Arquitectura del Sistema de Banca por Internet – BP

Autor: Franklin Garzón

Fecha: 23/03/2025

Versión: 1.00

Control de Cambios

Versión	Fecha	Aprobado Por	Cambio Realizado

Contenido

Introducción 2
Requerimientos Funcionales 2
Seguridades 2
Autenticación y Autorización de usuarios 2
Onboarding 2
Normativas y Reglamentos 2
Estándares de seguridad 3
Justificación del Uso de React.js y React Native para los Frontends 3
Arquitectura 4
Diagrama de Contexto 4
Diagrama de contenedores 5
Diagrama de Componentes 10
Autenticación – Onboarding 10

Introducción

Este documento describe la arquitectura de solución para el sistema de banca en línea para BP. El sistema permite a los usuarios acceder a su historial de movimientos, realizar transferencias y pagos, y recibir notificaciones. La arquitectura se basa en servicios escalables, seguros y conformes con la normativa.

Requerimientos Funcionales

- Acceso al histórico de movimientos bancarios.
- Transferencias y pagos interbancarios.
- Notificaciones de movimientos a clientes.
- Autenticación mediante OAuth 2.0 con flujo Authorization Code Flow.
- Onboarding con reconocimiento facial.
- Persistencia de datos de clientes frecuentes.

Seguridades

Autenticación y Autorización de usuarios

En base al contexto de la solución diseñada se establece que el flujo de autorización a seguir es PKCE (Proof Key for Code Exchange), es un mecanismo de seguridad utilizado en el flujo de autorización de OAuth 2.0, especialmente diseñado para aplicaciones móviles y SPA. Su objetivo principal es mitigar ataques de interceptación de códigos de autorización

Onboarding

El onboarding debe garantizar que el nuevo cliente sea una persona real, es por ello que el reconocimiento facial es esencial, se optó por usar: Azure AD B2C, orientado a la gestión de identidades y la autenticación de usuarios en aplicaciones externas.

Normativas y Reglamentos

- **Reglamento a la Ley de Instituciones del Sistema Financiero**.- Establece disposiciones generales sobre la operación de las entidades financieras, incluyendo el manejo de sistemas informáticos.
- **Normativa de Seguridad de la Información de la Superintendencia de Bancos.-** Incluye lineamientos sobre la protección de datos, gestión de riesgos y la implementación de controles de seguridad.
- **Ley de Protección de Datos Personales**.- Regula el manejo y tratamiento de la información personal de los clientes, garantizando su privacidad y derechos.

- **Normas de Conformidad con PCI-DSS**.- Si manejan información de tarjetas de crédito, deben cumplir con los estándares de seguridad de datos para la industria de tarjetas de pago. Incluyendo el acceso inhouse de los datos y acceso a los equipos físicos.
- **Políticas de prevención de lavado de activos y financiamiento del terrorismo.- (LA/FT)**: Establecen protocolos para el monitoreo y reporte de actividades sospechosas.
- **Normas sobre continuidad del negocio y recuperación ante desastres**.- Aseguran que los bancos tengan planes efectivos para mantener operaciones en caso de incidentes graves.
- **Regulaciones sobre ciberseguridad.-** Incluyen medidas para proteger los sistemas de información de ataques y brechas de seguridad.

Estándares de seguridad

- **ISO/IEC 27001**.- Establece un marco para la gestión de la seguridad de la información.
- **NIST Cybersecurity Framework.-** Proporciona directrices para identificar, proteger, detectar, responder y recuperarse de ciber-incidentes.
- **PCI-DSS**.- Aplica para la protección de datos de tarjetas de pago, asegurando la seguridad en transacciones.
- **OWASP Top Ten.-** Recomendaciones sobre las principales vulnerabilidades de seguridad en aplicaciones web.
- **Normativa de Seguridad de la Información de la Superintendencia de Bancos**.- Directrices específicas para el sector financiero.
-

Justificación del Uso de React.js y React Native para los Frontends

La elección de **React.js** para el frontend web y **React Native** para el frontend móvil se basa en su flexibilidad, eficiencia y capacidad de reutilización de código, lo que permite una arquitectura escalable y mantenible.

Unificación del Stack Tecnológico

El uso de React.js y React Native permite compartir lógica, componentes y estilos entre las plataformas web y móvil, reduciendo la duplicación de código y mejorando la mantenibilidad.

