Asasasav

O navegador manda um requisição para o servidor
E o servidor devolve a reposta com oos dados da página
Isso é uma aplicação Monolitica
O servidor é o que gera códigos HTML e a regra de negócio

Single Page Aplication - faz tudo na mesma página sem deixar ela piscar

Todo processamento do site fica do lado do servidor

REACT JS - Uma biblioteca para trabalhar com single pag app

O HTTP sempre é composto por uma requisição e uma resposta

Sempre que o Navegador fazer uma requisição para alguma url, ela sempre vai ser em método GET

No Body é onde vai ser carregado os dados
Os dados que vão no corpo dessa resposta ela pode ter vários formatos

Os dados que ficam na URL são query parameters

NODE JS - Runtime Javascript, roda do lado do servidor

O comando "node" é usado para executar arquivos JavaScript no computador

O POSTMAN serve para testar as requisições

A API Rest funciona em camadas então é útil se usar o padrão MVC

Existem 3 formas de trabalhar com Banco de Dados:
- query Nativas - escreve a query e manda executar
- query Builders - contrói as query e vai chamando as funções(no node tem como exemplo o Knex)
- ORM (Sequelize) - abstrair a camada de banco de dados

NPM - Gerenciador de pacotes no nodeJS

O NPM vai fazer a instalação dos pacotes, das bibliotecas, inicializar a aplicação, rodar scripts, entre outros...

node init -y - cria um pacote no formato json das nossas configurações. O '-y' diz sim para todas as perguntas que ele faria

npm install express - instala o pacote express

package-lock.json - roda a versão do pacote que está sendo usada

O pacote express vai roda o protocolo http

listen() - app que vai ser responsável por rodar a aplicação

CTRL+C(no terminal) - reinicia o servidor

-------INICIALIZANDO UM SERVIDOR-------
// Este método faz associação com a classe Postagem
-Para abrir a janela do Power Shell dentro da pasta - SHIFT + BOTÃO DIREITO DO MOUSE

-Verificar se o nodeJS está instalado - node -v (mostra a versão do nodeJS)

-Verificar se o npm está instalado - npm -v (mostra a versão do npm)

-Inicializar o projeto node - npm init -y (após digitar este comando, automaticamente ele criará um package.json. lá estará algumas configurações do nosso projeto)

A principal funcionalidade do package.json é armazenar todos os pacotes que o projeto irá utilizar

-------INSTALANDO UM SERVIDOR-------

Instalar o pacote express - npm install express

O express é um micro-framework para gerar servidores http

Ao executar o comando, ele gerará um arquivo chamado "package.-lock.json". Esse arquivo será responsável pelo armazenamento das versões das dependencias do projeto, para que no futuro, quando outro desenvolvedor for usar, usar especificadamente essas versões

-------PASTA SRC-------

Todos os arquivos de códigos ficaram na pasta src

Para rodar um script no node, basta entrar no terminal, digitar node "nomeDoDiretorio".

-------INSTALANDO O NODEMON-------

O nodemon serve para reiniciar o servidor de forma automática

Instalar o pacote nodemon - npm install nodemon -D (-D : vai ser instalado como dependencia de desenvolvimento)

Executar o nodemon ou qualquer outro pacote - npx nodemon .\src\server.js

Toda vez que for rodar o script dev no ambiente de desenvolvimento, vai executar o nodemon
"dev": "nodemon src/server.js"

-------CRIAÇÃO DE ROTAS-------

Usando o método GET para cadastrar no servidor

"/" - indica que é a rota raiz

o método get pede uma função, onde ele tem os parâmetros requisição e resposta

request - requisição
response - reposta

Exemplo:
app.get("/", (request,response) => {
response.send(pessoa);
});

-------FUNÇÕES USADOS NA CONTROLLER-------

Normalmente no Controller temos algumas funções como:
- index(requisição, resposta){} - Listagem
- store(requisição, resposta){} - Inserir
- update(requisição, resposta){} - Atualizar
- delete(requisição, resposta){} - Deletar

Essas funções recebem como parâmetro a requisição e a resposta

O "./" na hora do import, significa que eu quero acessar algum conteúdo que esta dentro da mesma pasta

-------INSTALANDO O ORM-------

Instalar o Sequelize, Sequelize-cli e o mysql2 - npm install sequelize, sequelize-cli, mysql2

-------ORM-------

ORM é o mapeamento objeto relacional.

Consiste basicamente em mapear os dados e a estrutura do banco em objetos no nosso projeto.

