PROJETO PARA CONSTRUIR UM BLOG COM EDITOR DE TEXTO ÁREA DE LOGIN E PAINEL ADMINISTRATIVO

1. Tipos de relacionamentos :
   1. - 1 para 1 – Uma atividade pertence a uma Aula.
   2. - 1 para N – Um vendedor atende N clientes
   3. - N para N – Um produtos está em varias notas, Uma nota possui vários produtos,
2. Base do projeto
3. Vamos instalar todas as bibliotes, como o express, o body-parser, o sequelize e o ejs.
4. Vamos configurar o projeto com bootstrap, vamos criar o layout inicial;
5. Cria a pasta do projeto, ‘GuiaPress’

NO TERMINAL

1. Entrar na pasta do projeto
2. Digitar ‘npm init’ para iniciar a estrutura do projeto
3. Digitar ‘npm install –save express’
4. Digitar ‘npm install –save sequelize’
5. Digitar ‘npm install –save mysql2’ – dependência do sequelize para se comunicar com o mysql.
6. Digitar ‘npm install –save body-parser’ – biblioteca do express para trabalhar com formulários
7. Digitar ‘npm install –save ejs’ – que é a template engine para renderizar html

NA PASTA DO PROJETO

1. Criar o arquivo principal index.js

ABRIR O ARQUIVO index.js

1. Importar o express para dentro do index.js – ‘const express = require(‘express’);’
2. Criar uma instância do express – ‘const exp = express();
3. Criar a rota principal – ‘app.get(‘/’,(req, res)=>{res.sender(“Bem vindo ao meu site ’)};
4. Iniciando a aplicação – ‘app.listen(8080,()=>{ console.log(“O servidor está rodando)});
5. Testar a aplicação com – ‘nodemon index.js’;
6. Aapós a crianção da instância do express, configurar a engine ejs - ‘app.set(‘view engine’, ‘ejs’);’

NA PASTA DO PROJETO

1. Criar a pasta Views

NA PASTA VIEWS

1. Criar o arquivo ‘index.ejs’ – primeira view do projeto

ABRIR O ARQUIVO INDEX.JS

1. Após a configuração da view engine, configurar também o express.static para configurar a pasta dos arquivos estáticos(css, img, js)
2. ‘app.use(express.static(‘public’));

NA PASTA DO PROJETO

1. Criar a pasta public

ABRIR O MYSQLWORKBECH

1. Criar o banco de dados chamado ‘guiapress’

NA PASTA DO PROJETO

1. Criar a pasta database

NA PASTA database

1. Criar o arquivo database.js
2. Importar o sequelize – ‘const Sequelize = require(‘selequelize);’
3. Criar a conexão com o banco de dados – ‘const connection = new Sequelize(‘guiapress’,’root’,’123456’,{ host: ‘localhost’, dialect:’mysql’});
4. Exportar a conexão – ‘module exports = connection;’

NO ARQUIVO INDEX.JS

1. Importar a conexão do banco de dados – ‘const connection = require(‘./database/database’);
2. Realizar a autenticação com o banco de dados – ‘connection.authenticate().then(()=>{ console.log(“conexão feita com sucesso!”).catch((error)=>{console.log(error)});

NA PÁGINA DO BOOTSTRAP

1. Baixar o bootstrap

NA PASTA public

1. Colar conteúdo do bootstrap

NA PASTA views

1. Criar a pasta partials

NA PASTA partials

1. Criar o arquivo header.ejs,

NO ARQUIVO header.ejs

1. Digitar
2. <!DOCTYPE html>
3. <html lang="pt-br">
4. <head>
5. <meta charset="UTF-8">
6. <meta http-equiv="X-UA-Compatible" content="IE=edge">
7. <meta name="viewport" content="width=device-width, initial-scale=1.0">
8. <title>GuiaPress</title>
9. <link rel="stylesheet" href="/css/bootstrap.min.css">
10. </head>
11. <body>

NA PASTA partials



1. Criar o arquivo footer.ejs

NO ARQUIVO footer.ejs

1. Digitar
2. </body>
3. <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js" integrity="sha384-DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj" crossorigin="anonymous"></script>
4. <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.1/dist/umd/popper.min.js" integrity="sha384-9/reFTGAW83EW2RDu2S0VKaIzap3H66lZH81PoYlFhbGU+6BZp6G7niu735Sk7lN" crossorigin="anonymous"></script>
5. <script src="/js/bootstrap.min.js"></script>
6. </html>

NA PASTA views

1. Criar o arquivo principal index.ejs

NO ARQUIVO index.ejs

1. Incluir o header da página
2. Digitando <%- include ('partials/header.ejs'); %>
3. Incluir o footer da página
4. Digitando <%- include ('partials/footer.ejs'); %>
5. OBS: O conteúdo da página deve estar localizada entre o header e o footer
6. Para melhorar a arquitetura e organização do projeto vamos criar várias pastas para separar a lógica do programa por assuntos no formato MVC

NA PASTA DO PROJETO

1. Criar as pastas categories e articles

NA PASTA categories

1. Criar o arquivo categoriesController.js

NO ARQUIVO categoriesController

1. Importar o express – ‘const express = require(‘express’);’
2. Carregar o router que será um objeto para carregar as rotas
3. Const router = express.Router();
4. Criar a rota categories
5. Digite router.get(‘categories’,(req,res)=>{ res.send(‘Rota de categorias’)});
6. Criar a rota para criar nova categoria
7. Digite router.get(‘/admin/categories/new’,(req,res)=>{res.send(‘Rota para criar categorias)});
8. Exportar o arquivo do router para que ele possa ser utilizado no arquivo principal
9. Digite module.exports = router;

NO ARQUIVO index.js

1. Importar o router
2. Digite ‘const categoriesController = require(‘./categoriesController/CategoriesControlle’);
3. Digite ‘app.use(‘/’,categoriesController);’ para o aplicativo poder utilizar o conteúdo do controller CategoriesController; e a rota funcionar.
4. O ‘/’ anterior serve somente como prefixo ou seja para se acessar a rota de categorias é necessário antes colocar o prefixo. Poderia ser outro qualquer.

