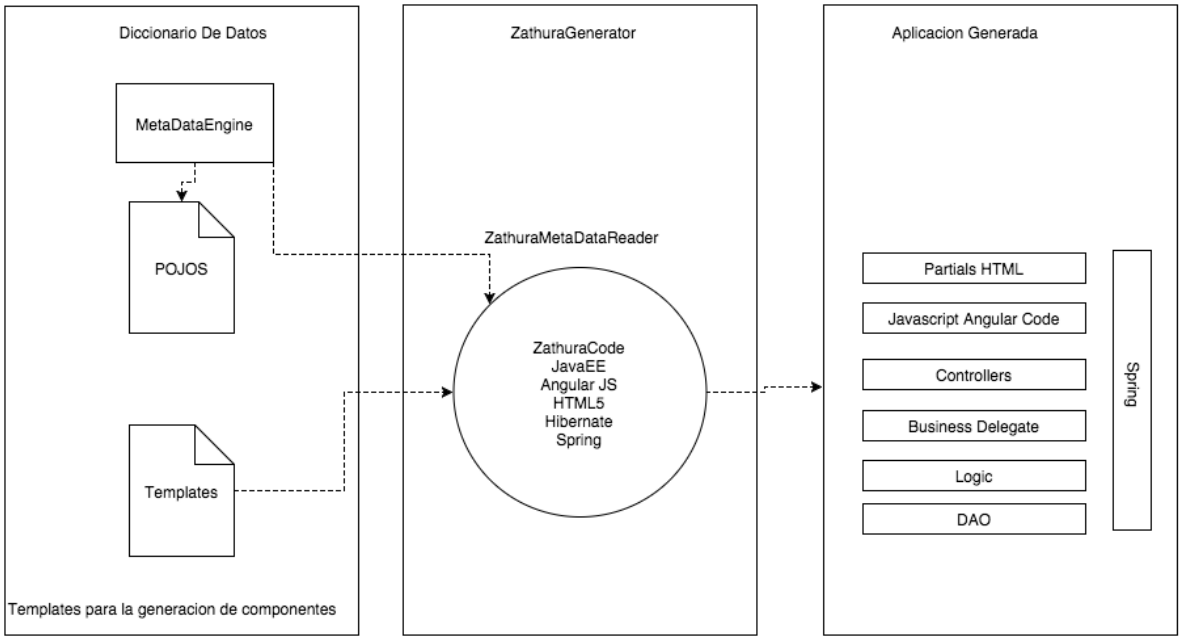
Evidentemente la prueba también es realizada por mi persona dando como resultado lo siguiente:

La base de datos de prueba en postgres está compuesta por 4 tablas las cuales son,Clientes, Mascotas, Vacunas y Vacunas_Mascotas

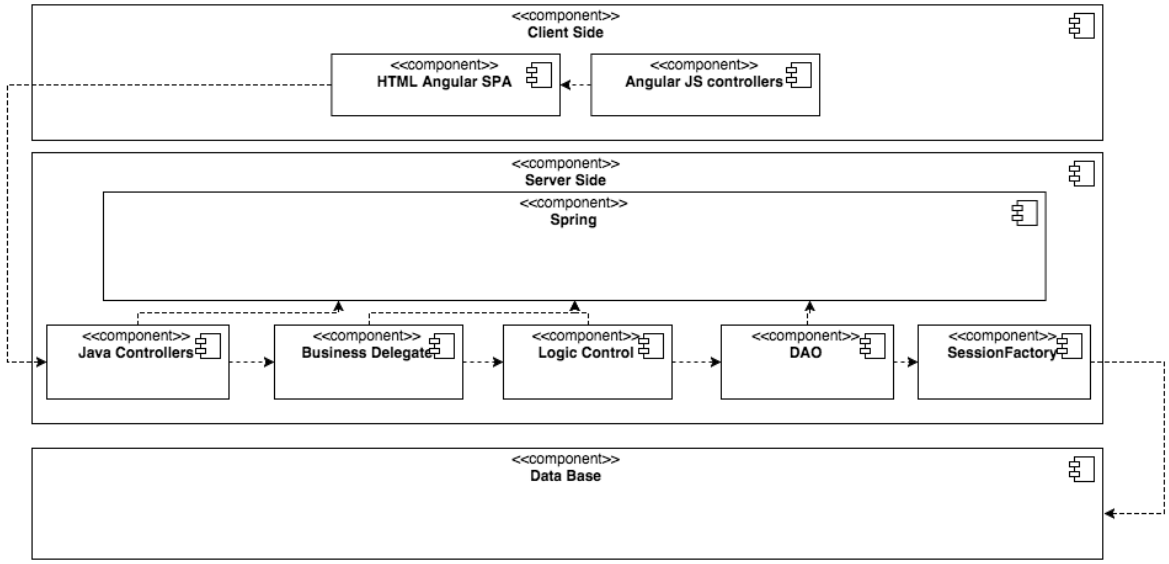
A partir de allí inicia la generación de código bajo las dos arquitecturas descritas en los siguiente diagrama de componentes:



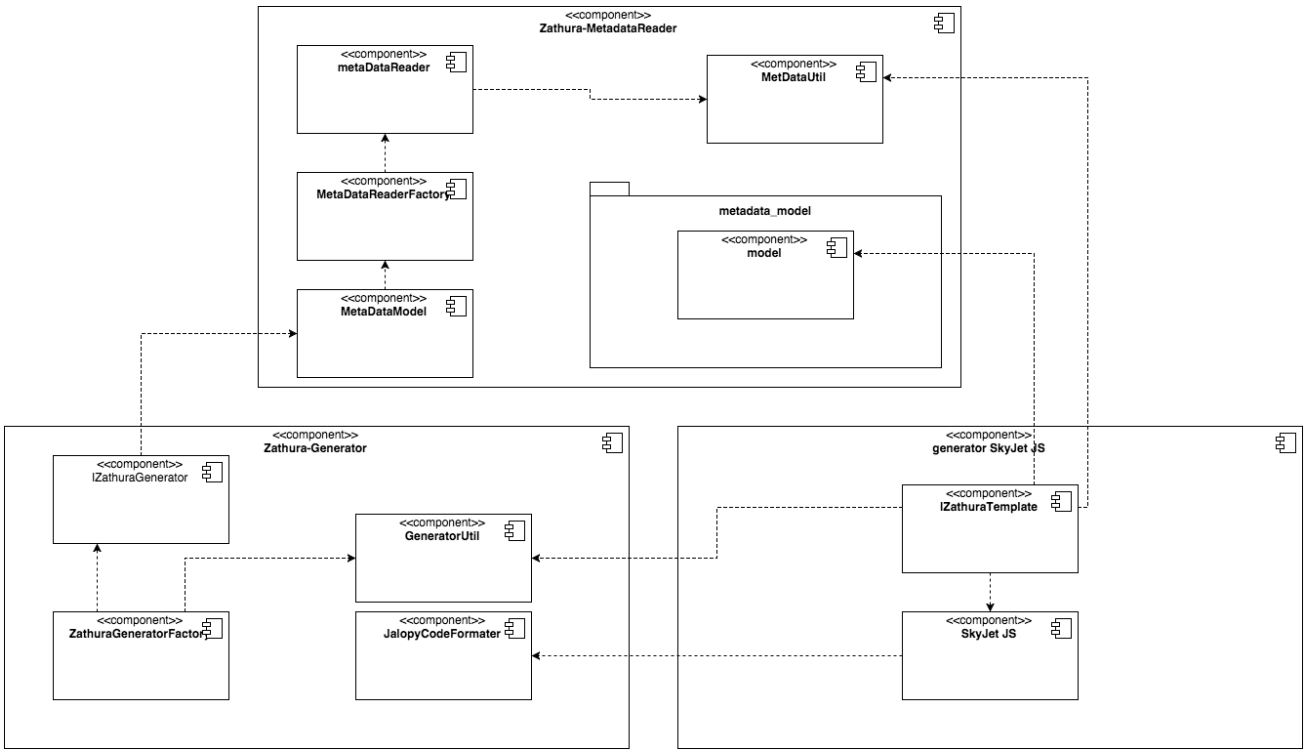
de igual manera se muestra en el siguiente gráfico cómo interactúa el generador de código para la generación del proyecto, tanto sus entradas como sus salidas:



