

Documento de Arquitetura - Sistema de Gestão de Fluxo de Caixa

1. Visão Geral do Sistema

1.1 Contexto de Negócio

O sistema **Carrefour CaseFlow** foi desenvolvido para atender às necessidades de controle de fluxo de caixa de um comerciante, proporcionando:

- Controle detalhado de lançamentos financeiros (débitos e créditos)
- Consolidação automática de saldos diários
- Relatórios de posição financeira em tempo real
- Histórico de movimentações por período

1.2 Objetivos Arquiteturais

- **Escalabilidade:** Suportar crescimento de volume de transações
- **Resiliência:** Garantir disponibilidade mesmo com falhas de componentes
- **Desempenho:** Processar 50 req/s com máximo 5% de perda
- **Manutenibilidade:** Facilitar evolução e correções
- **Observabilidade:** Monitoramento completo do sistema

2. Mapeamento de Domínios e Capacidades de Negócio

2.1 Domínios Funcionais Identificados

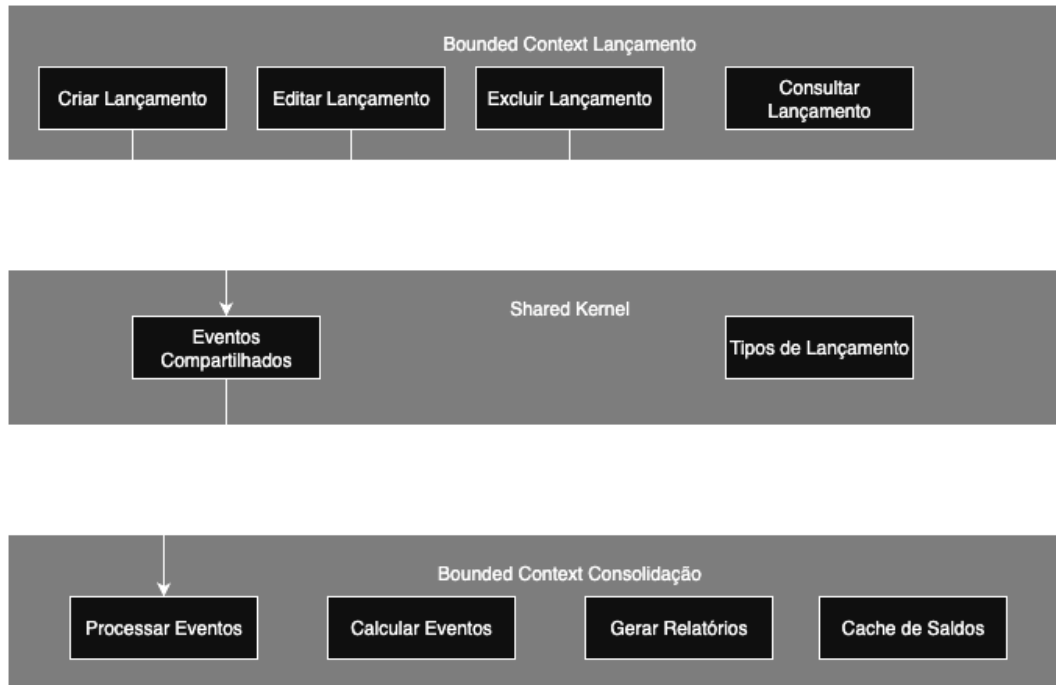
Domínio de Lançamentos

- **Responsabilidade:** Gestão completa do ciclo de vida dos lançamentos financeiros
- **Capacidades:**
 - Criação de lançamentos (débitos/créditos)
 - Edição de lançamentos existentes
 - Exclusão de lançamentos
 - Consulta de lançamentos por ID
 - Listagem paginada de lançamentos
 - Busca por período específico
 - Validação de regras de negócio

Domínio de Consolidação

- **Responsabilidade:** Consolidação e cálculo de saldos diários
- **Capacidades:**
 - Processamento de eventos de lançamentos
 - Cálculo automático de saldos consolidados
 - Manutenção de histórico de saldos
 - Cache de consultas frequentes
 - Relatórios de posição financeira