Exemplo:
Tabela pessoa

```
Nome Idade Sexo

class pessoa {
    String nome;
    Int idade;
    String sexo

    public save(){
        // Inserir no banco de dados
    }
}
```

Exemplos de ORMs
-Pyton - Flex
-Ruby - Rail
-Java - Hibernate
-PHP - Eloquent
-Typescript - Typeom
-Javascript - Sequelize

- npm run dev - roda o servidor no nodemon

-------MANIPULANDO O BANCO DE DADOS PELO SEQUELIZE-------

- npx sequelize -h - mostra todos os comandos que podemos usar no sequelize
- npx sequelize db:create - cria um database
- npx sequelize migration:create --name nome-arquivo - cria um arquivo de migração
- npx sequelize db:migrate - executa a migration
- npx sequelize db:migrate:undo - executa a versão anterior do projeto
  Obs: a versão foi configurada no arquivo "database.js"

As migrations estão sendo configuradas nos arquivos .sequelizerc

CTRL + SHIFT + P - abre uma aba de pesquisa do vsCode
SHIFT + ALT + SETA PARA BAIXO - duplica a linha

A função do Router é mandar a requisição para a controller apropriada

-------CRIANDO UMA MIGRATION-------

npx sequelize migration:create --name "nome-da-migration" - cria uma migration

-------CRIANDO A FUNCIONALIDADE POSTAGEM-------

1 - npx sequelize migration:create --name "nome-da-migration" - executar este comando para criar a migration
Lá serão configuradas a criação da Tabela e dos campos

2 - Criar na Model a classe "Postagem - Lá será onde vamos mapear o banco de dados

3 - Criar na Controller o arquivo postagem.js - ela vai intermediar entre o arquivo "routes.js" e o banco de dados que está na model

4 - Configurar a rota da aplicação no arquivo "routes.js"

-------CRIANDO A FUNCIONALIDADE DE APAGAR A POSTAGEM-------

Sempre quando vamos deletar ou desativar algo utilizamos o método DELETE

1 - Entre na pasta controller e crie o método delete

2 - Configure a rota da aplicação no arquivo routes.js

-------ASYNC E AWAIT-------

O JavaScript executa os códigos de forma assincrona, então ele não espera uma linha de comando terminar de ser executada para ir para a próxima.

Quando colocamos o await, dizemos para o JS que deve esperar aquela linha terminar de executar para ir para a próxima.

Quando não colocamos o await, o resultado vai ser uma promisse ou vazio

Quando colocamos o async no começo da função, estamos dizendo que esta função pode ser pausada para esperar um retorno

-------CONCEITOS-------

API

Web sevices

Interface

-------AUTORIZAÇÃO-------

Basic Auth - pede um usuário e senha no cabeçalho da requisição. Ele envia duas informações, o prefixo que é o tipo de autorização e um dado embaralhado, que é a codificação do usuário e senha passados

Bearer Token -

JWT - padrão de mercado que defini como transmitir e armazenar objetos JSON de forma compacta e segura entre diferentes aplicações. Ele é formado por 3 seções: HEADER, PAYLOAD E SIGNATURE

No HEADER temos:
- Algoritmo de criptografia que esta sendo usado
- Tipo de autenticação

No PAYLOAD temos:
- id do usuário(sub)
- emissor do Token(iss)
- data da expiração(exp)

No SIGNATURE temos:
- chave secreta da aplicação(secret)

-------IMPLEMENTANDO O JWT-------

1 - Instalar 2 pacotes:
- jsonwebtoken npm - irá facilitar a criação do JWT
- bcrypt.js - Gerar um rash da nossa senha
(npm install jsonwebtoken bcryptjs)

Para verificar se o pacote foi instalado, basta ir no package.json e procurar pelo nome

2 - Configurar no POSTMAN a sessao de autenticação(pasta sessao)

3 - Configurar a rota no arquivo routes.js

4 - Configurar a Controller no arquivo sessao.js

O hash é algo que é gerado e não tem retorno, ou seja, não tempo como descriptografar

O Token serve para verificar se a requisição qeu estamos enviando para nossa api ela pode ser atendida ou não

O arquivo auth.json vai guardar a chave da nossa aplicação em formato de rash

O JWT é uma aplicação que foi criada para não precisar usar o banco de dados

Para proteger as nossas rotas, usamos a estratégia de Middlewares

Middlewares - algo que vai rodar antes ou no meio de alguma coisa

Duvidas para segunda:
-como funciona o await
-se eu insiro o token da Maisa na hora de listar a postagem, pq tambem carrega a do Gustavo?