



JS na Web: CRUD com JavaScript assíncrono

Prof. Marcondes Alexandre

Descrição

A utilização de ferramentas como Node.js e MongoDB na implementação de uma aplicação JavaScript, realizando o CRUD de forma assíncrona.

Propósito

Definir o front-end e o back-end de sistemas Web utilizando a linguagem JavaScript, obedecendo ao padrão arquitetural MVC, por meio de ferramentas como Node.js e MongoDB, as quais viabilizam a criação de aplicações que realizam as operações de banco de dados CRUD de forma assíncrona.

Preparação

Antes de iniciar seu estudo, baixe o [projeto LojaWeb](#) desenvolvido no conteúdo, a fim de acompanhar a implementação dos códigos de programação.

Objetivos

Módulo 1

O que é o CRUD e conectando ao SGBD

Criar um banco de dados orientado a documento utilizando o MongoDB.

Módulo 2

Operações de banco de dados

Implementar as operações de consulta, inserção, alteração e exclusão de banco de dados com o MongoDB.

Módulo 3

Aplicação JS na Web

Construir uma aplicação JavaScript implementando as operações do CRUD (Create, Read, Update e Delete) de forma assíncrona.



Introdução

No desenvolvimento de código-fonte, os programadores utilizam várias teorias, técnicas e ferramentas na elaboração de seus sistemas de informação.

Veremos que o uso da linguagem de programação JavaScript para sistemas Web possibilita a integração com sistemas de banco de dados, seja para trabalhar com dados estruturados ou semiestruturados de maneira eclética, constituindo uma forte característica da referida linguagem.

As operações de banco de dados serão implementadas utilizando o banco orientado a documento MongoDB, incluindo o CRUD, ou seja, a inserção, consulta, atualização e exclusão de dados.

Ao final, implementaremos uma aplicação utilizando o framework Node.js, com destaque para a implementação de operações CRUD assíncronas, utilizando código JavaScript.

Durante seus estudos lembre-se, geralmente, o mundo da computação é colaborativo. Sendo assim, é comum, quando encontramos erros nas bibliotecas utilizadas, copiarmos o log de erro e procurarmos em qualquer motor de busca. Provavelmente, alguma resposta será retornada em algum fórum de discussão, como Stack Overflow, GitHub, Reddit, entre outros. Isso não só é uma prática comum na comunidade de desenvolvimento de software e computação, como também nos possibilita aprender cada vez mais. Vamos lá!



1 - O que é o CRUD e conectando ao SGBD

Ao final deste módulo, você será capaz de criar um banco de dados orientado a documento utilizando o MongoDB.

O que é o CRUD e conectando ao SGBD

JavaScript

Na dinâmica imposta de mudanças e atualizações cada vez mais frequentes e urgentes, a equipe de desenvolvimento de software busca encontrar meios que auxiliem sua produtividade, eficiência e qualidade das entregas mediante a alterações de requisitos pelo cliente. Nesse contexto, a adoção de uma ferramenta que pudesse favorecer as atividades de cada membro da equipe de desenvolvimento de maneira a organizar as tarefas e redução parece algo intangível, não é mesmo?

O **JavaScript** foi criado por Brendan Eich em 1995, com o objetivo de ser uma linguagem:



Simples



Produtiva



Versátil

O objetivo era facilitar a vida dos web designers a manipularem e estenderem a linguagem HTML que possui certas restrições, dentre elas o acesso a dados de forma dinâmica. Por meio da **API DOM**, que consiste num conjunto de interfaces fornecido pelo navegador, é possível realizar essa tratativa de adequações ao HTML. Logo, depois disso, **DHTML** se tornou o termo popular, referindo-se às interfaces de usuário mais dinâmicas que o JavaScript permitia, desde estados de botões de rolagem animados até validação de formulários do lado do cliente. A seguir podemos ilustrar esquematicamente essa relação do **JavaScript e HTML**:



Estruturação do DHTML.

Devido às suas capacidades crescentes, o JavaScript atraiu uma comunidade apaixonada que impulsiona seu crescimento e onipresença. E devido à sua considerável popularidade, agora existem inúmeras maneiras diferentes de fazer a mesma coisa em JavaScript. Existem milhares de estruturas, bibliotecas e utilitários disponíveis.

A linguagem também está mudando em uma base quase constante em reação às crescentes demandas de suas aplicações. Isso cria um grande desafio!

Reflexão

Entre todas essas mudanças, sendo empurradas e puxadas em diferentes direções, como podemos saber como escrever o melhor código possível? Quais frameworks devemos usar? Que convenções devemos empregar? Como devemos testar nosso código?

Aplicando o conceito de CRUD

O termo “aplicativo CRUD”, parece estranho e soa de certa maneira esquisito, não é verdade? Pois bem, na verdade, é um acrônimo que foi popularizado pela primeira vez no início dos anos 1980 por James Martin em referência a aplicativos que criam, leem, atualizam e excluem dados, portanto, CRUD em inglês poderia ser entendido assim: **Create**, **Read**, **Update** e **Delete**. Agora parece mais amigável o termo?

Em nosso cotidiano utilizamos aplicações que executam essas operações, bem como em sistema de informação em empresas. Imagine um sistema de controle de estoque em uma organização em que o colaborador irá criar alguns dados, portanto cadastrando produtos, depois poderá precisar consultar esses dados e, por fim, atualizar ou excluir caso seja necessário. Ou seja, CRUD representa as **quatro operações básicas** possíveis em uma **base de dados**.

Independentemente da linguagem de programação de acesso a dados que você possa utilizar, todas possuem comandos para operações CRUD. É evidente que a linguagem pode apresentar uma sintaxe ligeiramente diferente, no entanto, executará a mesma função. Logo, o desenvolvedor poderá elaborar seu código-fonte para manipular os dados por meios de métodos e utilizar uma estrutura de

armazenamento dos dados para que possa ser vinculada ao seu programa. Há basicamente dois tipos de dados:

Dados estruturados



Possuem uma estrutura previamente determinada, não sendo flexível e as mudanças podem acarretar mais impactos nos sistemas de informação. Para que você possa compreender melhor, vamos imaginar uma planilha eletrônica, como Excel:

Dados estruturados em planilha.

Observe que temos as colunas: data, item e anotações. Pois bem, todos os registros da nossa lista de compra obrigatoriamente terão essa estrutura independentemente de você inserir o dado na coluna ou não. Logo, podemos inferir que essa estrutura é estática para todos os registros de nossa planilha.

Dados não estruturados



Possuem flexibilidade em sua estrutura, logo não obriga uma modelagem de dados prévia para armazenamento dos dados, o que facilita a escalabilidade e contribui para uma maior disponibilidade.

Dados não estruturados.

Exemplos: imagens, vídeos, relatórios textuais, áudios, entre outros similares.

Organizando os dados em um banco de dados

Dados vs. informação

As empresas trabalham com dados todos os dias, seja para efetivar pagamentos, controlar estoque de produtos, registrar as vendas e tantas outras coisas. Essas necessidades ressaltam a importância do uso de um banco de dados. Agora, para iniciarmos essa organização dos dados, precisamos elucidar dois conceitos fundamentais, que são: dados e informação.



Dados

Os dados representam algo “bruto”, sem valor ainda associado que possa claramente trazer uma compreensão ao lermos esses dados.



Informação

A informação tem um caráter maior na compreensão do que está analisando, logo, proporciona um significado contextualizado em nossa análise.

Veja a seguir uma planilha eletrônica sobre uma lista de compras:

Lista de Compras

DATA	ITEM	ANOTAÇÕES
[Data]	[Item]	[Observação]
[Data]	[Item]	[Observação]
[Data]	[Item]	[Observação]
[Data]	[Item]	[Observação]

Dados estruturados em planilha.

As colunas data, item e anotações são os campos nos quais iremos colocar os dados. A informação poderia ser algo como: “qual item tem mais anotações?” Em um sistema de controle de estoque de uma empresa, poderia ser, por exemplo, “qual produto tem mais saída e de qual fornecedor?” Ficou clara a diferença entre dado e informação?

Banco de dados

Podemos definir um banco de dados como sendo uma coleção de dados organizados que representa o que se deseja armazenar e recuperar por meio de um sistema de informação de forma ágil. No banco de dados é factível termos milhares de registros armazenados e, se fôssemos fazer o controle de todos os registros manualmente em uma planilha eletrônica, seria uma atividade gigantesca e plausível de erros.

Felizmente, não precisamos fazer essa aventura, ufa!

Com o uso de um banco de dados, é possível informar o que desejamos recuperar por meio de instruções que são enviadas para um servidor que atuará como servidor de banco de dados. Esse servidor processará essa consulta no banco de dados e retornará as informações requeridas pelo usuário por meio de um programa de computador que realizou essa solicitação.

Como dissemos, os bancos de dados são frequentemente encontrados em servidores de banco de dados para que possam ser acessados por vários usuários e fornecendo os dados que necessitamos. Mas, afinal de contas, eu consigo acesso direto a um banco de dados?

Resposta

Você acessa o banco de dados **indiretamente**. Há um software específico que é responsável pela gestão, organização e acesso aos bancos de dados, que chamamos de SGBD (Sistema Gerenciador de Banco de Dados). É por meio dele que podemos criar um banco de dados e as demais estruturas necessárias para armazenar, bem como submeter as instruções para recuperação dos dados.

Uma das principais vantagens do processamento de JavaScript no lado do servidor via Node.js é o acesso a um **Sistema de Gerenciamento de Banco de Dados**.

Usando JavaScript com Visual Studio Code

Instalação do Visual Studio Code

O Microsoft Visual Studio Code é um poderoso **editor de código**, seja para sistemas de informação locais numa empresa, como também

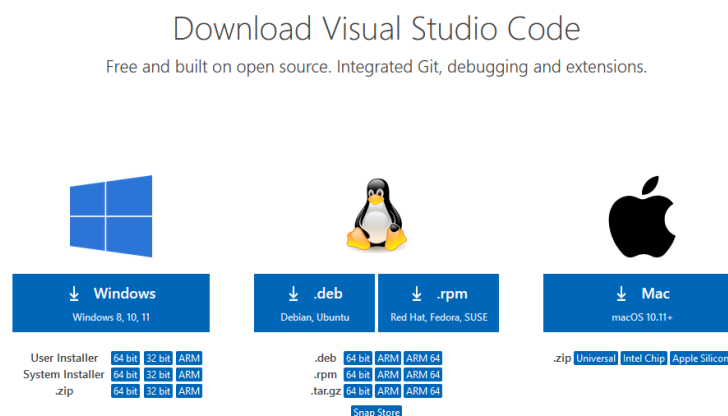
nativamente projetados para um ambiente de computação em nuvem. Apresenta uma interface amigável e intuitiva para o desenvolvedor, além de não necessitar de muito recurso de processamento para sua utilização na estação de trabalho do programador.

Podemos utilizá-los para elaborar código-fonte tendo como premissa a linguagem de sua preferência, haja vista que ele suporta diversas linguagens de programação, como: Python, C#, Node.js e Rubi, dentre outras. Outra característica importante é ser **multiplataforma**, portanto, você pode desenvolver programação no sistema operacional Linux, Windows e MacOS.

Vamos começar a praticar?

Inicialmente, você precisa instalar o Visual Studio Code em sua estação de trabalho. Escolha a opção para download da ferramenta que está de acordo com o seu sistema operacional. Para ambiente Windows, é recomendável a utilização da versão 64bits por apresentar a capacidade de endereçamento de alocação de memória acima de 4GB.

Já para o sistema operacional Linux, a Microsoft recomenda que seja instalado em algumas distribuições em virtude do suporte oferecido quando é celebrado contrato de suporte de uma empresa com a própria Microsoft. É de notório saber que existem diversas distribuições Linux. Caso sua escolha seja diferente das listadas aqui, a estabilidade da ferramenta não é garantida porque foi homologada para essas distribuições. A imagem a seguir ilustra as opções para uso de acordo com o sistema operacional:

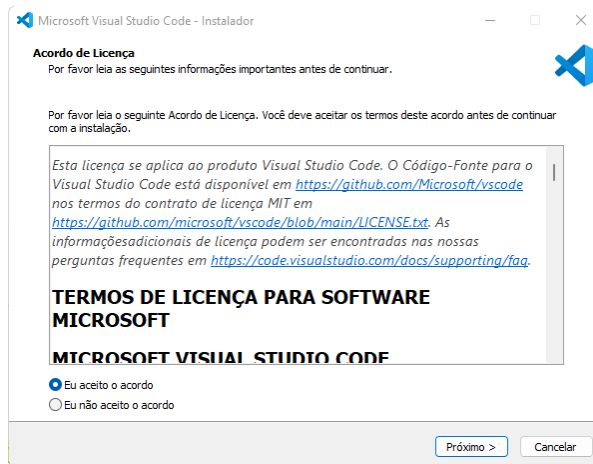


Escolhendo a Visual Studio Code para download.

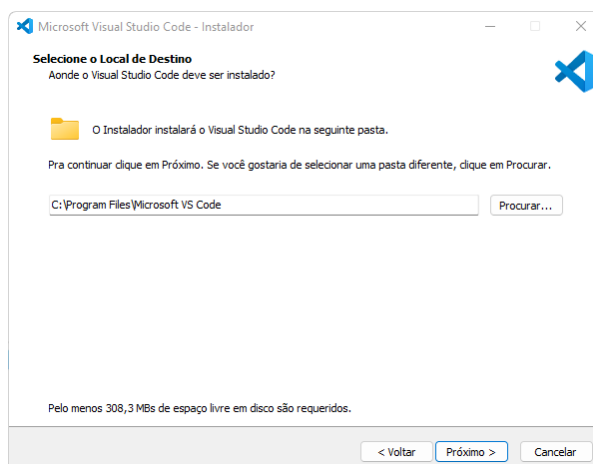
Agora que você realizou o download, vamos proceder com a instalação da ferramenta:



Nesta imagem, vemos a tela com o acordo de licença de uso. É importante ressaltar que essa ferramenta não tem custo para sua aquisição. No entanto, se fosse definido pelo time de desenvolvedores a utilização do Visual Studio, poderia incorrer em custo de licença de acordo com as características das edições disponíveis para uso.

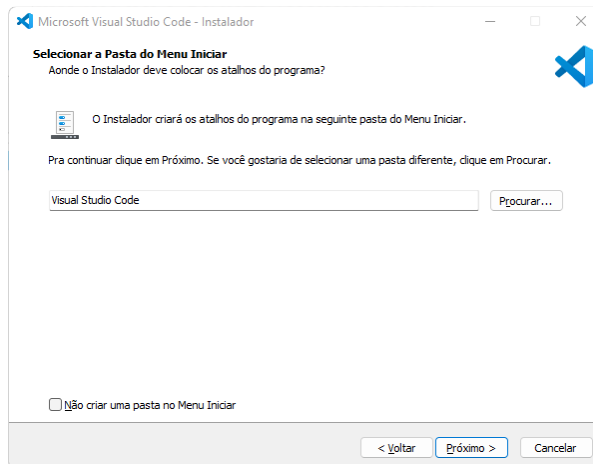


Na sequência, vemos uma tela em que você poderá escolher em qual diretório os binários da ferramenta serão instalados. Não há nenhum problema em você deixar a localização padrão apresentada pelo sistema operacional. Na imagem, é exibida a opção padrão do sistema operacional Windows.



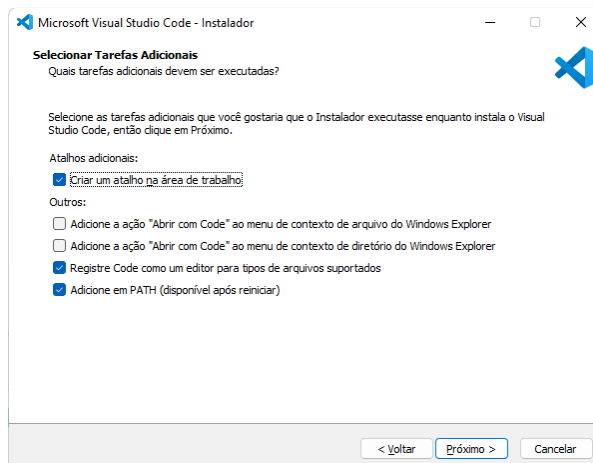
3

Aqui vemos de que forma o atalho do programa será referenciado no sistema operacional.



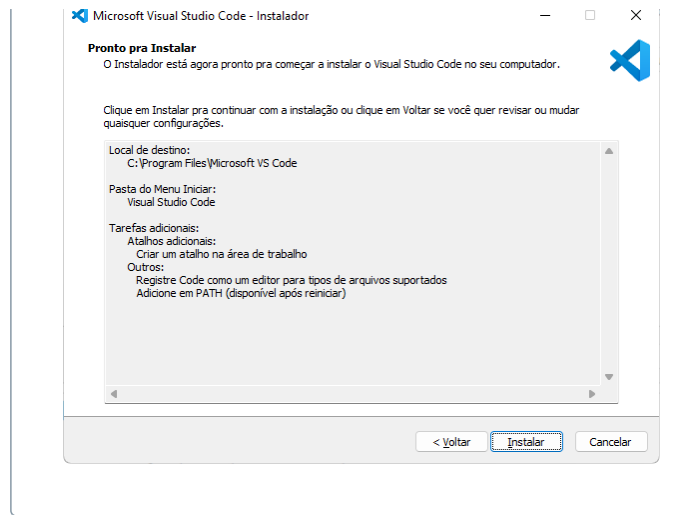
4

Nesta etapa, é possível adicionar tarefas ao processo de instalação, como: criar um atalho na área de trabalho, adicionar um PATH e registrar Code como editor para tipos de arquivos suportados, por exemplo.



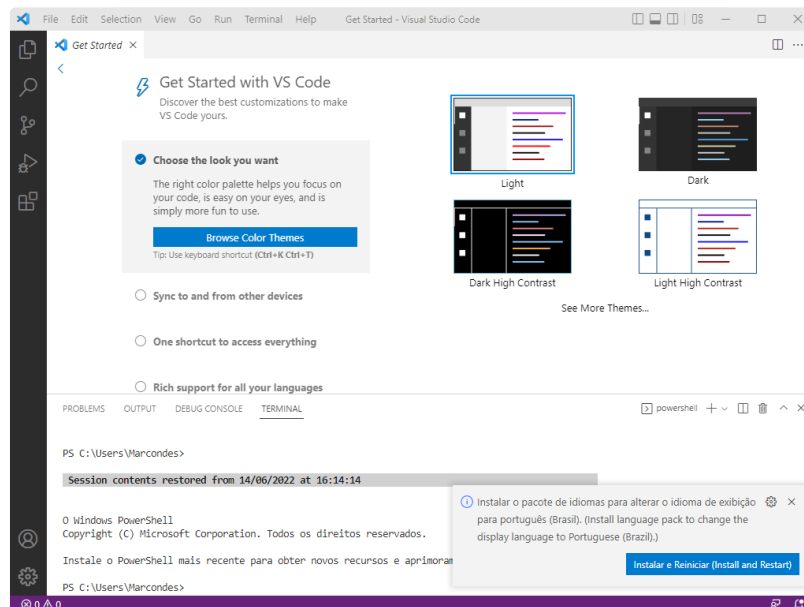
5

Por fim, chegamos à etapa final do processo de instalação que apresenta um resumo das escolhas que foram feitas para a instalação do Visual Studio Code. Caso deseje alterar algumas dessas definições, será necessário clicar no botão Voltar e proceder com a modificação. Clique em instalar para que o assistente de instalação execute essa atividade, conforme mostra a imagem.



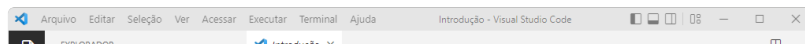
Aí, sim! Aqui é mostrado que a instalação foi realizada com sucesso e que você poderá iniciar o Visual Studio Code agora.

Seja bem-vindo ao Visual Studio Code ou, como também é conhecido o programa, ao VS Code. Na imagem a seguir, é ilustrada a página inicial de configuração na qual você poderá escolher o tema a ser aplicado para a ferramenta, bem como instalar o pacote de idioma de acordo com a configuração do seu sistema operacional.



Definindo o tema do VS Code.

Já temos o VS Code pronto para começarmos a programar. Agora veja a tela inicial após aplicação do pacote de idioma.



Tela inicial do VS Code.

Precisamos agora de um SGBD para armazenar os dados e, pela aplicação no Visual Studio Code, estabelecer a conexão com ela. Vamos lá?

Banco de dados MongoDB

Instalação

Existem várias formas de representar dados em um banco de dados, desde o modelo relacional, orientado a objetos, como também orientado a documento. O MongoDB encontra-se classificado como banco orientado a documento, sendo amplamente utilizado para representação de dados semiestruturados no formato JSON (Java Script Object Notation).

A linguagem na qual você poderá implementar as instruções para recuperação de dados no MongoDB **não** será a linguagem SQL ANSI-ISO utilizada nos bancos de dados relacionais. Ele oferece uma linguagem própria para uso que é tão rica e eficiente quanto a linguagem SQL.

Num banco de dados orientado a documento, não é empregado o conceito de normalização de dados, que é caracterizado pela criação de tabelas e a definição de relacionamento entre elas, obedecendo restrições com o intuito de eliminar redundância dos dados e desperdício do espaço armazenado. Assim, podemos entender o MongoDB como:

Um banco de dados orientado a documento, que emprega a redundância de dados evitando a criação de tabelas relacionais para armazenar dados que serão reutilizados.

