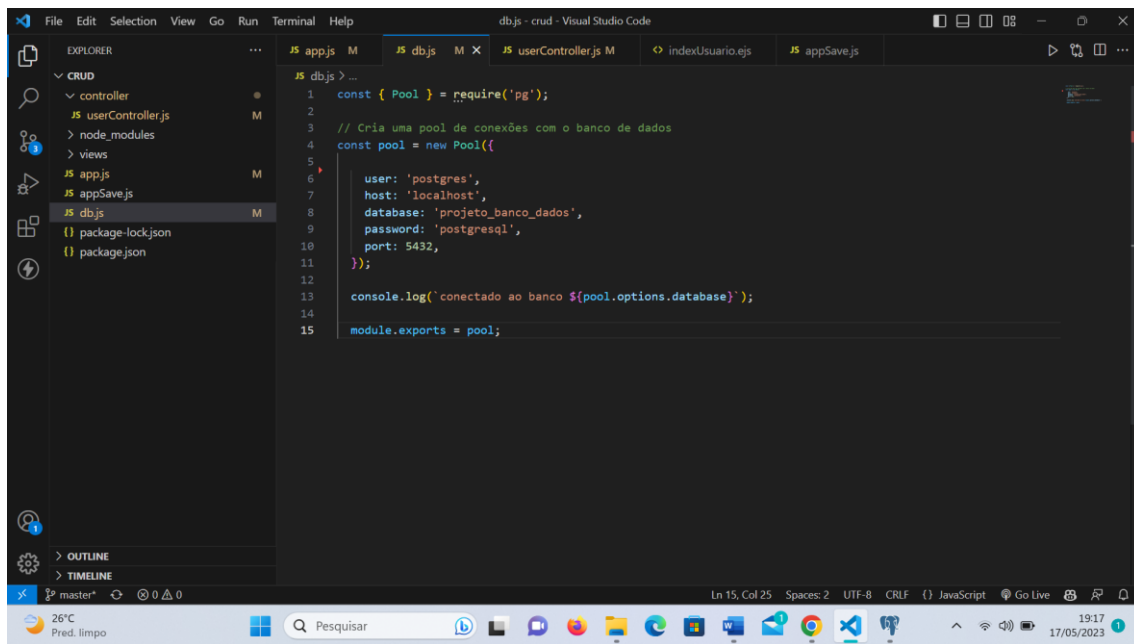


Print da conexão com o banco de dados

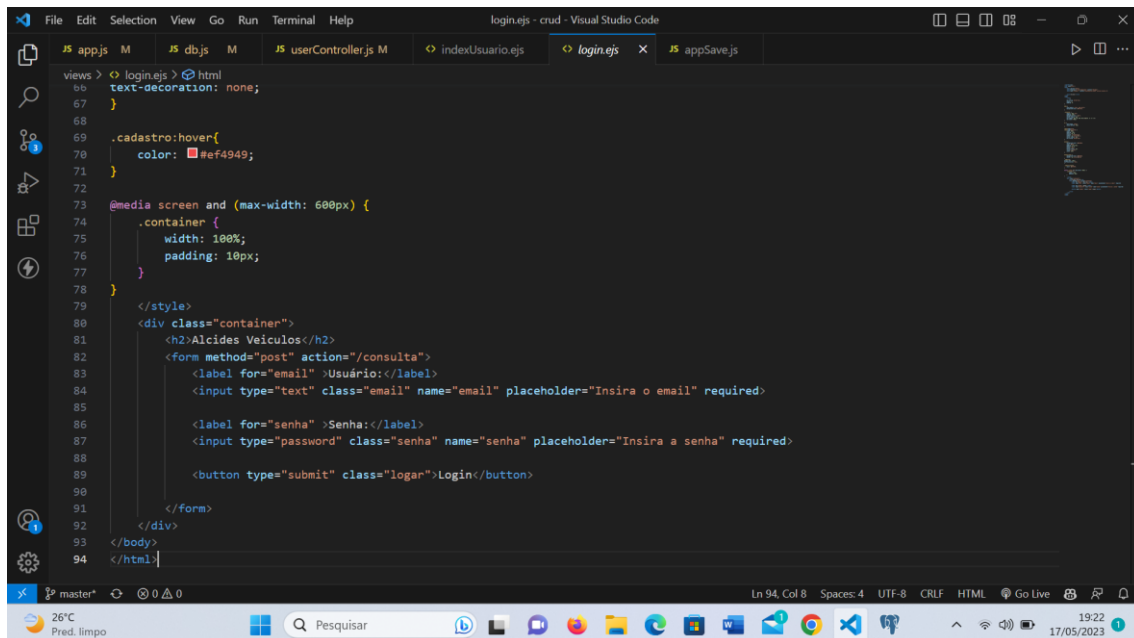


```
1 const { Pool } = require('pg');
2
3 // Cria uma pool de conexões com o banco de dados
4 const pool = new Pool({
5
6   user: 'postgres',
7   host: 'localhost',
8   database: 'projeto_banco_dados',
9   password: 'postgresql',
10  port: 5432,
11 });
12
13 console.log('conectado ao banco ${pool.options.database}');
14
15 module.exports = pool;
```