2.2 Bounded Contexts



3. Levantamento de Requisitos

3.1 Requisitos Funcionais Refinados

RF001 - Gestão de Lançamentos

- Descrição: O sistema deve permitir CRUD completo de lançamentos financeiros
- Critérios de Aceite:
 - Validação de valores positivos
 - Classificação obrigatória (débito/crédito)
 - Descrição obrigatória (máximo 500 caracteres)
 - Categoria opcional (máximo 100 caracteres)
 - Controle de auditoria (created_at, updated_at)

RF002 - Consolidação Automática

- Descrição: O sistema deve consolidar automaticamente os saldos diários
- Critérios de Aceite:
 - Processamento em tempo real via eventos
 - Cálculo incremental de saldos
 - Manutenção de saldo anterior como base
 - Atualização automática do cache

RF003 - Relatórios Financeiros

- Descrição: O sistema deve gerar relatórios de posição financeira
- Critérios de Aceite:
 - Saldo atual (tempo real)
 - Saldo por data específica
 - Relatório por período (máximo 365 dias)
 - Resumo com totais de créditos/débitos

3.2 Requisitos Não Funcionais Refinados

RNF001 - Disponibilidade

- Métrica: 99.5% de uptime
- Implementação:
 - Health checks automáticos
 - Restart automático de containers
 - Isolamento entre serviços

RNF002 - Performance

- Métrica: 50 req/s com máximo 5% de perda
- Implementação:
 - Cache Redis para consultas frequentes
 - Índices otimizados no banco
 - Processamento assíncrono via Kafka

RNF003 - Escalabilidade

- Métrica: Suportar 10x o volume atual
- Implementação:
 - Arquitetura stateless
 - Particionamento horizontal do banco
 - Load balancing automático

