Requerimiento

Banca Por Internet para BP

Versión 1.0

IDEA PRESENTADA: Banca Por Internet

Autor: Bryan Rodríguez

SOFTKA

Febrero / 2025

Contenido

Entidad	Nombres y Apellidos	Cargo	Fecha
BP	NA	Líder de Tribu Canales	Junio /2024
BP	NA	Product Owner Canal Banca Web.	Junio /2024
BP	NA	Product Owner Canal Banca Móvil.	Junio /2024
BP	Bryan Rodríguez	Arquitecto de Soluciones	Junio /2024
Devsu	NA	Líder de Arquitectura	Junio /2024

1. DATOS INFORMATIVOS

Requerimiento:	Diseño de Sistema de Banca por Internet
Originador del Requerimiento:	Banco Pichincha (BP)

2. ALCANCE DE LA SOLUCIÓN

El sistema de Banca por Internet de BP debe permitir a los usuarios acceder al historial de movimientos de sus cuentas, así como realizar transferencias y pagos entre cuentas propias e interbancarias. Toda la información de los clientes será obtenida de dos sistemas: una plataforma Core que contiene información básica del cliente, movimientos y productos, y un sistema independiente que complementa la información del cliente cuando se requieren detalles adicionales.

Dado que la normativa exige que los usuarios sean notificados sobre las transacciones realizadas, el sistema utilizará al menos dos métodos de notificación, que pueden ser sistemas externos o propios.

Este sistema incluirá dos aplicaciones frontales: una aplicación de página única (SPA) y una aplicación móvil desarrollada en un framework multiplataforma. Se consideran dos opciones para el desarrollo móvil: Flutter y Xamarin, justificadas por su rendimiento, capacidad de personalización y eficiencia en el desarrollo.

Ambas aplicaciones autenticarán a los usuarios mediante un servicio basado en el estándar OAuth 2.0. La compañía ya cuenta con un producto configurado para este propósito, por lo que no es necesario implementar toda la lógica, aunque se deben proporcionar recomendaciones sobre el mejor flujo de autenticación a utilizar según el estándar.

El sistema de Onboarding para nuevos clientes en la aplicación móvil utilizará reconocimiento facial, por lo que la arquitectura debe considerar este aspecto como parte del flujo de autorización y autenticación. A partir del Onboarding, el nuevo usuario podrá ingresar al sistema mediante usuario y clave, huella dactilar u otro método de autenticación robusto.

El sistema incluirá una base de datos de auditoría que registre todas las acciones del cliente y debe persistir información para clientes frecuentes. Para este propósito, se propone una alternativa basada en patrones de diseño que relacione los componentes necesarios para lograr este objetivo.

Para obtener los datos del cliente, el sistema pasará por una capa de integración compuesta por un API Gateway y consumirá los servicios necesarios según el tipo de transacción. Inicialmente, se dispondrá de tres servicios principales: consulta de datos básicos, consulta de movimientos y transferencias, los cuales realizarán llamados a servicios externos según sea necesario. Se pueden agregar más servicios para mejorar el rendimiento y la respuesta a los clientes.

2.1. PROCESOS DE NEGOCIOS AFECTADOS

Procesos de Negocio	Detalle del impacto en el proceso
Gestión de Canales y Servicios	El área de canales debe implementar nuevas características en la banca por internet que generen valor al negocio. Esto incluye la optimización de la experiencia del usuario, la incorporación de nuevas funcionalidades y el mejoramiento de la eficiencia operativa. El objetivo es aumentar la satisfacción del cliente y la competitividad en el mercado, asegurando que las nuevas características tecnológicas se integren de manera eficiente y efectiva en los procesos actuales.

2.2. CANALES AFECTADOS

Canales	Detalle del impacto en el canal
Banca Móvil	Se realizarán modificaciones significativas en el proceso de consulta de movimientos, transferencias y pagos interbancarios en la aplicación de Banca Móvil. Estas modificaciones están diseñadas para mejorar la usabilidad y eficiencia, permitiendo a los usuarios gestionar sus transacciones de manera más rápida y sencilla desde sus dispositivos móviles. Además, se integrarán nuevas funcionalidades que optimizarán la experiencia del usuario y garantizarán un acceso seguro y confiable a los servicios bancarios móviles.

2.3. APLICACIONES AFECTADAS

Aplicaciones	Detalle del impacto en la aplicación
Banca Web	Se implementarán modificaciones en la aplicación de Banca Web para mejorar el proceso de consulta de movimientos, así como para facilitar pagos y transferencias entre cuentas propias e interbancarias. Estas mejoras están diseñadas para ofrecer una experiencia de usuario más intuitiva y eficiente, garantizando que los usuarios puedan acceder y gestionar sus transacciones de manera rápida y segura desde la web.
Banca Móvil	Similar a la Banca Web, la aplicación de Banca Móvil será modificada para optimizar la consulta de movimientos, y para permitir pagos y transferencias entre cuentas propias e interbancarias. Estas modificaciones asegurarán que los usuarios móviles tengan acceso a las mismas funcionalidades mejoradas, con una interfaz adaptada a dispositivos móviles para una experiencia de usuario óptima.

2.4. AREAS AFECTADAS

Empresa	Área	Detalle del impacto en Áreas
BP	Canales	Se realizará la preparación de comunicaciones y campañas dirigidas hacia los clientes para informarles sobre las nuevas características y mejoras en el sistema

		de Banca por Internet. Esto incluye el desarrollo de materiales informativos, campañas de marketing y atención personalizada para garantizar que los clientes comprendan y utilicen eficazmente las nuevas funcionalidades.
BP	Canales	Se llevará a cabo la capacitación de los especialistas en el nuevo proceso. Esta formación garantizará que el personal encargado de los canales esté completamente preparado para apoyar a los clientes y manejar cualquier consulta o problema relacionado con las nuevas características y funcionalidades del sistema.
BP	Operaciones	Se proporcionará capacitación sobre los nuevos procesos al personal de operaciones. Esto asegurará que todos los empleados comprendan los cambios y puedan operar eficientemente dentro del nuevo sistema, manteniendo la calidad del servicio y la continuidad de las operaciones.
BP	Tecnología de la Información (TI)	El área de TI será responsable de la implementación técnica del nuevo sistema, incluyendo la integración con los sistemas existentes, la gestión de la infraestructura y el soporte técnico continuo. Además, el personal de TI necesitará formación sobre las nuevas tecnologías y procedimientos para garantizar una implementación y operación exitosas.
BP	Seguridad	Se deberán actualizar y reforzar las políticas y procedimientos de seguridad para proteger la información de los clientes y garantizar el cumplimiento de las normativas. Esto incluye la implementación de nuevas medidas de seguridad, como la autenticación multifactor y el monitoreo de seguridad.
BP	Atención al Cliente	El personal de atención al cliente deberá recibir capacitación sobre el nuevo sistema y sus funcionalidades para proporcionar un soporte eficiente y resolver las consultas de los usuarios de manera efectiva. Además, se deberán actualizar los scripts y materiales de soporte para reflejar los cambios.
BP	Legal y Cumplimiento	Se necesitará la revisión y actualización de las políticas de privacidad y cumplimiento para asegurar que el nuevo sistema cumpla con todas las regulaciones locales e internacionales, incluyendo la protección de datos personales y las normativas financieras.

2.5. FUERA DE ALCANCE Y SUPOSICIONES

.

- **Acceso a aplicaciones del BP:** En caso de ser necesario, se deberán gestionar los permisos correspondientes para acceder a las aplicaciones del Banco Pichincha (BP). Esto incluye la obtención de las autorizaciones necesarias para integrar el nuevo sistema con las plataformas existentes.

- **Acompañamiento durante la fase de pruebas:** BP participará con acompañamiento directo durante la fase de pruebas del nuevo sistema. Esto asegurará que se identifiquen y solucionen posibles problemas de manera oportuna, y que el sistema cumpla con los requisitos y expectativas de la entidad antes de su implementación final.

2.6. POSIBLES RESTRICCIONES

No.	Detalle de Restricción
1	**Integración con Sistemas Legados**: La integración con los sistemas existentes del Banco Pichincha puede presentar desafíos debido a posibles incompatibilidades tecnológicas y la necesidad de asegurar una migración de datos sin interrupciones.
2	**Cumplimiento Normativo**: El sistema debe cumplir con todas las normativas locales e internacionales, incluyendo regulaciones sobre protección de datos y seguridad financiera. Esto puede implicar restricciones en la manera de manejar y almacenar los datos de los clientes.
3	**Presupuesto y Recursos**: Las limitaciones presupuestarias pueden afectar la selección de tecnologías, la contratación de personal especializado y la adquisición de herramientas y servicios necesarios para el desarrollo e implementación del sistema.
4	**Capacitación del Personal**: El tiempo y los recursos necesarios para capacitar al personal en el uso del nuevo sistema y en las nuevas políticas de seguridad y operativas pueden ser limitantes.
5	**Disponibilidad de Tecnología**: La disponibilidad de tecnologías y servicios en la nube necesarios para la implementación, como Azure o AWS, puede estar sujeta a restricciones geográficas o de infraestructura.
6	**Dependencias Externas**: La dependencia de proveedores externos para servicios como notificaciones SMS, correos electrónicos y autenticación biométrica puede introducir restricciones en términos de disponibilidad y tiempos de respuesta.

3. SOLUCIÓN PROPUESTA

3.1. MARCOS DE TRABAJO

Para optimizar la arquitectura empresarial del sistema de Banca por Internet, se emplearán los marcos de trabajo TOGAF y BIAN. BIAN, un marco de referencia específico para la industria financiera, se estructurará en diversas capas para facilitar la gestión y funcionalidad del sistema.

Capas de la Arquitectura

Capa de Aplicaciones:

Esta capa incluye tanto la aplicación de página única (SPA) como la aplicación móvil.

