Am0

Presentación de Caso

Mayo 2025

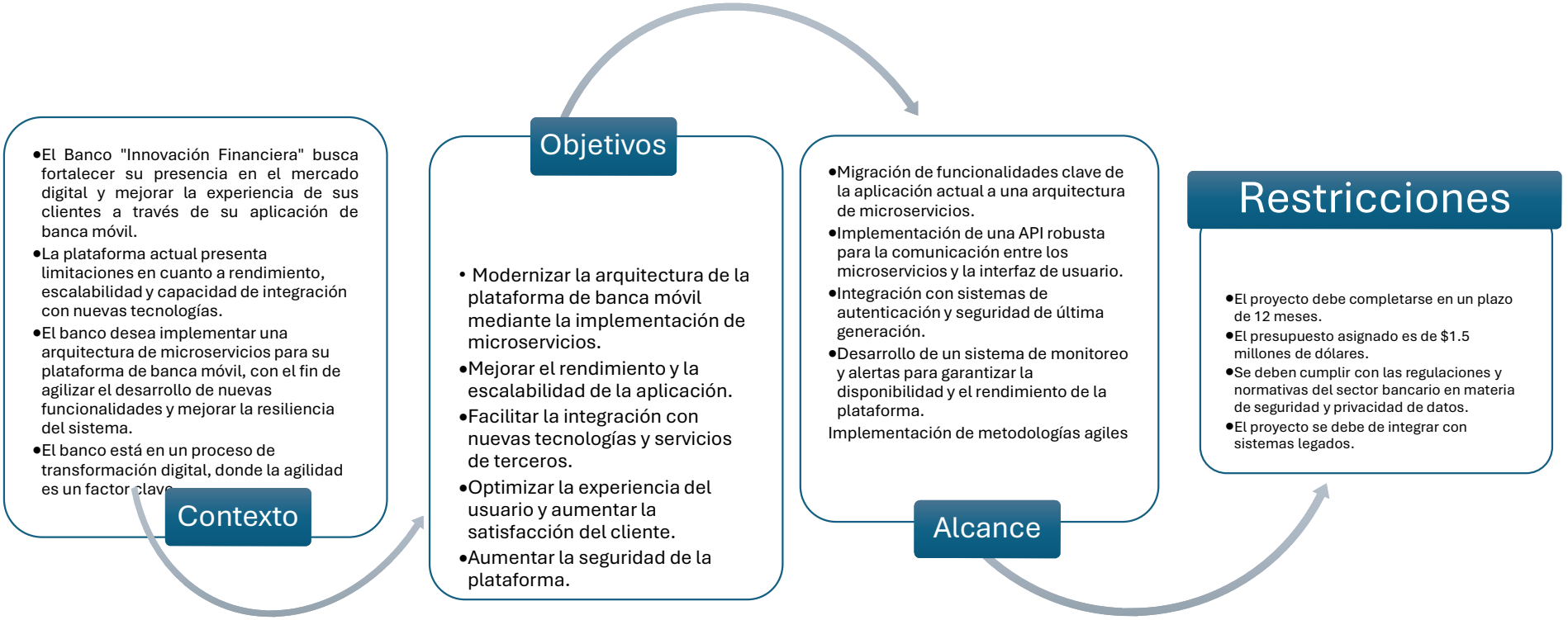
Diapositiva 1

Am0 Slide de Portada

Asistencia MKT, 2024-03-04T21:03:50.715

Planteamiento del Caso

BP busca modernizar su plataforma de Banca Móvil



Planeación y estrategia

¿Cómo abordaría la planificación de este proyecto, considerando la complejidad de la migración a microservicios?

1.Análisis y Evaluación Inicial:

1. Evaluación de la arquitectura actual y sus limitaciones.
2. Identificación de servicios críticos a migrar y dependencias con sistemas legados.
3. Definición de objetivos técnicos y funcionales alineados con la estrategia del banco (OKR).

2.Diseño de la Arquitectura de Microservicios:

1. Definición de dominios funcionales y modelado de microservicios siguiendo DDD (Domain-Driven Design).
2. Diseño de APIs y contratos de comunicación entre microservicios (Open Api, REST, gRPC, Mensajería asíncrona).
3. Selección de tecnologías para contenerización (Docker, Kubernetes) y orquestación.