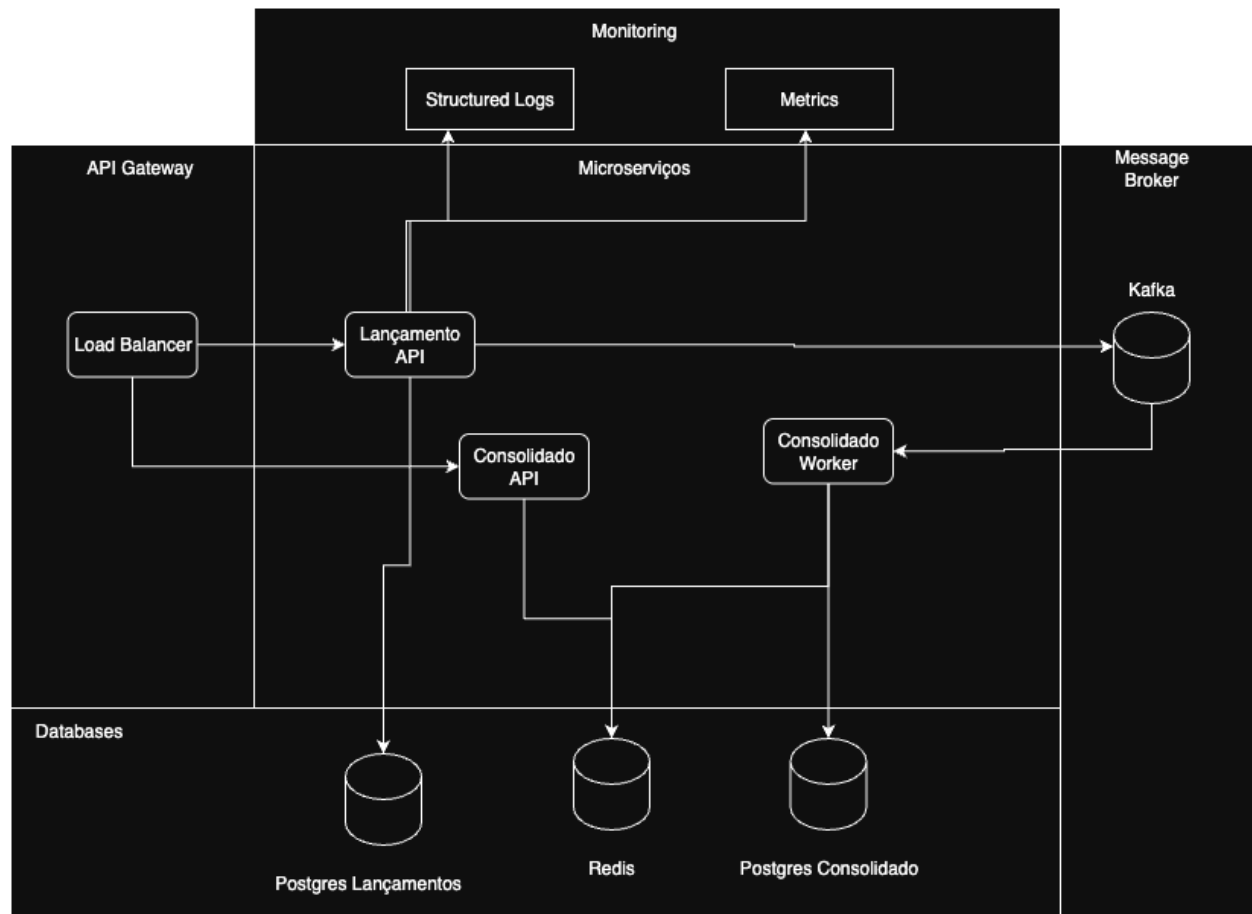
RNF004 - Resiliência

- Métrica: Serviço de lançamentos independente da consolidação
- Implementação:
 - Event-driven architecture

- Circuit breaker patterns
- Graceful degradation

4. Arquitetura Alvo

4.1 Visão Geral da Arquitetura



4.2 Padrões Arquiteturais Utilizados

Event-Driven Architecture

- Justificativa: Garantir desacoplamento e resiliência
- Implementação: Apache Kafka com eventos tipados
- Benefícios:
 - Processamento assíncrono
 - Tolerância a falhas

- Escalabilidade independente

CQRS (Command Query Responsibility Segregation)

- Justificativa: Otimização de leitura vs escrita
- Implementação:
 - Commands: Lançamentos API
 - Queries: Consolidado API
- Benefícios:
 - Performance otimizada
 - Modelos específicos por operação

Domain-Driven Design (DDD)

- Justificativa: Alinhamento com regras de negócio
- Implementação:
 - Bounded contexts bem definidos
 - Entities com invariantes
 - Value objects tipados
- Benefícios:
 - Código expressivo
 - Manutenibilidade alta

Clean Architecture

- Justificativa: Separação de responsabilidades
- Implementação:
 - Domain (regras de negócio)
 - Application (casos de uso)
 - Infrastructure (tecnologia)
 - API (interface)

4.3 Componentes Detalhados

Lançamentos API

- Tecnologia: .NET 9, ASP.NET Core
- Responsabilidades:
 - Validação de entrada (FluentValidation)
 - Processamento de comandos (MediatR)
 - Publicação de eventos (Kafka)
 - Persistência (Entity Framework + PostgreSQL)
- Endpoints:
 - `POST /api/v1/lancamentos` - Criar lançamento
 - `GET /api/v1/lancamentos/{id}` - Buscar por ID
 - `GET /api/v1/lancamentos` - Listar com paginação
 - `GET /api/v1/lancamentos/periodo` - Buscar por período
 - `PUT /api/v1/lancamentos/{id}` - Atualizar lançamento