NA PASTA articles

1. Criar outro arquivo com o nome ArticleController que conterá as rotas dos articles

NO ARQUIVO ArticleController

1. Copiar todo conteúdo do controller CategoriesController e colar neste arquivo em branco
2. Alterar as rotas substituindo por articles

NA PASTA categories

1. Criar o arquivo Category.js que será o model de categorias

NO ARQUIVO Category.js

1. Importar o sequelize – ‘const Sequelize = require(‘selequelize’);
2. Importar a conexão com o banco de dados
3. Digitar ‘const connection = require(‘../database/database’);
4. Definir o model com os campos que irão compor a tabela
5. Const Category = connection.define(‘categories’, {
6. Tilte:{
   * 1. Type: Sequelize.STRING,
     2. allowNull: false}, slug:{ type: Selequize.TEXT, allowNull: false}});
7. Criando a tabela
8. Digite ‘Article.sync({force:true});
9. OBS: Deve-se comentar ‘Article.sync após a tabela ser criada no banco de dados para não ocorrer erros
10. Exportando o model
11. Digite module.exports = Category;

NA PASTA articles

1. Criar o arquivo Article.js que será o model da tabela articles

NO ARQUIVO Article

1. Importar o sequelize – ‘const Sequelize = require(‘selequelize’);
2. Importar a conexão com o banco de dados
3. Digitar ‘const connection = require(‘../database/database’);
4. Importar o outro model que fará parte do relacionamento
5. Const Category = require(‘./categories/categories’);
6. Definir o model com os campos que irão compor a tabela
7. Const Article = connection.define(‘articles, {
8. Tilte:{
   * 1. Type: Sequelize.STRING,
     2. allowNull: false}, slug:{ type: Selequize.TEXT, allowNull: false},
     3. body:{ type:Sequelize.TEXT, allowNull:false}});
9. Definindo os relacionamento
10. Relacionamento um para um
11. Digite ‘Article.belongsTo(Category);’
12. Relacionamento um para muitos
13. Digite ‘Category.hasMany(Article);
14. Criando a tabela, após criar a tabela no banco de dados, deve-se comentar a linha abaixo
15. Digite ‘Article.sync({force:true});
16. Exportando o model
17. Digite module.exports = Article;

NO ARQUIVO index.js

1. Importar os models
2. ‘const Article = require(‘./articles/Article’);’
3. ‘const Category = require(‘./categories/Category’);’

08/08/21

NA PASTA views

1. Criar uma pasta chamada admin que terá todas as páginas referentes a cadastros e gerenciamento do site.

NA PASTA admin

1. Cria uma pasta ‘categories’ para armazenar todas as páginas de gerenciamento de categorias

NA PASTA categories

1. Criar o arquivo new.ejs que será a página para cadastro das categorias

NO ARQUIVO new.ejs

1. Incluir os arquivos partials digite:
2. <%- include(‘../../partials/header’);
3. <%- include(‘../../partials/footer’);
4. OBS: todo conteúdo da página deve estar entre o header e o footer

NO ARQUIVO categorieController

1. Alterar a rota que foi criada para acessar a página new
2. Digitar dentro da rota ‘res.render(‘admin/categories/new’);
3. Dentro do método render deve-se digitar todo o endereço quando o arquivo não estiver em view.

NA PASTA partials

1. Criar o arquivo navbar.ejs que será a navbar das páginas

NO ARQUIVO navbar.ejs

1. Digitar
2. <nav class="navbar navbar-light bg-primary">
3. <a class="navbar-brand" href="/">
4. GuiaPress
5. </a>
6. </nav>

NO ARQUIVO index.ejs

1. Incluir a nav bar após o header da página
2. <%- include ('partials/navbar.ejs'); %>

NO ARQUIVO new.ejs

1. Incluir a navbar após o header da página
2. <%- include ('../../partials/header.ejs'); %>

NO ARQUIVO categoriesController

1. Criar a rota para receber os dados do formulário da pagina new e salvar na tabela categorie.
2. Digite
3. //criando a rota para receber os dados do formulário e salvar na tabela categories
4. router.get('/categories/save',(req, res)=>{
5. //pegar os dados do formulário criando uma variável que
6. //irá armazenar o valor do campo input 'title'
7. var title = req.body.title;
8. //verificar se o título existe
9. if(title != undefined){
11. }else{
12. //redirecionando o usuário para a página news pois ele não
13. //o campo de title está vazio
14. res.redirect('/admin/categories/new');
15. }
16. });
17. Importar o modulo de categoria para acesso a tabela categories digite
18. const category = require('../categories/Category');
19. instalar a bibliote Slugfy do node para salvar o slug, que será o título sem espaço ou ponto. As palavras serão separadas por um ‘ – ‘;

NO TERMINAL

1. Digite ‘npm install –save slugify’

NO ARQUIVO categoriesController

1. Carregar o slugfy digite no topo:
2. ‘const slugify = require(‘slugigy’);’
3. Após if(title !=undefined) digite
4. ‘Category.create({
5. title: title,
6. //slug é a versão otimizada para url utilizando a bibliteca
7. // do node chamada slugify
8. slug: slugify(title)
9. });
10. Para capturar dados de formulários é necessário realizar a configuração do express digite
11. //utilizando o express.urlencoded para trabalhar com formulários
12. router.use(express.urlencoded({extended:false}));
13. //configuracao para utilizar os dados do formulário em json
14. router.use(express.json());
15. Criando rota para realizar consulta à tabela categories
16. Digite
17. /criando rota para pegar todos os dados da tabela categories e
18. //mostrá-los na página index.ejs da pasta categorias
19. router.get('/admin/categories',(req,res)=>{
20. //busca todos os registros sem filtro 'where'
21. Category.findAll({
22. raw: true,
23. order:[['title','ASC']]
24. }).then(categories =>{
25. res.render('admin/categories/index',{categories: categories})
26. });
27. });

NA PASTA views/categories

1. Crie o arquivo index.ejs

NO ARQUIVO views/categories/index.ejs

1. Faça os includes do header, navbar e footer
2. Construa a tabela conforme abaixo:
3. <div class="container">
4. <hr>
5. <h2>Categorias</h2>
6. <hr>
7. <table class="table table-bordered">
8. <thead>
9. <tr>
10. <th>Id</th>
11. <th>Título</th>
12. <th>Slug</th>
13. <th>Ações</th>
14. </tr>
15. </thead>
16. <tbody>
17. <% categories.forEach(category=>{ %>
18. <tr>
19. <th><%=category.id %></th>
20. <th><%=category.title %></th>
21. <th><%=category.slug %></th>
22. <th></th>
23. </tr>
24. <%} ) %>
25. </tbody>
26. </table>
27. </div>

NO ARQUIVO navbar.ejs na pasta views/partials

1. Digite
2. <!--Para criar um menu utilizamos a class = navbar-nav
3. mr-auto serve para configurar uma margem total à direita
4. -->
5. <ul class="navbar-nav mr-auto ">
6. <!--Para criar um item  utilizamos a class = nav-item -->
7. <li class="nav-item">
8. <!--O link do item do menu deve ter a class = nav-link
9. para aparecer no menu -->
10. <a class="nav-link" href="/admin/categories">Categorias</a>
11. </li>
12. </ul>

NO ARQUIVO views/categories/index.ejs

1. No arquivo altere o texto do forEach conforme abaixo
2. <% categories.forEach(category=>{ %>
3. <tr>
4. <th><%=category.id %></th>
5. <th><%=category.title %></th>
6. <th><%=category.slug %></th>
7. <th>
8. <a class="btn btn-warning" href="">Editar</a>
9. <a class="btn btn-danger" href="" >Excluir</a>
10. </th>
11. </tr>
12. <%} ) %>
13. Incluir o botão de criar nova categoria abaixo do <hr> do topo da página digite:
14. <a class="btn btn-success" href="../../admin/categories/new" >Criar nova Categoria</a>
15. NO ARQUIVO CategoriesController
16. Criar uma rota para apagar uma categoria específica
17. //rota para realizar a exclusão de uma categoria
18. router.post('/categories/delete',(req, res)=>{
19. var id = req.body.id;
20. if((id != undefined) || (!isNaN(id))){
21. //deletando a categoria
22. Category.destroy({
23. where: {
24. id: id
25. }
26. }).then(()=>{
27. res.redirect('/admin/categories');
28. });
29. }else{
30. res.redirect('/admin/categories');
31. }
32. });

