Desenvolvimento Backend

Aula 11 - implementando interface em EJS

Material Didático do Instituto Metrópole Digital - IMD

Termo de uso

Apresentação

Nesta aula, iremors pôr em prática o conhecimento adquirido no *EJS* através do desenvolvimento de um blog.

Objetivos

- Conhecer sobre a criação de layout;
- Conhecer como tratar os dados;
- Conhecer como utilizar datas na aplicação.

Apresentação e organização do projeto

Link do video da aula: https://youtu.be/W1FEGFZl97c

Na disciplina de *front-end* foi desenvolvido um layout de um blog com conteúdo estático. Porém, agora é preciso dar vida a essa página.

Baixando o projeto

Para ter acesso ao projeto *front-end* baixe, o arquivo aqui.

Organizando as rotas

Para essa nova etapa, podemos limpar nossos arquivos, retirando códigos que não serão utilizados. Então, remova as rotas de **cursos**, **alunos** e **/** criadas anteriormente.

Após essa remoção, é preciso fazer a diferenciação entre as rotas que retornam *json*, no caso a *api*, e as rotas que renderizam uma página HTML. Para isso, a ideia é colocar um prefixo */api* em todas as rotas da *api*. Ao término das alterações, o arquivo deve ficar assim:

```
const express = require('express')
const rotaUsuario = require('./rotas/usuario.rota')
const rotaPost = require('./rotas/posts.rota')
var expressLayouts = require('express-ejs-layouts')

const indexRoute = require('./rotas/index.rota')

const app = express()
```

```
    app.use(express.json())
    app.set('view engine', 'ejs')

    app.set('layout', 'layouts/layout')

    app.use(expressLayouts)

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

    app.use('/api/usuarios', rotaUsuario)
    app.use('/api/posts', rotaPost)

    app.get('/api', (req, res) => {
            res.json({msg: "Hello from Express!"})
    })

    app.listen(8080, () => {
            console.log(`Iniciando no ambiente
${process.env.NODE_ENV}`)
            console.log('Servidor pronto na porta 8080')
    })
```

Por fim, acompanhe a aula e altere os arquivos *.http* para fazer os testes.

Populando dados

Link do video da aula: https://youtu.be/dByi-cvvLoo

Agora, iremos trabalhar com os dados, para isso será deletado o arquivo *dev.sqlite*, que corresponde ao banco de dados de desenvolvimento.

Recriando o banco

Após apagar o banco antigo, precisamos agora de um novo. Para isso, execute no terminal o comando:

```
    npm run migrate-dev
```

Isso vai gerar um novo arquivo de banco de dados de desenvolvimento, mas agora as tabelas estarão vazias.

Preenchendo o banco

Para ter um banco já populado, acompanhe a aula e utilize o arquivo *usuario.http* para criar usuários novos no banco. Após isso, com o auxílio do *Insominia*, adicione novos *posts* à tabela.

Criando layout do Blog em EJS (Parte 1)

Link do video da aula: https://youtu.be/ejhgiYF-dO0

Iniciando a construção do layout do blog iremos até o projeto disponibilizado na primeira aula e abriremos o arquivo *index.html*. Esse arquivo é uma página inicial, totalmente estática, de um blog.

