Ejercicio Práctico

- Cree un documento PDF con la respuesta al ejercicio.
    - El documento debe estar bien organizado y ser fácil de leer.
    - Agregue imágenes para explicar los diagramas. Puedes usar herramientas como http://draw.io/ para generar las imágenes.
    - Cada uno de los diagramas de C4 es un entregable importante. Asegúrese de hacerlos correctos y detallados.
    - Preste también atención a los otros diagramas requeridos.
    - Asegúrese de añadir cualquier texto explicativo que considere necesario.
- Suba el documento como respuesta a este ejercicio. Asegúrese de subirlo dentro del tiempo de resolución.

- **Adicional**, crear un **repositorio público en GitHub** y **subir el PDF a ese repositorio**.
  Colocar la **URL al repositorio en los comentarios de este ejercicio.**

Recomendaciones MUY valoradas:

- Cada decisión arquitectónica en el diseño debe tener una justificación teórica (Mínimo 2 por cada una). Ejemplo: ¿Por qué se decidió por determinados patrones, tecnologías, componentes, protocolos, etc.? ¿Qué opciones evaluó?

Se debe incluir los siguientes diagramas en el diseño de la solución:

- **Diagrama de Contexto:** Para usuarios no técnicos. Muestra sistemas internos y externos, actores clave, breves descripciones y conexiones explicativas.
- **Diagrama de Contenedores:** Para técnicos. Representa aplicaciones, servicios, bases de datos y mensajería, incluyendo componentes cloud sin mucho detalle. Conexiones con descripciones breves.
- **Diagrama de Componentes:** Mayor detalle técnico. Incluye microservicios, patrones arquitectónicos y protocolos de comunicación con seguridad. Destaca el uso de componentes cloud.

Descripción del ejercicio:

Usted ha sido contratado por una entidad llamada BP como arquitecto de soluciones para diseñar un sistema de banca por internet, en este sistema los usuarios podrán acceder al histórico de sus movimientos, realizar transferencias y pagos entre cuentas propias e interbancarias.

Toda la información referente al cliente se tomará de 2 sistemas, una plataforma Core que contiene información básica de cliente, movimientos, productos y un sistema independiente que complementa la información del cliente cuando los datos se requieren en detalle.

Debido a que la norma exige que los usuarios sean notificados sobre los movimientos realizados, el sistema utilizará sistemas externos o propios de envío de notificaciones, mínimo 2.

Este sistema contará con 2 aplicaciones en el Front, una SPA y una Aplicación móvil desarrollada en un Framework multiplataforma. (Mencione 2 opciones y justifique el porqué de su elección).

Ambas aplicaciones autenticarán a los usuarios mediante un servicio que usa el estándar Oauth2.0, para el cual no requiere implementar toda la lógica, ya que la compañía cuenta con un producto que puede ser configurado para este fin; sin embargo, debe dar recomendaciones sobre cuál es el mejor flujo de autenticación que se debería usar según el estándar.

Tenga en cuenta que el sistema de Onboarding para nuevos clientes en la aplicación móvil usa reconocimiento facial, por tanto, su arquitectura deberá considerarlo como parte del flujo de autorización y autenticación, a partir del Onboarding el nuevo usuario podrá ingresar al sistema mediante usuario y clave, huella o algún otro método especifique alguno de los anteriores dentro de su arquitectura, también puede recomendar herramientas de industria que realicen estas tareas y robustezca su aplicación.

El sistema utiliza una base de datos de auditoría que registra todas las acciones del cliente y cuenta con un mecanismo de persistencia de información para clientes frecuentes, para este caso proponga una alternativa basada en patrones de diseño que relacione los componentes que deberían interactuar para conseguir el objetivo.

Para obtener los datos del cliente el sistema pasa por una capa de integración compuesta por un api Gateway y consume los servicios necesarios de acuerdo con el tipo de transacción, inicialmente usted cuenta con 3 servicios principales, consulta de datos básicos, consulta de movimientos y transferencias que realiza llamadas a servicios externos dependiendo del tipo, si considera que debería agregar más servicios para mejorar el rendimiento de su arquitectura o agregar más servicios para mejorar la repuesta de información a sus clientes, es libre de hacerlo.

