Sistema Distribuído para Gerenciamento de Medicamentos usando gRPC

Conheça uma solução cliente-servidor robusta para controle remoto eficiente de estoque.

A arquitetura clara garante escalabilidade e segurança no gerenciamento de medicamentos.

Apresentação feita por Thiago Medeiros e Carlos Eduardo

Arquitetura Cliente-Servidor do Sistema

Cliente

Interface web em Node.js/Express acessível e responsiva.

Servidor

Java Spring Boot lida com lógica e persistência.

Comunicação

Stubs gRPC para comunicação remota transparente.

Comunicação Eficiente via RPC/gRPC

Alta Eficiência

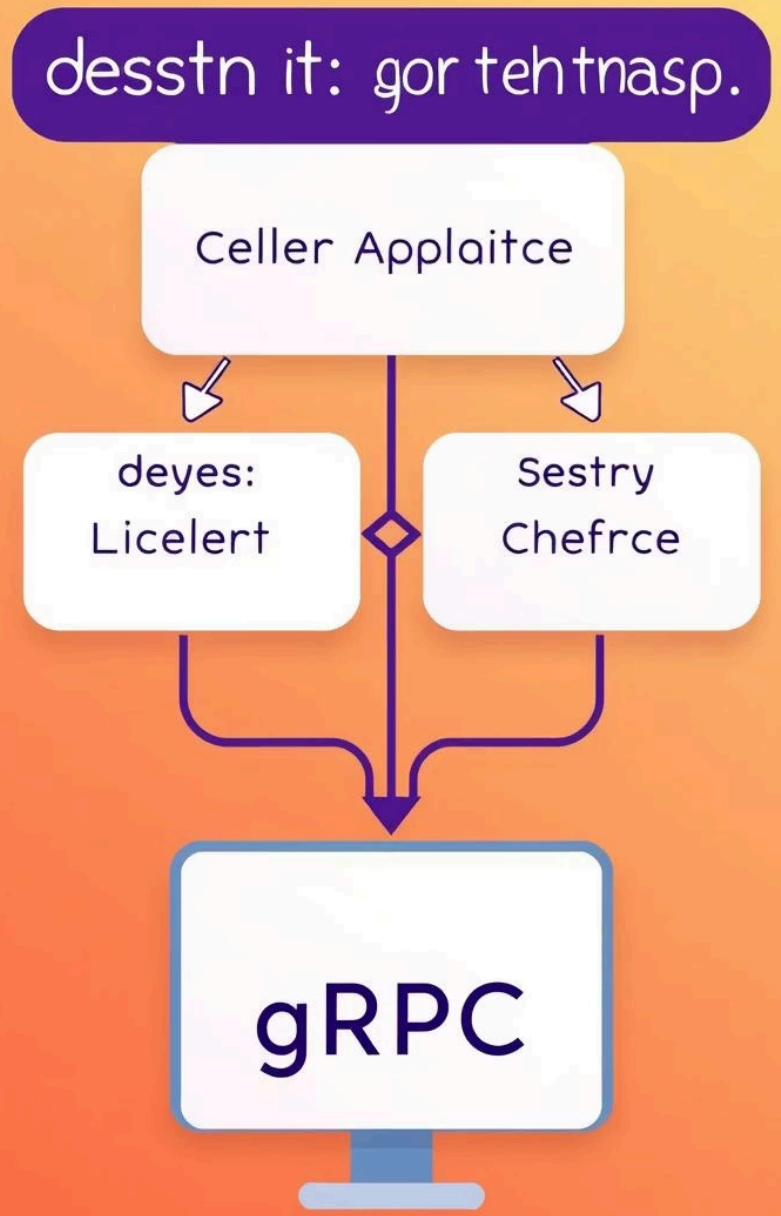
Redução da latência em até 30% comparado ao REST.

Protocol Buffers

Serialização leve e rápida para troca ágil de dados.

Segurança

Criptografia TLS integrada para proteção dos dados.



Função dos Stubs no gRPC

Invocação Remota Simples

Permitem chamar métodos remotos como se fossem locais.

Redução de Código Boilerplate

Economia de 40% no código repetitivo manual.

Geração Automática

Gerados automaticamente a partir de arquivos .proto.