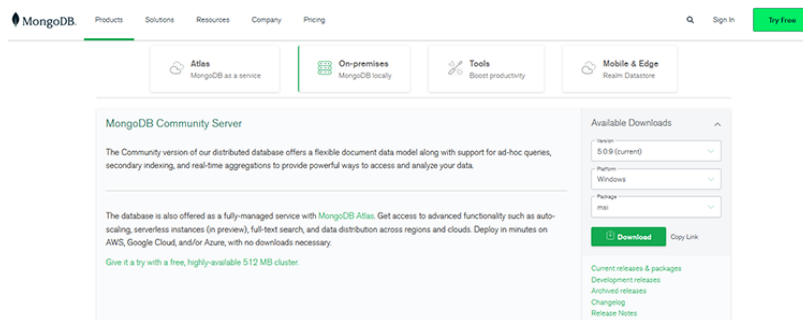
Bem, agora que você já tem uma ideia de como é o MongoDB, precisamos instalá-lo para que ele seja o SGBD que armazenará os dados que utilizaremos em nossa aplicação Node.js no Visual Studio Code.

Há várias opções no portfólio do MongoDB que você poderá utilizar. Optamos pela versão **Community Server**, por ser destinada para desenvolvedores sem custo adicional e por possuir características como consulta *ad hoc*, agregações em tempo real, bem como indexação também dos documentos.

A seguir, escolha a versão, plataforma e tipo de pacote para instalação do MongoDB em sua estação de trabalho:

Ad hoc

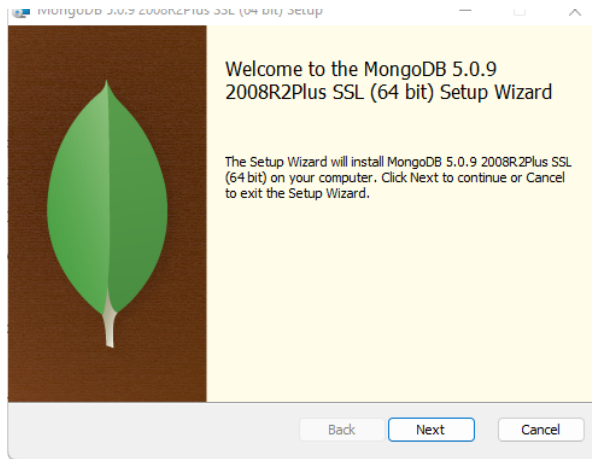
A expressão *Ad hoc* significa “para este propósito”. Ou seja, a consulta é criada apenas para satisfazer aquela necessidade específica, aquele propósito, em um momento específico.



Download do MongoDB.



Após a finalização do download, execute o pacote de instalação do MongoDB que carregará uma assistente de instalação.

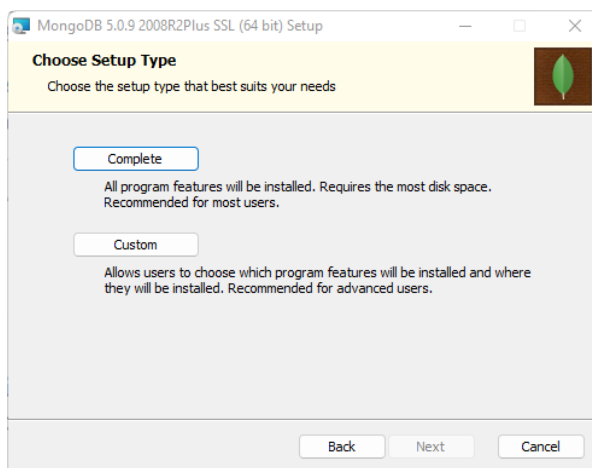


2

Em seguida são apresentados os termos e condições de uso do MongoDB Community Server.

3

Aqui são exibidos os dois tipos de instalação que podem ser realizados. Na instalação **Complete**, todas as funcionalidades presentes no produto serão instaladas; na instalação **Custom**, é possível a personalização dos recursos que devem ser instalados. Na maioria dos cenários, o tipo de instalação Complete atende às necessidades dos desenvolvedores.



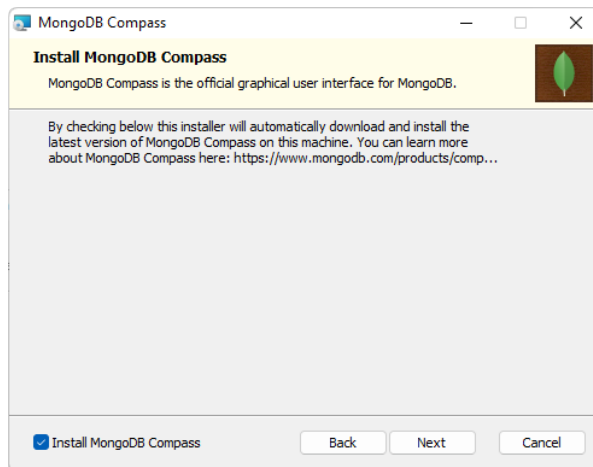
4

Aqui vemos as principais configurações que devem ser feitas na instalação do MongoDB. Como qualquer SGBD, o MongoDB não é diferente e,

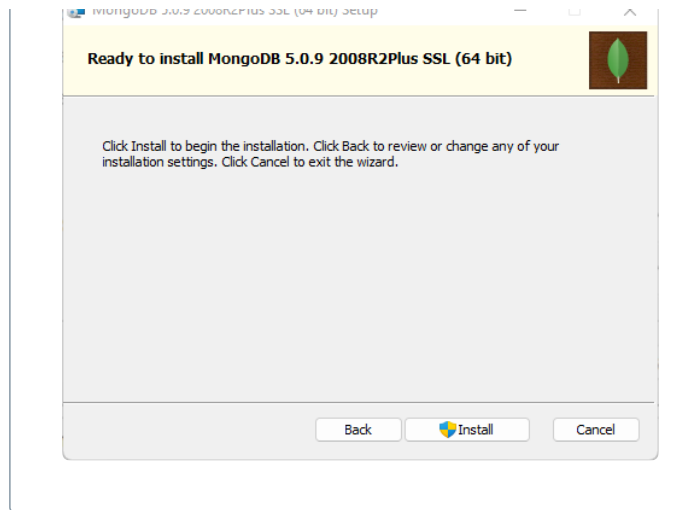
sendo assim, precisa ser instalado como serviço para que esteja executando no sistema operacional de maneira automática. Também é possível definir que conta executará o serviço de banco de dados. Existem **duas opções** possíveis. Por fim, definir a localização dos arquivos de dados e log do servidor do MongoDB. A distribuição desses arquivos em discos físicos sólidos separados consiste na melhor prática, pois favorece geralmente um melhor tempo de execução das consultas.



Neste momento, é importante marcar a opção **MongoDB Compass** por ser a interface gráfica de uso do MongoDB. Caso não seja selecionada, todas as instruções para o SGBD serão via console apenas.



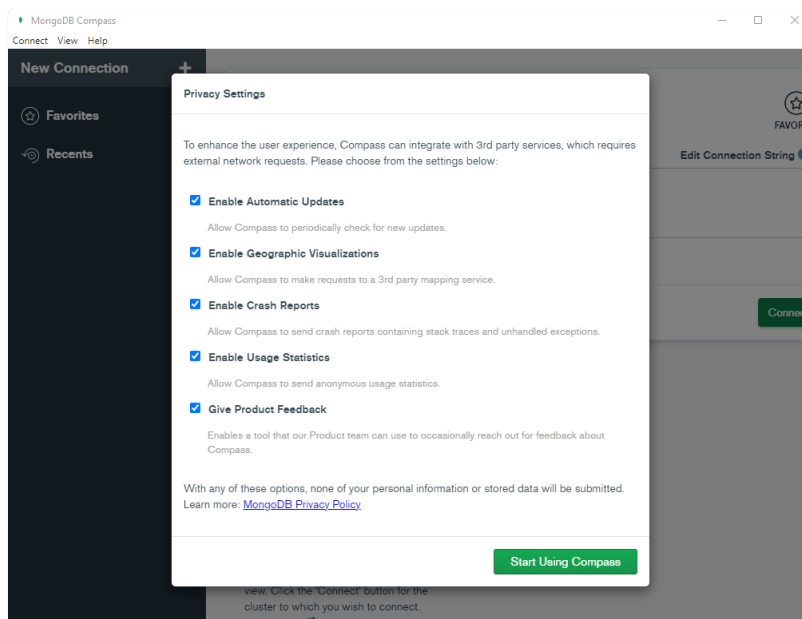
Agora que todas as definições para instalação do MongoDB foram feitas, podemos executar a instalação, de acordo com esta imagem.



Duas opções

Se a sua escolha for em uma instalação local em sua estação de trabalho, a opção **Run Service as Network Service user** atenderá sem qualquer problema. No entanto, se desejar instalá-lo em uma máquina específica que atuará como servidor de banco de dados, a opção, **Run Service as a local or domain user** deverá ser usada e configurada com as credenciais de uma conta de seu sistema de diretório.

Ao término da instalação, será apresentada uma tela semelhante ao que é exibido na imagem seguinte, na qual você poderá optar em definir algumas configurações para uso do MongoDB. Clique em **Start using Compass** para proceder com a utilização da interface gráfica.



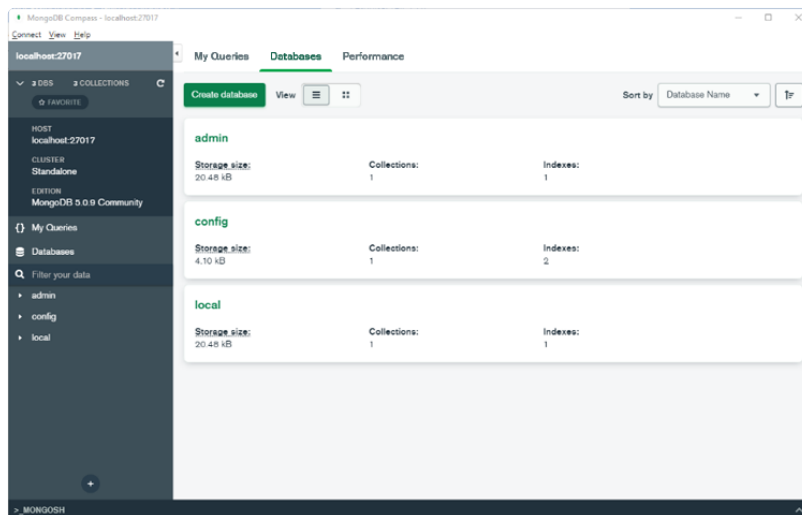
Configurações pós-instalação do MongoDB.

Criação de um banco de dados

Para conectar ao SGBD MongoDB que acabamos de instalar, você precisará selecionar no menu **Connect** a opção **New Connection**. Será mostrado o endereço de acesso, bem como a porta de comunicação com o banco de dados. Clique em **Connect** para estabelecer a conexão.

Criando uma conexão com o MongoDB.

Agora, vamos criar um banco de dados para que possamos armazenar os documentos que serão criados. Para isso, clique na opção **Databases** e, em seguida, **Create Database**, conforme esta imagem:



Tela de criação de banco de dados.

Informe o nome do banco de dados e coleção que será usada para armazenar os itens do documento. Por fim, clique em **Create Database** para que sejam criados.

×

Create Database

Database Name

Lojaweb

Collection Name

Produto

Advanced Collection Options (e.g. Time-Series, Capped, Clustered collections)

Cancel

Create Database

Criando um banco de dados e coleção.

JSON

Podemos definir que JSON é um formato baseado em texto, leve e que é usado na troca de dados entre cliente e servidor. O JSON é derivado do

JavaScript e, portanto, é muito semelhante aos objetos JavaScript. Também ele se torna independente de linguagem, e todas as principais linguagens suportam esse formato, tal como: C#, PHP, Java, C++, Python e Ruby. O JSON pode ser utilizado em aplicativos da Web para transferir dados.

Antes do JSON, o formato XML é usado para fazer esse intercâmbio de dados entre cliente e servidor. A análise XML requer uma implementação específica do lado do cliente, chamada de XML DOM para obter a resposta XML e, em seguida, consultar a resposta usando XPath para acessar e recuperar os dados. Isso aumenta a complexidade do desenvolvimento do código, porque as consultas de dados precisam ser executadas em dois níveis:

Primeiro

No lado do servidor, consultando os dados do banco de dados.

Segundo

No lado do cliente, usando XPath.

A grande vantagem do uso do JSON se concentra nessa questão, pois não requer nenhuma implementação específica; o mecanismo JavaScript no navegador lida com a análise JSON.

Como você pode notar, o JSON torna a vida do desenvolvedor mais tranquila do ponto de vista da comunicação cliente-servidor. O banco de dados orientado a documentos MongoDB armazena os dados nesse formato. A estrutura do formato JSON consiste em uma coleção de pares (chave e valor), podendo ser caracterizado por um objeto ou registro, por exemplo. Um item da coleção deve ser delimitado por chaves. Cada chave é seguida pelo caractere ':' (dois pontos), na sequência, o valor a ser associado, conforme ilustra o exemplo abaixo:





Exemplo de registros.

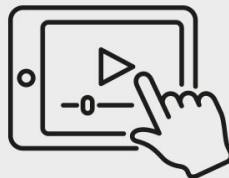
Bastante interessante mesmo é a simplicidade que o formato JSON oferece no armazenamento de dados. Pois bem, neste módulo você conheceu os principais conceitos de um sistema gerenciador de banco de dados (SGBD), CRUD e JSON. Já no próximo módulo, vamos instrumentalizar um código-fonte para comunicar com o banco de dados e realizar as operações do CRUD. Mas antes disso, assista ao vídeo a seguir!



Criação de um banco de dados no MongoDB

Veja como é criado um banco de dados no MongoDB.

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.



Falta pouco para atingir seus objetivos.

Vamos praticar alguns conceitos?

Questão 1

Analise as afirmativas sobre as operações CRUD:

I. As operações de um CRUD consistem em realizar a implementação das estruturas para armazenamento de dados em um banco de dados.

Porque

II. Permite que a aplicação desenvolvida possa recuperar os dados demandados de forma mais eficiente, segura e íntegra em banco de dados.

A

I é verdadeira, II é verdadeira e II justifica a I.

B

I é verdadeira, II é verdadeira, porém II não justifica a I.

C

I e II são falsas.

D

I é falsa e II é verdadeira.

E

I é verdadeira e II é falsa.

Parabéns! A alternativa D está correta.

O acrônimo CRUD (Create, Read, Update, Delete) representa as operações de consultas e recuperação de dados em um SGBD e, portanto, não apenas a definição de estrutura para o seu armazenamento.

Questão 2

Analise as afirmativas sobre dados estruturados e semiestruturados:

I. Dados estruturados apresentam uma flexibilidade na representação dos dados no registro de um banco de dados.

Porque

II. Ao utilizar a abordagem de armazenamento de dados semiestruturados, estamos definindo uma estrutura de atributos fixos para todos os registros a serem inseridos.

A

I é verdadeira, II é verdadeira e II justifica a I.

B

I é verdadeira, II é verdadeira, porém II não justifica a I.

C

I e II são falsas.

D

I é falsa e II é verdadeira.

E

I é verdadeira e II é falsa.

Parabéns! A alternativa C está correta.

Os dados estruturados, como o próprio nome sugere, apresentam uma estrutura fixa de campos que todos os registros a serem inseridos deverão usar. No entanto, quando é usada uma abordagem de dados semiestruturados, temos uma flexibilidade nessa estrutura porque cada registro pode possuir os atributos que forem necessários.



2 - Operações de banco de dados

Ao final deste módulo, você será capaz de implementar as operações de consulta, inserção, alteração e exclusão de banco de dados com o MongoDB.

Visual Studio Code vs. MongoDB

Instalação de extensão

Para que você possa representar os dados corretamente no banco de dados, é necessário levantamento prévio dos requisitos e modelagem de dados, sendo indispensável a validação com o cliente que demandou o sistema de informação. Muitos problemas podem ser mitigados se na fase inicial do projeto o time de desenvolvedores tiver essa preocupação. O custo de manutenção de sistema sempre é mais oneroso quando ele se encontra em produção.

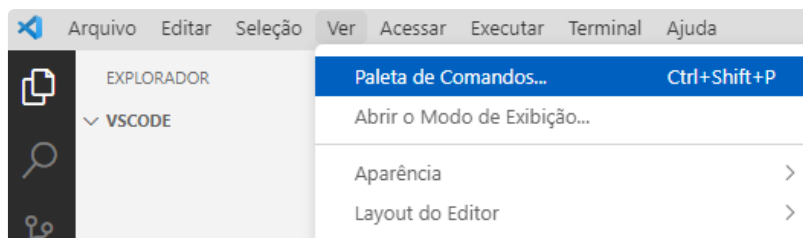
Agora, vamos carregar o Visual Studio Code e instalar a extensão do MongoDB para que possamos trabalhar conectados com o servidor de banco de dados.

Essa funcionalidade de extensões é uma característica notável do VS Code, pois é possível customizar o nosso projeto com os recursos necessários para seu funcionamento. Pressione **CRTL+SHIFT+X** para que seja mostrada uma caixa de pesquisa para instalação ou desinstalação de extensões. Então, para instalar a extensão do MongoDB, digite **mongodb**.



Instalando a extensão do MongoDB.

Com a extensão instalada do MongoDB, podemos na IDE do Visual Studio Code executar comandos do próprio MongoDB, por meio da paleta de comandos. Portanto, para isso, selecione no menu Ver, a opção Paleta de comandos, conforme imagem abaixo:



Acessando a Paleta de comandos.

A seguir, visualizamos os principais comandos para iniciar suas atividades com o MongoDB no VS Code. Ah! Mas antes de criarmos a conexão com o banco de dados que utilizaremos para realizar as operações do CRUD, precisamos configurar no SGBD de que forma será feita a autenticação das credenciais de acesso.

Principais comandos do MongoDB.

Configuração de uma nova conexão

Retorne ao servidor do MongoDB e criei uma nova conexão. Clique na opção **Advanced Connection Options** para definir o esquema da string de conexão a ser usado pela aplicação para conectar com o banco de dados.

Para executar uma instância instalada localmente do MongoDB em sua estação de trabalho que utiliza a **porta padrão 27017**, a string de conexão será: **mongodb://localhost:27017?authMechanism=DEFAULT**.

O método de autenticação default não exige usuário nem senha para estabelecer a conexão com o servidor de banco de dados. Já quando optamos pelo esquema **mongodb+srv**, desejamos estabelecer uma conexão externa para um servidor de banco de dados e, para isso, o sufixo **srv** indica ao cliente que o nome do host possui um registro do tipo SRV em um servidor de resolução de nomes(DNS). A principal vantagem de utilizar esse tipo de conexão é que permite mais flexibilidade de implantação e a capacidade de alterar os servidores em modificações no lado do cliente. Clique no botão **Connect** para finalizar a configuração, conforme imagem a seguir:

New Connection

Connect to a MongoDB deployment



FAVORITE

URI ⓘ

Edit Connection String

mongodb://localhost:27017/

Advanced Connection Options

General

Authentication

TLS/SSL

Proxy/SSH

In-Use Encryption

Advanced

Authentication Method

None

Username/Password

X.509

Kerberos

LDAP

AWS IAM

Connect

Definindo a string de conexão do MongoDB.

Conectado ao servidor do MongoDB, acesse o MongoShell para que possamos definir um usuário e senha para ser usado na autenticação do banco de dados pela aplicação. Esse ambiente suporta tanto Javascript como node.js e pode ser usado para interagir diretamente com o MongoDB de forma programática.

A instrução **use admin** define que trabalharemos no banco de dados admin. Para configuração que seja possível, realizamos a criação do usuário e senha. Observe que o prompt do mongo shell muda para **admin>** após a execução da instrução. Na imagem a seguir, utilizaremos o método **db.createUser** para criação de um novo usuário, sendo necessário informar os parâmetros para execução dessa atividade, que são:

User

Nome do usuário que será criado.

Pwd

Senha que será utilizada pelo usuário para autenticação.

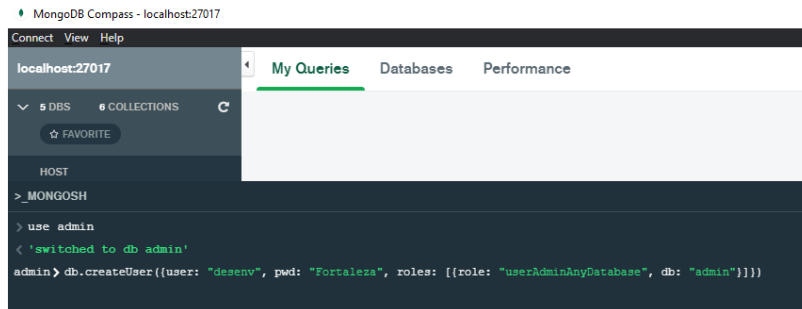
Roles

São funções concedidas ao usuário para realização de ações no banco de dados.

Se for especificado um vetor vazio - [], nenhum papel é aplicado ao usuário que está sendo criado. No entanto, há funções prontas disponibilizadas pelo MongoDB que podem ser usadas e que apresentam permissões já definidas. Também lhe é facultada a possibilidade de criar suas próprias funções de acordo com a gestão de

acesso de segurança que você queria estabelecer para seu banco de dados pelo usuário conectado.

A role **userAdminAnyDatabase** possibilita que o usuário possa realizar qualquer ação independentemente do banco de dados. Uma boa prática que pode ser usada para o usuário e que será usada pela aplicação é a role **ReadWrite**, pois contempla todas as operações necessárias em um CRUD.



```
MongoDB Compass - localhost:27017
Connect View Help
localhost:27017
My Queries Databases Performance
DBS COLLECTIONS
FAVORITE
HOST
> _MONGOSSH
> use admin
< 'switched to db admin'
admin> db.createUser({user: "desenv", pwd: "Fortaleza", roles: [{role: "userAdminAnyDatabase", db: "admin"}]})
```

Criando um usuário no MongoDB.

