


Lista 18 – CRUD com API REST e ORM Prisma

A versão 2.0 do sistema tem suporte a fotos.



Cadastro de Usuários

127.0.0.1:5502/front-end/



Incluir um novo usuário

Usuários

ID	Nome	E-mail	Foto	Ações
1	Herculano De Biasi	herculano.debiasi@gmail.com		Alterar Excluir
2	Patricia	Domingues Ferreira		Alterar Excluir

Cadastro de Usuários

127.0.0.1:5502/front-end/incluir_usuario.html



Inclusão de Usuário

[Voltar](#)

Nome
Herculano De Biasi

E-mail
herculano.debiasi@gmail.com

Arquivo: eu.jpg [ESCOLHER O ARQUIVO](#)



[✓ Salvar](#) [✗ Cancelar](#)

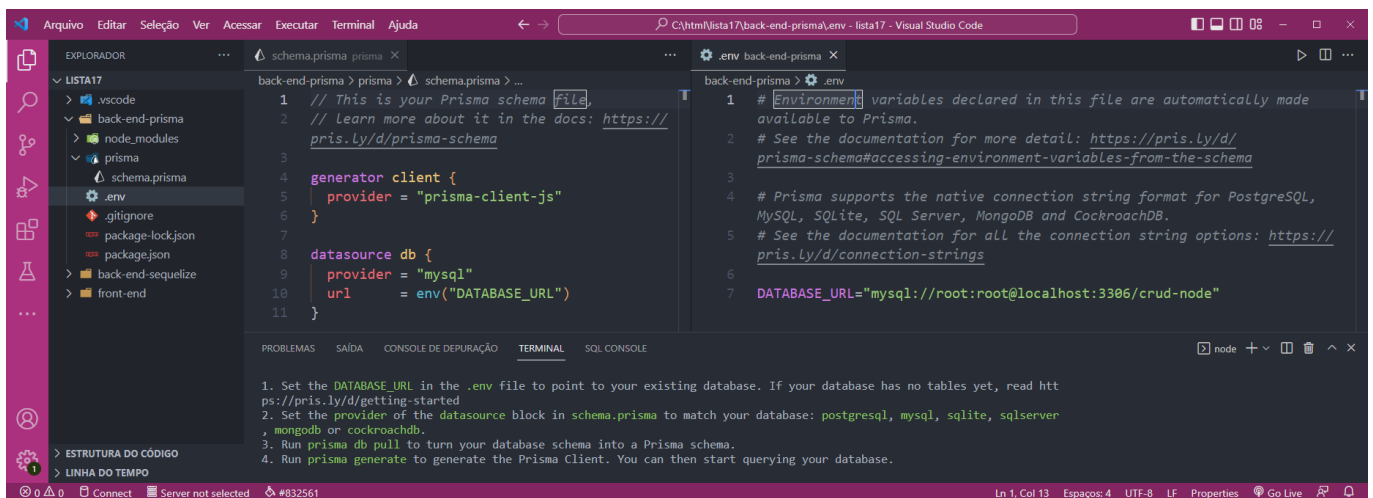
1. Crie uma pasta chamada *lista18* e dentro dela as pastas *back-end-prisma* e *front-end*. Dentro da pasta *back-end-prisma* crie outra pasta chamada *src* e dentro da pasta *src* crie o arquivo *server.js*. Certifique-se que os nomes dos arquivos e pastas estão corretos e que o arquivo *server.js* está realmente dentro da pasta *src*.
2. Abra o terminal e digite os comandos abaixo para criar o projeto node.js
 - `cd .\back-end-prisma\`
 - `npm init -y`
 - `npm i express cors multer mysql2`
 - `npm i nodemon prisma --save-dev`

Acrescentar ao *package.json*

```
"scripts": {
  "devF": "nodemon ./src/server.js",
  "test": "echo \"Error: no test specified\" && exit 1"
},
```

3. O próximo comando irá inicializar um projeto Prisma, criando as pastas e arquivos necessários para o *framework*.
 - `npx prisma init`

Modifique os arquivos *schema.prisma* e *.env* como mostrado abaixo.