Consideraciones.

Para este reto, mencione aquellos elementos normativos que considere importantes a tener en cuenta para entidades financieras, Ejemplo ley de datos personales, seguridad, etc.

Garantice en su arquitectura, alta disponibilidad (HA), tolerancia a fallos, recuperación ante desastres (DR), Seguridad y Monitoreo, Excelencia operativa y auto-healing.

Si lo considera necesario, su arquitectura puede contener elementos de infraestructura en nube, Azure u AWS, garantice baja latencia, cuenta con presupuesto para esto.

En lo posible plantee una arquitectura desacoplada con elementos y reusables y cohesionados para otros componentes que puedan adicionarse en el futuro.

El modelo debe ser desarrollado bajo **modelo C4** (Modelo de Contexto, Modelo de aplicación o contenedor y componentes), describa hasta el modelo de componentes, la infraestructura la puede modelar como usted lo considere usando la herramienta de su preferencia.

Criterios de calificación

Se calificarán los siguientes elementos:

- Que la solución satisfaga los requerimientos y todo cuente con justificación
- Calidad y profundidad de los diagramas (Contexto, Contenedores y Componentes)
- Segmentación de Responsabilidades y Desacoplamiento
- Uso de patrones de arquitectura
- Integración con servicios externos
- Calidad de arquitectura de aplicación front-end y móvil
- Arquitectura de acceso a datos
- Conocimientos de Nube (AWS o Azure)
- Manejo de costos
- Arquitectura de Autenticación
- Arquitectura de Integración con Onboarding
- Diseño de Solución de Auditoría
- Conocimientos de regulaciones bancarias y estándares de seguridad
- Implementación de Alta Disponibilidad y Tolerancia a Fallos
- Implementación de Monitoreo

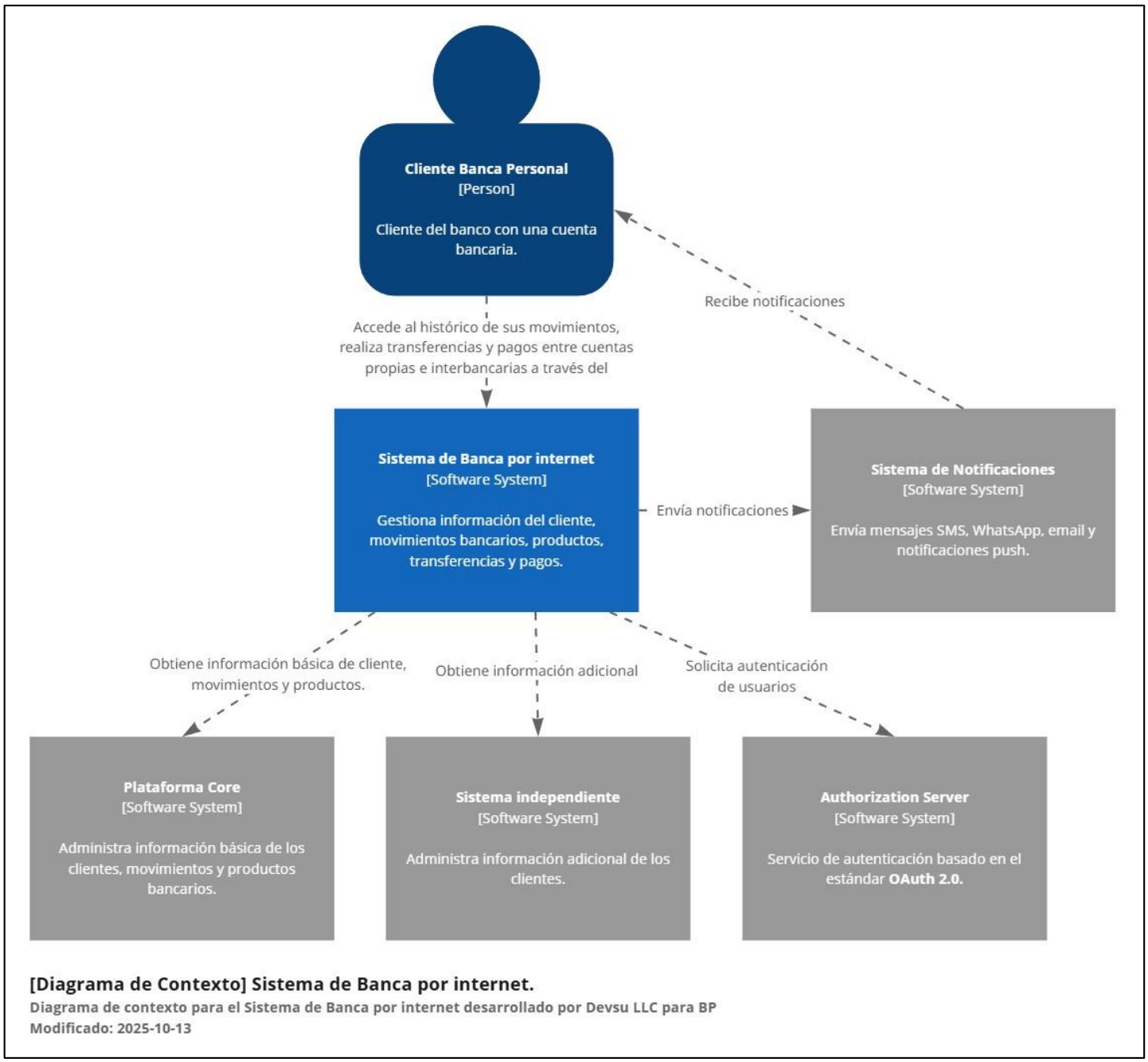
¡Éxitos!