Capa de Procesos de Negocio:

En esta capa se gestionan los procesos de negocio, como las transferencias interbancarias, que requieren un proceso de autorización para montos superiores a un valor parametrizable, utilizando un sistema de gestión de procesos de negocio (BPM).

Capa de Eventos y Reglas de Negocio:

Aquí se gestionan las reglas de negocio y los eventos que controlan el flujo del sistema.

Capa de Actividades Robóticas:

No se prevé el uso de automatización de procesos robóticos (RPA) en este proyecto.

Capa de APIs:

Utilización de plataformas de gestión de APIs como Azure API Management y frameworks de desarrollo como Spring Boot para la creación de APIs.

Capa de Servicios (Arquitectura Orientada a Servicios - SOA / Microservicios):

Uso de IBM Integration Bus para la integración y acceso a los servicios CORE y satélites. Desarrollo de microservicios utilizando el Spring Framework.

Capa de Documentos Electrónicos:

Esta capa no se utilizará en el presente caso.

Capa de Componentes:

Incluye los procedimientos almacenados en la base de datos SQL Server.

Comprende sistemas como Core Banking, sistemas de Transferencias, sistemas de Autenticación IAM, sistemas de Correo Electrónico, sistemas de Notificación SMS, sistemas de Autenticación Biométrica, sistemas de Datos Adicionales y las bases de datos asociadas.

3.2. INFORMACION TÉCNICA

Database	Backend	Frontend
Sql Server AWS	Java – Framework Spring API REST Security Lógica de Negocios Acceso a Datos API Management (Azzure Api Managemet)	SPA: Angular o React App: Flutter o Xamarin, o Kotlin and Swift.

A continuación, se presentan los criterios generales para la selección de tecnologías para el front-end, back-end y la gestión de la base de datos, además de los aspectos relacionados con la seguridad y la gestión de APIs.

Gestión de la Base de Datos

Para la gestión de la base de datos, se recomienda utilizar SQL Server debido a su robustez, escalabilidad y soporte para transacciones ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad).

Gestión de APIs

Para la gestión de APIs, se sugiere utilizar Azure API Management, una solución de Microsoft que proporciona herramientas para crear, publicar, gestionar y supervisar APIs de forma segura y escalable. Azure API Management ofrece características como la seguridad integrada, la escalabilidad automática y la supervisión detallada del rendimiento de las APIs.

Back-End

Para el desarrollo del back-end de la aplicación bancaria, se recomienda utilizar Java en combinación con el framework Spring.

- **Java**: Java es un lenguaje de programación robusto y ampliamente utilizado en la industria, conocido por su estabilidad, seguridad y portabilidad. Es una opción confiable para aplicaciones empresariales de alto rendimiento.
- **Framework Spring**: Spring es un framework de desarrollo de aplicaciones Java ampliamente adoptado en la industria debido a su modularidad, escalabilidad y facilidad de integración. Spring Boot, una extensión de Spring, simplifica el desarrollo de aplicaciones web y ofrece características como la configuración automática y el empaquetado de aplicaciones listas para producción.

Seguridad

Para garantizar la seguridad de la aplicación bancaria, se recomienda implementar una arquitectura de seguridad en capas que incluya:

- **Autenticación y autorización basadas en tokens JWT**: Utilizando bibliotecas como Spring Security en el back-end para gestionar la autenticación y autorización de usuarios.
- **Cifrado de datos sensibles**: Los datos sensibles, como las contraseñas de los usuarios, deben cifrarse antes de almacenarse en la base de datos.
- **TLS/SSL**: Utilizar conexiones seguras a través de HTTPS para proteger la comunicación entre el cliente y el servidor.
- **Protección contra ataques de seguridad conocidos**: Implementar medidas de seguridad para protegerse contra vulnerabilidades comunes, como ataques de inyección SQL, XSS y CSRF.
- **Gestión de acceso e identidades (IAM)**: Utilizar soluciones como WSO2, Single Sign-On de RedHat o Azure IAM que cumplen con las normas de seguridad requeridas para la industria bancaria.
- **Almacén de secretos**: Utilizar Azure Key Vault para almacenar de forma segura los secretos y credenciales utilizadas por los desarrolladores.

Front-End

Para el desarrollo del front-end de la aplicación bancaria, se recomienda utilizar una arquitectura de una sola página (SPA), ya que proporciona una experiencia de usuario fluida y rápida, mejorando la velocidad y capacidad de respuesta de la aplicación. Angular es una excelente opción para implementar una SPA debido a su robustez, alto rendimiento y amplio soporte de la comunidad. Las razones para elegir Angular incluyen:

- **Framework sólido y maduro**: Angular es un framework bien establecido, respaldado por Google, que ofrece un amplio conjunto de características y herramientas para el desarrollo de aplicaciones web complejas.

- **TypeScript**: Angular se basa en TypeScript, un superconjunto de JavaScript que agrega características de programación orientada a objetos y detección de errores estáticos, mejorando la calidad y mantenibilidad del código.
- **Soporte de la comunidad**: Angular cuenta con una gran comunidad de desarrolladores y una abundante cantidad de recursos, documentación y bibliotecas disponibles para facilitar el desarrollo y la resolución de problemas.

Aplicación Móvil

Para el desarrollo de la aplicación móvil, se consideran dos opciones: Flutter y Xamarin.

- **Flutter**: Flutter es un framework de código abierto desarrollado por Google para construir aplicaciones móviles nativas para iOS y Android desde una única base de código. Algunas razones para elegir Flutter son su rendimiento, capacidad de personalización y eficiencia en el desarrollo.
- **Xamarin**: Xamarin es un framework de desarrollo de aplicaciones móviles multiplataforma que permite desarrollar aplicaciones nativas para iOS y Android utilizando el lenguaje de programación C#. Es una excelente opción para empresas que ya tienen experiencia en el ecosistema Microsoft y prefieren utilizar C#.

3.3. CUMPLIMIENTO NORMATIVO

Uso de OAuth2.0

Para la aplicación de Banca por Internet, es fundamental implementar OAuth2.0 utilizando el flujo de autorización y el código de autorización. Este enfoque asegura que las aplicaciones autentiquen a los usuarios de manera segura y gestionen las autorizaciones de acceso de forma eficiente.

Gestión de la Seguridad de la Información

La arquitectura de seguridad de la información no solo implica conceptos técnicos, sino también la implementación de tópicos administrativos esenciales. La efectividad de la seguridad de la información radica en la gestión y el mantenimiento continuo, lo cual se puede lograr a través de un Sistema de Gestión de la Seguridad de la Información (SGSI). Este sistema, propuesto por la serie 27000 de la ISO (Organización Internacional para la Estandarización) y específicamente la norma ISO 27001, define cómo organizar la seguridad de la información en cualquier tipo de organización.

La ISO 27001 proporciona un modelo internacional para establecer, implementar, utilizar, monitorear, revisar, mantener y mejorar un SGSI. La implementación de actividades de monitoreo y revisión garantiza que el sistema se mantenga actualizado y se mejore continuamente, lo cual impacta positivamente en la disponibilidad de los servicios tecnológicos y en el cumplimiento de los objetivos misionales de la organización.

Ley Orgánica de Protección de Datos Personales

En el Registro Oficial Suplemento No. 459 del 26 de mayo de 2021, se publicó la Ley Orgánica de Protección de Datos Personales. Esta ley tiene como objetivo garantizar el derecho a la protección

de datos personales, incluyendo el acceso, la decisión sobre la información y los datos personales, así como su correspondiente protección.

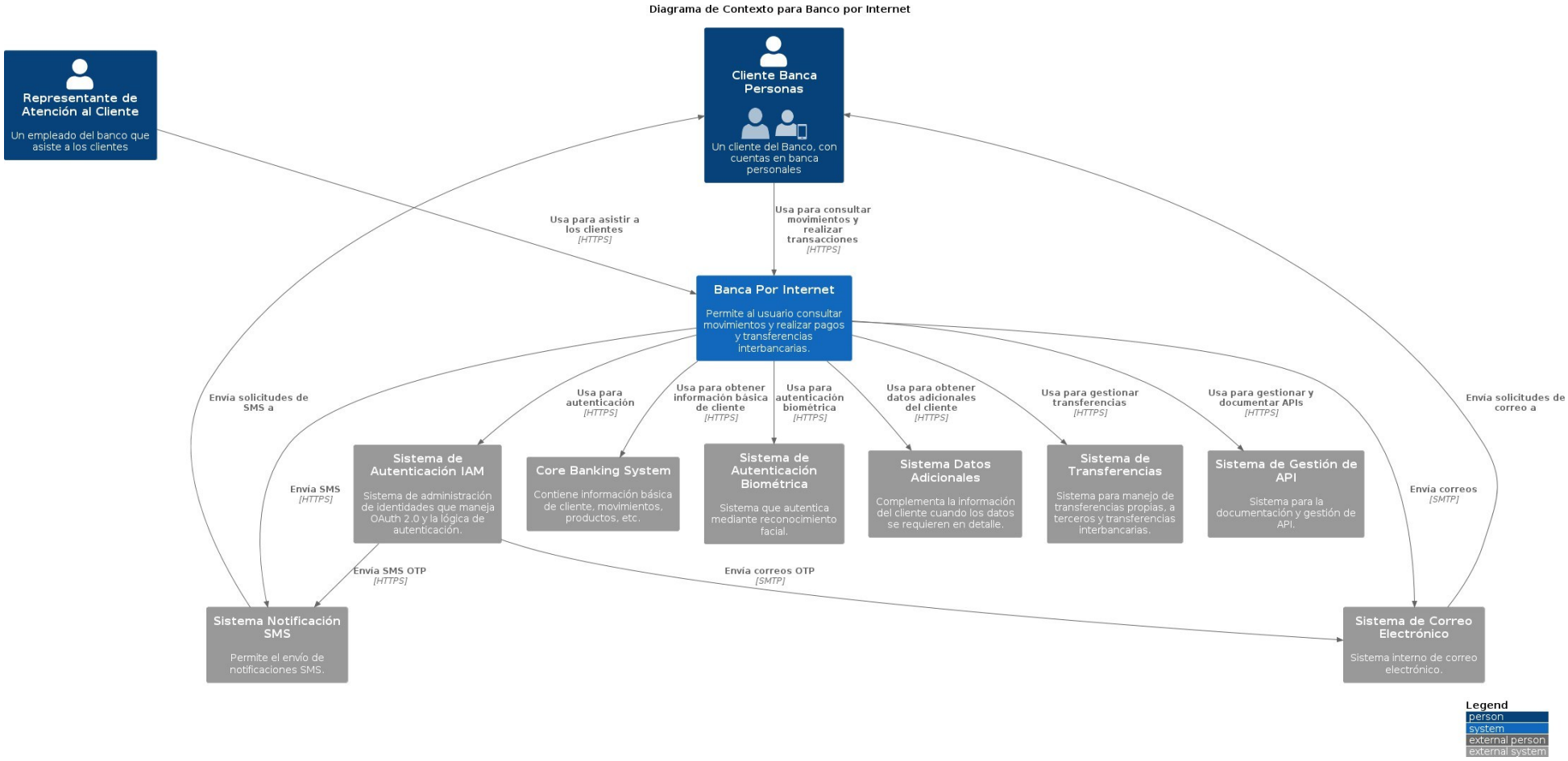
Consideraciones Legales para el Almacenamiento de Datos Bancarios

