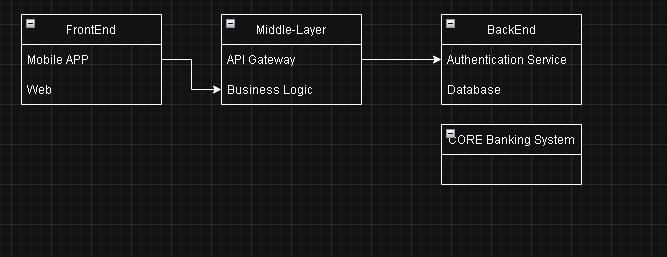
Luis Alberto Garcia Diaz

Eval. Tec. Jefe de Desarrollo/Líder técnico

1. 
   1. Autenticación y Autorización
      1. OAuth 2.0 + JWT para delegar acceso entre front-end y back-end de forma tokenizada
      2. Tokens firmados con claves asimétricas (RSA/ECDSA) y con expiración corta
      3. Validación de scopes y roles en cada capa intermedia
   2. Encriptación de Datos en Tránsito
      1. TLS 1.3 obligatorio en todos los canales (app ↔ API ↔ core bancario)
   3. Intercambio Seguro entre Microservicios
      1. Comunicación basada en gRPC con TLS o APIs internas con MTLS (mutua autenticación)
      2. Uso de certificados rotativos.
   4. Firewalls Lógicos por Capa
      1. Cada capa ubicada en una zona segura aislada
      2. Reglas de comunicación estrictas: solo los puertos y protocolos necesarios
   5. Control de Acceso Interno Granular
      1. Políticas RBAC o ABAC en servicios backend y bases de datos
      2. Validación cruzada entre el token del usuario y el sistema intermedio antes de ejecutar operaciones
   6. Auditoría y Trazabilidad
      1. Todos los llamados entre capas deben ser auditables, con requestId y userId propagados
      2. Logs firmados y protegidos contra alteraciones
   7. Protección contra Ataques Comunes
      1. Validación estricta en el front-end y en el middleware (anti-inyección, anti-XSS)
      2. Rate limiting y detección de anomalías para prevenir ataques de fuerza bruta
   8. Capa de Observabilidad con Seguridad
      1. Dashboards que visualicen únicamente metadatos necesarios
      2. Acceso a monitoreo restringido por perfil técnico y separado del entorno productivo
2. Tecnologias:
   1. Front:
      1. Flutter o MAUI, Desarrollo unificado para ambas plataformas
      2. React: madurez y alto rendimiento
      3. TypeScript: Tipado estricto
      4. Redux: Escalabilidad en los datos
   2. Back:
      1. .net 8: Seguridad y escalibidad
      2. gRPC:Intercomunicacion
      3. IdentityServer: Oauth
      4. SQL Server: base de datos
      5. Redis: Cache y sesiones
      6. Vault: Gestion de credenciales
      7. Docker+ kubernets: Orquestacion.
   3. Core Bancario:
      1. Kafka para desacoplar flujos
      2. Conexión vía VPN segura o circuitos dedicados
   4. DevOps
      1. GitHub Actions
      2. Terraform
      3. GitHub: Code Repository.
      4. Elastic: logs
3. Seguridad:
   1. Autenticación y Autorización
      1. Implementación de OAuth 2.0 con JWT, validando scopes, roles, y origen de la solicitud.
      2. Tokens firmados con RSA y rotación periódica de claves.
      3. Login adaptativo (según dispositivo, ubicación o comportamiento).
      4. Integración con MFA y biometría en la app móvil.
   2. Protección de Datos en Tránsito y en Reposo
      1. TLS 1.3 obligatorio en todas las capas: comunicación interna, móvil-backend, y con el Core bancario.
      2. mTLS para microservicios sensibles que requieren autenticación mutua.
      3. Cifrado de datos en reposo usando AES-256, con claves gestionadas por Vault o KMS empresarial.
      4. Separación lógica de datos sensibles (credenciales, transacciones) y datos operativos.
   3. Gestión de Sesiones y Tokenización
      1. Tokens stateless con expiración corta y refresco seguro.
      2. Detección y revocación de sesiones comprometidas.
      3. Tokenización de identificadores sensibles (como cuenta CLABE) antes de almacenarlos o transmitirlos.
      4. Persistencia segura en Redis con TTL dinámico según tipo de sesión.
   4. Buenas Prácticas OWASP contra Ataques Comunes
      1. Validación de entradas desde front-end hasta middleware: anti-inyección, anti-XSS, anti-CSRF.
      2. Rate limiting, CAPTCHA y detección de anomalías en tiempo real.
      3. Uso de Content Security Policy (CSP) para navegación web.
      4. Escaneo automático de dependencias y paquetes para vulnerabilidades conocidas.
      5. Sandbox de componentes expuestos o con ejecución remota.
4. CD/CI/Logs
   1. Integración Continua (CI)
      1. Repositorio centralizado (GitHub) con ramas protegidas (main, release, hotfix)
      2. Validación automática por PR (Pull Request):
      3. Linter y pruebas de sintaxis (ESLint)
      4. Unit tests con JUnit
      5. Análisis de vulnerabilidades (GitHub)
      6. Versionado semántico automático (major. minor. patch) con etiquetas y changelogs
      7. Construcción de artefactos: imágenes Docker versionadas, empaquetado de apps móviles (.apk / .ipa)
   2. Despliegue Continuo (CD)
      1. Entornos separados: Dev, QA, Staging, Producción (con pipelines independientes y controles por ambiente)
      2. Despliegue progresivo:
         1. Blue/Green deployments o Canary releases para minimizar riesgos
      3. Rollback automático si hay fallos en health checks
      4. Orquestación con Kubernetes y Helm Charts para mantener consistencia entre entornos
      5. Cifrado y gestión de secretos:
      6. Azure Key Vault para inyectar credenciales vía sidecar o config maps
   3. Auditoría y visibilidad
      1. Logs centralizados con Elastic
      2. Trazabilidad con OpenTelemetry, propagando requestId y userId entre servicios
   4. Automatización y Gobernanza
      1. GitOps: infraestructura como código con Terraform, validada y auditada por commit
      2. Pipelines como código (azure-pipelines.yml) con historial completo
      3. Validación de firmas de commit y autorización por roles para el despliegue a producción
      4. Monitoreo de eficiencia del pipeline: tiempo promedio por paso, frecuencia de errores, tasa de rollback
5. RoadMap
   1. Fases:
      1. Fundamentos:(Mes 1- mes 2)
      2. Servicios Claves (mes3-mes 4
      3. Optimización (mes 5, mes 6)
   2. Roles: (nombre|Rol|Complemento)
      1. Front-End | Flutter, React | UX, pruebas adaptativas |
      2. Back-End | .NET , gRPC | Seguridad, integración core |
      3. DevOps | CI/CD, Helm, Vault, Kubernetes | SRE, alertamiento |
      4. Seguridad | Criptografía, protección OWASP | Sesiones, MFA, monitoreo |
      5. QA | Automatización, stress testing | Validación continua |
      6. Product | Priorización, roadmap | Comunicación con stakeholders |

