Modulo 05

Projeto #01 Criando uma plataforma de perguntas e respostas com Mysql

* Aula 02 versao Bootstrap para esse projeto

<https://getbootstrap.com/docs/4.5/getting-started/introduction/>

* Aula 03 EJS e criação de projeto

Criamos uma nova pasta chamada guiaPerguntas

Vamos iniciar um novo projeto node nessa pasta pelo cmd

npm init -y

vamos instalar o express e o ejs(engine template):

npm install express --save

npm install ejs --save

* Aula 04 Configurando e exibindo HTML com EJS

No arquivo index.js coloque o código

app.set('view engine', 'ejs')

crie uma pasta views em seu projeto, onde ficará todos os seus arquivos html

como estamos ultilizando o EJS, na estensão iremos usar o .ejs

crie um arquivo index.ejs na pasta views para ser nossa pagina principal

agora no arquivo index.js

iremos setar o caminho ‘/’, como sendo res.render e não res.send, dizendo o nome do arquivo que está na pasta views, nesse caso o arquivo index.ejs

index.js

app.get('/', (req, res) => {

    res.render('index')

})

Ao utilizar o render ele já vai direto na pasta views, e como se estivesse passando

app.get('/', (req, res) => {

    res.render('views/index')

})

Todo arquivo html deverá ficar na pasta views de seu projeto

* Aula 05 Exibindo variáveis no HTML

Com o EJS e possível passar variáveis do javascript para dentro do html

app.get('/', (req, res) => {

    let nome = 'Rodrigo'

    let lang = 'JavaScript'

    res.render('index')

})

Agora para passar esses valores vamos utilizar essa variáveis dentro do res.render()

index.js

app.get('/', (req, res) => {

    let nome = 'Rodrigo'

    let lang = 'JavaScript'

    res.render('index', {

        nome: nome,

        lang: lang,

        empresa: 'Guia do programador',

        inscrito: 8000

    })

})

Passamos para a view index 4 variaveis(nome, lang, empresa e inscrito)

Agora vamos pegar essa variáveis no arquivo index.ejs, usando a tag

<%= %>, passando o nome da variável

index.ejs

<body>

    <%= nome %>

    <%= lang %>

    <%= empresa %>

    <%= inscrito %>

</body>

Outra coisa que pode fazer e misturar essas tag com html

index.ejs

<body>

    <p>Nome: <%= nome %> </p>

    <p>Linguagem favorita: <%= lang %> </p>

    <p>Empresa: <%= lang %> </p>

    <p>Incritos: <%= inscrito %> </p>

</body>

Passagem de parâmetros obrigatorios direto no url

index.js

app.get('/:nome/:lang', (req, res) => {

    let nome = req.params.nome

    let lang = req.params.lang

    res.render('index', {

        nome: nome,

        lang: lang,

    })

})

Os parâmetros obrigatórios passados ficarão salvo em variáveis, que serão chamado no view index

Passagem de parâmetro opcinal

index.js

app.get('/:nome?/:lang?', (req, res) => {

    let nome = req.params.nome

    let lang = req.params.lang

    res.render('index', {

        nome: nome,

        lang: lang,

    })

})

Aqui temos dois parâmetros opacionais nome e lang, que podem ou não ser passado no endereço.

Se no endereço por passado paramentro ficara salvo na variável, se não for passado nada, nada ira ser mostrado

* Aula 06 Estrutura condicionais com EJS

<%= %> -> para chamar variaveis

<% if(){ %> -> para usar expressões

<% } %>

Ou

<% if(){ %> -> para usar expressões

<% } else { %>

<% } %>

index.js

app.get('/:nome/:lang', (req, res) => {

    let exibirMsg = true

    res.render('index', {

        msg: exibirMsg

    })

})

index.ejs

<body>

    <% if(msg == true){ %>

        <h3>Isso é uma msg de ERRO!</h3>

    <% }else { %>

        <h3>Nenhum erro!</h3>

    <% } %>

</body>

Nesse caso como a variável msg esta recebendo exibirMsg que por sua vez e true, será mostrada a msg de erro, de dentro do bloco