3.Estrategia de Migración:

1. **Migración progresiva con enfoque strangler pattern (Transformar, Coexistir, Eliminar)**, comenzando con servicios no críticos.
2. Implementación de un API Gateway para facilitar la convivencia entre sistemas legados y la nueva arquitectura.
3. Pruebas piloto en entornos controlados antes del despliegue completo.

4.Planificación del Proyecto:

1. **Metodología ágil (Scrum o SAFe)** con entregas incrementales cada 2-4 semanas. (mejor escenario cada 15 dias)
2. Equipos multidisciplinarios enfocados en diferentes dominios del negocio.
3. Implementación de CI/CD para automatizar despliegues y pruebas.

Recomendación integración con sistema legados

El desafío clave es conectar la nueva arquitectura de microservicios con los sistemas actuales. Se recomienda:

🔷 **Uso de APIs y middleware**: Para exponer servicios de sistemas antiguos sin modificar su código fuente.

🔷 **ETL y mensajería (Kafka, RabbitMQ)**: Para sincronizar datos entre sistemas nuevos y legados en tiempo real.

🔷 **Pruebas de compatibilidad**: Validar interoperabilidad y minimizar riesgos en la transición.

Planeación y estrategia

¿Qué metodologías de gestión de proyectos utilizaría y por qué?

- **Scrum** para desarrollo iterativo, asegurando entregas frecuentes y validaciones continuas. Puede cambiar las prioridades de acuerdo con las necesidades.
- **SAFe (Scaled Agile Framework)** para coordinar equipos grandes, alineando la visión estratégica con equipos técnicos.
- **Kanban** para la integración con sistemas legados y tareas de mantenimiento.

¿Cómo gestionaría la integración con los sistemas legados?

- **Uso de API Gateway** para exponer funcionalidades heredadas como APIs modernas.
- **Implementación de adaptadores (Anti-Corruption Layer)** para minimizar impactos en los sistemas legados.
- **Migración por fases**, priorizando servicios que aporten mayor valor y asegurando compatibilidad durante la transición.
- **Monitoreo y pruebas continuas** para detectar y mitigar riesgos de incompatibilidad. Pude implementarse pruebas unitarias en edad temprana para poder asegurar (Sonar Qube)

Gestión de riesgos

¿Cuáles son los principales riesgos que identifica en este proyecto y cómo los mitigaría?

Riesgo	Estrategia de Mitigación
Retrasos en la migración	Implementar MVPs funcionales, con entregas incrementales y monitoreo constante.
Incompatibilidad con sistemas legados	Uso de API Gateway y pruebas automatizadas de integración.
Falta de adopción por parte de usuarios	Diseño UX/UI basado en pruebas con usuarios y retroalimentación continua.
Riesgos regulatorios	Cumplimiento normativo desde la fase de diseño, con revisiones legales periódicas.

¿Cómo manejaría posibles retrasos o desviaciones en el presupuesto?

- **Priorización basada en valor:** Enfocar esfuerzos en servicios críticos primero.
- **Gestión ágil del alcance:** Ajustar funcionalidades no esenciales si el presupuesto se ve comprometido.
- **Monitoreo financiero continuo:** Evaluación mensual de costos y recursos.

Gestión de riesgos

¿Cómo manejaría los riesgos de ciberseguridad?

- **Zero Trust Architecture:** Implementación de autenticación robusta (MFA, OAuth 2.0, OpenID Connect).
- **Cifrado de extremo a extremo:** TLS 1.3, cifrado en base de datos y almacenamiento seguro de claves.
- **Pruebas de seguridad continuas:** Pentesting, auditorías de código y monitoreo de amenazas en tiempo real.
- **Gestión de accesos basada en roles (RBAC/ABAC).**

Liderazgo y Comunicación

¿Cómo gestionaría un equipo multidisciplinario de desarrolladores, arquitectos y especialistas en seguridad?

- **Equipos ágiles organizados por dominio funcional.**
- **Rituales Scrum:** Daily stand-ups, Sprint reviews, Retrospectivas.
- **Uso de herramientas de colaboración:** Azure Devops Boards, Jira, Confluence, Miro, Microsoft Teams.
- **Capacitación continua:** Talleres sobre microservicios, seguridad y metodologías ágiles.

¿Cómo garantizaría una comunicación efectiva con los stakeholders?

- **Reuniones de alineación periódicas:** Presentación de avances cada Sprint.
- **Dashboard de estado del proyecto:** Indicadores clave de progreso y riesgos.
- **Workshops con usuarios finales:** Para validar la experiencia de usuario.

