**DOCUMENTO DE ARQUITECTURA DE LA SOLUCION**

**Diseño de un Cajero Automático Multi-Moneda**

**Modelo 4 +1**

**Tabla de contenido**

[**Visión de Arquitectura** 2](#_Toc157079772)

[**Objetivo** 2](#_Toc157079773)

[**Alcance** 2](#_Toc157079774)

[**Contexto** 2](#_Toc157079775)

[**Decisiones de Arquitectura** 2](#_Toc157079776)

[**Drivers de Arquitectura** 3](#_Toc157079777)

[**Requerimientos funcionales significativos** 3](#_Toc157079778)

[**Restricciones** 4](#_Toc157079779)

[**Atributos de Calidad** 4](#_Toc157079780)

[**Vistas de Arquitectura** 5](#_Toc157079781)

**Control de Versión**

|  |  |  |  |
| --- | --- | --- | --- |
| **Versión** | **Fecha** | **Responsable** | **Descripción** |
| 1.0.0 | 20/01/2024 | Roger Chura | Versión Inicial. |
|  |  |  |  |

# **Visión de Arquitectura**

Proponer al banco LUNA S.A. una solución tecnológica centrada en la fidelización y expansión de su base de clientes mediante la introducción de un canal automatizado para la apertura de cuentas y la ejecución de transacciones. Esta implementación se traduciría en una disminución significativa de la carga operativa en plataforma y ventanilla, mejorando así la eficiencia y la experiencia general del cliente.

# **Objetivo**

Implementar un sistema de cajero automático que permita la creación de cuentas y sea capaz de manejar operaciones en moneda local y extranjera.

# **Alcance**

La solución debe utilizar una arquitectura de microservicios con Java y Spring Boot.

El sistema debe ser desplegado en un esquema de contenedores con apoyo de Docker.

Las monedas de operaciones deben ser en bolivianos y dólares.

El desarrollo solo alcanza a servicios Backend, queda fuera el desarrollo para Frontend.

# **Contexto**

Patrones de Software

Se debe considerar la provisión de las siguientes APIs

API de Gestión de Cuentas: Esta API permite la creación de cuentas en moneda bolivianos y dólares. Permite la consulta de saldos para una cuenta expresada en ambas monedas.

API de Gestión de Transacciones: Esta API permite realizar el depósito de fondos, retiro de fondos, realizar conversiones de moneda a una tasa de cambio, asimismo permite la consulta de las transacciones de una cuenta.

API de Gestión de Tasas de Cambio: Esta API permite la gestión de tasas de cambio (registro y modificaciones). Permite la consulta de las tasas de cambio para una fecha.

Los microservicios deben considerar sus propias bases de datos.

# **Decisiones de Arquitectura**

|  |  |  |
| --- | --- | --- |
| Número | Titulo | descripción |
| D-01 | Patrón nuclear | La solución estará basada en un patrón de arquitectura de Microservicios.  Cada API se implementará en un microservicio. Estas APIs deben exponer un conjunto de servicios que deben ser consumidos por un FrontEnd (ATM) y a su vez ser consumidos entre las mismas APIs. |
| D-02 | Patrón Complementario  Capas | La estructura de la aplicación se definirá por capas, teniendo capas de servicio, dominio, persistencia.  Se usará un arquetipo de tipo Maven. |
| D-03 | Patrón Complementario  Data Base per Service | De acuerdo con las consideraciones del ejercicio, cada microservicio debe manejar su propia base de datos, de manera separada, para persistir la información relacionada con la API. |
| D-04 | Infraestructura Base | Como infraestructura base se considera Docker como el soporte base para desplegar las Apis en un esquema de contenedores usando docker-compose. |
| D-05 | Patron Complementario  API Gateway | Se debe definir un único punto de acceso que permita la redirección a los servicios. |

# **Drivers de Arquitectura**

**Objetivo:**

Proveer a los usuarios un canal automatizado para la creación de cuentas, fidelizando al cliente y mejorando su experiencia.