Es importante tener en cuenta que la ubicación de los datos bancarios de una persona está sujeta a regulaciones y consideraciones legales. En particular, los datos bancarios de una persona no pueden ser almacenados en una nube fuera del Ecuador. Esto asegura el cumplimiento de las normativas locales y la protección de los datos personales de acuerdo con las leyes ecuatorianas.

En conclusión, la implementación de una arquitectura de seguridad robusta y el cumplimiento de las normativas legales y regulatorias son fundamentales para el éxito del proyecto de Banca por Internet. Adoptar el uso de OAuth2.0, establecer un SGSI basado en la norma ISO 27001, y cumplir con la Ley Orgánica de Protección de Datos Personales garantizará que el sistema no solo sea seguro y confiable, sino también legalmente conforme y protegido contra amenazas y vulnerabilidades.

3.4. MODELO – C4 DE ARQUITECTURA DE SOFTWARE

DIAGRAMA DE CONTEXTO



Uso del Sistema de Banca por Internet

Los clientes de BP utilizan el sistema de banca por Internet para consultar el historial de movimientos de sus cuentas bancarias y realizar pagos y transferencias interbancarias. Este sistema cuenta con dos canales en su capa de aplicaciones: Banca Web (SPA) y Banca Móvil.

Core Banking System

El **Core Banking System** almacena los datos básicos de los clientes, así como información sobre sus cuentas, productos y transacciones. Este sistema registra todos los movimientos de las transacciones y permite realizar operaciones de débito y crédito para transferencias entre cuentas propias y cuentas de terceros.

Sistema de Autenticación Biométrica

El **Sistema de Autenticación Biométrica** incluye funcionalidades de reconocimiento facial. Utiliza la nube de Azure para comparar la foto en vivo del usuario con la foto registrada durante la creación del cliente, garantizando una autenticación segura y precisa.

Sistema de Datos Adicionales

El **Sistema de Datos Adicionales** es un sistema externo que complementa la información del cliente cuando se requieren detalles adicionales. Proporciona datos complementarios que no se encuentran en el Core Banking System.

Sistema de Transferencias

El **Sistema de Transferencias** es un sistema externo que gestiona las transferencias interbancarias. Valida si la transferencia es interna (dentro del mismo banco) o externa (hacia otros bancos). Además, permite realizar transferencias interbancarias directas hacia bancos asociados con BANRED o el Banco Central. Este sistema encapsula las reglas de negocio relacionadas con la gestión de comisiones, así como los montos máximos y mínimos permitidos para las transferencias.

Sistema de Autenticación IAM

El **Sistema de Autenticación IAM (Identity and Access Management)** se utiliza para la autenticación de usuarios. Se recomienda el uso del flujo de autorización implícita de OAuth2.0 para aplicaciones SPA y móviles, que incluye la autorización mediante códigos de seguridad (one-time passwords, OTP). El IAM genera estos códigos de seguridad y los envía a los usuarios por correo electrónico y SMS.

Sistema de Correo Electrónico

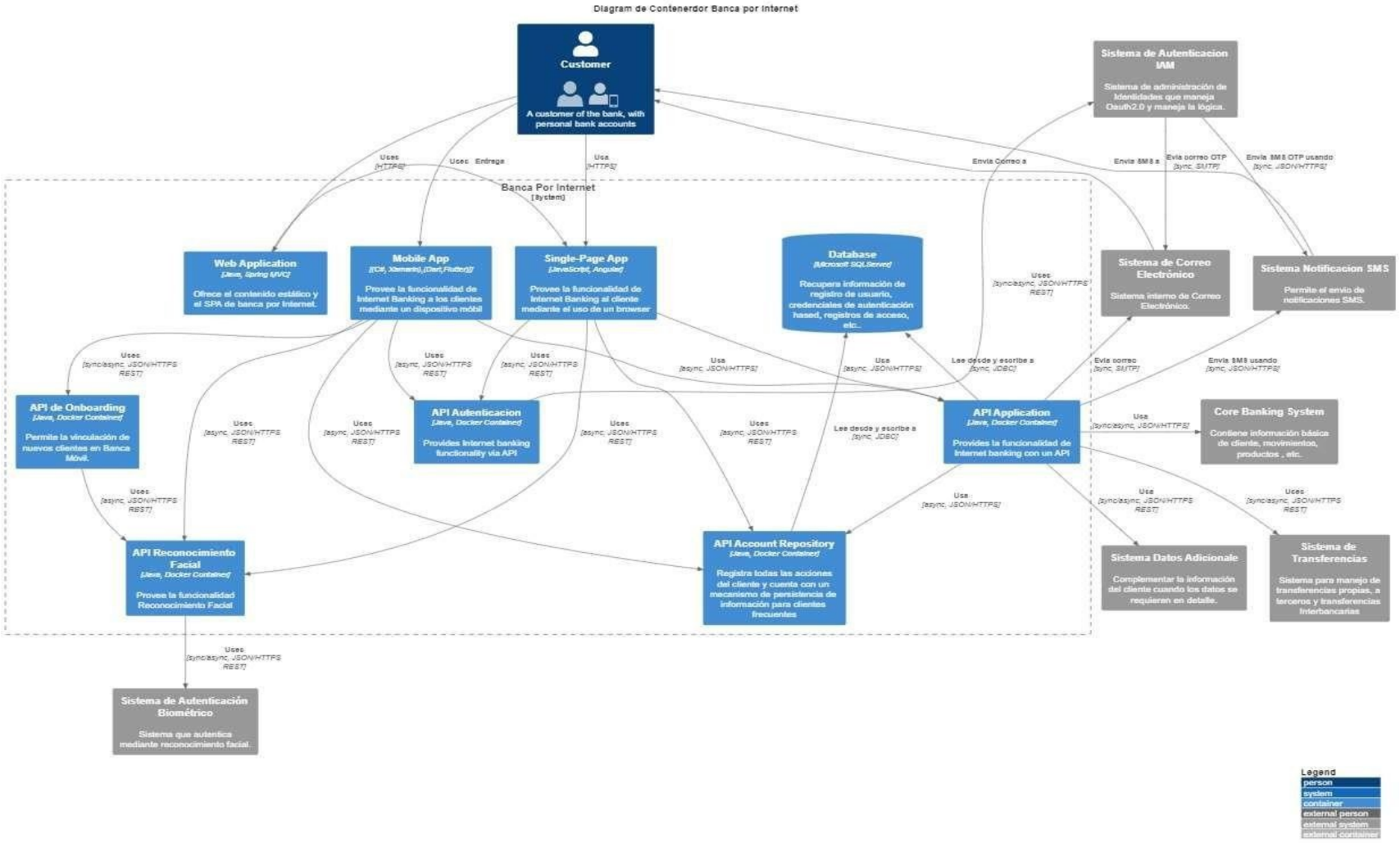
El **Sistema de Correo Electrónico** permite notificar a los clientes mediante el envío de notificaciones por correo electrónico. Este sistema asegura que los usuarios reciban actualizaciones y alertas importantes sobre sus transacciones y actividades bancarias.

Sistema de Notificación SMS

El **Sistema de Notificación SMS** permite enviar notificaciones a los clientes mediante mensajes SMS. Este sistema es crucial para la entrega rápida de alertas y códigos de seguridad, garantizando una comunicación efectiva y oportuna con los usuarios.

Estos componentes, integrados de manera eficiente, aseguran que el sistema de banca por Internet de BP sea robusto, seguro y capaz de satisfacer las necesidades de los clientes en cuanto a consultas de movimientos, pagos y transferencias interbancarias.

DIAGRAMA DE CONTENEDOR



Aplicación Web

La aplicación web de BP está desarrollada utilizando Java y Spring MVC, ofreciendo contenido estático (HTML, CSS y JavaScript), incluyendo la aplicación de página única (SPA).

Single-Page App (SPA) La SPA está desarrollada en JavaScript y Angular, y se ejecuta en el navegador del cliente, proporcionando todas las funciones necesarias para la banca por Internet.

Revisión de Alternativas React es una biblioteca de JavaScript, para construir interfaces de usuario interactivas y reutilizables. Sin embargo, la tendencia de Banco es usar Angular.