* Aula 07 Estrutura de repetição com EJS

index.js

app.get('/:nome/:lang', (req, res) => {

let produtos = [

        {nome: 'Doritos', preco: 3.14},

        {nome: 'Coca-Cola', preco: 5},

        {nome: 'Leite', preco: 1.45}

    ]

    res.render('index', {

        produtos: produtos

    })

})

No arquivo index.js, estamos enviando como uma variável produtos que está recendo uma lista de produtos, que no nosso arquivo de view poderemos usala chamando um forEach para ler todo o conteúdo dessa lista

index.ejs

<body>

<%= produtos[0].nome %>   <!-- Doritos -->

    <%= produtos[0].preco %>  <!-- 3.14 -->

    <%= produtos[2].nome %>   <!-- Leite -->

    <%= produtos[2].preco %>  <!-- 1.45 -->

</body>

Em nosso arquivo de views podemos recuperar esses valores que estão contido na variável produtos, podemos ultilizamos como desejarmos esses valores

Agora para percorrer essa variável precisamos utilizar um forEach como veremos no código abaixo:

<body>

    <% produtos.forEach(produto => { %> ABERTURA

       <p> <%= produto.nome %> </p>

       <p> <%= produto.preco %> </p>

    <% }) %>   FECHAMENTO

</body>

Observe que ao chamar a função de forEach a abertura dela está em uma tag ejs e seu fechamento em outra tag ejs, e como não e uma variável que estamos chamando e sim uma função a tag ejs não vem acompanhada do ‘=’, já o conteúdo que estamos pegando do forEach esse sim e uma variável por isso estamos chamando com o uso do ‘=’

index.ejs

<body>

    <table>

        <th>Produtos</th>

        <th>Preços</th>

        <% produtos.forEach(produto => { %>

            <tr>

                <td><%= produto.nome %> </td>

                <td><%= produto.preco %> </td>

            </tr>

        <% }) %>

    </table>

</body>

Podemos então montar nossa tabela da maneira que quisermos, ultilizando as tag html e recuperando o dados que foram passados.

* Aula 08 Arquivos estáticos

São arquivos que não são processados no seu backend, como arquivo css, imagens, Javascript do html.

Em nosso projetos criaremos uma pasta chamada public na raiz do nosso projetos, onde ficara todos esses arquivos estáticos, e em sequida informaremos ao express que ultilaremos essa pasta como arquivo estáticos informando o código abaixo:

index.js

app.use(express.static('public'))

por padrão no mercado, os arquivos estáticos ficam na pasta public

dentro dessa pasta public iremos criar uma pasta css, e dentro dela criaremos um arquivo style.css

e para utilizar essa folha de estilo iremos em nosso arquivo que desejamos utilizar esse css e fazer o link, como se fosse um html normal

index.ejs

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Principal</title>

    <link rel="stylesheet" href="/css/style.css">

</head>

A única direfença que no caminho para linkar o arquivo não e necessário colocar o caminho public, porem e necessário colocar a ‘/’

Para iniciar o endereço do arquivo

<link rel="stylesheet" href="../public/css/style.css">

Ficando assim:

<link rel="stylesheet" href="/css/style.css">

Mesma coisa seria para adicionar uma imagem, criaria uma pasta img dentro de public, colocaria uma imagem dentro dessa pasta e na para chamar esse arquivo usaria a tag img no arquivo de view