Depois de confirmar o comando **db.createUser** com enter, pressione CTRL+D para finalizar a conexão em uso a fim de fazer alteração para autenticação por usuário e senha. Agora, crie uma nova conexão e, em seguida, na opção **Advanced Connection Options**, mude para aba **Authentication**. Informe no campo **username** e **password** o nome do usuário e senha criados respectivamente. O mecanismo de autenticação default pode ser usado para início dos seus estudos, veja:

Configurando o usuário e senha para autenticação.

No entanto, para contexto corporativo de um servidor MongoDB de produção, é importante considerar o mecanismo SCRAM-SHA, que pode ser:

SCRAM-SHA-1

Padrão IETF, RFC 5802, que estabelece métodos de autenticação que verificam os dados fornecidos do usuário e senha, bem como banco de dados.

SCRAM-SHA-256

Padrão RFC 7677, que armazena no servidor a senha criptografada por meio de uma função hash e que impede

conexões não confiáveis.

Logo, é recomendável o uso do mecanismo de autenticação SCRAM-SHA-256 para ambiente de produção.

Retorne à aba **General**, e verifique que a URI da string de conexão foi modificada de acordo com as configurações realizadas no passo anterior, conforme apresentado na próxima imagem. Agora, clique no botão **Connect** para estabelecer a conexão com a configuração feita.

New Connection FAVORITE

Connect to a MongoDB deployment

URI ⓘ Edit Connection String

mongodb://desenv:Fortaleza@localhost:27017/?authMechanism=DEFAULT&directConnection=true

▼ Advanced Connection Options

General Authentication TLS/SSL Proxy/SSH In-Use Encryption Advanced

Connection String Scheme

mongodb mongodb+srv

Standard Connection String Format. The standard format of the MongoDB connection URI is used to connect to a MongoDB deployment: standalone, replica set, or a sharded cluster.

Host

localhost:27017 +

☒ **Direct Connection**
Specifies whether to force dispatch all operations to the specified host.

Connect

String de conexão com autenticação com usuário e senha.

No Visual Studio Code, no menu Ver, selecione Paleta de Comandos e digite mongodb. Escolha a opção **MongoDB:Connect with Connection String**, para usar a string de conexão criada no passo anterior.

Paleta de comandos do MongoDB.

Agora, informe a string de conexão com usuário e senha para estabelecer a conexão com o servidor do MongoDB, veja:

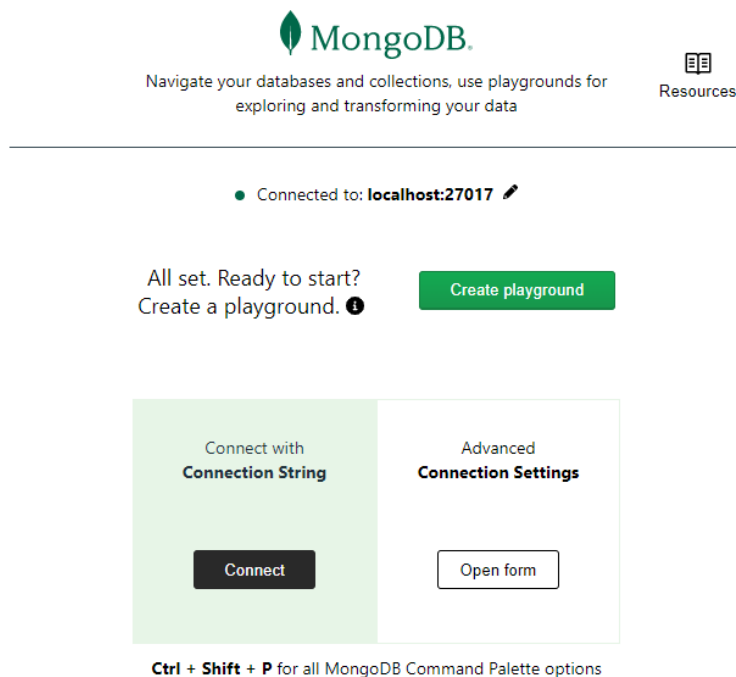
Ver Acessar Executar Terminal Ajuda VsCode - Visual Studio Code

mongodb://desenv:Fortaleza@localhost:27017/?authMechanism=DEFAULT&directConnection=true

Enter your connection string (SRV or standard) (Pressione 'Enter' para confirmar ou 'Escape' para cancelar)

String de conexão do MongoDB no VS Code.

Se a conexão com o banco de dados MongoDb foi realizada com sucesso, uma tela será apresentada de forma semelhante ao que é visto nesta imagem:



Conexão ao Mongo DB.

Você pode observar que no VS Code a conexão foi realizada com sucesso e são apresentados os três bancos de dados que o MongoDB possui como padrão. São eles:

● —————

admin

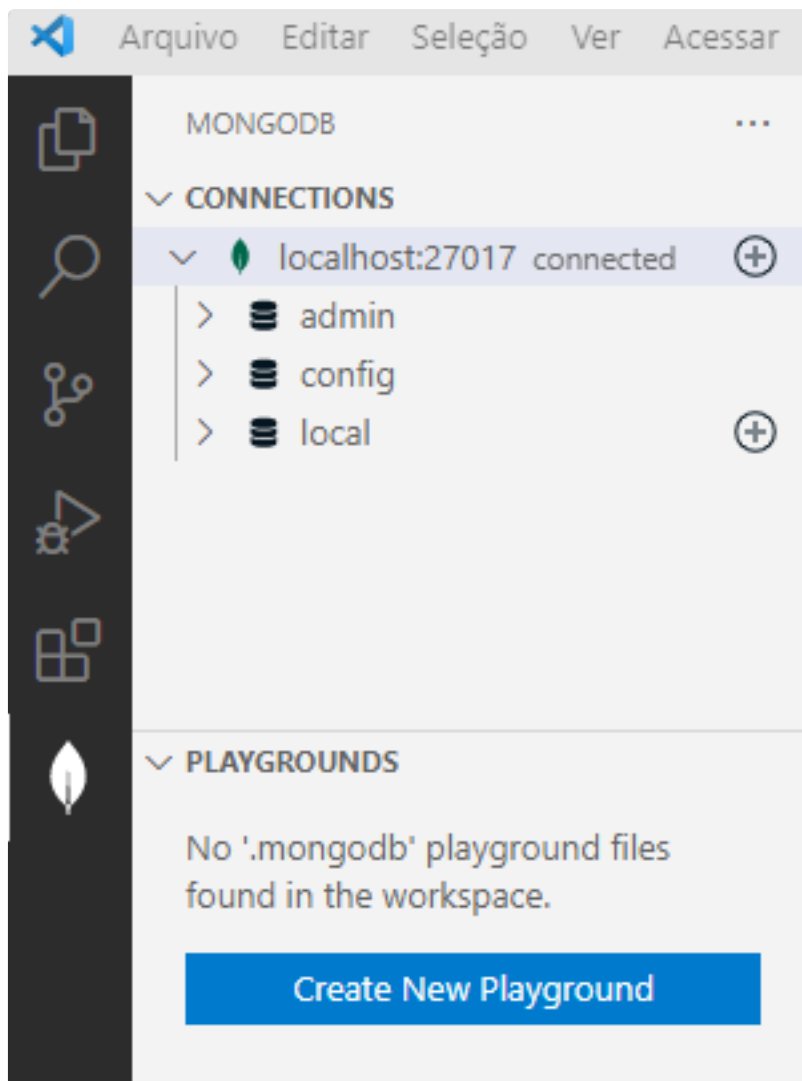
● —————

config

●

local

Veja na opção **Mostrar MongoDB** na referida paleta:

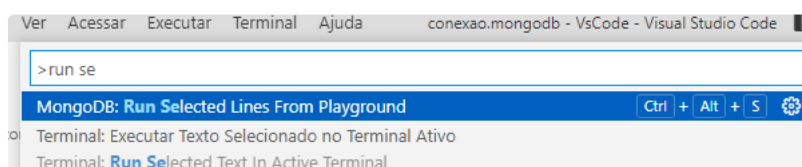


Definindo a string de conexão do MongoDB.

Inserção de dados no MongoDB

Criação de um banco de dados e coleção

Agora, no menu Ver, escolha a opção de Paleta de comandos e, na caixa de busca, digite **MongoDB:run Selected Lines From Playground**, de acordo com o que é exibido na imagem abaixo:



Executando comandos do MongoDB.

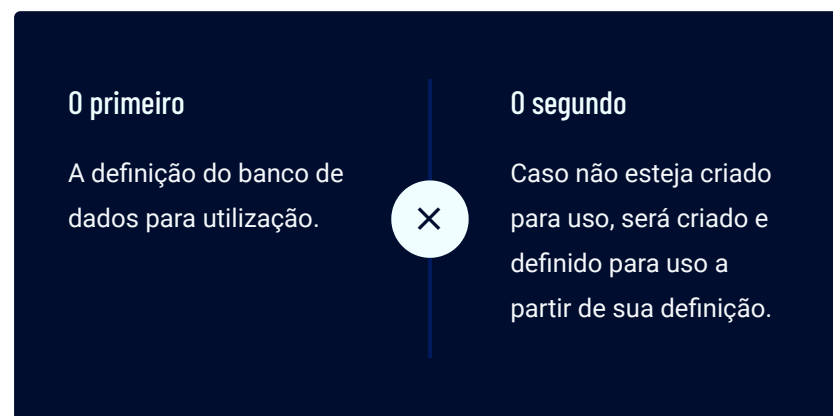
Vamos elaborar o código para conexão com o banco de dados e coleção de documentos que utilizaremos para demonstrar essa funcionalidade. Quando desejamos comentar as instruções contidas no

código, podemos usar a barra invertida duplamente e adicionar o texto conforme consta nas linhas 1, 6 e 9. Sempre a documentação é recomendável no desenvolvimento do código, além de mostrar, organizar e explicar o que foi feito e o porquê. Temos ainda menor tempo de manutenção por outros desenvolvedores que inicialmente não trabalhavam no código, visto que esse procedimento torna mais amigável o entendimento do código.

Nas linhas 3 e 4, foram definidas duas constantes chamadas `database` e `collection`, respectivamente.

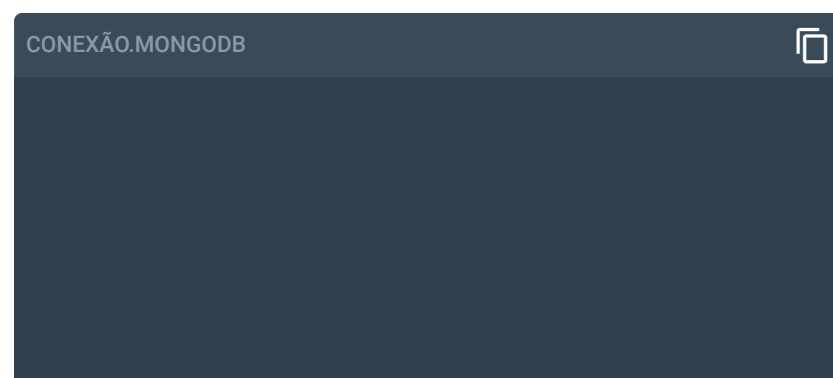
O principal objetivo de usarmos constantes no código-fonte é porque podemos usá-las diversas vezes no programa e teremos apenas um local de alteração.

É desejável que as constantes sejam criadas inicialmente no código-fonte. Na linha 7, ao empregarmos a instrução `use(database)`, podemos ter dois comportamentos distintos, são eles:



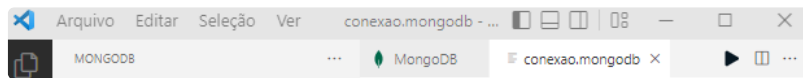
Já na linha 10, por meio do método `CreateCollection` podemos definir que desejamos criar uma coleção para o banco de dados LojaWeb, chamada de Produto.

O único parâmetro obrigatório para esse método é o nome da própria coleção que desejamos criar, porém, é possível definir com o parâmetro opções de como será criada a nova coleção, alocando, por exemplo, o tamanho máximo de bytes e o número máximo de documentos permitido na coleção. A seguir, ilustramos a codificação realizada:



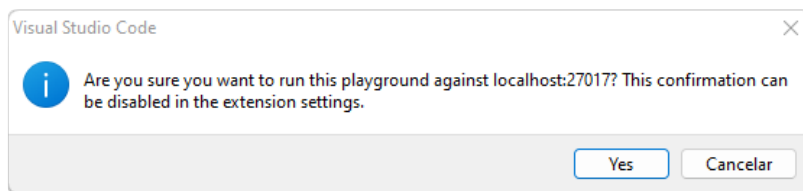
Criar banco de dados.

Para executar o código desenvolvido, é necessário clicar no botão **play** para que possa ser rodado, veja:



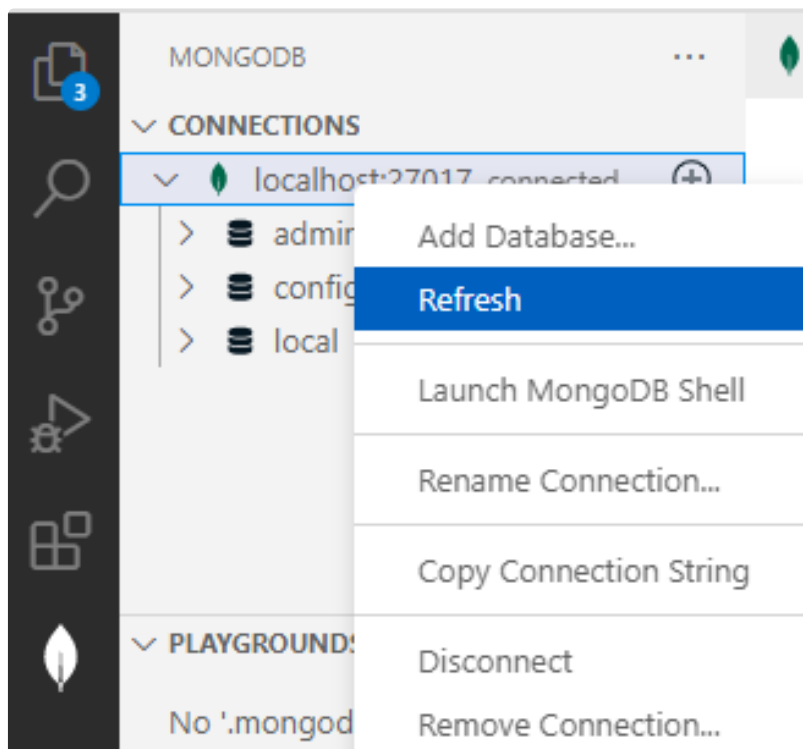
Criando banco de dados e coleção no MongoDB.

Será apresentada uma tela de confirmação perguntando se realmente você deseja executar esse script no VS Code. Pressione o botão **Yes**, de acordo com a seguinte imagem:



Tela de confirmação de execução do script.

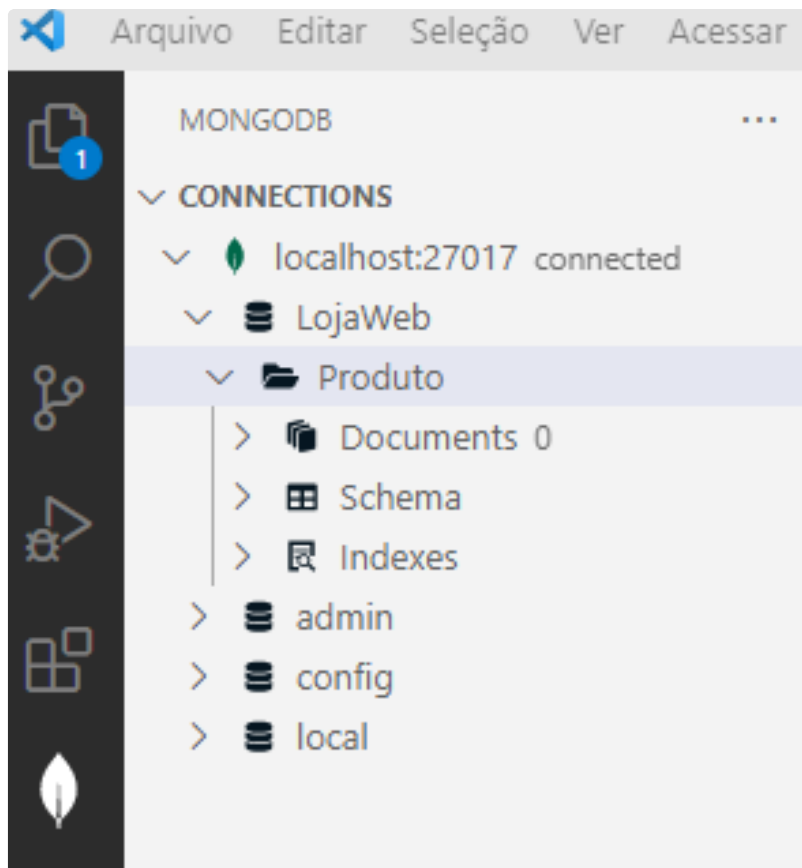
Para verificar se de fato o banco de dados e a coleção foram criados, clique com o botão direito na conexão do banco de dados feita e selecione a opção **Refresh** para atualizar a lista dos bancos de dados da conexão:



Atualizando a lista de banco de dados.

Selecione o banco de dados LojaWeb, expanda a coleção Produto e note que serão apresentados a estrutura de documentos, esquema e índices,

veja:



Validando a criação do banco de dados e coleção.

Operação de inclusão

Como temos o banco de dados criado e a coleção, chegou o momento de definirmos o documento que armazenará os dados que precisamos. Para essa ação, podemos usar dois métodos para inserção de documentos na coleção Produto, que são:

Primeiro método

`InsertOne()`.

Segundo método

`InsertMany()`.

O método **`InsertOne()`** insere na coleção apenas um documento, mesmo que você tenha adicionado vários documentos dentro da instrução. Na imagem a seguir, na linha 1, inicialmente definimos qual banco de dados desejamos utilizar e, na linha 2, precisamos informar a coleção que queremos a adição de apenas um documento em virtude do método `InsertOne()`. Na linha 3, precisamos especificar usando a notação JSON, portanto, o par chave e valor para informar o atributo e o valor associado

a ele para o documento. Os atributos podem ser do tipo inteiro, texto e data, por exemplo.

Na imagem a seguir, observamos que, ao executar, temos o resultado da operação com um identificado criado pela inserção realizada na coleção, portanto, linha 4.

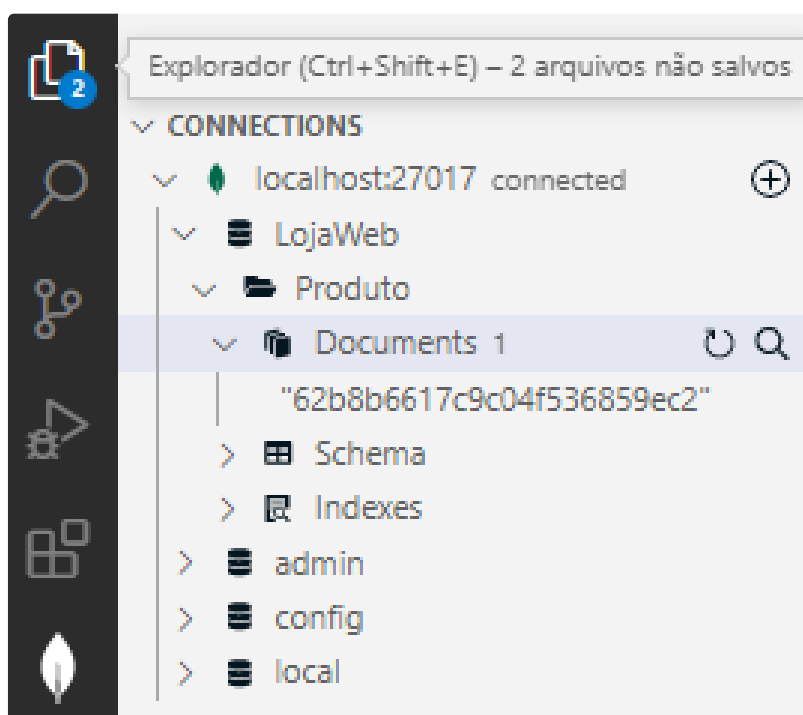
Playground Result ×

Include Import Statements | Include Driver Syntax

```
1 {  
2   "acknowledged": true,  
3   "insertedId": {  
4     "$oid": "62b8b6617c9c04f536859ec2"  
5   }  
6 }
```

Confirmação da inserção do documento.

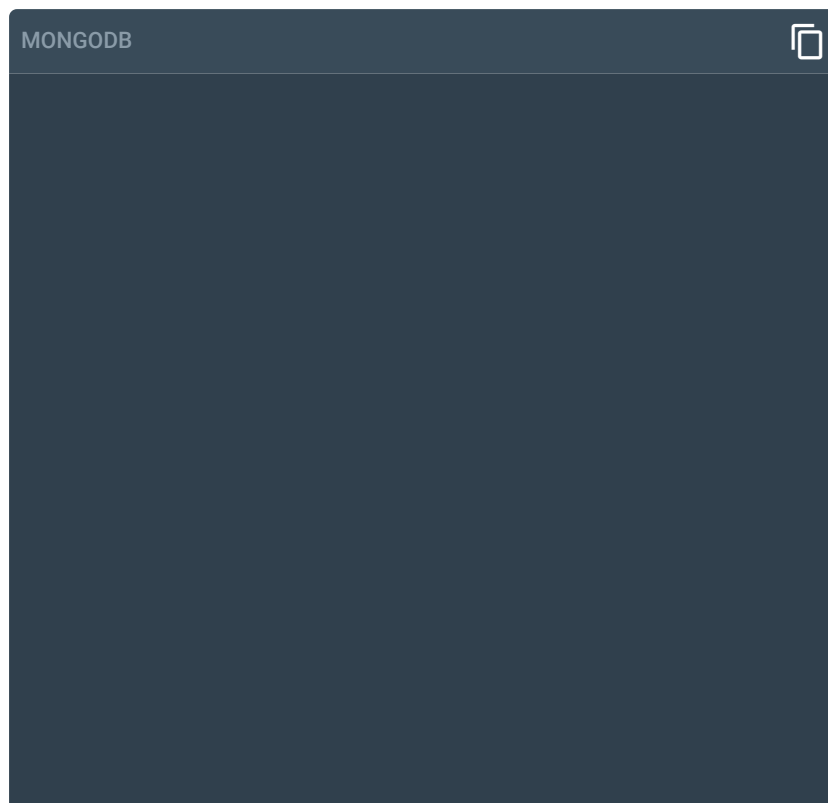
Atualizando a nossa coleção, podemos perceber que o documento foi criado, veja:



Validando documento inserido na coleção.

