26-7-2024

Banca Digital BP

Arquitectura de Solución

Autor:
ANDREE ALEJANDRO GALDOS CORREA

Contenido

Introducción

En proyecto de Banca Digital BP se ha realizado un análisis desde el punto de dolor, según el contexto del caso, que tiene actualmente la organización. Por esta razón, se ha realizado un trabajo desde el punto de vista de un arquitecto de soluciones lo que ha dado como resultado a nivel de diagramas los siguientes entregables.

1. **Diagramación con C4**: Implementé el modelo C4 para representar la arquitectura de la aplicación a diferentes niveles, incluyendo diagramas de contexto, contenedores y componentes. Esto facilitó una comprensión clara de las interacciones entre los usuarios, los sistemas y los componentes de la aplicación.
2. **Diagrama de Arquitectura de Desarrollo**: Desarrollé un diagrama detallado de la arquitectura de desarrollo que describe cómo los diferentes módulos y servicios se integran y colaboran durante el proceso de desarrollo. Este diagrama proporciona una visión clara de la estructura interna del sistema y las dependencias entre los componentes.
3. **Diagrama de Infraestructura**: Diseñé un diagrama de infraestructura en Amazon Web Services (AWS) que ilustra la disposición de los recursos y servicios en la nube. Este diagrama abarca la configuración de servidores, bases de datos, redes y servicios de seguridad, garantizando que la infraestructura sea escalable, segura y eficiente.

