# Arquitectura para el sistema de banca por Internet

## 1. ¿Qué frameworks usarías para el SPA y la Aplicación Móvil?

SPA:
1. Angular:
  - Justificación: Framework completo con herramientas integradas, adecuado para aplicaciones empresariales con necesidades sólidas y escalables. Su modularidad permite mantener un código organizado y reutilizable.

Aplicación Móvil:
1. Flutter:
  - Justificación: Le permite desarrollar aplicaciones nativas con una única base de código, ideal para experiencias ricas. Tiene soporte nativo para reconocimiento facial y de huellas dactilares.

## 2. ¿Qué flujo de OAuth 2.0 recomendaría?

Flujo recomendado: Authorization Code Flow with PKCE
- Justificación:
  - Este flujo es ideal para SPA y aplicaciones móviles, ya que no expone el token directamente.
  - Utilizar PKCE para evitar ataques de falsificación o interceptación de tokens.
  - Es seguro y cumple con los estándares industriales modernos de autenticación.

## 3. ¿Cómo se integra el reconocimiento facial en el Onboarding?

La propuesta de flujo con reconocimiento facial Herramientas sugeridas:
1. Herramientas sugeridas:
  - Amazon Rekognition (AWS): API robusta con alta precisión.
  - Face++: Ampliamente reconocido en la industria por sus capacidades biométricas avanzadas.
2. El flujo del proceso:
  - El usuario realiza un escaneo facial durante la incorporación para validar su identidad.
  - El usuario realiza un escaneo facial en la incorporación para validar su identidad.
Después de la incorporación:

- Nombre de usuario y contraseña.
- Huella digital APIs nativas del dispositivo móvil.
- Reconocimiento facial (opcional para mejorar la comodidad).

## 4. ¿Qué arquitectura de persistencia y auditoría se propone?

Arquitectura recomendada: Event Sourcing con CQRS
- Event Sourcing:
  - Todos los eventos (acciones del cliente) se registran y se almacenan como entradas inmutables en un sistema de eventos.
  - Ejemplo: Base de datos relacional como Amazon Aurora o Azure SQL.
- CQRS:
  - Desacopla las operaciones de lectura y escritura para optimizar consultas frecuentes.
  - Ejemplo: uso de Redis como caché para datos leídos con frecuencia.

## 5. ¿Qué servicios propone para la capa de integración?

Servicios identificados:
1. Consulta de datos básicos.
2. Consulta de movimientos.
3. Transferencias.
4. Notificaciones.
5. Preautorizaciones de traslados sensibles (nuevo servicio).
6. Almacenamiento en caché de consultas frecuentes (por ejemplo, Redis).

Optimización propuesta:
- Uso de un API Gateway como punto central de comunicación para desacoplar el frontend del backend.
- Servicios de agregación que combinan información fuentes para mejorar la experiencia del cliente.

## 6. ¿Qué elementos normativos/reguladores debes considerar?

Regulaciones importantes:
1. PCI DSS: Protege la información y las transacciones financieras.
2. ISO 27001: Garantiza la seguridad de la información.
3. GDPR o regulaciones locales: Cumplir con las leyes de protección de datos personales.
4. Autenticación multifactor (MFA): Fortalce la seguridad del acceso.

## 7. ¿Cómo se garantiza alta disponibilidad, tolerancia a fallos y recuperación ante desastres?

Alta disponibilidad (HA):
- Infraestructura desplegada en varias zonas de disponibilidad en AWS o Azure.
- Balanceadores de carga para la distribución del tráfico.

Tolerancia a fallas:
- Circuit Breakers (ejemplo ., Resilience4J, Polly.js) para manejar interrupciones de servicios externos.
- Mecanismos de failover para bases de datos.

Recuperación ante desastres (DR):
- Backups automatizados.
- Replicación geográfica para garantizar recuperación rápida en caso de desastres.

Auto-healing:
- Uso de contenedores en Kubernetes con capacidades de reinicio automático en caso de fallos.

## 8. ¿Qué herramientas usarías para monitoreo?

Propuesta de monitoreo:
1. AWS CloudWatch/Azure Monitor para métricas de monitoreo en tiempo real.
2. ELK Stack (Elasticsearch, Logstash, Kibana) para análisis de logs y trazabilidad de eventos.
3. Prometheus y Grafana: Monitoreo y visualización en tiempo real de la infraestructura y servicios.