Fluxo Detalhado das Requisições

Fluxo Transparente

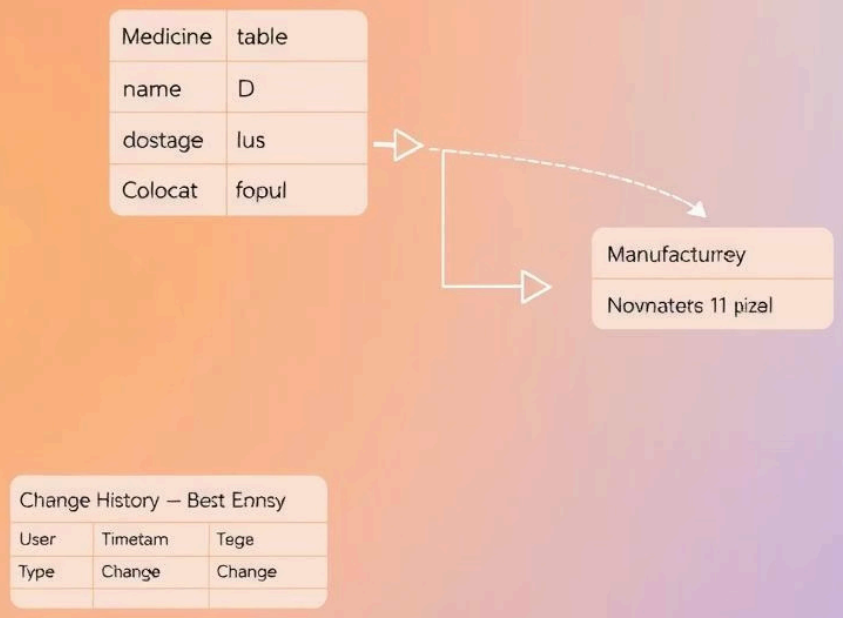
Requisição via REST é convertida para gRPC no cliente.

Serialização Automática

Protocol Buffers cuidam da eficiência no transporte.

Integridade Garantida

Dados retornam corretos e seguros para o usuário.



Modelo de Dados do Sistema

Tabela Remédio

Campos: id, nome, via, lote, validade, quantidade, laboratório, ativo

Tabela Histórico

Registro completo de alterações para auditoria.

Benefícios Técnicos do RPC/gRPC

Comunicação Eficiente

Baixa latência e alto desempenho via Protocol Buffers.

Interoperabilidade

Sistemas Java e Node.js comunicam com transparência.

Automação com Stubs

Código gerado automaticamente reduz erros e acelera o desenvolvimento.

Concorrência e Escalabilidade

Servidor lida com múltiplas conexões usando threads.