- **DELETE /api/v1/lancamentos/{id}** - Excluir lançamento

Consolidado API

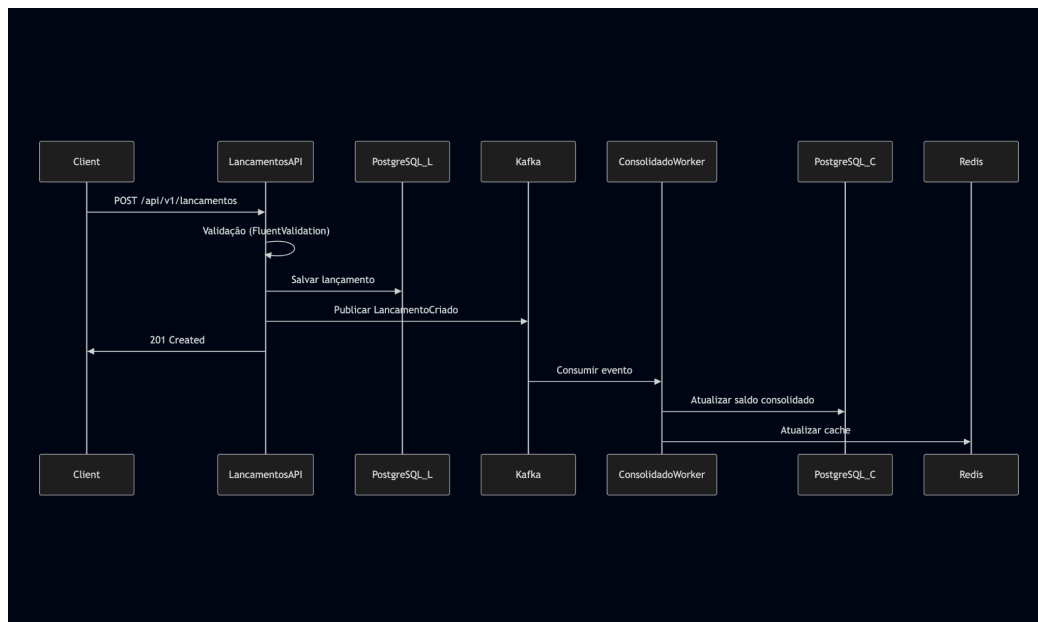
- Tecnologia: .NET 9, ASP.NET Core
- Responsabilidades:
 - Consulta de saldos consolidados
 - Cache inteligente (Redis)
 - Relatórios financeiros
- Endpoints:
 - **GET /saldo/atual** - Saldo atual
 - **GET /saldo/data/{data}** - Saldo por data
 - **GET /relatorio/periodo** - Relatório por período

Consolidado Worker

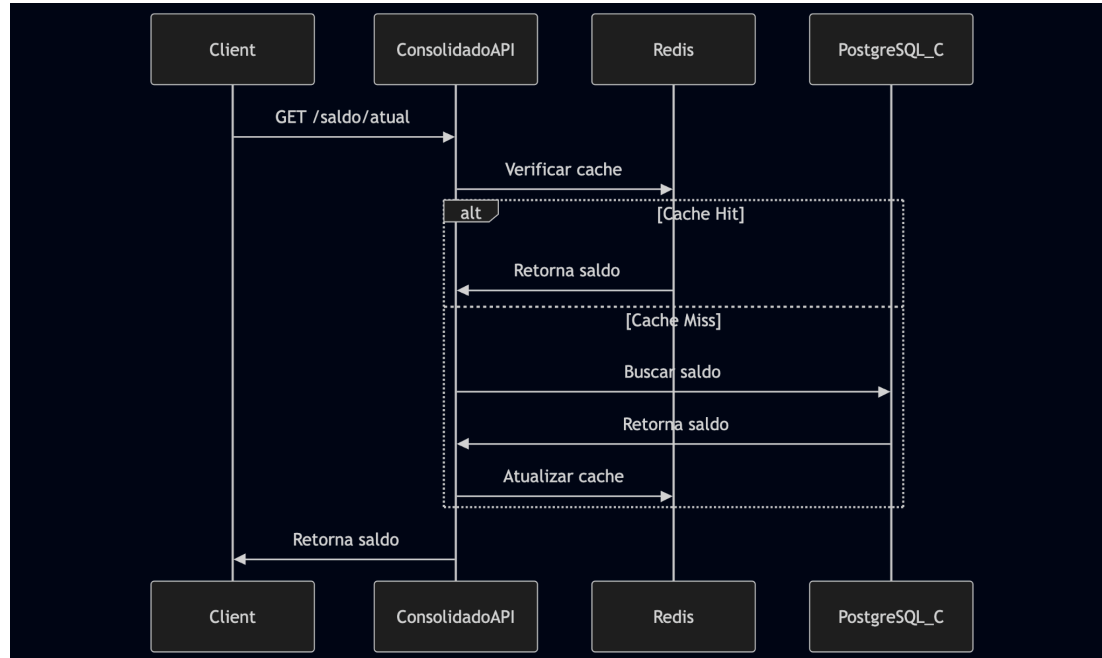
- Tecnologia: .NET 9, Background Service
- Responsabilidades:
 - Consumo de eventos Kafka
 - Processamento de consolidação
 - Atualização de cache
- Eventos Processados:
 - **LancamentoCriado**
 - **LancamentoAtualizado**
 - **LancamentoRemovido**

