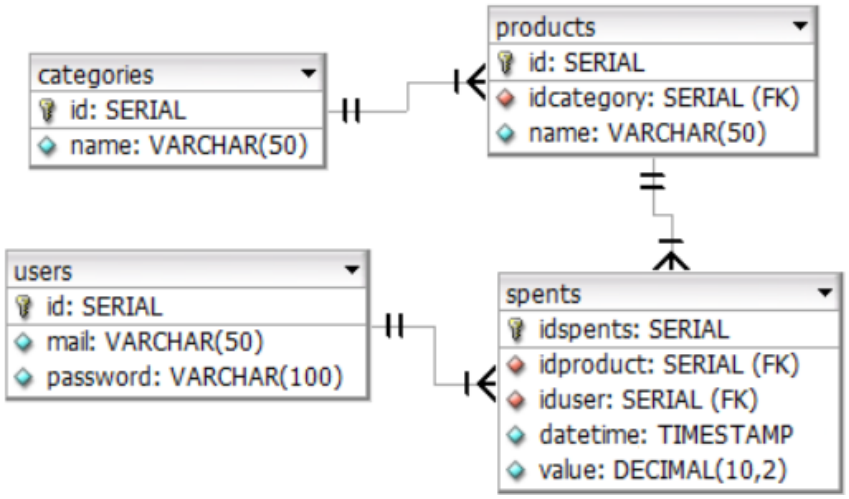
Descrição da atividade: o aplicativo disponível em https://github.com/arleysouza/jwt-postgresql possui as operações para fazer o CRUD nas tabelas representadas no modelo a seguir. Essas operações estão disponíveis através de rotas que possuem controle de acesso para usuários logados com o perfil *adm*, perfil *user* e sem a necessidade de estar logado. Ao efetuar o login, os dados do usuário são empacotados em um token e retornados para o cliente, que por sua vez, terá de enviar esse token em todas as requisições que requerem o controle de acesso. O token é gerado usando o pacote JWT (JSON Web Token). O código disponível no pacote *middlewares* é chamado antes da função objetivo da rota para decodificar o token e validar o perfil de acesso do usuário. Se o usuário não tiver permissão, a função middleware impede o acesso ao recurso mapeado pela rota.



Nota: 0,25 pts. na média final

Data de entrega: 28/maio

Forma de entrega: individual e presencial

Objetivos:

- Criar aplicação servidora usando Express;
- Conexão com o SGBD PostgreSQL;
- Hierarquia de rotas;
- Controle de acesso às rotas com token codificado usando JWT.

Exercícios

1) Atualmente o método list da classe ProductController retorna todos os registros da tabela products. Esse método é chamado através da rota HTTP GET /produto. O objetivo é chamar o método list passando pela URL um parâmetro de busca, no exemplo a seguir passou-se o termo ca. Faças as seguintes modificações no método list:
    - Caso não seja fornecido o termo de busca na URL: o método list deverá retornar os 5 primeiros produtos da tabela products, listados em ordem alfabética;

- Caso seja fornecido o termo de busca na URL: o método list deverá retornar os 5 primeiros produtos da tabela products que começam pelo termo fornecido na URL.

GET http://localhost:3010/produto


```
[
  {
    "id": 38,
    "name": "abacate",
    "category": "alimento"
  },
  {
    "id": 37,
    "name": "abacaxi",
    "category": "alimento"
  },
  {
    "id": 47,
    "name": "abóbora",
    "category": "alimento"
  },
  {
    "id": 66,
    "name": "abobrinha",
    "category": "alimento"
  },
  {
    "id": 3,
    "name": "açúcar",
    "category": "alimento"
  }
]
```


GET http://localhost:3010/produto/ca


```
[
  {
    "id": 94,
    "name": "cachaça",
    "category": "bebida"
  },
  {
    "id": 4,
    "name": "café",
    "category": "alimento"
  },
  {
    "id": 51,
    "name": "cajú",
    "category": "alimento"
  },
  {
    "id": 24,
    "name": "caldo de galinha",
    "category": "alimento"
  },
  {
    "id": 67,
    "name": "cará",
    "category": "alimento"
  }
]
```


Dicas:

- Será necessário criar duas rotas no arquivo src/routes/product.ts. Sendo uma rota para /produto e outra para /produto/:Search, mas ambas tendo como objetivo o método list da classe ProductController;
- No método list, codifique dois comandos SQL, sendo um para quando existir o termo de busca e outro para quando não existir;
- Use o operador SQL ilike para fazer a comparação com o termo de busca;
- Adicione o caractere especial % após o termo de busca para listar todos os nomes de produtos que começam pelo termo de busca.