Respuesta al Ejercicio Práctico

Creado por: Wilmer Mauricio Herrera Rentería

Diagrama de Arquitectura de Software (Modelo C4)

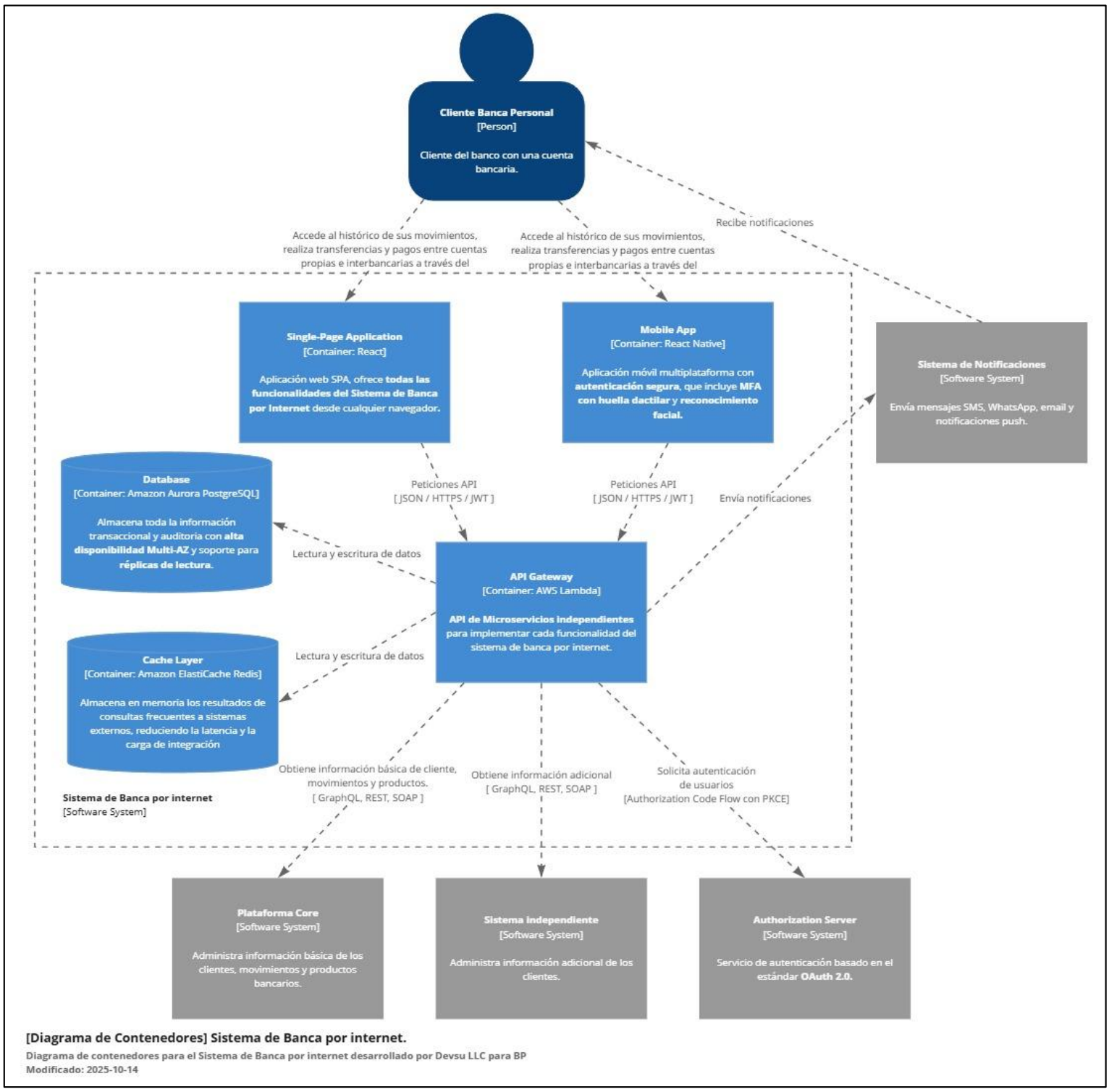
Diagrama de Contexto - Sistema de Banca por internet.