O método **InsertMany()** possibilita a inclusão de vários documentos em uma mesma instrução que será submetida ao banco de dados. A grande vantagem dessa abordagem consiste na redução de acesso ao disco para realizar a gravação física dos dados. A seguir, temos quatro documentos que estamos inserindo na coleção Produto. Isso significa que apenas um acesso será feito ao disco do servidor para gravação de todos os documentos ao invés de quatro. Pensando em milhares de operações e usuários, essa estratégia proporciona melhor desempenho

e tempo de resposta para inserção de documentos do que teríamos ao usar o método `InsertOne()` para cada documento.



Inclusão de vários documentos.

Na imagem a seguir, observamos que, ao executar o comando `insertMany`, temos o resultado da operação com um identificador criado para cada inserção realizada na coleção.

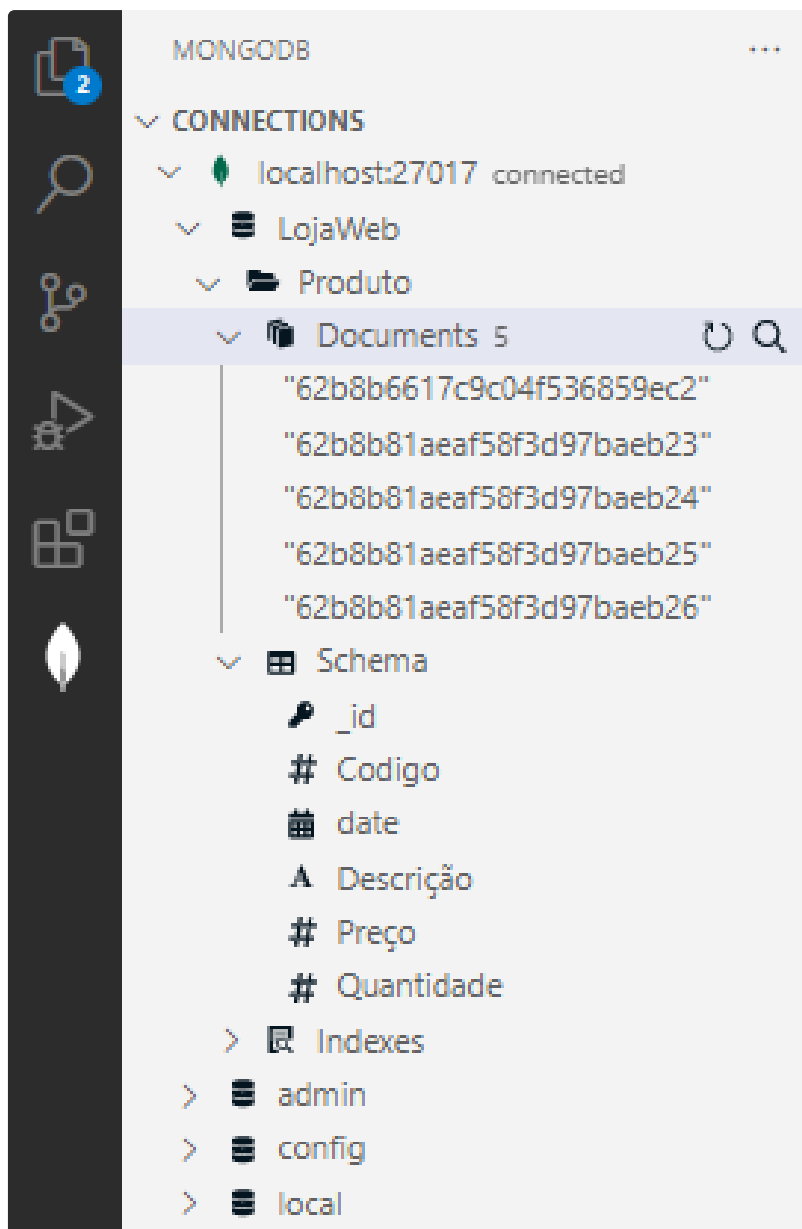
() Playground Result ×

Include Import Statements | Include Driver Syntax

```
1 {
2   "acknowledged": true,
3   "insertedIds": {
4     "0": {
5       "$oid": "62b8b81aeaf58f3d97baeb23"
6     },
7     "1": {
8       "$oid": "62b8b81aeaf58f3d97baeb24"
9     },
10    "2": {
11      "$oid": "62b8b81aeaf58f3d97baeb25"
12    },
13    "3": {
14      "$oid": "62b8b81aeaf58f3d97baeb26"
15    }
16  }
17 }
```

Confirmação da inserção de documentos.

Atualizando a nossa coleção, podemos perceber que os documentos foram inseridos e, também, podemos verificar o esquema da estrutura usado nos documentos, veja:



Validando a inserção de documentos.

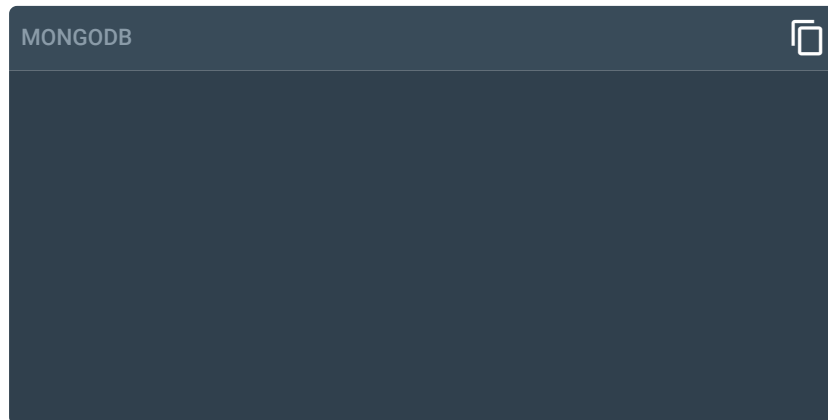
Muito legal, não é mesmo? Neste módulo, você aprendeu como fazer a conexão com o servidor de banco de dados MongoDB, criar seu primeiro banco de dados e também viu uma coleção para serem adicionados documentos. Vamos explorar outras operações do CRUD nos próximos módulos.

Operações de consulta

Método `getCollection()`

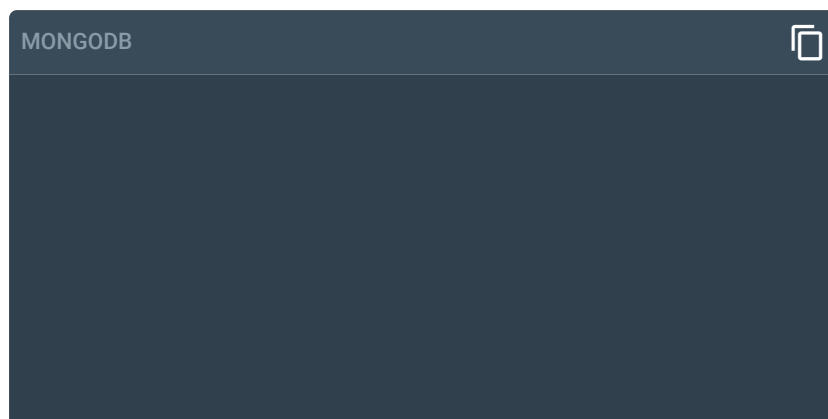
Uma vez que foram criados o banco de dados, uma coleção e inseridos documentos, podemos executar consultas no intuito de recuperar dados de acordo com os parâmetros informados na aplicação.

Para isso, podemos utilizar o método **`getCollection()`**, que pode retornar uma coleção inteira informando o nome dela, conforme exibido abaixo:



Retornar.

Ao utilizarmos o método `find()`, é possível retornar documentos com base em parâmetros informados. Caso não seja informado nenhum parâmetro, o resultado será a lista completa de todos os documentos de uma coleção. No código a seguir, na linha 1, inicialmente definimos qual banco de dados que desejamos utilizar e, na linha 2, precisamos informar a coleção que queremos recuperar todos os documentos com o **método `find()`**.



Encontrar.

O resultado esperado para execução da instrução do passo anterior deve ser algo semelhante ao que é apresentado na imagem a seguir:

Listando todos os documentos de uma coleção.

Os parâmetros que podem ser informados para o **método `find()`** são:

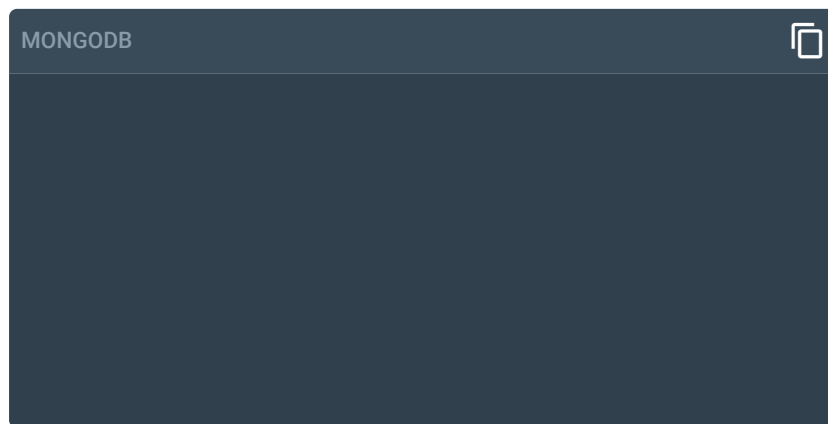
Consulta

Especifica os critérios de seleção, recuperando todos os campos do documento obrigatoriamente.

Projeção

Especifica os campos que deverão ser retornados do documento.

Ainda em relação ao parâmetro consulta, podemos visualizar melhor o seguinte código:



Consulta.

Método aggregate()

Também é possível realizarmos operações de processar muitos documentos e que possam agrupar em um único resultado. Para essa finalidade, utilizamos o **método aggregate()**. Na linha 3, definimos uma constante que recebe o valor agregado do TotalVendasProdutos pela operação de agrupamento feita na linha 5.



Processar documentos.

O resultado é ilustrado abaixo:

Resultado de agregação de dados.

Até aqui, você pôde conhecer qual método deve ser usado para recuperação dos dados nos documentos armazenados. Agora, vamos aprender como atualizar e excluir documentos.

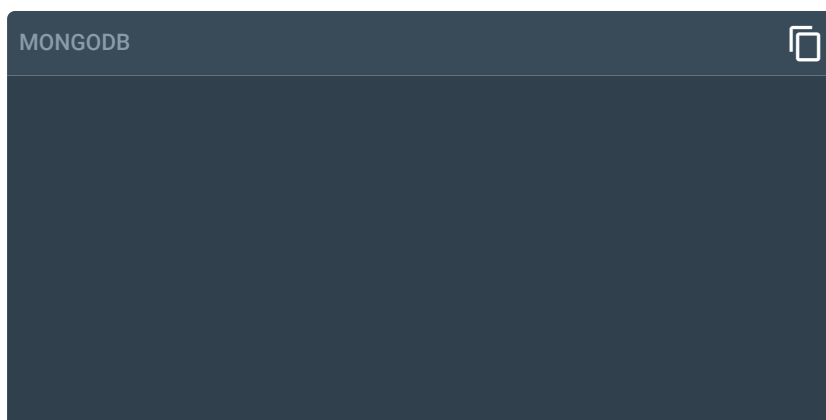
Operações de alteração

Método updateOne()

Uma vez que foi criada a coleção e documentos foram inseridos, podemos proceder com atualização ou exclusão dos campos, conforme a necessidade demandada pela aplicação.

O método `updateOne()`, ao ser utilizado, modificará apenas um único documento de acordo com o parâmetro informado. No código a seguir, na linha 1, definimos qual banco de dados desejamos utilizar e, na linha 2, a coleção que usaremos para atualizar com o método `updateOne()` que modificará o primeiro documento que atenda ao critério da consulta apresentada.

É importante ressaltar que mesmo sendo executada uma atualização no documento, o valor do campo **_id permanece** o mesmo de quando foi inserido. Na linha 4, utilizamos uma palavra-chave **\$inc** para incrementar o valor da quantidade do produto que apresenta código 1.



Atualização produto 1.

Na imagem seguinte, podemos verificar o resultado esperado na execução da atualização feita na etapa anterior.

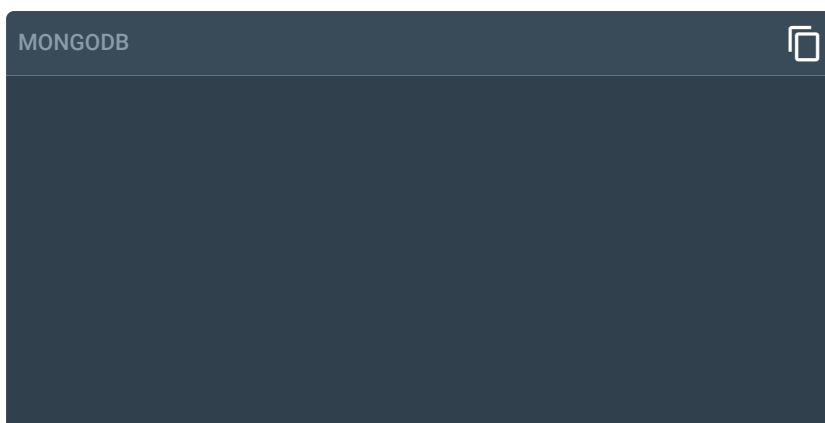
 Playground Result 

Include Import Statements | Include Driver Syntax

```
1 {
2   "acknowledged": true,
3   "insertedId": null,
4   "matchedCount": 1,
5   "modifiedCount": 1,
6   "upsertedCount": 0
7 }
```

Confirmação da atualização do documento.

Agora, se a sua necessidade de alteração do campo for a substituição, você deverá usar a palavra-chave **\$set**, a fim de executar essa ação. Na linha 4, com a palavra-chave **\$set** está sendo modificado o valor original da quantidade do produto de código 2 para o valor 40, de acordo com o que é mostrado no código abaixo:



Atualização produto 2.

A próxima imagem ilustra o resultado esperado na execução da atualização feita na etapa anterior.

 Playground Result 

Include Import Statements | Include Driver Syntax

```
1 {
2   "acknowledged": true,
3   "insertedId": null,
4   "matchedCount": 1,
5   "modifiedCount": 1,
6   "upsertedCount": 0
7 }
```

Confirmação da atualização do documento.

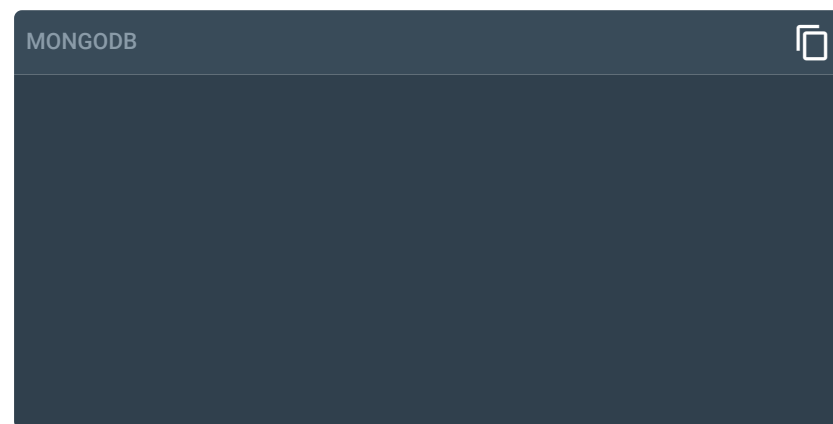
Método updateMany()

Certo! Mas se eu precisar modificar vários documentos de única vez, como posso fazer?

Resposta

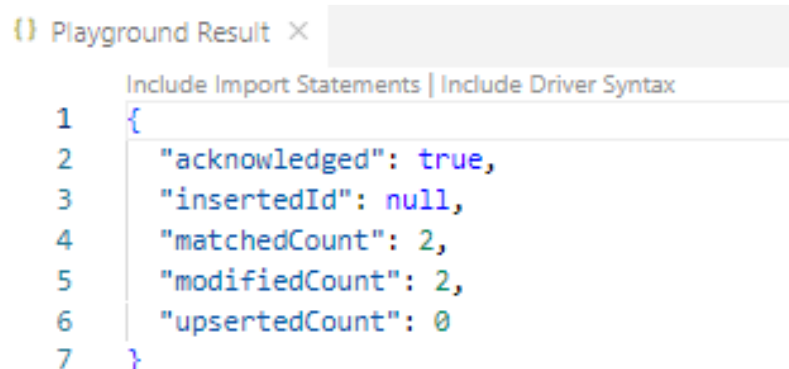
Nessa situação, você deverá fazer uso do método `updateMany()`.

No código a seguir, na linha 1, inicialmente definimos qual banco de dados desejamos utilizar e, na linha 2, a coleção que usaremos para atualizar com o método `updateMany()`. Com o critério de busca na linha 3, ou seja, que o preço do produto seja 2, atualizaremos adicionando cinco unidades a todos os documentos que satisfaçam o critério de busca.



Atualização de preço.

A próxima imagem mostra o resultado esperado na execução da atualização feita na etapa anterior.

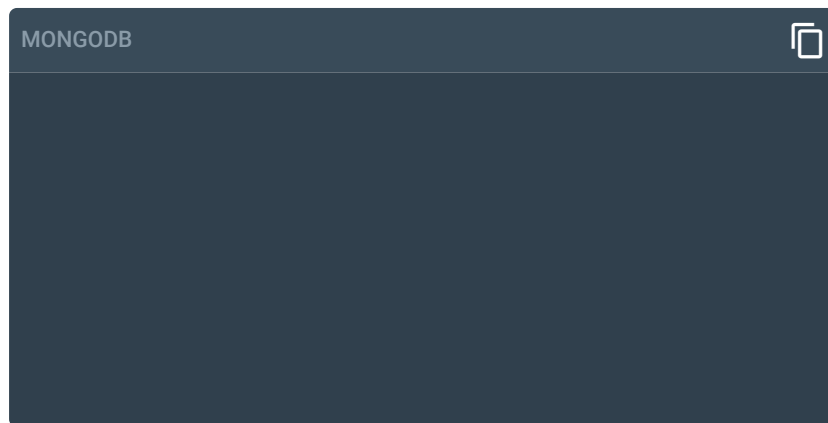


Confirmação da atualização de 2 documentos.

Operações de exclusão

Método `deleteOne()`

Por fim, caso seja necessário excluir um documento, você poderá usar o **método deleteOne()**. Ele deletará o primeiro documento da coleção que corresponder ao critério de seleção informado como parâmetro. No próximo código, na linha 1, definimos qual banco de dados desejamos utilizar e, na linha 2, a coleção que usaremos para excluir com o método deleteOne() passando o critério de seleção em que o código do produto seja 5. Dessa forma, o primeiro documento que atender a esse critério será excluído.



Deletar documento.

A imagem a seguir mostra o resultado esperado na execução da atualização feita na etapa anterior.

```
() Playground Result X
Include Import Statements | Include Driver Syntax
1  {
2    "acknowledged": true,
3    "deletedCount": 1
4  }
```

Confirmação da exclusão de um documento.

Método deleteMany()

Semelhante aos métodos InsertMany() e UpdateMany(), temos também o método deleteMany() para exclusão de vários documentos que atendam ao critério de seleção apresentado.

No código a seguir, na linha 1, inicialmente definimos qual banco de dados desejamos utilizar e, na linha 2, a coleção que usaremos para excluir com o método deleteMany() passando o critério de seleção em que o preço do produto seja 2. Dessa forma, todos os documentos que corresponderem a esse critério serão excluídos.





Definir banco de dados.

A próxima imagem ilustra o resultado esperado na execução da atualização feita na etapa anterior.

 Playground Result 

```
1 {  
2   "acknowledged": true,  
3   "deletedCount": 2  
4 }
```

Confirmação da exclusão de 2 documentos.

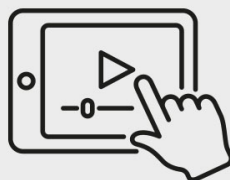
Neste módulo, você conheceu os métodos relacionados às operações de atualização, bem como exclusão de documentos de determinada coleção. Mas antes de concluirmos, assista ao vídeo!



Inserção de dados no MongoDB

Veja mais sobre inserção de dados no MongoDB.

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.



Falta pouco para atingir seus objetivos.

Vamos praticar alguns conceitos?

Questão 1

Analise as afirmativas sobre a inserção de dados no MongoDB:

I. O método `insertOne()` inserirá apenas um documento, mesmo que seja especificado mais de um documento em sua instrução.

Porque

II. Se você não adicionar o campo `_id` no documento na inserção, o MongoDB adicionará esse campo com o intuito de garantir a unicidade do documento na coleção.

A

I e II são verdadeiras e II justifica a I.

B

I e II são verdadeiras, porém II não justifica a I.

C

I e II são falsas.

D

I é falsa e II é verdadeira.

E

I é verdadeira e II é falsa.

Parabéns! A alternativa A está correta.

