Teste de rotas

Passo 1: Testando Rotas de Cliente

1. Registro de Cliente

- **Método**: `POST`
- **URL**: `/clientes`

Corpo da Requisição:

```
{
  "name": "Cliente Teste",
  "email": "cliente@example.com",
  "password": "senhaSegura"
}
```

-

Resposta Esperada (Status 201):

```
{
  "id": 1,
  "name": "Cliente Teste",
  "email": "cliente@example.com"
}
```

-
- **Descrição**: Cria um novo cliente no sistema.

2. Login de Cliente

- **Método**: `POST`
- **URL**: `/clientes/login`

Corpo da Requisição:

```
{
  "email": "cliente@example.com",
  "password": "senhaSegura"
}
```

-

Resposta Esperada (Status 200):

```
{
  "token": "jwt-token-aqui",
  "client": {
    "id": 1,
    "name": "Cliente Teste",
    "email": "cliente@example.com"
  }
}
```

-
- **Descrição**: Autentica o cliente e retorna o token JWT para autenticação.

3. Obter Dados do Cliente (Protegida)

- **Método**: `GET`
- **URL**: `/clientes/:id`
- **Cabeçalho**:
    - **Authorization**: `Bearer jwt-token-aqui`

Resposta Esperada (Status 200):

```
{
  "id": 1,
  "name": "Cliente Teste",
  "email": "cliente@example.com"
}
```

-
- **Descrição**: Obtém as informações de um cliente, protegida com JWT.

4. Atualizar Cliente (Protegida)

- **Método**: `PUT`
- **URL**: `/clientes/:id`
- **Cabeçalho**:
    - **Authorization**: `Bearer jwt-token-aqui`

Corpo da Requisição:

```
{
  "name": "Cliente Atualizado",
  "email": "clienteatualizado@example.com"
}
```

-

Resposta Esperada (Status 200):

```
{
  "id": 1,
  "name": "Cliente Atualizado",
  "email": "clienteatualizado@example.com"
}
```

-

5. Deletar Cliente (Protegida)

- **Método**: `DELETE`
- **URL**: `/clientes/:id`
- **Cabeçalho**:
  - **Authorization**: `Bearer jwt-token-aqui`
- **Resposta Esperada** (Status 204):
  - Sem corpo de resposta.

Passo 2: Testando Rotas de Funcionário

1. Registro de Funcionário

- **Método**: `POST`
- **URL**: `/employees`

Corpo da Requisição:

```
{
  "name": "Funcionario Teste",
  "email": "funcionario@example.com",
  "password": "senhaSegura"
}
```

-

Resposta Esperada (Status 201):
json
Copiar código

```
{
  "id": 1,
  "name": "Funcionario Teste",
```

```
  "email": "funcionario@example.com"
}
```

-
- **Descrição**: Cria um novo funcionário no sistema.

2. Login de Funcionário

- **Método**: `POST`
- **URL**: `/employees/login`

Corpo da Requisição:

```
{
  "email": "funcionario@example.com",
  "password": "senhaSegura"
}
```

-

Resposta Esperada (Status 200):

```
{
  "token": "jwt-token-aqui",
  "employee": {
    "id": 1,
    "name": "Funcionario Teste",
    "email": "funcionario@example.com"
  }
}
```

-
- **Descrição**: Autentica o funcionário e retorna o token JWT para autenticação.

3. Obter Dados do Funcionário (Protegida)

- **Método**: `GET`
- **URL**: `/employees/:id`
- **Cabeçalho**:
  - **Authorization**: `Bearer jwt-token-aqui`

Resposta Esperada (Status 200):

```
{
  "id": 1,
  "name": "Funcionario Teste",
```

```
  "email": "funcionario@example.com"
}
```

-
- **Descrição**: Obtém as informações de um funcionário, protegida com JWT.

4. Atualizar Funcionário (Protegida)

- **Método**: `PUT`
- **URL**: `/employees/:id`
- **Cabeçalho**:
    - **Authorization**: `Bearer jwt-token-aqui`

Corpo da Requisição:
json
Copiar código

```
{
  "name": "Funcionario Atualizado",
  "email": "funcionarioatualizado@example.com"
}
```

-

Resposta Esperada (Status 200):

```
{
  "id": 1,
  "name": "Funcionario Atualizado",
  "email": "funcionarioatualizado@example.com"
}
```

-

5. Deletar Funcionário (Protegida)

- **Método**: `DELETE`
- **URL**: `/employees/:id`
- **Cabeçalho**:
    - **Authorization**: `Bearer jwt-token-aqui`
- **Resposta Esperada** (Status 204):
    - Sem corpo de resposta.

Passo 3: Testando Rotas de Reserva

1. Criar Reserva

- **Método**: `POST`
- **URL**: `/reservations`

Corpo da Requisição:

```
{
  "clienteId": 1,
  "mesaId": 101,
  "restauranteId": 1,
  "dataReserva": "2024-12-10",
  "horaReserva": "19:30"
}
```

-

Resposta Esperada (Status 201):

```
{
  "id": 1,
  "clienteId": 1,
  "mesaId": 101,
  "restauranteId": 1,
  "dataReserva": "2024-12-10",
  "horaReserva": "19:30",
  "status": "Confirmada"
}
```

-

2. Obter Todas as Reservas

- **Método**: `GET`
- **URL**: `/reservations`

Resposta Esperada (Status 200):

```
[
  {
    "id": 1,
    "clienteId": 1,
    "mesaId": 101,
    "restauranteId": 1,
    "dataReserva": "2024-12-10",
    "horaReserva": "19:30",
    "status": "Confirmada"
```

```
  }
]
```

-

3. Obter Reservas por ID do Cliente

- **Método**: `GET`
- **URL**: `/reservations/clients/:clientId`

Resposta Esperada (Status 200):

```
[
  {
    "id": 1,
    "clienteId": 1,
    "mesaId": 101,
    "restauranteId": 1,
    "dataReserva": "2024-12-10",
    "horaReserva": "19:30",
    "status": "Confirmada"
  }
]
```

4. Atualizar Reserva

- **Método**: `PUT`
- **URL**: `/reservations/:id`

Corpo da Requisição:

```
{
  "clienteId": 1,
  "mesaId": 102,
  "restauranteId": 1,
  "dataReserva": "2024-12-12",
  "horaReserva": "20:00"
}
```

Resposta Esperada (Status 200):

```
{
```

```
  "id": 1,
  "clienteId": 1,
  "mesaId": 102,
  "restauranteId": 1,
  "dataReserva": "2024-12-12",
  "horaReserva": "20:00",
  "status": "Confirmada"
}
```

5. Cancelar Reserva

- **Método**: `PATCH`
- **URL**: `/reservations/:id/cancel`

Resposta Esperada (Status 200):

```
{
  "id": 1,
  "status": "Cancelada"
}
```

-

Testes no Postman

1. **Coleção de Testes**:
    - Crie uma coleção no Postman chamada "Sistema de Reservas".
    - Para cada rota descrita acima, crie uma nova requisição dentro dessa coleção.
    - Agrupe as rotas por "Clientes", "Funcionários" e "Reservas" para organização.
2. **Autenticação**:
    - Após fazer o login de um cliente ou funcionário, copie o token JWT retornado.
    - Use o token no cabeçalho de autorização nas rotas protegidas (`GET`, `PUT`, `DELETE`).
3. **Variáveis Globais**:
    - No Postman, configure variáveis globais para valores como `clienteId`, `funcionarioId`, `mesaId`, e `jwtToken` para facilitar os testes.