# **Requerimientos funcionales significativos**

|  |  |  |
| --- | --- | --- |
| Número | Titulo | Descripción |
| RQ-01 | Creación de cuentas | Los usuarios deben poder crear cuentas en bolivianos y dólares |
| RQ-02 | Consulta de saldo | Los usuarios deben poder consultar su saldo en ambas monedas en cualquier momento. |
| RQ-03 | Deposito de fondos | Los usuarios deben poder depositar fondos en bolivianos y dólares. |
| RQ-04 | Retiro de fondos | Los usuarios deben poder retirar fondos en bolivianos y dólares. |
| RQ-05 | Conversión de moneda | Los usuarios deben poder convertir de una moneda a otra a una tasa de cambio de la fecha. |
| RQ-06 | Consulta de Transacciones | El usuario debe poder ver las transacciones de su cuenta |
| RQ-07 | Disponibilidad del servicio | El servicio debe estar disponible 24x7. |

# **Restricciones**

|  |  |  |
| --- | --- | --- |
| Número | Titulo | Descripción |
| RS-01 | Monedas de uso | Las monedas por usarse en el sistema son bolivianos y dólares. |
| RS-02 | Seguridad de conexión. | Los servicios deben estar expuestos sobre un protocolo seguro HTTPS y deben ser consumidos mediante una VPN. En ningún caso estarán expuestos a Internet.  Por otro lado, las APIs deben contemplar un manejo de autenticación básica a nivel encabezado. |
| RS-03 | Acceso a Base de Datos | El acceso a BD debe estar restringido únicamente a personal de producción mediante contraseña compartida y los logs de acceso deben ser guardados para temas de auditoria. |

# **Atributos de Calidad**

|  |  |
| --- | --- |
| Titulo | Descripción |
| **Seguridad** | Las cuentas y saldos de los usuarios deben ser protegidas en todo momento, por lo cual la comunicación debe ser cifrada desde y hacia el servidor, mediante protocolos de seguridad.  Ref. RS-02 - RS-03 - D-03 |
| **Disponibilidad** | La solución estará basada en contenedores, por lo cual las APIs deben estar en un esquema de alta disponibilidad y con capacidad de ser escalable de acuerdo con la necesidad del negocio y la demanda de los usuarios.  Ref. D-01 - RQ-07 |
| **Interoperabilidad** | Las APIs deben ser consumidas por estaciones de Cajero Automático, mismas que deben intercambiar información con las APIs. El uso de Microservicios permite la interoperabilidad a la solución.  Ref. D-01 – D-04 |
| **Performance** | Las APIs deben ser consumidas por estaciones de Cajero Automático, mismas que deben intercambiar información con las APIs. El uso de Microservicios permite la interoperabilidad a la solución.  Ref. D-01 – D-04 |
| **Mantenibilidad y Manejabilidad** | La solución de las APIs a nivel código deben ser lo mas liviano posible, no superar más de 300 líneas de código por clase. A nivel de métodos las clases no deben tener más de 10 métodos.  Las APIs deben contar con documentación autogenerada, para este fin se utilizará la herramienta Swagger.  La solución de cada API se trabajará en forma de módulos, basándose en la arquitectura de microservicios, para permitir la reutilización en otras APIs. |

# **Vistas de Arquitectura**

**Vista Lógica**

Diagrama, Esquemático

Descripción generada automáticamente

**Vista de Desarrollo**

Diagrama, Esquemático

Descripción generada automáticamente

**Vista de Procesos**

Diagrama

Descripción generada automáticamente

Diagrama

Descripción generada automáticamente

Diagrama

Descripción generada automáticamente

Diagrama

Descripción generada automáticamente

Diagrama

Descripción generada automáticamente

**Vista Física**

**Interfaz de usuario gráfica, Diagrama

Descripción generada automáticamente**

**Escenarios**

Diagrama

Descripción generada automáticamente