

Tabela Categoria

```
INSERT INTO `categoria` (`id_categoria`, `titulo`, `slug`) VALUES
(1, 'Programação Web', 'programacao-web'),
(2, 'Programação Desktop', 'programacao-desktop'),
(3, 'Esportes Basquete', 'esportes-basquete');
```

Tabela Artigo

```
INSERT INTO `artigo` (`id_artigo`, `titulo`, `body`, `slug`, `categoria_id`) VALUES
(1, 'PHP ou Ruby?', '<h2>O que &eacute; Lorem Ipsum?</h2>\r\n<p style=\\"text-align: justify;\\"><strong>Lorem Ipsum</strong>&nbsp;&eacute; simplesmente uma simula&ccedil;&atilde;o de texto da ind&uacute;stria tipogr&aacute;fica e de impressos, e vem sendo utilizado desde o s&eacute;culo XVI, quando um impressor desconhecido pegou uma bandeja de tipos e os embaralhou para fazer um livro de modelos de tipos. Lorem Ipsum sobreviveu n&atilde;o s&oacute;a cinco s&eacute;culos, como tamb&eacute;m ao salto para a editora&ccedil;&atilde;o eletr&ocirc;nica, permanecendo essencialmente inalterado. Se popularizou na d&eacute;cada de 60, quando a Letraset lan&ccedil;ou decalques contendo passagens de Lorem Ipsum, e mais recentemente quando passou a ser integrado a softwares de editora&ccedil;&atilde;o eletr&ocirc;nica como Aldus PageMaker.</p>', 'php-ou-ruby', 1),
(2, 'Usando Express no NodeJs', '<h2>O que &eacute; Lorem Ipsum?</h2>\r\n<p style=\\"text-align: justify;\\"><strong>Lorem Ipsum</strong>&nbsp;&eacute; simplesmente uma simula&ccedil;&atilde;o de texto da ind&uacute;stria tipogr&aacute;fica e de impressos, e vem sendo utilizado desde o s&eacute;culo XVI, quando um impressor desconhecido pegou uma bandeja de tipos e os embaralhou para fazer um livro de modelos de tipos. Lorem Ipsum sobreviveu n&atilde;o s&oacute;a cinco s&eacute;culos, como tamb&eacute;m ao salto para a editora&ccedil;&atilde;o eletr&ocirc;nica, permanecendo essencialmente inalterado. Se popularizou na d&eacute;cada de 60, quando a Letraset lan&ccedil;ou decalques contendo passagens de Lorem Ipsum, e mais recentemente quando passou a ser integrado a softwares de editora&ccedil;&atilde;o eletr&ocirc;nica como Aldus PageMaker.</p>', 'usando-expressno-nodejs', 1),
(3, 'Quem usa ainda PowerBuilder?', '<h2 style=\\"text-align: justify;\\">De onde ele vem?</h2>\r\n<p style=\\"text-align: justify;\\">Ao contr&aacute;rio do que se acredita, Lorem Ipsum n&atilde;o &eacute; simplesmente um texto rand&ocirc;mico. Com mais de 2000 anos, suas ra&iacute;zes podem ser encontradas em uma obra de literatura latina cl&aacute;ssica datada de 45 AC. Richard McClintock, um professor de latim do Hampden-Sydney College na Virginia, pesquisou uma das mais obscuras palavras em latim, consetetur, oriunda de uma passagem de Lorem Ipsum, e, procurando por entre cita&ccedil;&otilde;es da palavra na literatura cl&aacute;ssica, descobriu a sua indubit&aacute;vel origem. Lorem Ipsum vem das se&ccedil;&otilde;es 1.10.32 e 1.10.33 do \\"de Finibus Bonorum et Malorum\\" (Os Extremos do Bem e do Mal), de C&iacute;ero, escrito em 45 AC. Este livro &eacute; um tratado de teoria da &eacute;tica muito popular na &eacute;poca da Renascen&ccedil;a. A primeira linha de Lorem Ipsum, \\"Lorem Ipsum dolor sit amet...\" vem de uma linha na se&ccedil;&atilde;o 1.10.32.</p>\r\n<p style=\\"text-align: justify;\\">O trecho padr&atilde;o original de Lorem Ipsum, usado desde o s&eacute;culo XVI, est&aacute; reproduzido abaixo para os interessados. Se&ccedil;&otilde;es 1.10.32 e 1.10.33 de \\"de Finibus Bonorum et Malorum\\" de Cicero tamb&eacute;m foram reproduzidas abaixo em sua forma exata original, acompanhada das vers&otilde;es para o ingl&ecirc;s da tradu&ccedil;&atilde;o feita por H. Rackham em 1914.</p>', 'quem-usa-ainda-powerbuilder', 2),
(4, 'Programando Usando Delphi', '<h2>De onde ele vem?</h2>\r\n<p>Ao contr&aacute;rio do que se acredita, Lorem Ipsum n&atilde;o &eacute; simplesmente um texto rand&ocirc;mico. Com mais de 2000 anos, suas ra&iacute;zes podem ser encontradas em uma obra de literatura latina cl&aacute;ssica datada de 45 AC. Richard McClintock, um professor de latim do Hampden-Sydney College na Virginia, pesquisou uma das mais obscuras palavras em latim, consetetur, oriunda de uma passagem de Lorem Ipsum, e, procurando por entre cita&ccedil;&otilde;es da palavra na literatura cl&aacute;ssica, descobriu a sua indubit&aacute;vel origem. Lorem Ipsum vem das se&ccedil;&otilde;es 1.10.32 e 1.10.33 do \\"de Finibus Bonorum et Malorum\\" (Os Extremos do Bem e do Mal), de C&iacute;ero, escrito em 45 AC. Este livro &eacute; um tratado de teoria da &eacute;tica muito popular na &eacute;poca da Renascen&ccedil;a. A primeira linha de Lorem Ipsum, \\"Lorem Ipsum dolor sit amet...\" vem de uma linha na se&ccedil;&atilde;o 1.10.32.</p>\r\n<p style=\\"text-align: justify;\\">O trecho padr&atilde;o original de Lorem Ipsum, usado desde o s&eacute;culo XVI,
```

