**GABRIEL FERNANDES MOURA DOS SANTOS**

ENGENHARIA DE SOFTWARE

**DOCUMENTO DE ESPECIFICAÇÃO DE REQUISITOS TÉCNICOS**

**API MICROSERVIÇOS FINANCEIRA COM BLOCKCHAIN, REDIS, MÉTRICAS PROMETHEUS E MENSAGERIA ASSÍCRONA**

Versão: 1.0

26 de Junho de 2025

Rio de Janeiro

**DESCRIÇÃO DO PROJETO**  
Este documento detalha o desenvolvimento da **API Financeira Integrada**, uma aplicação backend (parte do sistema que roda no servidor, responsável pela lógica e dados) de alta complexidade voltada para o setor financeiro, que incorpora tecnologias emergentes e consolidadas como Blockchain (tecnologia para registro seguro e imutável de transações), WebSocket (protocolo para comunicação em tempo real entre servidor e cliente), RabbitMQ (ferramenta para gerenciar filas de mensagens e tarefas assíncronas), Redis (banco de dados rápido usado para cache e controle de sessões), Prometheus (sistema para coleta e monitoramento de métricas) e Grafana (ferramenta para visualização gráfica dessas métricas). O projeto foi concebido com base em estudos aprofundados das necessidades atuais do mercado financeiro, visando garantir segurança, escalabilidade (capacidade de crescer sem perder performance) e alta performance (rapidez e eficiência).

A API implementa autenticação e autorização robustas via Spring Security (framework para segurança em aplicações Java) e JWT (JSON Web Token, método para garantir identidade e permissões do usuário), garantindo controle de acesso granular e proteção contra ameaças. Utiliza Blockchain para registro imutável e transparente das transações financeiras, assegurando integridade (garantia que os dados não foram alterados) e rastreabilidade (possibilidade de acompanhar o histórico) dos dados.

Para garantir respostas em tempo real e interação dinâmica com os usuários, integra WebSocket, enquanto RabbitMQ possibilita o processamento assíncrono (tarefas que ocorrem em segundo plano sem bloquear o sistema) e desacoplado de eventos, incluindo envio de e-mails e notificações.

A utilização de Redis permite cache eficiente (armazenamento temporário para acelerar acessos) e controle de sessões (manutenção do estado do usuário logado), além de suportar mecanismos avançados como blacklist de tokens (lista de tokens inválidos para impedir acessos não autorizados) e rate limiting (limitação de número de requisições para evitar abusos). Métricas detalhadas sobre desempenho e saúde do sistema são coletadas via Micrometer (biblioteca para monitoramento em Java) e expostas através do Actuator (componente do Spring Boot que fornece endpoints para gerenciamento da aplicação), com monitoramento contínuo por Prometheus e visualização em dashboards Grafana.

Este documento técnico apresenta a arquitetura da aplicação, especificação das rotas da API, integração das tecnologias, fluxo de dados, bem como estratégias de testes automatizados e manuais para garantir qualidade e confiabilidade do sistema. O objetivo é fornecer uma base sólida para desenvolvimento, análise e futuras melhorias, fundamentada em pesquisas técnicas e melhores práticas do mercado.

