Registro de Decisões Arquiteturais (ADR)

ADR 001: Seleção da Arquitetura Escalável e Resiliente

Contexto

O objetivo é desenvolver um sistema de controle de caixa que suporte alta disponibilidade, segurança e desempenho, considerando:

- Escalabilidade para lidar com aumento de carga.
- Resiliência para garantir recuperação de falhas.
- Integração segura e eficiente entre componentes.
- Padrões arquiteturais que equilibrem simplicidade e flexibilidade.

Decisão

Adotar uma arquitetura baseada em microsserviços, combinada com o padrão CQRS (Command Query Responsibility Segregation) para garantir modularidade, escalabilidade e separação clara de responsabilidades. As principais escolhas foram estruturadas da seguinte forma:

1. **Escalabilidade**:
    - Dimensionamento horizontal por serviço.
    - Uso de balanceadores de carga para distribuição de requisições.
    - Estratégias de cache em diferentes camadas (e.g., CDN e cache local).
2. **Resiliência**:
    - Implementar redundância de serviços críticos.
    - Configurar mecanismos de failover automáticos.
    - Monitoramento proativo com alertas e logging centralizado.
3. **Segurança**:
    - Uso de autenticação baseada em tokens (e.g., OAuth2).
    - Criptografia de dados em trânsito (TLS) e em repouso (AES-256).
    - Implementar WAF (Web Application Firewall) para proteção contra ataques.
4. **Integração**:
    - Protocolo REST para comunicação entre serviços, com suporte a JSON como formato padrão.
    - Uso de filas de mensagens (e.g., RabbitMQ ou Kafka) para comunicação assíncrona.
5. **Uso de CQRS**:
    - Separar responsabilidades de leitura e escrita nos serviços, otimizando cada operação para seu objetivo.

    - Operações de escrita serão realizadas por comandos que modificam o estado do sistema.
    - Operações de leitura serão realizadas por consultas otimizadas para desempenho, utilizando bancos de dados especializados quando necessário.
    - Essa combinação assegura que o sistema possa atender demandas altas e reduz conflitos entre operações de leitura e escrita.
6. **Requisitos Não-Funcionais**:
    - O serviço de controle de lançamento não deve ficar indisponível se o sistema de consolidado diário cair.
    - Em dias de pico, o serviço de consolidado diário deve suportar até 50 requisições por segundo, com no máximo 5% de perda de requisições.
7. **Padrões Arquiteturais**:
    - **Domain-Driven Design (DDD)**: Organizar o sistema em torno de contextos delimitados para alinhar requisitos de negócios à arquitetura.
    - **Service Mesh**: Gerenciar comunicação segura e resiliente entre microsserviços, incluindo balanceamento de carga e autenticação mútua (mTLS).
    - **API Gateway**: Centralizar a entrada de requisições nos microsserviços, agregando autenticação, autorização e rate limiting.
    - **Circuit Breaker**: Prevenir falhas em cascata e aumentar a resiliência com circuitos que isolam falhas.

Consequências

- **Benefícios**:
    - Facilita o aumento de capacidade conforme necessidade.
    - Melhora a resiliência do sistema, reduzindo o impacto de falhas.
    - Aumenta a segurança ao implementar práticas modernas de proteção.
    - O uso combinado de microsserviços e CQRS melhora a organização e a eficiência do sistema.
- **Riscos**:
    - Maior complexidade de gerência e orquestração de serviços.
    - Necessidade de profissionais especializados para manutenção da infraestrutura.
    - O uso de CQRS pode aumentar a complexidade no design inicial e exigir ferramentas adicionais para sincronização de dados.