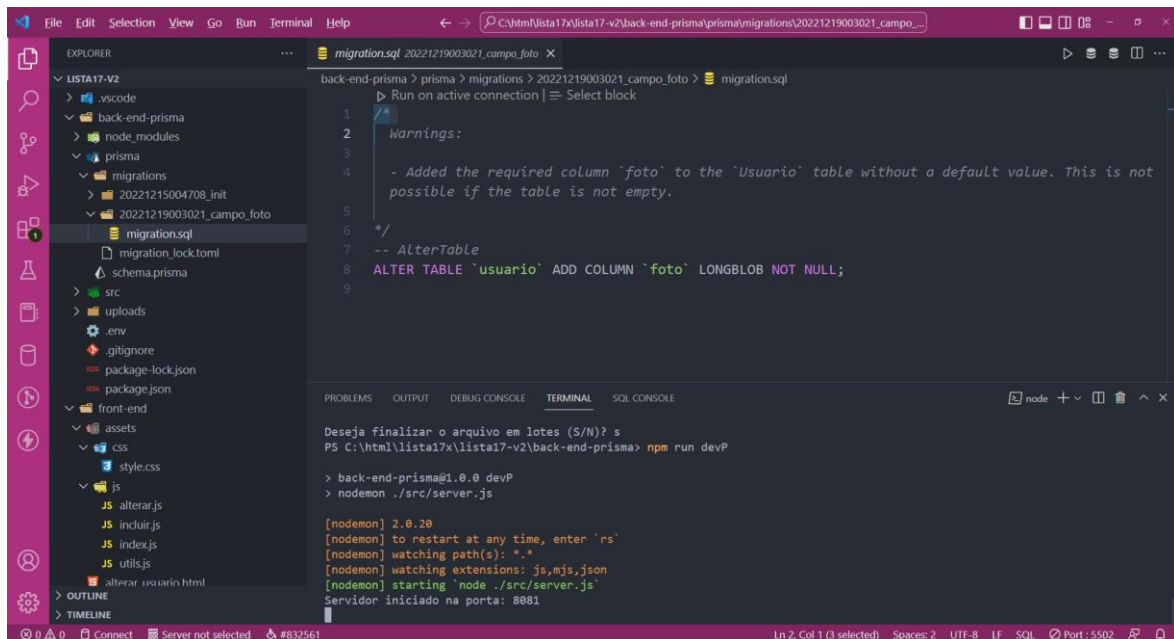
4. O sistema é feito com base no anterior desenvolvido na *lista17*, logo, o banco de dados e tabela já existem, sendo necessário apenas uma migração para incluir um novo campo.

Preencha o modelo (descrição) da tabela no arquivo *schema.prisma* e após isso execute o comando a seguir:

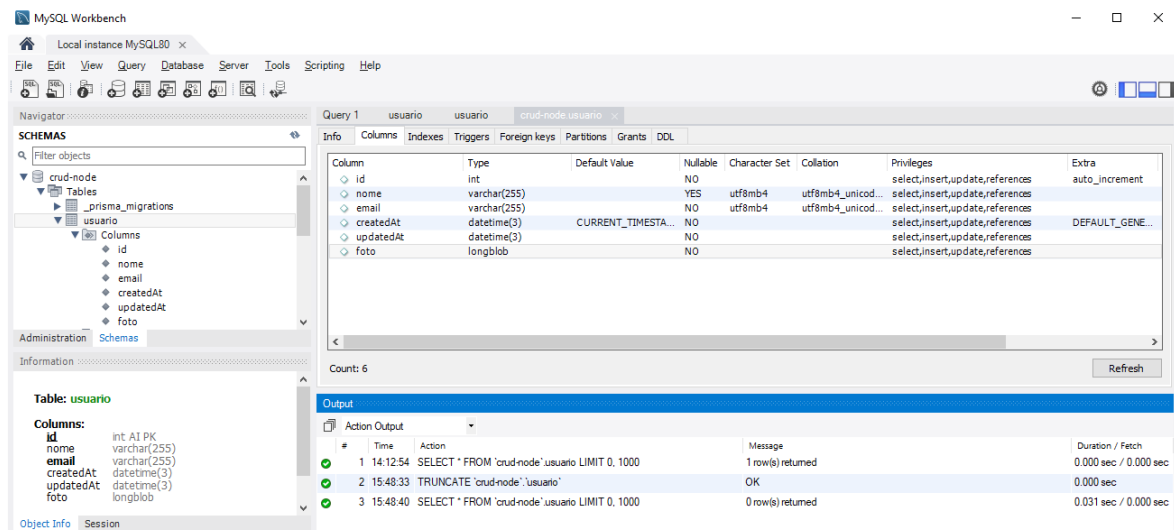
- `npx prisma migrate dev --name campo_foto`

```
lista17-v2 - schema.prisma
1 // This is your Prisma schema file,
2 // Learn more about it in the docs: https://pris.ly/d/prisma-schema
3
4 generator client {
5   provider = "prisma-client-js"
6 }
7
8 datasource db {
9   provider = "mysql"
10  url       = env("DATABASE_URL")
11 }
12
13 model Usuario {
14   id      Int      @id @default(autoincrement())
15   nome    String?  @db.VarChar(255)
16   email   String   @unique @db.VarChar(255)
17   foto    Bytes    @db.LongBlob
18   createdAt DateTime @default(now())
19   updatedAt DateTime @updatedAt
20 }
```

