M4 - Tarefa 2 | Prática de TDD, Refactoring usando Caso de Ensino

Henrique Menna Barreto Araujo, Leonardo Fernandes Martins, Victor Gago Goncalves da Silva e Vitor Ferreira Michel

- Cenário 1: Listagem de todos os itens do estoque.

Dado que existem itens registrados no estoque
Quando eu solicitar a lista de todos os itens
Então eu devo receber a lista completa dos itens em estoque.

- Cenário 2: Consulta de um item específico do estoque.

Dado que o item com ID 123 está registrado no estoque
Quando eu solicitar os detalhes do item com ID 123
Então eu devo receber as informações detalhadas desse item.

- Cenário 3: Tentativa de consulta de um item que não existe.

Dado que o item com ID 999 não está registrado no estoque
Quando eu solicitar os detalhes do item com ID 999
Então eu devo receber uma mensagem informando que o item não foi encontrado.

- Cenário 4: Adicionando um novo item ao estoque.

Dado que quero adicionar um novo item ao estoque
Quando eu enviar as informações do novo item
Então o novo item deve ser registrado no estoque e eu devo receber uma confirmação.

- Cenário 5: Atualização de um item existente no estoque.

Dado que o item com ID 123 está registrado no estoque
Quando eu enviar as informações atualizadas para o item com ID 123
Então as informações do item devem ser atualizadas no estoque e eu devo receber uma confirmação.

Como evidência do desenvolvimento utilizando TDD, segue abaixo a sequência e dinâmica aplicada ao desenvolver os testes e endpoints.

- Controller da aplicação, método para buscar todos os itens de estoque:

```
getAllItems: async (req, res) => {
    try {
        res.status(200).json([]);
    } catch (error) {
        res.status(500).json({ message: "Erro ao buscar os itens.", error });
    }
},
```

- Teste unitário:

```
it('should fetch all items successfully', async () => {
  const mockItems = [{ id: 1, name: 'item1' }, { id: 2, name: 'item2' }];

  StockModel.getAllItems.mockResolvedValue(mockItems);

  const mockReq = {};
  const mockRes = {
    status: jest.fn().mockReturnThis(),
    json: jest.fn()
  };

  await getAllItems(mockReq, mockRes);

  expect(mockRes.status).toHaveBeenCalledWith(200);
  expect(mockRes.json).toHaveBeenCalledWith(mockItems);
});
```

- Execução do teste:

```
● StockController › should fetch all items successfully

  expect(jest.fn()).toHaveBeenCalledWith(...expected)

  - Expected
  + Received

  - Array [
  -   Object {
  -     "id": 1,
  -     "name": "item1",
  -   },
  -   Object {
  -     "id": 2,
  -     "name": "item2",
  -   },
  - ]
  + Array [],

  Number of calls: 1

    25 |
    26 |     expect(mockRes.status).toHaveBeenCalledWith(200);
  > 27 |     expect(mockRes.json).toHaveBeenCalledWith(mockItems);
       |                          ^
    28 |   });
    29 |
    30 |   it('should handle errors when fetching items', async () => {

    at Object.toHaveBeenCalledWith (src/controllers/stockController.test.js:27:26)
```

- Ajuste no código para passar no teste:

```
getAllItems: async (req, res) => {
    try {
        const items = await StockModel.getAllItems();
        res.status(200).json(items);
    } catch (error) {
        res.status(500).json({ message: "Erro ao buscar os itens.", error });
    }
},
```

- Teste executado com sucesso

```
 PASS  src/controllers/stockController.test.js
  StockController
    ✓ should fetch all items successfully (2 ms)

Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        0.761 s, estimated 1 s
```

Diagrama de Classes