A conexão do Postgres usando o Node.js é feita criando uma instância da classe “Pool” e definindo os parâmetros do banco dentro dele, como usuário, senha, porta etc.

Após isso o trecho “module.exports” permite a visibilidade da conexão com o banco no restante do projeto.

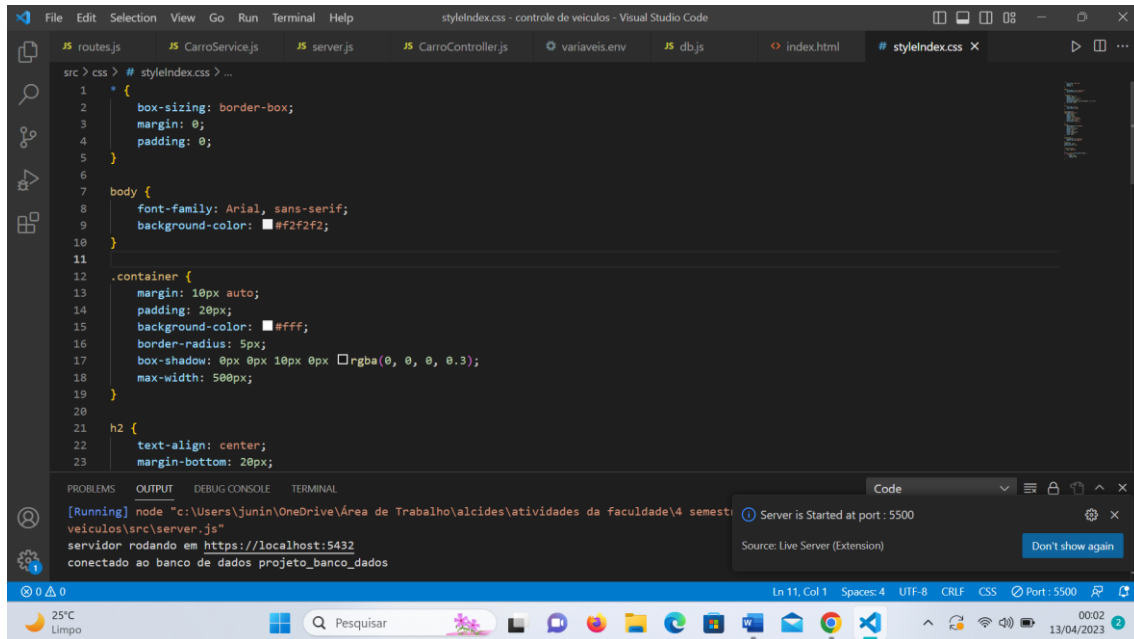
Código HTML da tela de login



```
66 text-decoration: none;
67 }
68
69 .cadastro:hover{
70   color: #ef4949;
71 }
72
73 @media screen and (max-width: 600px) {
74   .container {
75     width: 100%;
76     padding: 10px;
77   }
78 }
79
80 </style>
81 <div class="container">
82   <h2>Alcides Veiculos</h2>
83   <form method="post" action="/consulta">
84     <label for="email" >Usuário:</label>
85     <input type="text" class="email" name="email" placeholder="Insira o email" required>
86
87     <label for="senha" >Senha:</label>
88     <input type="password" class="senha" name="senha" placeholder="Insira a senha" required>
89
90     <button type="submit" class="login">Login</button>
91   </form>
92 </div>
93 </body>
94 </html>
```

Uma tela simples de login contendo dois inputs para “email” e “senha” que serão consultados no banco.

