Arquitetura Técnica Detalhada - Fintech Digital MVP

1. Visão Geral da Arquitetura

A arquitetura proposta para o MVP da Fintech Digital é baseada em **Microservices** e utiliza uma abordagem **Cloud-Native** para garantir **escalabilidade horizontal**, **resiliência** e **alta disponibilidade**. O objetivo é separar as preocupações de negócio em serviços independentes, facilitando o desenvolvimento, a implantação e a manutenção.

2. Componentes e Camadas

A arquitetura é dividida em quatro camadas principais: Acesso, Aplicação (Microservices), Dados e Infraestrutura/Observabilidade.

2.1. Camada de Acesso

Componente	Tecnologia Sugerida	Fu
Clientes	Mobile (iOS/Android - React Native/Flutter) e Web (React/Vue)	I
API Gateway	AWS API Gateway / Kong / Nginx	F F e
WAF (Web Application Firewall)	Cloud-native WAF (ex: AWS WAF)	F X

2.2. Camada de Aplicação (Microservices)

Os serviços devem ser construídos utilizando linguagens modernas (ex: Go, Java/Kotlin, Python/FastAPI) e empacotados em containers Docker.

Microservice	Responsabilidade Principal
Service Accounts	Gerenciamento de contas de usuário, saldos e extratos.
Service Payments	Processamento de transações (PIX, TED, Boleto) e integração com provedores de pagamento (PSPs).
Service Cards	Emissão, bloqueio e gestão de cartões virtuais e físicos (integração com *card processor*).
Service KYC/AML	Fluxo de onboarding, validação documental (OCR), score de risco, e monitoramento de transações para Prevenção à Lavagem de Dinheiro.
Service Notifications	Envio de notificações (Push, E-mail, SMS) e webhooks para eventos de transação.
Service Compliance	Geração de relatórios regulatórios e logs de auditoria.

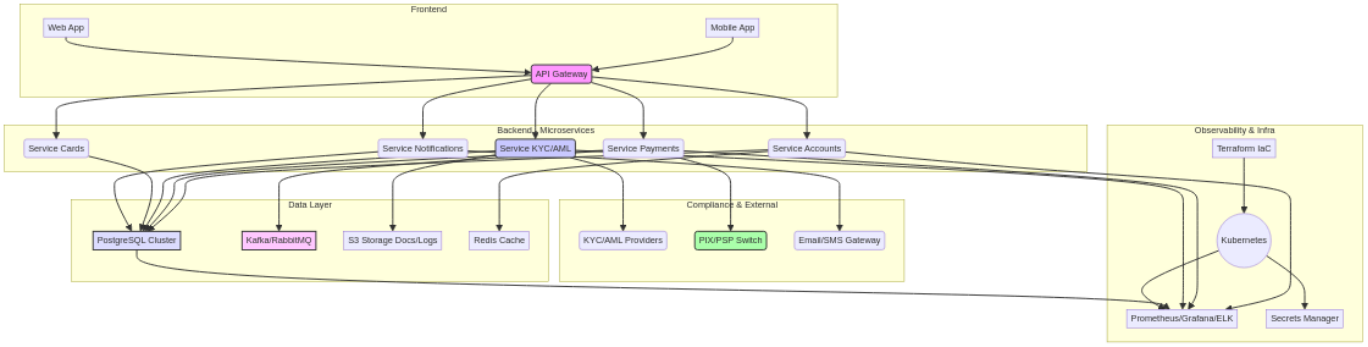
2.3. Camada de Dados

Componente	Tecnologia Sugerida	Funç
Banco de Dados Transacional	**PostgreSQL** (Cluster com réplica para alta disponibilidade)	Arm usu
Message Broker / Event Store	**Apache Kafka** ou **RabbitMQ**	Utili mic dur
Cache	**Redis**	Arm (ses leitu
Data Warehouse	Snowflake / Google BigQuery	Arm dad
Storage de Documentos	S3-compatible Storage (ex: AWS S3)	Arm com