En varios sistemas bancarios Open Source, se observó que muchos utilizan Angular en su front-end. Angular, desarrollado por Google, es un framework robusto que facilita la creación de aplicaciones web modernas y escalables.

Aplicación Móvil

La aplicación móvil proporciona la funcionalidad de la banca por Internet a través de dispositivos móviles. Se recomienda el uso de frameworks multiplataforma como Flutter o Xamarin para su desarrollo.

Experiencia con Herramientas Multiplataforma Anteriormente, se utilizaron herramientas nativas como Android Studio con Kotlin y Java para desarrollar aplicaciones para Android. Sin embargo, al buscar una solución que también funcione en iOS, se optó por Flutter. Flutter permite crear aplicaciones nativas para iOS y Android desde una única base de código, además de generar aplicaciones para Windows y la web con el mismo código fuente.

Recomendaciones de Herramientas Multiplataforma

- **Flutter**:
  - **Desarrollo Rápido**: Hot reload permite realizar cambios en tiempo real.
  - **Interfaz de Usuario Personalizable**: Ofrece widgets personalizables para crear interfaces atractivas.
  - **Alto Rendimiento**: Compila a código nativo para un rendimiento óptimo.
  - **Unificación de Código**: Permite un único código base para iOS y Android.
  - **Comunidad Activa**: Soporte y recursos frecuentes por parte de Google.
- **Xamarin**:
  - **Reutilización de Código**: Comparte gran parte del código entre iOS y Android.
  - **Desarrollo Rápido**: Eficiencia de C# y .NET.
  - **Acceso a Funcionalidades Nativas**: Aprovecha las API nativas de las plataformas.
  - **Integración con Visual Studio**: Ambiente de desarrollo robusto y familiar.
  - **Soporte de Comunidad**: Documentación extensa y comunidad activa.

Arquitectura de la Aplicación Móvil con Flutter

- **Flutter Widgets**: Construcción de la interfaz y gestión de la lógica de presentación.
- **Comunicación con Backend**: Llamadas a API RESTful.
- **Gestión de Estado**: Uso de Provider o Riverpod.
- **Integración con Servicios de Notificación**: Conexión con servicios de correo y SMS.
- **Acceso a Funciones Nativas**: Canales de plataforma para acceso nativo.

Herramientas Recomendadas:

- **Flutter** y **Dart**: Para interfaz de usuario y lógica.
- **Provider** o **Riverpod**: Gestión del estado.
- **Dio**: Biblioteca para peticiones HTTP.

Arquitectura de la Aplicación Móvil con Xamarin

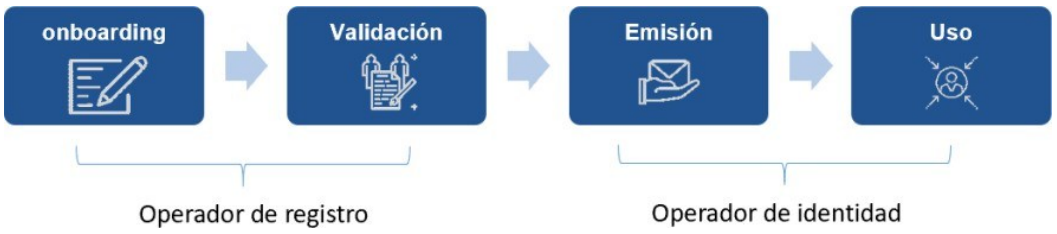
- **UI/UX Consistente**: Uso de Xamarin.Forms.
- **Comunicación con Backend**: Servicios RESTful.
- **Acceso a Funciones Nativas**: Implementación de Xamarin Native.
- **Flujo de Onboarding con Reconocimiento Facial**: Integración de bibliotecas de reconocimiento facial.

Herramientas Recomendadas:

- **Xamarin.Forms**: Interfaz de usuario compartida.
- **Xamarin.Essentials**: Acceso a características nativas.
- **Xamarin.Auth**: Implementación de OAuth2.0.
- **Bibliotecas de Reconocimiento Facial de Xamarin**: Onboarding con reconocimiento facial.

La elección de tecnologías como Angular para la web y Flutter o Xamarin para aplicaciones móviles asegura una plataforma de banca por Internet robusta, eficiente y capaz de proporcionar una excelente experiencia de usuario.

Api de Onboarding:



La aplicación móvil de BP cuenta con un sistema de onboarding que incluye un módulo de reconocimiento facial, registro de usuario y autenticación posterior al onboarding. Este proceso se alinea con la ley de protección de datos personales, comenzando con la presentación de términos y condiciones, que incluyen el acuerdo sobre el uso de datos personales. Para validar la identidad del usuario, se genera un código de seguridad enviado a su celular, cuya confirmación se registra en los logs como aceptación digital.

El proceso de onboarding incluye la captura de datos mínimos del cliente a través de un formulario digital que solicita identificación y correo electrónico. Con esta información, el banco verifica si el cliente ya es usuario del banco o consulta datos alternos si no lo es. Si es necesario, se realiza una llamada al registro civil para obtener datos biométricos. Luego, se lleva a cabo el enrolamiento biométrico facial utilizando el sistema de autenticación biométrica. Se realizan validaciones contra listas negras y observadas, así como otras políticas internas del banco. Finalmente, se crea el perfil del cliente, se generan los productos asociados y se habilitan los servicios.

Base de Datos

Se utiliza una base de datos Microsoft SQL Server 2019 para almacenar información como registros de usuario, credenciales de autenticación hasheadas y registros de acceso.

API Application

La aplicación API proporciona la funcionalidad de banca por internet a través de una interfaz de programación de aplicaciones (API), utilizando Java y Docker, y está instalada en Azure API Management. Esta API se comunica con el Core Banking mediante un bus de datos que utiliza una interfaz JSON/HTTPS, permitiendo el acceso a microservicios que gestionan información sobre el historial de cuentas bancarias, transacciones de débito y crédito para transferencias, y consultas de datos básicos del cliente. También integra el sistema de correo electrónico y el sistema de notificaciones SMS para enviar comunicaciones a los clientes.

API de Reconocimiento Facial

Un servicio de reconocimiento facial que proporciona funcionalidad biométrica a través de una API, utilizando Java y Docker, instalado en Azure API Management. Este servicio interactúa con la API de reconocimiento facial de Azure.

API de Autenticación

Un servicio de autenticación que gestiona la autenticación de usuarios para el sistema de banca por internet. Utiliza Java y Docker, está instalado en Azure API Management y se integra con los microservicios del IAM del banco, generando códigos de seguridad para cumplir con OAuth2.0.

API de Auditoría

Un servicio de auditoría que registra todas las acciones de los clientes y cuenta con un mecanismo de persistencia para información de clientes frecuentes. Utiliza el patrón de diseño de repositorio para separar la lógica de acceso a datos de la lógica empresarial, proporcionando métodos específicos para manejar datos de clientes frecuentes.

Patrones de Diseño

Arquitectura de Microservicios: El sistema está dividido en contenedores independientes, representando servicios como la aplicación web, móvil, y las APIs de autenticación y reconocimiento facial. Esto permite escalabilidad e independencia en la implementación de cada componente.

Patrón Modelo-Vista-Controlador (MVC): Separa las responsabilidades entre la vista (aplicaciones web y móviles), el controlador (APIs de autenticación y otras APIs) y el modelo (base de datos y repositorio de cuentas).

Patrón Repositorio: Gestiona la persistencia de datos de clientes frecuentes, separando la lógica de acceso a datos de la lógica de negocio.

Patrón de Autenticación: Maneja la autenticación de usuarios mediante APIs dedicadas, permitiendo separación de preocupaciones y fácil escalabilidad.

Patrón de Comunicación Asíncrona: Utiliza comunicación asíncrona JSON/HTTPS REST entre los diferentes componentes.

Patrón de Mensajería: Usa sistemas externos de correo electrónico y notificación SMS para enviar mensajes a los clientes.

Integración con Servicios Externos

Integración con APIs HTTP/HTTPS: La aplicación web, SPA y móvil consumen la API de backend mediante comunicación asíncrona JSON/HTTPS.

