API de Finanças e Investimentos

Um Sistema de Gestão de Carteira e Integração de Mercado

AUTORES

Miguel Antônio Gregório Firme

Lorenzo Viero Sartori

O Problema e a Solução

A necessidade de acompanhar investimentos em tempo real

PROBLEMA

Dificuldade em gerenciar finanças pessoais e acompanhar a performance de ativos voláteis do mercado.

SOLUÇÃO

Desenvolvimento de uma API RESTful para simular um serviço de gestão de investimentos com dados reais de mercado.

OBJETIVO CENTRAL

Aplicar conceitos de desenvolvimento Backend (Spring Boot, JPA, Segurança) e integrar com dados reais de mercado.

Stack Tecnológico

Java, Spring Boot e MySQL

LINGUAGEM
Java 17

FRAMEWORK
Spring Boot 3.x

PERSISTÊNCIA (DB)
MySQL

MAPEAMENTO
Spring Data JPA / Hibernate

INTEGRAÇÃO EXTERNA
OpenFeign / RestTemplate

SEGURANÇA
Spring Security

Arquitetura de Dados

Entidades e Relacionamentos Chave

USUARIO

ID (UUID), Nome, Email, Senha

1:1 com Carteira

CARTEIRA

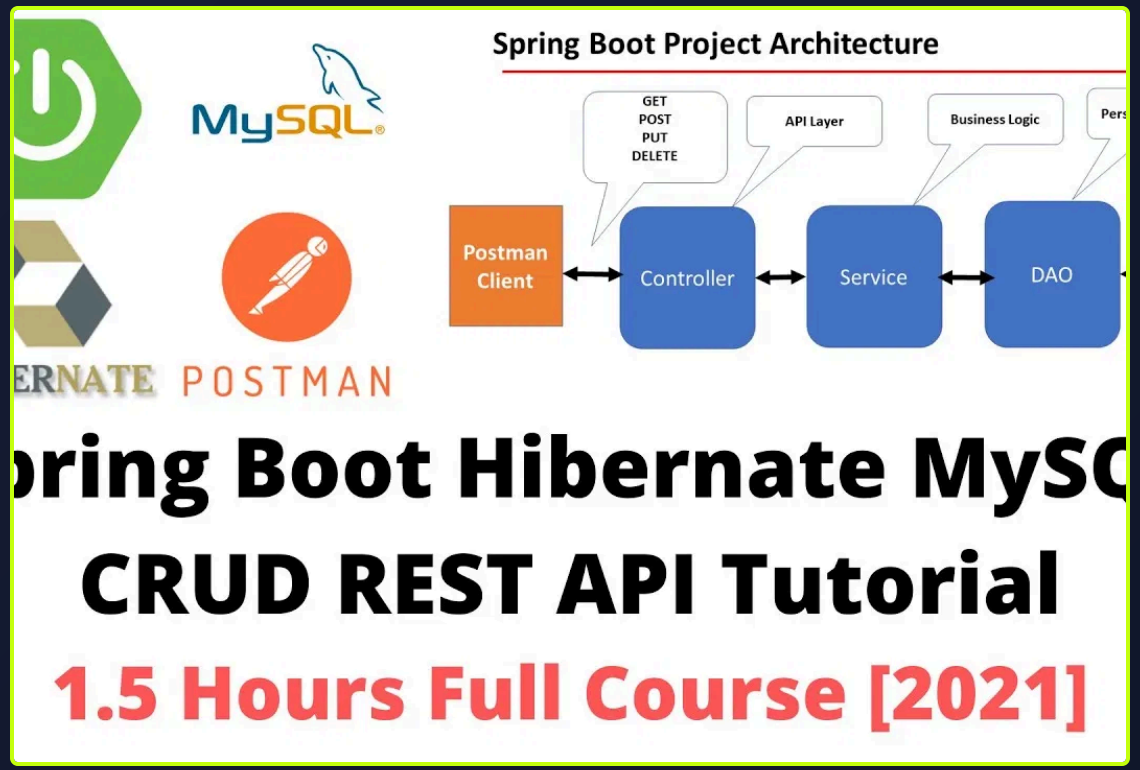
ID (UUID), Referência ao Usuario

1:N com Investimento

INVESTIMENTO

ID (Long), Ticker, Valor, Dias, Cálculos

Pertence a uma Carteira



O Coração da Carteira

A Entidade Investimento

CAMPOS ESSENCIAIS

TICKER
`String`

Código do ativo (ex: PETR4, MGLU3)

VALOR INVESTIDO
`Double`

Montante inicial investido no ativo

DIAS
`Integer`

Período de projeção em dias

CAMPOS CALCULADOS

PREÇO NO MOMENTO
`Double`

Cotação atual do ativo (API externa)

TAXA ESTIMATIVA
`Double`

Taxa diária estimada de rendimento

VALOR ESTIMADO FINAL
`Double`

Projeção do valor após o período

RENDIMENTO ESTIMADO
`Double`

Ganho estimado em reais

Camada de Serviços

O InvestimentoService

RESPONSABILIDADE PRINCIPAL

O InvestimentoService orquestra toda a lógica de negócio para criar, atualizar e deletar investimentos. É o coração da aplicação, responsável por validações, cálculos de projeção e persistência de dados.

