



Projeto de Bloco: Engenharia de Softwares Escaláveis [25E3_5]

TP02

Aluno: André Augusto Ferrarez

Profº: Leonardo Silva da Gloria

26/09/2025

SUMÁRIO

Link.....	2
Segunda Entrega: Desenvolver uma Camada de Persistência Real.....	2
1. Modelagem de Dados:.....	2
2. Integração de JPA com Spring Data:.....	2
3. Gerenciamento de Dados:.....	3
4. Integração de Funcionalidades de Histórico de Dados:.....	3
5. Implementação de Testes:.....	3

Link

<https://github.com/AndreFerrarez/transacoes>

Segunda Entrega: Desenvolver uma Camada de Persistência Real

1. Modelagem de Dados:

A solução proposta é manter uma tabela separada **“transacao_historico”** (modelo simples, explícito) e gravar nela sempre que o service executar operações que alterem o estado (salvar/excluir). Essa abordagem é simples, transparente e funciona sem dependências extras (ex.: Envers).

Explicação: A classe **“TransacaoHistorico”** armazena uma "foto" da transação no momento da operação, com um campo **“operacao”** para identificar o tipo de evento (CREATE/UPDATE/DELETE). Mantive a **“transacaoid”** para ligar ao **“Transacao”** original.

Foi feito também uma melhoria na classe **“Transacao”** para incluir construtores e garantir que a **“data”** seja persistida corretamente.

Explicação: adicionei anotações de validação (@NotNull, @Positive) para reforçar integridade em nível de aplicação (o Spring Validation irá usá-las no controller se ativarmos). Mantive a **“data”** com valor padrão de now().

2. Integração de JPA com Spring Data:

Foi adicionado um repositório Spring Data para **“Transacao”** (com métodos de consulta úteis) e criei a interface **“TransacaoHistorico”** com métodos úteis para consultar histórico por **“transacaoid”** e por **“moeda”**.

Explicação: adicionei ao **“TransacaoRepository”** os metodos **findByMoedaIgnoreCase** e **findByTipoIgnoreCase** para consultas mais eficientes no serviço (evitar **findAll().stream().filter(...)**).

E a **“TransacaoHistoricoRepository”** foi adicionado **findByTransacaoidOrderByDataOperacaoDesc**

e `findByMoedaIgnoreCaseOrderByDataOperacaoDesc` para as consultas de `transacaoid` e `moeda`.

Quando delegado ao banco, ganhamos performance e possibilidade futura de paginação.

3. Gerenciamento de Dados:

Garantir que sempre que criamos/atualizamos/excluimos uma transação, gravamos uma entrada no histórico.

Explicação detalhada:

- A lógica de gravação do histórico foi colocada no Service (não em EntityListener) porque o service é gerenciado pelo Spring e tem acesso aos repositórios — isso simplifica a injeção e transações.
- O método `salvar` detecta se é CREATE (`id == null`) ou UPDATE e grava um `TransacaoHistorico` correspondente.
- O método `excluir` primeiro busca a entidade (snapshot), grava histórico com `operacao = "DELETE"`, e só então exclui.
- Usei `@Transactional` nos métodos que gravam histórico + alteram dados para garantir atomicidade: ou salva transação + histórico ambos, ou falha tudo.

4. Integração de Funcionalidades de Histórico de Dados:

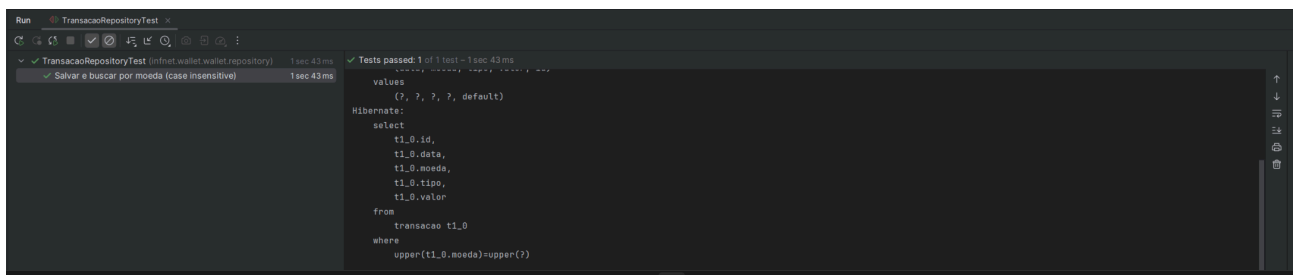
Foi atualizado a controller “`TransacaoController`” para expor endpoints que consultem histórico “`transacaoid`” e por “`moeda`”.