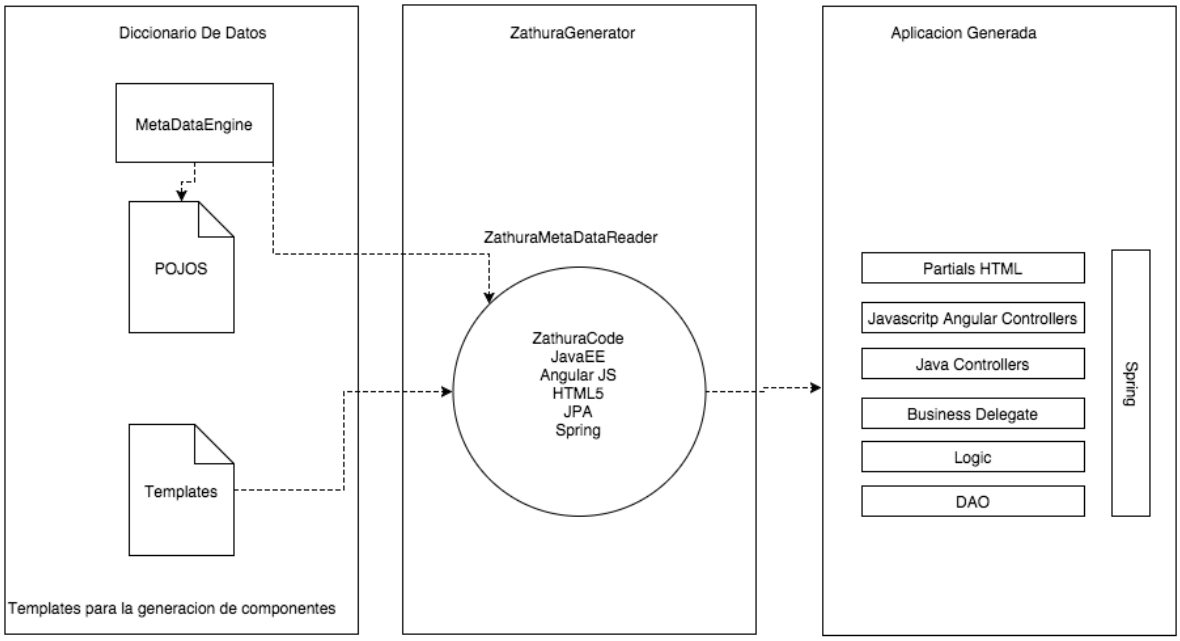
La aplicación resultante se basa en el siguiente diagrama de componentes para explicar cómo funciona la arquitectura del proyecto generado:



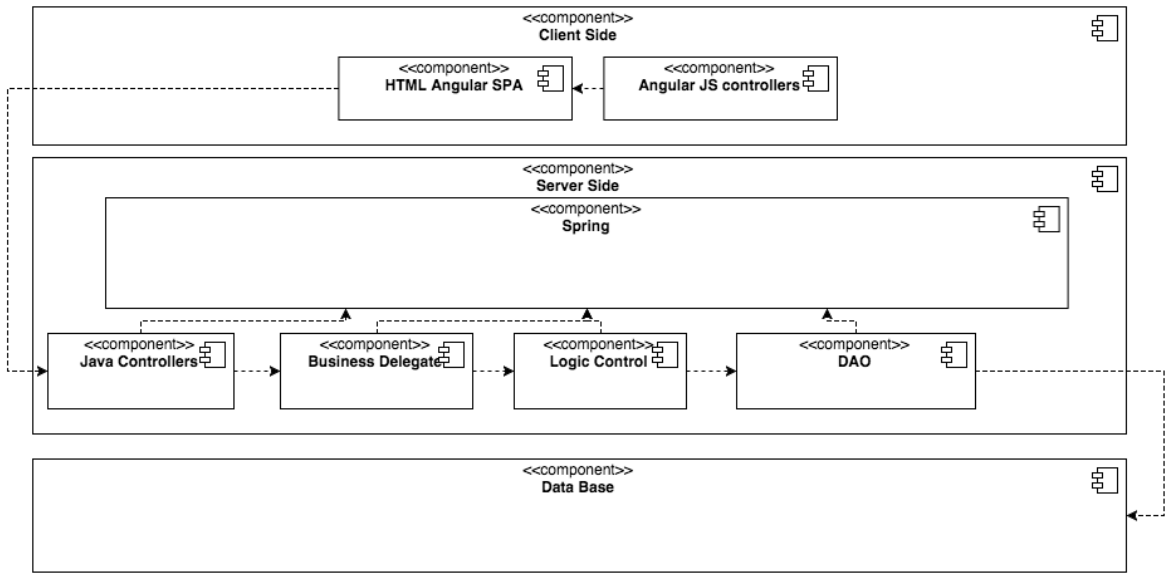
Así mismo para la arquitectura creada por el robot Skyjet JS que es descrita en los siguientes diagramas de componentes:



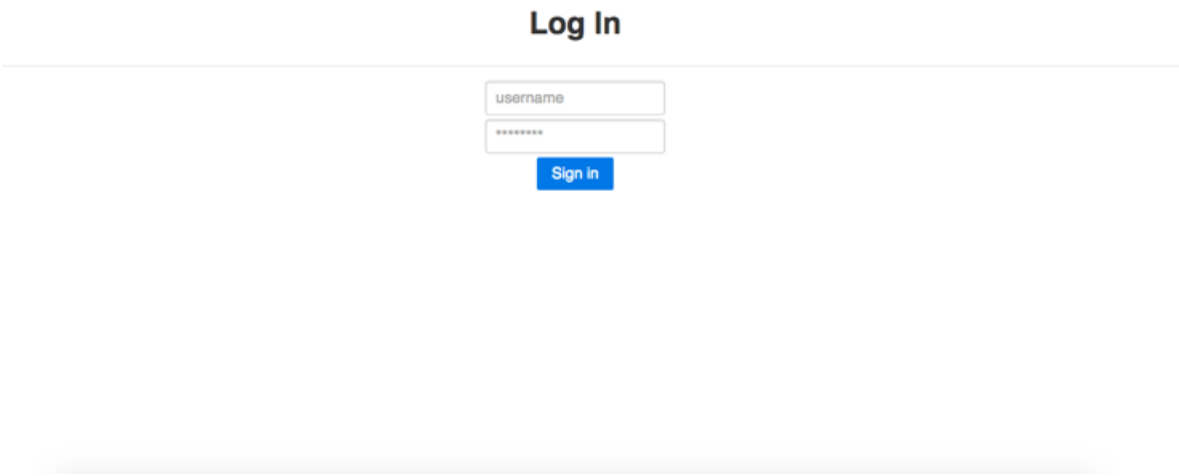
de igual manera se muestra en el siguiente gráfico cómo interactúa el generador de código para la generación del proyecto, tanto sus entradas como sus salidas:

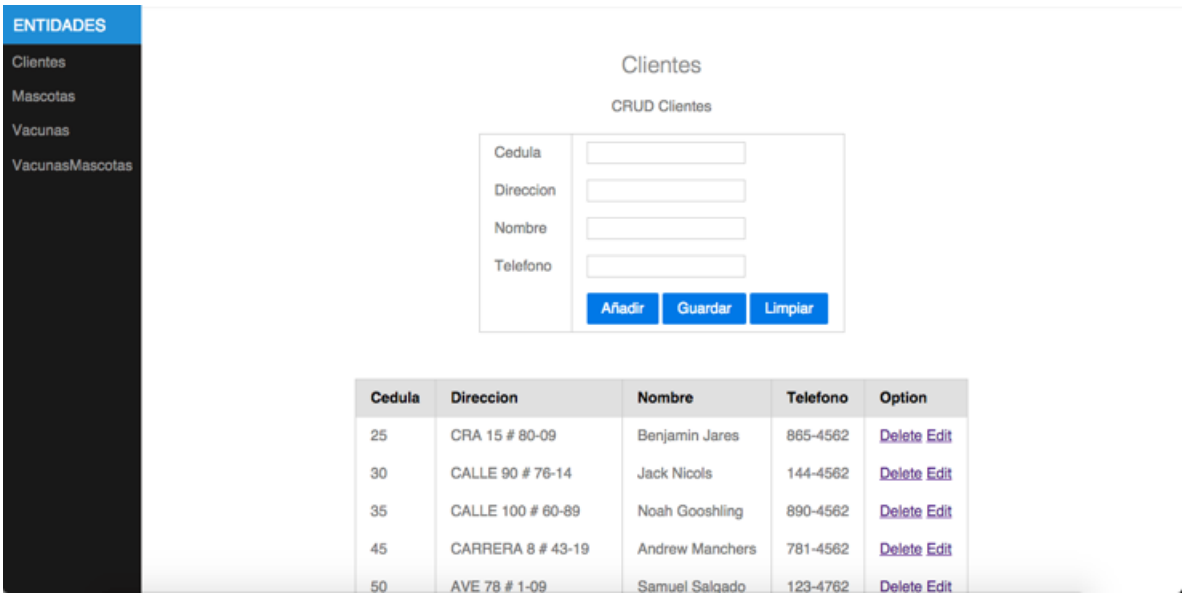


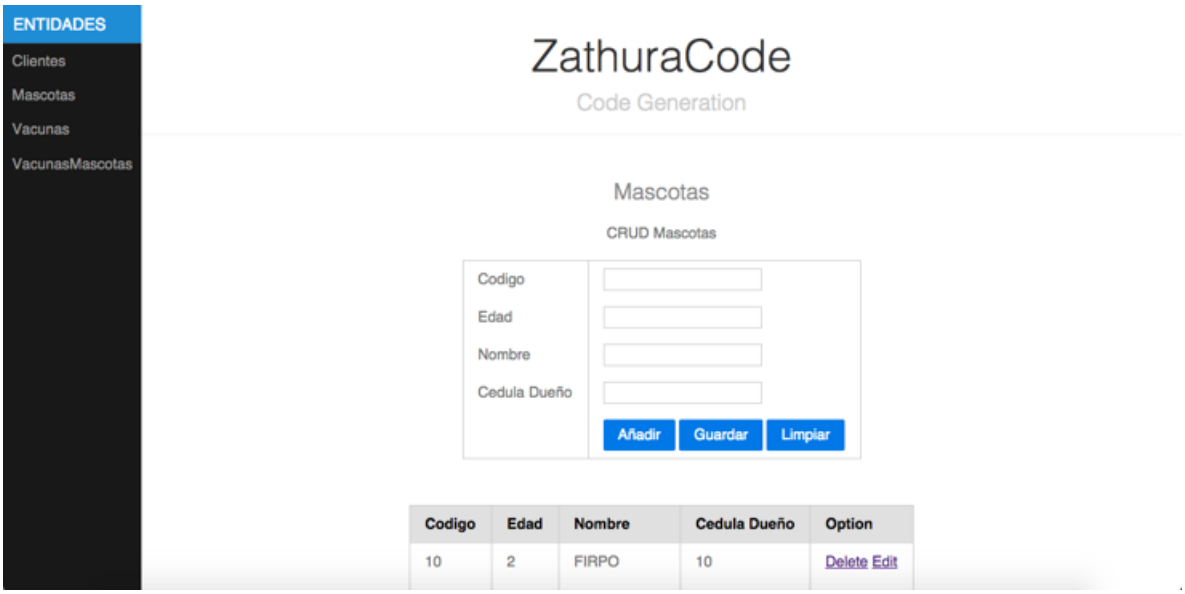
La aplicación resultante se basa en el siguiente diagrama de componentes para explicar cómo funciona la arquitectura del proyecto generado:

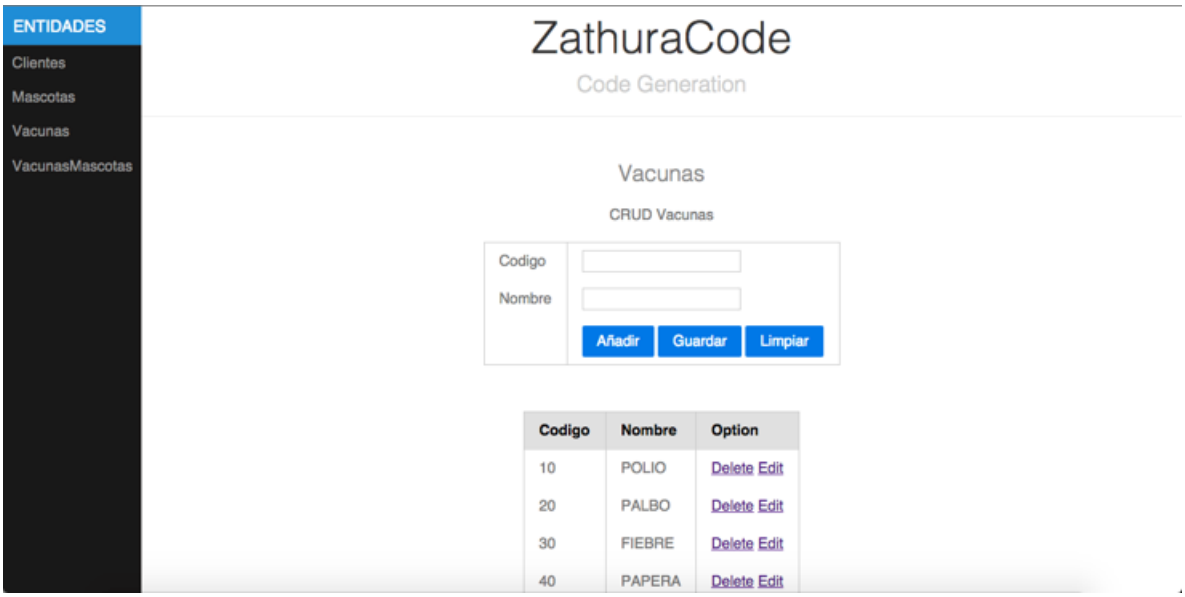


Tenemos como resultado una aplicacion web Java la cual se muestra al programador de la siguiente manera:




ENTIDADES
Clientes
Mascotas
Vacunas
VacunasMascotas
Clientes
CRUD Clientes
Cedula
Direccion
Nombre
Telefono
Añadir
Guardar
Limpiar
Cedula
Direccion
Nombre
Telefono
Option
25
CRA 15 # 80-09
Benjamin Jares
865-4562
Delete Edit
30
CALLE 90 # 76-14
Jack Nicols
144-4562
Delete Edit
35
CALLE 100 # 60-89
Noah Gooshling
890-4562
Delete Edit
45
CARRERA 8 # 43-19
Andrew Manchers
781-4562
Delete Edit
50
AVE 78 # 1-09
Samuel Salgado
123-4762
Delete Edit


ENTIDADES
Clientes
Mascotas
Vacunas
VacunasMascotas
ZathuraCode
Code Generation
Mascotas
CRUD Mascotas
Codigo
Edad
Nombre
Cedula Dueño
Añadir
Guardar
Limpiar
Codigo
Edad
Nombre
Cedula Dueño
Option
10
2
FIRPO
10
Delete Edit