[Diagrama de Contexto] Sistema de Banca por internet.
Diagrama de contexto para el Sistema de Banca por internet desarrollado por Devsu LLC para BP
Modificado: 2025-10-13

[Diagrama de Contenedores] Sistema de Banca por internet.

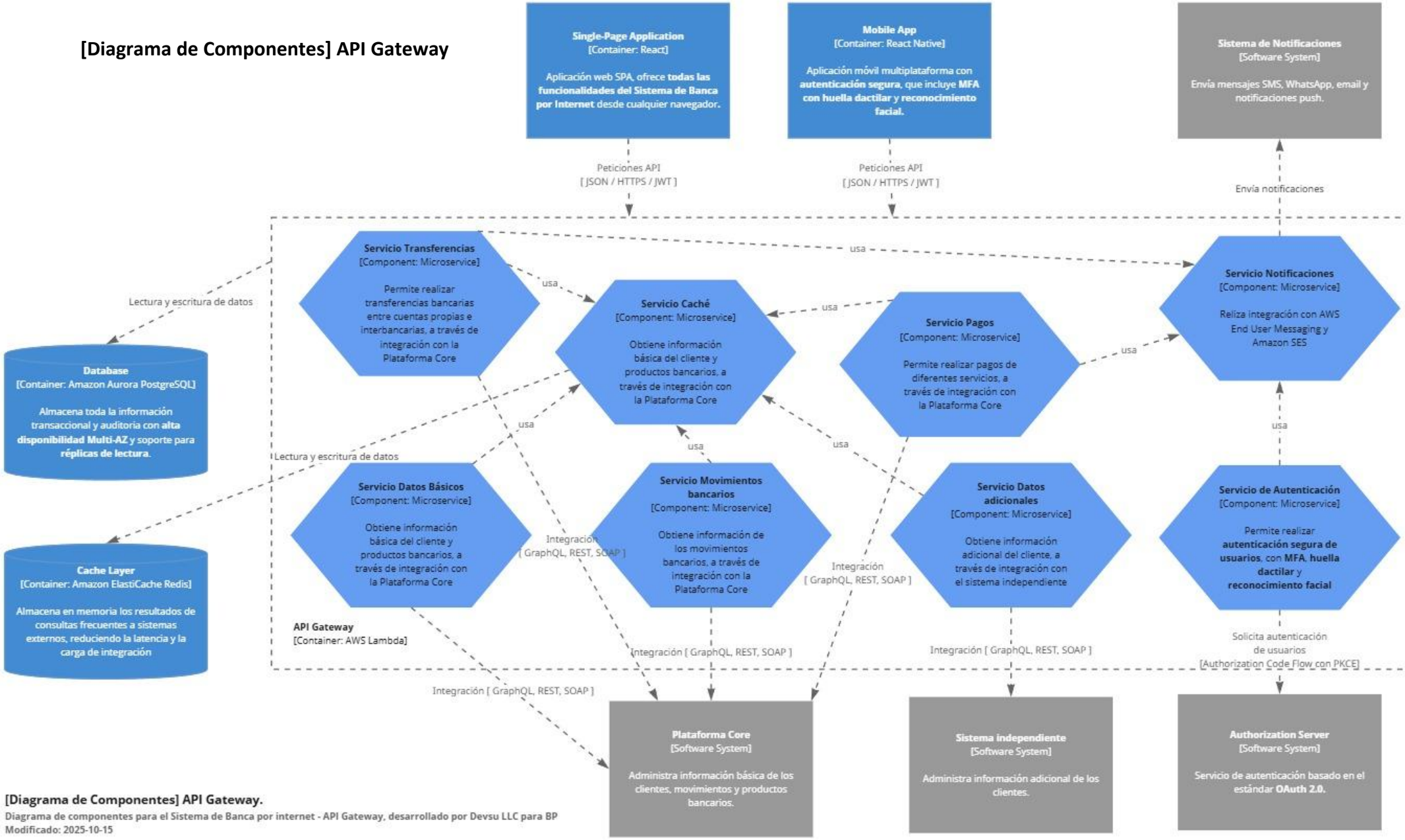
El Sistema de Banca por internet gestiona la información del cliente, movimientos bancarios, productos, transferencias y pagos.