est&acut; reproduzido abaixo para os interessados. Seções 1.10.32 e 1.10.33 de \"de Finibus Bonorum et Malorum\" de Cicero também foram reproduzidas abaixo em sua forma exata original, acompanhada das versões para o inglês da tradução feita por H. Rackham em 1914.</p><p>', 'programando-usando-delphi', 2), (5, 'Maior Campe&acut;o da NBA', '<h2>Porque nôs o usamos?</h2>\r\n<p style=\"text-align: justify;\">Á um fato conhecido de todos que um leitor se distrair&acut; com o conteúdo de texto legível de uma p&acut;gina quando estiver examinando sua diagramação. A vantagem de usar Lorem Ipsum &acut; que ele tem uma distribuição normal de letras, ao contr&acut;rio de \"Conteúdo aqui, conteúdo aqui\", fazendo com que ele tenha uma aparência similar a de um texto legível. Muitos softwares de publicação e editores de p&acut;ginas na internet agora usam Lorem Ipsum como texto-modelo padrão, e uma r&acut;pida busca por 'lorem ipsum' mostra v&acut;rios websites ainda em sua fase de construção. V&acut;rias versões novas surgiram ao longo dos anos, eventualmente por acidente, e às vezes de prop&ocute;sito (injetando humor, e coisas do gênero).</p>', 'maior-campeao-da-nba', 3);

Um erro deve ocorrer pois a como j&acut;a estú mandando os arquivos j&acut;a codificados n&acut;o est&acut;a encontrando o \"artigo\" para a apresentação vamos arrumar

```
conexao.authenticate().then(()=>{
  console.log("CONECTADO COM O BANCO");
}).catch((errorMsg)=>{
  console.log(errorMsg);
})

app.get("/", (req, res) => {
  Artigo.findAll().then((artigo) => {
    res.render("primeiro", {artigo});
  })
})
```

Sô para estudo vamos olhar o arquivo de apresentação dos artigos que nosso arquivo \"primeiro.ejs\"

```
1 <%- include ("partials/header.ejs") %>
2 <%- include ("partials/navbar.ejs") %>
3 <body>
4   <div class="container">
5     <br><br>
6     <% artigo.forEach((artigo) =>{ %>
7
8       <div class="card">
9         <div class="card-header">
10          <h2><%= artigo.titulo %></h2>
11        </div>
12        <div class="card-body">
13          <a href="/ler/<%= artigo.slug %>" class="btn btn-success">Ler artigo</a>
14        </div>
15      </div>
16      <br>
17    <% }) %>
18  </div>
19 </div>
20 </body>
21 <%- include ("partials/footer.ejs") %>
```