Diagramas C4

Diagrama de Contexto

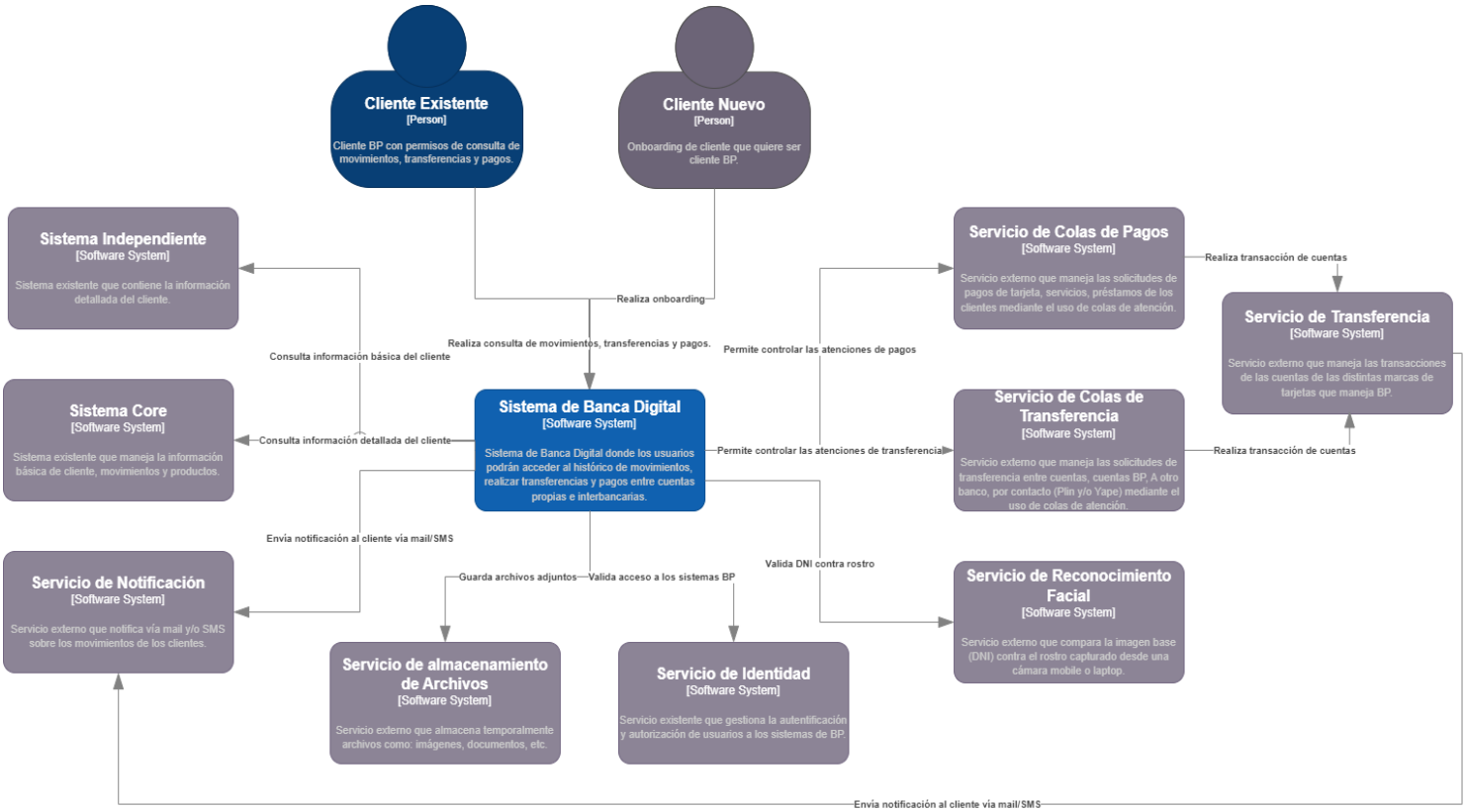