NO ARQUIVO views/categories/index.ejs

1. Configure um form para ao clicar um botão ir para a rota /categories/dele enviando o valor do id da categoria.
2. Substitura o último <th> do forEach.
3. <th>
4. <button class="btn btn-warning">Editar</button>
5. <form  method="POST" action="/categories/delete" style="display: inline;">
6. <input type="hidden" name="id" value="<%=category.id%>">
7. <button class="btn btn-danger">Excluir</button>
8. </form>
9. </th>
10. No final do arquivo após o include do footer. Criar a função
11. Digite :
12. !--criando um script para confirmar a exclusão da categoria-->
13. <script>
14. function confirmarDelecao(event, form){
15. event.preventDefault();
16. //verifica se o usuário quer deletar
17. var decision = confirm('Você quer deletar a categoria?');
18. if(decision){
19. form.submit();
20. };
21. };
22. </script>

NO ARQUIVO CategoriesController.js

1. Criar a rota para abrir o formulário de editar categorias
2. Digite
3. //rota para abrir o formulário para editar a categoria
4. router.get('/admin/categories/edit/:id',(req, res)=>{
5. var id = req.params.id;
6. //será realizado uma pesquisa pelo PK
7. Category.findOne({Where:{id:id}).then(categoria=>{
8. if(categoria != undefined){
9. res.render('admin/categories/edit',{categoria: categoria});
10. }else{
11. res.redirect('/admin/categories');
12. };
13. }).catch(erro=>{
14. res.redirect('/admin/categories');
15. });
16. });

NA PASTA views/categories

1. Criar o arquivo edit.ejs que será a página para edição das categorias

No ARQUIVO edit.ejs

1. Digite
2. <%- include ('../../partials/header.ejs'); %>
3. <%- include ('../../partials/navbar.ejs'); %>
4. <div class="container">
5. <br>
6. <div class="card">
7. <div class="card-header">
8. <h2>Edição de categorias</h2>
9. </div>
10. <div class="card-body">
11. <form method="POST" action="/categories/update">
12. <input type="hidden" name="id" value="<%=category.id %>">
13. <input class="form-control" type="text" name="title" placeholder="Defina o título da categoria" value="<%=category.title%>">
14. <br>
15. <button class="btn btn-primary">Atualizar</button>
16. </form>
17. </div>
18. </div>
19. </div>
20. <%- include ('../../partials/footer.ejs'); %>

NO ARQUIVO categoriesController

1. Crie uma rota para receber os dados do formulário de edição e salvar no banco de dados.
2. Digite
3. /rota para salvar a alteração da categoria
4. router.post('/categories/update',(req, res)=>{
5. var title = req.body.title;
6. var id = req.body.id;
7. var slug = slugify(title);
8. //alterando o valor do campo
9. Category.update({
10. title: title,
11. slug: slug
12. },{where:{
13. id:id
14. }
15. }).then(()=>{
16. res.redirect('/admin/categories');
17. });
18. });

NA PASTA views/admin

1. Crie a pasta articles

NA PASTA views/admin/articles

1. Crie o arquivo new.ejs para cadastrar os artigos

NO ARQUIVO articles/new.ejs

1. Cole o conteúdo do arquivo categories/new.ejs e faça as alterações para mostrar os campos que devem ser preenchidos em categories
2. Digite
3. <%- include ('../../partials/header.ejs'); %>
4. <%- include ('../../partials/navbar.ejs'); %>
5. <div class="container">
6. <br>
7. <div class="card">
8. <div class="card-header">
9. <h2>Novo Artigo</h2>
10. </div>
11. <div class="card-body">
12. <form method="POST" action="/articles/save">
13. <label for="tile">Título:</label>
14. <input class="form-control" type="text" name="title" placeholder="Defina o título do artigo">
15. <label for="cars">Categoria:</label>
16. <select class="form-control" id="cars" name="cars" >
17. <% categories.forEach(category=>{ %>
18. <option value="category.title"><%=category.title%></option>
19. <%})%>
20. </select>
21. <label for="body">Conteúdo:</label>
22. <textarea class="form-control" type="text" name="body" placeholder="Digite o conteúdo do artigo" ></textarea>
24. <br>
25. <button class="btn btn-primary">Cadastrar</button>
26. </form>
27. </div>
28. </div>
29. </div>
30. <%- include ('../../partials/footer.ejs'); %>

NO ARQUIVO articles/ArticlesControllers

1. importe o slugify digite ‘const slug = require(‘slugijy’);’
2. importe os models de Category e Articles
3. //importando o model de categorias e artigos
4. const Category = require('./../categories/Category');
5. const Article = require('./Article');
6. Crie a rota para abrir a página de cadastro de categorias
7. Digite
8. //criando a rota de categorias e enviando as categorias para a página new.ejs
9. router.get('/admin/articles',(req,res)=>{
10. Category.findAll({
11. raw:true,
12. order:[['id','DESC']]}).then(categories=>{
13. res.render('admin/articles/new',{categories:categories});
14. });
15. });

BAIXANDO E CONFIGURANDO O TinyMCE

1. O tinyMCE é um editor de texto opensource que será configurado na página de artigos dando ‘super-poderes’ para o textarea. Para o usuário configurar todo o artigo antes de salvar.
2. Entre na página [www.tiny.cloud/get-tiny/self-hosted/](http://www.tiny.cloud/get-tiny/self-hosted/)
3. Clicar para baixar no botão TinyMCE Community
4. Abrir a pasta public do projeto.
5. Abrir o arquivo baixado.zip do tiny entrar na pasta ‘/js’ copiar a pasta tinymce e colar dentro da pasta public

NO ARQUIVO view/admin/articles/new.ejs

1. No final da página, importar o tinymce digitando
2. <script src="/tinymce/tinymce.min.js"></script>
3. No <textarea> incluir uma id=’article’. Essa id será fornecida ao tinymce para ele fornecer os recursos extras ao textarea.
4. Após importar, realizar a configuração do tinymce
5. <script>
6. tinymce.init({
7. selector: '#article'
8. })
9. </script>
10. Para melhorar o tinymce instalando mais plugins basta inserir na configuração do tiny conforme abaixo.
11. <script>
12. tinymce.init({
13. selector: '#article',
14. plugins: ['advlist autolink link image lists print preview hr searchreplace wordcount fullscreen insertdatetime media save table paste emoticons']
15. })
16. </script>

TRADUZINDO O TinnyMCE

1. Entrar no endereço
2. <https://www.tiny.cloud/get-tiny/language-packages/>, localizar o indioma português-brasil e clicar em donwload.
3. Abrir o arquivo zip, localizar a pasta ‘langs’ e copiar o arquivo pt\_BR.js para dentro da pasta ‘langs’ que está dentro da pasta public/tinnymce.