Abriu o arquivo \"alte_artigo.ejs\" dentro da pasta \"artigos\". ESTE ARQUIVO J&acut; MANDEI NO ARQUIVO ZIPADO

```

1 <%- include("../partials/header.ejs") %>
2 <%- include("../partials/navbar.ejs") %>
3 <div class="container">
4 <hr>
5 <div class="card">
6 <div class="card-header">
7 <h2>Alterar Artigo</h2>
8 </div>
9 <div class="card-body">
10 <form method="POST" action="/updateartigo">
11
12 <input class="form-control" type="text" name="titulo" placeholder="Defina o título do artigo" value="<%= artigo.titulo %>"><br>
13
14 <textarea class="form-control" placeholder="Escreva o artigo aqui!" name="body" id="artigo"><%= artigo.body %></textarea><br>
15
16 <input type="hidden" name="id" value="<%= artigo.id_artigo %>">
17
18 <label>Categoria</label>
19
20 <select name="categoria" class="form-control">
21 <option value="">Selecione Categoria</option>
22
23 <%= categoria.forEach((categ)=>{ %>
24 <%= if(categ.id_categoria == artigo.categoria_id){ %>
25 <%= <option value="<%= categ.id_categoria %>" selected><%= categ.titulo %></option>
26 <%= }else { %>
27 <%= <option value="<%= categ.id_categoria %>"><%= categ.titulo %></option>
28 <%= } %>
29 <%= } %>
30
31 </select>
32 <br>
33 <br>
34 <button class="btn btn-success">Alterar</button>
35 </form>
36 </div>
37 </div>
38 </div>
39 <%- include("../partials/footer.ejs") %>
40 <script src="/tinymce/tinymce.min.js"></script>
41 <script>
42 tinymce.init({
43 language: 'pt_BR',
44 selector: "#artigo",
45 plugins: [
46 'advlist autolink link image lists print preview hr searchreplace'
47 + 'wordcount fullscreen insertdatetime media save table paste emoticons'
48 ]
49 })
50 </script>

```

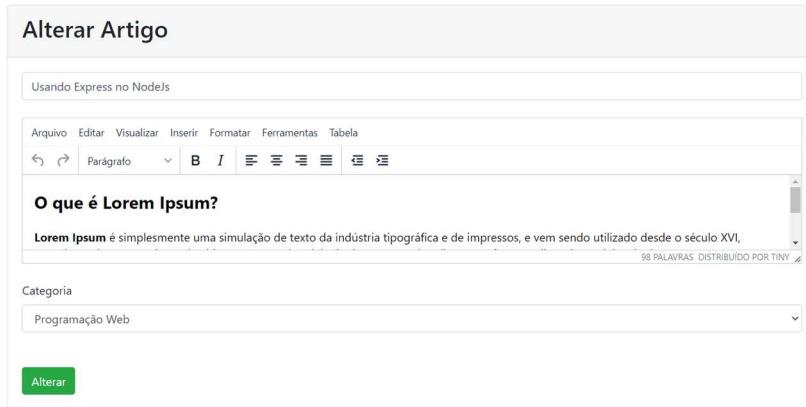
Vamos fazer alteração no arquivo "**controleArtigo.js**", lembre-se que quando o usuário clicar em "Editar" ele vai chamar a rota **"/editartigo/:id"**. Adicione as linhas abaixo neste arquivo.

```

16 router.get("/controleartigo", (req, res) => {
17   Artigo.findAll({
18     include: [{model: Categoria, as: 'categoria'}]
19   }).then((artigo) => {
20     res.render("artigos/controle_artigos", {artigo});
21   })
22 })
23
24 router.get("/editartigo/:id", (req, res) => {
25   let id = req.params.id;
26   Artigo.findByPk(id).then((artigo) => {
27     Categoria.findAll().then((categoria) => {
28       res.render("artigos/alte_artigo", {artigo, categoria});
29     });
30   });
31 });

```

Se tudo der certo deve aparecer todos os dados agora vamos configurar a nossa rota de alteração,



Vamos adicionar o código abaixo no nosso arquivo **"controleArtigo.js"**

```
24 router.post("/updateartigo", (req, res) => {
25   let id = req.body.id;
26   let titulo = req.body.titulo;
27   let body = req.body.body;
28   let categoria = req.body.categoria;
29   Artigo.update({
30     titulo: titulo,
31     body: body,
32     categoria: categoria,
33     slug: slugify(titulo),
34   },
35   {
36     where: {
37       id_artigo: id
38     }
39   }).then(() => {
40     res.redirect("/controleartigo")
41   })
42 })
```

Vamos aos testes, para ver se as alterações são feitas.