Componentización y Reutilización de Código

- **React.js y React Native** permiten desarrollar aplicaciones mediante una arquitectura basada en **componentes reutilizables**, lo que acelera el desarrollo y mejora la cohesión del código.
- Librerías como **React Native Web** permiten compartir componentes entre ambas plataformas.

Eficiencia y Rendimiento

- **React.js** utiliza **Virtual DOM**, optimizando la renderización y mejorando la experiencia del usuario en aplicaciones web.

- **React Native** se comunica con los componentes nativos mediante un bridge optimizado, logrando un rendimiento cercano al de las aplicaciones nativas.

Ecosistema y Comunidad

- React cuenta con un ecosistema maduro y ampliamente soportado, con una gran variedad de bibliotecas y herramientas como **Redux, React Query, Zustand** para gestión de estado y **React Navigation** para navegación en móviles.
- Amplio soporte por parte de **Meta (Facebook)** y una comunidad activa que garantiza la evolución y mantenimiento de la tecnología.

Experiencia de Usuario Consistente

- Permite construir interfaces modernas y dinámicas con una UX/UI consistente entre plataformas.
- Uso de **Design Systems** unificados con bibliotecas como **Material-UI, Ant Design, React Native Paper**.

Integración con Backend y APIs

- Soporte nativo para consumo de APIs REST y GraphQL mediante herramientas como **Axios, Apollo Client y SWR**.
- Facilidad para manejar autenticación e integración con servicios en la nube como **Azure AD B2C o Firebase Auth**.

Desarrollo Ágil y Entregas Continuas

- Compatibilidad con enfoques **CI/CD** utilizando **GitHub Actions, Azure DevOps o Bitrise**.
- Hot Reloading y Fast Refresh permiten iteraciones rápidas en el desarrollo.

Arquitectura

Diagrama de Contexto

- **Cuentahabientes**.- Ingresan al sistema desde una aplicación SPA y móvil.
- **Servicio para cuentahabientes**.- Registra las transacciones, pagos e historial.
- **Azure AD B2C**.- Provee el servicio para el ingreso con reconocimiento facial.
- **Detalles del cuentahabiente**.- Sistema independiente que provee de la información detallada del cliente.
- **Transferencias Interbancaria**s.- Servicios externos que permiten el registro de transferencias interbancarias.

- **Notificación a Usuarios**.- Servicio externo para el envío de notificaciones y mensajes a los usuarios del sistema.