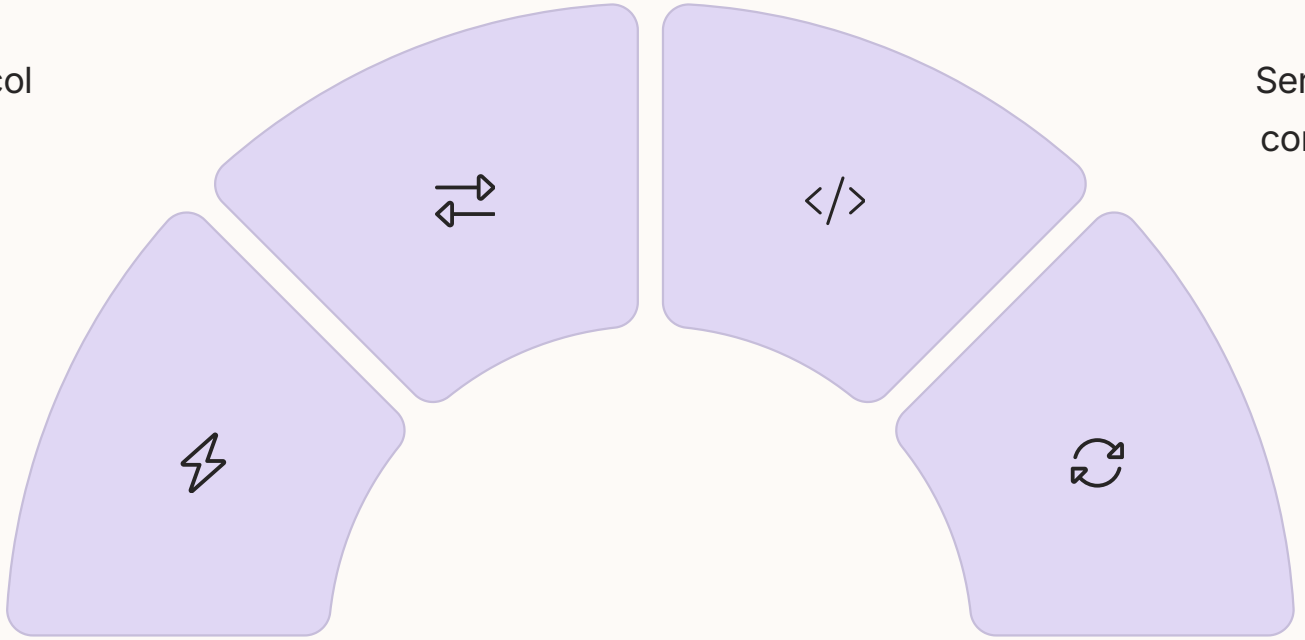
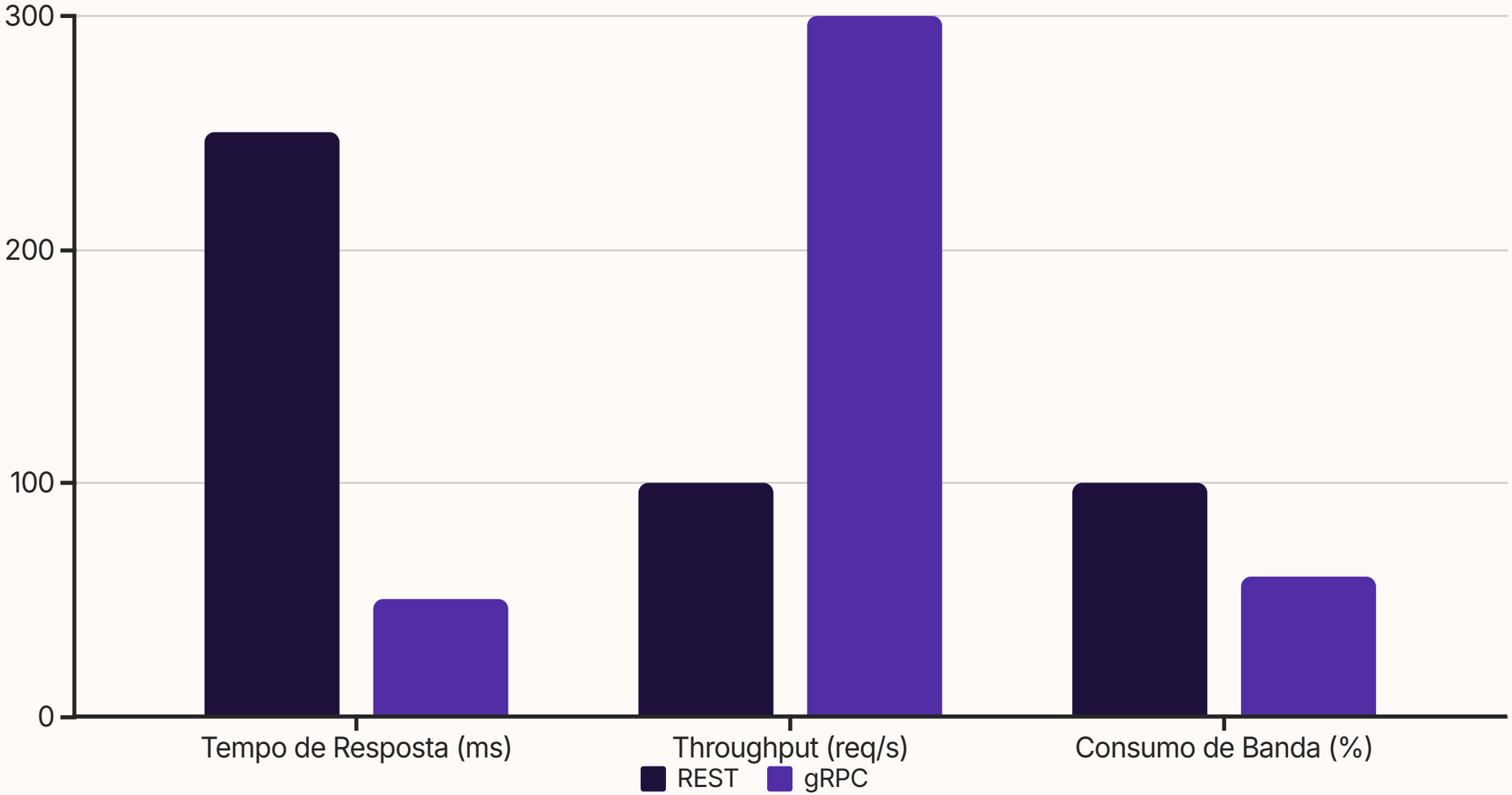


Gráfico Comparativo REST vs gRPC



Sistemas Distribuídos na Prática

Arquitetura Cliente-Servidor

Responsabilidades claras e escalabilidade horizontal.

Comunicação via gRPC

Middleware leve com Protocol Buffers eficiente.

Processos e Threads

Servidor gerencia múltiplas requisições paralelas.

Nomeação e Endereçamento

Serviços localizados com endereços de rede fixos.

Tolerância a Falhas

Tratamento de exceções e persistência garantem continuidade.

Dúvidas, Sugestões ou Comentários?

Fluxo Principal:

1. Usuário Web via REST
2. Cliente Express + Stub gRPC via gRPC
3. Servidor Java/Spring Boot processa via JDBC
4. MySQL armazena dados persistentes

- **RPC Transparente**
- TLS: Segurança de comunicação
- Protocol Buffers: Performance

O sistema foi projetado para comunicação eficiente, concorrência e escalabilidade.

Alguém tem perguntas ou sugestões?