TALLER ARQUITECTURA SERVICIOS CON DOCKER

Presentado Por:
ZULMA FARIDE CONTRERAS VARGAS
CAMILO ANDRES ALBARRACIN
MANUEL FERNANDO QUIROGA MUNAR
NESSLER DUVAN CARDENAS HERRERA
OMAR DAVID COTES HERNANDEZ

Profesor:
ING. GERMAN ALONSO SUAREZ GUERRERO
MODELADO Y VALIDACIÓN DE ARQUITECTURA

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERÍA
ESP. ARQUITECTURA EMPRESARIAL DE SOFTWARE
BOGOTÁ
2017

Versión	Fecha	Descripción de la modificación
1	15/11/2017	Creación del documento

1. Introducción

2. Contexto

Containerization

Canonical Expression

Microservice Deployment

Redundant Implementation

Capability Composition

3.1 Objetivos de la arquitectura de la solución

3.2 Requerimientos Funcionales Significativos

3.3 Restricciones

3.4 Atributos de calidad

Disponibilidad

Performance

Escalabilidad

Usabilidad

4. Vistas de arquitectura

5. Decisiones de arquitectura

1. Introducción

El presente documento muestra cómo se encuentra construida la arquitectura para el Banco ABC, los patrones seleccionados para realizar la solución en mención y la justificación de las decisiones de arquitectura tomadas en el presente desarrollo.

1.1 Objetivo

Plantear una arquitectura orientada a microservicios mediante contenedores docker para dar solución al problema planteado de ofrecer el pago de facturas por diferentes proveedores a través del banco, dichos proveedores serán integrados de manera dinámica al sistema del banco.

2. Contexto

2.1 Fundamentos de la solución

Los patrones usados para la solución son:

PATRON	DEFINICIÓN
SOA	Se definió que la arquitectura debe ir en servicios debido a que va a dockerizar
Layers	Se definen capa de datos,capa de presentación, capa de integración (servicio de enrutador y despachador)
Dynamic Router	Se hace uso de tabla de enrutamiento dependiendo del proveedor
Data Format Transformation	Se realiza transformación de los datos entre los contratos de los servicios del banco a los contratos de un proveedor dado y viceversa mediante XSLT
Containerization	Servicios manejados con contenedores docker
Canonical Expression	Se observa el canónico cuando se lleva a cabo la transformación de la información retornada de los servicios de los proveedores a los contratos estándares de los servicios del banco.
Microservice Deployment	Despliegue de microservicios en contenedores docker.

Redundant Implementation	Los microservicios pueden ser desplegados en varios contenedores para generar redundancia y alta disponibilidad
Capability Composition	Composición de servicios mediante coreografía

3. Drivers de Arquitectura

3.1 Objetivos de la arquitectura de la solución

- Ver la interoperabilidad entre servicios contenerizados
- Tener una aplicación con alta disponibilidad y escalable
- Realizar la integración con los diferentes canales de pago que deseen ingresar con el banco.
- Obtener seguridad transaccional y trazabilidad en las transacciones realizadas

3.2 Requerimientos Funcionales Significativos

El sistema debe permitir realizar:

- Consulta de saldo a Pagar de la factura
- Pago del Servicio
- Compensación de pago (Opcional)