Pelo simples fato de usar o método `insertOne()`, ele apenas inserirá o primeiro documento independentemente de que haja outros passados como parâmetro.

Questão 2

Analise as afirmativas sobre os operadores de exclusão:

I. O método `deleteMany()` removerá todos os documentos de determinada coleção que atendem ao critério de seletividade estabelecido.

Porque

II. Dessa forma, é possível deletar mais de um documento de uma única vez.

A

I e II são verdadeiras e II justifica a I.

B

I e II são verdadeiras, porém II não justifica a I.

C

I e II são falsas.

D

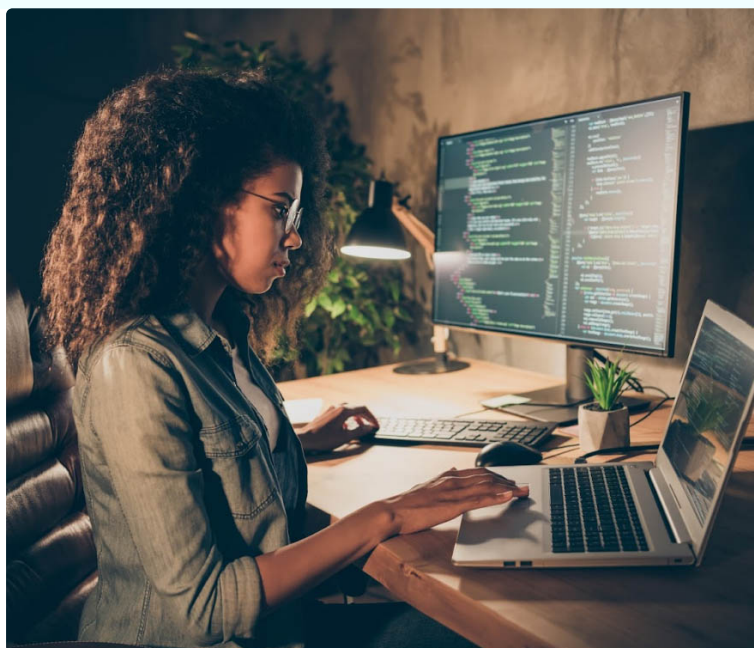
I é falsa e II é verdadeira.

E

I é verdadeira e II é falsa.

Parabéns! A alternativa A está correta.

Ambas as assertivas estão corretas, porque o método `deleteMany()` excluirá todos os documentos de uma coleção que correspondam ao critério de seleção de uma única vez.



3 - Aplicação JS na Web

Ao final deste módulo, você será capaz de construir uma aplicação JavaScript implementando as operações do CRUD (Create, Read, Update e Delete) de forma assíncrona.

Estrutura do projeto LojaWeb

Arquitetura MVC

Vamos desenvolver uma aplicação para implementar as operações CRUD apresentadas utilizando a linguagem JavaScript. Para isso, siga as orientações fornecidas para o êxito em suas atividades. Vamos considerar que você já tenha instalado em sua estação de trabalho o Visual Studio Code.

Adotaremos o padrão de arquitetura MVC (Model View Controller) que faz a separação de uma aplicação em três componentes principais:

Modelo



É a parte responsável pelo domínio de dados, ou seja, pelas operações de recuperação e armazenamento de dados de um banco de dados. De forma mais objetiva, é nessa camada que fazemos o mapeamento das tabelas do banco de dados e de suas respectivas colunas, com classes que serão gerenciadas pela aplicação.

Visão



É o componente responsável pela interface do usuário com a aplicação. Exemplos de aplicações da camada de visão são:

- Telas para operações de cadastro, consulta e edição de tabelas;
- Exibição de relatórios.

Controlador



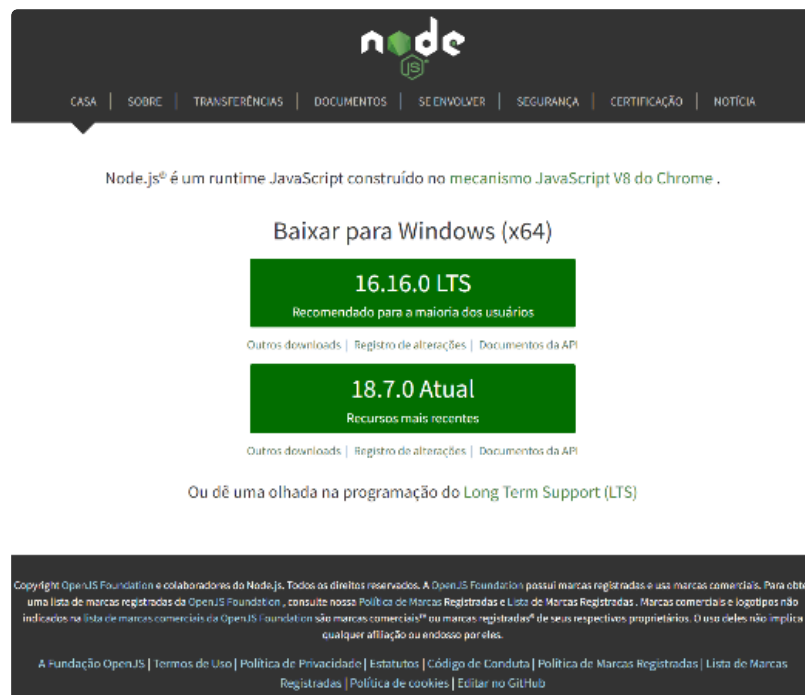
São os componentes que tratam da interação do usuário com o modelo. Isso significa que o controlador é quem faz tratamentos de dados, como conversões, consultas e quaisquer procedimentos associados à manipulação lógica dos dados, de modo que a camada de visão receba esses resultados em um formato que possa ser exibido sem a necessidade de outras operações.

O padrão MVC nos ajuda a segmentar as diferentes partes de uma aplicação em lógica de acesso a dados, lógica de negócios e lógica da interface do usuário e nos dá ao mesmo tempo um mecanismo para realizar um acoplamento fraco entre esses elementos. Essas características são muito úteis para desenvolvimento de grandes projetos em equipe. Mas o MVC é apenas uma **arquitetura**. Para utilizá-la na prática, precisamos de ferramentas que nos auxiliem nesse

processo. Aqui, vamos utilizar o **Node.js**, utilizando como editor o **VS Code**.

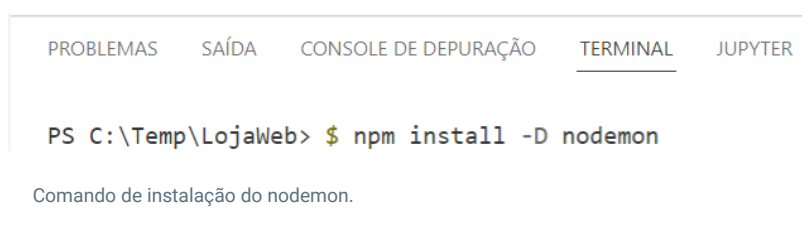
Inicialmente, necessitaremos instalar o compilador Javascript Node.JS para rodar nosso projeto por meio do terminal, sendo a versão de download ilustrada na imagem a seguir (usar versão LTS). O Node.js é um software de código aberto, multiplataforma, baseado no interpretador V8 do Google e que permite a execução de códigos Javascript fora de um navegador Web.

O Npm, a ser utilizado em prompt Windows ou no VS Code, é um gerenciador de pacotes para o Node.JS. Nós o utilizaremos para instalar as dependências que estão no nosso projeto, sendo ele já instalado automaticamente quando da instalação do Node.js.



Download do Node.js.

Vamos também instalar a **biblioteca nodemon** que auxilia o desenvolvimento de sistemas com o Node.js, reiniciando automaticamente o servidor, ou seja, permite reiniciar o servidor toda vez que ocorre alguma alteração no código, evitando que a cada mudança no código precise parar o servidor e iniciar novamente para carregar as novas alterações. A instalação deverá ser realizada no VS Code de acordo com a seguinte imagem:



Criação do projeto

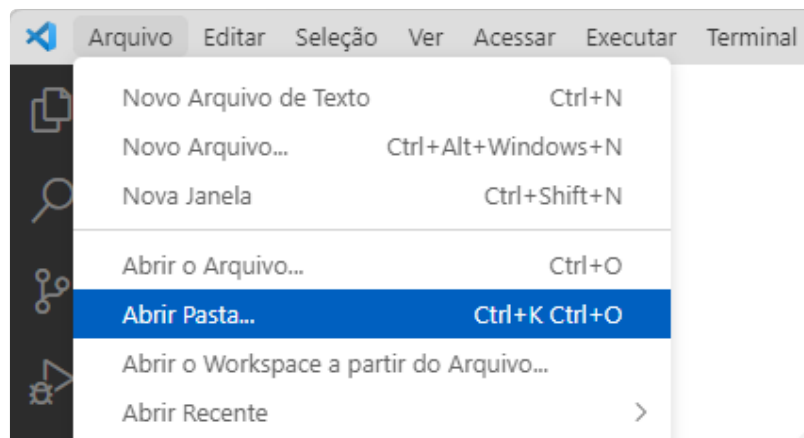
Após a instalação do Node.js, vamos criar uma pasta para o projeto denominada de “LojaWeb”, instanciando o novo projeto a partir do prompt do windows ou o terminal do VS Code (comando “npm init -y”), veja:

```
C:\Temp\LojaWeb>npm init -y
npm WARN config global '--global', '--local' are deprecated. Use '--location=global' instead.
Wrote to C:\Temp\LojaWeb\package.json:

{
  "name": "lojaweb",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}
```

Comando de criação do projeto no Node.js.

A seguir, no VS Code, vamos abrir a pasta do projeto de acordo com a imagem.

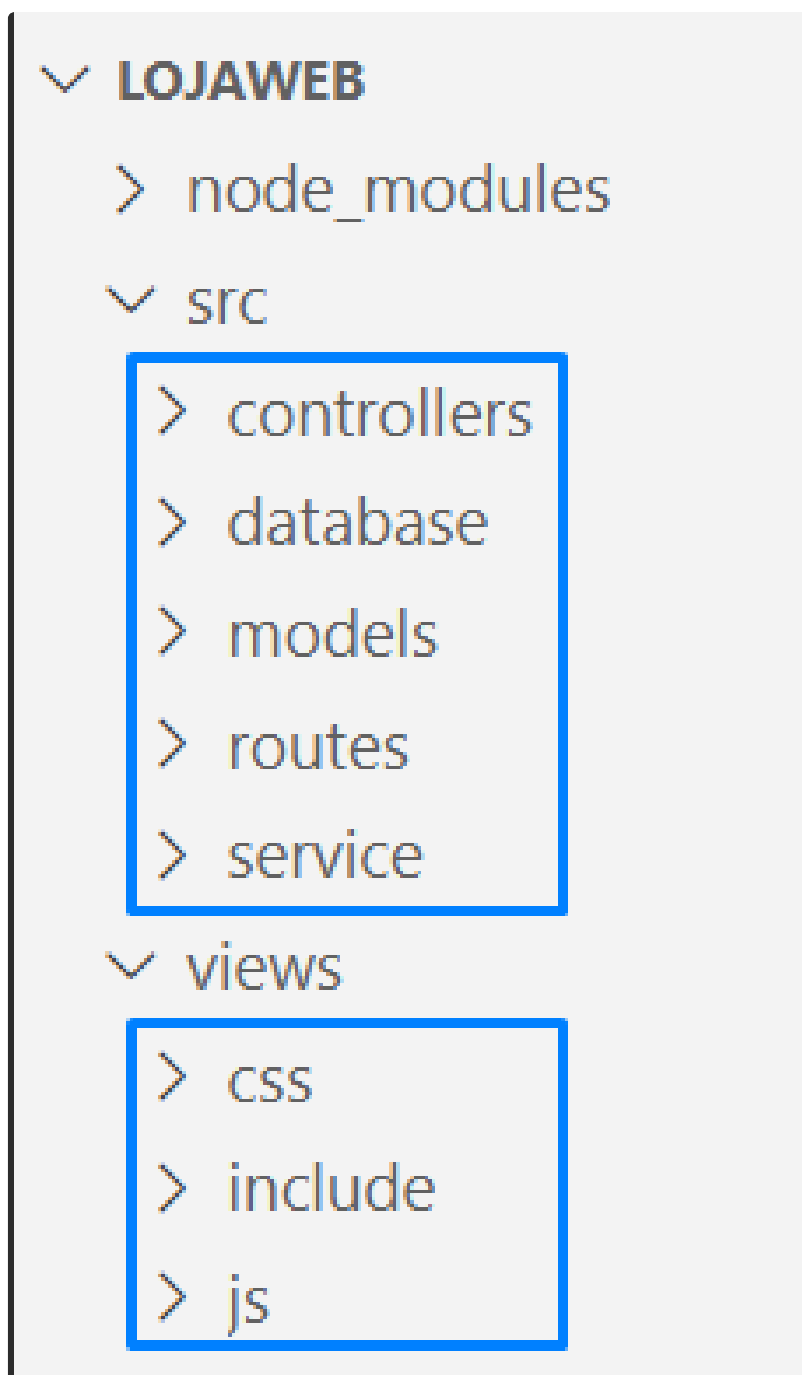


Abrindo pasta do projeto.

Vamos criar a estrutura do projeto, iniciando pelas pastas “**src**”, usada principalmente para armazenar os arquivos de código-fonte, e “**views**”, que armazena os códigos da camada visão do modelo MVC, veja:

Criação das pastas “src” e “views”.

Em seguida, criamos as subpastas:



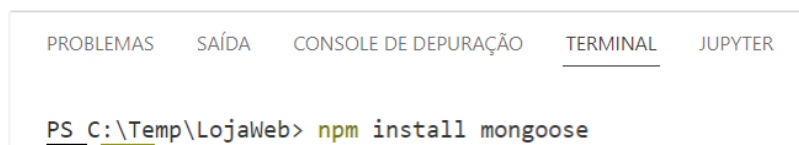
Criação das subpastas de "src" e "views".

Códigos-fonte JavaScript

Pasta de configuração Database

Precisamos instalar a biblioteca de **Modelagem de Dados de Objeto** (ODM) para MongoDB e Node.js denominada **Mongoose**. Ela gerencia o relacionamento entre dados, fornece a validação de esquemas, sendo usada como tradutor entre objetos no código e a representação desses objetos no MongoDB.

Vamos utilizar no VS Code o comando **"npm install mongoose"**.

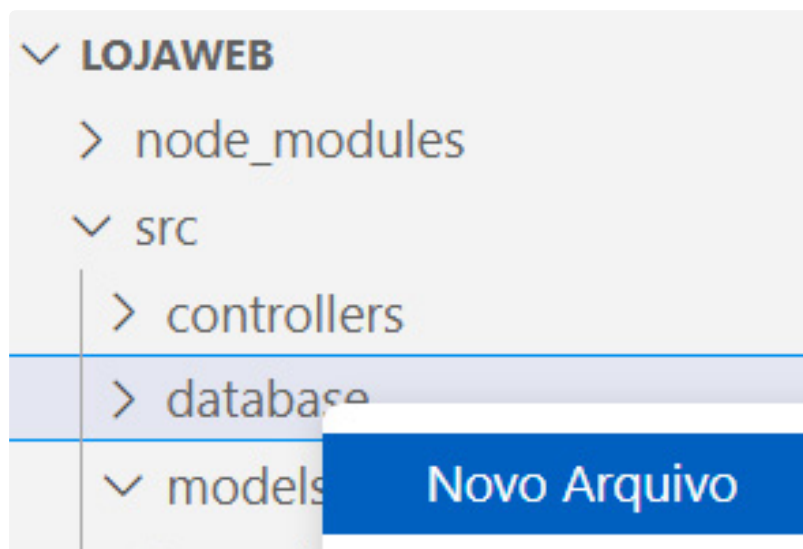


```
PROBLEMAS  SAÍDA  CONSOLE DE DEPURAÇÃO  TERMINAL  JUPYTER

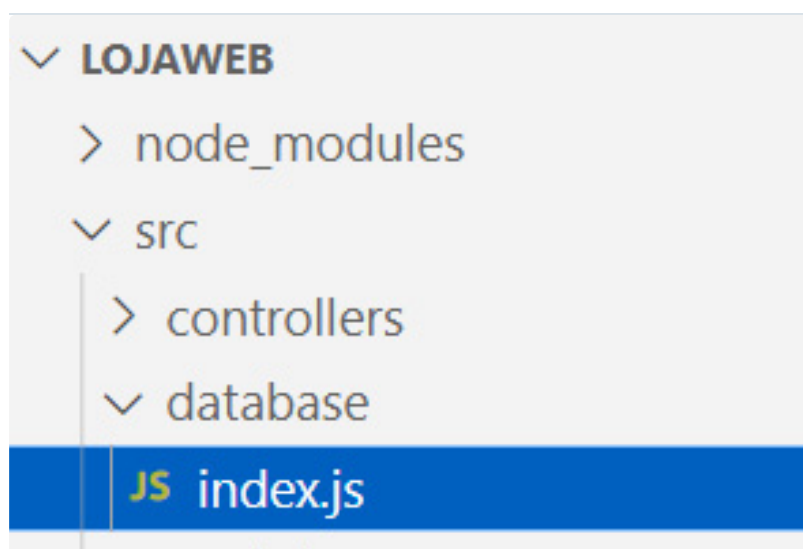
PS C:\Temp\LojaWeb> npm install mongoose
```

Instalação da biblioteca mongoose.

Em seguida, vamos criar o arquivo "index.js" na pasta de configuração "Database", de acordo com as imagens a seguir:



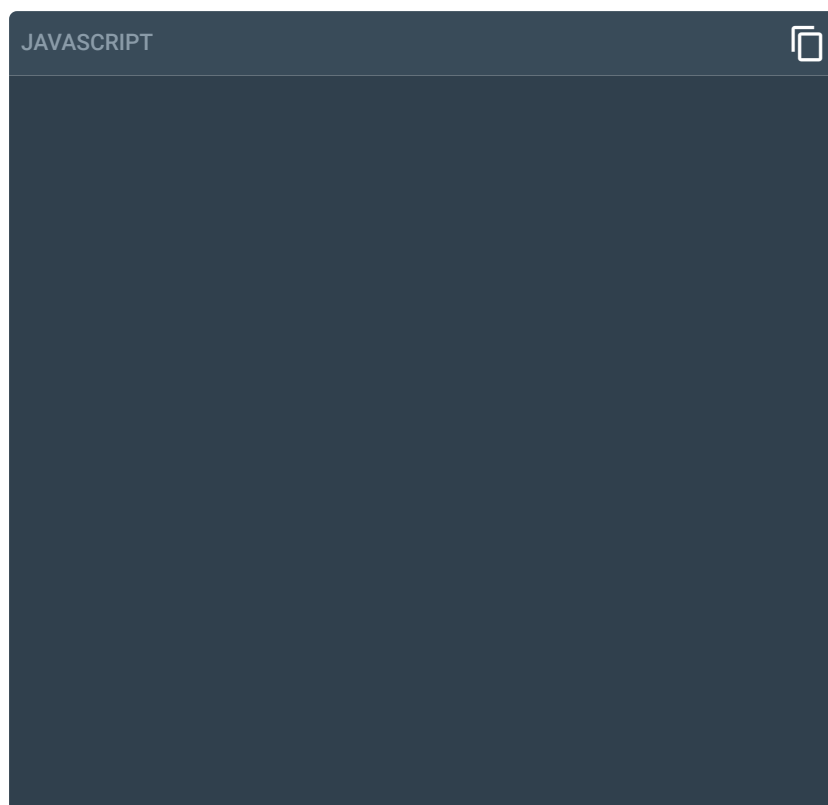
Criação de um arquivo JS em uma pasta.



Criação do arquivo index.js.

O código do arquivo index.js fará a importação da biblioteca mongoose, estabelecendo a conexão com o banco de dados MongoDB por meio da string de conexão **"mongodb://desenv:Fortaleza@localhost:27017/?authMechanism=DEFAULT"**; caso a conexão seja realizada com sucesso, retornará uma mensagem de conectado, caso haja erro, enviará uma mensagem de erro no console. Por fim, exportamos o

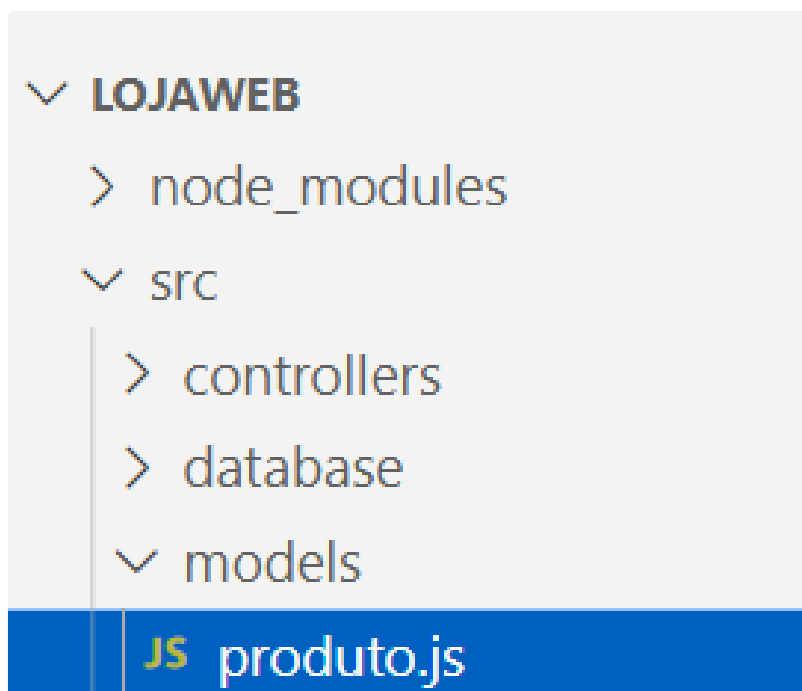
mongoose para que fique disponível em toda a aplicação. Veja o código do arquivo **index.js**:



Código de conexão em JavaScript.

