Roteiro de Testes para a API "Boards and Cats"

Setup Inicial: Obter Tokens de Autenticação

1. Login como `joao.silva`

- **Método:** `POST`
- **URL:** `http://localhost:8080/login`
- **Body:** `{"username": "joao.silva", "password": "P4ssword"}`
- **Ação:** Copie o token da resposta. Salve como `TOKEN_JOAO`.

2. Login como `ana.souza`

- **Método:** `POST`
- **URL:** `http://localhost:8080/login`
- **Body:** `{"username": "ana.souza", "password": "P4ssword"}`
- **Ação:** Copie o token da resposta. Salve como `TOKEN_ANA`.

2. Login como `maria.lima`

- **Método:** `POST`
- **URL:** `http://localhost:8080/login`
- **Body:** `{"username": "maria.lima", "password": "P4ssword"}`
- **Ação:** Copie o token da resposta. Salve como `TOKEN_MARIA`.

A. Testes de Endpoints Públicos

A.1: Listar todas as Categorias (Sucesso)

- **Método:** `GET`
- **URL:** `http://localhost:8080/categories`
- **Authorization:** `No Auth`
- **Resultado Esperado:** Status `200 OK` e a lista de categorias.

A.2: Registrar um Terceiro Usuário (Sucesso)

- **Método:** `POST`
- **URL:** `http://localhost:8080/users/register`
- **Authorization:** `No Auth`
- **Body (JSON):**{

  "username": "alice.santos",

  "displayName": "Alice Santos",

  "password": "P4ssword",

  "phone": "11977776666"

}

- **Resultado Esperado:** Status `201 Created`.

B. Testes de CRUD de Endereços (`/addresses`)

B.1: Listar endereços de João (Sucesso)

- **Método:** `GET`
- **URL:** `http://localhost:8080/addresses`
- **Authorization:** `Bearer Token` (`TOKEN_JOAO`)
- **Resultado Esperado:** Status `200 OK` com a lista de endereços do João.

B.2: João tenta ver o endereço de Ana por ID (Falha de Permissão - Teste Crítico)

- **Contexto:** O endereço da Ana tem `id=3`.
- **Método:** `GET`
- **URL:** `http://localhost:8080/addresses/3`
- **Authorization:** `Bearer Token` (`TOKEN_JOAO`)
- **Resultado Esperado:** Status `403 Forbidden`.

B.3: Ana deleta seu próprio endereço (Sucesso)

- **Contexto:** O endereço da Ana tem `id=3`.
- **Método:** `DELETE`
- **URL:** `http://localhost:8080/addresses/3`
- **Authorization:** `Bearer Token` (`TOKEN_ANA`)
- **Resultado Esperado:** Status `204 No Content`.

C. Testes de Carrinho e Pedidos (/orders)

Parte 1: Gerenciando um Carrinho Existente (Usuária: Maria)

A usuária `maria.lima` já começa com um carrinho ativo, conforme `data.sql`.

1.1: Ver o carrinho existente (Sucesso)

- **Método:** `GET`
- **URL:** `http://localhost:8080/orders/cart`
- **Authorization:** `Bearer Token` (`TOKEN_MARIA`)
- **Resultado Esperado:**
  - **Status:** `200 OK`
  - **Corpo:** O JSON do carrinho da Maria, contendo "Catan" e "Codenames". Anote o `id` do carrinho (deve ser `5`) e os `id`s dos produtos (1 e 5).

1.2: Atualizar a quantidade de um item (Sucesso)

- **Método:** `PUT`
- **URL:** `http://localhost:8080/orders/cart/items/1` (Atualizando o "Catan", `productId=1`)
- **Authorization:** `Bearer Token` (`TOKEN_MARIA`)
- **Body (raw, JSON):**{

  "productId": 1,

  "quantity": 3

}

- **Resultado Esperado:**
  - **Status:** `200 OK`
  - **Corpo:** O JSON do carrinho atualizado, com a quantidade de "Catan" igual a 3 e o `total` recalculado.

