**Socket e Client Information**:

Socket: Representa a conexão de rede entre o servidor e o cliente.

client: Uma referência ao objeto Socket que representa o cliente.

**Request Information (Detalhes da solicitação):**

httpVersion: A versão do protocolo HTTP usada na solicitação (1.1).

method: O método HTTP da solicitação (GET, POST, etc.).

url: O caminho (path) da URL solicitada ('/' ou '/favicon.ico').

headers: Os cabeçalhos da solicitação HTTP, incluindo informações como User-Agent, Accept-Encoding, etc.

rawHeaders: Uma matriz contendo pares de cabeçalho bruto.

rawTrailers: Uma matriz de trailers de cabeçalho bruto.

**Response Information (Detalhes da resposta):**

statusCode: O código de status HTTP que será enviado na resposta (por enquanto, está como null).

statusMessage: A mensagem de status correspondente ao código de status (por enquanto, também está como null).

**Event Listeners:**

\_events: Uma coleção de manipuladores de eventos registrados para o objeto.

**Server and Connection Information:**

server: Uma referência ao objeto Server que representa o servidor HTTP.

allowHalfOpen: Indica se conexões half-open (em que o cliente não fechou a conexão após a resposta do servidor) são permitidas.

**Reading and Writable State:**

\_readableState: O estado de leitura do objeto.

\_writableState: O estado de escrita do objeto.

**Other Flags and Properties:**

complete: Indica se a solicitação foi completamente lida.

aborted: Indica se a conexão foi abortada.

upgrade: Indica se a conexão foi atualizada.

\_consuming: Indica se a solicitação está sendo consumida.

\_dumped: Indica se os dados foram descartados.

Módulo no nodeJs

Vou tentar explicar os módulos do Node.js de uma forma simples, como se estivesse falando com uma criança!

\*\*Imagina que você está brincando com peças de Lego. Cada peça tem uma função diferente, como rodas para fazer um carro, janelas para fazer uma casa, e assim por diante.\*\*

\*\*Agora, pense em um jogo de construção muito grande, onde você pode construir toda uma cidade com essas peças de Lego. No entanto, você não precisa construir tudo sozinho. Algumas pessoas já criaram conjuntos especiais de peças que fazem coisas legais, como carros, casas e árvores.\*\*

\*\*Esses conjuntos são como os módulos no Node.js. São pedaços de código que outras pessoas escreveram para fazer coisas específicas. Quando você quer fazer algo em seu programa, pode usar esses conjuntos de peças prontas, ou seja, módulos, em vez de escrever tudo do zero. Isso economiza muito tempo e esforço!\*\*

\*\*Por exemplo, se você quiser desenhar uma árvore no seu programa, em vez de criar todas as peças da árvore, você pode usar um módulo de "árvore" que alguém já fez. Esse módulo de árvore já sabe como desenhar uma árvore bonita, então você só precisa dizer a ele onde você quer colocar a árvore.\*\*

\*\*Os módulos do Node.js funcionam assim. Eles são como conjuntos especiais de peças de código que as pessoas criaram para realizar tarefas específicas. Você pode usá-los para fazer suas próprias coisas legais no seu programa sem ter que começar do zero!\*\*

\*\*Então, quando você ouvir falar sobre módulos no Node.js, pense neles como conjuntos de peças de Lego especiais que ajudam a construir coisas incríveis em seus programas de computador sem ter que fazer tudo sozinho!\*\*

No diretório public ficam todos os arquivos estáticos

Curiosidades EJS

Com o comando abaixo é possível adicionar pedaços de código html em outra página html:

<%- include(' **diretorio da pasta** ’)%>

Digamos que esse código abaixo está na pasta home> codigo 1

 <div class="logo">

                    <h2><span style="font-weight:lighter;">Danki</span><b>News</b></h2>

                </div><!--logo-->

E tenho outro código que está na pasta home > codigo 2

<form>

                        <i class="fa fa-search" aria-hidden="true"></i>

                        <input name="busca" type="text" placeholder="Busca..." />

                    </form>

Agora eu quero juntar os codigo em uma nova pasta home > codigo 3

<%- include(' ./home/codigo 1’)%>

<%- include(' ./home/codigo 2’)%>

div class="title-destaque-descricao">

                        <h2>Quer Fugir das Praias de Verão?</h2>

                        <p>Donec consectetur libero eget bibendum ullamcorper. Quisque auctor nulla eu enim ultrices laoreet. Nam eget neque odio. Sed vehicula imperdiet diam, a lacinia dui dictum a. In hac habitasse platea dictumst. Sed dapibus tellus ac turpis elementum, in sollicitudin lacus vehicula. Orci varius natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Nulla nisl nibh, porttitor ut magna sed, sollicitudin vulputate lectus. Donec euismod a lectus non feugiat. Suspendisse vitae magna sit amet tortor ultricies ornare. Morbi sollicitudin massa jl</p>

                    </div>

Conectando com o banco de dados mongo

const mongoose = require('mongoose');

mongoose.connect('mongodb+srv://root:senhamongo1@cluster0.7tyxizl.mongodb.net/test?retryWrites=true&w=majority',

{useNewUrlParser: true, useUnifiedTopology: true})

.then(() => {

console.log('Sucesso na conexão com o DB!');

}.catch((err) => {

console.log(err.message);

});

O que significa esse segundo parametro do método connect?  
  
Os parâmetros `{ useNewUrlParser: true, useUnifiedTopology: true }` que você passa para o método `mongoose.connect()` são opções de configuração que afetam o comportamento da conexão do Mongoose com o banco de dados MongoDB. Aqui está o que cada uma dessas opções significa:

1. `useNewUrlParser`: Quando definido como `true`, o Mongoose usará o novo analisador de URL do MongoDB. Isso é necessário porque a biblioteca do MongoDB atualizou a forma como as URLs de conexão são analisadas. Definindo isso como `true`, você garante que está usando a versão mais recente do analisador de URL.

2. `useUnifiedTopology`: Quando definido como `true`, o Mongoose usará a nova camada de topologia unificada do driver do MongoDB. Essa camada unificada oferece uma implementação mais moderna e eficiente da gestão de conexões com o banco de dados. Definir isso como `true` é recomendado, pois a camada de topologia unificada é a escolha padrão a partir do MongoDB 4.0 e superior.

Portanto, essas opções são usadas para garantir que você esteja usando as últimas tecnologias e abordagens recomendadas pelo MongoDB ao estabelecer a conexão com o banco de dados, mantendo a compatibilidade com as versões mais recentes do driver e da biblioteca de cliente do MongoDB.

Como consultar informações do banco de dados?

const mongoose = require('mongoose');

const Schema = mongoose.Schema;

const postsSchema = new Schema({

    titulo: String,

    image: String,

    categoria: String,

    conteudo : String,

    slug: String

},{collation:'noticias'})

const post = mongoose.model('Posts', postsSchema)

module.exports = post