**DOCUMENTO ARQUITECTURA DE LA SOLUCION**

****

**Especialización**

**Arquitectura Empresarial de Software**

**Modelado y validación de arquitectura**

**Tutor:**

**German Alonso Suarez Guerrero**

**Presentado por:**

**John Barrera**

**Mario Arcila**

**Andres Sarabanda**

**Bogotá Mayo 2019**

|  |  |  |
| --- | --- | --- |
| **Versión** | **Fecha** | **Descripción de la modificación** |
| 1.0 | 20/05/2019 | Documento inicial de arquitectura |

Tabla de Contenido

Contenido

[1. Introducción 4](#_Toc9194442)

[1.1 Objetivo 4](#_Toc9194443)

[1.2 Definiciones, Acrónimos y Abreviaturas 4](#_Toc9194444)

[1.3 Referencias 5](#_Toc9194445)

[1.4 Alcance 6](#_Toc9194446)

[2. Proyecto publicado en GitHub, junto con su documentación. 6](#_Toc9194447)

[3. Justificaciones de arquitectura sobre cada una de las decisiones tomadas. 6](#_Toc9194448)

[4. Modelo. 7](#_Toc9194449)

[5. Generación de los artefactos de servicios identificados para el inventario 8](#_Toc9194450)

[6. Creación del inventario de servicio y registro de los mismos. 8](#_Toc9194451)

[7. Creación de un servicio basado en el patrón Intermediate Routing 9](#_Toc9194452)

[8. Implementar los servicios que soportan el proceso 9](#_Toc9194453)

[9. Realizar la composición de los servicios 9](#_Toc9194454)

[10. Generar servicios que sean autocontenidos e independientemente desplegables en Docker 9](#_Toc9194455)

[11. Implementar un servicio que modele el patrón de API Proxy 9](#_Toc9194456)

# Introducción

El Banco ABC está realizando varios proyectos de actualización tecnológica los cuales le permiten ofrecer sus productos financieros de manera más ágil y de ésta forma responder a nuevas necesidades del mercado.

El Banco acaba de firmar una alianza estratégica con diferentes proveedores de servicios públicos (Agua, Gas, Luz, Telefonía) o también llamados convenios, para permitir a los clientes del banco a través de los diferentes canales de servicio (Cajeros Automáticos, Cajero de Oficina, Teléfono, Portal Web, Aplicación Móvil) permitir el pago de los mismos.

## Objetivo

Implementar una solución utilizando una aproximación orientada a servicios, la cual permita al Banco cumplir con las necesidades de negocio, basándose en los principios de diseño de servicios, patrones y estrategias arquitecturales.

## Definiciones, Acrónimos y Abreviaturas

Servidor: Un servidor de base de datos es un programa que provee servicios de base de datos a otros programas u otras computadoras, como es definido por el modelo cliente-servidor. También puede hacer referencia a aquellas computadoras (servidores) dedicadas a ejecutar esos programas, prestando el servicio.

REST: Deriva de "Representational State Transfer", que traducido vendría a ser “transferencia de representación de estado”, lo que tampoco aclara mucho, pero contiene la clave de lo que significa. Porque la clave de REST es que un servicio REST no tiene estado (es stateless), lo que quiere decir que, entre dos llamadas cualesquiera, el servicio pierde todos sus datos.

SOAP: (originalmente las siglas de Simple Object Access Protocol) es un protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML. Este protocolo deriva de un protocolo creado por Dave Winer en 1998, llamado XML-RPC. SOAP fue creado por Microsoft, IBM y otros. Está actualmente bajo el auspicio de la W3C. Es uno de los protocolos utilizados en los servicios Web.

Eclipse Fundación: Es una corporación independiente sin fines de lucro que actúa como administrador de la comunidad de desarrollo de software de código abierto Eclipse. Es una organización apoyada por más de 275 miembros. La Fundación se centra en servicios clave como: gestión de propiedad intelectual (IP), desarrollo de ecosistemas, proceso de desarrollo e infraestructura de TI. Sus miembros incluyen líderes de la industria que han adoptado el código abierto como un habilitador clave para la estrategia empresarial.