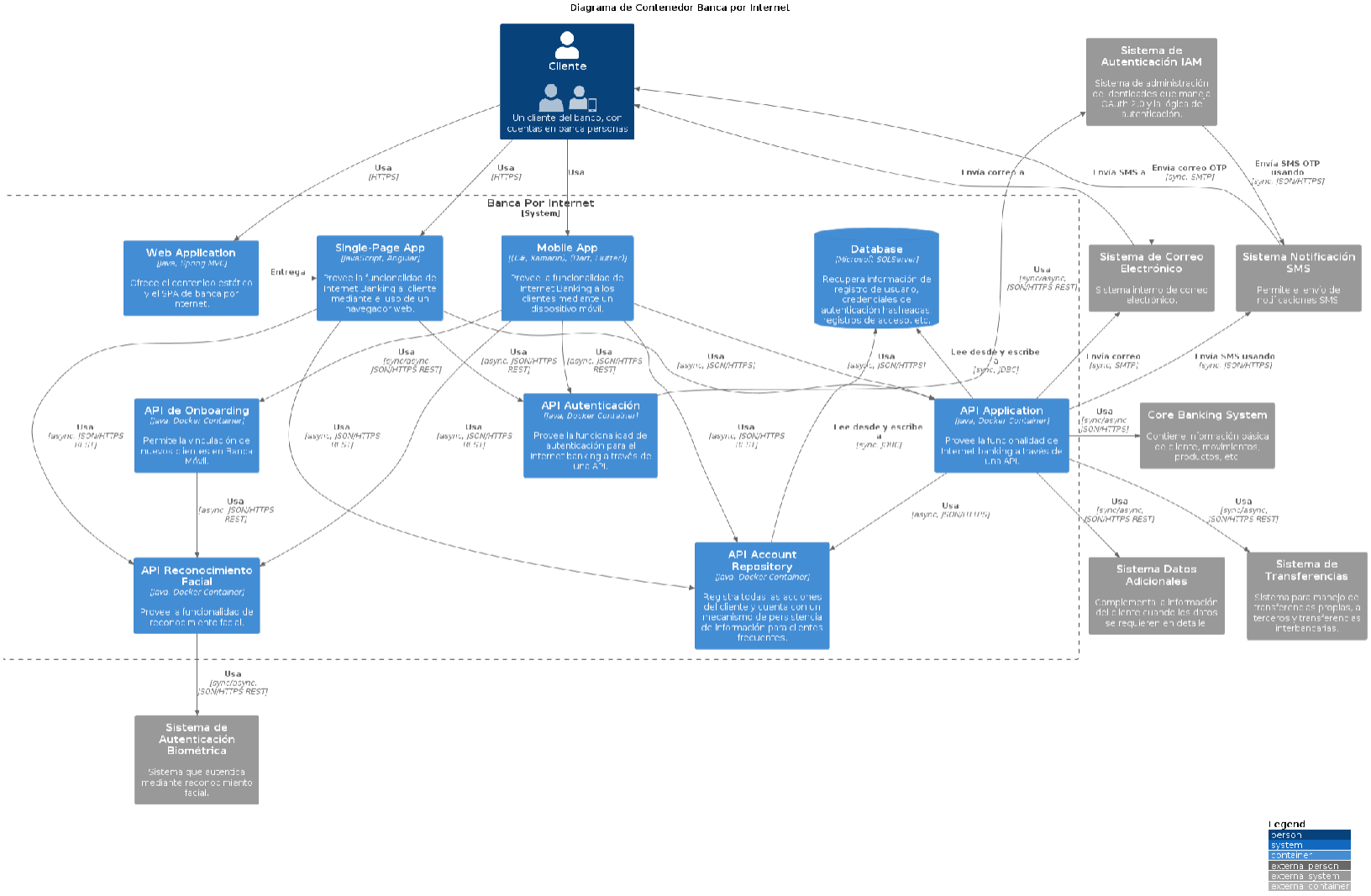
Integración con Bases de Datos usando JDBC: La API Application se integra con la base de datos mediante JDBC para realizar operaciones en SQL Server.

Integración con Sistemas de Correo Electrónico y SMS: Utiliza SMTP para el sistema de correo electrónico y JSON/HTTPS para el sistema de notificación SMS.

Integración con APIs RESTful: Las APIs de autenticación, reconocimiento facial y otras (como la de Onboarding) se exponen como servicios RESTful y son consumidas por las aplicaciones web y móvil.

Integración con Sistemas de Autenticación IAM: La API de autenticación utiliza IAM para manejar la autenticación de usuarios, interactuando mediante solicitudes asincrónicas JSON/HTTPS REST para generar tokens de acceso.

Diagrama de Componentes de Banca por Internet - API Application



El diagrama ilustra la arquitectura de componentes para la aplicación de API del sistema de Banca por Internet. Esta aplicación ofrece funcionalidades bancarias a través de una interfaz de programación de aplicaciones (API), facilitando la interacción efectiva con diversos sistemas y clientes.

Componentes Principales

1. **Single Page Application (SPA)**:
    - Proporciona la funcionalidad completa de la banca por Internet a través de un navegador web.
    - Desarrollada en JavaScript y Angular.
2. **Mobile App**:
    - Ofrece la funcionalidad esencial de la banca por Internet a través de dispositivos móviles.
    - Desarrollada utilizando tecnologías multiplataforma como Xamarin o Flutter.
3. **Database**:
    - Almacena información relacionada con los usuarios, credenciales de autenticación hasheadas, registros de acceso, entre otros datos.
    - Utiliza Microsoft SQL Server 2019.
4. **API Account Repository**:
    - Registra todas las acciones del cliente y proporciona un mecanismo de persistencia de información para clientes frecuentes.
    - Implementada como una API en un contenedor Docker utilizando Java.

Integraciones Externas

- **Core Banking System**:
    - Contiene información básica de cliente, movimientos, productos, entre otros.
    - Se integra con el API Account Repository para proporcionar información detallada sobre los clientes.
- **Sistema de Transferencias**:
    - Maneja transferencias propias, a terceros e interbancarias.
- **Sistema de Correo Electrónico y Sistema de Notificación SMS**:
    - Utilizados para enviar notificaciones por correo electrónico y mensajes SMS, respectivamente.
- **Sistema de Datos Adicionales**:
    - Complementa la información del cliente cuando se requieren detalles adicionales.

Arquitectura de Componentes

La arquitectura de componentes se organiza en varios módulos clave:

- **Single Page Application (SPA)**: Ofrece la funcionalidad completa de la banca por Internet a través de un navegador web. Desarrollada en Angular.
- **Mobile App**: Proporciona funcionalidades esenciales de la banca por Internet en dispositivos móviles. Desarrollada utilizando tecnologías multiplataforma como Xamarin o Flutter.
- **API Application**: Abarca los controladores para diferentes operaciones bancarias, manejo de transferencias, consultas de movimientos, y gestión de datos básicos y complementarios. Incluye servicios para el envío de correos electrónicos y SMS, así como interfaces para acceder al Core Banking System.

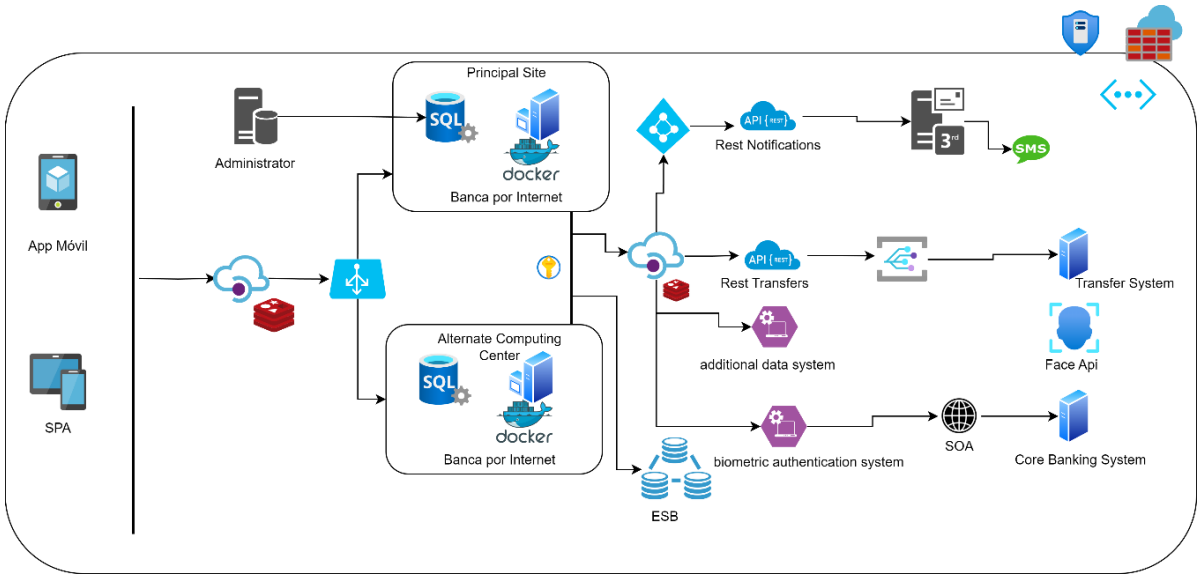
- **API Management**: Incluye un API Gateway para enrutar las solicitudes de manera eficiente y proporcionar acceso a diferentes servicios a través de interfaces API.

Relaciones entre Componentes

- **Controladores de la API Application**:
  - **Transferencias Controller**: Maneja las operaciones de transferencia propias e interbancarias.
  - **Histórico de Movimientos Controller**: Proporciona acceso a los registros de movimientos de las cuentas.
  - **Consulta de Datos Básicos Controller**: Facilita la consulta de información básica de los clientes.
  - **Consulta de Datos Complementarios Controller**: Permite la consulta de información adicional del cliente.
- **Servicios Core**:
  - **Transferencias Core**: Realiza operaciones de transferencia dentro del banco y hacia otras entidades.
  - **Histórico de Movimientos Core**: Proporciona acceso a los registros históricos de movimientos.
  - **Consulta de Datos Básicos Core**: Permite la consulta de datos básicos de los clientes desde el Core Banking System.
  - **Consulta de Datos Complementarios Core**: Facilita el acceso a datos adicionales necesarios para la operación bancaria.
- **Integración con Sistemas Externos**:
  - **Core Banking System**: Contiene información básica de cliente, movimientos, productos, etc., y se integra con los componentes de la API Application para proporcionar información detallada.
  - **Sistema de Transferencias**: Gestiona las transferencias interbancarias y a terceros.
  - **Sistema de Correo Electrónico**: Utilizado para enviar notificaciones a los clientes.
  - **Sistema de Notificación SMS**: Envío de mensajes SMS a los clientes para notificaciones importantes.
  - **Sistema de Datos Adicionales**: Complementa la información del cliente cuando se requieren detalles adicionales.
- **API Account Repository**:
  - Utiliza la base de datos para leer y escribir datos relacionados con las acciones del cliente y la persistencia de información para clientes frecuentes.
- **Interacción entre Componentes**:
  - Los componentes de la API Application y API Management se comunican entre sí y con los sistemas externos para proporcionar una experiencia bancaria completa y eficiente a través de interfaces API.

Este diagrama de componentes refleja una arquitectura de microservicios bien organizada, con una clara separación de responsabilidades y una interacción eficiente entre los distintos servicios y sistemas, asegurando una gestión óptima y escalabilidad de las funcionalidades bancarias.