Abaixo, dentro da pasta *migrations*, pode ser visto o arquivo de migração que foi criado.



Pode-se ver também o banco de dados e tabela criados no MySQL Workbench.



5. Código do arquivo `server.js`.

```
lista17-v2 - server.js

1 const express = require('express');
2 const cors = require('cors');
3
4 const app = express();
5 const porta = 8081;
6
7 // Configura o CORS
8 app.use(cors({ origin: '*' }));
9
10 // Configura o tratamento das requisições POST
11 app.use(express.urlencoded({ extended: true }));
12 app.use(express.json());
13
14 app.use('/api/usuarios', require('./rotas'));
15
16 // Responde a requisição no endereço http://localhost:8081/
17 app.get('/', (request, response) => {
18   response.status(200).json({ hello: 'Olá Mundo!' });
19 });
20
21 // Instancia o servidor
22 app.listen(porta,
23   () => console.log(`Servidor iniciado na porta: ${porta}`)
24 );
```

6. Crie o arquivo *rotas.js* dentro da pasta *src* com o código abaixo. Crie também a pasta *uploads* dentro de *back-end-prisma*.

```
lista17-v2 - rotas.js

1 const controlador = require('./controllers/controladorUsuario.js');
2 const roteador = require('express').Router();
3
4 // Faz o parsing no formato multipart
5 const multer = require('multer');
6 const multerConfig = multer.diskStorage({
7   destination: 'uploads/',
8   filename(req, arquivo, callback) {
9     return callback(null, 'imagem.png');
10  },
11 });
12 const upload = multer({ storage: multerConfig });
13
14 roteador.get('/', controlador.listarUsuarios);
15 roteador.post('/', upload.single("nFoto"), controlador.criarUsuario);
16 roteador.get('/:id', controlador.buscarUsuarioPorID);
17 roteador.put('/:id', upload.single("nFoto"), controlador.alterarUsuario);
18 roteador.delete('/:id', controlador.excluirUsuario);
19
20 module.exports = roteador;
```

O tratamento dos dados do formulário e da imagem enviada por ele é feita pelo *middleware* *multer*. A biblioteca [multer](#) consegue manipular dados no formato *multipart/form-data*, utilizado principalmente para o *upload* de arquivos. O Express 4.x não está preparado nativamente para dar suporte a este tipo de estrutura de dados. Por esta razão ele não consegue analisar e processar (*parse*) os dados da requisição e retorna um objeto vazio já que não encontrou as informações enviadas.

A linha 5 carrega a biblioteca *multer*. A linha 6 cria o objeto de configuração que informa como o arquivo enviado pelo formulário será gravado no servidor. A linha 7 configura a pasta *uploads* como o lugar onde os arquivos enviados serão temporariamente gravados. A linha 9 indica que o arquivo será sempre gravado com o nome *imagem.png*.

A linha 12 instancia um objeto *multer* com a configuração definida anteriormente.

As operações de inclusão (POST) e alteração (PUT) utilizam o *middleware* de forma a gravar a foto enviada no servidor. Deve-se observar que o campo *nFoto* deve ter o mesmo nome que o campo *input file* no *front-end*.

