Sumário

[Repositório aplicação inicial 1](#_TOC_250005)

[Criar uma instancia do mongo no Docker 1](#_TOC_250004)

[Configurar a conexão no app 2](#_TOC_250003)

[Criar a Model 2](#_TOC_250002)

[Criar a repository 3](#_TOC_250001)

[Criar a Controller 4](#_TOC_250000)

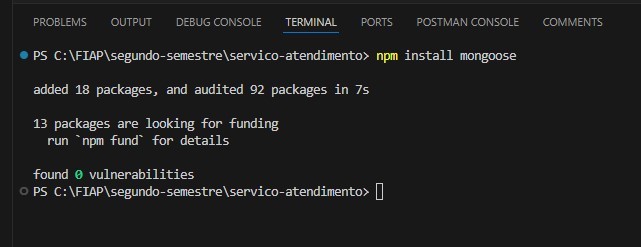
# Repositório aplicação inicial

https://github.com/professorthiagoxavier/microsservice/branches

branch: aula-1-status-code

Instalar o mongoose

1 - Abrir o cmd e utilizar o comando:

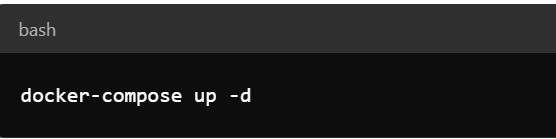


# Criar uma instancia do mongo no Docker

É importante configurar se o Docker está iniciado.

Após isso, vamos utilizar o Docker-compose disponibilizado no Repositório para iniciar a instância.

docker-compose up -d



**Se tiver algum problema do container subir e cair em seguida é problema no compose**

**Pode dar um Docker-compose down pra excluir tudo e depois o up -d denovo quando consertar**

**Pode dar Docker logs “nome do container”**

**Pode dar docker ps -a para ver todas imagens disponíveis**

**docker exec -it “nome do container”/bin/bash**

**mongosh --username root --password example --authenticationDatabase admin (nesse caso o docker-compose estava configurado para root e example)**

**Docker-compose do professor estava errado, não tava rodando o mongo-express, o do “hub.docker.com” funcionou**

**https://hub.docker.com/\_/mongo**

**Testando mongo:**

**show dbs (admin, config, local = indica que nenhum banco de dados adicional foi criado)**

**use “nome de um banco”**

**show collections**

**Criar uma coleção: db.createCollection("products")**

**Inserir um documento: db.products.insert({ name: "Produto 1", price: 100 })**

**Verificar se o documento foi inserido: db.products.find()**

**Dá npm start pra tetsar no postman as outras coisas, get/post dos produtos**

**No fim do projeto você deve pausar/parar os containers:**

**1. Pausar os contêineres (docker pause): se você vai continuar trabalhando no projeto mais tarde e quer manter os contêineres "prontos", mas não quer que eles estejam consumindo recursos (CPU/memória). Isso preserva o estado do contêiner.**

**docker pause nome\_ou\_id\_do\_container**

**docker unpause nome\_ou\_id\_do\_container**

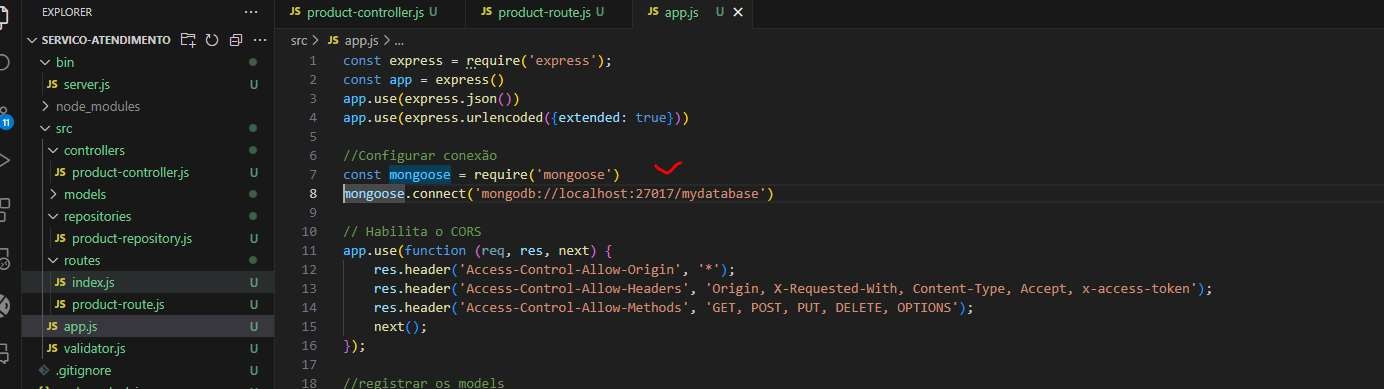
**2. Parar os contêineres (docker stop): se você não precisa que os contêineres fiquem em execução agora, mas deseja manter o contêiner (e seus volumes) para retomá-lo facilmente depois.**