2.4. Camada de Infraestrutura e Segurança

Componente	Tecnologia Sugerida
Orquestração	**Kubernetes (K8s)**
Infraestrutura como Código (IaC)	**Terraform**
Gerenciamento de Segredos	**HashiCorp Vault** / AWS Secrets Manager
Observabilidade	Prometheus (Métricas) + Grafana (Dashboards) + ELK Stack (Logs)

3. Diagrama de Arquitetura (Visão Geral)



3.1. Código Fonte do Diagrama (Mermaid)

O diagrama abaixo ilustra a comunicação entre os principais componentes da arquitetura.

```mermaid graph TD subgraph Frontend A[Mobile App] -->|HTTPS| B(API Gateway) C[Web App] -->|HTTPS| B end

Plain Text

```
subgraph Backend - Microservices
    B --> MS1(Service Accounts)
    B --> MS2(Service Payments)
    B --> MS3(Service KYC/AML)
    B --> MS4(Service Cards)
    B --> MS5(Service Notifications)
end

subgraph Data Layer
```
```

```
    MS1 --> DB1[PostgreSQL Cluster]
    MS2 --> DB1
    MS3 --> DB1
    MS4 --> DB1
    MS5 --> DB1
    MS2 --> MB[Kafka/RabbitMQ]
    MS3 --> S3[S3 Storage (Docs/Logs)]
    MS1 --> Cache[Redis Cache]
end

subgraph Compliance & External
    MS3 --> EXT1(KYC/AML Providers)
    MS2 --> EXT2(PIX/PSP Switch)
    MS5 --> EXT3(Email/SMS Gateway)
end

subgraph Observability & Infra
    K8s((Kubernetes))
    TF[Terraform (IaC)]
    Obs[Prometheus/Grafana/ELK]
    Vault[Secrets Manager]

    MS1 --> Obs
    MS2 --> Obs
    MS3 --> Obs
    DB1 --> Obs
    K8s --> Obs

    TF --> K8s
    K8s --> Vault
end

style B fill:#f9f,stroke:#333,stroke-width:2px
style MS3 fill:#ccf,stroke:#333,stroke-width:2px
style DB1 fill:#ddf,stroke:#333,stroke-width:2px
style MB fill:#fcf,stroke:#333,stroke-width:2px
style EXT2 fill:#afa,stroke:#333,stroke-width:2px
```

```

# 4. Estratégias de Segurança

1. **Segurança de Dados:** Uso obrigatório de **HSM** (Hardware Security Module) ou serviço de gerenciamento de chaves para chaves PIX e certificados TLS. Criptografia de ponta a ponta.
```

2. **Segurança da Aplicação:** Implementação de **RBAC** (Role-Based Access Control) em todos os microservices. Uso de **SAST/DAST** (Static/Dynamic Application Security Testing) no pipeline de CI/CD.
3. **Segurança de Rede:** Segmentação de rede (VPC/VNet) para isolar a camada de dados da camada de aplicação. VPN para acesso administrativo.
4. **Auditoria:** Logs de auditoria imutáveis (WORM - Write Once, Read Many) armazenados no S3/Data Warehouse, com retenção mínima de 5 anos (conforme regulamentação financeira).

5. Estratégias de Implantação

- **CI/CD:** Uso de pipelines automatizados (ex: GitLab CI, GitHub Actions) para testes, construção de imagens Docker e implantação no Kubernetes (GitOps).
- **Ambientes:** Mínimo de três ambientes: Desenvolvimento (Dev), Homologação (HML) e Produção (Prod). O ambiente HML deve ser usado para testes de integração com PSPs e o PIX Sandbox do BCB.
- **Deployment:** Estratégias de *Canary Deployment* ou *Blue/Green* para minimizar o risco de *downtime* durante atualizações.