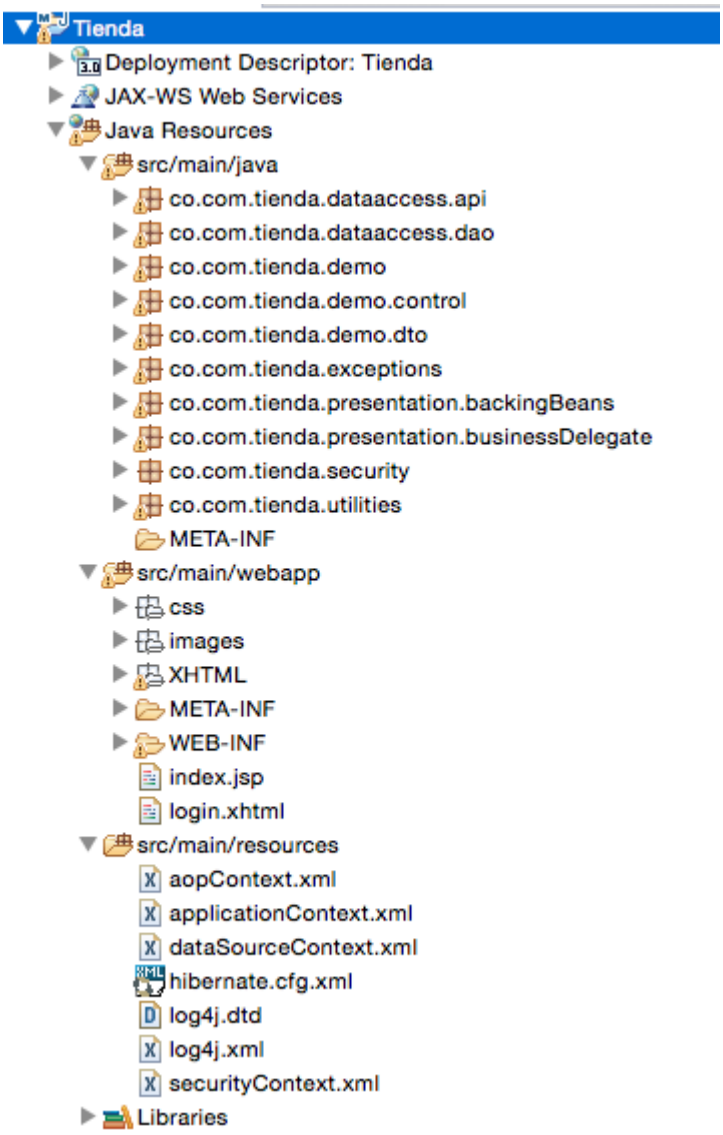


como se puede apreciar en las imágenes anteriores, la aplicación generada es un CRUD de todas las entidades que se encuentran en la base de datos.