2) Atualmente o método list da classe SpentController retorna todos os gastos do usuário logado. Alterar essa função para retornar apenas 5 gastos por vez, ou seja, os dados serão paginados com 5 registros. Faça as seguintes modificações no método list:
   - O método receberá o número da página e deverá retornar 5 registros a partir da página fornecida;
   - Caso o usuário não forneça o número da página, o método retornará os 5 primeiros gastos, ordenados em ordem decrescente de data;

- Caso o usuário forneça o número da página além da última, o método retornará os registros da última página;
- O método deverá retornar o número de páginas (pages), a página atual (page), a quantidade de gastos do usuário (count), o valor médio dos gastos (average) e os gastos da página (spents).

Dicas:

- Será necessário substituir a rota de HTTP GET /gasto pelas rotas HTTP GET /gasto/listar e /gasto/listar/:page. Ambas tendo como objetivo o método list da classe SpentController;
- No método list, faça um comando para obter a quantidade total e o valor médio dos gastos que são do usuário logado. Será necessário dividir a quantidade por 5 para obter a quantidade de páginas;
- No método list, monte a instrução SQL usando o número da página no termo OFFSET para obter os registros da paginação.

Exemplos de resultado quando logado como usuário maria@teste.com.

GET http://localhost:3010/gasto/listar


```
{
  "pages": 8,
  "page": 1,
  "count": 39,
  "average": 9.79,
  "spents": [
    {
      "id": 31,
      "name": "feijão",
      "value": 7.9,
      "datetime": "2024-05-16T15:45:55.000Z"
    },
    {
      "id": 14,
      "name": "arroz",
      "value": 24.99,
      "datetime": "2024-05-16T15:45:19.000Z"
    },
    {
      "id": 4,
      "name": "queijo",
      "value": 19.5,
      "datetime": "2024-05-16T15:45:09.000Z"
    },
    {
      "id": 69,
      "name": "óleo",
      "value": 6.5,
      "datetime": "2024-05-16T15:42:39.000Z"
    },
    {
      "id": 51,
      "name": "açúcar",
      "value": 3.15,
      "datetime": "2024-05-16T15:42:09.000Z"
    }
  ]
}
```


GET http://localhost:3010/gasto/listar/10


```
{
  "pages": 8,
  "page": 8,
  "count": 39,
  "average": 9.79,
  "spents": [
    {
      "id": 47,
      "name": "açúcar",
      "value": 3.2,
      "datetime": "2024-03-22T14:34:29.000Z"
    },
    {
      "id": 28,
      "name": "feijão",
      "value": 7.5,
      "datetime": "2024-03-22T14:34:22.000Z"
    },
    {
      "id": 11,
      "name": "arroz",
      "value": 25,
      "datetime": "2024-03-22T14:34:12.000Z"
    },
    {
      "id": 1,
      "name": "queijo",
      "value": 18.4,
      "datetime": "2024-03-22T14:34:11.000Z"
    }
  ]
}
```

3) Codificar o método statsByProduct, na classe SpentController, para retornar para cada produto que existe na tabela spents: o valor médio, a quantidade de vezes que o produto aparece na tabela de gastos, o menor e maior valor registrado para o produto. Ao lado tem-se parte do resultado.

   Crie a rota HTTP GET /gasto/produtos com restrição de acesso para usuário com perfil adm e coloque como objetivo da rota o método statsByProduct.

GET http://localhost:3010/gasto/produtos


```
[
  {
    "name": "açúcar",
    "avg": 3.47,
    "count": 18,
    "max": 3.99,
    "min": 2.99
  },
  {
    "name": "arroz",
    "avg": 25.14,
    "count": 17,
    "max": 26.2,
    "min": 23.9
  },
  {
    "name": "detergente",
    "avg": 3.41,
    "count": 21,
    "max": 4.2,
    "min": 2.75
  },
```


4) Codificar o método statsByUser, na classe SpentController, para retornar para cada usuário que existe na tabela spents: o total gasto e a quantidade de gastos.

   Crie a rota HTTP GET /gasto/usuarios com restrição de acesso para usuário com perfil adm e coloque como objetivo da rota o método statsByUser.

GET http://localhost:3010/gasto/usuarios


```
[
  {
    "mail": "ana@teste.com",
    "sum": 327.64,
    "count": 37
  },
  {
    "mail": "joao@teste.com",
    "sum": 396.47,
    "count": 38
  },
  {
    "mail": "maria@teste.com",
    "sum": 381.96,
    "count": 39
  },
  {
    "mail": "pedro@teste.com",
    "sum": 403.43,
    "count": 34
  }
]
```