Código CSS



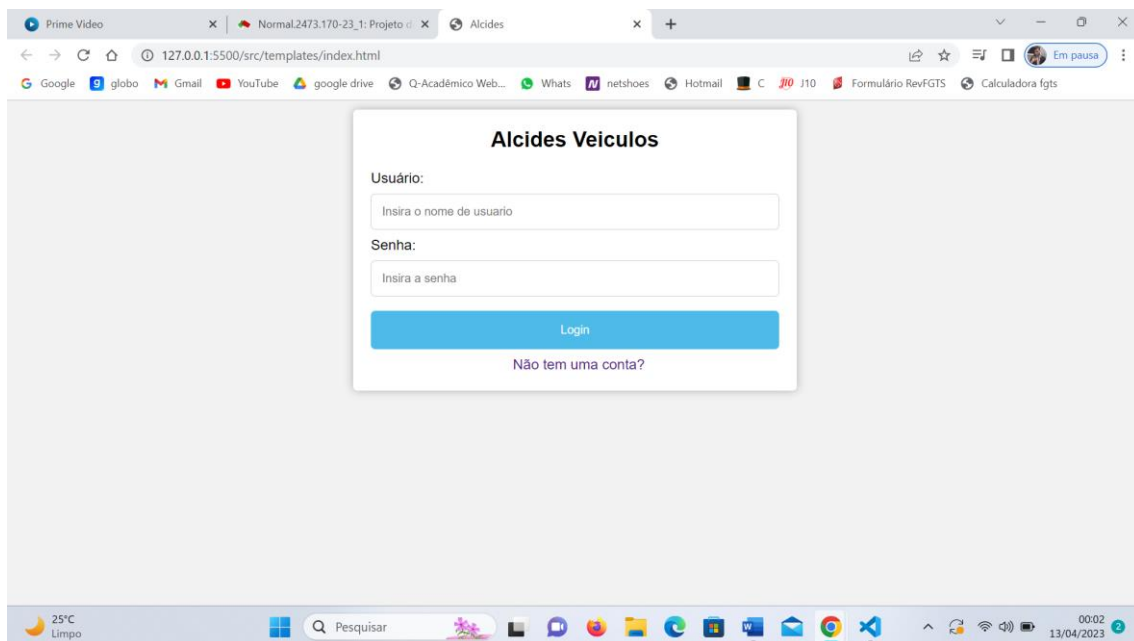
```
1 *  
2 {  
3   box-sizing: border-box;  
4   margin: 0;  
5   padding: 0;  
6 }  
7  
8 body {  
9   font-family: Arial, sans-serif;  
10  background-color: #f2f2f2;  
11 }  
12  
13 .container {  
14   margin: 10px auto;  
15   padding: 20px;  
16   background-color: #fff;  
17   border-radius: 5px;  
18   box-shadow: 0px 0px 10px 0px rgba(0, 0, 0, 0.3);  
19   max-width: 500px;  
20 }  
21  
22 h2 {  
23   text-align: center;  
24   margin-bottom: 20px;
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

[Running] node "c:\Users\junin\OneDrive\Área de Trabalho\alcides\atividades da faculdade\4 semest...
veiculos\src\server.js"
servidor rodando em https://localhost:5432
conectado ao banco de dados projeto_banco_dados

Server is Started at port : 5500
Source: Live Server (Extension)
Don't show again

Print da tela de Login



Validação do login no banco de dados

```
async consultaUsuario(req,res){
  const { email, senha } = req.body;

  // Consulta os dados do formulário no banco de dados
  pool.query('SELECT * FROM usuario WHERE email = $1 AND senha = $2', [email, senha], (error, results) => {
    if (error) {
      console.error(error);
      res.status(500).send('Erro ao executar consulta no banco de dados');
    } else if (results.rows.length > 0) {

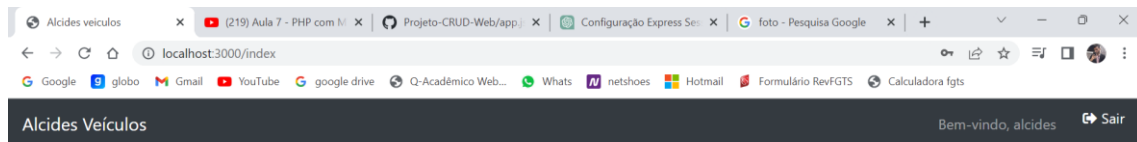
      // Armazena o ID do usuário na sessão
      req.session.userID = results.rows[0].cod_usuario;

      //Armazena o Nome do usuario na sessão
      req.session.username = results.rows[0].nome;