Diagrama de Arquitectura General de la Plataforma



Canales de Atención y Comunicación

Los canales de atención del sistema de Banca por Internet utilizan HTTPS, asegurado con SSL/TLS, para comunicarse con Azure API Management. Las APIs emplean JWT para la autenticación, y Azure API Management se coloca detrás de un Azure Load Balancer, que distribuye la carga entre los servidores de backend. Estos servidores de backend operan en contenedores Docker, asegurando alta disponibilidad y facilitando las prácticas de DevSecOps.

El diagrama muestra un sitio principal y un sitio alterno, implementando redundancia a nivel de sitio y servidores críticos. Esta arquitectura en la nube de Azure asegura la escalabilidad del sistema.

Azure Redis Cache: Azure Redis Cache es un servicio de almacenamiento en caché en memoria basado en la popular base de datos Redis. Proporciona acceso rápido a datos temporales que se utilizan con frecuencia, mejorando el rendimiento y la capacidad de respuesta de las aplicaciones.

Almacenamiento de Datos

Es esencial cumplir con la legislación que exige que los datos confidenciales y personales de los ciudadanos se almacenen dentro del país. Esto garantiza el cumplimiento de las normativas de protección de datos.

Autenticación y Seguridad

El sistema de autenticación utiliza un IAM (Identity and Access Management) que soporta OAuth2.0. Los usuarios se autentican mediante usuario y contraseña, con un segundo factor de autenticación que es un código de seguridad. Alternativamente, se puede utilizar la autenticación biométrica mediante Face API. Este sistema emplea Azure Cognitive Services para integrar el reconocimiento facial en el flujo de autenticación.

Servicios de Notificación

- **Correo Electrónico**: El envío de notificaciones por correo electrónico se realiza a través de un microservicio que utiliza el protocolo SMTP.
- **Notificaciones SMS**: Las notificaciones SMS se envían mediante un microservicio que interactúa con proveedores de servicios SMS. Esto puede incluir empresas como Movistar o Claro, o distribuidores de servicios SMS.

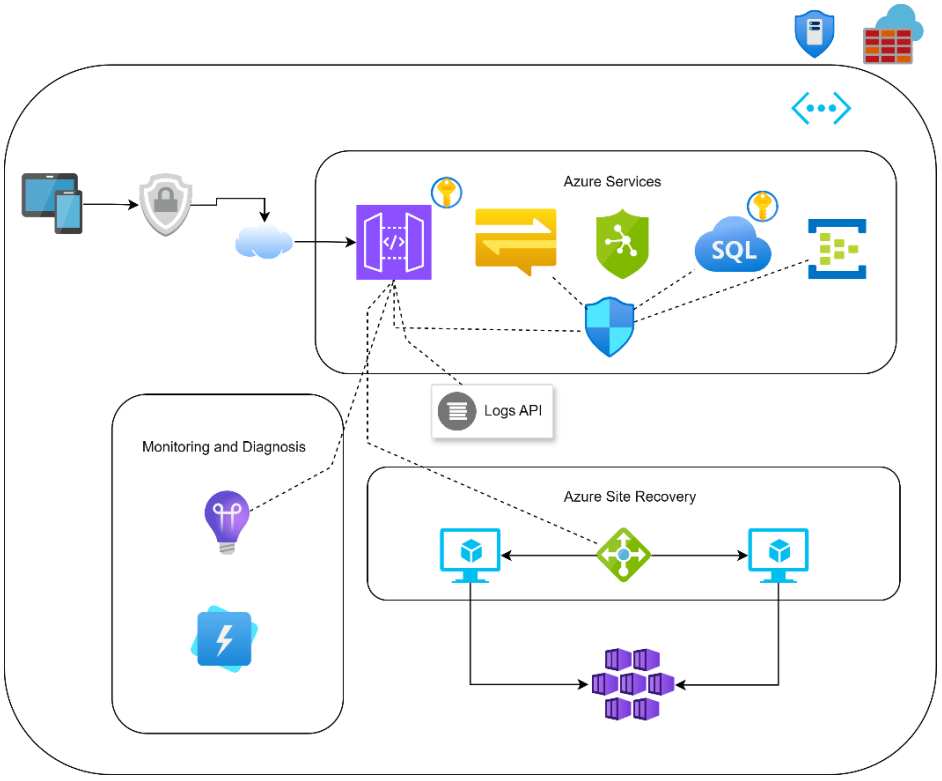
Servicios de Transferencias

La banca por Internet accede a los servicios de transferencias mediante APIs REST, que se conectan al switch transaccional para realizar transferencias.

Acceso a Core Banking

Para las peticiones al Core Banking, se utilizan microservicios que se comunican a través de un bus de datos de IBM. Este bus se encarga de gestionar las solicitudes a los servicios del Core Banking System, asegurando un acceso eficiente y seguro a los datos y funcionalidades requeridas.

Diagrama de Arquitectura de Infraestructura General en Azure



El diagrama ilustra la infraestructura general implementada en Azure, detallando los componentes clave que aseguran la alta disponibilidad, integración, notificaciones, seguridad, y monitoreo y gestión.

Alta Disponibilidad

- **Redundancia en Servidores Críticos**: La infraestructura incluye tanto un sitio principal como un sitio alterno, asegurando la continuidad del servicio en caso de fallos.
- **Escalabilidad**: Se utilizan servicios en la nube de Azure para garantizar que el sistema pueda escalar según la demanda.

Capa de Integración

- **Azure API Management**: Configurado para manejar las solicitudes de los clientes, dirigiéndolas a los servicios correspondientes. Actúa como un gateway para todas las API, asegurando una comunicación eficiente y segura.

Notificaciones

- **Azure Notification Hub**: Envía notificaciones a los usuarios sobre movimientos y actividades en sus cuentas, integrándose con sistemas externos cuando es necesario.

Seguridad

- **Azure Security Center**: Proporciona medidas de seguridad avanzadas para proteger los datos del cliente y las transacciones financieras.
- **Azure Key Vault**: Almacena y gestiona claves de API y otros secretos sensibles, garantizando que la información crítica esté segura.
- **Azure Firewall**: Protege la red contra amenazas externas.
- **Azure DDoS Protection**: Protege contra ataques de denegación de servicio distribuido.

Monitoreo y Gestión

- **Azure Application Insights**: Monitorea el rendimiento de las aplicaciones y ofrece insights detallados sobre el comportamiento del usuario y la salud del sistema.
- **Azure Monitor**: Supervisa en tiempo real la salud de todos los servicios, proporcionando alertas y métricas detalladas para detectar y responder rápidamente a cualquier anomalía.

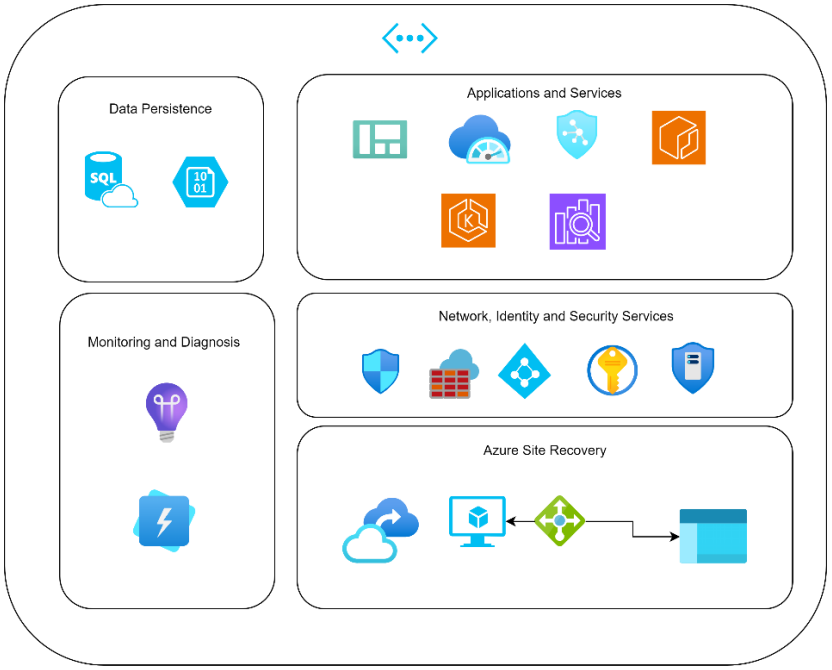
Descripción Detallada de los Componentes

1. **Dispositivos de Usuario**:
    - Los usuarios acceden al sistema a través de dispositivos móviles y navegadores web, asegurados con SSL para garantizar una conexión segura.
2. **Azure Services**:
    - **API Gateway**: Maneja la entrada de solicitudes y direcciona el tráfico a los servicios backend adecuados.
    - **Notification Hub**: Gestiona y envía notificaciones push a los dispositivos de los usuarios.
    - **Azure Security Center**: Implementa y gestiona políticas de seguridad para proteger los datos y las transacciones.
    - **Azure SQL**: Almacena datos relacionales de manera segura y escalable.
    - **Event Hub**: Facilita la ingesta y procesamiento de eventos en tiempo real.
3. **Entornos de Producción y Alterno**:
    - **Default/Production**: El entorno principal para la operación diaria.
    - **Alternate**: Un entorno alterno configurado para tomar el relevo en caso de falla del entorno principal, asegurando la continuidad del servicio.
4. **Servicios Backend**:
    - **Transferencias**: Gestiona las transferencias financieras entre cuentas.
    - **CORE**: Centraliza las operaciones críticas del sistema bancario.
    - **Otros Servicios**: Proporciona funcionalidades adicionales necesarias para el funcionamiento del sistema.
5. **Seguridad y Monitoreo**:
    - **Azure Key Vault**: Gestiona las claves y secretos de manera segura.
    - **Logs API**: Recolecta y almacena logs para auditorías y monitoreo.
    - **Application Insights y Azure Monitor**: Supervisan y reportan sobre la salud y rendimiento del sistema, facilitando una gestión proactiva.