Diagrama de Contenedores

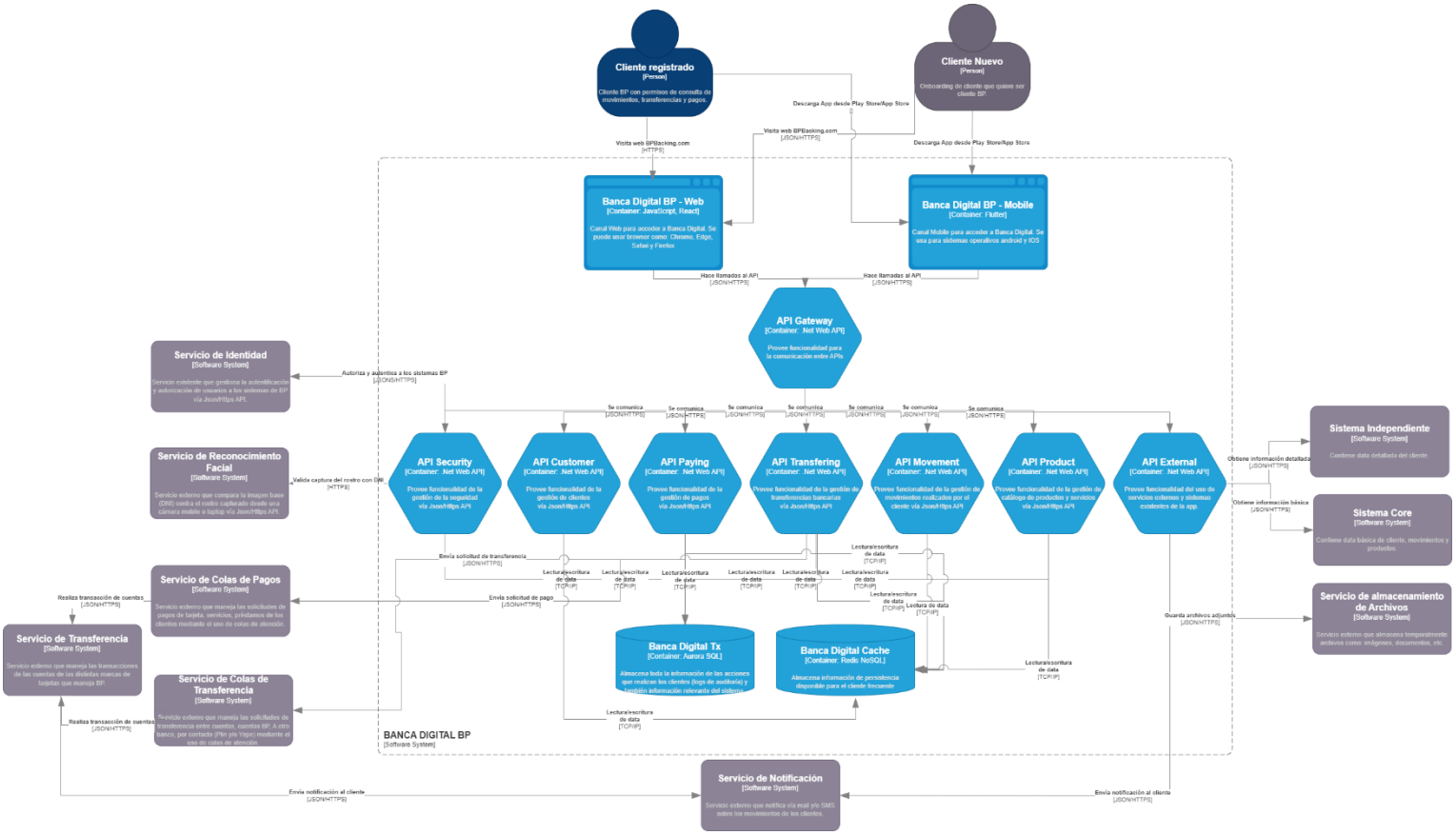


Diagrama de Componentes