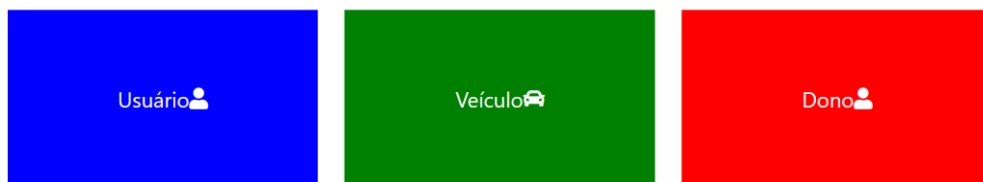
      // O login e a senha existem no banco de dados
      res.redirect('/index');
    } else {
      // O login e/ou a senha estão incorretos
      res.status(401).send('Login e/ou senha incorretos');
    }
  });
},
```

Neste trecho, a rota “/consulta” faz um select na tabela usuário e compara com o email e senha fornecidos pelo usuário, se a consulta não for feita retornará o erro de status=500, Se a consulta for bem-sucedida e retornarem alguma linha de resultado, significa que existe o login e senha no banco e guarda o id do usuário que conectou e o nome do usuário, redireciona para a página Index do projeto, caso não retorne nada na consulta, retornará o erro 401 de “login e/ou senha incorretos”.

Tela inicial do projeto



Alcides Veículos



Deste ponto em diante só é possível acessar fazendo login, na navbar retorna bem vindo e o primeiro nome do usuário que logou, no botão de sair faz o logout do usuário e os 3 quadrados redirecionam para os respectivos CRUDS.

```
// autentificação

function protecao(req, res, next) {
  if (req.session.userID) {
    // O usuário está autenticado, permita o acesso à próxima rota
    next();
  } else {
    req.session.message = 'Faça o login para acessar esta página.';
    // O usuário não está autenticado, redirecione para a página de login
    res.redirect('/');
  }
}

module.exports = protecao;
```

Função de proteção de telas usando a biblioteca express session, ela checa se existe a variável req.session.userID(ela existe quando o login é bem sucedido) se existir ela deixa avançar no projeto, se não ela retorna uma mensagem de que precisa de login para acessar a página e retorna para o formulário de login

Método de logout

```
async logOut(req, res){
  req.session.destroy((error)=>{
    if(error){
      console.error(error);
      res.status(500).send('Erro ao fazer logout');
    }else{
      res.render('logout');
    }
  });
},
```

Destroi a sessão do usuário e renderiza a página de logout, que mostra que a sessão foi encerrada e retorna para a página de login

Página index

```
async paginaIndex(req,res){
  const username = req.session.username;

  res.render('index', { username })
},
```

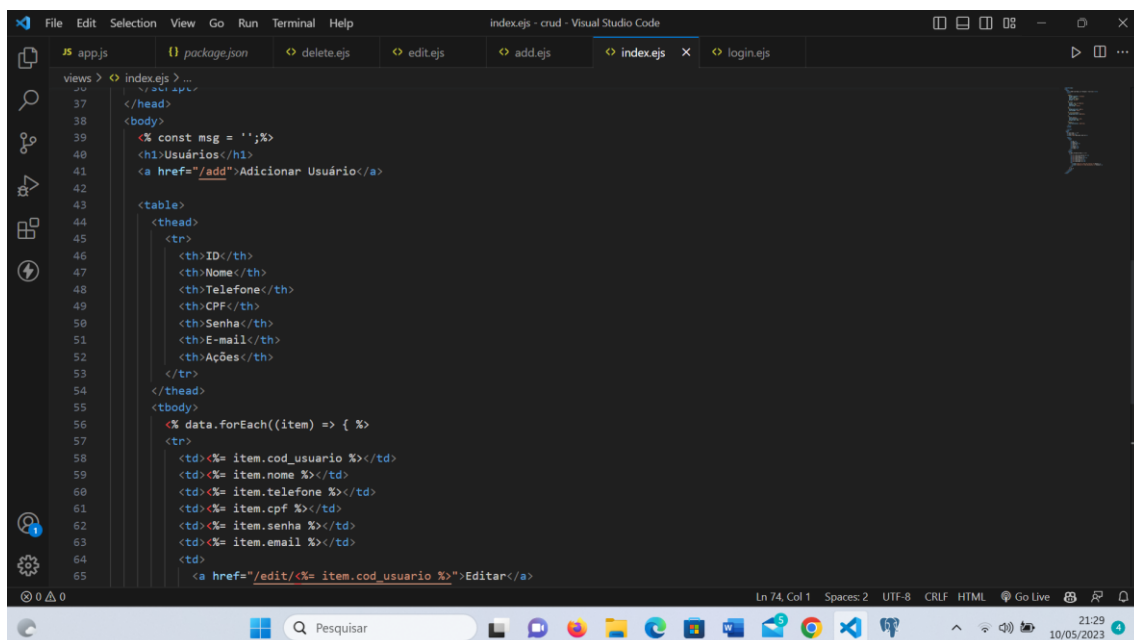
Renderiza a página inicial com a variável username(que é criada no login)

Rota da página inicial Crud usuário(READ)