4.4 Fluxo de Dados

Fluxo de Criação de Lançamento



Fluxo de Consulta de Saldo



5. Justificativas Técnicas

5.1 Escolha de Tecnologias

.NET 9

- Justificativa:
 - Performance superior (AOT, minimal APIs)
 - Ecossistema maduro para microserviços
 - Suporte robusto para containerização
 - Tooling avançado para desenvolvimento
- Alternativas Consideradas: Node.js, Java Spring Boot
- Decisão: .NET oferece melhor balance entre performance e produtividade

PostgreSQL

- Justificativa:
 - ACID completo para dados financeiros
 - Suporte nativo a JSON para flexibilidade
 - Performance excelente em consultas complexas
 - Extensibilidade com índices especializados
- Alternativas Consideradas: SQL Server, MySQL
- Decisão: PostgreSQL oferece melhor custo-benefício

Apache Kafka

- Justificativa:
 - Throughput alto (50+ req/s)
 - Durabilidade garantida
 - Particionamento para escalabilidade
 - Ecossistema robusto
- Alternativas Consideradas: RabbitMQ, Azure Service Bus
- Decisão: Kafka é ideal para event-driven architecture

Redis

- Justificativa:
 - Latência ultra-baixa (<1ms)
 - Estruturas de dados especializadas
 - Persistência configurável
 - Clustering nativo
- Alternativas Consideradas: Memcached, In-memory cache
- Decisão: Redis oferece mais funcionalidades

5.2 Padrões de Integração

Event-Driven Integration

- Protocolo: Kafka com serialização JSON
- Vantagens:
 - Desacoplamento temporal
 - Tolerância a falhas
 - Replay de eventos
- Trade-offs:
 - Eventual consistency
 - Complexidade de debugging

REST APIs

- Protocolo: HTTP/HTTPS com OpenAPI
- Vantagens:
 - Padronização universal
 - Facilidade de teste
 - Documentação automática
- Trade-offs:
 - Acoplamento síncrono
 - Latência de rede

6. Requisitos Não Funcionais - Implementação

6.1 Escalabilidade

- Dimensionamento Horizontal => Usar Keda para o Worker

Estratégias de Cache

- **L1 Cache:** In-memory caching na API
- **L2 Cache:** Redis distribuído
- **Cache Warming:** Pré-carregamento de dados críticos
- **Cache Invalidation:** Event-driven cache updates

6.2 Resiliência

- Circuit Breaker Pattern
- Health Checks

Graceful Degradation

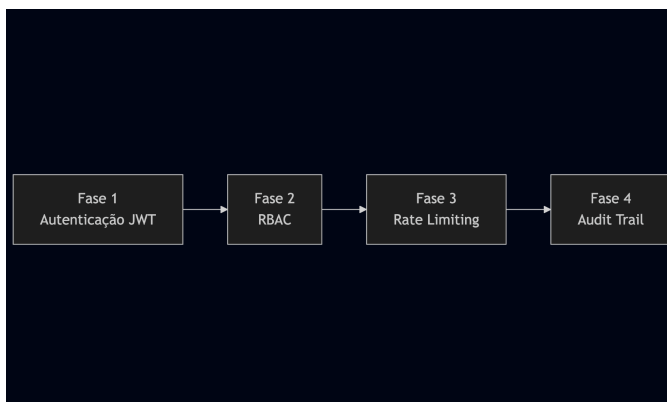
- Fallback para cache local se Redis falhar
- Mode readonly se banco principal falhar
- Retry automático com backoff exponencial

6.3 Segurança

Implementação Atual

- HTTPS enforced em produção
- Input validation com FluentValidation
- SQL injection prevention via EF Core
- Structured logging sem dados sensíveis

Roadmap de Segurança



6.4 Observabilidade

- Logging Estruturado

Métricas de Negócio

- Volume de lançamentos por hora
- Tempo médio de consolidação
- Taxa de erro por endpoint
- Utilização de cache