5. Implementação de Testes:

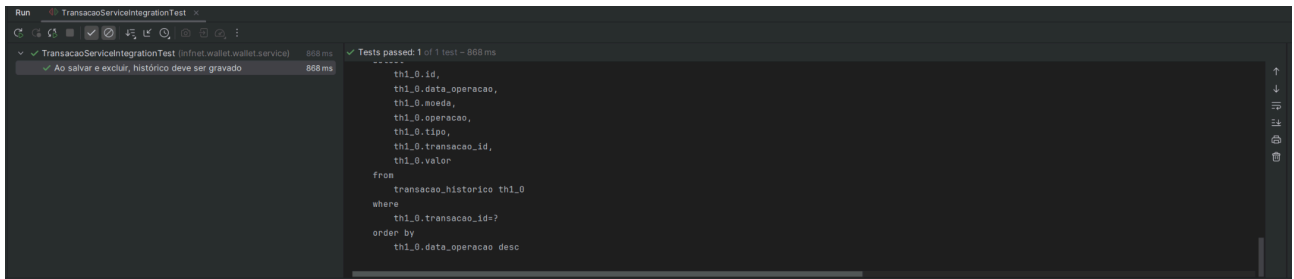
Foi implementado testes que validem a camada de persistência (repositórios e o fluxo que grava histórico).

Dois testes principais:

- `TransacaoRepositoryTest` com `@DataJpaTest` para operações básicas e método `findByMoedaIgnoreCase`



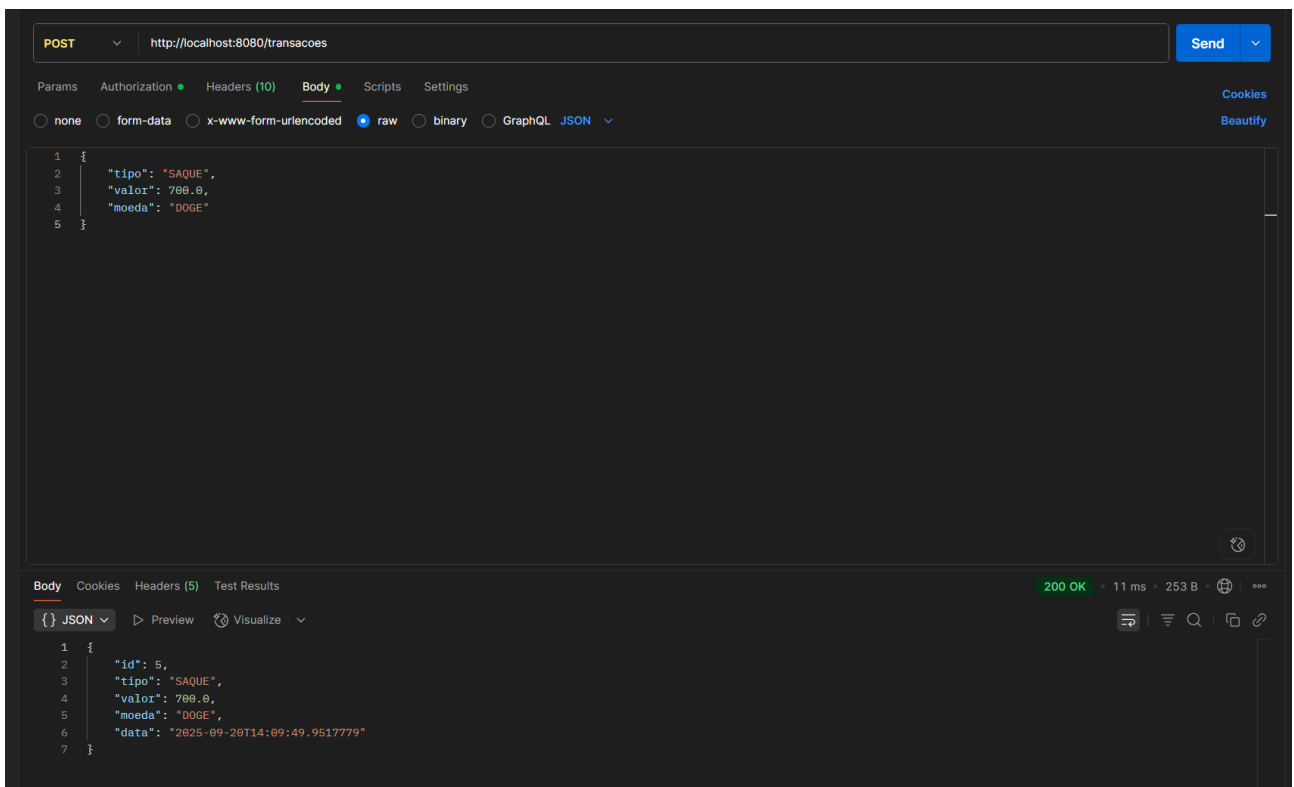
- TransacaoServiceIntegrationTest com @SpringBootTest para validar que salvar/excluir grava o histórico.



Prints que representam a aplicação rodando em H2 para fins de teste local, confirmando a implementação do histórico.

Mantendo assim a aplicação com rastreabilidade de cada operação feita.