**DIAGRAMA DE ARQUITETURA**

**FLUXO DE ROTAS**

**MODELO DE DADOS**

**DIGRAMA DE SEQUENCIA**

**PLANO DE TESTES**

**LISTA DE ENDPOINTS**

**1. Autenticação e Usuário (10 rotas)**

1. POST /api/auth/register — Registrar novo usuário  
   (Spring MVC, Spring Data JPA, Bean Validation, Spring Security PasswordEncoder, RabbitMQ, Swagger, Redis opcional)
2. POST /api/auth/login — Login, retorno do JWT  
   (Spring MVC, Spring Security, JWT, Redis opcional, Swagger)
3. POST /api/auth/logout — Logout, invalidar token  
   (Spring MVC, Spring Security, Redis, JWT, Swagger)
4. POST /api/auth/refresh-token — Atualizar token JWT  
   (Spring MVC, Spring Security, JWT, Redis opcional, Swagger)
5. POST /api/auth/forgot-password — Solicitar reset de senha  
   (Spring MVC, Spring Data JPA, JavaMailSender, RabbitMQ, Redis, Swagger)
6. POST /api/auth/reset-password — Resetar senha com token  
   (Spring MVC, Redis, Spring Data JPA, Spring Security PasswordEncoder, Swagger)
7. GET /api/users/me — Perfil do usuário logado  
   (Spring MVC, Spring Security, Spring Data JPA, Swagger)
8. PUT /api/users/me — Atualizar perfil  
   (Spring MVC, Spring Security, Bean Validation, Spring Data JPA, Swagger)
9. GET /api/users/{id} — Obter perfil público de outro usuário  
   (Spring MVC, Spring Security (ou público), Spring Data JPA, DTOs, Swagger)
10. DELETE /api/users/me — Deletar própria conta  
    (Spring MVC, Spring Security, Spring Data JPA, RabbitMQ opcional, Swagger)

**2. Gestão de Papéis, Permissões e Segurança (5 rotas)**

1. GET /api/roles — Listar todos papéis  
   (Spring MVC, Spring Security (role-based access), Spring Data JPA, Swagger)
2. POST /api/roles — Criar papel novo (admin)  
   (Spring MVC, Spring Security (authorization/admin), Spring Data JPA, Bean Validation, Swagger)
3. PUT /api/users/{id}/roles — Atualizar papéis do usuário  
   (Spring MVC, Spring Security (authorization/admin), Spring Data JPA, Swagger)
4. GET /api/users/{id}/roles — Ver papéis de um usuário  
   (Spring MVC, Spring Security (authorization/admin), Spring Data JPA, Swagger)
5. GET /api/security/audit-logs — Logs de ações (admin)  
   (Spring MVC, Spring Security (authorization/admin), Spring Data JPA, Event Listener (para logs), Swagger)

**3. Blockchain e Transações (8 rotas)**

1. POST /api/blockchain/transactions — Enviar nova transação  
   (Spring MVC, Spring Security (JWT), Blockchain Core, Spring Data JPA, RabbitMQ (eventos para processamento assíncrono), Swagger)
2. GET /api/blockchain/blocks — Listar blocos  
   (Spring MVC, Spring Security (JWT), Blockchain Core, Spring Data JPA, Swagger)
3. GET /api/blockchain/blocks/{blockId} — Detalhes de bloco  
   (Spring MVC, Spring Security (JWT), Blockchain Core, Spring Data JPA, Swagger)
4. GET /api/blockchain/transactions/{txId} — Detalhes da transação  
   (Spring MVC, Spring Security (JWT), Blockchain Core, Spring Data JPA, Swagger)
5. GET /api/blockchain/transactions/user/{userId} — Histórico de um usuário  
   (Spring MVC, Spring Security (JWT), Blockchain Core, Spring Data JPA, Swagger)
6. GET /api/blockchain/pending-transactions — Transações pendentes  
   (Spring MVC, Spring Security (JWT), Blockchain Core, Cache Redis, Swagger)
7. POST /api/blockchain/validate — Validar bloco/transação (interno)  
   (Spring MVC, Blockchain Core, RabbitMQ (para orquestração eventos), Spring Security (interno), Swagger)
8. GET /api/blockchain/status — Status da blockchain (altura, consenso)  
   (Spring MVC, Spring Security (JWT), Blockchain Core, Swagger)

**4. Notificações e Eventos (5 rotas)**

1. GET /api/notifications — Listar notificações do usuário  
   (Spring MVC, Spring Security (JWT), Spring Data JPA, Redis (cache), Swagger)
2. POST /api/notifications/mark-read — Marcar notificações como lidas  
   (Spring MVC, Spring Security (JWT), Spring Data JPA, Redis (cache), Swagger, RabbitMQ (eventos para atualizar notificações))
3. POST /api/notifications — Criar notificação manual (admin)  
   (Spring MVC, Spring Security (roles/authorities), Spring Data JPA, RabbitMQ (para envio assíncrono), Swagger)
4. GET /api/notifications/unread-count — Contagem notificações não lidas  
   (Spring MVC, Spring Security (JWT), Spring Data JPA, Redis (cache), Swagger)
5. DELETE /api/notifications/{id} — Remover notificação  
   (Spring MVC, Spring Security (JWT), Spring Data JPA, Swagger)