7. Crie uma pasta chamada *controllers* dentro de *src* e dentro dela um arquivo chamado *controladorUsuario.js*.

```
lista17-v2 - controladorUsuario.js

1 const { PrismaClient } = require('@prisma/client');
2 const { Usuario } = new PrismaClient();
3 const fs = require('fs');
4
5 module.exports = {
6   async listarUsuarios(req, res) {
7     try {
8       const usuarios = await Usuario.findMany({});
9
10      if (usuarios.length === 0) {
11        return res.status(200).json({mensagem: 'Não existem usuários cadastrados!'});
12      }
13
14      res.status(200).json({usuarios});
15    } catch (error) {
16      res.json(error);
17    }
18  }
19 }
```

8. Crie a seguinte estrutura de pastas e arquivos:
- a) Dentro da pasta *front-end*, crie uma pasta chamada *assets* e dentro dela as pastas *css* e *js*.
 - b) Dentro da pasta *css* crie o arquivo *style.css*.
 - c) Dentro da pasta *js* crie 3 arquivos JavaScript: *index.js*, *incluir.js*, *alterar.js* e *utils.js*.
 - d) Diretamente dentro da pasta *front-end* crie 3 arquivos HTML: *index.html*, *incluir_usuario.html* e *alterar_usuario.html*.
9. O arquivo *style.css* terá o seguinte conteúdo:

```
lista17-v2 - style.css

1  body {
2    font-family: 'Work Sans', sans-serif;
3    line-height: 1.25;
4  }
5
6  h2 {
7    font-family: sans-serif;
8  }
9
10 table, th, tr, td {
11   margin: auto;
12   padding: 10px;
13   border: 1px solid black;
14   border-collapse: collapse;
15   text-align: center;
16   vertical-align: middle;
17 }
18
19 tr td:nth-child(2), td:nth-child(3) { text-align: left; }
20
21 #iFoto {
22   opacity: 0;
23 }
24
25 #iFoto-label {
26   position: absolute;
27   top: 50%;
28   left: 1rem;
29   transform: translateY(-40%);
30 }
31
32 #area-imagem {
33   border: 5px dashed rgba(255, 0, 0, 0.7);
34   background-color: #eee;
35   padding: 1rem;
36   position: relative;
37   width: 20%;
38 }
```

10. Arquivo *utils.js*:

```
lista17-v2 - utils.js

1  export function lerURL(input) {
2    if (input.files && input.files[0]) {
3      const reader = new FileReader();
4
5      reader.onload = function (e) {
6        const img = document.querySelector('#resultado-imagem');
7        img.setAttribute('src', e.target.result);
8      };
9
10     reader.readAsDataURL(input.files[0]);
11   }
12 }
```

11. Arquivo `index.js`:

```
lista17-v2 - index.js

1  const FRONT = 'http://localhost:5502/front-end/';
2  const API = 'http://localhost:8081/api/usuarios';
3
4  window.onload = function (e) {
5      listarUsuarios();
6  };
7
8  async function listarUsuarios() {
9      const res = await axios.get(API, {});
10
11      criarTabela(res.data.usuarios);
12  }
13
14  function criarTabela(usuarios) {
15      const corpoTabela = document.querySelector('#usuarios');
16      corpoTabela.innerHTML = '';
17
18      if (usuarios) {
19          const linhas = usuarios.map(usuario => {
20              const tdId = document.createElement('td');
21              tdId.innerHTML = usuario.id;
22
23              const tdNome = document.createElement('td');
24              tdNome.innerHTML = usuario.nome;
25
26              const tdEmail = document.createElement('td');
27              tdEmail.innerHTML = usuario.email;
28
29              const tdFoto = document.createElement('td');
30              let img = document.createElement('img');
31              img.setAttribute('width', '100px');
32
33              // Transforma um array de bytes vindos do banco em uma imagem no formato Base64
34              const base64String = new Uint8Array(usuario.foto.data);
35              const blob = new Blob([base64String], { type: 'image/png' });
36              img.src = URL.createObjectURL(blob);
37              tdFoto.appendChild(img);
38
39              const acaoAlterar = document.createElement('a');
40              acaoAlterar.innerHTML = 'Alterar';
41              acaoAlterar.setAttribute('href', FRONT + 'alterar_usuario.html?id=' + usuario.id);
42              acaoAlterar.classList.add('btn', 'btn-primary', 'me-2');
43
44              const acaoExcluir = document.createElement('a');
45              acaoExcluir.innerHTML = 'Excluir';
46              acaoExcluir.classList.add('btn', 'btn-danger');
47
48              acaoExcluir.addEventListener('click', function (event) {
49                  if (confirm('Tem certeza que deseja excluir?')) {
50                      axios.delete(API + '/' + usuario.id, { })
51                          .then(res => {
52                              listarUsuarios();
53                          });
54                  }
55              }, false);
56
57              const tdAcoes = document.createElement('td');
58              tdAcoes.appendChild(acaoAlterar);
59              tdAcoes.appendChild(acaoExcluir);
60
61              const tr = document.createElement('tr');
62              tr.appendChild(tdId);
63              tr.appendChild(tdNome);
64              tr.appendChild(tdEmail);
65              tr.appendChild(tdFoto);
66              tr.appendChild(tdAcoes);
67
68              return tr;
69          });
70
71      linhas.forEach(linha => corpoTabela.appendChild(linha));
72  }
73 }
```