¿Cómo gestionaría los conflictos dentro del equipo?

- **Fomentar una cultura de feedback abierto y colaboración.**
- **Resolver desacuerdos con reuniones 1:1 y facilitación de mediadores.**
- **Definir claramente roles y responsabilidades para evitar fricciones.**

Aspectos técnicos

¿Qué tecnologías y herramientas recomendaría para la implementación de la arquitectura de microservicios?

Componente	Tecnología
Lenguaje	.NET 8, C#, Java , Spring Boot, Maven , Gradle
Contenerización	Docker, Kubernetes (AKS)
API Gateway	Ocelot o Kong, pasarela actual de banco
Mensajería	RabbitMQ, Kafka
Base de Datos	SQL Server o postgress para datos transaccionales, Redis para caching
Autenticación	OAuth 2.0, OpenID Connect, Azure AD
Monitoreo	Prometheus, Grafana, Elastic Stack, Sonar, herramienta de medición código que BP dispone.

Aspectos técnicos

¿Cómo garantizaría la seguridad y escalabilidad de la plataforma?

- **Uso de arquitectura event-driven con Kafka para escalabilidad.**
- **Pruebas de carga y estrés con JMeter.**
- **Políticas de seguridad reforzadas (CORS, CSP, validación de entradas).**

¿Cómo plantearía la estrategia de pruebas?

- **Pruebas Unitarias:** xUnit, NUnit.
- **Pruebas de Integración:** Postman, Swagger, RestAssured.
- **Pruebas de Carga:** JMeter, k6.
- **Pruebas de Seguridad:** OWASP ZAP, Burp Suite.

Seguimiento y Control

¿Qué KPIs utilizaría para medir el éxito del proyecto?

- **Disponibilidad del sistema (>95.9%).**
- **Tiempo medio de respuesta (<200ms).**
- **Índice de satisfacción del usuario (>85%).**
- **Tiempo promedio de desarrollo por funcionalidad.**

¿Cómo realizaría el seguimiento del progreso del proyecto y la gestión del presupuesto?

- **Dashboards con KPIs en Power BI o Grafana.**
- **Reuniones quincenales con dirección para revisión de hitos.**
- **Análisis de desviaciones presupuestarias mensuales.**

¿Cómo aseguraría que el proyecto cumpla con los objetivos de negocio?

- **Alineación con estrategia digital del banco.**
- **Validación de usuarios finales en cada iteración.**
- **Estrategia de go-live con monitoreo post-despliegue.**

Cumplimiento normativo

Para garantizar la conformidad con normativas del sector financiero (ISO 27001, PCI DSS, GDPR, Ley de Protección de Datos), se deben implementar:

✅ **Autenticación segura**: MFA (Multi-Factor Authentication) y cifrado robusto (AES-256, TLS 1.2/1.3).

✅ **Gestión de accesos**: Zero Trust y segregación de roles.

✅ **Monitoreo continuo**: SIEM para detección de amenazas en tiempo real.

✅ **Cumplimiento regulatorio**: Auditorías periódicas y documentación de seguridad.

Plan ejecución 12 meses

Para cumplir con el plazo, se recomienda una metodología ágil (Scrum o SAFe) con ciclos de desarrollo iterativos. Se dividirá el proyecto en **fases clave**:

Fase	Duración	Actividades Clave
1. Análisis y planificación	2 meses	Levantamiento de requerimientos, análisis de sistemas legados, definición de arquitectura y roadmap.
2. Desarrollo de microservicios núcleo	4 meses	Implementación de módulos críticos como autenticación, cuentas y transacciones.
3. Integración con sistemas legados	2 meses	Desarrollo de APIs y middleware para comunicación con sistemas actuales.
4. Pruebas y optimización	2 meses	Pruebas de carga, seguridad y usabilidad.
5. Go-Live y monitoreo	2 meses	Despliegue gradual, monitoreo y soporte postproducción.

Manejo del presupuesto 1.5 Millones

Se debe distribuirse eficientemente considerando:

Se priorizarán soluciones **cloud-native** (AWS, Azure) para reducir costos de hardware y asegurar escalabilidad.

Área	% del Presupuesto	Monto Estimado (USD)
Desarrollo y personal técnico	50%	$750,000
Infraestructura en la nube	20%	$300,000
Seguridad y cumplimiento normativo	15%	$225,000
Monitoreo y soporte post-Go-Live	10%	$150,000
Capacitación y comunicación	5%	$75,000