[Diagrama de Contenedores] Sistema de Banca por internet.

Diagrama de contenedores para el Sistema de Banca por internet desarrollado por Devsu LLC para BP
Modificado: 2025-10-14

[Diagrama de Componentes] API Gateway



[Diagrama de Componentes] API Gateway.

Diagrama de componentes para el Sistema de Banca por internet - API Gateway, desarrollado por Devsu LLC para BP

Modificado: 2025-10-15

En esta Arquitectura se considera una Aplicación web SPA con React, y una Aplicación móvil multiplataforma con React Native, se elige estas tecnologías porque se requiere de un equipo con experiencia en React, si el equipo ya sabe React, aprender React Native es mucho más fácil, ya que los conceptos de componentes, estado y flujo de datos son similares.

Con React Native, el equipo puede crear aplicaciones móviles tanto para iOS como para Android sin necesidad de escribir dos aplicaciones separadas, y además tenemos integración con APIs de biometría (Face ID y Touch ID en iOS, Android Biometric API).

El despliegue de nuestra Aplicación web SPA con React se realizará con AWS Amplify, nos permite utilizar un CDN (Amazon CloudFront), despliegue continuo, generar SSL para soporte de HTTPS, seguridad a través de WAF, monitoreo y nos permite escalar a millones de usuarios.

Vamos a construir un API Gateway con tecnología Node.js + TypeScript y desplegado sobre AWS Lambda con API Gateway, por lo tanto, será una arquitectura Serverless, esto nos brinda muchas ventajas:

1. No necesitamos gestionar servidores.
2. Las funciones Lambda escalan automáticamente para manejar diferentes niveles de tráfico, asegurando alta disponibilidad y rendimiento.
3. Solo se paga por cada ejecución.
4. Genera logs automáticamente en CloudWatch Logs.
5. Podemos configurar alarmas de errores en CloudWatch para cada microservicio.
6. Activamos AWS X-Ray para Monitoreo y observabilidad.

Toda nuestra API Gateway brindará muchas funcionalidades, cada una de las funcionalidades será un microservicio y se ejecutará en un Lambda independiente, esto nos permite delegar responsabilidades y seguir agregando más funcionalidades con un nuevo microservicio.

Para la capa de persistencia de información utilizaremos Aurora PostgreSQL, que a diferencia de Oracle RDS que también tenemos en AWS, Aurora PostgreSQL es open source, tenemos alta disponibilidad con Multi-AZ y podemos configurar autoscaling. También necesitamos crear una read replica para acelerar las consultas solo de lectura, generar informes, integrar con BI, etc.