8. Monitoramento e Observabilidade

8.1 Estratégia de Monitoramento

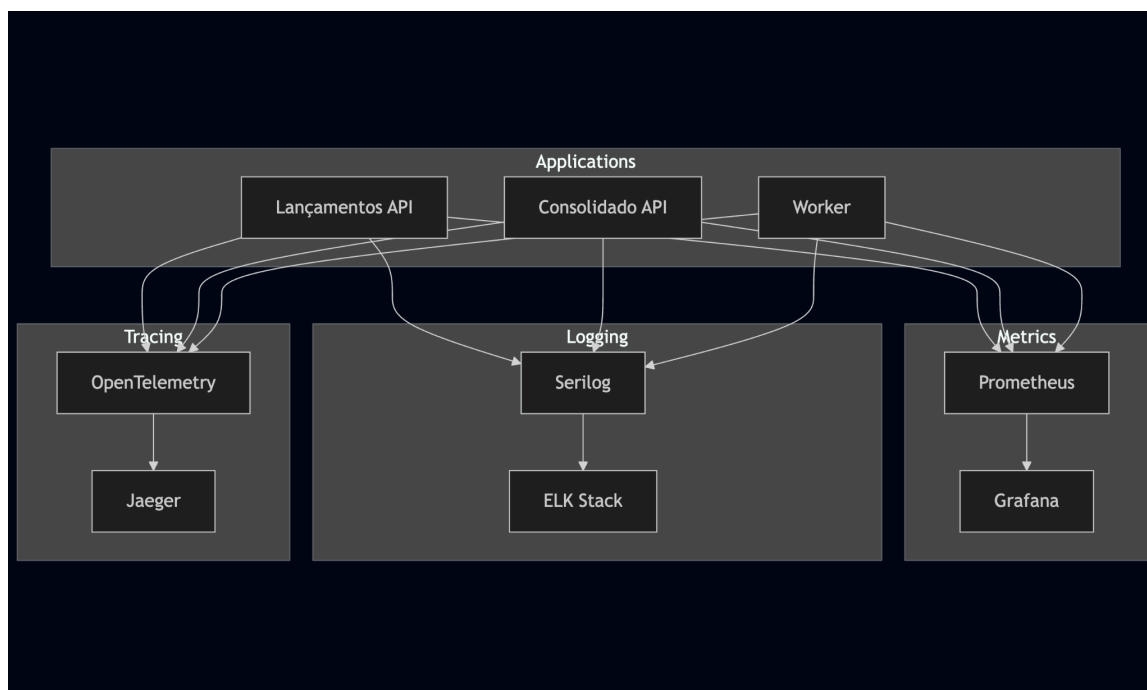
- SLIs (Service Level Indicators)

SLOs (Service Level Objectives)

- Disponibilidade: 99.5% uptime mensal
- Performance: P95 < 200ms, P99 < 500ms
- Throughput: Suportar 50 req/s sustained
- Consistência: Consolidação em < 5 segundos

8.2 Ferramentas de Observabilidade

Stack Recomendado



9. Critérios de Segurança para Integração

9.1 Autenticação e Autorização

- Implementação Futura - JWT
- RBAC (Role-Based Access Control)

9.2 Comunicação Segura

TLS/HTTPS

- Certificados SSL/TLS para todas as APIs
- mTLS para comunicação inter-serviços
- Certificate rotation automática

9.3 Segurança de Dados

Encryption at Rest

- Database: PostgreSQL com TDE (Transparent Data Encryption)
- Cache: Redis com encryption
- Logs: Encrypted storage

Encryption in Transit

- HTTPS para APIs externas
- TLS para conexões de banco
- SSL para Kafka

9.3 Segurança de Dados

Encryption at Rest

- Database: PostgreSQL com TDE (Transparent Data Encryption)
- Cache: Redis com encryption
- Logs: Encrypted storage