Docker: Es un proyecto de código abierto que automatiza el despliegue de aplicaciones dentro de contenedores de software, proporcionando una capa adicional de abstracción y automatización de virtualización de aplicaciones en múltiples sistemas operativos. Docker utiliza características de aislamiento de recursos del kernel Linux, tales como cgroups y espacios de nombres (namespaces) para permitir que "contenedores" independientes se ejecuten dentro de una sola instancia de Linux, evitando la sobrecarga de iniciar y mantener máquinas virtuales. El soporte del kernel Linux para los espacios de nombres aísla la vista que tiene una aplicación de su entorno operativo, incluyendo árboles de proceso, red, ID de usuario y sistemas de archivos montados, mientras que los cgroups del kernel proporcionan aislamiento de recursos, incluyendo la CPU, la memoria, el bloque de E/S y de la red.

Eureka: Eureka es un servicio REST, utilizándose principalmente en la nube de AWS, a la cuál está estrechamente ligado. Eureka se comporta como servidor, cuyo objetivo es registrar y localizar microservicios existentes, informar de su localización, su estado y datos relevantes de cada uno de ellos. Además, nos facilita el balanceo de carga y tolerancia a fallos.

Por último, añadir que gracias a Spring Cloud Netflix, nos proporciona una fácil integración con el proyecto Netflix OSS (Open Source Software) para aplicaciones Spring Boot.

HTTP: El Protocolo de Transferencia de HiperTexto (Hypertext Transfer Protocol) es un sencillo protocolo cliente-servidor que articula los intercambios de información entre los clientes Web y los servidores HTTP. La especificación completa del protocolo HTTP 1/0 está recogida en el RFC 1945. Fue propuesto por Tim Berners-Lee, atendiendo a las necesidades de un sistema global de distribución de información como el World Wide Web

Maven: Maven es una herramienta de automatización de compilación utilizada principalmente para proyectos Java. Aborda dos aspectos del software de construcción: primero, describe cómo se construye el software, y segundo, describe sus dependencias. A diferencia de las herramientas anteriores, como Apache Ant , utiliza convenciones para el procedimiento de compilación, y solo se deben escribir las excepciones. Un archivo XML describe el proyecto de software que se está construyendo, sus dependencias de otros módulos y componentes externos, el orden de compilación, los directorios y los complementos necesarios. Viene con objetivos predefinidos para realizar ciertas tareas bien definidas, como la compilación de código y su empaquetado.

## Referencias

Servidor:<http://www.alegsa.com.ar/Dic/servidor_de_base_de_datos.php>

Rest: <https://desarrolloweb.com/articulos/que-es-rest-caracteristicas-sistemas.html>

Soap: <https://es.wikipedia.org/wiki/Simple_Object_Access_Protocol>

Eclipse: <https://www.eclipse.org/org/foundation/>

Docker: <https://www.docker.com/>

Eureka: <https://blog.bi-geek.com/arquitecturas-spring-cloud-netflix-eureka/>

Http: <http://neo.lcc.uma.es/evirtual/cdd/tutorial/aplicacion/http.html>

Maven: https://en.wikipedia.org/wiki/Apache\_Maven

## Alcance

La solución estará limitada solo a la demostración entregable donde se puedan consumir los servicios expuestos por los proveedores para los convenios inscritos de agua, gas y compensación, a través de herramientas tecnológicas como Rest, Soap, Eureka, Api Gateway, Kong, Docker entre otros.

# Proyecto publicado en GitHub, junto con su documentación.

A continuación, se indica la dirección electrónica en la cual se encuentra publicado los entregables del proyecto

Indicar URL

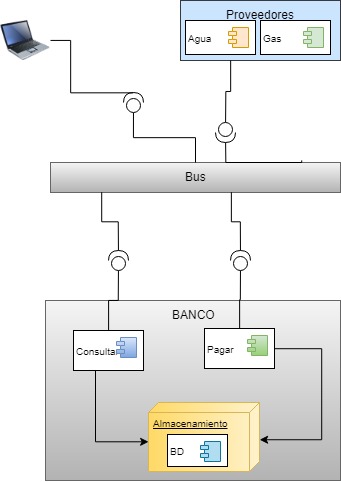
# Justificaciones de arquitectura sobre cada una de las decisiones tomadas.