A parte de apresentação de artigo já está pronta como podemos notar

Mas vamos fazer a parte da apresentação, como pode notar o arquivo "lerarquivo.ejs", Vamos dar olhada nele

```
<%- include ("partials/header.ejs") %>
<%- include ("partials/navbar.ejs") %>
<body>
  <div class="container">
    <hr />
    <div class="card">
      <div class="card-header">
        <h2><%= artigo.titulo %></h2>
      </div>
      <div class="card-body">
        <h2><%= artigo.body %></h2>
      </div>
    </div>
  </div>
</body>
<%- include ("partials/footer.ejs") %>
```

```
28 app.get("/", (req, res) => {
29   Artigo.findAll().then((artigo) => {
30     res.render("primeiro", {artigo});
31   })
32 })
33
34 app.get("/ler/:slug", (req, res) => {
35   var slug = req.params.slug;
36   Artigo.findOne({
37     where: {
38       slug: slug,
39     },
40   }).then((artigo) => {
41     res.render("lerarquivo", { artigo });
42   });
43 });
```

PHP ou Ruby ?

Ler artigo

Usando Express no NodeJs

Ler artigo

PHP ou Ruby ?

O que é Lorem Ipsum PowerBuilder?

Lorem Ipsum é simplesmente uma simulação de texto da indústria tipográfica e de impressos, e vem sendo utilizado desde o século XVI, quando um impressor desconhecido pegou uma bandeja de tipos e os embaralhou para fazer um livro de modelos de tipos. Lorem Ipsum sobreviveu não só a cinco séculos, como também ao salto para a editoração eletrônica, permanecendo essencialmente inalterado. Se popularizou na década de 60, quando a Letraset lançou decalques contendo passagens de Lorem Ipsum, e mais recentemente quando passou a ser integrado a softwares de editoração eletrônica como Aldus PageMaker.

VAMOS TRABALHAR COM USUÁRIOS

Como pode observar já temos uma pasta **"usuarios"** que está dentro dentro da pasta **"views"**, dentro desta pasta já temos codificado dois arquivos que vamos utilizar, **"cadastrouusuario.ejs"** e **"loginusuario,ejs"**.

Vamos criar nossa tabela **"usuario"** e chamar o nosso arquivo o arquivo **"cadastrouusuario.ejs"**

Crie a pasta **"cont_usuario_usuario"** dentro dela vamos criar o arquivo **"Usuario.js"**, veja o código abaixo.

Usuario.js

```
1 const Sequelize = require("sequelize");
2 const conexao = require('../database/basedados');
3 const Usuario = conexao.define('usuario', {
4   id_usuario: {
5     type: Sequelize.INTEGER,
6     autoIncrement: true,
7     allowNull: false,
8     primaryKey: true
9   },
10  login: {
11    type: Sequelize.STRING,
12    allowNull: false
13  },
14  senha: {
15    type: Sequelize.STRING,
16    allowNull: false
17  }
18 });
19
20 Usuario.sync();
21 module.exports = Usuario
```

Vamos chamar nosso arquivo no "index.js"

```
1 const express = require('express');
2 const app = express();
3 const conexao = require('../database/basedados');
4 const Categoria = require('../cont_categoria/Categoria');
5 const Artigo = require('../cont_artigo/Artigo');
6
7 const Usuario = require('../cont_usuario/Usuario')
8
9 const ControleCategoria = require('../cont_categoria/controleCategoria');
10 const ControleArtigo = require('../cont_artigo/controleArtigo');
11
```

Vamos criar o arquivo "controleUsuario.js", veja o código abaixo.

```
1 const express = require('express');
2 const router = express.Router();
3 const bodyParser = require('body-parser');
4 const Usuario = require('../Usuario');
5
6 router.use(bodyParser.urlencoded({ extended: true }));
7
8 router.get("/novousuario", (req, res) => {
9   res.render("usuarios/cadastrausuario")
10 })
11
12 module.exports = router;
```

Agora vamos fazer as mudanças necessárias (adicionar) no arquivo "index.js".

```
8 const ControleCategoria = require('./cont_categoria/controleCategoria');
9 const ControleArtigo = require('./cont_artigo/controleArtigo');
10
11 const ControleUsuario = require('./cont_usuario/controleUsuario')
12
13 app.use("/", ControleUsuario)
14
15 app.use("/", ControleCategoria)
16 app.use("/", ControleArtigo)
```

Vamos testar se esta renderizando a nossa página de cadastro de usuário.