POST api/transacoes + JSON



localhost:8080/h2-console/login.do?jsessionId=6720249bbae5ddc936e278d91c42ec93

Auto commit ☒ Max rows: 1000 Auto complete Off Auto select On

jdbc:h2:mem:testdb

- TRANSACAO
- TRANSACAO_HISTORICO
- INFORMATION_SCHEMA
- Users
- H2 2.3.232 (2024-08-11)

Run Run Selected Auto complete Clear SQL statement:

SELECT * FROM TRANSACAO_HISTORICO

SELECT * FROM TRANSACAO_HISTORICO;

VALOR	DATA_OPERACAO	ID	TRANSACAO_ID	MOEDA	OPERACAO	TIPO
500.00	2025-09-20 14:08:54.340375	1	3	BTC	CREATE	DEPOSITO
1500.00	2025-09-20 14:09:26.499277	2	4	DOGE	CREATE	DEPOSITO
700.00	2025-09-20 14:09:49.953671	3	5	DOGE	CREATE	SAQUE

(3 rows, 1 ms)

Edit

GET api/transacos

https://gestao-academia.onrender.com/api/disciplinas

GET http://localhost:8080/transacos Send

Params Authorization Headers (10) Body Scripts Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   "tipo": "SAQUE",
3   "valor": 700.0,
4   "moeda": "DOGE"
5 }
```

Body Cookies Headers (5) Test Results

JSON Preview Visualize

```
1 [
2   {
3     "id": 1,
4     "tipo": "SAQUE",
5     "valor": 1000.00,
6     "moeda": "BTC",
7     "data": "2025-09-20T14:04:13.066443"
8   },
9   {
10    "id": 2,
11    "tipo": "SAQUE",
12    "valor": 5000.00,
13    "moeda": "ETC",
14    "data": "2025-09-20T14:04:13.066443"
15  },
16  {
17    "id": 3,
18    "tipo": "DEPOSITO",
19    "valor": 500.00,
20    "moeda": "BTC",
21    "data": "2025-09-20T14:08:54.318331"
22  },
23  {
24    "id": 4,
25    "tipo": "DEPOSITO",
26    "valor": 1500.00,
27    "moeda": "DOGE",
28    "data": "2025-09-20T14:09:26.49726"
29  },
30  {
31    "id": 5,
32    "tipo": "SAQUE",
33    "valor": 700.00,
34    "moeda": "DOGE",
35    "data": "2025-09-20T14:09:49.951778"
36  }
37 ]
```

200 OK 15 ms 620 B

DELETE api/transacoes/{id}

https://gestao-academia.onrender.com/api/disciplinas

DELETE http://localhost:8080/transacoes/1

Send

Params Authorization Headers (10) Body Scripts Settings

Query Params

Key	Value	Description
Key	Value	Description

Body Cookies Headers (4) Test Results

200 OK 55 ms 123 B

Raw Preview Visualize

1

localhost:8080/h2-console/login.do?sessionId=6720249bbae5ddc936e278d91c42ec93

Auto commit Max rows: 1000 Auto complete Off Auto select On

jdbc:h2:mem:testdb

- TRANSACAO
- TRANSACAO_HISTORICO
- INFORMATION_SCHEMA
- Users
- H2 2.3.232 (2024-08-11)

Run Run Selected Auto complete Clear SQL statement:

SELECT * FROM TRANSACAO_HISTORICO

SELECT * FROM TRANSACAO_HISTORICO;

VALOR	DATA_OPERACAO	ID	TRANSACAO_ID	MOEDA	OPERACAO	TIPO
500.00	2025-09-20 14:08:54.340375	1	3	BTC	CREATE	DEPOSITO
1500.00	2025-09-20 14:09:26.499277	2	4	DOGE	CREATE	DEPOSITO
700.00	2025-09-20 14:09:49.953671	3	5	DOGE	CREATE	SAQUE
1000.00	2025-09-20 14:13:54.218577	4	1	BTC	DELETE	SAQUE

(4 rows, 0 ms)

Edit

GET(histórico) api/transacoes/historico/moeda/BTC

The screenshot shows a REST client interface with the following details:

- URL:** `http://localhost:8080/transacoes/historico/moeda/BTC`
- Method:** GET
- Response Status:** 200 OK
- Response Time:** 18 ms
- Response Size:** 435 B
- Response Body (JSON):**

```
[
  {
    "id": 4,
    "transacaoId": 1,
    "tipo": "SAQUE",
    "valor": 1000.00,
    "moeda": "BTC",
    "operacao": "DELETE",
    "dataOperacao": "2025-09-26T14:13:54.218577"
  },
  {
    "id": 1,
    "transacaoId": 3,
    "tipo": "DEPOSITO",
    "valor": 500.00,
    "moeda": "BTC",
    "operacao": "CREATE",
    "dataOperacao": "2025-09-26T14:08:54.348375"
  }
]
```