NO ARQUIVO view/admin/articles/new.ejs

1. Realizar alteração na configuração do tynni inclua a linguagem.
2. <script>
3. tinymce.init({
4. Language: ‘pt\_BR’,
5. selector: '#article',
6. plugins: ['advlist autolink link image lists print preview hr searchreplace wordcount fullscreen insertdatetime media save table paste emoticons']
7. })
8. </script>
9. Realizado alteração no formulário, para incluir o id da categoria dentro da tag <option> e poder utilizar quando salvar no banco de dados
10. Substitua o texto abaixo no formulário
11. <select class="form-control" id="category" name="category" onselect="">
12. <% categories.forEach(category=>{ %>
13. <option value="<%=category.id%>"><%=category.title%></option>
14. <%})%>
15. </select>
16. NO ARQUIVO articles/ArticlesControllers
17. Cria a rota para receber os dados do formulário da página News.ejs e salvar no banco de dados.
18. //criando a rota para salvar no banco de dados a nova categoria
19. router.post('/articles/save',(req,res)=>{
20. var categoryId = req.body.category;
21. var body = req.body.article;
22. var title = req.body.title;
23. var slug = slugify(title);
24. //salvando o artigo
25. Article.create({
26. title: title,
27. body: body,
28. slug: slug,
29. categoryId:categoryId
30. }).then(()=>{
31. res.redirect('/admin/articles');
32. });
33. });
34. NO ARQUIVO database/database.js
35. Ajustando a data de registro no sequelize faça a alteração da const connection conforme abaixo:
36. const connection = new Sequelize('guiapress','root','123456',{
37. host: 'localhost',
38. dialect: 'mysql',
39. timezone: '-3:00' //consultar na net
40. });
41. NO ARQUIVO articles/ArticlesController
42. Criar a rota para pagina que lista os artigos
43. //rota para abrir a página com a lista de artigos
44. router.get('/admin/articles',(req,res)=>{
45. //consultandos os registros da tabela de artigos
46. Article.findAll({
47. raw:true,
48. order:[['id','DESC']]
49. }).then(articles=>{
50. res.render("admin/articles/index",{articles:articles});
51. })
52. });

NA PASTA articles

1. Criar o arquivo index.ejs

NO ARQUIVO articles/index.ejs

1. <%- include ('../../../views/partials/header.ejs'); %>
2. <%- include ('../../../views/partials/navbar.ejs'); %>
3. <div class="container">
5. <div style=" padding: 10px;">
6. <h2 style="float: left;">Artigos</h2>
7. <a style="float: right;" class="btn btn-success" href="../../admin/articles/new" >Criar novo artigo</a>
8. </div>
10. <table class="table table-bordered">
11. <thead>
12. <tr class="head\_table">
13. <th>Id</th>
14. <th>Título</th>
15. <th>Slug</th>
16. <th>Categoria</th>
17. <th>Ações</th>
18. </tr>
19. </thead>
20. <tbody>
21. <% articles.forEach(article=>{ %>
22. <tr>
23. <td><%= article.id %></td>
24. <td><%= article.title %></td>
25. <td><%= article.slug %></td>
26. <td><%= article.category.title %></td>
27. <td style="text-align: center;">
28. <a class="btn btn-warning" href="/admin/articles/edit/<%=article.id%>">Editar</a>
30. <form  method="POST" action="/article/delete" style="display: inline;" onsubmit="confirmarDelecao(event, this)">
31. <input type="hidden" name="id" value="<%=article.id%>">
32. <button class="btn btn-danger">Excluir</button>
33. </form>
34. </td>
35. </tr>
36. <%} ) %>
37. </tbody>
38. </table>
39. </div>
40. <%- include ('../../../views/partials/footer.ejs'); %>
41. <!--criando um script para confirmar a exclusão da categoria-->
42. <script>
43. function confirmarDelecao(event, form){
44. event.preventDefault();
45. //verifica se o usuário quer deletar
46. var decision = confirm('Você quer deletar o artigo?');
47. if(decision){
48. form.submit();
49. };
50. };
51. </script>

NO ARQUIVO partials navbar.ejs

1. Alterar a nav bar para incluir o item artigos e utilizando a classe navbar-expand-lg corrigir bug na navbar
2. <nav class="navbar navbar-dark bg-primary navbar-expand-lg" >
3. <a class="navbar-brand" href="/">
4. GuiaPress
5. </a>
6. <!--Para criar um menu utilizamos a class = navbar-nav
7. mr-auto serve para configurar uma margem total à direita
8. -->
9. <ul class="navbar-nav mr-auto "  >
10. <!--Para criar um item  utilizamos a class = nav-item -->
11. <li class="nav-item"  >
12. <!--O link do item do menu deve ter a class = nav-link
13. para aparecer no menu -->
14. <a class="nav-link"  href="/admin/categories">Categorias</a>
15. </li>
16. <li class="nav-item"  >
17. <!--O link do item do menu deve ter a class = nav-link
18. para aparecer no menu -->
19. <a class="nav-link"  href="/admin/articles">Artigos</a>
20. </li>
21. </ul>
22. </nav>
23. NO ARQUIVO ArticlesController
24. Criar um join para carregar o title da category na página index.ejs de articles
25. Basta incluir um include do model Category no findAll da rota que irá abrir a página index.
26. /rota para abrir a página com a lista de artigos
27. router.get('/admin/articles',(req,res)=>{
28. //consultandos os registros da tabela de artigos
29. Article.findAll({
30. order:[['id','DESC']],
31. include:[{model:Category}]
32. }).then(articles=>{
33. res.render("admin/articles/index",{articles:articles});
34. })
35. });
36. NA PASTA partials
37. Criar o arquivo homenavbar.ejs que será a navbar da home

No arquivo partials/homenavbar.ejs

1. Editar a navbar conforme abaixo
2. <nav class="navbar navbar-expand-lg navbar-dark bg-primary">
3. <a class="navbar-brand" href="/">
4. GuiaPress
5. </a>
6. <ul class="navbar-nav mr-auto">
7. <li class="nav-item">
8. <a class="nav-link" href="/">Home</a>
9. </li>
10. </ul>
11. </nav>

NA ARQUIVO views/index.ejs

1. Criar a home page da aplicação que mostra os artigos presentes no blog
2. <%- include ('partials/header.ejs'); %>
3. <%- include ('partials/homenavbar.ejs');%>
4. <div class="container">
5. <div style="margin-top: 10px;" class="jumbotron">
6. <h1 class="display-4">Bem vindo!</h1>
7. <p class="lead">Este é um simples blog do guia do programador.</p>
8. <hr class="my-4">
9. <p>Aqui vocês encontraram artigos sobre algumas dúvidas relacionadas a programação.</p>
10. <p class="lead">
11. <a class="btn btn-primary btn-lg" href="#" role="button">Leia mais</a>
12. </p>
13. </div>
14. <% articles.forEach(article=>{ %>
15. <div style="margin-top: 10px;" class="card">
16. <div class="card-header">
17. <%=article.title%>
18. </div>
19. <div class="card-body">
20. <a class="btn btn-success" href="">Ler Artigo</a>
21. </div>
22. </div>
23. <% }); %>
24. </div>