- **Azure Virtual Network (VNet)**: Encapsula todos los componentes dentro de una red privada y segura.
- **Network Security Groups (NSG)**: Proporciona reglas de seguridad para controlar el tráfico hacia y desde los recursos.

Este diseño asegura que el sistema de banca por Internet sea robusto, seguro y capaz de manejar una alta carga de usuarios mientras mantiene un rendimiento óptimo y un alto nivel de disponibilidad.

Diagrama de Arquitectura de Infraestructura Sitio Principal en Azure



El diagrama ilustra la infraestructura del sitio principal en Azure, destacando los componentes clave utilizados para almacenamiento, auditoría, despliegue, orquestación, monitoreo y seguridad.

Almacenamiento y Auditoría

- **Azure Blob Storage**: Utilizado para almacenar registros de auditoría y otros datos no estructurados.
- **Base de Datos de Auditoría**: Registra todas las acciones del cliente para cumplir con los requisitos normativos.
- **Base de Datos SQL**: Implementada con Microsoft SQL Server 2019 para la persistencia de datos transaccionales y relacionales.

Despliegue y Orquestación

- **Elastic Kubernetes Service (EKS)**: Administra y orquesta los contenedores Docker para asegurar la escalabilidad y alta disponibilidad del sistema.

Monitoreo y Seguridad

- **Azure Dashboard**: Proporciona una visión centralizada del estado del sistema, permitiendo la supervisión en tiempo real.
- **Azure Monitor**: Supervisa continuamente la salud de los servicios y aplicaciones, ofreciendo métricas y alertas.
- **Azure Defender**: Protege los recursos en la nube contra amenazas y vulnerabilidades.
- **Azure Blob Storage**: Almacena de manera segura grandes volúmenes de datos no estructurados.
- **Container Registry**: Gestiona y almacena las imágenes de los contenedores Docker.
- **Elastic Kubernetes Service (EKS)**: Gestiona la orquestación de contenedores Docker para escalabilidad y alta disponibilidad.
- **Elasticsearch Managed Service**: Almacena y analiza registros, facilitando el monitoreo de la salud y rendimiento del sistema.
- **Azure Application Insights**: Proporciona monitoreo de aplicaciones en tiempo real, recolectando datos de rendimiento y uso para identificar problemas, diagnosticar errores y obtener insights sobre el comportamiento de los usuarios. Esto ayuda a mejorar la fiabilidad y el rendimiento de las aplicaciones web y móviles.
- **Azure Log Analytics**: Ofrece análisis y visualización de registros recopilados de diversos recursos y aplicaciones. Permite consultar, analizar y correlacionar datos de registros, facilitando la supervisión, solución de problemas y optimización del entorno de TI en la nube.

Backup y Recuperación

- **Azure Backup**: Proporciona soluciones de backup y recuperación para proteger los datos contra pérdidas y desastres.

Balanceo de Carga

- **Load Balancer**: Distribuye el tráfico entrante entre los servidores para asegurar una carga equilibrada y mejorar la disponibilidad y resiliencia del sistema.

Red y Seguridad

- **Azure Virtual Network (VNet)**: Permite crear una red privada en la nube para asegurar la comunicación segura entre los recursos internos del sistema de banca por internet.
- **Network Security Groups (NSG)**: Controla el tráfico de red hacia y desde los recursos de Azure, proporcionando reglas de seguridad para permitir o denegar el tráfico basado en la configuración.
- **Azure Firewall**: Protege la red contra amenazas externas y proporciona control centralizado sobre las políticas de seguridad.
- **Azure Active Directory (AAD)**: Gestiona las identidades y los accesos, permitiendo la autenticación y autorización de los usuarios que acceden a las aplicaciones.
- **Azure Key Vault**: Almacena y gestiona secretos, claves de cifrado y certificados, asegurando que estos elementos sensibles estén protegidos y accesibles de forma segura.
- **Azure DDoS Protection**: Protege las aplicaciones y recursos en la nube contra ataques de denegación de servicio distribuido (DDoS), asegurando la disponibilidad del sistema.

Este diseño de infraestructura en Azure asegura que el sistema de banca por Internet sea robusto, seguro, y capaz de manejar una alta carga de usuarios mientras mantiene un rendimiento óptimo y una alta disponibilidad.

ANEXOS

Diagrama de Contexto

```
@startuml
!include <c4/C4_Context.puml>
'ref http://plantuml.com/stdlib
!include <office/Users/user.puml>
!include <office/Users/mobile_user.puml>
LAYOUT_WITH_LEGEND()

title Diagrama de Contexto para Banco por Internet

Person(customer, "Cliente Banca Personas", "<$user> <$mobile_user>\nUn cliente del Banco, con cuentas en banca personales")
Person(customer_service_rep, "Representante de Atención al Cliente", "Un empleado del banco que asiste a los clientes")

System(banking_system, "Banca Por Internet", "Permite al usuario consultar movimientos y realizar pagos y transferencias interbancarias.")
System_Ext(mail_system, "Sistema de Correo Electrónico", "Sistema interno de correo electrónico.")
System_Ext(sms_system, "Sistema Notificación SMS", "Permite el envío de notificaciones SMS.")
System_Ext(mainframe, "Core Banking System", "Contiene información básica de cliente, movimientos, productos, etc.")
System_Ext(Biometrico, "Sistema de Autenticación Biométrica", "Sistema que autentica mediante reconocimiento facial.")
System_Ext(Complementarios, "Sistema Datos Adicionales", "Complementa la información del cliente cuando los datos se requieren en detalle.")
System_Ext(trf_ext, "Sistema de Transferencias", "Sistema para manejo de transferencias propias, a terceros y transferencias interbancarias.")
System_Ext(iam, "Sistema de Autenticación IAM", "Sistema de administración de identidades que maneja OAuth 2.0 y la lógica de autenticación.")
System_Ext(api_management, "Sistema de Gestión de API", "Sistema para la documentación y gestión de API.")

Rel(customer, banking_system, "Usa para consultar movimientos y realizar transacciones", "HTTPS")
Rel_Back(customer, mail_system, "Envía solicitudes de correo a")
Rel_Back(customer, sms_system, "Envía solicitudes de SMS a")
Rel(banking_system, mail_system, "Envía correos", "SMTP")
Rel(banking_system, sms_system, "Envía SMS", "HTTPS")
Rel(iam, mail_system, "Envía correos OTP", "SMTP")
Rel(iam, sms_system, "Envía SMS OTP", "HTTPS")
Rel(banking_system, mainframe, "Usa para obtener información básica de cliente", "HTTPS")
Rel(banking_system, Biometrico, "Usa para autenticación biométrica", "HTTPS")
Rel(banking_system, trf_ext, "Usa para gestionar transferencias", "HTTPS")
Rel(banking_system, iam, "Usa para autenticación", "HTTPS")
Rel(banking_system, Complementarios, "Usa para obtener datos adicionales del cliente", "HTTPS")
Rel(customer_service_rep, banking_system, "Usa para asistir a los clientes", "HTTPS")
Rel(banking_system, api_management, "Usa para gestionar y documentar APIs", "HTTPS")

@enduml
```