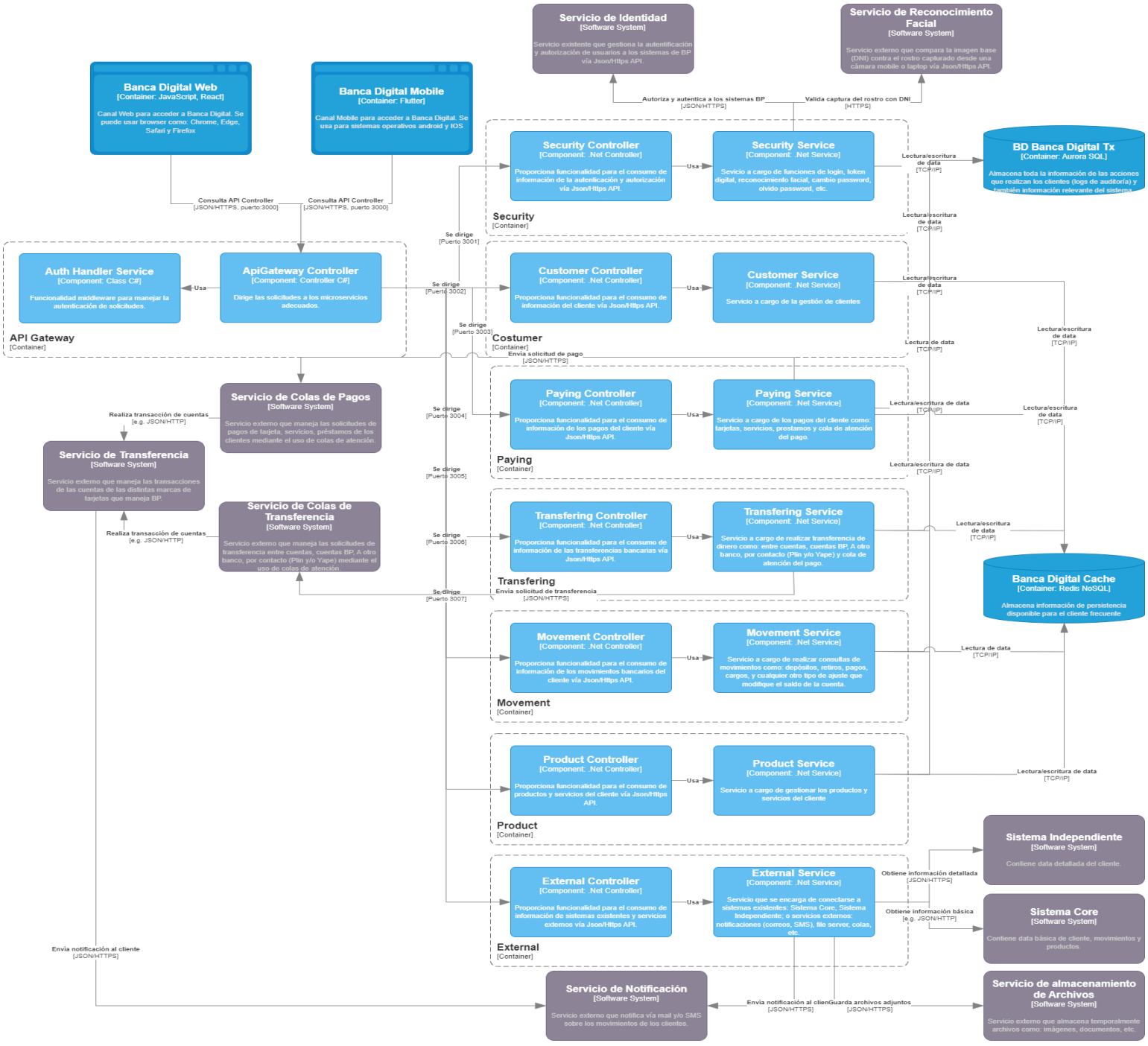
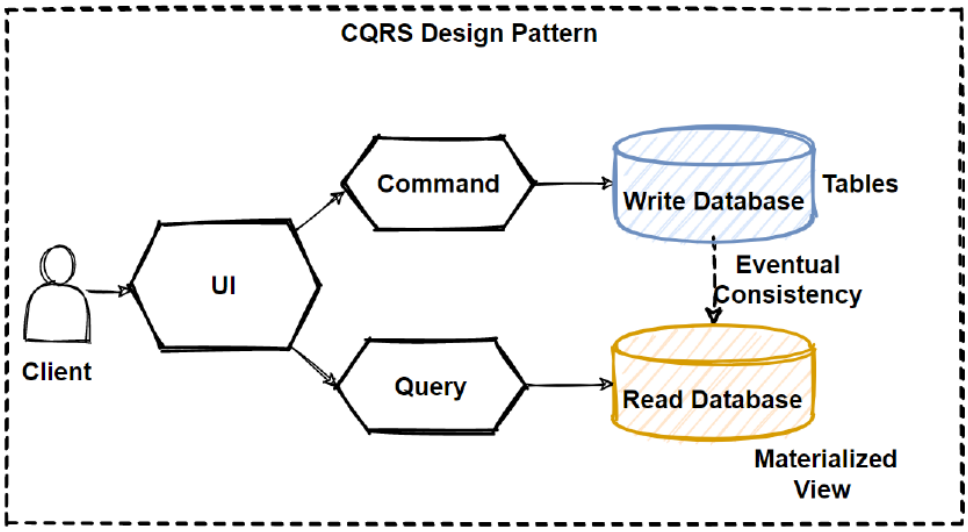