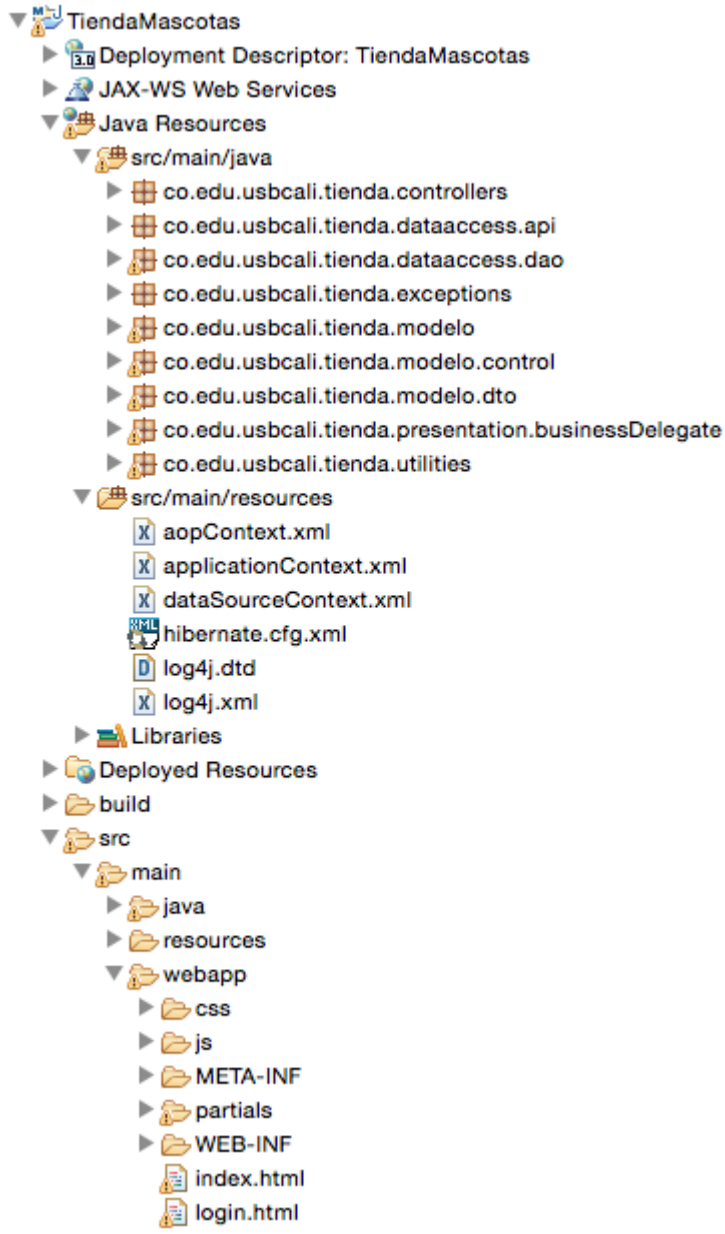
16. COMPARATIVA DE LAS ARQUITECTURAS ENTREGADAS EN JENDER Y JENDER JS

Luego de generar el proyecto en ambas arquitecturas se harán comparativas entre la estructura de los paquetes de cada proyecto generado y se darán detalles y comentarios de las diferencias entre la distribución de paquetes entregados.

Estructura de paquetes generada a partir del robot generador Jender:



Estructura de paquetes generada a partir del robot generador Jender JS:



Como se puede apreciar las diferencias en las imágenes las diferencias son las siguientes:

Jender	**Jender JS**
Paquetes Backing Beans y Security.	Paquete Controllers
Carpetas XHTML, Images y CSS	Carpetas partials, js, css

Estas diferencias hacen que las dos arquitecturas sean distintas ya que los paquetes Backing Beans son reemplazados con el paquete controllers esto se debe a que los Backing Beans eran los encargados de hacer la comunicación entre el Business Delegate y las pantallas realizadas en primefaces, y en el caso de Jender JS el paquete controllers es el encargado de hacer la comunicación entre los controllers de Angular Js y el business delegate.

La carpeta XHTML es reemplazada por la carpeta partials ya que en Jender las extensiones de las pantallas generadas estaban en primefaces con extensión XHTML y en Jender Js las pantallas generadas son generadas en HTML5 con Angular JS de extensión HTML es llamada ‘partials’ porque estas solo serán parciales ya que el usuario luego podra modificarlas.

La carpeta Js en la arquitectura de Jender Js es creada para alojar los archivos Javascript encargados de interactuar con las páginas alojadas en la carpeta ‘partials’.

Aunque ambas arquitecturas tienen una carpeta para archivos de estilo CSS estas contienen diferentes archivos, en Jender se alojan estilos propios de primefaces, y en Jender Js se alojan estilos propios de Angular JS.

Obviamente los proyectos generados fueron a partir de una misma base de datos, de allí que todo el proyecto genera código a partir de la misma MetaData.

17. CONCLUSIONES

- La investigación fue provechosa puesto que se logró el objetivo alcanzado, el cual era integrar las nuevas tecnologías en las arquitecturas del generador de código.
- En el desarrollo de la investigación se encontraron nuevas mejoras que podrían hacerse en el camino de estas dos nuevas arquitecturas.
- El tener que aprender el funcionamiento del plugin e indagar en sus características conllevo a poder corregir ciertos puntos que no estaban considerados.
- La investigación abrió muchas posibilidades a que el generador de código en sus próximas versiones pueda integrar muchas más tecnologías.
- La investigación me permitió conocer nuevas tecnologías y aprender cómo estas interactúan entre sí.
- La inclusión de nuevas arquitecturas que permitan aumentar el catálogo de opciones de generación de código continuarán 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 GRADLE 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.
- 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.

18. REFERENCIAS Y BIBLIOGRAFÍA

- Gradle (2007-2015), Gradle Documentation.: Gradle Org https://gradle.org/
- Angular Js (2009-2015), Angular Js Documentation.: Angular Org https://angularjs.org/
- Gradle, (s.f). En Wikipedia. Recuperado el 25 de Julio de 2015 de https://en.wikipedia.org/wiki/Gradle
- HTML5, (s.f). En Wikipedia. Recuperado el 25 de Julio de 2015 de https://es.wikipedia.org/wiki/HTML5
- Angular JS, (s.f). En Wikipedia. Recuperado el 25 de Julio de 2015 de https://es.wikipedia.org/wiki/AngularJS
- Steve Hanson, (2013) Simple Spring MVC Web Application using Gradle,http://www.javacodegeeks.com/2013/04/simple-spring-mvc-web-application-using-gradle.html
- Benjamin Muschko, (2014) Gradle in Action
- Shyam Seshadri, Brad Green,(2014) Angular: Up and Runing