12. Arquivo *index.html*:

```
lista17-v2 - index.html

1  <!DOCTYPE html>
2  <html lang="pt-BR">
3
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, shrink-to-fit=no">
8
9      <link href="https://fonts.googleapis.com/css2?family=Work+Sans:wght@200;400;500;700&display=swap"
10 rel="stylesheet">
11
12      <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/css/bootstrap.min.css" rel="stylesheet" integrity=
13 "sha384-Zenh87qX5JnK2Jl0vWa8Ck2rdkQ2Bzep5IDxbcnCeuOxjzrPF/et3URy9Bv1WTRI" crossorigin="anonymous">
14      <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/font-awesome@6.2.0/css/all.min.css" crossori
15 gin="anonymous">
16      <link rel="stylesheet" href="./assets/css/style.css">
17
18      <script src="https://cdn.jsdelivr.net/npm/axios/dist/axios.min.js"></script>
19
20      <script src="./assets/js/index.js"></script>
21
22      <title>Cadastro de Usuários</title>
23 </head>
24
25 <body>
26     <div class="container my-2 col">
27         <div class="d-flex justify-content-center">
28             
29         </div>
30
31         <hr>
32         <a href="incluir_usuario.html" class="btn btn-success">
33             Incluir um novo usuário
34             <i class="fa-solid fa-user"></i>
35         </a>
36         <hr>
37
38         <table class="table table-striped table-condensed table-hover table-responsive">
39             <colgroup>
40                 <col class="col-3 col-sm-1 bg-secondary">
41                 <col class="col-3 col-sm-4">
42                 <col class="col-3 col-sm-3">
43                 <col class="col-3 col-sm-1">
44                 <col class="col-3 col-sm-3">
45             </colgroup>
46
47             <caption style="font-size: 2em; font-weight: bold; text-align: center; caption-side: top">
48                 Usuários
49             </caption>
50
51             <thead class="table-dark">
52                 <tr>
53                     <th>ID</th>
54                     <th>Nome</th>
55                     <th><em>E-mail</em></th>
56                     <th>Foto</th>
57                     <th>Ações</th>
58                 </tr>
59             </thead>
60
61             <tbody id="usuarios"></tbody>
62         </table>
63         <hr>
64     </div>
65
66     <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/js/bootstrap.bundle.min.js" integrity="sha384-OER
67 cA2EqjJCMA+/3y+gxIOqMEjwtxJY7qPCqsdltbNJua0e923+mo//f6V8Qbsw3" crossorigin="anonymous"></script>
68 </body>
69 </html>
```

13. Para a inclusão acrescente o código no *controladorUsuario.js*:

```
JS controladorUsuario.js controllers X
back-end-prisma > src > controllers > JS controladorUsuario.js > ...
1  const { PrismaClient } = require('@prisma/client');
2  const { Usuario } = new PrismaClient();
3  const fs = require('fs');
4
5  module.exports = {
6  >  async listarUsuarios(req, res) { ...
18  },
19
20  async criarUsuario(req, res) {
21      const { nNome: nome, nEmail: email } = req.body;
22
23      // Carrega a foto da pasta 'uploads' gravada pelo middleware multer
24      const foto = req.file? fs.readFileSync(req.file.path) : new Buffer('', 'base64');
25
26      try {
27          let usuario = await Usuario.findUnique({ where: { email } });
28          if (usuario) {
29              return res.status(401).json({ error: 'E-mail já cadastrado!' });
30          }
31
32          usuario = await Usuario.create({
33              data: { nome, email, foto }
34          });
35
36          res.status(200).json({ mensagem: 'Usuário incluído com sucesso!', usuario });
37      } catch (error) {
38          res.json(error);
39      } finally {
40          fs.unlinkSync('uploads/imagem.png');
41      }
42  }
43  }
```