Diagrama de Arquitectura de Desarrollo usando un patrón CQRS

Para el patrón de arquitectura de desarrollo para la solución propuesta usaremos CQRS (Command Query Responsibility Segregation), ya que separa las operaciones de lectura (queries) de las operaciones de escritura (commands) en una aplicación. Esto significa que el modelo de datos usado para leer información y el modelo de datos usado para escribir información son distintos (desacoplados). Esto permitirá optimizar y escalar cada una de manera independiente.

Cómo funciona

- **Commands**: Se encargan de modificar el estado del sistema (escrituras). Estos comandos cambian los datos y suelen ser responsables de la lógica de negocio.
- **Queries**: Se encargan de leer información (lecturas). Estas consultas no modifican el estado del sistema y optimizan el acceso a los datos para las lecturas.



Ventajas de CQRS

1. **Optimización de Lecturas y Escrituras**

   Permite optimizar el modelo de datos para consultas y comandos de manera independiente, mejorando el rendimiento en ambos casos.

2. **Escalabilidad**

   Se puede escalar las operaciones de lectura y escritura por separado, adaptándose mejor a las demandas específicas de la aplicación.

3. **Simplificación de Lógica de Negocio**

   La lógica de lectura y escritura se encuentra desacoplada, lo que puede simplificar el diseño y el mantenimiento del sistema.

4. **Mayor Flexibilidad**

Permite usar diferentes almacenes de datos y tecnologías para consultas y comandos, adaptándose mejor a los requisitos de cada una.

5. **Seguridad y Auditoría Mejoradas**

Facilita la implementación de mecanismos de auditoría y control de acceso específicos para operaciones de lectura y escritura.

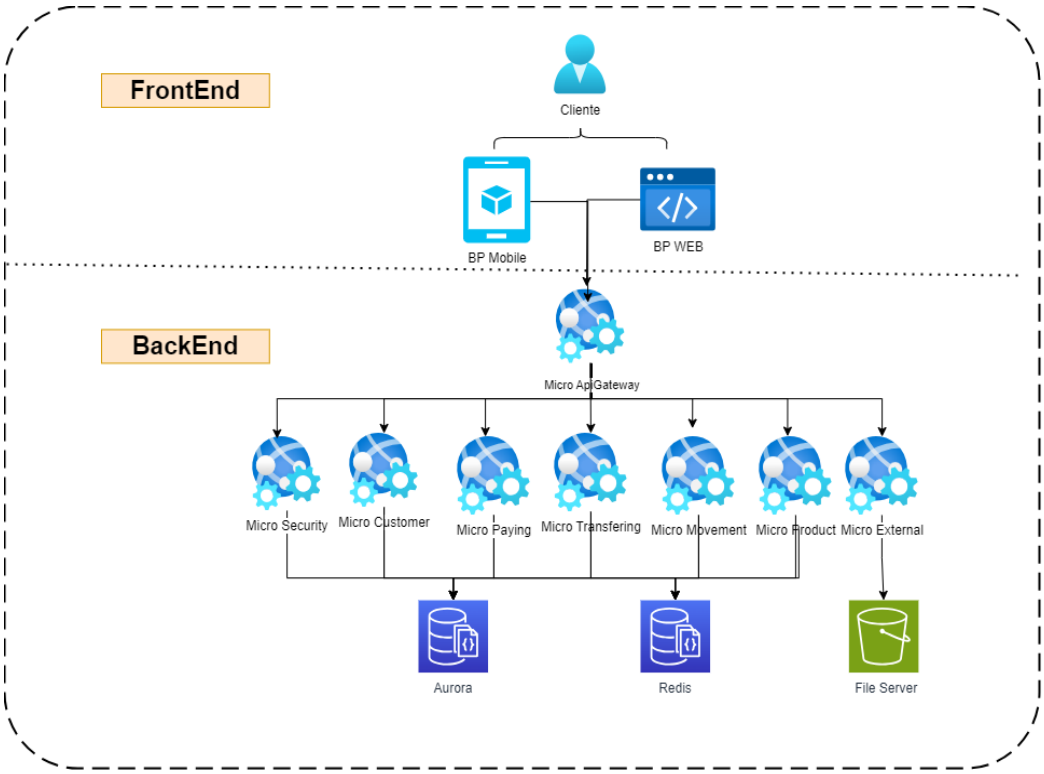
6. **Historial de Cambios y Versionado**