|  |  |  |  |
| --- | --- | --- | --- |
| Fase | Sprint | Objetivo Clave | Roles |
| 1 | 1 | Setup de CI/CD, gestión de secretos, diseño de arquitectura | DevOps |
|  | 2 | Módulo de autenticación, mTLS entre servicios, base de datos | Back-end, Seguridad, QA |
|  | 3 | Primer prototipo UI/UX móvil, login biométrico | Front-end,QA |
| 2 | 4 | Módulos: consulta de saldo, movimientos, notificaciones | Back-end, Front-end, QA |
|  | 5 | Integración controlada con Core bancario (Kafka o adaptadores) | DevOps, Front-end, Back-End |
|  | 6 | Implementación de alertas, logs distribuidos, monitoreo | Back-end, DevOps, product |
| 3 | 7 | Pruebas de estrés, hardening, ajustes de rendimiento | QA, Seguridad, DevOps |
|  | 8 | Orquestación completa en Kubernetes, habilitación multicanal | DevOps, Front-end |
|  | 9 | Publicación controlada, feedback de usuarios, governance | Product Owner, QA |

Extras:

* Normativas Aplicables:
  + La arquitectura propuesta se fundamenta en estándares internacionales como ISO 27001, PCI DSS y prácticas OWASP, asegurando que cada capa del sistema cumpla con requisitos de confidencialidad, integridad y disponibilidad, de acuerdo con la normatividad mexicana y global.

|  |  |  |
| --- | --- | --- |
| **Norma** | **Aplicación** | **Observaciones** |
| **ISO/IEC 27001** | Gestión de la seguridad de la información | Requiere controles organizativos, físicos y técnicos, ideal para justificar trazabilidad, controles de acceso y logs |
| **ISO/IEC 27002** | Prácticas de seguridad complementarias | Aplica directamente en componentes como hardening de servidores, políticas de contraseña y segregación de redes |
| **PCI DSS** | Protección de datos financieros | Especialmente relevante si el sistema procesa transacciones con tarjetas (no se menciona en el contexto, pero se mencionó en la reunión previa), exige cifrado, tokenización y auditoría |
| **OWASP ASVS / Top 10** | Desarrollo seguro de aplicaciones | Alinear validaciones anti-XSS, CSRF, inyección y otros controles a esta guía. Se acepta como estándar global en desarrollo seguro |
| **Ley de Protección de Datos Personales (LGPDPPSO)** | México | Para reforzar el componente legal de privacidad, pero vale destacar su implicación en trazabilidad y derechos ARCO |
| **Normatividad CNBV (México)** | Requisitos de seguridad bancaria | Ideal para justificar separación de ambientes, monitoreo continuo, autenticación fuerte y resiliencia ante fallos |

* 🔧 **Marcos DevOps y CI/CD**

|  |  |  |
| --- | --- | --- |
| **Norma/Guía** | **Aplicación** | **Comentario** |
| **NIST SP 800-218 (SSDF)** | Desarrollo seguro integrado | Propone prácticas para CI/CD y gestión de vulnerabilidades. Puedes justificar escaneo de código y validación por commit |
| **CNCF Best Practices** | Entornos Kubernetes y contenedores | Puedes alinear tu orquestación con Helm, gestión de secretos con Vault |
| **GitOps Principles** (por Weaveworks) | Gobernanza y trazabilidad | Refuerza que toda infraestructura se gestiona como código, auditada y versionada |