NA PASTA views

1. Criar o arquivo da página de visualização dos artigos, article.ejs
2. <%- include('partials/header.ejs'); %>
3. <%- include('partials/navbar.ejs'); %>
4. <div class="container">
5. <div class="card">
6. <div class="card-header">
7. <h2><%= article.title%></h2>
8. </div>
9. <div class="card-body">
10. <!--Utilizase o sinal de '-' pois caso contrário
11. os formatos que foram salvos serão perdidos ao
12. carregar os dados do bd-->
13. <%- article.body%>
14. </div>
15. </div>
16. </div>
17. <%- include('partials/footer.ejs');%>

NO ARQUIVO index.js

1. Criamos a rota para acessar o artigo pelo slug enviado pela página article.ejs
2. //rota que irá abrir o artigo selecionado na pagina principal
3. //na página de leitura do artigo
4. app.get('/:slug',(req, res)=>{
5. var slug = req.params.slug;
6. //
7. Article.findOne({
8. where: {slug:slug}
9. }).then(article=>{
10. if(article != undefined){
11. res.render("article",{article:article});
12. }else{
13. res.redirect('/');
14. }
15. }).catch(err => {
16. res.redirect('/');
17. });
18. });
19. NO ARQUIVO index.js
20. Vamos adicionar as categorias na homenavbar.ejs
21. Altere as rotas incluindo uma busca das categorias pelo método findAll e depois disso envie para a página que sera renderizada, conforme alteração abaixo.
22. /criando rota para mostrar a página principal com todos os artigos
23. //cadastrados
24. app.get('/',(req,res)=>{
25. Article.findAll({
26. order:[['id','DESC']]
27. }).then(articles=>{
28. if(articles != undefined){
29. /\*localizando as categorias que serão passadas
30. para a homenavbar da página\*/
31. Category.findAll().then(categories=>{
32. res.render('index',{articles:articles, categories:categories});
33. });
34. }
35. });
36. });
37. Foi criado também uma rota para carregar uma página com todos os artigos de uma determinada categoria, filtrada pelo slug da categoria.
38. Nesta rota será enviada para a página algumas variáveis que serão utilizadas para carregar as informações solicitadas na pagina
39. /\*criando rota para carrega
40. somente os artigos de uma categoria específica\*/
41. app.get('/category/:slug',(req,res)=>{
43. var slug = req.params.slug;
44. Category.findOne({
45. where: {slug:slug},
46. //aqui é onde serão adicionados todos os artigos que pertencem
47. //a categoria procurada
48. include:[{model:Article}]
49. }).then(category =>{
50. if(category != undefined){
51. //será enviado tambem todas as categorias que serao
52. //utilizadas para comport a homenavbar
53. Category.findAll().then(categories=>{
54. if(categories != undefined){
55. //será enviado também a variavel category que
56. //passará o titulo para o texto da página
57. //articlesCategory
58. res.render('articlesCategory',{articles:category.articles, categories:categories, category:category })
59. }else{
60. res.redirect('/');
61. }
62. });
63. }else{
64. res.redirect('/');
65. }
67. });

NA PASTA views

1. Criado a página articlesCategory.ejs que irá carregar todos os artigos de uma determinada categoria.

No arquivo articlesCategory.ejs

1. Criado a página que irá receber os artigos de uma determinada categoria
2. <%- include ('partials/header.ejs'); %>
3. <%- include ('partials/homenavbar.ejs');%>
4. <div class="container">
5. <div style="margin-top: 10px;" class="jumbotron">
6. <h1 class="display-5">Categoria <%= category.title%> !</h1>
7. <hr class="my-4">
8. <p>Aqui vocês encontraram artigos sobre algumas dúvidas relacionadas a categoria <%= category.title%> .</p>
9. </div>
10. <% articles.forEach(article=>{ %>
11. <div style="margin-top: 5px;" class="card">
12. <div class="card-header">
13. <%=article.title%>
14. </div>
15. <div class="card-body">
16. <a class="btn btn-success" href="/<%= article.slug%>">Ler Artigo</a>
17. </div>
18. </div>
19. <% }); %>
20. </div>
21. <%- include ('partials/footer.ejs');%>

NO ARQUIVO ArticleController.js

1. Criado a rota para abrir a página onde será realizado a edição do artigo selecionado na página ‘articles/index.ejs’. Na rota de edição será lançado os valores do artigo e categoria que serão tratados na página ‘articles/edit.ejs’ criada.”
2. //rota para abrir a página edit.ejs de articles e
3. //carregar os valores do artigo selecionado nos campos
4. //do formulário
5. router.get('/admin/articles/edit/:id',(req, res)=>{
6. var id = req.params.id;
7. Article.findOne({
8. where: {id:id},
9. //incluimos a categoria junto com o artigo encontrado
10. //na página edit utilizaremos o title da categoria
11. include:[{model:Category}]
12. }).then(article =>{
13. if(article != undefined){
14. Category.findAll().then(categories=>{
15. res.render('../views/admin/articles/edit.ejs',{article:article, categories:categories});
16. });
17. }else{
18. res.redirect('/admin/article');
19. }
20. }).catch(err =>{
21. res.redirect('/admin/article');
22. });
23. });

NA PASTA views/admin/article/

1. Criado o arquivo edit.ejs que será a página onde será realizado a edição das informações do artigo.Foi criado uma div que receberá o conteúdo do article.body. Esse conteúdo será passado para o Tinymce, através de uma função de call-back utilizando jquery.Criado um if no select para carregar o id da categoria no value.
2. <%- include ('../../partials/header.ejs'); %>
3. <%- include ('../../partials/navbar.ejs'); %>
4. <div style="margin-top: 0px;" class="container">
5. <br>
6. <div style="margin-top: 0px;" class="card">
7. <div  style="margin-top: 0px;" class="card-header">
8. <h2>Edição de artigos</h2>
9. </div>
10. <div style="margin-bottom: 0px;" class="card-body">
11. <form method="POST" action="/articles/update">
12. <label for="tile">Título:</label>
13. <input class="form-control" type="text" name="title" value="<%= article.title%>">
14. <label for="article">Conteúdo:</label>
15. <textarea class="form-control" type="text" id="article" name="article" value="<%- article.body %>" ></textarea>
16. <label for="cars">Categoria:</label>
17. <select class="form-control" id="category" name="category" onselect="" >
18. <% categories.forEach(category=>{ %>
19. <!--<%=article.category.id%>-->
20. <% if(category.id == article.categoryId){ %>
21. <option value="<%=article.category.id%>" selected><%=article.category.title%></option>
22. <%}else{ %>
23. <option value="<%=category.id%>"><%=category.title%></option>
24. <%} %>
25. <%});%>
27. </select>
28. <br>
29. <input name="articleId" type="hidden" value="<%=article.id%>">
30. <button style="margin-top: 5px;" class="btn btn-primary">Atualizar</button>
31. </form>
32. </div>
33. </div>
34. </div>
35. <div style="display: none;" id="content">
36. <%- article.body%>
37. </div>
38. <%- include ('../../partials/footer.ejs'); %>
39. <script src="/tinymce/tinymce.min.js"></script>
40. <script>
41. tinymce.init({
42. language: 'pt\_BR',
43. selector: '#article',
44. plugins: ['advlist autolink link image lists print preview hr searchreplace wordcount fullscreen insertdatetime media save table paste emoticons'],
45. //utilizando uma instancia onde toda vez que o tiny for iniciado
46. //ele setará dentro o componente capturado "tinymce.get()"
47. //o conteúdo carregado via jquery'.setContent($('#content').html())
48. init\_instance\_callback: () => {
49. tinymce.get('article').setContent($('#content').html())
50. }
51. })
52. </script>
53. <script language="javascript" type="text/javascript">
54. function validarCampos(){
55. var title = form1.title.value;
56. var article = form1.article.value;
57. var category = form1.category.value;
58. if(title == " "){
59. alert("Preencha o título da categoria");
60. form1.title.focus();
61. return false;
62. }
63. if(article == ""){
64. alert("Preencha o artigo");
65. form1.body.focus();
66. return false;
67. }
68. }
69. </script>

