




ADR: Arquitetura do User Service

 **Data:** 22/02/2025
 **Status:** Aprovado
 **Autor:** Clever Santoro Lopes

1. Contexto

O **User Service** é responsável por gerenciar os dados de usuários na plataforma de e-commerce. Ele deve garantir:

- ☒ **Cadastro, atualização e exclusão de usuários.**
- ☒ **Gerenciamento de perfis e permissões.**
- ☒ **Integração com o Auth Service para autenticação.**
- ☒ **Alta disponibilidade e segurança dos dados.**

2. Decisão

Optamos por um **serviço independente para gerenciamento de usuários**, garantindo escalabilidade e segurança.

Tecnologias Escolhidas

Componente	Tecnologia	Justificativa
Linguagem	Node.js (NestJS) ou GoLang	Performance e suporte a APIs
Banco de Dados	PostgreSQL	Armazena usuários e permissões
Cache	Redis	Armazena dados temporários e sessões
Autenticação	Integração com Auth Service	Segurança centralizada
Mensageria	Kafka / RabbitMQ	Notificação de eventos de usuários
Monitoramento	Prometheus + Grafana	Logs e métricas

Arquitetura do User Service

- 1 API Gateway** → Direciona requisições para o User Service.
- 2 User Service** → Gerencia operações de CRUD de usuários.
- 3 Banco de Dados** → Armazena perfis e permissões.
- 4 Integração com Auth Service** → Autenticação centralizada.
- 5 Mensageria** → Envia eventos de alteração de usuários para outros serviços.

3. Alternativas Consideradas

3.1 Armazenar Usuários no Auth Service

🚫 **Rejeitado** – Aumenta acoplamento entre serviços e dificulta evolução independente.

3.2 Monólito com Usuários e Autenticação juntos

🚫 **Rejeitado** – Menos escalável e limita flexibilidade para futuros ajustes.

3.3 User Service Independente

✅ **Aprovado** – Permite escalabilidade e integração modular com outros serviços.

4. Consequências

✅ Benefícios

- ✓ **Modularidade** – Evolução independente do Auth Service.
- ✓ **Escalabilidade** – Capacidade de distribuir carga entre instâncias.
- ✓ **Segurança** – Proteção de dados de usuários separada da autenticação.
- ✓ **Integração facilitada** – Notificação de eventos via mensageria.

⚠️ Desafios

- ⚠️ **Sincronização com Auth Service** → Necessário garantir consistência de dados.
 - ⚠️ **Gerenciamento de Permissões** → Pode aumentar a complexidade inicial da implementação.
-

5. Próximos Passos

- 🚀 Implementação de **controle de acesso granular (RBAC/ABAC)**.
 - 🔍 **Monitoramento** com OpenTelemetry para rastrear acessos e alterações de usuários.
 - 📊 **Testes de carga e segurança** para validar resiliência e performance.
-

🎯 Conclusão

O **User Service** fornecerá um gerenciamento de usuários seguro, escalável e modular, garantindo flexibilidade na evolução da plataforma.