Diagrama de Contenedor

```
@startuml
!include <c4/C4_Container.puml>
!include <office/Users/user.puml>
!include <office/Users/mobile_user.puml>
LAYOUT_WITH_LEGEND()

title Diagrama de Contenedor Banca por Internet

Person(customer, "Cliente", "<$user> <$mobile_user>\nUn cliente del banco, con cuentas en banca personas")

System_Boundary(c1, "Banca Por Internet") {
 Container(web_app, "Web Application", "Java, Spring MVC", "Ofrece el contenido estático y el SPA de banca por Internet.")
 Container(spa, "Single-Page App", "JavaScript, Angular", "Provee la funcionalidad de Internet Banking al cliente mediante el uso de un navegador web.")
 Container(mobile_app, "Mobile App", "(C#, Xamarin), (Dart, Flutter)", "Provee la funcionalidad de Internet Banking a los clientes mediante un dispositivo móvil.")
 ContainerDb(database, "Database", "Microsoft SQLServer", "Recupera información de registro de usuario, credenciales de autenticación hasheadas, registros de acceso, etc.")
 Container(backend_api, "API Application", "Java, Docker Container", "Provee la funcionalidad de Internet banking a través de una API.")
 Container(facial_api, "API Reconocimiento Facial", "Java, Docker Container", "Provee la funcionalidad de reconocimiento facial.")
 Container(autenticacion_api, "API Autenticación", "Java, Docker Container", "Provee la funcionalidad de autenticación para el Internet banking a través de una API.")
 Container(ApiAccountRepository, "API Account Repository", "Java, Docker Container", "Registra todas las acciones del cliente y cuenta con un mecanismo de persistencia de información para clientes frecuentes.")
Container(Onboarding, "API de Onboarding", "Java, Docker Container", "Permite la vinculación de nuevos clientes en Banca Móvil.")
}

System_Ext(email_system, "Sistema de Correo Electrónico", "Sistema interno de correo electrónico.")
System_Ext(sms_system, "Sistema Notificación SMS", "Permite el envío de notificaciones SMS.")
System_Ext(banking_system, "Core Banking System", "Contiene información básica de cliente, movimientos, productos, etc.")
System_Ext(Biometrico, "Sistema de Autenticación Biométrica", "Sistema que autentica mediante reconocimiento facial.")
System_Ext(Complementarios, "Sistema Datos Adicionales", "Complementa la información del cliente cuando los datos se requieren en detalle.")
System_Ext(trf_ext, "Sistema de Transferencias", "Sistema para manejo de transferencias propias, a terceros y transferencias interbancarias.")
System_Ext(iam, "Sistema de Autenticación IAM", "Sistema de administración de identidades que maneja OAuth 2.0 y la lógica de autenticación.")

Rel(customer, web_app, "Usa", "HTTPS")
Rel(customer, spa, "Usa", "HTTPS")
```

```
Rel(customer, mobile_app, "Usa")

Rel_Neighbor(web_app, spa, "Entrega")
Rel(spa, backend_api, "Usa", "async, JSON/HTTPS")
Rel(mobile_app, backend_api, "Usa", "async, JSON/HTTPS")
Rel(backend_api, ApiAccountRepository, "Usa", "async, JSON/HTTPS")
Rel_Back(database, backend_api, "Lee desde y escribe a", "sync, JDBC")
Rel_Back(database, ApiAccountRepository, "Lee desde y escribe a", "sync, JDBC")
Rel(spa, autenticacion_api, "Usa", "async, JSON/HTTPS REST")
Rel(mobile_app, autenticacion_api, "Usa", "async, JSON/HTTPS REST")
Rel(spa, ApiAccountRepository, "Usa", "async, JSON/HTTPS REST")
Rel(mobile_app, ApiAccountRepository, "Usa", "async, JSON/HTTPS REST")
Rel(Onboarding, facial_api, "Usa", "async, JSON/HTTPS REST")
Rel(spa, facial_api, "Usa", "async, JSON/HTTPS REST")
Rel(mobile_app, facial_api, "Usa", "async, JSON/HTTPS REST")

Rel_Back(customer, email_system, "Envía correo a")
Rel_Back(email_system, backend_api, "Envía correo", "sync, SMTP")
Rel_Neighbor(backend_api, banking_system, "Usa", "sync/async, JSON/HTTPS")
Rel(backend_api, Complementarios, "Usa", "sync/async, JSON/HTTPS REST")
Rel(backend_api, trf_ext, "Usa", "sync/async, JSON/HTTPS REST")
Rel_Back(customer, sms_system, "Envía SMS a")
Rel_Back(sms_system, backend_api, "Envía SMS usando", "sync, JSON/HTTPS")
Rel(mobile_app, Onboarding, "Usa", "sync/async, JSON/HTTPS REST")
Rel(facial_api, Biometrico, "Usa", "sync/async, JSON/HTTPS REST")
Rel(autenticacion_api, iam, "Usa", "sync/async, JSON/HTTPS REST")
Rel(iam, email_system, "Envía correo OTP", "sync, SMTP")
Rel(iam, sms_system, "Envía SMS OTP usando", "sync, JSON/HTTPS")

@enduml
```

Diagrama de Componentes

```
@startuml
!include <c4/C4_Component.puml>
LAYOUT_WITH_LEGEND()

title Diagrama de Componentes de Banca por Internet - API Application

Container(spa, "Single Page Application", "JavaScript, Angular", "Provee toda la funcionalidad de Banca por Internet mediante un navegador web.")
Container(ma, "Mobile App", "Xamarin o Flutter", "Provee una funcionalidad limitada de la banca por Internet a través de dispositivos móviles.")
ContainerDb(db, "Database", "SQLServer", "Recupera información de registro de usuario, credenciales de autenticación hasheadas, registros de acceso, etc.")
Container(ApiAccountRepository, "API Account Repository", "Java, Docker Container", "Registra todas las acciones del cliente y persiste información para clientes frecuentes.")

System_Ext(mbs, "Core Banking System", "Contiene información básica de cliente, movimientos, productos, etc.")
```

```
System_Ext(trf, "Sistema de Transferencias", "Sistema para manejo de transferencias propias, a terceros y transferencias interbancarias.")
System_Ext(email_system, "Sistema de Correo Electrónico", "Sistema interno de correo electrónico.")
System_Ext(sms_system, "Sistema Notificación SMS", "Permite el envío de notificaciones SMS.")
System_Ext(Complementarios, "Sistema Datos Adicionales", "Complementa la información del cliente cuando los datos se requieren en detalle.")

Container_Boundary(api, "API Application") {
  Container_Boundary(Gateway, "API Management") {
    Component(trf_ag, "Transferencias API", "API Gateway", "Permite realizar transferencias entre cuentas propias, de terceros o interbancarias.")
    Component(DatosB_ag, "Consulta Datos Básicos API", "API Gateway", "Permite realizar la consulta de datos básicos.")
    Component(DatosC_ag, "Consulta Datos Complementarios API", "API Gateway", "Permite realizar la consulta de datos complementarios como contactos, transacciones, correo, celular.")
    Component(accounts_ag, "Histórico de Movimientos API", "API Gateway", "Provee la consulta de movimientos del cliente.")
  }

  Container_Boundary(EBS, "SOA/Microservicios") {
    Component(mbsfacade, "Consulta Históricos Core Wrapper", "Spring Bean", "Wrapper del sistema core banking.")
    Component(mbsDebitoCredito, "DébitoCrédito Core Wrapper", "Spring Bean", "Wrapper del sistema core banking.")
    Component(mbsDBasicos, "Consulta Datos Básicos Core Wrapper", "Spring Bean", "Wrapper del sistema core banking.")
    Component(trf_c, "Transferencias Controller", "MVC Rest Controller", "Permite realizar transferencias entre cuentas propias, de terceros o interbancarias.")
    Component(DatosB_c, "Consulta Datos Básicos Controller", "MVC Rest Controller", "Permite realizar la consulta de datos básicos.")
    Component(DatosC_c, "Consulta Datos Complementarios Controller", "MVC Rest Controller", "Permite realizar la consulta de datos complementarios como contactos, transacciones, correo, celular.")
    Component(accounts_c, "Histórico de Movimientos Controller", "MVC Rest Controller", "Provee la consulta de movimientos del cliente.")
    Component(accounts, "Histórico de Movimientos", "Spring Bean", "Provee la consulta de movimientos del cliente.")
    Component(trfbanco, "Transf Banco", "Spring Bean", "Provee la funcionalidad de transferencias entre cuentas propias y del mismo banco.")
    Component(DatosC, "Consulta Datos Complementarios", "Spring Bean", "Permite realizar la consulta de datos complementarios como contactos, transacciones, correo, celular.")
    Component(trfInter, "Transferencias Interbancarias", "Spring Bean", "Provee la funcionalidad de transferencias interbancarias, pago directo o Banco Central.")
    Component(email_c, "Envio Correo Controller", "MVC Rest Controller", "Envía correo electrónico.")
    Component(sms_c, "Envio SMS Controller", "MVC Rest Controller", "Envía mensajes SMS.")
    Component(email, "Envio Correo", "Spring Bean", "Wrapper del sistema core banking.")
    Component(sms, "Envio SMS", "Spring Bean", "Provee la funcionalidad de envío de SMS.")
  }

  Rel(accounts_c, accounts, "Usa", "JSON/HTTPS")
  Rel(accounts, mbsfacade, "Usa", "JSON/HTTPS")
  Rel(accounts, ApiAccountRepository, "Usa", "JSON/HTTPS")
  Rel(trf_c, trfbanco, "Usa", "JSON/HTTPS")
  Rel(trf_c, trfInter, "Usa", "JSON/HTTPS")
```

```
 Rel(DatosC, Complementarios, "Usa", "JSON/HTTPS")
 Rel(trfbanco, mbsDebitoCredito, "Usa", "JSON/HTTPS")
 Rel(trfbanco, email_c, "Usa", "JSON/HTTPS")
 Rel(trfbanco, sms_c, "Usa", "JSON/HTTPS")
 Rel(trfbanco, ApiAccountRepository, "Usa", "JSON/HTTPS")
 Rel(trfInter, email_c, "Usa", "JSON/HTTPS")
 Rel(trfInter, sms_c, "Usa", "JSON/HTTPS")
 Rel(trfInter, ApiAccountRepository, "Usa", "JSON/HTTPS")
 Rel(trfInter, trf, "Usa", "JSON/HTTPS")
 Rel(mbsfacade, mbs, "Usa", "JSON/HTTPS")
 Rel(trfInter, mbsDebitoCredito, "Usa", "JSON/HTTPS")
 Rel(DatosB_c, mbsDBasicos, "Usa", "JSON/HTTPS")
 Rel(email_c, email, "Usa", "JSON/HTTPS")
 Rel(sms_c, sms, "Usa", "JSON/HTTPS")
 Rel(DatosC_c, DatosC, "Usa", "JSON/HTTPS")
}

Rel_Neighbor(spa, accounts_ag, "Usa", "JSON/HTTPS")
Rel(mbsDBasicos, mbs, "Usa", "JSON/HTTPS")
Rel(mbsDebitoCredito, mbs, "Usa", "JSON/HTTPS")
Rel(ma, trf_ag, "Usa", "JSON/HTTPS")
Rel(ma, accounts_ag, "Usa", "JSON/HTTPS")
Rel(spa, DatosB_ag, "Usa", "JSON/HTTPS")
Rel(spa, trf_ag, "Usa", "JSON/HTTPS")
Rel(ma, DatosB_ag, "Usa", "JSON/HTTPS")
Rel(ma, DatosC_ag, "Usa", "JSON/HTTPS")
Rel(DatosC_ag, DatosC_c, "Usa", "JSON/HTTPS")
Rel(DatosB_ag, DatosB_c, "Usa", "JSON/HTTPS")
Rel(trf_ag, trf_c, "Usa", "JSON/HTTPS")
Rel(accounts_ag, accounts_c, "Usa", "JSON/HTTPS")
Rel(email, email_system, "Usa", "JSON/HTTPS")
Rel(sms, sms_system, "Usa", "JSON/HTTPS")
Rel(ApiAccountRepository, db, "Lee y escribe en", "JDBC")

@enduml
```