Facilita el rastreo de cambios y la implementación de versionado en las operaciones de escritura, proporcionando un historial más claro.

7. **Desacoplamiento de Componentes**

Reduce el acoplamiento entre el modelo de datos para lectura y el modelo de datos para escritura, lo que permite desarrollar y desplegar estos componentes de manera independiente.

Diagrama de arquitectura de desarrollo



Banca Digital

FrontEnd Web: React

FrontEnd Mobile: Flutter

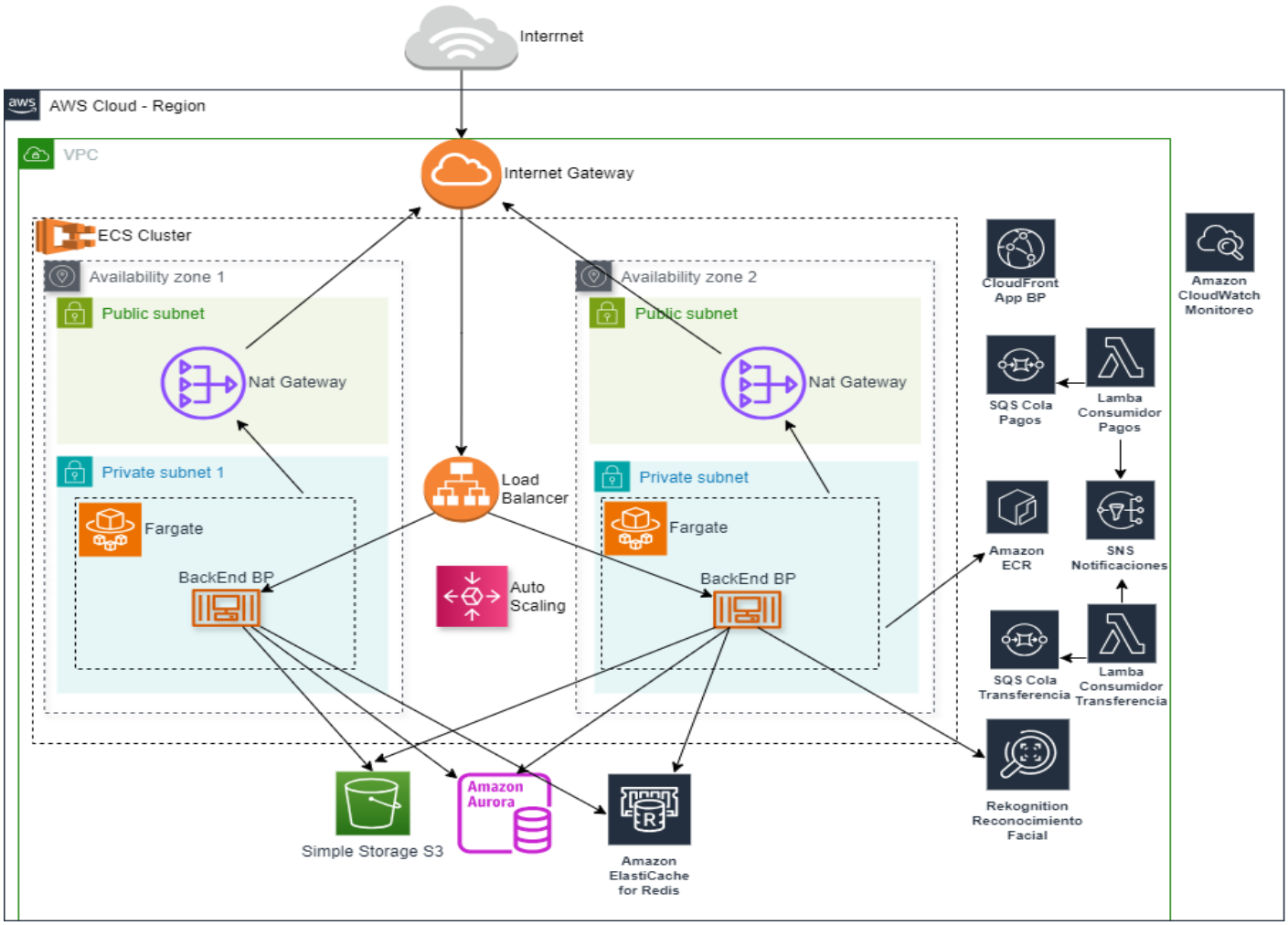
BackEnd: .Net 8

- **Api Gateway** → conecta 7 micros
- **Security**→ Usuarios, perfiles, Conf . Menu, Usuarios, Notificaciones
- **Customer**→ Gestionarclientes
- **Paying** → Gestiona los pagos
- **Transfering** → Gestiona transferencia de dinero.
- **Movement** → Consulta los movimientos del cliente
- **Product**: Gestiona los productos y servicios del cliente
- **External** → consulta de afuera/adentro o adentro/afuera, servicio google maps/ servicio policía nacional

BD: Aurora SQL y Redis

File Server: Sistema de archivos S3

Diagrama de arquitectura propuesta en AWS



Alta disponibilidad, tolerancia a fallos y monitoreo mediante la solución propuesta en una infraestructura en AWS

En base a los servicios que se han usado en la solución propuesta, vamos a ver como estas satisface las necesidades del negocio en los puntos de: Alta Disponibilidad, Tolerancia a Fallos y el Monitoreo de nuestra aplicación.

Alta Disponibilidad

1. **ECS Fargate**:

   Permite ejecutar contenedores sin necesidad de administrar servidores. La alta disponibilidad se logra mediante el despliegue de servicios en varias zonas de disponibilidad (AZs) y la integración con **Application Load Balancer** (ALB) para distribuir el tráfico de manera equilibrada entre las instancias de contenedores.

2. **Load Balancer**:

   Distribuye el tráfico entrante entre varias instancias de contenedores o instancias. El ALB está diseñado para funcionar en múltiples AZs, lo que proporciona alta disponibilidad al garantizar que el tráfico sea redirigido a instancias saludables en caso de fallos.

3. **Auto Scaling**:

   Ajusta automáticamente el número de instancias EC2 o tareas de Fargate según la demanda. Esto asegura que haya suficiente capacidad para manejar la carga y que la infraestructura se ajuste en respuesta a picos y caídas en el tráfico.

4. **Aurora BD**:

   Ofrece alta disponibilidad mediante la replicación en varias AZs y copias de seguridad automáticas. Aurora se asegura de que haya réplicas en diferentes AZs para proporcionar disponibilidad continua en caso de fallos de la base de datos primaria.

5. **Redis ElastiCache**:

   Proporciona alta disponibilidad mediante la replicación y la creación de clústeres de Redis en múltiples AZs. La conmutación por error automática asegura que el servicio se mantenga disponible en caso de fallos de nodos o redes.

6. **SQS**:

   Es un servicio de cola de mensajes altamente disponible que asegura que los mensajes se entreguen de manera confiable. SQS almacena mensajes en varias ubicaciones para garantizar la disponibilidad incluso si hay fallos en alguna parte del sistema.

7. **Lambda**:

   Es un servicio sin servidor que distribuye automáticamente las funciones en varias AZs. Esto asegura que las funciones Lambda sean resilientes a fallos en una AZ específica y puedan manejar solicitudes a nivel global.

Tolerancia a Fallos

1. **ECS Fargate**:

   Maneja fallos automáticamente al reiniciar tareas fallidas y redistribuir la carga entre las tareas restantes. Los contenedores se pueden distribuir en múltiples AZs para minimizar el impacto de fallos en una AZ.

2. **Load Balancer**:

   Realiza verificaciones de estado en las instancias y redirige el tráfico a instancias saludables. En caso de que una instancia falle, el ALB la elimina automáticamente del grupo de destino.

3. **Auto Scaling**:

   Detecta instancias fallidas y reemplaza automáticamente las instancias inactivas. Esto asegura que el número de instancias en funcionamiento se mantenga dentro de los parámetros especificados.