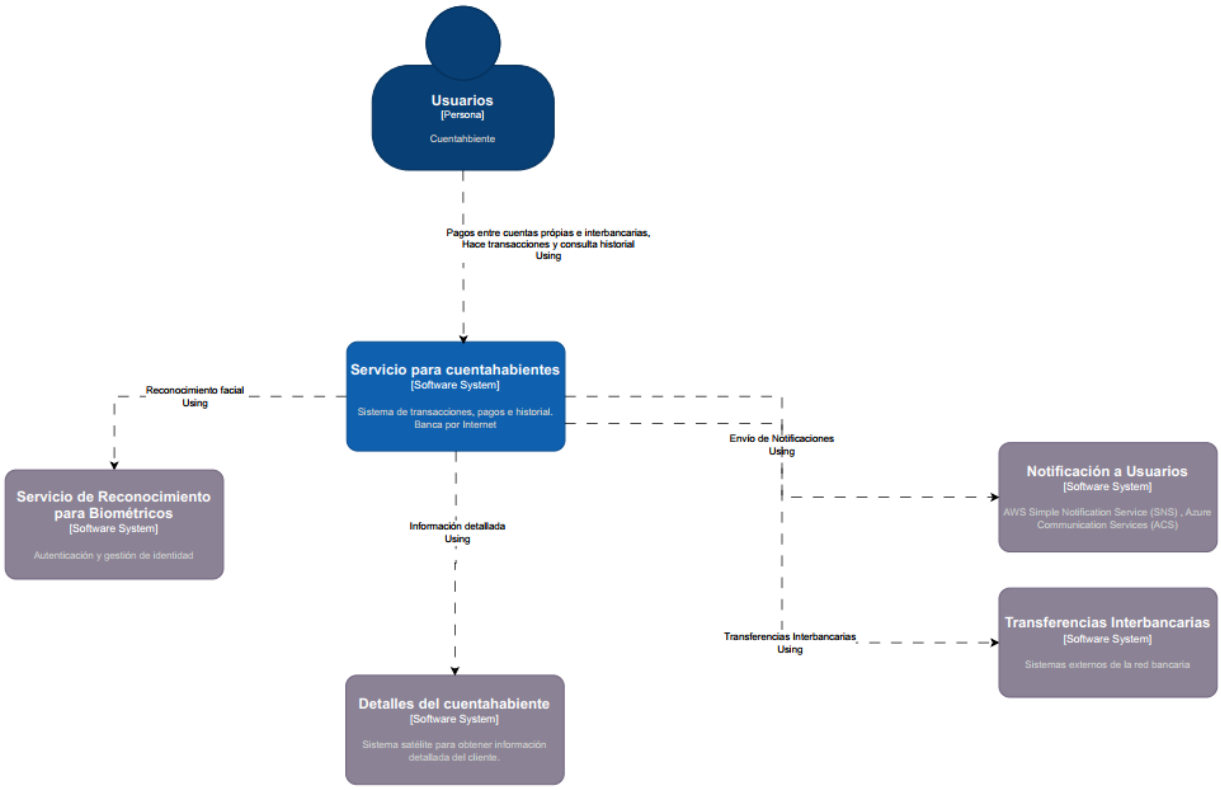

Diagrama de contenedores

Arquitectura Desacoplada - Orientada a Eventos en Azure

Usar una arquitectura orientada a eventos con **Azure Functions**, **Azure Kubernetes Service (AKS)** y **Azure Service Bus** ofrece varios beneficios clave:

Escalabilidad

- Azure Functions escala automáticamente según la carga de trabajo, permitiendo ejecutar funciones en respuesta a eventos sin intervención manual. Las funciones se facturan según el tiempo de ejecución y los recursos consumidos.

- Azure Kubernetes Service (AKS) permite escalar contenedores de aplicaciones en Kubernetes según la demanda. AKS facilita la implementación y administración de contenedores a gran escala.
- La comunicación entre microservicios via gRPC es la mejor opción para la transmisión de información con grandes volúmenes.

Desacoplamiento

- Los servicios pueden comunicarse a través de Azure Service Bus, lo que reduce las dependencias directas entre componentes y facilita el mantenimiento y la evolución independiente de cada servicio.

Flexibilidad de Implementación

- Se puede combinar Azure Functions (serverless) y AKS (contenedores) para aprovechar lo mejor de ambos mundos.

Resiliencia y Manejo de Errores

- Azure Service Bus actúa como un buffer, almacenando mensajes hasta que puedan ser procesados, lo que evita la pérdida de datos en caso de fallos. El uso de Dead-letter queues en Service Bus asegura que los mensajes no procesados se almacenen para su posterior análisis.
- Azure Event Grid también permite gestionar eventos de forma resiliente, entregando los mensajes a los suscriptores incluso si algunos servicios no están disponibles temporalmente.

Costo-efectividad

- Azure Functions solo factura por las ejecuciones de funciones y los recursos consumidos, lo que optimiza los costos operativos.
- AKS cobra por los recursos que se utilizan, lo que garantiza que solo se pague por lo que se consume, optimizando también los costos operativos.

Desarrollo Ágil

- Facilita el desarrollo de aplicaciones basadas en microservicios, permitiendo que los equipos trabajen de forma independiente y en paralelo. Esto mejora la velocidad de desarrollo y la flexibilidad en el manejo de nuevas características o cambios.
- Implementación de Azure DevOps para el CI/CD con Test automatizado y Sonar-Q, esta solución es robusta y asegura la calidad en un entorno de desarrollo ágil.

Monitoreo y Trazabilidad

- Integración con Azure Monitor y Azure Application Insights para monitorear y depurar aplicaciones de manera efectiva.
    - Azure Monitor permite recopilar métricas y registros de la infraestructura.

- Application Insights proporciona trazabilidad a nivel de aplicación, permitiendo a los equipos identificar problemas en el flujo de trabajo.