**docker stop nome\_ou\_id\_do\_container**

**docker start nome\_ou\_id\_do\_container**

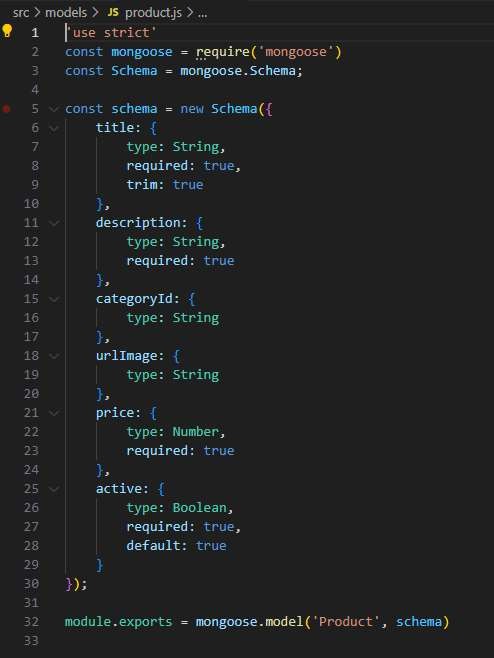
# Configurar a conexão no app

Abra o arquivo app.js e configure a conexão;



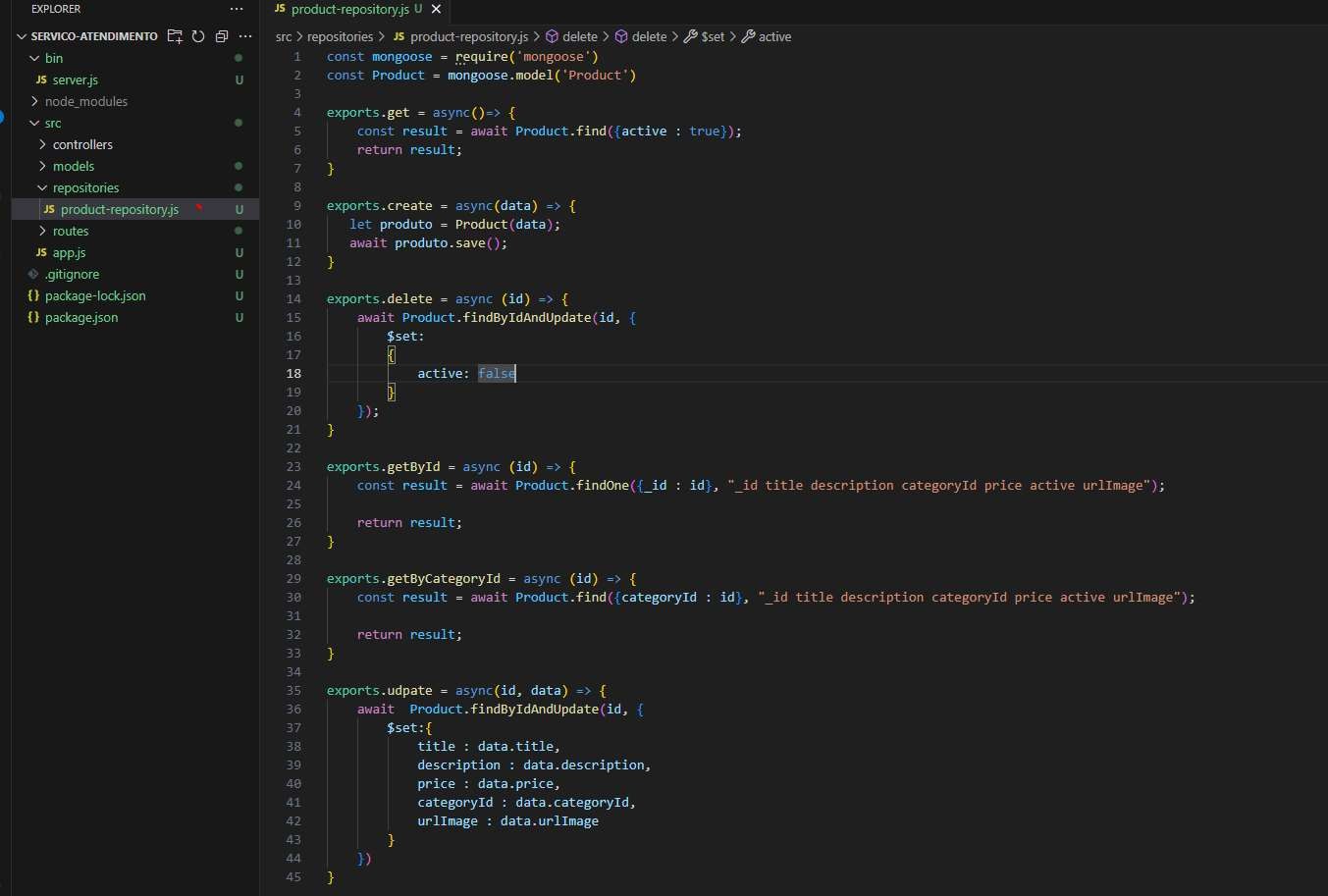