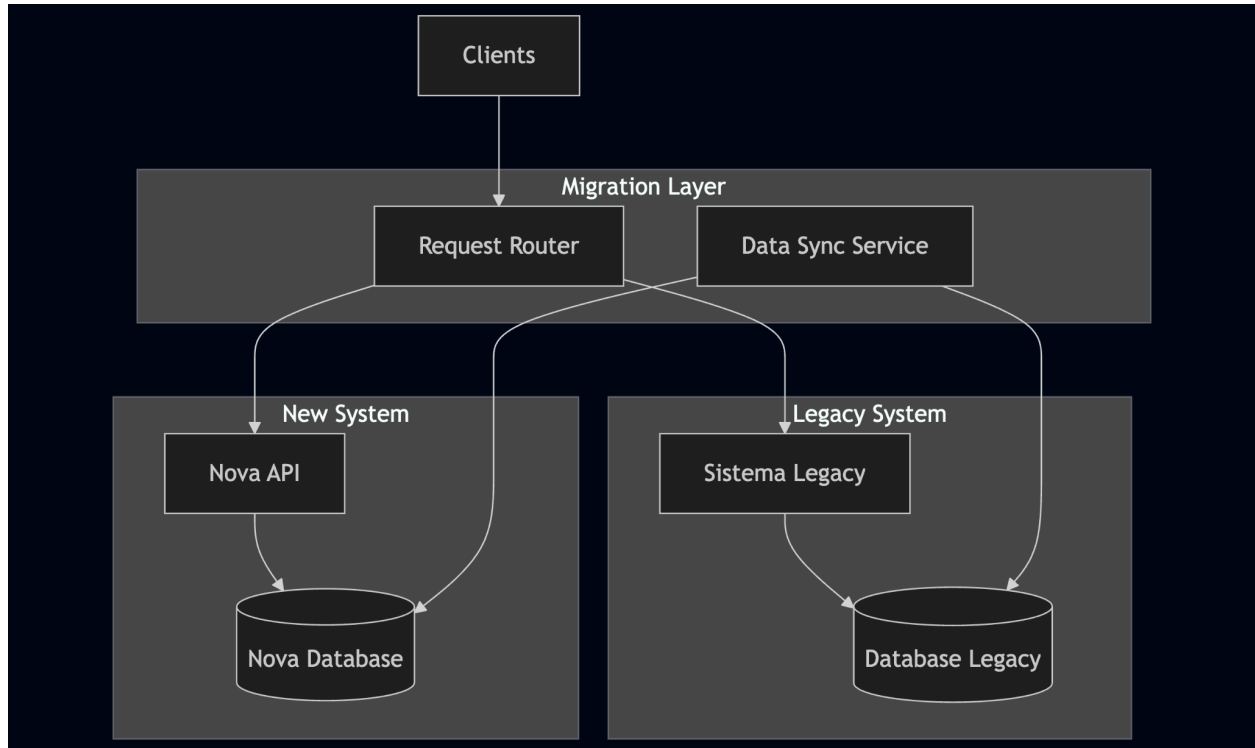
Encryption in Transit

- HTTPS para APIs externas
- TLS para conexões de banco
- SSL para Kafka

10. Arquitetura de Transição

10.1 Estratégia de Migração (Cenário Legacy)

Fase 1: Coexistência (Strangler Fig Pattern)



Fase 2: Migração Gradual

1. Semana 1-2: Deploy da nova arquitetura em paralelo
2. Semana 3-4: Migração de dados históricos
3. Semana 5-6: Redirecionamento gradual (10% → 50% → 100%)
4. Semana 7-8: Desativação do sistema legacy





Fase 3: Otimização

- Performance tuning baseado em dados reais
- Ajuste de capacidade
- Implementação de features avançadas





11. Evoluções Futuras

11.1 Roadmap Técnico





Trimestre 1: Fundação

-  Implementação core dos microserviços
-  Event-driven architecture
-  Containerização com Docker
-  Testes automatizados completos





Trimestre 2: Produção

-  Autenticação e autorização (JWT)
-  Rate limiting
-  Circuit breakers
-  Distributed tracing

Trimestre 3: Escalabilidade

-  Kubernetes orchestration
-  Auto-scaling baseado em métricas
-  Database sharding
-  CDN para assets estáticos