3.3 Restricciones

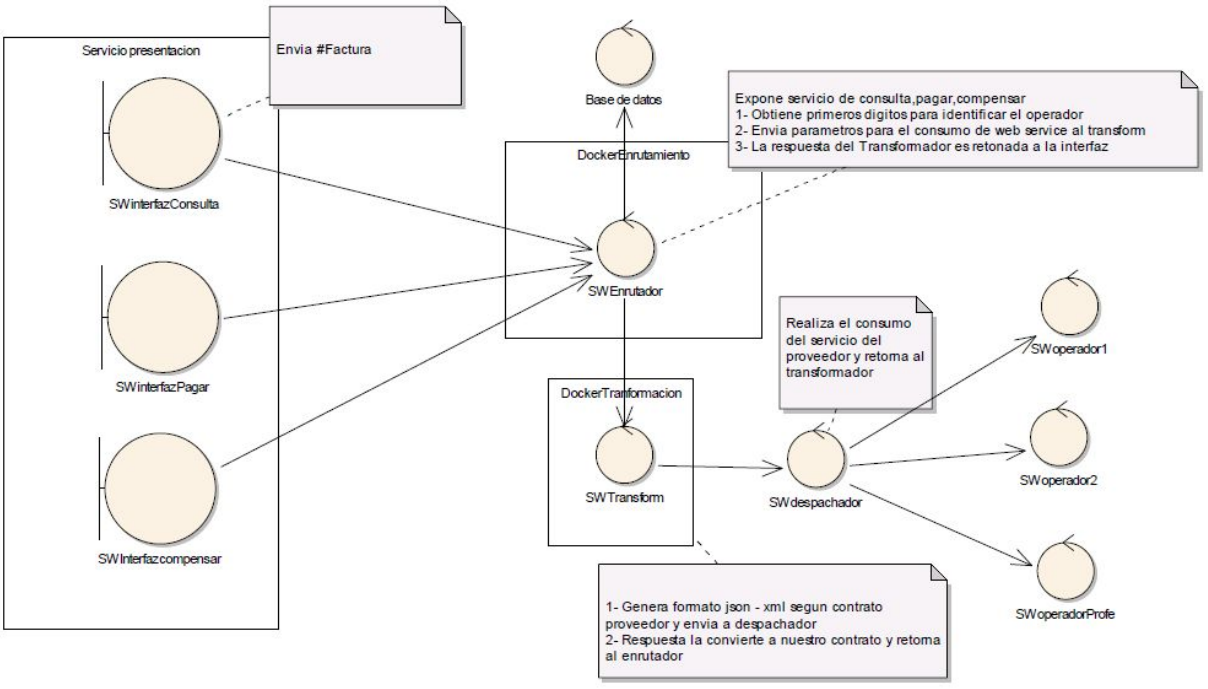
Arquitectura orientada a servicios realizados en php, con base de datos MySQL, los cuales están contenerizados con Docker.

3.4 Atributos de calidad

Atributo de Calidad	Unidad de medida	Importancia del atributo	Aclaración de Importancia
Disponibilidad	99.99%	Alta	Con el fin de satisfacer las necesidades de los clientes del

			banco se garantiza una disponibilidad del 99.99%
Performance	100 transacciones por minuto.	Alta	Para cumplir con la expectativas de los clientes en cuanto al tiempo de respuesta y el ingreso de datos correctos, permitiéndonos así ofrecer una arquitectura robusta, visionando una alta concurrencia de usuarios a futuro.
Escalabilidad	2 contenedores con microservicios desplegados de manera redundante, garantizando la disponibilidad	Alta	Evitar caídas por saturación de uso de la aplicación.
Usabilidad	5 clics para llevar a cabo el pago de una factura	Alta	Mostrar una aplicación intuitiva y de fácil manejo para una correcta navegabilidad y exitosa solicitud.
Mantenibilidad	30 minutos para realizar la integración con un proveedor nuevo.	Alta	Con el fin de minimizar el impacto en caso de realizar cambios en la infraestructura o servicios, con el manejo de microservicios desplegados de manera independientes estos cambios serán transparentes sin dejar la plataforma fuera de servicio

4. Vistas de arquitectura



5. Decisiones de arquitectura

DES_001	Microservicios mediante contenedores con docker
DES_002	Separación de responsabilidades con los microservicios del banco para interactuar con los servicios de los proveedores de la siguiente manera: ● Servicio enrutador ● Servicio transformador ● Servicio despachador ● Servicio de presentación
DES_003	Registro de servicios en base de datos
DES_004	Registro de plantillas de presentación