# Criar a Model

1. Crie a pasta na seguinte estrutura src/models
2. Crie um arquivo chamado produto.js



# Criar a repository

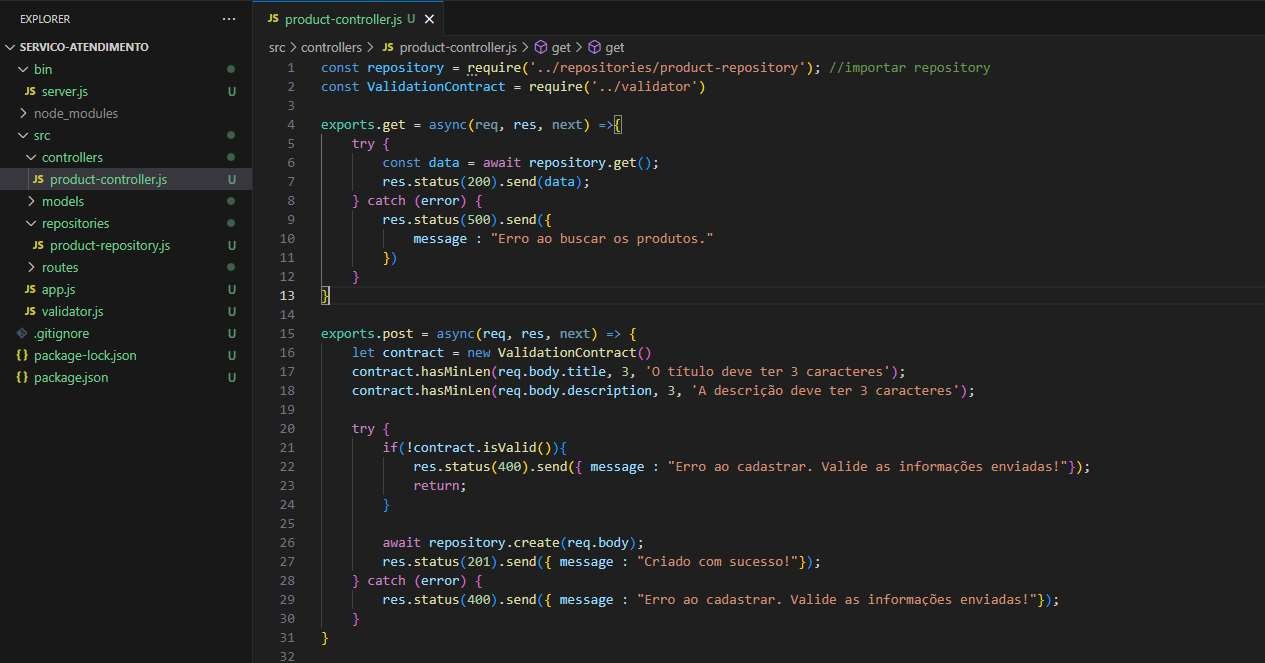
1. Crie a pasta na seguinte estrutura src/repository
2. Crie um arquivo chamado produto-repository.js