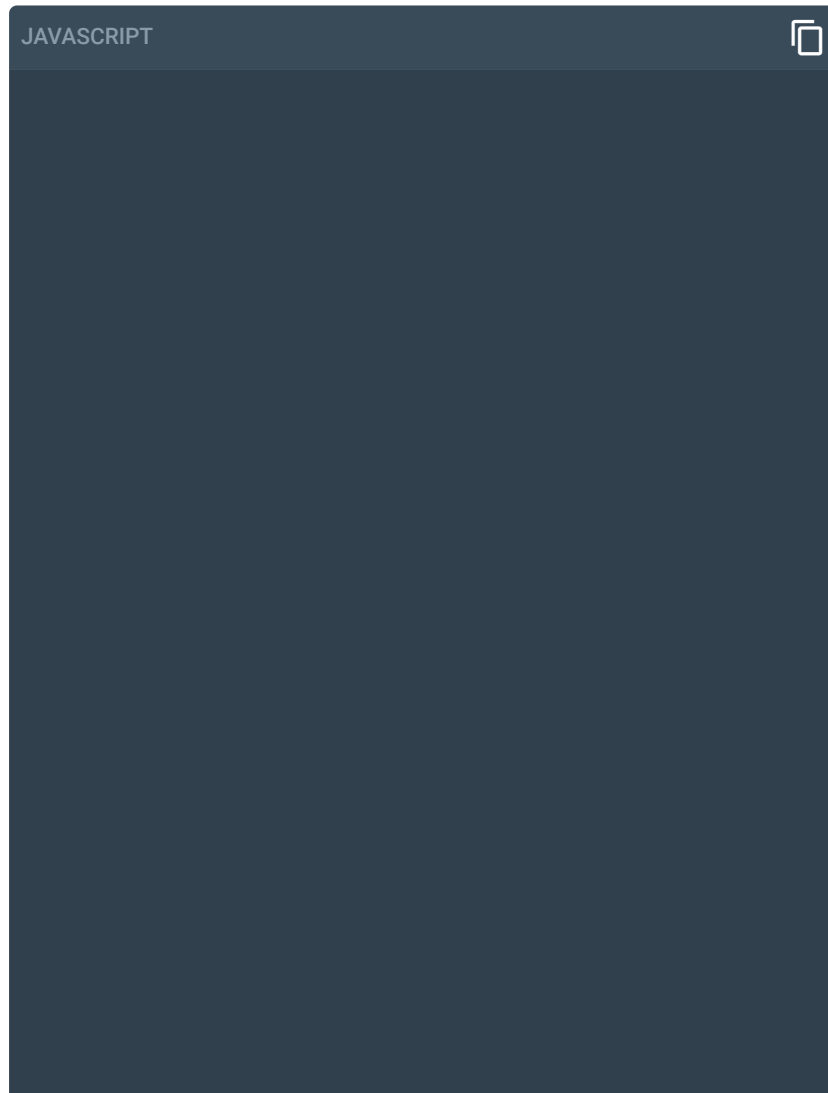
Camada Model

Crie na pasta “**models**” o novo arquivo JavaScript denominado **produto.js**, sendo o modelo da camada MVC que contém o esquema de dados que a aplicação manipula.



Criação do arquivo produto.js.

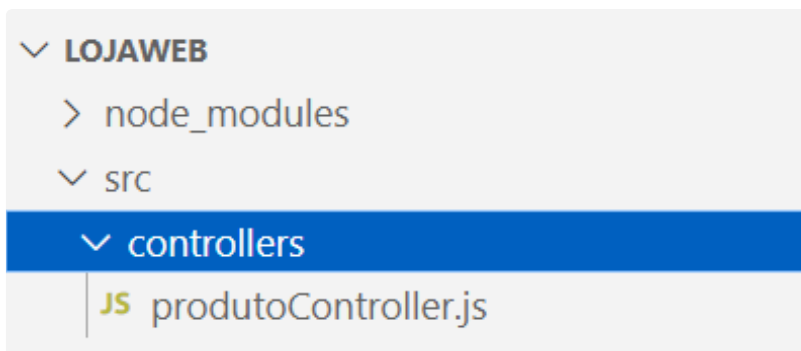
Na criação do modelo de dados, que vai ser salvo no banco, é fornecido o nome de cada objeto e o tipo de dado que receberá, bem como a data de criação de cada item. Em seguida, por meio da variável do **Schema**, adicionamos ao nosso modelo de dados o nome da respectiva tabela no banco e, ao final, fazemos a exportação. Veja o código do arquivo `produto.js` listado a seguir:



Definir estrutura de dados.

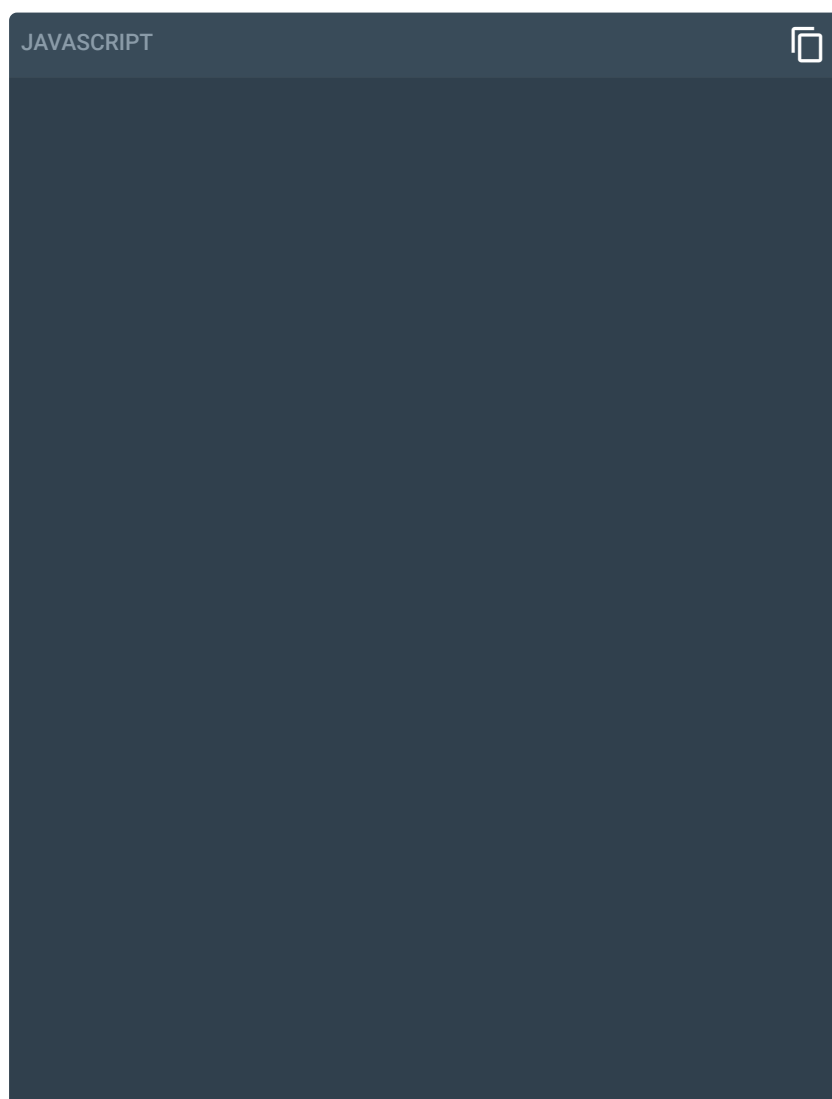
Camada Controller

Crie na pasta “controllers” o novo arquivo JavaScript denominado de `produtoController.js`, sendo o controlador da camada MVC que contém os serviços que a aplicação disponibiliza.



Criação de um arquivo produtoController.js.

Os serviços inclusos no arquivo produtoController.js estão listados no seguinte código:



Definir uma controladora.

Destacamos que estamos projetando operações assíncronas. Vamos observar a linha de comando **“exports.cadastrar = async (req, res)”**, em que **async** é multi-thread, o que significa que operações ou programas podem ser executados em paralelo. A sincronização é de thread único, portanto, apenas uma operação ou programa será executado por vez. O

async não é bloqueante, o que significa que enviará várias solicitações para um servidor.

Pasta de configuração Service

Precisamos instalar a biblioteca Axios, que é um cliente HTTP baseado em **Promises** para fazer requisições. Pode ser utilizado tanto no navegador quanto no Node.js ou qualquer serviço de API. Nesse projeto, é utilizada para receber e enviar informações entre o front-end e o back-end, tais como gerar a listagem dos produtos, a edição de dados e o envio do ID quando deletar um produto da lista. A instalação da biblioteca Axios no VS Code está ilustrada na imagem:

Promises

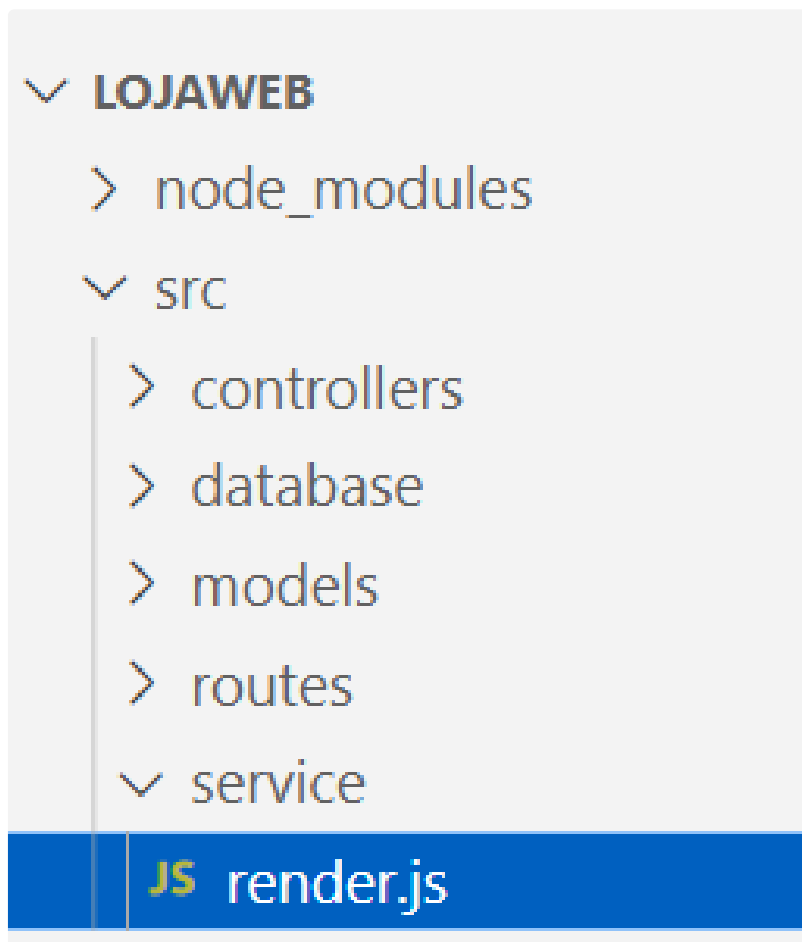
Promise é um objeto usado para processamento assíncrono.

PROBLEMAS SAÍDA CONSOLE DE DEPURAÇÃO TERMINAL JUPYTER

```
PS C:\Temp\LojaWeb> npm install axios
```

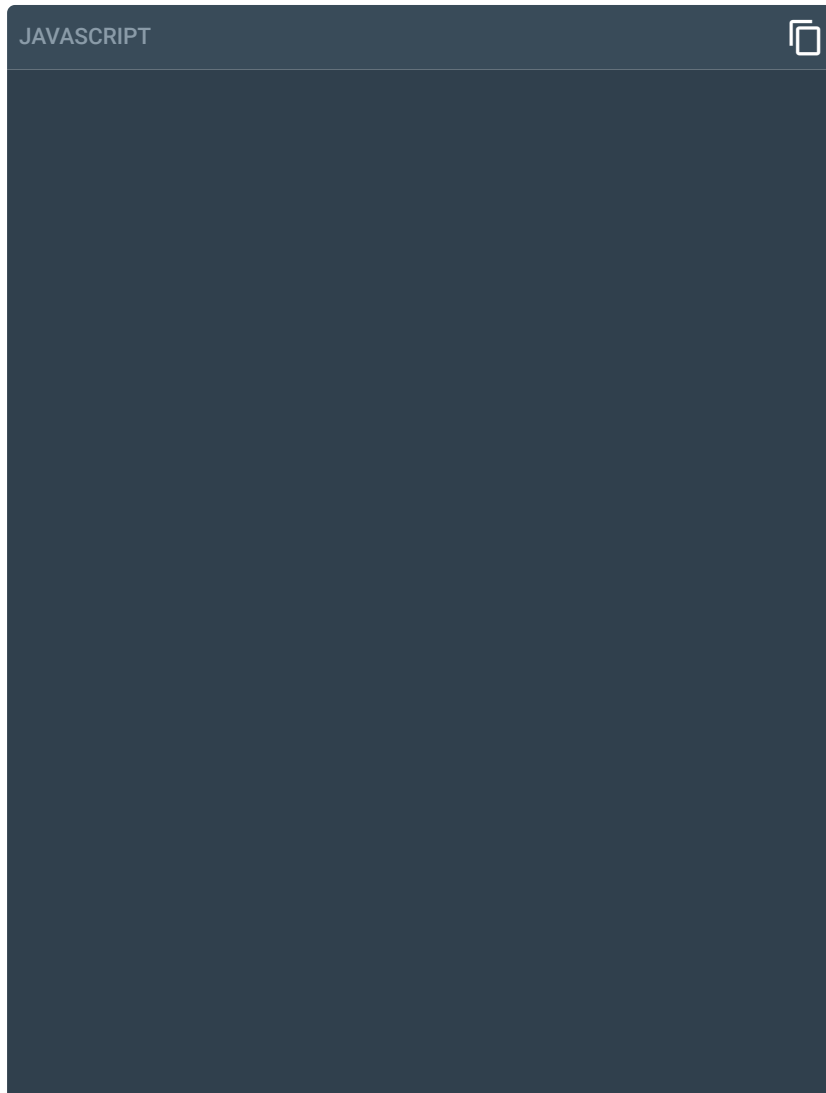
Instalação da biblioteca Axios.

Em seguida, vamos criar na pasta **"Service"** o arquivo **"render.js"**



Criação do arquivo render.js.

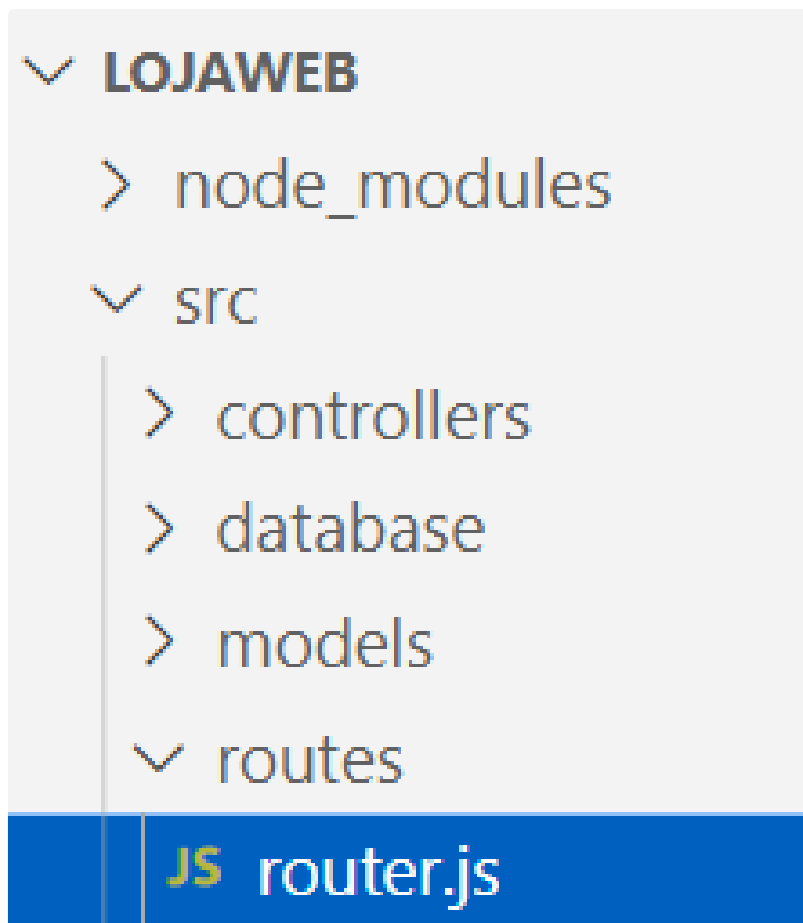
O arquivo render.js contém requisições da API para o back-end como listar todos os produtos (homeRouter), criar um novo produto (novo_produto) e, por fim, atualizar um produto da lista (atualizar_produto), sendo que cada rota executa um método diferente como GET ou POST. Vejamos o código:



Código de roteamento.

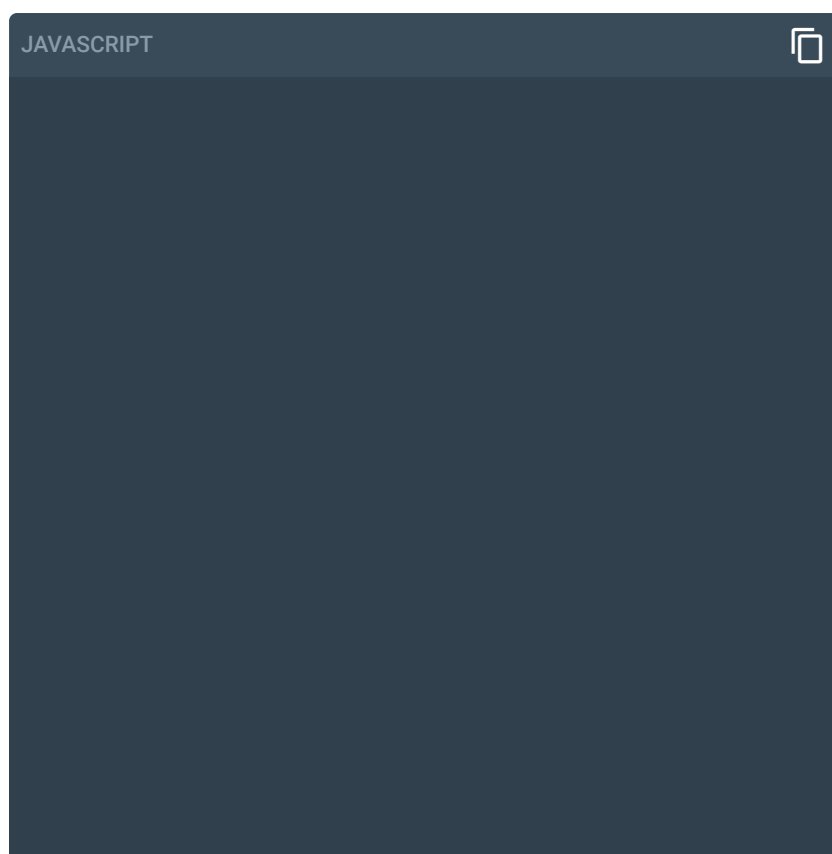
Definição de rotas

Crie na pasta “**routes**” o novo arquivo JavaScript denominado **router.js**.



Criação de um arquivo router.js.

O arquivo render.js é o controlador de rotas da aplicação, sendo cada rota executada por um método diferente, como GET, POST, PUT ou DELETE, de acordo com o código a seguir (a biblioteca “express” será instalada mais à frente):

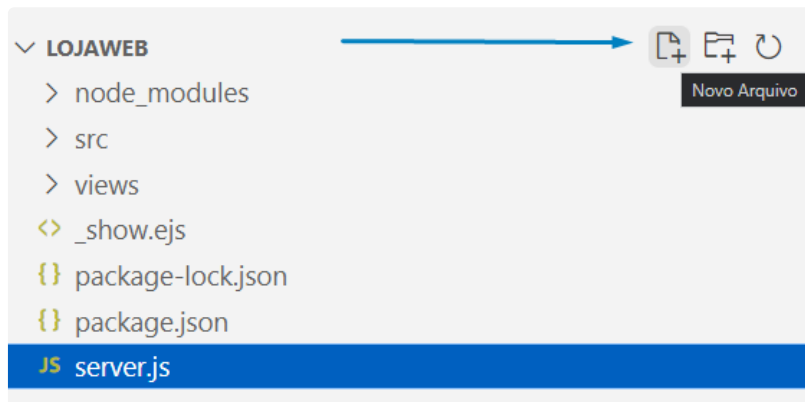




Código de roteamento.