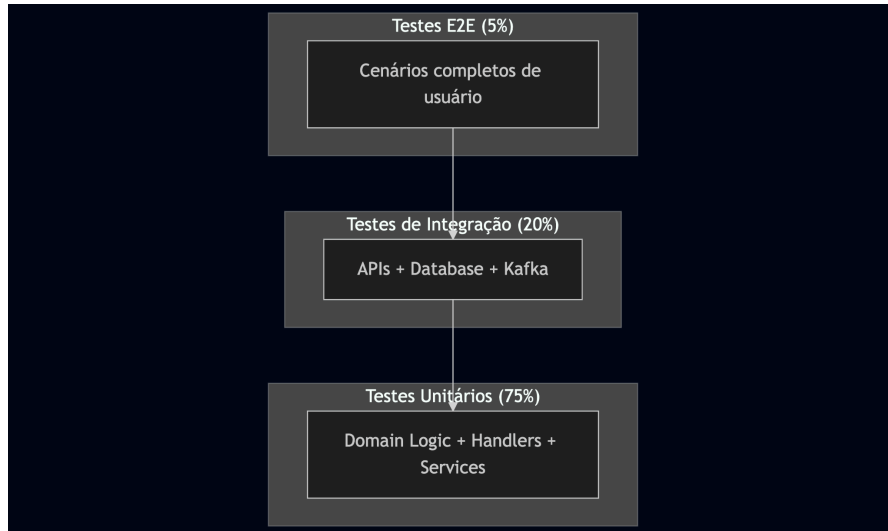
Trimestre 4: Inteligência

-  Machine Learning para detecção de anomalias
-  Predição de fluxo de caixa
-  Analytics avançados
-  Dashboards executivos

12. Testes e Qualidade

12.1 Estratégia de Testes

Pirâmide de Testes



13. Estimativa de Custos

13.1 Infraestrutura Cloud (AWS)

Ambiente de Produção

Compute - Amazon EKS

- EKS Control Plane: 1 cluster gerenciado = \$73/mês
- Worker Nodes t3.medium: 3 nodes (2 vCPU, 4GB RAM cada) = \$95/mês
- Worker Nodes t3.small: 2 nodes para workers (1 vCPU, 2GB RAM cada) = \$30/mês
- EKS Fargate: Containers serverless pay-per-use = \$25/mês
- **Subtotal Compute: \$223/mês**

Database - Amazon RDS

- RDS PostgreSQL Multi-AZ: 2 instâncias db.t3.medium (2 vCPU, 4GB cada) = \$130/mês
- Storage GP3: 100GB + backups (200GB total) = \$25/mês
- Read Replica: 1 instância db.t3.small = \$32/mês
- Automated Backups: 7 dias de retenção (incluído) = \$0/mês
- **Subtotal Database: \$187/mês**

Cache - Amazon ElastiCache

- ElastiCache for Redis: 2 nodes cache.t3.medium (alta disponibilidade) = \$85/mês
- Backup Storage: Snapshots automatizados (5GB) = \$2/mês
- **Subtotal Cache: \$87/mês**

Messaging - Amazon MSK (Managed Streaming for Kafka)

- MSK Cluster: 3 brokers kafka.t3.small = \$180/mês
- Storage: GP3 100GB por broker (300GB total) = \$30/mês
- Data Transfer: Dentro da AZ (500GB) = \$25/mês
- **Subtotal Messaging: \$235/mês**

Load Balancing e Networking

- Application Load Balancer: ALB com terminação SSL = \$23/mês
- Network Load Balancer: Para acesso ao Kafka = \$23/mês
- NAT Gateway: Para subnets privadas (2 AZs) = \$64/mês
- Data Transfer Out: Tráfego para internet (200GB) = \$18/mês
- VPC Endpoints: S3 e ECR (2 endpoints) = \$14/mês
- **Subtotal Networking: \$142/mês**

Storage e Container Registry

- Amazon ECR: Registry de containers (10GB) = \$1/mês
- Amazon S3: Logs e backups (50GB) = \$1.50/mês
- EBS Volumes: Storage dos nodes (300GB) = \$30/mês
- **Subtotal Storage: \$32.50/mês**

Monitoramento e Observabilidade

- CloudWatch Logs: Logs de aplicação (10GB ingestão) = \$5/mês
- CloudWatch Metrics: Métricas customizadas (1000 métricas) = \$3/mês
- CloudWatch Alarms: Alertas (50 alarmes) = \$5/mês
- AWS X-Ray: Distributed tracing (1M traces) = \$5/mês
- Prometheus on EKS: Self-managed (incluído no compute) = \$0/mês
- **Subtotal Monitoring: \$18/mês**

Segurança

- AWS Certificate Manager: Certificados SSL (gratuito) = \$0/mês
- AWS Secrets Manager: Credenciais do banco (5 secrets) = \$2.50/mês
- AWS WAF: Web application firewall (1 web ACL) = \$12/mês
- GuardDuty: Detecção de ameaças por conta = \$30/mês
- **Subtotal Security: \$44.50/mês**