Criando o layout do blog

Para prosseguir, devemos manter somente o conteúdo que não sofre alterações, por exemplo: cabeçalho, rodapé e barra lateral. Então, crie um arquivo na pasta *layout* nomeando-o de '*layout-blog.ejs*' e adicione o código abaixo:

```
    <!DOCTYPE html>
    <html lang="en">

    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-
scale=1.0">
        <meta http-equiv="X-UA-Compatible" content="ie=edge">
        <title>Blog do IMD</title>

        <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstr
ap.min.css"
            integrity="sha384-
ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T"
crossorigin="anonymous">

    </head>

    <body>
```

```
    <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
        <div class="container">
            <a class="navbar-brand" href="#">Blod do IMD</a>
            <button class="navbar-toggler" type="button" data-
toggle="collapse" data-target="#navbarNav"
                aria-controls="navbarNav" aria-expanded="false"
aria-label="Alterna navegação">
                <span class="navbar-toggler-icon"></span>
            </button>
            <div class="collapse navbar-collapse" id="navbarNav">
                <ul class="navbar-nav">
                    <li class="nav-item active">
                        <a class="nav-link" href="#">Home <span
class="sr-only">(Página atual)</span></a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="#">Clientes</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="#">Serviços</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="#">Contanto</a>
                    </li>
                </ul>
            </div>
        </div>

    </nav>

    <nav aria-label="breadcrumb">
        <ol class="breadcrumb">
            <li class="breadcrumb-item active" aria-
current="page">Home</li>
        </ol>
    </nav>

        <div class="container">
            <div class="row">
                    <div class="col-md-8">
                            <%- body %>
                    </div>
                    <div class="col-md-4">
```

```
                            <div class="card mb-4">
                                <div class="card-header">
                                    Busca
                                </div>
                                <div class="card-body">
                                    <div class="input-group">
                                        <input class="form-control"
type="text" placeholder="Digite aqui...">
                                        <button class="btn btn-
secondary ml-2">Buscar</button>
                                    </div>
                                </div>
                            </div>

                            <div class="card mb-4">
                                <div class="card-header">
                                    Categorias
                                </div>
                                <div class="card-body">
                                    <dib class="row">
                                        <div class="col-lg-6">
                                            <ul class="list-
unstyled">
                                                <li><a href="#">Web
Design</a></li>
                                                <li><a
href="#">Javascript</a></li>
                                                <li><a
href="#">CSS</a></li>
                                            </ul>
                                        </div>
                                        <div class="col-lg-6">
                                            <ul class="list-
unstyled">
                                                <li><a href="#">Web
Design</a></li>
                                                <li><a
href="#">Javascript</a></li>
                                                <li><a
href="#">CSS</a></li>
                                            </ul>
                                        </div>
                                    </dib>
                                </div>
```

```
                            </div>

                            <div class="card mb-4">
                                <div class="card-header">
                                    Autor
                                </div>
                                <div class="card-body">
                                    <div class="text-center">
                                        <img
src="/static/imgs/avatar.jpg" class="rounded-circle" alt="">
                                    </div>
                                    <p class="py-3">Lorem ipsum
dolor sit amet consectetur, adipisicing elit. Ex, sit architecto
                                        tenetur, odit impedit quam
corporis a libero assumenda suscipit fugiat in nesciunt labore
                                        pariatur dolor ipsam sed
ducimus nemo.</p>
                                </div>
                            </div>

                        </div>
            </div>

        </div>

    <footer class="bg-dark py-3">
        <p class="text-white text-center">Desenvolvido pelo IMD</p>
    </footer>

        <script
src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
            integrity="sha384-
q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo"
            crossorigin="anonymous"></script>
        <script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.3/umd/pop
per.min.js"
            integrity="sha384-
ZMP7rVo3mIykV+2+9J3UJ46jBk0WLaUAdn689aCwoqbBJiSnjAK/l8WvCWPIPm49"
            crossorigin="anonymous"></script>
        <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/js/bootstrap
.min.js"
            integrity="sha384-
```

```
ChfqqxuZUCnJSK3+MXmPNIyE6ZbWh2IMqE241rYiqJxyMiZ6OW/JmZQ5stwEULTy"
            crossorigin="anonymous"></script>

    </body>

    </html>
```

Criando uma nova rota

Após o layout criado, é preciso criar uma rota para que quando o usuário acesse essa rota seja realizado algum processamento e renderizada a página. Então, na pasta de "rotas", crie um arquivo chamado *index.rota.js* e implemente o código abaixo:

```
const express = require('express')
const router = express.Router()

router.get('/', async (req, res) => {
    res.render('pages/post', {layout: 'layout/layout-blog.ejs'})
})

module.exports = router
```

Criando a página

A rota anterior irá buscar um arquivo que ainda não existe na pasta *pages*. Então, é preciso criá-lo com o nome de ''*posts.ejs* e implementar o código que será injetado na marcação feita no *layout*. A implementação deve ser feita conforme se vê abaixo:

```
<%- contentFor('body') %>

<p> Conteúdo injetado! </p>
```

Adicionando a rota ao *express*

Apesar de ter criado a rota, o *express* ainda não pode enxergá-la. Então, no arquivo *index.js* inclua as linhas abaixo:

```
    const indexRoute = require('./rotas/index.rota')

    app.use('/', indexRoute)
```

Criando layout do Blog em EJS (Parte 2

Link do video da aula: https://youtu.be/DJJlkqd55mg

Agora a página já tem um conteúdo injetado, porém, é preciso fazer a captura dos dados no banco de dados para criar a *interface*.

Buscando os *posts*

A ideia é que na rota *index.rota.js*, antes de renderizar o layout, seja feita uma consulta ao banco e passadas as informações à página.

Para buscar os *posts* no banco, acompanhe a aula alterando o arquivo '*index.rota.js*'. O código deve ficar conforme se observa a seguir:

```
const express = require('express')
const router = express.Router()
const { Post, Usuario } = require('../db/models')

router.get('/', async (req, res) => {
    const posts = await Post.findAll({
        include: [{
           model: Usuario
        }], raw: true, nest: true
    })

    res.render('pages/posts', {posts: posts, layout:
'layouts/layout-blog.ejs'})

})

module.exports = router
```

Manipulando os dados para apresentação

Link do video da aula: https://youtu.be/-z_f5ZNGjPg

Os dados do banco já estão sendo capturados, porém, em um formato não usual

para mostrar ao usuário, e também existem informações que não devem ser exibidas.

Tratando os dados

Para remover os dados desnecessários, será implementada uma função que irá deletar os dados que não precisam ser retornados para a criação da página.

Agora, já temos os dados retornados. No entanto, ainda é preciso enviar datas e, além de enviá-las, transformá-las para o formato que deve ser exibido no layout. Para tratar disso, instale a biblioteca *Moment.js* pelo comando:

```
    npm install moment --save
```

Após isso, já poderá ser feito o uso da biblioteca e a data será formatada. Para isso, siga a aula e implemente o código que fará essa manipulação.