**5. Emails e Comunicação (4 rotas)**

1. POST /api/email/send — Enviar email (suporte, marketing)  
       (Spring MVC, Spring Security (JWT/roles), JavaMailSender, RabbitMQ (para envio assíncrono), Swagger)
2. GET /api/email/templates — Listar templates de email  
       (Spring MVC, Spring Security (JWT/roles), Spring Data JPA, Swagger)
3. POST /api/email/templates — Criar/editar template  
       (Spring MVC, Spring Security (roles/admin), Spring Data JPA, Swagger)
4. DELETE /api/email/templates/{id} — Deletar template  
       (Spring MVC, Spring Security (roles/admin), Spring Data JPA, Swagger)

**6. Administração e Dashboard (6 rotas)**

1. GET /api/admin/users — Listar usuários com filtros e paginação  
       (Spring MVC, Spring Security (roles/admin), Spring Data JPA, Swagger, Pagination, Metrics)
2. DELETE /api/admin/users/{id} — Deletar usuário (admin)  
       (Spring MVC, Spring Security (roles/admin), Spring Data JPA, Swagger)
3. GET /api/admin/metrics — Dados de uso, erros, performance  
       (Spring MVC, Spring Security (roles/admin), Micrometer/Prometheus, Swagger, Metrics)
4. GET /api/admin/logs — Acesso a logs para auditoria  
       (Spring MVC, Spring Security (roles/admin), Logback/ELK Stack, Swagger)
5. GET /api/admin/settings — Configurações do sistema  
       (Spring MVC, Spring Security (roles/admin), Spring Data JPA, Swagger)
6. PUT /api/admin/settings — Atualizar configurações  
       (Spring MVC, Spring Security (roles/admin), Spring Data JPA, Swagger)

**7. WebSocket / Tempo Real (endpoint + tópicos)**

1. /ws — Endpoint para conexão WebSocket  
       (Spring WebSocket, STOMP, Spring Security (autenticação WebSocket), Metrics)
2. /topic/transactions — Notificações de novas transações (broadcast)  
       (Spring WebSocket, STOMP, Redis Pub/Sub para escalabilidade)
3. /topic/blocks — Notificações de novos blocos (broadcast)  
       (Spring WebSocket, STOMP, Redis Pub/Sub)
4. /user/queue/notifications — Notificações pessoais em tempo real (unicast)  
       (Spring WebSocket, STOMP, Spring Security para identificação do usuário, Redis para escalabilidade)

**8. Cache, Métricas e Suporte (4 rotas)**

1. GET /api/cache/clear — Limpar cache Redis (admin)  
       (Spring Cache com Redis, Spring Security (autorização), Redis)
2. GET /api/metrics — Expor métricas customizadas  
       (Micrometer, Spring Actuator, Prometheus, Grafana)
3. GET /api/support/tickets — Listar tickets de suporte  
       (Spring Data JPA, PostgreSQL, Spring Security)
4. POST /api/support/tickets — Criar ticket  
       (Spring Data JPA, PostgreSQL, RabbitMQ para processamento assíncrono, Spring Security)

**Tecnologias integradas**

* **Spring Security + JWT:** autenticação, autorização, papéis, segurança
* **Redis:** blacklist tokens, cache, sessões, rate limiting
* **RabbitMQ:** fila de eventos, envio de email, processamento assíncrono
* **Spring WebSocket:** notificações em tempo real para usuário e admin
* **JavaMailSender:** envio de email (recuperação de senha, notificações)
* **Micrometer + Actuator:** métricas, health checks, dados de performance
* **Prometheus:** coleta e armazenamento de métricas (integra-se com Micrometer)
* **Grafana:** dashboards visuais para métricas coletadas (Prometheus e outras fontes)
* **Swagger/OpenAPI:** documentação interativa das rotas
* **Testes via Postman:** testar todas as rotas com dados reais

**MONITORAMENTO E LOGS**

**ROADMAP E PROXIMOS ETAPAS**