1.3: Remover um item do carrinho (Sucesso)

- **Método:** `DELETE`
- **URL:** `http://localhost:8080/orders/cart/items/5` (Removendo "Codenames", `productId=5`)

- **Authorization:** `Bearer Token` (`TOKEN_MARIA`)
- **Resultado Esperado:**
    - **Status:** `200 OK`
    - **Corpo:** O JSON do carrinho atualizado, agora contendo apenas o "Catan", e com o `total` recalculado.

1.4: Limpar o carrinho inteiro (Sucesso)

- **Método:** `DELETE`
- **URL:** `http://localhost:8080/orders/cart`
- **Authorization:** `Bearer Token` (`TOKEN_MARIA`)
- **Resultado Esperado:**
    - **Status:** `204 No Content`.

Parte 2: Criando um Carrinho e Finalizando a Compra (Usuário: João)

O usuário `joao.silva` começa sem um carrinho ativo.

2.1: Adicionar item a um carrinho inexistente (Sucesso)

- **Método:** `POST`
- **URL:** `http://localhost:8080/orders/cart/items`
- **Authorization:** `Bearer Token` (`TOKEN_JOAO`)
- **Body (raw, JSON):**{

  "productId": 2,

  "quantity": 1

}

- **Resultado Esperado:**
    - **Status:** `200 OK`
    - **Corpo:** O JSON de um **novo** carrinho criado para o João, contendo 1 "Ticket to Ride".

2.2: Finalizar a compra (Checkout)

- **Contexto:** João usará seu endereço (ID 1) e método de pagamento (ID 1).
- **Método:** `POST`
- **URL:** `http://localhost:8080/orders/cart/checkout`

- **Authorization:** `Bearer Token` (`TOKEN_JOAO`)
- **Body (raw, JSON):**{

  "addressId": 1,

  "paymentMethodId": 1

}

- **Resultado Esperado:**
  - **Status:** `201 Created`
  - **Corpo:** O JSON do pedido agora finalizado, com status `PENDING`.

2.3: Verificar se o carrinho foi esvaziado após o checkout (Sucesso)

- **Método:** `GET`
- **URL:** `http://localhost:8080/orders/cart`
- **Authorization:** `Bearer Token` (`TOKEN_JOAO`)
- **Resultado Esperado:**
  - **Status:** `204 No Content`.

Parte 3: Testes de Segurança e Regras de Negócio

3.1: João tenta ver o carrinho da Maria (Falha de Segurança)

- **Método:** `GET`
- **URL:** `http://localhost:8080/orders/cart`
- **Authorization:** `Bearer Token` (`TOKEN_JOAO`)
- **Resultado Esperado: Status `204 No Content`**. Ele não vê o carrinho da Maria, apenas o seu (que está vazio). A segurança está funcionando.

3.2: Listar histórico de pedidos finalizados (Sucesso)

- **Método:** `GET`
- **URL:** `http://localhost:8080/orders`
- **Authorization:** `Bearer Token` (`TOKEN_JOAO`)
- **Resultado Esperado: Status `200 OK`** com a lista de todos os pedidos do João que **não** estão com o status `CART`.

3.3: João tenta ver detalhes de um pedido da Ana (Falha de Permissão)

- **Contexto:** O pedido 3 pertence à Ana.

- **Método:** `GET`
- **URL:** `http://localhost:8080/orders/3`
- **Authorization:** `Bearer Token` (`TOKEN_JOAO`)
- **Resultado Esperado: Status `403 Forbidden`.**

3.4: João tenta cancelar um pedido já enviado (Falha de Lógica de Negócio)

- **Contexto:** O pedido 2 do João tem o status `SHIPPED`.
- **Método:** `POST`
- **URL:** `http://localhost:8080/orders/2/cancel`
- **Authorization:** `Bearer Token` (`TOKEN_JOAO`)
- **Resultado Esperado: Status `500 Internal Server Error`** com a mensagem `"Não é possível cancelar um pedido com status SHIPPED"`.