4. **Aurora BD**:

   Incluye la conmutación por error automática y réplicas de lectura en múltiples AZs. Si la base de datos principal falla, Aurora puede conmutar automáticamente a una réplica.

5. **Redis ElastiCache**:

   Soporta la conmutación por error automática y la replicación de datos. En caso de fallo de un nodo principal, el sistema puede promover un nodo réplica a principal sin pérdida de datos.

6. **SQS**:

   Garantiza la entrega de mensajes incluso si hay fallos en las instancias o servicios que consumen mensajes. Los mensajes se almacenan de forma redundante para garantizar que no se pierdan.

7. **Lambda**:

   Maneja automáticamente los fallos de ejecución. Si una función Lambda falla, el servicio puede reintentar la ejecución según la configuración y las políticas de reintento.

Auditoría

1. **ECS Fargate**:

   Puede integrarse con **CloudWatch Logs** para capturar y almacenar logs de contenedores, lo que permite auditar las actividades y comportamientos de los contenedores.

2. **Load Balancer**:

   Ofrece acceso a registros de acceso y eventos a través de **CloudWatch Logs**, permitiendo la auditoría del tráfico y la actividad del balanceador de carga.

3. **Auto Scaling**:

   Puede generar eventos y logs sobre las actividades de escalado, que se pueden capturar en **CloudWatch Logs** para auditoría y análisis.

4. **Aurora BD**:

   permite habilitar el registro de eventos de la base de datos y el log de auditoría. Los logs pueden ser enviados a **CloudWatch Logs** para su revisión.

5. **Redis ElastiCache**:

   Puede enviar logs de eventos y métricas a **CloudWatch Logs** para permitir la auditoría y el monitoreo de la actividad del clúster de Redis.

6. **SQS**:

   Permite registrar eventos y métricas en **CloudWatch Logs**. Esto ayuda a auditar la actividad de las colas y el procesamiento de mensajes.

7. **Lambda**:

   Integra automáticamente los logs de ejecución con **CloudWatch Logs**, proporcionando un registro completo de las invocaciones y errores de la función.

Monitoreo

1. **ECS Fargate**:

   Proporciona métricas de rendimiento y estado a través de **CloudWatch Metrics**. Puedes monitorear el uso de recursos, el estado de las tareas y otros indicadores clave de rendimiento.

2. **Load Balancer**:

   Genera métricas y logs sobre el tráfico, el rendimiento y el estado de las instancias en **CloudWatch Metrics** y **CloudWatch Logs**.

3. **Auto Scaling**:

   Ofrece métricas relacionadas con el escalado, como la cantidad de instancias y la actividad de escalado, que se pueden monitorear a través de **CloudWatch Metrics**.

4. **Aurora BD**:

   Envía métricas de base de datos, rendimiento y estado a **CloudWatch Metrics**. También puedes configurar alarmas para notificar sobre problemas de rendimiento.

5. **Redis ElastiCache**:

   Proporciona métricas sobre el rendimiento del clúster y el uso de recursos a **CloudWatch Metrics**. Los logs también pueden ser enviados a **CloudWatch Logs**.

6. **SQS**:

   Ofrece métricas sobre el número de mensajes en cola, el tiempo de espera y otros aspectos importantes en **CloudWatch Metrics**.

7. **Lambda**:

Proporciona métricas sobre el número de ejecuciones, la duración y la tasa de errores en **CloudWatch Metrics**. Los logs de ejecución también se envían a **CloudWatch Logs**.

Manejo de costos de la arquitectura en AWS

En base a los servicios que se han usado en la solución propuesta, se van a considerar estrategias para gestionar y optimizar eficazmente los costos en AWS, asegurando una infraestructura eficiente y económica.

1. **Monitoreo y Análisis de Costos**

- **AWS Cost Explorer**: Visualiza y analiza los costos y el uso de servicios.
- **AWS Budgets**: Establece presupuestos y recibe alertas de sobrepaso.
- **AWS Cost and Usage Report**: Obtén informes detallados sobre costos y uso.

2. **Optimización de Recursos**

- **Auto Scaling**: Ajusta automáticamente el número de instancias según la demanda.
- **Instancias Reservadas y Savings Plans**: Compra capacidad a largo plazo para obtener descuentos.
- **Spot Instances**: Utiliza capacidad no utilizada a precios reducidos.
- **Right-Sizing**: Ajusta el tamaño de los recursos para evitar sobre aprovisionamiento.