## 9. ¿ Cómo se propone una arquitectura desacoplada?

Mi propuesta para la arquitectura:
- Microservicios:
  - Implementación de cada funcionalidad como un servicio autónomo.
  - Capacidad de escala horizontal y conveniencia para el mantenimiento.
- API Gateway:
  - Abstracción de las llamadas a los microservicios y simplifica la comunicación entre frontend y backend.
- Eventos:

- Empleo de sistema on-event-based para las interacciones asincrónicas, (por ejemplo, RabbitMQ, Kafka).

## 10. Diseño basado en el modelo C4

1. Modelo de Contexto:

Propósito:

- Mostrar una descripción general de alto nivel del sistema, destacando las interacciones entre los actores (usuarios y sistemas externos) y el sistema subyacente..

¿Qué incluye?:

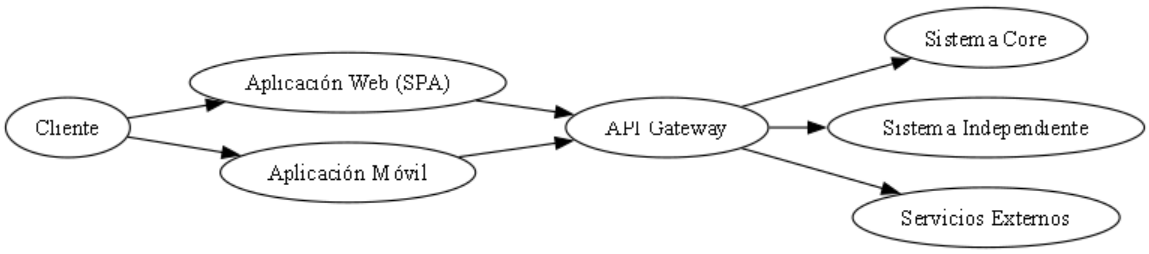
• Usuario: la persona o sistema que interactúa con la aplicación (por ejemplo, un cliente o empleado).

• Sistema central: el sistema que usted diseña, como la banca en línea en este ejemplo.

• Sistemas externos: otros sistemas con los que interactúa su sistema, como servicios de notificación, API bancarias o plataformas de verificación de identidad.

Ejemplo:

- • El cliente utiliza SPA (aplicación de página única) o aplicación móvil para transferir dinero. Estas aplicaciones se comunican con puertas de enlace API, que se comunican con sistemas externos como bases de datos o servicios de notificación.




2. Modelo de contenedor:

Propósito:

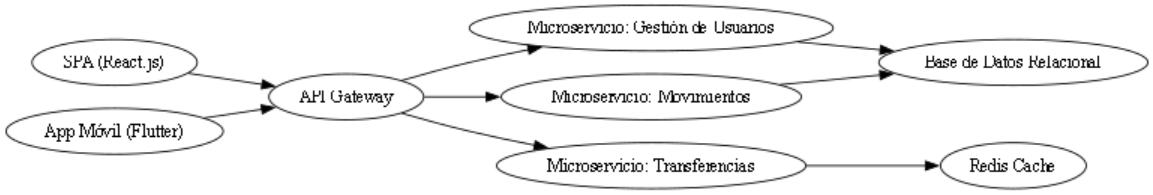
- Describe la arquitectura interna del sistema desde un nivel más detallado. Muestra cómo se dividen las responsabilidades entre aplicaciones y servicios principales (contenedores).

¿Qué incluye?:

- Frontend: el componente que interactúa directamente con el usuario, como un SPA o una aplicación móvil.
- Backend: servicios y microservicios que gestionan la lógica empresarial. Como por ejemplo transmisión, notificación o autenticación.
- Base de datos: almacenamiento de datos estructurado o en caché.
- Capa de integración: puerta de enlace API que actúa como intermediario entre los servicios externos e internos..

Ejemplo:

- Una SPA/App movil envía una solicitud al API Gateway para consultar los movimientos de una cuenta. El API Gateway redirige esta solicitud a un microservicio encargado de las consultas, que accede a una base de datos relacional.




3. Modelo de Componentes:
Propósito:

- Obtener una comprensión más profunda de la arquitectura a nivel de componentes dentro del contenedor. Esto describe qué módulos específicos componen el servicio.

¿Qué incluye?:

- Módulo específico: clases, funciones o servicios que trabajan juntos para realizar una función específica.
- Conexiones internas: cómo estos módulos interactúan entre sí, dentro del contenedor.
- Integraciones clave: conexión a servicios externos, Como sistema de autenticación o auditoría.

Ejemplo:

- El API Gateway se comunica con:
    - Un servicio de autenticación que utiliza OAuth2 para autenticar a los usuarios.
    - Un servicio de auditoría para registrar las solicitudes.

- Un servicio de notificaciones para enviar alertas al cliente.