14. O código do arquivo *incluir.js* é mostrado a seguir:

```
lista17-v2 - incluir.js
1  import { lerURL } from './utils.js';
2
3  let form, fotoForm, fotoInfoForm;
4  const sURL = 'http://localhost:8081/api/usuarios/';
5
6  window.onload = function (e) {
7      form = document.querySelector('#form');
8
9      fotoForm = document.querySelector('#iFoto');
10     fotoInfoForm = document.querySelector('#iFoto-label');
11
12     fotoForm.addEventListener('change', (e) => {
13         lerURL(fotoForm);
14
15         const nomeArquivo = e.currentTarget.files[0].name;
16         fotoInfoForm.textContent = 'Arquivo: ' + nomeArquivo;
17     });
18 };
19
20 async function incluirUsuario() {
21     try {
22         const dadosForm = new FormData(form);
23         await axios.post(sURL, dadosForm);
24     } catch (error) {
25         console.log(error);
26     } finally {
27         window.location = 'http://localhost:5502/front-end';
28     }
29 }
30
31 window.incluirUsuario = incluirUsuario;
```


15. Arquivo `incluir_usuario.html`:

```
lista17-v2 - incluir_usuario.html

1 <!DOCTYPE html>
2 <html lang="pt-BR">
3
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, shrink-to-fit=no">
8
9   <link href="https://fonts.googleapis.com/css2?family=Work+Sans:wght@200;400;500;700&display=swap" rel="stylesheet">
10
11   <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-Zenh87qX
12 5JnK2Jl0vWa8Ck2rdkQ2Bzep5IDxbcnCeu0xjzrPF/et3URy9Bv1WTRI" crossorigin="anonymous">
13
14   <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/axios/dist/axios.min.css" crossorigin="anonymous">
15
16   <link rel="stylesheet" href="./assets/css/style.css">
17
18   <script src="https://cdn.jsdelivr.net/npm/axios/dist/axios.min.js"></script>
19
20   <script type="module" src="./assets/js/incluir.js"></script>
21
22   <title>Cadastro de Usuários</title>
23 </head>
24 <body>
25   <div class="container my-2 col">
26     <div class="d-flex justify-content-center">
27       
28     </div>
29
30     <h2 class="alert alert-info mt-2">Inclusão de Usuário</h2>
31
32     <a href="#" onclick="window.history.back()">Voltar</a>
33     <hr>
34
35     <form class="was-validated" id="form" class="row gx-3 gy-0">
36       <div class="form-floating mb-3">
37         <input type="text" name="nNome" id="iNome" class="form-control" placeholder="Entre com seu nome" required>
38         <label for="iNome">Nome</label>
39       </div>
40
41       <div class="form-floating mb-3">
42         <input type="email" name="nEmail" id="iEmail" class="form-control" placeholder="Entre com seu e-mail" required>
43         <label for="iEmail"><em>E-mail</em></label>
44       </div>
45
46       <div class="form-group">
47         <div class="input-group mb-3 px-2 py-2 rounded-pill bg-white shadow-sm">
48           <input id="iFoto" type="file" name="nFoto" id="iFoto" class="form-control" accept="image/*">
49           <label id="iFoto-label" for="iFoto" class="font-weight-light text-muted">Selecione uma foto</label>
50
51           <div class="input-group-append">
52             <label for="iFoto" class="btn btn-dark m-0 rounded-pill px-4">
53               <i class="fa fa-cloud-upload mr-2"></i>
54               <small class="text-uppercase font-weight-bold">Escolher o arquivo</small>
55             </label>
56           </div>
57         </div>
58
59         <div id="area-imagem" class="mt-4 mb-4 mx-auto">
60           
61         </div>
62       </div>
63
64       <div class="form-group mb-3 text-center">
65         <div id="operacao" class="d-inline">
66           <button type="button" class="btn btn-success" onclick="incluirUsuario()">
67             <i class="fa-solid fa-check"></i>
68             Salvar
69           </button>
70
71           <button type="button" class="btn btn-danger" onclick="window.history.back()">
72             <i class="fa-solid fa-cancel"></i>
73             Cancelar
74           </button>
75         </div>
76       </div>
77     </form>
78
79     <div id="mensagem"></div>
80 </div>
81
82 <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/js/bootstrap.bundle.min.js" integrity="sha384-0ERcA2EqJCMA+/3y+g
83 xIOqMEjwtxJY7qPCqsdltbNjua0e923+mo//f6V8Qbsw3" crossorigin="anonymous"></script>
84
85 </html>
```