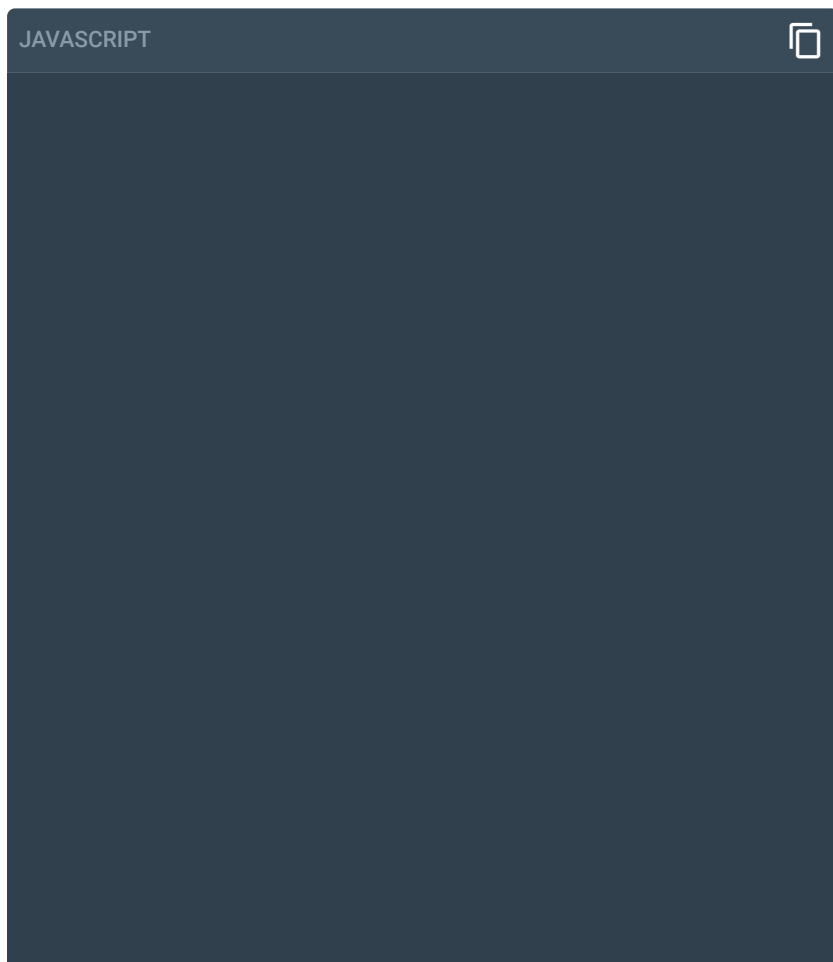
Configuração do servidor

Crie na pasta raiz do projeto “LojaWeb” o novo arquivo JavaScript denominado server.js.



Criação de um arquivo server.js.

O arquivo server.js contém o seguinte código:



Código de configuração de servidor.

Vejamos as demais bibliotecas necessárias.

Biblioteca "body-parser"

Permite converter o body da requisição para vários formatos, sendo que um desses formatos é json, exatamente o que queremos. Veja a instalação:

Instalação da biblioteca body-parser.

Biblioteca "express"

Veja no código anterior que a opção "extended" diz para o express qual biblioteca deve utilizar para fazer o parsing do conteúdo das requisições recebidas. Quando "extended : true", vai utilizar a biblioteca qs, e quando for false, ele vai utilizar a biblioteca querystring. Veja a instalação:

Instalação da biblioteca express.

Biblioteca Concurrently

Permite executar vários comandos simultaneamente. Nesse projeto, é usada para executar tanto o servidor quanto o front-end (Pagina Web). Veja a instalação:

Instalação da biblioteca Concurrently.

Biblioteca EJS

Uma engine de visualização que permite de uma maneira fácil e simples transportar dados do back-end para o front-end. Basicamente conseguimos utilizar códigos em JavaScript no HTML de nossas páginas. Aplicaremos essa biblioteca no nosso projeto para gerar nossa página front-end com as informações que serão recebidas do back-end via a API (Axios). Veja a instalação:

Instalação da biblioteca EJS.

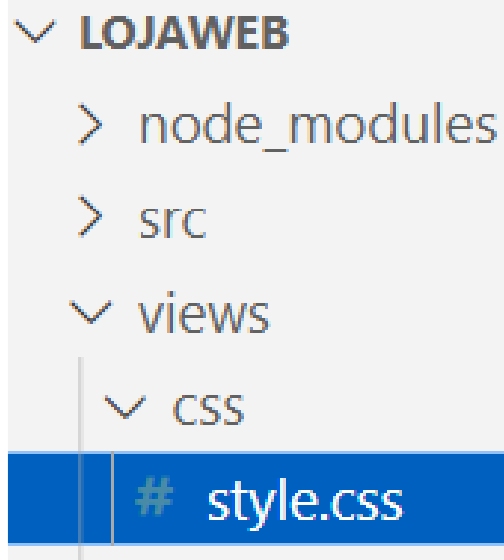
Camada de visualização

Pasta CSS

Com o papel estrutural que o HTML assumiu, vários dos atributos de configuração tipográfica das versões antigas se tornaram obsoletos, deslocando-se a responsabilidade para as folhas de estilo, ou CSS (Cascade Style Sheets). Em termos práticos, temos seletores, que podem ser tags do HTML, classes ou identificadores, sobre os quais são aplicadas formatações, definidas entre chaves, com um conjunto de elementos do tipo chave-valor separados por ponto e vírgula.

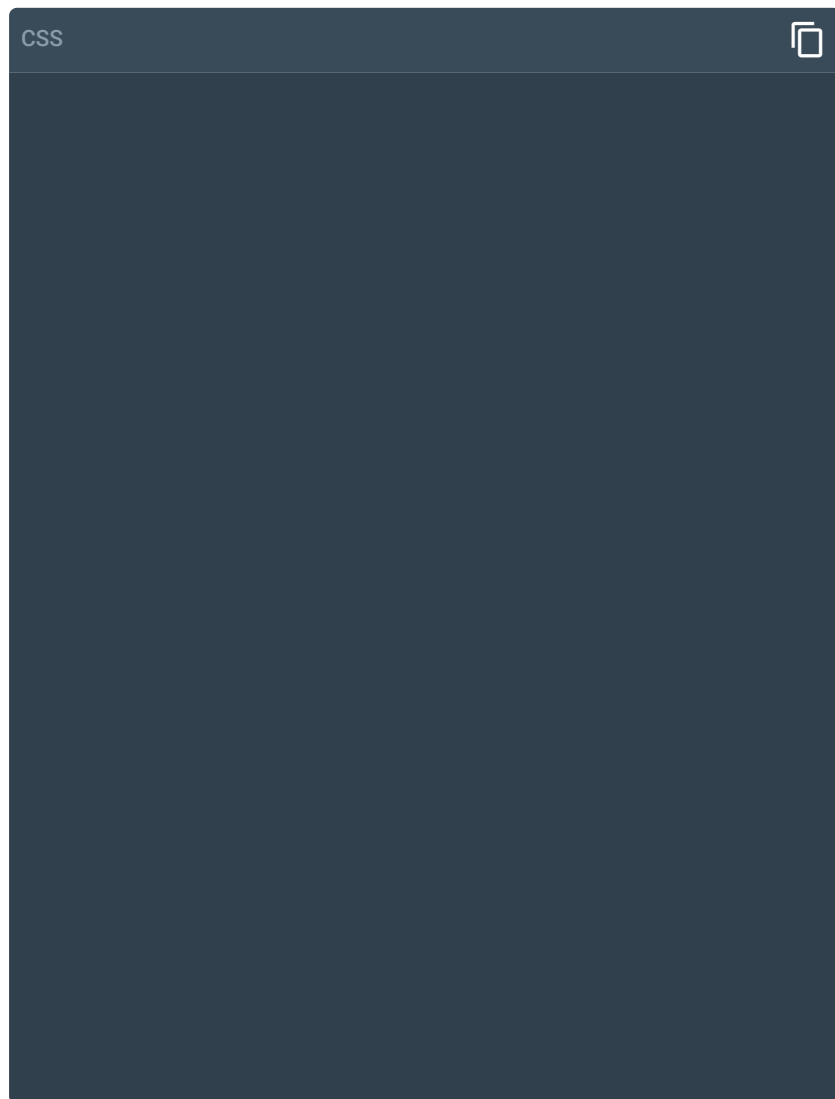
As folhas de estilo podem ser adicionadas na tag do HTML (inline), em um trecho delimitado por style, ou em um arquivo externo. Em termos de reutilização e responsividade, a melhor opção é o uso de arquivo externo, pois permite centralizar as formatações e versionar para cada plataforma, ou seja, utilizar pontos de quebra para formatações específicas.

Vamos agora criar o arquivo **"style.css"** na pasta CSS, de acordo com a imagem a seguir:



Criação do arquivo style.css.

Agora veja o código do arquivo style.css:



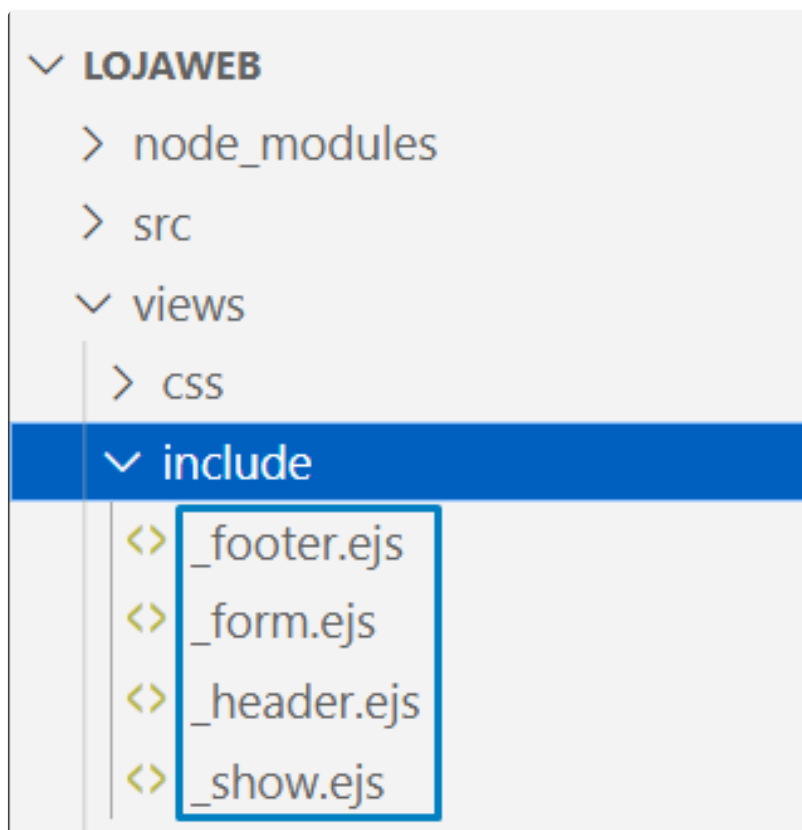
Código de formatação de tela.

Criação dos templates das páginas

Pasta include

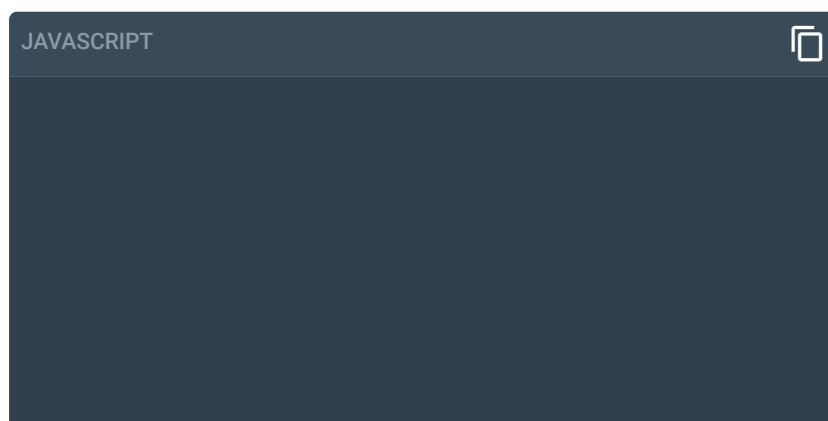
A pasta “include” do nosso projeto possui os templates que serão inseridos quando da execução da aplicação, que é caracterizada pela chamada ao arquivo “index.ejs”. Cabe destacar que a extensão EJS (Embedded JavaScript templating) é uma linguagem de modelagem simples que permite gerar marcação HTML com JavaScript simples.

A pasta “include” inclui os arquivos de formatação do cabeçalho arquivo “_header.ejs”, o template para o cadastro de um novo produto – arquivo “_form.ejs” e o template do rodapé – arquivo _footer.ejs.



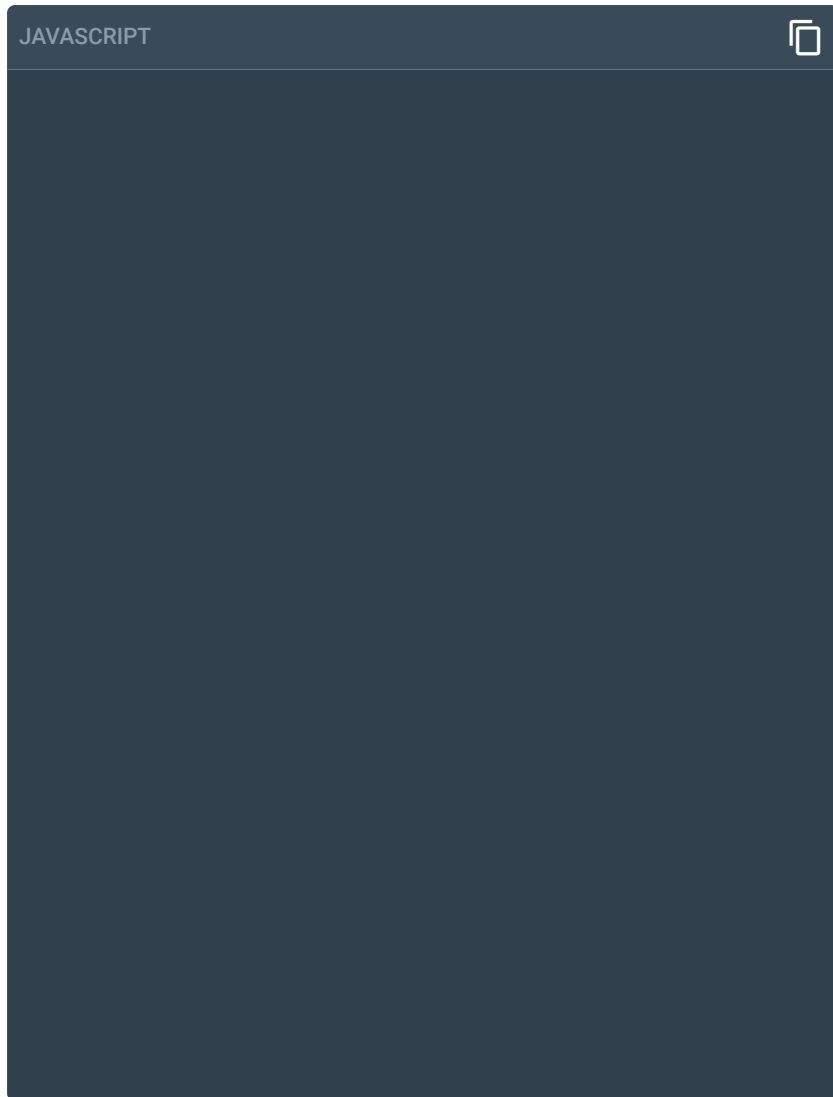
Estrutura da pasta include.

Vejamos os códigos a seguir, iniciando pelo **Arquivo _footer.ejs**:



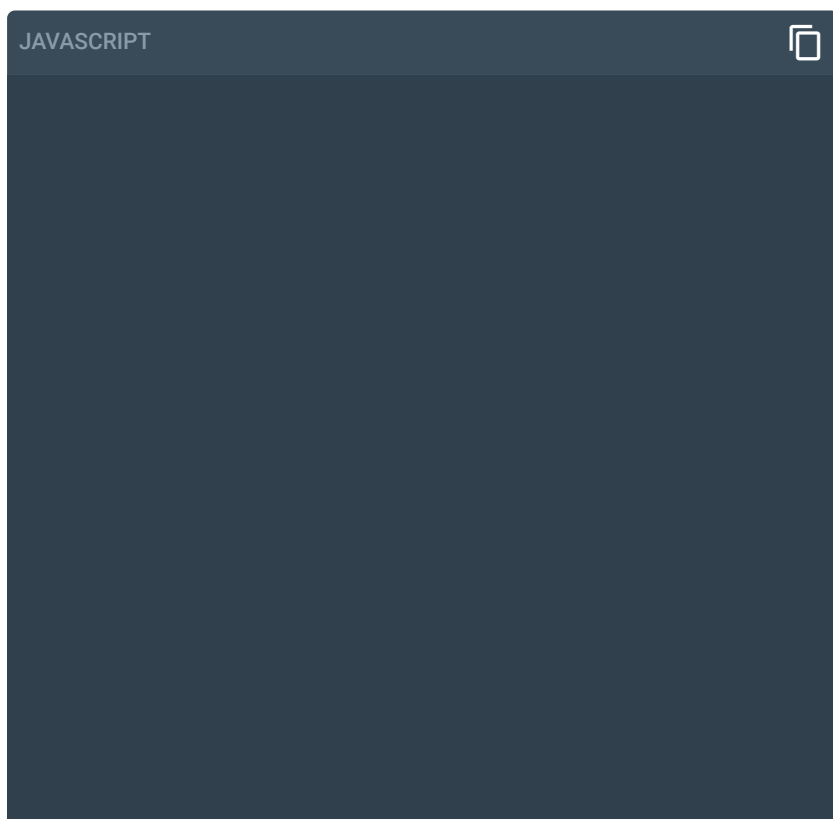
Código de formatação de tela.

Agora vejamos o código para **Arquivo _form.ejs**:



Código de formatação de tela.

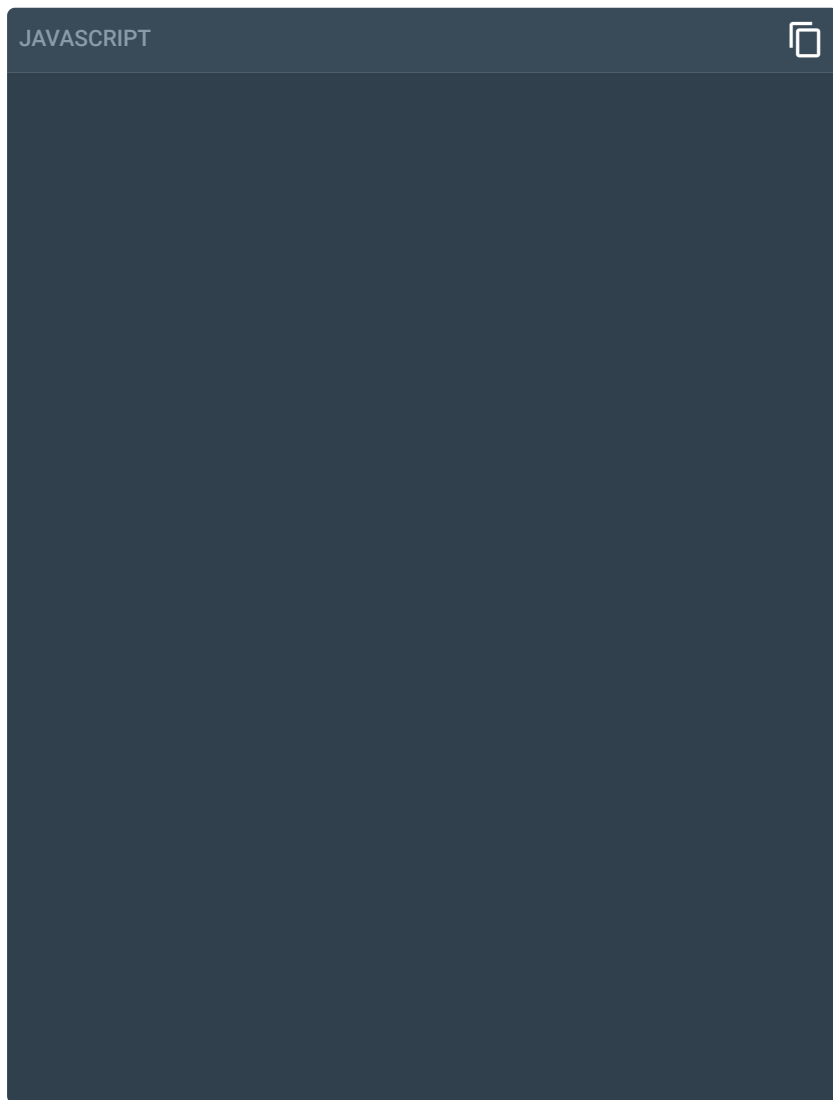
Dando sequência, temos o código **Arquivo _header.ejs**:





Código de formatação de tela.

Por fim, temos o código **Arquivo _show.ejs**:



Código de formatação de tela.