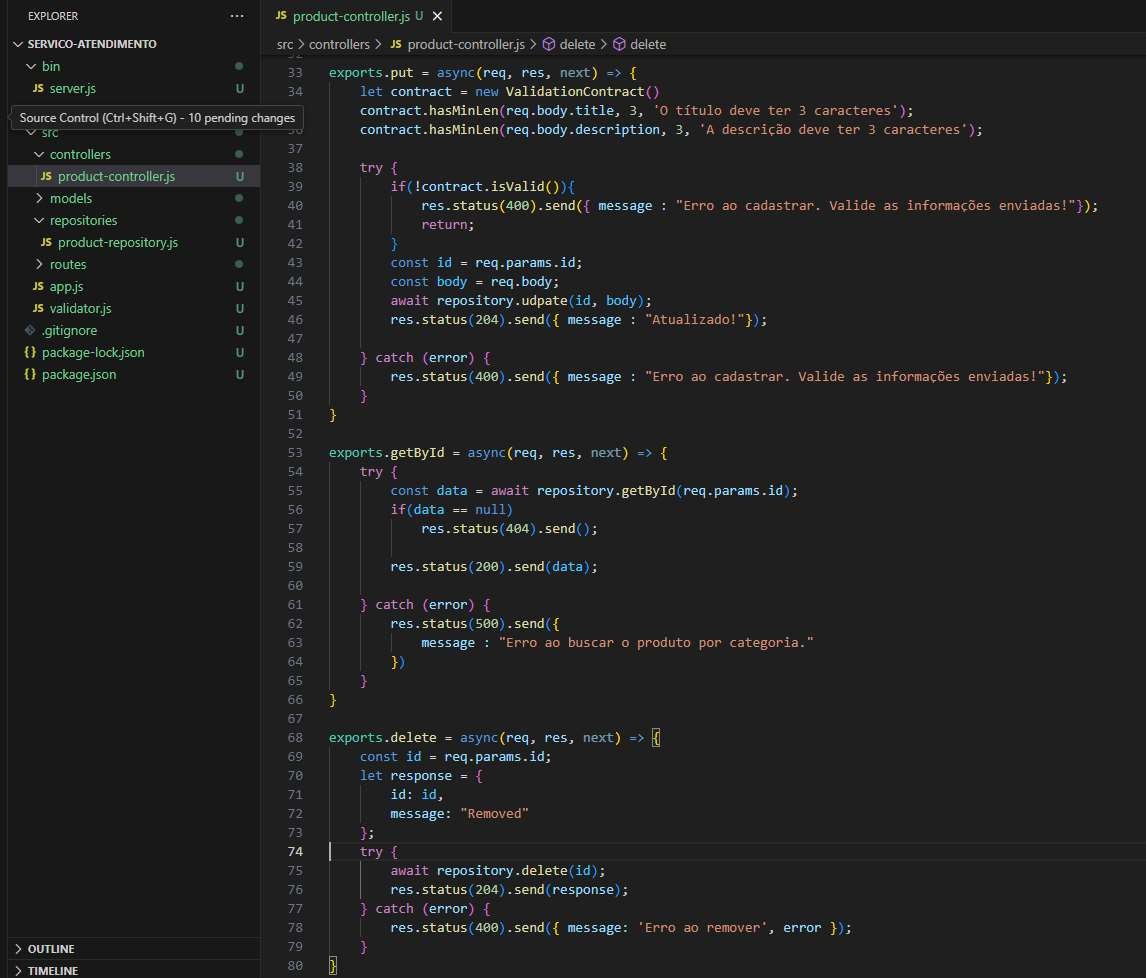
# Criar a Controller

Ponto de atenção para o arquivo de validação, para tratarmos bad requests (400)