3. **Gestión de Datos y Almacenamiento**

- **S3 Storage Classes**: Usa diferentes clases de almacenamiento según el acceso y la durabilidad de los datos.
- **Lifecycle Policies**: Configura políticas para mover datos a clases de almacenamiento más económicas o eliminarlos.

4. **Reducción de Costos de Red y Transferencia**

- **Amazon CloudFront**: Utiliza un CDN para reducir costos de transferencia de datos.
- **VPC Endpoints**: Minimiza costos de transferencia de datos al mantener el tráfico dentro de la red de AWS.

5. **Uso de Etiquetas y Contabilidad**

- **Etiquetas y Cost Allocation Tags**: Aplica etiquetas para seguir y agrupar costos por proyecto o equipo.

6. **Costos de Servicios y Aplicaciones**

- **AWS Trusted Advisor**: Obtén recomendaciones para optimizar recursos y reducir costos.
- **AWS Compute Optimizer**: Analiza el uso de instancias EC2 y ofrece recomendaciones para optimizar el tamaño y tipo de instancia.

7. **Control de Gastos en Tiempo Real**

- **AWS Cost Anomaly Detection**: Identifica y alerta sobre desviaciones inusuales en el gasto.
- **Alerts and Notifications**: Configura alertas para mantener el control sobre el gasto y el uso.

Recomendaciones sobre el uso del servicio Oauth 2.0 interno

Para cumplir con el estándar OAuth 2.0, se deben implementar correctamente los flujos de autorización, proteger los tokens de acceso, autenticar y autorizar clientes adecuadamente, y proteger contra ataques comunes *Cross-Site Request Forgery*, *Cross-Site Scripting* y *Token Replay Attack*. Además, asegurarse de mantener una documentación completa, cumplir con normativas de protección de datos, y realizar pruebas y actualizaciones regulares para asegurar la seguridad y la eficacia de la implementación.

Regulaciones Bancarias y estándares de seguridad

Las regulaciones bancarias como los estándares de seguridad son fundamentales para garantizar la protección de los datos financieros, la privacidad de los clientes y la integridad de las transacciones. A continuación, se detallan algunas de las principales regulaciones bancarias y estándares de seguridad relevantes para el uso de aplicaciones bancarias:

Regulaciones Bancarias

Cada país maneja sus regulaciones bancarias en el caso de Perú se manejan las siguientes:

1. **Ley de Protección de Datos Personales (Ley 29733)**

   Regula la protección de datos personales y establece derechos y obligaciones en la gestión de la información personal.

2. **Ley General del Sistema Financiero (Ley 26702)**

   Regula las actividades de las instituciones financieras y establece normas para su funcionamiento y supervisión.

Estándares de Seguridad

1. **Estándar de Seguridad de Datos para la Industria de Tarjeta de Pago**

   Proporciona directrices para proteger la información de tarjetas de pago. Incluye requisitos para el cifrado de datos, el control de acceso, y la monitorización de redes.

2. **ISO/IEC 27001**

   Proporciona un marco para la gestión de la seguridad de la información. Define un sistema de gestión de seguridad de la información (ISMS) para proteger la confidencialidad, integridad y disponibilidad de los datos.

3. **ISO/IEC 27018**

Especifica controles para proteger la información personal en la nube. Se centra en el manejo de datos personales y la privacidad en los servicios de nube.

Consideraciones Adicionales

- **Autenticación Multifactor (MFA)**: Implementar MFA para añadir una capa adicional de seguridad a las aplicaciones bancarias.
- **Cifrado de Datos**: Asegurar que los datos en tránsito y en reposo estén cifrados para proteger la confidencialidad e integridad de la información.
- **Gestión de Identidades y Accesos (IAM)**: Establecer políticas claras para el acceso a los sistemas y datos críticos.
- **Monitorización y Auditoría**: Implementar herramientas para monitorizar actividades y realizar auditorías regulares para detectar y responder a incidentes de seguridad.

Conclusiones de la solución propuesta

El resultado que ofrece esta solución propuesta es adoptar un desarrollo basado en un patrón CQRS sobre AWS, BP puede construir una infraestructura robusta que no sólo cumpla con los requisitos de alta disponibilidad, tolerancia a fallos y recuperación ante desastres, sino que también garantice la seguridad y la capacidad de monitoreo continuo. Este enfoque integral asegura que tanto la aplicación y la infraestructura estén preparadas para enfrentar desafíos operativos y de seguridad, mientras se mantiene eficiente y adaptable a las necesidades cambiantes de la organización.

Repositorio Github

https://github.com/AGALDOSCORREA/DevsuBancaDigitalBP