NO ARQUIVO ArticlesController

1. Criamos a rota para efetuar a atualização dos dados alerados do artigo
2. //rota para salvar as alterações realizadas no artigo
3. router.post('/articles/update',(req,res)=>{
4. //variaveis da página de edição
5. var id = req.body.articleId;
6. var title = req.body.title;
7. var body = req.body.article;
8. var categoryId = req.body.category;
9. var slug = slugify(title);
10. Article.update({
11. title:title,
12. body:body,
13. categoryId:categoryId,
14. slug:slug
15. },
16. {where: {id:id}
17. }).then(()=>{
18. alert("Registro atualizado com sucesso!");
19. res.redirect('/admin/articles');
20. }).catch(err => {
21. res.redirect('/admin/articles');
22. });
23. });
24. LOGICA DE PAGINAÇÃO DO BLOG
25. NO ARQUIVO ArticlesController
26. Criar uma rota para a paginação da página dos artigos
27. /logica para a paginaca dos artigos
28. router.get('/articles/page/:num',(req, res)=>{
29. /\*logica da paginacao
30. offset =  pag \*
31. limit = 4
32. 1  =  0 - 3
33. 2  =  4 - 7
34. 3  =  8 - 11
35. \*/
36. var page = req.params.num;
37. //limit será o número de elementos que serão visualizados
38. //na página
39. var limit = 4;
40. //ofsset é o primeiro elemento que será mostrado na tela
41. var offset = 0;
42. //verificar se a página é valida se não for o offset é 0
43. if((isNaN(page)) || (page === 1)){
44. offset = 0;
45. }else{
46. offset = parseInt(page) \* limit;
47. }
48. //retornando todos os elementos e a quantidade de elementos da tabela
49. Article.findAndCountAll({
50. //paramento limit será o limite de elementos por página
51. limit: limit,
52. //parametro offset retorna dados apártir de uma posição
53. offset: offset
54. }).then(articles=>{
55. //verifica se existe outra página, ou seja
56. //se o somatorio de offset + aquantidade de itens
57. //por página form maior que a quantidade total de itens
58. //'articles.count' então não terá outra página
59. //next irá receber a resposta se tem outra página ou não
60. var next;
61. if(offset + limit >= articles.count){
62. next  = false;
63. }else{
64. next = true;
65. }
66. //resultado final da páginação
67. var results = {
68. next:next,
69. articles : articles
70. }
71. //retorna os artigos no formato JSON
72. //res.json(results);
73. });
74. });
75. NA PASTA VIEWS/ADMIN/ARTICLES
76. Criamos o arquivo page.ejs que terá os articles páginados
77. Pegamos o conteúdo da index.ejs principal e retiramos o “jumbotron”
78. <%- include ('../../partials/header.ejs'); %>
79. <%- include ('../../partials/homenavbar.ejs');%>
80. <div class="container">
81. <!--Aqui foi utilizado a propriedade rows pois no controller
82. utilizamos o método findAllCount() que retorna o número total
83. de elementos e as rows. Alem disso foi passado os articles e o
84. count para a variável results-->
85. <% results.articles.rows.forEach(article=>{ %>
86. <div style="margin-top: 10px" class="card">
87. <div class="card-header"><%=article.title%></div>
88. <div class="card-body">
89. <a class="btn btn-success" href="/<%= article.slug%>">Ler Artigo</a>
90. </div>
91. </div>
92. <% }); %>
93. </div>
94. <%- include ('../../partials/footer.ejs');%>

NO CONTROLER index.js

1. Incluímos um limit para limitar a exibição dos articles na página principal index.ejs
2. //criando rota para mostrar a página principal com todos os artigos
3. //cadastrados
4. app.get('/',(req,res)=>{
5. Article.findAll({
6. order:[['id','DESC']],
7. limit: 4
8. }).then(articles=>{
9. if(articles != undefined){
10. /\*localizando as categorias que serão passadas
11. para a homenavbar da página\*/
12. Category.findAll().then(categories=>{
13. res.render('index',{articles:articles, categories:categories});
14. });
15. }
16. });
17. });
18. NO ARQUIVO DA PÁGINA PRINCIPAL index.ejs
19. Criamos uma div no fina do articles.forEach para inserirmos o link para a próxima pagina
20. <% articles.forEach(article=>{ %>
21. <div style="margin-top: 10px;" class="card">
22. <div class="card-header">
23. <%=article.title%>
24. </div>
25. <div class="card-body">
26. <a class="btn btn-success" href="/<%= article.slug%>">Ler Artigo</a>
27. </div>
28. </div>
29. <% }); %>
30. <!—Foi incluído estas linhas abaixo da paginação-->
31. <div style="margin-top: 10px;" class="row">
32. <div class="col col-12">
33. <a href="/articles/page/1" style="float: right;">Next >>></a>
34. </div>
35. <hr>
36. </div>
37. <hr>
38. </div>
39. <%- include ('partials/footer.ejs');%>

NO CONTROLER ArticlesController

1. Vamos ordenar as artigos pelo id realizando a alteração abaixo
3. Article.findAndCountAll({
4. //paramento limit será o limite de elementos por página
5. limit: limit,
6. //parametro offset retorna dados apártir de uma posição
7. offset: offset,
8. //realizando a ordenacao dos artigos pelo id
9. order: [
10. ['id','DESC']]
11. Incluímos também mais uma variável no array result que o número da página atual que esta sendo passado via parâmetro na url.
12. //resultado final da páginação
13. var results = {
14. //mostra qual a página atual
15. page: parseInt(page),
16. //mostra se terá outra página para exibir
17. next: next,
18. //carrega todos os artigos da página que será exibida
19. articles : articles
20. }