O formulário tem um cabeçalho com o título "Criação de Usuário". Abaixo dele, há um campo de texto rotulado "Login Usuário - Email". Logo abaixo, há um campo de texto rotulado "Senha". Na base do formulário, há um botão verde com o texto "Criar".

Como pode ser observado no arquivo "cadastrosusuario.ejs"
Mas antes de criarmos a nossa rota "/user/addsenha". Vamos ver um assunto "hash"

Hash: o que são?

Uma função de hash criptográfico, muitas vezes é conhecida simplesmente como hash , é um algoritmo matemático que transforma qualquer bloco de dados em uma série de caracteres de comprimento

Vamos baixar o **bcryptjs**, para criar nosso **hash**.

npm install bcryptjs --save

Vamos mexer no arquivo "controleUsuario.js".

Muita atenção neste código vamos fazê-lo e prestar atenção nos detalhes que ele tem. Vamos criar a nossa rota "/user/addsenha"

```

6  const bcrypt = require('bcryptjs')
7
8  router.use(bodyParser.urlencoded({ extended: true }));
9
10 router.get("/novousuario", (req, res) => {
11   res.render("usuarios/cadastrausuario")
12 })
13
14 router.post("/user/addsenha", (req, res) => {
15   let login = req.body.login;
16   Usuario.findOne({
17     where: {login: login}
18   }).then((usuario) => {
19     if (usuario === undefined) {
20       let senha = req.body.senha;
21       let criahash = bcrypt.genSaltSync(10);
22       let hash = bcrypt.hashSync(senha, criahash);
23       Usuario.create({
24         login: login,
25         senha: hash,
26       }).then(() => {
27         res.render("usuarios/cadastrausuario")
28       })
29     } else {
30       res.redirect("/")
31     }
32   })
33 })
34 })

```

Cadastre um **login (email)** e senha para teste eu usei **admin@teste.com.br** senha: **123**

Vamos fazer o login de usuário, dentro do arquivo **"navbar.ejs"** vamos adicionar como na imagem abaixo a parte de login.

```

14 <<li class= "nav-item mr-2">
15 <<a class= "nav-link" href="/novoartigo">Artigos</a>
16 <</li>
17 <<li class= "nav-item mr-2">
18 <<a class= "nav-link" href="/controleartigo">Controle Artigos</a>
19 <</li>
20 <<li class= "nav-item mr-2">
21 <<a class= "nav-link" href="/novousuario">Cadastro Usuário</a>
22 <</li>
23
24 <<li class= "nav-item mr-2">
25 <<a class= "nav-link" href="/loginusuario">Login Usuário</a>
26 <</li>
27 |
28 <</ul>
29 <</nav>

```

Vamos alterar nosso arquivo **"controleUsuario.js"**, vamos configurar a rota **"/loginusuario"**.

```

11 router.get("/novousuario", (req, res) => {
12   res.render("usuarios/cadastrausuario")
13 })
14
15
16 router.get("/loginusuario", (req, res) => {
17   res.render("usuarios/loginusuario")
18 })
19

```

Você deve ter notado que quando, foi renderizado a página, o menu superior devr estar mudado, e que vamo trabalhar com sessões a seguir.

Vamos antes de validar o usuário, vamos trabalhar com sessões vamos instalar o responsável pelo controle das sessões

Vamos instalar o pacote responsável pelo controle das sessões.

npm install express-session --save

Vamos fazer um teste, para ver se nossas sessões estão sendo criadas e se podem ser usadas.

Arquivo "index.js"

```
11 const session = require('express-session');
12
13 app.use(session({
14   secret:"qualquercoisa",
15   resave:false,
16   saveUninitialized:false,
17   //cookie:{maxAge: 20000}
18 })))
19
20 app.get("/teste/sessao",(req,res) =>{
21   req.session.txt = "teste",
22   req.session.usuario={
23     nome:"Joao da Silva",
24     senha:123
25   }
26   res.send("gerado")
27 })
28
29 app.get("/teste/leitura",(req,res)=>{
30   res.json({
31     texto : req.session.txt,
32     usuario : req.session.usuario
33   })
34 })
35
```

Antes de fazermos o login direto vamos ver algumas mudanças que vamos fazer em nosso código, pode notar que já temos um arquivo codificado chamado "navbar01.js" e também um arquivo chamado "primeiro1.ejs".