Alta Disponibilidad

La alta disponibilidad se logra mediante:

Azure Functions

- **Sin servidor**: **Azure Functions** se ejecuta en múltiples **zonas de disponibilidad (AZ)**, lo que garantiza que las funciones estén siempre disponibles, incluso si una zona de disponibilidad falla.
- **Escalabilidad automática**: **Azure Functions** se adapta automáticamente a la carga, asegurando que haya capacidad para manejar picos de tráfico sin interrupciones, escalando según sea necesario en función de la demanda.

Azure Kubernetes Service (AKS)

- **Orquestación de contenedores**: AKS gestiona automáticamente el despliegue y escalado de aplicaciones en contenedores, distribuyéndolos en múltiples zonas de disponibilidad (AZ) para asegurar alta disponibilidad.
- **Autoescalado y gestión de fallos**: AKS utiliza Kubernetes para escalar y reiniciar pods automáticamente en caso de fallos, manteniendo la disponibilidad de las aplicaciones. Además, Azure Monitor y Azure Log Analytics se integran para proporcionar visibilidad y monitoreo en tiempo real.

Ambos servicios, al estar diseñados para la **resiliencia** y la **escalabilidad**, permiten construir aplicaciones que pueden recuperarse rápidamente de fallos y manejar variaciones en la carga de trabajo, asegurando un servicio ininterrumpido.

Tolerancia a Fallos

Las estrategias de **Tolerancia a Fallos** como **Retry** y **Circuit Breaker** son componentes esenciales para garantizar la resiliencia y la estabilidad de las aplicaciones en entornos distribuidos. Ambas estrategias contribuyen significativamente a la gestión de fallos y a la minimización del impacto de errores temporales o recurrentes.

Retry (Reintentos)

- La estrategia de **Retry** se implementa para reintentar operaciones que han fallado debido a errores temporales o transitorios.
- Se configuran **retrasos** entre los intentos y un **número máximo de reintentos** para evitar sobrecargar el sistema y proporcionar tiempo para que los problemas transitorios se resuelvan de manera autónoma.
- Esta estrategia es útil para operaciones de red, acceso a servicios externos o interacciones con bases de datos que pueden experimentar fallos temporales.

Circuit Breaker (Interruptor de Circuito)

- El **Circuit Breaker** actúa como un mecanismo de protección para evitar que una aplicación intente realizar operaciones que probablemente fallarán, lo que podría afectar el rendimiento del sistema.
- Después de un número determinado de fallos consecutivos en una operación, el **circuito se "abre"**, lo que bloquea temporalmente las solicitudes y previene que el sistema realice más intentos fallidos.
- Tras un período de espera, el **circuito se "cierra"** para permitir que el sistema reintente la operación, evaluando si las condiciones de fallo se han resuelto.

Ambas estrategias permiten que el sistema se recupere de manera eficiente ante fallos, mejorando la experiencia del usuario y manteniendo la estabilidad operativa en arquitecturas distribuidas.

Monitoreo

El monitoreo de aplicaciones en Azure con Azure Monitor ofrece varias ventajas clave:

Visibilidad completa

- Azure Monitor proporciona métricas y registros en tiempo real, permitiendo a los usuarios observar el rendimiento de sus aplicaciones y recursos a lo largo del tiempo.

Alarmas y notificaciones

- Azure Monitor permite configurar alertas para detectar anomalías en el rendimiento o en la infraestructura, y enviar notificaciones automáticas a los administradores para facilitar una respuesta proactiva ante cualquier problema detectado.

Integración con otros servicios de Azure

- Azure Monitor se integra de manera fluida con otros servicios de Azure, como Azure Application Insights, Azure Log Analytics y Azure Security Center, ofreciendo un monitoreo centralizado y una gestión eficiente de los recursos y aplicaciones distribuidas en la nube.

Análisis de logs

- Azure Log Analytics proporciona potentes herramientas para analizar logs y diagnosticar problemas, permitiendo la identificación de fallos y la optimización del rendimiento de las aplicaciones.

Dashboards personalizados

- Los usuarios pueden crear dashboards personalizados en Azure Monitor para visualizar métricas relevantes de sus aplicaciones y recursos, lo que facilita la toma de decisiones informadas y la monitorización en tiempo real.

Escalabilidad

- Azure Monitor está diseñado para manejar grandes volúmenes de datos, lo que es ideal para aplicaciones en crecimiento que requieren un monitoreo continuo y detallado de su infraestructura.