Arquivos EJS da pasta views

Na pasta views criaremos a página principal da aplicação – **index.ejs**, que inclui os templates definidos anteriormente, são eles:



_header

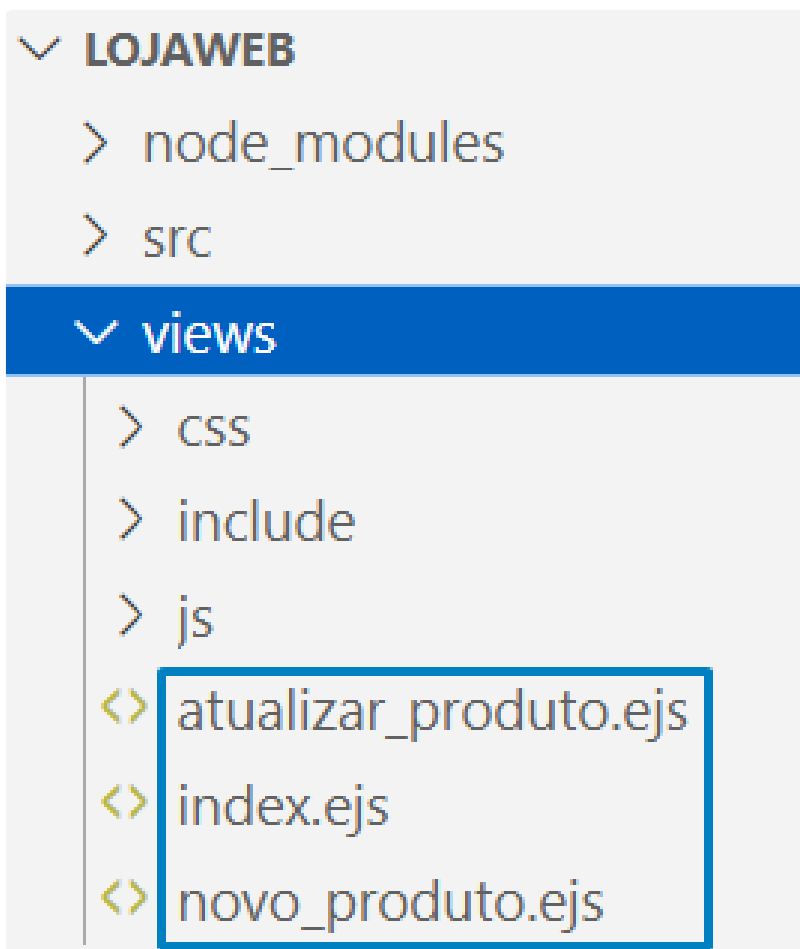
•

_show

•

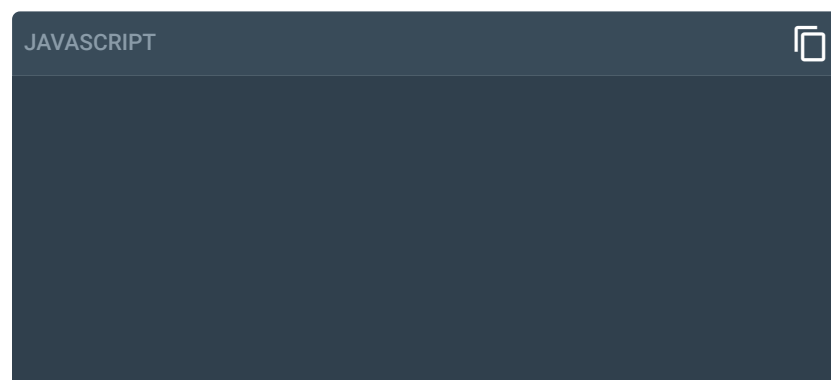
_footer

Na mesma pasta, criaremos os templates para cadastro de um novo produto – **novo_produto.ejs** e para atualização de produto – **atualizar_produto.ejs**. Veja os arquivos:



Estrutura dos arquivos ejs da pasta views.

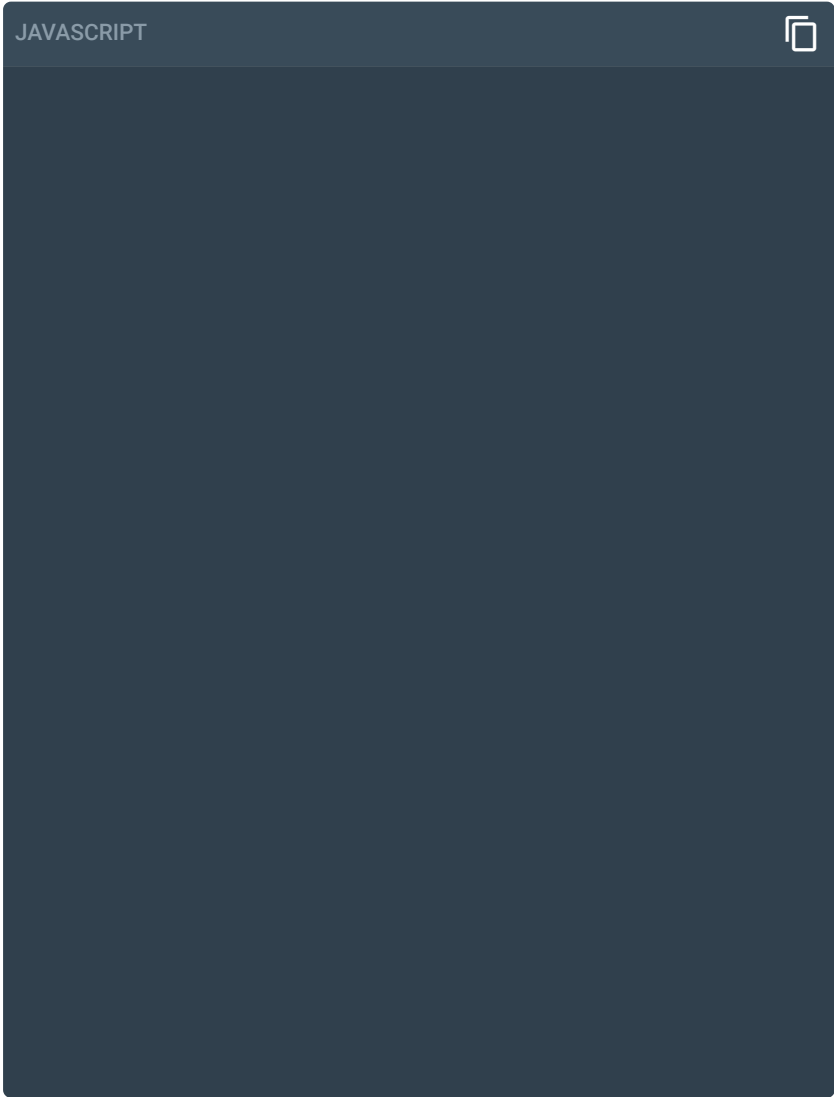
Vejamos os códigos, iniciando pelo **Arquivo index.ejs**:





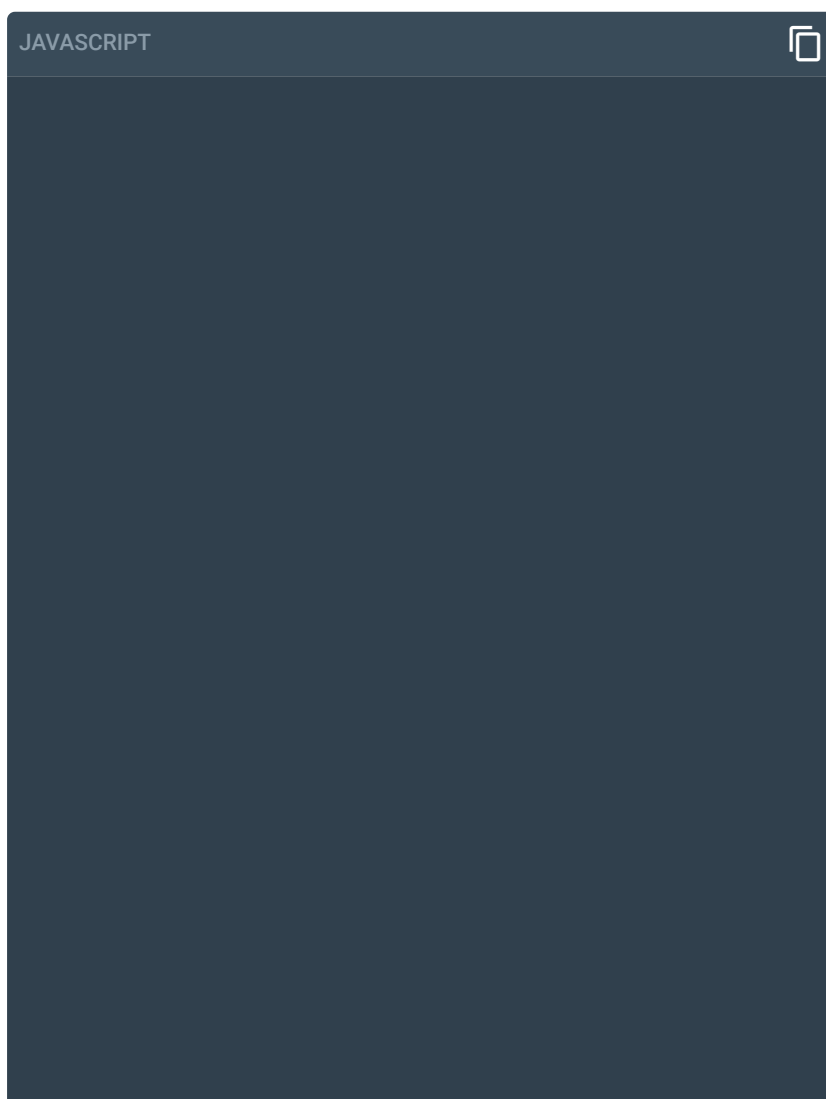
Código de formatação de tela.

Agora, vejamos o código para **Arquivo novo_produto.ejs**:



Código de formatação de tela.

Por fim, vejamos o código para **Arquivo atualizar_produto.ejs**:

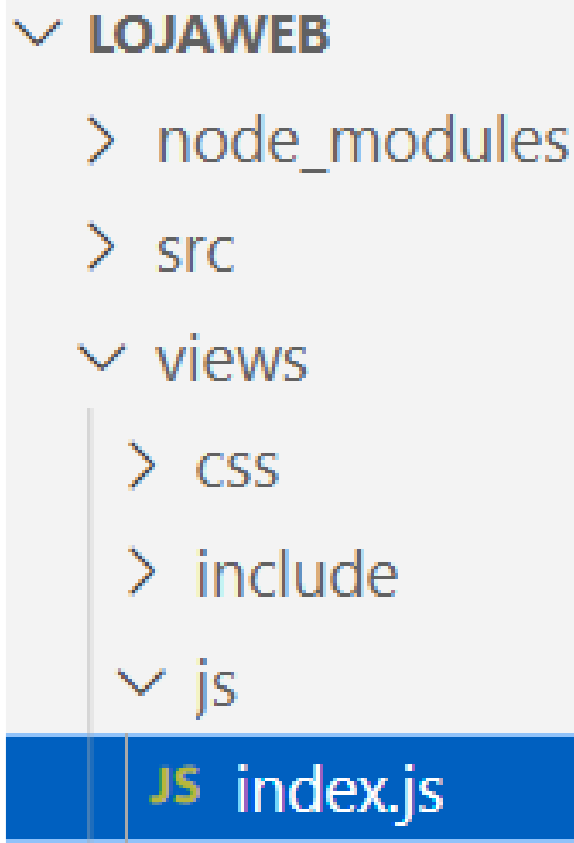


Código de formatação de tela.

Arquivo de eventos

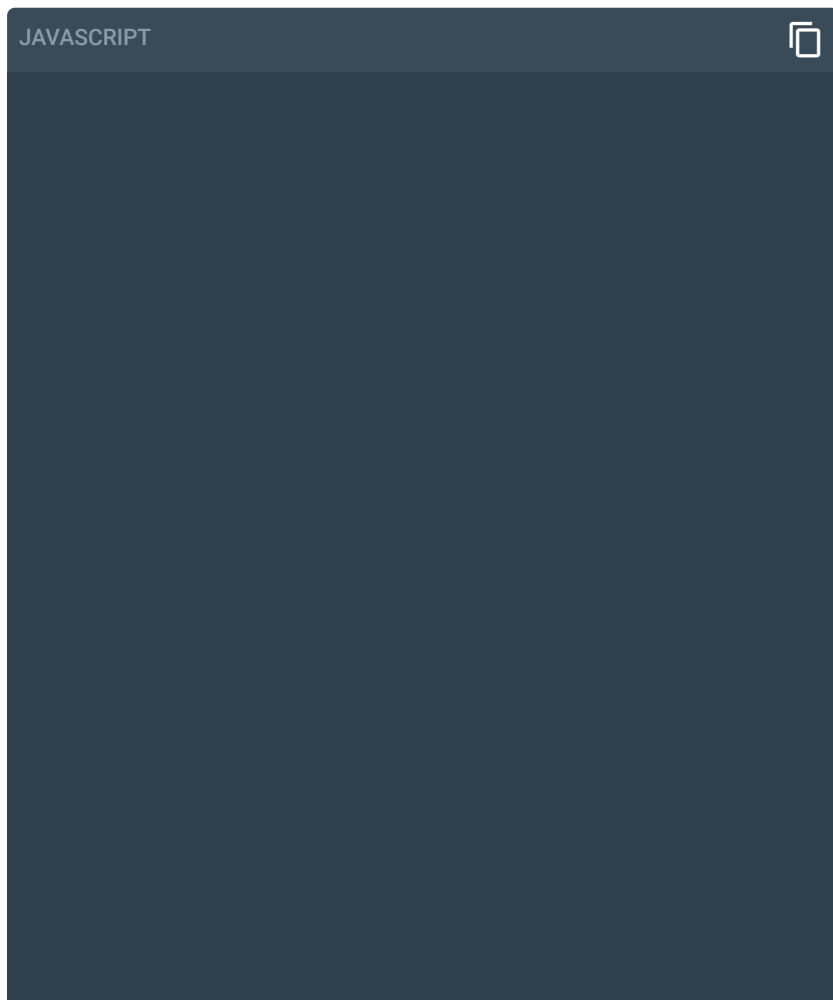
Arquivo index.js

Finalizando a camada de visualização do projeto, criaremos o arquivo Javascript **"index.js"** na pasta **"js"** que implementa os eventos **"submit"** associados ao cadastro, atualização e exclusão de um produto. Cabe destacar a utilização de chamadas assíncronas utilizando a tecnologia conhecida como AJAX. Vejamos a pasta de criação desse arquivo:



Criação do arquivo index.js.

A seguir, o código do arquivo “**index.js**”:



Código de requisição de serviço.

Executando a aplicação

Por fim, para que aplicação possa estar em execução, precisamos instrumentalizar essa operação. Para isso, no menu Terminal, selecione **Novo Terminal** no VS Code. Agora digite o comando **“npm start run”**, para que seja iniciada a aplicação que irá rodar no **localhost:3000**, portanto, o sistema ficará ativo nesta rota. Veja:

PROBLEMAS SAÍDA CONSOLE DE DEPURAÇÃO TERMINAL JUPYTER

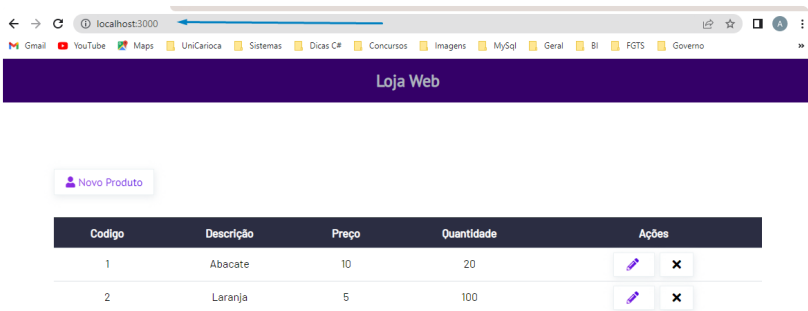
```
PS C:\Temp\LojaWeb> npm start run

> lojaweb@1.0.0 start
> concurrently "npm run server" "run"

[1] 'run' não é reconhecido como um comando interno
[1] ou externo, um programa operável ou um arquivo em lotes.
[1] run exited with code 1
[0]
[0] > lojaweb@1.0.0 server
[0] > nodemon server.js
[0]
[0] [nodemon] 2.0.19
[0] [nodemon] to restart at any time, enter `rs`
[0] [nodemon] watching path(s): *.*
[0] [nodemon] watching extensions: js,mjs,json
[0] [nodemon] starting `node server.js`
[0] Servidor Rodando em http://localhost:3000
[0] Conectado ao MongoDB!
```

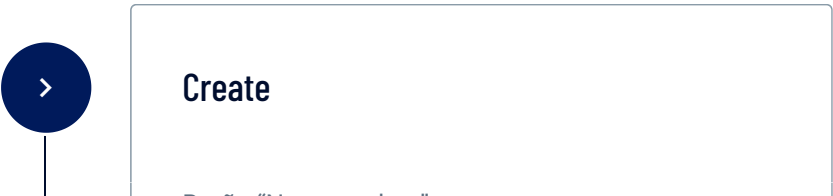
Execução da aplicação.

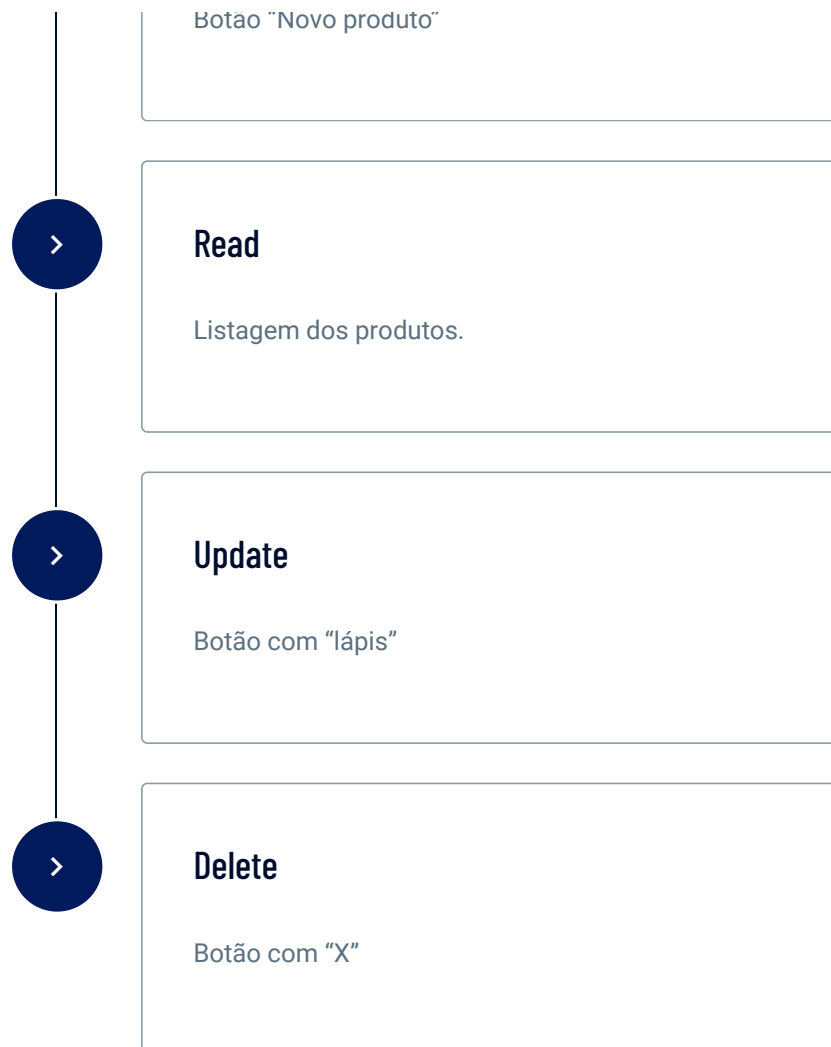
Acessando o navegador de sua preferência, o endereço Localhost:3000, é esperada a seguinte página inicial:



Tela principal da aplicação LojaWeb.

As operações CRUD estão disponíveis da seguinte forma:





Caso queira visualizar os dados no MongoDB, podemos observar a seguir a criação do banco de dados **"test"** na conexão criada anteriormente e especificada no arquivo **"index.js"** na pasta database do projeto LojaWeb.

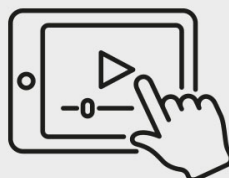
Visualização de dados no MongoDB.



Códigos-fonte JavaScript

Acompanhe agora a criação dos códigos-fonte JavaScript.

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.



Falta pouco para atingir seus objetivos.

Vamos praticar alguns conceitos?

Questão 1

Qual biblioteca gerencia o relacionamento entre dados, fornece a validação de esquemas, sendo usada como tradutor entre objetos no código e a representação desses objetos no MongoDB?

A

EJS

B

Concurrently

C

Mongoose

D

Body-parser

E

Nodemon

Parabéns! A alternativa C está correta.

Mongoose é uma biblioteca do Node.js que proporciona uma solução baseada em esquemas para modelar os dados da sua aplicação, possuindo um sistema de conversão de tipos, validação, criação de consultas e hooks para lógica de negócios.

Questão 2

Qual biblioteca permite de uma maneira fácil e simples transportar dados do back-end para o front-end, possibilitando o uso de códigos em JavaScript no HTML das páginas?

A

EJS

B

Concurrently

C

Express

D

Body-parser

E

Nodemon

Parabéns! A alternativa A está correta.

O EJS (Embedded JavaScript templating) é uma Template Engine que podemos utilizar com Node.js no transporte de dados do back-end para o front-end. O EJS segue uma sintaxe muito semelhante ao HTML, dessa forma, qualquer desenvolvedor que já conhece HTML não terá nenhuma dificuldade de trabalhar com o EJS.

Considerações finais

Você compreendeu conceitos, estrutura e comandos de operações fundamentais de um CRUD que correspondem às principais atividades de um sistema de informação.

Utilizamos o Visual Studio Code integrado ao banco de dados MongoDB, sendo apresentada a criação de um esquema de banco de dados e respectiva coleção. A partir desse banco, aplicamos as principais operações de consulta, inclusão, atualização e exclusão, ou seja, as operações CRUD assíncronas com códigos utilizados pela linguagem JavaScript por meio do framework Node.js.

Para encerrar, ouça um resumo dos principais aspectos abordados neste conteúdo.

Para ouvir o *áudio*, acesse a versão online deste conteúdo.



Explore +

Confira as indicações que separamos especialmente para você!

Pesquise a documentação oficial do Node.js.

Pesquise sobre a sintaxe e estruturas de comando do JavaScript no site w3schools.com.

Referências

FLANAGAN, D.; **JavaScript, o guia definitivo**. 6. ed. California: O'Reilly, 2013.

OLIVEIRA, C. L. V.; ZANNETI, H. A. P. **JavaScript Descomplicado**: Programação para a Web, IOT e Dispositivos Móveis. 1. ed. São Paulo: Érica, 2020.

PADOLSEY, J. **Clean Code in JavaScript**. 1. ed. Birmingham, UK: Packt, 2020.



Material para download

Clique no botão abaixo para fazer o download do conteúdo completo em formato PDF.



Download material

O que você achou do conteúdo?



 Relatar problema