16. Para alterar um usuário é necessário a função `buscarUsuarioPorID()` no arquivo `controladorUsuario.js`:

```
JS controladorUsuario.js controllers X
back-end-prisma > src > controllers > JS controladorUsuario.js > [?] <unknown> > criarUsuario
1  const { PrismaClient } = require('@prisma/client');
2  const { Usuario } = new PrismaClient();
3  const fs = require('fs');
4
5  module.exports = {
6  >   async listarUsuarios(req, res) { ...
18   },
19
20 >   async criarUsuario(req, res) { ...
42   },
43
44   async buscarUsuarioPorID(req, res) {
45       try {
46           const { id } = req.params;
47
48           const usuario = await Usuario.findUnique({ where: { id: Number(id) } });
49           if (!usuario) {
50               return res.status(200).json({ mensagem: 'Usuário não encontrado!' });
51           }
52
53           res.status(200).json(usuario);
54       } catch (error) {
55           res.json(error);
56       }
57   },
58 }
```

E também a função `alterarUsuario()`, no mesmo arquivo:

```
JS controladorUsuario.js controllers X
back-end-prisma > src > controllers > JS controladorUsuario.js > ...
58
59   async alterarUsuario(req, res) {
60       try {
61           const { id } = req.params;
62           const { nNome: nome, nEmail: email } = req.body;
63           const foto = req.file? fs.readFileSync(req.file.path) : new Buffer.from('', 'base64');
64
65           let usuario = await Usuario.findUnique({ where: { id: Number(id) } });
66           if (!usuario) {
67               return res.status(200).json({ mensagem: 'Usuário não encontrado!' });
68           }
69
70           usuario = await Usuario.update({
71               where: { id: Number(id) },
72               data: { nome, email, foto }
73           });
74
75           return res.status(200).json({ mensagem: 'Usuário alterado com sucesso!', usuario });
76       } catch (error) {
77           res.json(error);
78       } finally {
79           fs.unlinkSync('uploads/imagem.png');
80       }
81   }
82 }
```

17. A seguir, o arquivo *alterar.js*:

```
lista17-v2 - alterar.js

1  import { lerURL } from './utils.js';
2
3  let id, nomeForm, emailForm, form, fotoForm, fotoInfoForm, imgForm;
4  const sURL = 'http://localhost:8081/api/usuarios/';
5
6  window.onload = async function (e) {
7      form = document.querySelector('#form');
8
9      nomeForm = document.querySelector('#iNome');
10     emailForm = document.querySelector('#iEmail');
11
12     fotoForm = document.querySelector('#iFoto');
13     fotoInfoForm = document.querySelector('#iFoto-label');
14     imgForm = document.querySelector('#resultado-imagem');
15
16     fotoForm.addEventListener('change', (e) => {
17         lerURL(fotoForm);
18
19         const input = e.currentTarget;
20         const nomeArquivo = input.files[0].name;
21         fotoInfoForm.textContent = 'Arquivo: ' + nomeArquivo;
22     });
23
24     const query = window.location.search;
25     const parametros = new URLSearchParams(query);
26     id = parametros.get('id');
27
28     const usuario = await buscarUsuario(id);
29     preencherForm(usuario);
30 };
31
32 function preencherForm(usuario) {
33     nomeForm.value = usuario.nome;
34     emailForm.value = usuario.email;
35
36     // Recupera e mostra foto armazenada no banco de dados
37     const base64String = new Uint8Array(usuario.foto.data);
38     const blob = new Blob([base64String], { type: 'image/png' });
39     imgForm.src = URL.createObjectURL(blob);
40 }
41
42 async function buscarUsuario(id) {
43     const resposta = await axios.get(sURL + id);
44
45     return resposta.data;
46 }
47
48 async function alterarUsuario(e) {
49     try {
50         const dadosForm = new FormData(form);
51
52         const resposta = await axios.put(sURL + id, dadosForm);
53         console.log(resposta);
54     } catch (error) {
55         console.log(error);
56     } finally {
57         window.location = 'http://localhost:5502/front-end';
58     }
59
60     return false;
61 }
62
63 window.alterarUsuario = alterarUsuario;
```