Get e post



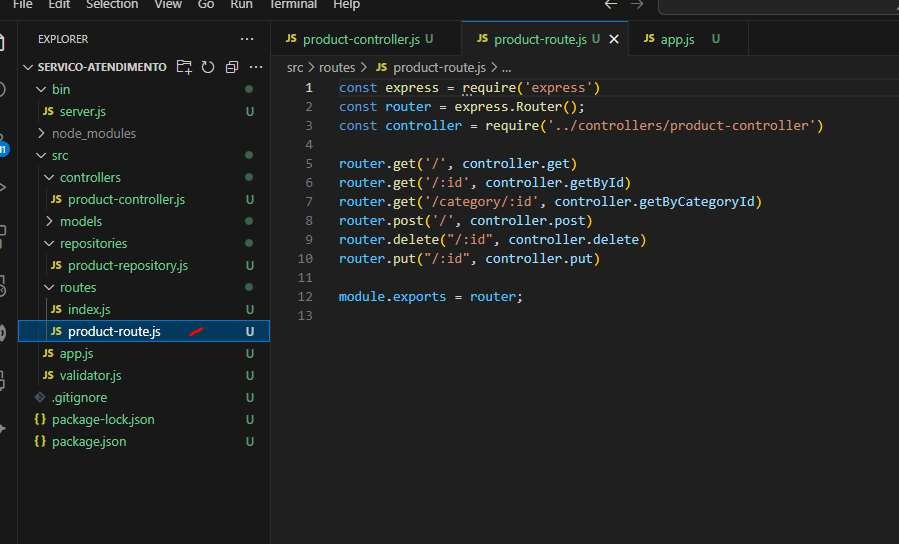
Put, GetById e Delete



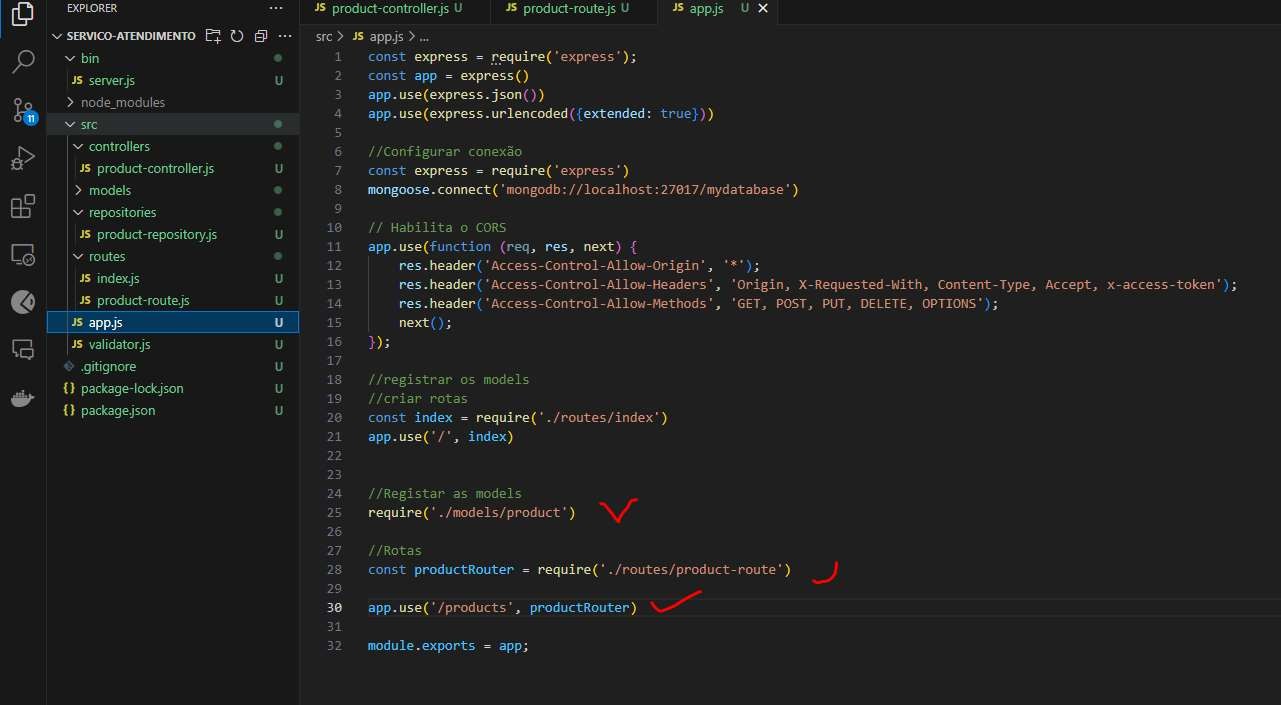
Salvar o arquivo validator.js (arquivo está no repositório na branch da aula) no mesmo nível do app.js.

# Criar a rota Product

1. Abra a pasta de rota e adicione o arquivo product-route.js



Regitrar a rota e a Model



Agora é só ex