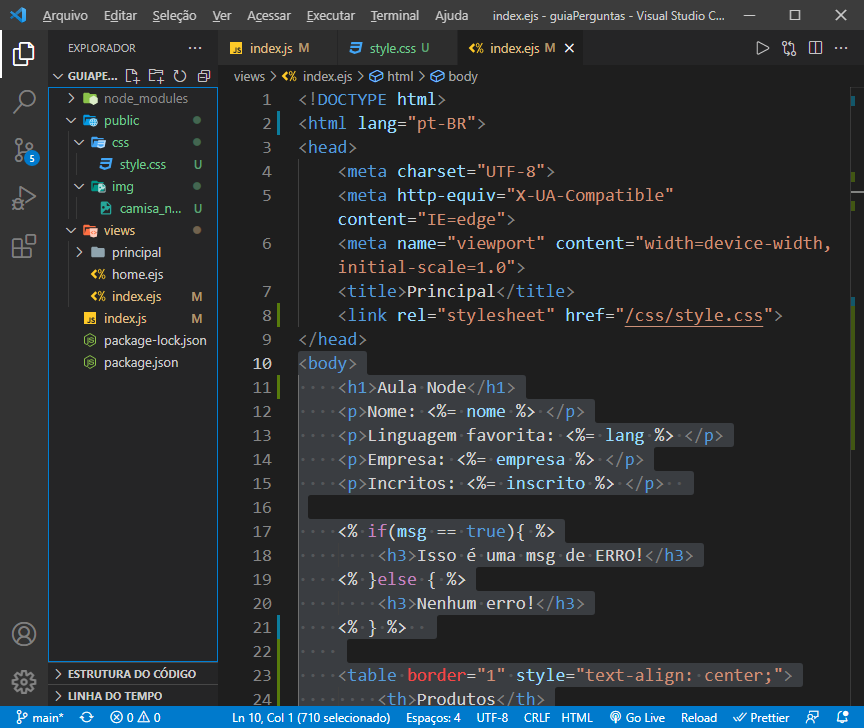
index.ejs

<body>

  <img src="/img/camisa\_node.jpg" alt="">

</body>

Seguindo o mesmo esquema de inserir imagem do html, so mudando o caminho, como esse arquivo está na pasta public, não e necessário informar esse caminho completo, somento o caminho apartir da pasta public começando por ‘/’.



Nosso projeto estará dessa maneira, pasta public para arquivo estáticos(css/img/js), pasta views(index.ejs/sobre.ejs) para as paginas ejs e nosso arquivo index.js sendo nosso arquivo principal onde estão os caminhos completos.

* Aula 09 Apresentando e instalando o Bootstrap

Para criar seu projeto com bootstrap, baixe os arquivos dele e coloque as pastas css e js(baixadas do boot), em seu projeto na pasta public, em sequida faça o link bootstrap.min dos arquivos tanto do css quando do js em seus arquivos de views que iram utilizar o boot

Importante e necessário copiar o link do jquery e do popper e colar em seu projeto também antes do link js do bootstrap

Terá que ficar na ordem, primeiro o jquery, depois popper e por ultimo do bootstrap.min.js, nessa ordem

Index.ejs

<head>

    ...

    <link rel="stylesheet" href="/css/bootstrap.min.css">

</head>

<body>

    <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js" integrity="sha384- DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkf j" crossorigin="anonymous"></script>

    <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.1/dist/umd/pop per.min.js" integrity="sha384- 9/reFTGAW83EW2RDu2S0VKaIzap3H66lZH81PoYlFhbGU+6BZp6G7niu735Sk7l N" crossorigin="anonymous"></script>

    <script src="/js/bootstrap.min.js"></script>

</body>

Feito isso seu bootstrap estará instalado na sua pagina do projeto

* Aula 10 Criando formulário de perguntas

Ensinamento bootstrap btn

* Aula 11 Adicionando card no formulário

Ensinamento bootstrap card

* Aula 12 Css customizado com bootstrap

Para se trabalhar com css junto com bootstrap, a chamada do link css deve vim depois da chamada do link do bootstrap, para surtir efeito

index.ejs

<link rel="stylesheet" href="/css/bootstrap.min.css">

    <link rel="stylesheet" href="/css/style.css">

* Aula 13 Finalizando front com navbar

Ensinando bootstrap navbar

* Aula 14 AVISO: nova forma de utilizar partials – Correção de erro

Antes <%- include partials/header.ejs %>

Agora <%- include (‘partials/header.ejs’) %>

* Aula 15 Reutilizando HTML com partials

Recurso do ejs, que são partes de códigos html que você pode reutilizar essas partes em outros html, assim um recurso e alterado, todos as paginas que tenha esse partials será alterado, sem precisar ficar alternado o código manuamente