Se incluye una capa de caché con Amazon ElasticCache Redis (Cache distribuido en memoria de alta velocidad), para reducir la latencia en llamadas repetitivas a los sistemas externos (Plataforma Core y Sistema independiente), es decir antes de solicitar información de sistemas externos verificamos en nuestra capa de caché, si esta en caché devuelve inmediatamente. Se debe establecer la estrategia de cache de acuerdo a los tipos de datos que vamos a manejar y establecer el TTL.

Para el Sistema de Notificaciones utilizaremos AWS End User Messaging (SMS, WhatsApp y notificaciones push) y Amazon SES (correos electrónicos), podemos utilizar para:

- Verificación de contraseñas de un solo uso (OTP) por SMS o WhatsApp códigos OTP únicos y con plazos de entrega.
- Notificaciones push y emails para notificar inicios de sesión, transferencias, pagos enviados o recibidos, etc.

Las comunicaciones con la Plataforma Core y el Sistema independiente no están definidas, por lo tanto, se propone definir el tipo de tecnología para realizar las integraciones se puede utilizar GraphQL, REST, SOAP.

En el caso del Authorization Server, que será el sistema que implementa el estándar OAuth 2.0, se recomienda el flujo Authorization Code con PKCE por su resistencia ante ataques de interceptación y su compatibilidad con aplicaciones públicas sin almacenamiento de secretos, también podemos analizar una alternativa adicional y es utilizar Amazon Cognito, como nuestros servicios son Lambdas podemos validar los tokens JWT emitidos por el Authorization Server antes de permitir el acceso a los microservicios.

Regulaciones Bancarias y Estándares de Seguridad Aplicados:

PCI-DSS (Payment Card Industry Data Security Standard)

- **Cifrado de datos**: Protegemos los datos sensibles en tránsito con HTTPS en todas las capas de la arquitectura y cuando viajan hacia la base de datos Aurora PostgreSQL utiliza SSL/TLS, para la persistencia de datos y backups utilizamos AWS KMS, para el tratamiento de usuarios y contraseñas utilizamos pgcrypto.
- **Autenticación multifactor (MFA):** Para la validación de usuarios, tanto en la aplicación web como móvil.
- **Monitoreo y auditoría:** Asegura que todas las transacciones y accesos sean registrados y monitoreados.

ISO/IEC 27001 (Gestión de Seguridad de la Información)

Implementaremos el estándar internacional para la gestión de la seguridad de la información, debido que nuestra arquitectura que involucra datos sensibles, es esencial para proteger la información:

- **Confidencialidad:** Aseguramos de que la información solo sea accesible por las personas y sistemas autorizados.
- **Integridad:** Implementamos controles para garantizar que los datos no sean alterados sin autorización.
- **Disponibilidad:** Aseguramos que los servicios estén siempre disponibles.

Regulación sobre Protección de Datos Personales

La Ley Orgánica de Protección de Datos Personales exige que las instituciones financieras adopten medidas de seguridad para proteger los datos personales de los clientes. Esto incluye:

- Consentimiento explícito para la recolección y tratamiento de datos.
- Seguridad en el almacenamiento y manejo de datos.

OWASP Top 10

La OWASP (Open Web Application Security Project) publica un conjunto de directrices para la seguridad de aplicaciones web y móviles. Debemos estar al día, evaluar y probar las vulnerabilidades del OWASP Top 10 de nuestra arquitectura, para mitigar nosotros utilizamos:

- **Autenticación y gestión de sesiones:** Implementa OAuth 2.0 con PKCE para asegurar que los flujos de autenticación sean robustos.
- **Protección contra ataques de inyección** (como SQL injection o XML injection), especialmente en la interacción con Aurora PostgreSQL.
- **Protección contra Cross-Site Scripting (XSS) y Cross-Site Request Forgery (CSRF)**.
- Controles de Exposición de Datos Sensibles, todos nuestros microservicios están protegidos por Token JWT.

Para evaluar y probar las vulnerabilidades del **OWASP Top 10**, es fundamental realizar una combinación de pruebas manuales y automatizadas.