NO ARQUIVO page.ejs

1. Incluímos as regras de paginação, copiando da página inicial e colando no final da página antes do footer, criado a logica para mostrar as tags previus e next
2. <!--paginação-->
3. <div style="margin-top: 10px;" class="row">
4. <div class="col col-12">
5. <!--calculando link para a próxima página utilizando
6. a varivel result passada pelo controller, será verificado
7. tambem se existe uma página anterior-->
8. <% if(results.page >2 ){ %>
9. <a href="/articles/page/<%=results.page - 1%>" style="float: left;"><<< Previous</a>
10. <%}else if(results.page == 2){%>
11. <a href="/" style="float: left;"><<< Previous</a>
12. <%} %>
13. <% if(results.next){ %>
14. <a href="/articles/page/<%=results.page + 1%>" style="float: right;">Next >>></a>
15. <%} %>
17. </div>
18. <hr>
19. </div>
20. <hr>
21. </div>
22. <%- include ('../../partials/footer.ejs');%>

BASE DO LOGIN

1. Suporte de login para o administrador
2. Criado a pasta User

NA PASTA user

1. Criado o model User.js e controller UserController

NO ARQUIVO User.sj

1. Copiado o conteúdo do model Article.js
2. Realizado as alterações
3. //importanto o sequelize
4. const Sequelize = require('sequelize');
5. //importando a conexão do banco de dados
6. const connection = require('../database/database');
7. //definindo o model da tabela artigos
8. const User = connection.define('users',{
9. mail:{
10. type: Sequelize.STRING,
11. allowNull: false
12. },
13. password:{
14. type: Sequelize.STRING,
15. allowNull: false
16. }
17. });
18. //sincronizando ou atualizando ou criando a tabela no banco de dados deve-se comentar este código assim
19. //que a tabela for criada no banco de dados
20. User.sync({force:true});
21. //exportando o model Article
22. module.exports = User;

NO ARQUIVO UserController

1. Copie as configurações básicas e crie duas rotas
2. //importando o express
3. const express = require('express');
4. //utilizando o router para carregar as rotas
5. const router = express.Router();
6. /\*\*Banco de dados \*/
7. //importando o model de user
8. const User = require('../users/User');
9. //rota para abrir a página com a lista de usuário
10. router.get('/admin/users',(req,res)=>{
11. //consultandos os registros da tabela de usuários
12. res.render('../views/admin/users/index');
13. });
14. //rota para criar os usuário
15. router.get('/admin/users/create',(req,res)=>{
16. res.render('../views/admin/users/create');
17. });
18. //exportando o router
19. module.exports = router;

NO CONTROLER PRINCIPAL index.js

1. Importe as rotas do controller UsersController
2. const usersController = require('./users/UsersController');
3. Importe o model de usuários
4. const User = require('./users/User');
5. informe ao aplicativo que deverá utilizar o controlador que foi importado
6. app.use('/',usersController);

NA PASTA views/admin/users

1. Crie o arquivo create.eja onde será construído o formulário para cadastro dos usuários
2. <%- include('../../partials/header.ejs');%> <%-
3. include('../../partials/navbar.ejs');%>
4. <div style="width: 600px; padding: 5px" class="container">
5. <div class="card mt-4">
6. <div style="text-align: center" class="card-header">
7. <h2>Cadastro de usuários</h2>
8. </div>
9. <div class="card-body">
10. <form name="form1" method="POST" action="/users/save">
11. <input
12. name="email"
13. type="email"
14. class="form-control"
15. placeholder="email@exemplo.com"
16. />
17. <input
18. name="password"
19. type="password"
20. class="form-control mt-3"
21. placeholder="\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"
22. />
23. <button class="form-control btn btn-success mt-3">Cadastrar</button>
24. </form>
25. </div>
26. </div>
27. </div>
28. <%- include('../../partials/footer.ejs');%>
29. <script language="javascript" type="text/javascript">
30. function validarCampos() {
31. var email = form1.email.value;
32. var password = form1.password.value;
33. if (email == " ") {
34. alert("Digite o email do usuario");
35. form1.email.focus();
36. return false;
37. }
38. if (password == " ") {
39. alert("Digite a senha do usuario");
40. form1.password.focus();
41. return false;
42. }
43. }
44. </script>
45. Crie também o arquivo index.ejs que será a página principal do usuário com a lista de todos os usuário e a opção de editar ou excluir usuário
46. <%- include('../../partials/header.ejs');%>
47. <%- include('../../partials/navbar.ejs');%>
48. <div style="margin-top: 10px; padding: 5px;" class="container">
49. <div class="card">
50. <div class="card-header">
51. <h2 style="float: left;">Lista de usuários</h2>
52. <a style="float: right;" class="btn btn-success" href="../../admin/users/create">Cadastrar Usuário</a>
53. </div>
54. <div class="card-body">
55. <table class="table table-bordered">
56. <thead>
57. <th>Id</th>
58. <th>Email</th>
59. <!--  <th style="text-align: center;">Ações</th> -->
60. </thead>
61. <tbody>
62. <% if(users != undefined){ %>
63. <% users.forEach(user =>{ %>
64. <tr>
65. <td><%= user.id %></td>
66. <td><%= user.mail %></td>
67. <!--  <td style="text-align: center;">
68. <a href="/admin/users/edit/<%= user.id%>" class="btn btn-warning" >Editar</a>
69. <form method="POST" action="/user/delete" style="display: inline;" onsubmit="confirmarExclusao(event, this)">
70. <button class="btn btn-danger" >Excluir</button>
71. <input type="hidden" value="<%= user.id%>" name="userId">
72. </form>
73. </td> -->
74. </tr>
75. <% }); %>
76. <%};%>
77. </tbody>
78. </table>
79. </div>
80. </div>
81. </div>
82. <%- include('../../partials/footer.ejs');%>
83. <!--criando um script para confirmar a exclusão da categoria-->
84. <script>
85. function confirmarExclusao(event, form){
86. event.preventDefault();
87. //verifica se o usuário quer deletar
88. var decision = confirm('Você quer deletar a categoria?');
89. if(decision){
90. form.submit();
91. };
92. };
93. </script>
94. NO ARQUIVO UsersController
95. Altere a rota para abrir a index.ejs de users
96. //rota para abrir a página com a lista de usuário
97. router.get('/admin/users',(req,res)=>{
98. //consultandos os registros da tabela de usuários
99. User.findAll({
100. raw:true,
101. order:[['id','DESC']]
102. }).then(users=>{
103. if(users != undefined){
104. res.render('../views/admin/users/index',{users:users});
105. }else{
106. res.redirect('../views/admin/users/create');
107. }
108. });
110. });
111. Altere também a rota para salvar o usuário no banco de dados
112. //rota para salvar no banco de dados o usuário
113. router.post('/users/save',(req, res)=>{
114. var mail = req.body.email;
115. var password = req.body.password;
116. User.create({
117. mail: mail,
118. password: password
119. }).then(()=>{
120. res.redirect('/admin/users');
121. }).catch(err =>{
122. res.send("Erro desconhecido. "+ err);
123. });
124. });
125. Crie a rota para abrir o formulário de cadastro de usuário
126. //rota para carregar a página com o formulário de cadastro dos usuário
127. router.get('/admin/users/create',(req,res)=>{
128. res.render('../views/admin/users/create');
129. });
130. NO raiz do projeto
131. Realizar a instalação do bcrypt para transformar a senha do usuário em um hash
132. Comandos: ‘npm install - - save bcryptjs;
133. NO UsersController
134. Importar o bcrypt:
135. Const bcrypt = require(‘bcryptjs’);
136. Criar um “sal” para melhorar o bcrypt
137. Var salt = bcrypt.genSaltSync(10); OBS: o número é fornecido pelo programador.
138. Criar o hash da senha e passamos a senha como parâmetro e o salt.
139. Var hash = bcrypt.hashSync(password, salt);
140. Vamos salvar no banco de dados o hash.
141. Altera a rota conforme abaixo
142. //rota para salvar no banco de dados o usuário
143. router.post('/users/save',(req, res)=>{
144. var mail = req.body.email;
145. var password = req.body.password;
146. var salt = bcrypt.genSaltSync(10);
147. var hash = bcrypt.hashSync(password, salt);
148. //utilizado para teste antes de popular no banco de dados
149. //res.json({mail, hash});
150. //verifica se o ja existe o email cadastrado
151. User.findOne({where: {mail: mail}}).then(user=>{
152. if(user == undefined){
153. User.create({
154. mail: mail,
155. password: hash
156. }).then(()=>{
157. res.redirect('/');
158. }).catch(err =>{
159. res.send("Erro desconhecido. "+ err);
160. });
161. }else{
162. res.redirect('/admin/users/create');
163. }
164. });
165. Cria a rota para apagar usuário
166. //rota para excluir um usuário
167. router.post('/user/delete',(req, res)=>{
168. var userId = req.body.userId;
169. User.destroy({
170. where:{ id: userId }
171. }).then(()=>{
172. res.redirect('/admin/users');
173. }).catch(err=>{
174. res.send("Erro desconhecido. "+ err);
175. });
176. });