**A = 38**

**A = 38**

**A = 38**

**Pagina 1**

**Partials Principal**

**Pagina 2**

Ao alterar o partials todas as paginas que tenham esse partials serão alteradas

**A = 57**

**A = 57**

**Pagina 1**

**Partials Principal**

**A = 57**

**Pagina 2**

Para se utilizar partials, e recomendado na pasta de views criar uma pasta chamada partials, para conter cada pedaço das parciais do seu código

Para incluir basta colocar o código ejs com include seguido do caminho do arquivo dentro de aspas e parênteses:

perguntar.ejs

<%- include ('partials/header.ejs') %>

<body>

    <%- include ('partials/navbar.ejs') %>

    <h1>Teste de partials</h1>

</body>

<%- include ('partials/footer.ejs') %>

* Aula 16 Envio de formulário para o Node

Para trabalhar com formulário, a primeira coisa e definir com qual método o formulário vai trabalhar, GET ou POST

<form method="GET">...</form>

Ou

<form method="POST">...</form>

GET -> quando os dados desse formulário forem enviado, ele será enviado usando o query params, logo não e tanto recomendado usa-lo formulário com GET

Ex: http://localhost:8080/perguntar?nome=Fulano&senha:123456

POST -> os dados não são passado na url e sim dentro da requisição,

Os dados meio que ficam escondidos ao olho do usuário

A segunda coisa que precisamos fazer e definir para qual rota será enviado esses dados

Para isso e preciso criar uma rota em seu node

index.js

app.post('/salvarpergunta', (req, res) => {

    res.send('Formulario Recebido')

})

Você utiliza rota do tipo post geralmente para receber dados de formulário

Agora em nosso formulário dizemos para onde ele irá apontar usando o action

perguntar.ejs

<form method="POST" action="/salvarpergunta">

    <label for="">Título:</label>

    <input class="form-control" type="text" placeholder="Título">

</form>

Ao enviar esse formulário os dados serão enviado para o caminho

http://localhost:8080/salvarpergunta

agora se precisamos recuperar esses dados enviados

* Aula 17 Capturando dados de um formulário

Para pegar os dados enviado pelo express precismos istalar uma biblioteca chamada body parser em seu projeto

npm install body-parser --save

agora precisamos import o body parser em nosso projeto

index.js

const bodyParser = require('body-parser')

depois temos que linkar o body parser no express, porque o body parser e resposavel por traduzir os dados enviados pelo formulário em uma estrtura Javascript que consiga utilizar no backend

index.js

app.use(bodyParser.urlencoded({extended:false}))

depois podemos colocar um comando opcional bem útil, que permite que a gente leia dado de formulário enviado via json, muito utilizado no uso de api

index.js

app.use(bodyParser.json())

feito isso seu projeto já terá suporte ao body parser, importante que seus campos no formulário tenha nome para que eles possam ser identificados no seu backend

perguntar.ejs

<input type="text" name="titulo">

<textarea name="descricao"></textarea>

E para pegar esse dado passado usamos o código abaixo

index.js

app.post('/salvarpergunta', (req, res) => {

  let titulo = req.body.titulo

  let descricao = req.body.descricao

  res.send('Formulario Título: '+titulo +' Descrição: '+descricao)

})

Criamos uma variável e pegamos pela requisição feita no body e o nome da tag que queremos pegar

Nesse caso estamos pegando o valor passado no input que tem o name titulo e descricao

* Aula 18 MySQL Workbench

Interface gráfica para o mysql

* Aula 19 Apresentando e instalando o Sequelize