Dentro de los patrones incluidos en la implementación se destacan el Api Gateway como único punto de entrada para el manejo y traducciones de protocolos, Domain Inventory para la conocer el inventario de servicios en ejecución, Federated Endpoint Layer con el objetivo de mantener un gobierno de federado para los servicios.

Para realizar la implementación se utilizaron las siguientes herramientas de desarrollo: Eclipse Jee 2019 para el desarrollo de los servicios, Docker 18.09 para implementación de imágenes Dokerizadas como BD en MySql, Kong para Transformar peticiones y respuestas, Eureka como servidor de registro para microservicios.

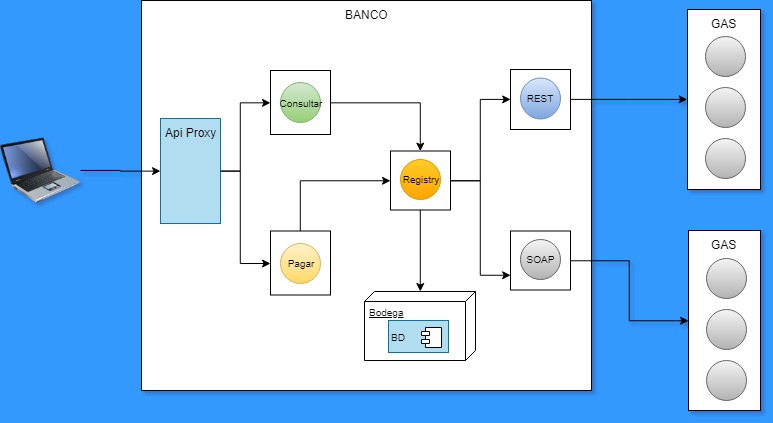
Trade-off se descartaron herramientas que requerían de licenciamiento y/o eran muy complejas para el resultado en el corto tiempo, ejemplo de ello es Juddi, wso2 carbon studio, .net.

Para realizar la implementación se abordaron diferentes modelos arquitecturales que fueron descartándose en el camino pues carecían de reutilización y/o funcionalidad.



*La imagen anterior representa uno de los modelos descartados, el motivo hace referencia a un Bus de servicio el cual soporta el 100% de la operación se descartó como arquitectura base, debido a restricciones y costos propios de la herramienta.*

# Modelo de solución.



*El modelo presentado en el diagrama anterior es una abstracción arquitectural de la solución para la inclusión de diferentes convenios con el banco ABC.*

# Generación de los artefactos de servicios identificados para el inventario

Para la construcción de la solución se implementaron los contratos de consultaSaldoFactura.yaml , pagarSaldoFactura.yaml, enrutarFactura.yaml, compensar.yaml, adaptadorRest.yaml los cuales se implementaron con el api de Swagger

# Creación del inventario de servicio y registro de los mismos.

Para la construcción de la solución se implementaron los servicios de: consultaSaldoFactura.java, pagarSaldoFactura.java, compensarSaldo.java, los cuales se implementaron con Maven

# Creación de un servicio basado en el patrón Intermediate Routing.

Se implementó el patrón de Intermédiate Routing a través del artefacto enrutarFactura.yaml, el cual permite el enrutamiento de los servicios dependiente de su tipo de operación con base en la información guardada en base de datos.

# Implementar los servicios que soportan el proceso

NO APLICA

# Realizar la composición de los servicios

usando cualquiera de las aproximaciones de composición (basado en Orquestación o Coreografía, se puede utilizar FUSE, Camel Kafka, ActiveMQ, RabbitMQ pero está prohibido utilizar BPEL y BPMS)

NO SE QUE CARRETA PONER AQUI

# Generar servicios que sean autocontenidos e independientemente desplegables en Docker

NO SE QUE CARRETA PONER AQUI

# Implementar un servicio que modele el patrón de API Proxy

El API Proxy debe ser capaz de trabajar con diferentes representaciones de los datos, tanto para los request como para los response. De esta manera se pueden tener requests/responses XML/JSON

NO SE QUE CARRETA PONER AQUI