Optimización de costos

- Azure Monitor ayuda a identificar recursos infrautilizados, lo que permite optimizar el gasto en infraestructura al garantizar que los recursos sean asignados y utilizados de manera eficiente.

Estas ventajas hacen que Azure Monitor sea una herramienta fundamental para la gestión y monitoreo de aplicaciones en la nube, permitiendo una visibilidad completa, una gestión eficiente de recursos y una respuesta rápida ante incidentes.

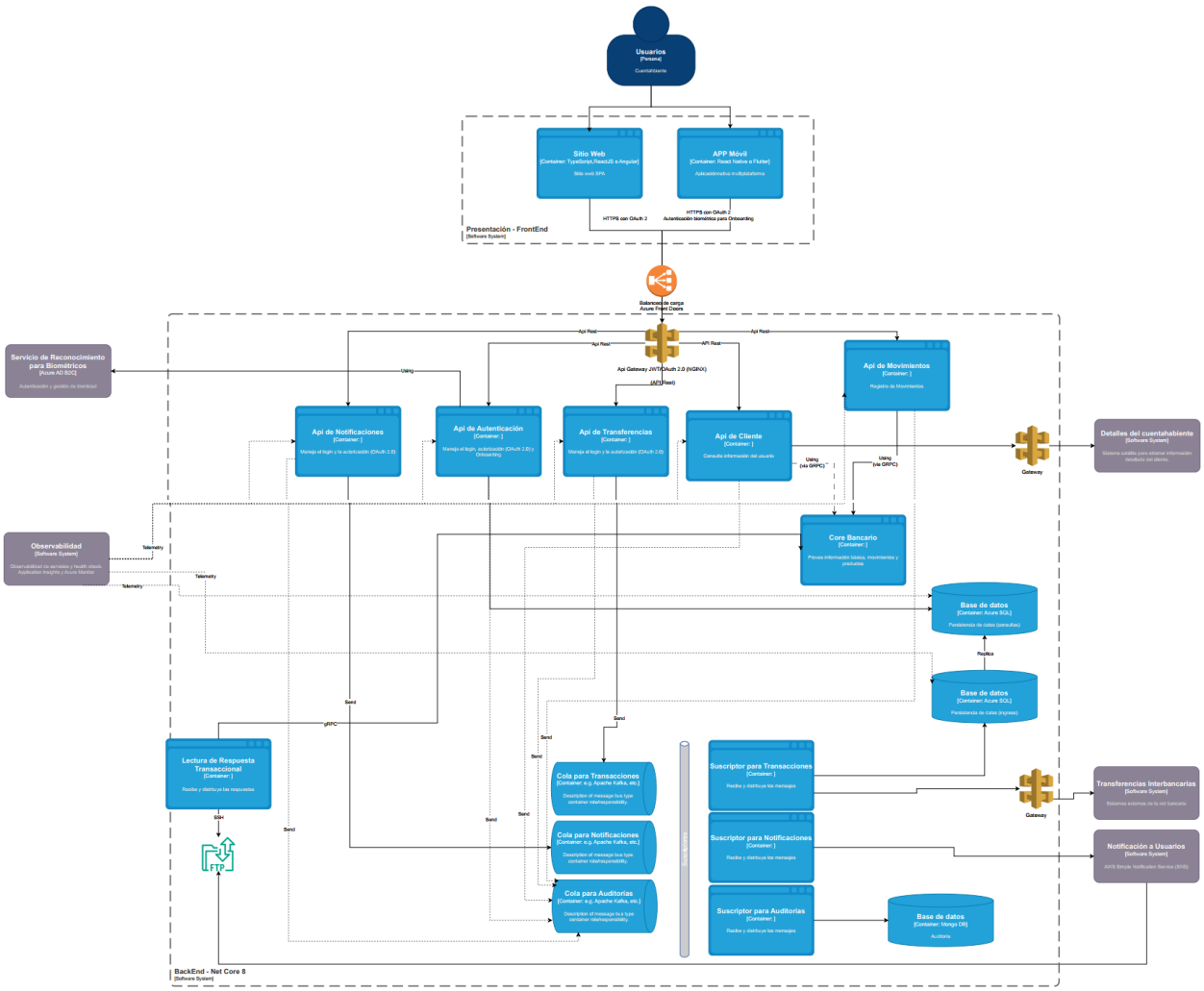

Diagrama de Componentes

Autenticación – Onboarding