REGRA DE VALIDAÇÃO

O valor investido deve ser maior ou igual ao preço atual do ativo:

```
if (valor < precoAtual) { throw ValidationException }
```

CÁLCULO DE PROJEÇÃO

Utiliza taxa anual para calcular rendimento composto:

```
taxaDiaria = (1 + taxaAnual)^(1/252) - 1 valorFinal = valor × (1 + taxaDiaria)^dias
```

Integração com Mercado

Conectando com o Mundo Real

SERVIÇO CHAVE

AtivoService atua como cliente da API Brapi para buscar dados reais de mercado

TECNOLOGIA

Utiliza RestTemplate do Spring para fazer requisições HTTP GET

```
GET https://brapi.dev/api/quote/{ticker}
```

DADOS OBTIDOS

Preço atual (regularMarketPrice) e taxa de rendimento (regularMarketChangePercent, dividendYield)

TRATAMENTO DE ERROS

Lança ResourceNotFoundException se o ativo não for encontrado na API

Eficiência na Gestão

Processamento de Investimentos em Lote

ENDPOINT

Método POST para upload de arquivo

```
POST /api/investimentos/usuario/{usuarioId}/upload
```

FORMATO DO ARQUIVO

Arquivo de texto (.txt) com linhas no padrão:

```
TICKER;VALOR;DIAS
```

```
PETR4;5000;30
MGLU3;1500;90
VALE3;8000;60
```

LÓGICA DE PROCESSAMENTO

- Lê o arquivo linha por linha
- Valida o formato (3 campos)
- Aplica mesma lógica de validação
- Calcula projeção de rendimento
- Retorna relatório (sucesso/erro)

RESPOSTA DA API

Retorna um resumo do upload:

- Total de registros processados
- Quantidade de sucessos
- Quantidade de erros
- Detalhes de cada erro

Segurança da API

Proteção dos Dados com Spring Security

MECANISMO DE AUTENTICAÇÃO

Autenticação **Basic Auth** (username:password em Base64)

```
.httpBasic(httpBasic -> {})
```

CONFIGURAÇÃO DO SPRING SECURITY

Habilitação de segurança com **@EnableWebSecurity** e desabilitação de CSRF

```
.csrf(csrf -> csrf.disable())
```

AUTORIZAÇÃO

Todas as requisições exigem autenticação válida

```
.anyRequest().authenticated()
```

CREDENCIAIS DE TESTE

Usuário em memória para testes e demonstração

```
user / 123456789
```

Resumo dos Endpoints

Principais Rotas da Aplicação

USUÁRIOS

POST
`/api/usuarios`
Cria um novo usuário e sua carteira

GET
`/api/usuarios`
Lista todos os usuários

GET
`/api/usuarios/{id}`
Busca um usuário específico

DELETE
`/api/usuarios/{id}`
Deleta um usuário

INVESTIMENTOS

POST
`/api/investimentos/usuario/{id}`
Cria um investimento

GET
`/api/investimentos/usuario/{id}`
Lista investimentos do usuário

POST
`/api/investimentos/usuario/{id}/upload`
Upload em lote de investimentos

PUT
`/api/investimentos/{id}`
Atualiza um investimento

CARTEIRA

GET
`/api/carteiras/usuario/{id}`
Carteira do usuário

GET
`/api/carteiras/{id}`
Carteira por ID

ATIVOS

GET
`/api/ativos/{ticker}`
Cotação em tempo real (API Externa)

Validação e Qualidade

Como Garantimos o Funcionamento

TESTES MANUAIS COM INSOMNIA

- Validar cada endpoint (CRUD)
- Testar requisições com autenticação
- Verificar respostas HTTP esperadas

TESTES DE INTEGRAÇÃO

- Validar comunicação com API Brapi
- Testar busca de cotações em tempo real
- Verificar tratamento de erros da API

VALIDAÇÃO DE REGRAS DE NEGÓCIO

- Testar exceção quando valor < preço do ativo
- Validar cálculos de projeção
- Verificar persistência de dados

TESTES DE SEGURANÇA

- Testar acesso sem credenciais (401)
- Validar autenticação Basic Auth
- Verificar autorização de endpoints

Conclusão e Aprendizados

Lições Aprendidas e Futuro do Projeto

APRENDIZADOS TÉCNICOS

- Aplicação prática de Spring Boot
- Mapeamento JPA e Hibernate
- Spring Security e autenticação
- Consumo de APIs externas
- Modelagem de dados relacional

TRABALHO EM EQUIPE

Reforço na colaboração, comunicação efetiva e versionamento de código com Git.
Divisão clara de responsabilidades entre backend e integração.

PRÓXIMOS PASSOS

- Implementar autenticação JWT
- Expandir cobertura de testes unitários
- Adicionar mais métricas de rentabilidade
- Integrar com mais APIs de mercado
- Implementar cache para otimização

Obrigado! — Miguel Antônio Gregório Firme & Lorenzo Viero Sartori