18. E, finalmente, o arquivo `alterar_usuario.html`:

```
lista17-v2 - alterar_usuario.html

1 <!DOCTYPE html>
2 <html lang="pt-BR">
3
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, shrink-to-fit=no">
8
9   <link href="https://fonts.googleapis.com/css2?family=Work+Sans:wght@200;400;500;700&display=swap" rel="stylesheet">
10
11   <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/css/bootstrap.min.css" rel="stylesheet"
12     integrity="sha384-Zenh87qX5JnK2Jl0vWa8Ck2rdkQ2Bzep5IDxbcnCeu0xjzrPF/et3URy9Bv1WTRI" crossorigin="anonymous">
13
14   <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.2.0/css/all.min.css"
15     crossorigin="anonymous">
16
17   <link rel="stylesheet" href="./assets/css/style.css">
18
19   <script src="https://cdn.jsdelivr.net/npm/axios/dist/axios.min.js"></script>
20
21   <script type="module" src="./assets/js/alterar.js"></script>
22
23   <title>Cadastro de Usuários</title>
24 </head>
25
26 <body>
27   <div class="container my-2 col">
28     <div class="d-flex justify-content-center">
29       
30     </div>
31
32     <h2 class="alert alert-info mt-2">Alteração de Usuário</h2>
33
34     <a href="#" onclick="window.history.back()">Voltar</a>
35     <hr>
36
37     <form class="was-validated" id="form" class="row gx-3 gy-0">
38       <div class="form-floating mb-3">
39         <input type="text" name="nNome" id="iNome" class="form-control" placeholder="Entre com seu nome"
40           required>
41         <label for="iNome">Nome</label>
42       </div>
43
44       <div class="form-floating mb-3">
45         <input type="email" name="nEmail" id="iEmail" class="form-control" placeholder="Entre com seu e-mail"
46           required>
47         <label for="iEmail"><em>E-mail</em></label>
48       </div>
49
50       <div class="form-group">
51         <div class="input-group mb-3 px-2 py-2 rounded-pill bg-white shadow-sm">
52           <input id="iFoto" type="file" name="nFoto" id="iFoto" class="form-control" accept="image/*">
53           <label id="iFoto-label" for="iFoto" class="font-weight-light text-muted">Escolher o arquivo</label>
54
55           <div class="input-group-append">
56             <button for="iFoto" class="btn btn-dark m-0 rounded-pill px-4">
57               <i class="fa fa-cloud-upload mr-2"></i>
58               <small class="text-uppercase font-weight-bold">Selecione uma foto</small>
59             </button>
60           </div>
61         </div>
62
63         <div id="area-imagem" class="mt-4 mb-4 mx-auto">
64           
65         </div>
66       </div>
67
68       <div class="form-group mb-3 text-center">
69         <div id="operacao" class="d-inline">
70           <button type="button" class="btn btn-success" onclick="alterarUsuario()">
71             <i class="fa-solid fa-check"></i>
72             Salvar
73           </button>
74         </div>
75
76         <button type="button" class="btn btn-danger" onclick="window.history.back()">
77           <i class="fa-solid fa-cancel"></i>
78           Cancelar
79         </button>
80       </div>
81     </form>
82
83     <div id="mensagem"></div>
84   </div>
85
86   <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/js/bootstrap.bundle.min.js"
87     integrity="sha384-OERcA2EqjJCMA+/3y+gxIOqMEjwtxJY7qPCqsdltbNJua0e923+mo//f6V8Qbsw3"
88     crossorigin="anonymous"></script>
89 </body>
90
91 </html>
```

19. Para a operação de exclusão basta acrescentar mais um método no controlador:

```
JS controladorUsuario.js controllers X
back-end-prisma > src > controllers > JS controladorUsuario.js > [?] <unknown>
83   async excluirUsuario(req, res) {
84     try {
85       const { id } = req.params;
86
87       const usuario = await Usuario.findUnique({ where: { id: Number(id) } });
88       if (!usuario) {
89         return res.status(200).json({ mensagem: 'Usuário não encontrado!' });
90       }
91
92       await Usuario.delete({ where: { id: Number(id) } });
93       res.status(200).json({ mensagem: 'Usuário excluído!' });
94     } catch (error) {
95       res.json(error);
96     }
97   }
98 }
```