Ao final, seu código deve estar como se vê abaixo:

```
    const express = require('express')
    const router = express.Router()
    const { Post, Usuario } = require('../db/models')
    const moment = require('moment')
    moment.locale('pt-br')

    router.get('/', async (req, res) => {
            const posts = await Post.findAll({
                    include: [{
                        model: Usuario
                    }], raw: true, nest: true
            })

            const postResult = posts.map((post) =>
prepararResultado(post))
            res.render('pages/posts', {posts: postResult, layout:
'layouts/layout-blog.ejs'})
    })

    function prepararResultado(post){

            const result = Object.assign({}, post)
            result.postadoEm = moment(new
Date(result.createdAt)).format('DD [de] MMMM [de] yyyy [as] HH:mm')
```

```
            if (result.createdAt) delete result.createdAt
            if (result.updatedAt) delete result.updatedAt
            if (result.userId) delete result.userId
            if (result.Usuario){
                    if (result.Usuario.senha) delete
result.Usuario.senha
                    if (result.Usuario.createdAt) delete
result.Usuario.createdAt
                    if (result.Usuario.updatedAt) delete
result.Usuario.updatedAt
            }
            return result
    }

    module.exports = router
```

Listando postagens

Link do video da aula: https://youtu.be/vcNDbDD9eqo

Nesse momento, os dados já estão tratados e prontos para serem exibidos. Acompanhe a aula e organize a forma como essas informações serão exibidas.

Criando o *layout* com os dados recebidos

Inicialmente, remova as pastas **alunos** e **cursos** da pasta *view*, pois elas não serão mais usadas. Após isso, abra o arquivo '*posts.ejs*' para implementação da listagem de *posts*.

```
<%- contentFor('body') %>

<h1 class="mb-4">
  <small>Posts Recentes</small>
</h1>

<hr />

<% posts.forEach(function (post) { %>

<div class="card mb-4">
  <img
    class="card-img-top"
```

```
    src="<%=post.foto%>"
    alt="Imagem de capa do card"
  />

  <div class="card-body">
    <h5 class="card-title"><%=post.titulo%></h5>
    <p class="card-text">
      <%-post.texto.substr(0,200)%>...
    </p>
    <a href="/post/<%=post.id%>" class="btn btn-primary">Leia
mais</a>
  </div>
  <div class="card-footer text-muted">
    Postado por <a href="#"><%=post.Usuario.email%></a>
<%=post.postadoEm%>
  </div>
</div>

<% }) %>
```

Exibindo detalhes de uma postagem

Link do video da aula: https://youtu.be/KMeU3jT6gBI

Até o momento, já temos a página principal do blog, no entanto, ainda falta fazer o botão "Leia Mais" abrir uma página interna com todo o texto do *post*.

Limitando o número de *posts*

A página principal está carregando todos os *posts* do banco de dados, e isso pode gerar problemas de performance futuramente. Para evitar isso, é importante colocar um limite de *posts* que devem ser exibidos.

Então, na rota '/' principal do arquivo '*index.rota.ejs*', adicione o limite e a ordem de ordenação. Seguindo a aula, a rota deverá ficar como se vê abaixo:

```
router.get('/', async (req, res) => {
    const posts = await Post.findAll({
        limit: 10,
        order: [['createdAt', 'DESC']],
        include: [{
            model: Usuario
        }], raw: true, nest: true
```

```
    })

    const postResult = posts.map((post) => prepararResultado(post))
    res.render('pages/posts', {posts: postResult, layout:
'layouts/layout-blog.ejs'})
})
```

Buscando um *post* específico no banco

Para o funcionamento do botão "Leia Mais", é preciso buscar, no banco, somente pelo *post* que usuário está querendo ver no blog e redirecionar para uma página com as informações completas. Então, ainda no arquivo '*index.rota.ejs*', crie uma rota similar a esta que apresento a seguir:

```
router.get('/post/:id', async (req, res) => {
    const post = await Post.findByPk(req.params.id,
        {include: [{model: Usuario}], raw: true, nest: true})
    res.render('pages/post', {post:prepararResultado(post), layout:
'layouts/layout-blog.ejs'})
})
```

Criando a página

Com os dados disponíveis para uso, é necessário criar a página que será exibida. Portanto, na pasta *pages*, crie um arquivo e o nomeie de '*post.ejs*'.

Já com o arquivo criado, preencha-o com o código apresentado a seguir:

```
 <%- contentFor('body') %>

<h1 class="mb-4">
    <%=post.titulo%>
</h1>
<p class="lead">Postado por <a
href="#"><%=post.Usuario.email%></a></p>
<hr>
<p class="text-muted">postado em <%=post.postadoEm%></p>
<hr>
<img class="img-fluid rounded mb-4" src="<%=post.foto%>"
alt="Computador">

<%-post.texto%>
```

Resumo

Nesta aula, foram revisados conceitos e práticas já vistas anteriormente, contudo dessa vez aplicados a um blog funcional. Ademais, foi mostrado como usar a API que está sendo desenvolvida em um projeto front-end já pronto.

Também foram inseridas novas ferramentas, como o *Moment.js*, que trabalha com datas.