CRIANDO SESSÕES

1. Instalamos a biblioteca de sessões do express
2. Npm install express-session –save

NO ARQUIVO index.js

1. Importar a biblioteca session
2. Const session = require(‘express-session’);
3. //configurando o gerenciamento de sessões
4. app.use(session({
5. //texto utilizado para aumetar a segurança das sessões
6. secret:"qualquerPalavra",
7. //cookie é uma referencia para sessão no servidor
8. //pode-se setar o tempo de uma sessão pela
9. //propriedade maxAge que é em milisegundos ou seja cada segundo e 1000 milisegundo
10. cookie: {maxAge: 30000}
11. }));
12. Criando rotas para manipular sessões
13. //exemplo rota para criar a sessão
14. app.get('/session',(req,res)=>{
15. //criando informações da sessão
16. req.session.treinamento = "Formação Node.js"
17. req.session.ano = 2019
18. req.session.email = 'claudisnei@teste'
19. req.session.user = {
20. username: 'claudisnei',
21. email: 'email@qualquer',
22. id:'10'
23. }
24. //resposta da rota
25. res.send('Sessão gerada');
26. });
27. //exemplo de rota para ler a sessão
28. app.get('/leitura',(req,res)=>{
29. res.json({
30. treinamento: req.session.treinamento,
31. ano: req.session.ano,
32. email: req.session.email,
33. user: req.session.user
34. })
35. });

Autenticação

1. Criando o formulário de login
2. Na pasta views/admin/users
3. Criar o arquivo login.ejs

No arquivo login.ejs

1. <%- include('../../partials/header.ejs');%> <%-
2. include('../../partials/navbar.ejs');%>
3. <div style="width: 600px; padding: 5px" class="container">
4. <div class="card mt-4">
5. <div style="text-align: center" class="card-header">
6. <h2>Login de admin</h2>
7. </div>
8. <div class="card-body">
9. <form name="form1" method="POST" action="/authenticate" onsubmit="validarCampos()">
10. <input
11. name="email"
12. type="email"
13. class="form-control"
14. placeholder="email@exemplo.com"
15. />
16. <input
17. name="password"
18. type="password"
19. class="form-control mt-3"
20. placeholder="\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"
21. />
22. <button type="submit" class="btn btn-block btn-success mt-3">Entrar</button>
23. </form>
24. </div>
25. </div>
26. </div>
27. <%- include('../../partials/footer.ejs');%>
28. <script language="javascript" type="text/javascript">
29. function validarCampos() {
30. var email = form1.email.value;
31. var password = form1.password.value;
32. if (email == "") {
33. alert("Digite o email do usuario");
34. form1.email.focus();
35. return false;
36. }
37. if (password == "") {
38. alert("Digite a senha do usuario");
39. form1.password.focus();
40. return false;
41. }
42. }
43. </script>
44. No arquivo UsersController.js
45. Criando a rota para carregar o formulário do login
46. //rota para carregar o formulário de login do usuário
47. router.get('/users/login',(req,res)=>{
48. res.render('../views/admin/users/login');
49. });
50. Criando rota para validar se o usuário é cadastrado
51. //rota para fazer o login
52. router.post('/authenticate',(req, res)=>{
53. //recebendo os dados do formulário
54. var email = req.body.email;
55. var password = req.body.password;
56. //pesquisar se existe o usuário no banco de dados
57. User.findOne({where:{mail: email}}).then(user=>{
58. if(user!= undefined){
59. //validar a senha o decript compara se a senha
60. //informada pelo usuario é igual a salva no banco de dados
61. var correct = bcrypt.compareSync(password,user.password);
63. //verifica se a senha é a correta
64. if(correct){
65. //criando uma sessão user
66. req.session.user = {
67. id: user.id,
68. email: user.mail
69. }
70. //testando
71. res.json(req.session.user);
72. }else{
73. res.redirect('/admin/users/login')
74. }
75. }else{
76. res.redirect('/admin/users/login')
77. }
78. }).catch(err =>{
79. res.send("Erro "+ err);
80. })
81. });
82. --------------------------------------------
83. Configurando a permissão de acesso através de um Midleware.
84. Criando um midleware que é uma função
85. Criar uma pasta chamada middleware
86. Na pasta middleware
87. Criar o arquivo adminauth.js
88. No arquivo adminauth.js
89. function adminAuth(req, res, next){
90. //o parâmetro next serve para dar continuidade na requisição
91. //se a sessão user existe, significa que o usuário está logado
92. //então vai para next() ou seja vai para o próximo caminho
93. if(req.session.user != undefined){
94. next();
95. }else{
96. //se a sessão não existir será dado um redirect para home page
97. res.redirect('/');
98. }
99. }
100. module.exports = adminAuth;
101. No controlador dos artigos articlesController;
102. Inserir ‘const adminAuth = require(‘adminAuth’);
103. Agora basta inserir este middleware em toda rota que precisar de autenticação