Caso queira comentar ou excluir o código acima pode fazer, mas deixe comentado para termos de exemplo nas próximas configurações.

loginusuario.ejs

```
<%- include ("../partials/header.ejs") %>
<%- include ("../partials/navbar1.ejs") %>
<style>
  div {
    max-width: 300px;
  }
  .text-input{
    width:100px
  }
</style>
<div class="container" align="center">
  <hr />
  <div class="card mt-5" style="max-width: 100">
    <div class="card-header">
      <h2>Login de Usuário</h2>
    </div>
    <div class="card-body">
      <form action="/user/loginuser" method="POST">
        <label>Login Usuário - Email</label>
        <input class="form-control" name="login" type="email" placeholder="" required /><br />
        <label>Senha</label>
        <input class="form-control" name="senha" type="senha" required/><br />
        <button type="submit" class="btn btn-success btn-block" size="5">Logar</button>
      </form>
    </div>
  </div>
</div>
<%- include ("../partials/footer.ejs") %>
```

Vamos alterar o nosso arquivo "controleUsuario.js"

```

21 router.post("/user/loginuser", (req, res) => {
22   ...let login = req.body.login;
23   ...let senha = req.body.senha;
24   ...Usuario.findOne({
25     ...where: {
26       ...login: login
27     }
28   }).then((usuario) => {
29     ...if (usuario != undefined) {
30       ...var correta = bcrypt.compareSync(senha, usuario.senha)
31       ...if (correta) {
32         ...req.session.usuario = {
33           ...id: usuario.id_usuario,
34           ...login: usuario.login
35         }
36         ...res.redirect("/")
37       } else {
38         ...res.redirect("/loginusuario");
39       }
40     } else {
41       ...res.redirect("/loginusuario")
42     }
43   })
44 })

```

Mas vamos fazer uma mudança no arquivo "index.js".

```

53 app.get("/", (req, res) => {
54   if(req.session.usuario != undefined){
55     Artigo.findAll().then((artigo) => {
56       res.render("primeiro",{artigo});
57     })
58   }else {
59     Artigo.findAll().then((artigo) => {
60       res.render("primeiro1",{artigo});
61     })
62   }
63 }
64 })

```

Note que com o que fizemos, agora para ter acesso as mudanças a categorias e artigo o usuário é preciso estar logado.

Mas se o usuário colocar outros endereços como por exemplos

localhost:3000/controlcategoria

Pode notar que passou, vamos arrumar isso

Um **middleware (intermediário)** no Express é a maneira de fazer alguma coisa antes da requisição ser processada. Como verificar se o usuário está autenticado.

Vamos criar a pasta "middleware" dentro dela vamos criar o arquivo "adminAutoriz.js"

```

1 function adminAut(req, res, next) {
2   ...if (req.session.usuario != undefined) {
3     ...next()
4   } else {
5     ...res.redirect("/")
6   }
7 }
8
9 module.exports = adminAut;

```

Vamos alterar o arquivo "controleUsuario.js"

```
1  const express = require("express");
2  const router = express.Router();
3  const bodyParser = require("body-parser");
4  const Categoria = require("../cont_usuario/Usuario");
5  const Usuario = require("../Usuario");
6  const bcrypt = require("bcryptjs");
7
8  const adminAut = require("../middleware/adminAutoriz");
9
```

Veja o exemplo abaixo vamos chamar nosso middleware, onde queremos que ele controle o acesso

```
10  router.get("/novo", adminAut, (req, res) => {
11    res.render("categorias/cad_categoria");
12  });
13
14  router.get("/controlecategoria", adminAut, (req, res) => {
15    Categoria.findAll().then((categoria) => {
16      res.render("categorias/controle_categ", { categoria });
17    });
18  });
```

Coloque a chamada do middleware onde achar necessário

```
1  <style>
2    .nav-link {
3      font-size:15px;
4    }
5  </style>
6  <nav class="navbar navbar-expand-lg navbar-dark bg-primary ">
7    <a class="navbar-brand h5" href="#">TURMA 351 - COLOQUE SEU NOME</a>
8    <ul class="navbar-nav">
9      <li class="nav-item mr-2 ml-4 ">
10       <a class="nav-link " href="/">Home </a>
11     </li>
12     <li class="nav-item mr-2">
13       <a class="nav-link" href="/novo">Categorias</a>
14     </li>
```

E vamos fazer o logout.Vamos apenas adicionar o código que esta abaixo no nosso arquivo "controleUsuario.js"

```
68  router.get("/logout", (req, res) => {
69    req.session.usuario = undefined/
70    res.redirect("/")
71  })
72
```