```
app.get('/index', async (req, res) => {
  try {
    const result = await pool.query('SELECT * FROM usuario ORDER BY COD_USUARIO');
    res.render('index', { data: result.rows });
  } catch (err) {
    console.error(err);
    res.send('Erro ao buscar dados');
  }
});
```

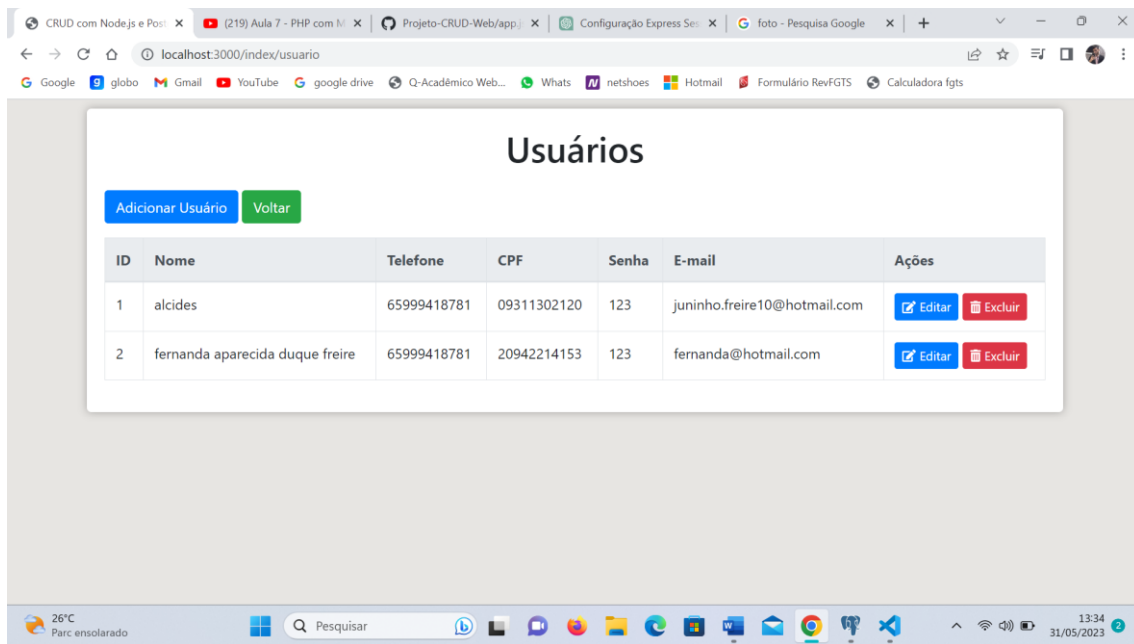
Neste trecho, é feito o select no banco das informações do usuário ordenadas pelo código, a cláusula await é usado para aguardar a consulta antes de seguir com a execução, se a consulta for bem-sucedida, ira retornar a página index do meu projeto com a listagem dos resultados do select, se não retornará um erro

Código da página



Usando a view engine “ejs” para renderizar minhas páginas, a parte estática do meu formulário foi feita com código html e css, e a parte dinâmica é feita uma cláusula foreach passando o parâmetro item onde cada item dentro do banco ira alimentar as linhas “td” dentro do meu formulário html com as informações do banco.

Tela que retorna



A parte dinâmica desta tela foi gerada com os dados dentro do banco

Rota para a página de adição de usuário

```
app.get('/add', (req, res) => {  
  res.render('add');  
});
```

Rota acionada quando clicar em “adicionar usuário” na pagina de listagem, ela renderizará a pagina “add” do meu projeto

Rota e logica de adição (CREATE) de usuário

```

app.post('/add', async (req, res) => {
  try {
    const {nome, telefone, cpf, senha, email} = req.body;

    const result = await pool.query(
      'INSERT INTO usuario (nome, telefone, cpf, senha, email) VALUES ($1, $2, $3, $4, $5)',
      [nome, telefone, cpf, senha, email]
    );
    res.redirect('/index');
  } catch (err) {
    console.error(err);
    res.send('Erro ao adicionar dados');
  }
});

```

Neste trecho, defini os parâmetros nome, telefone, cpf, senha e email como os atributos do “body” que tem a tag name correspondente, efetuei o código sql de inserção no banco dos valores correspondentes e redirecionei para a página de listagem, caso capture algum erro, retornará o erro.

Tela de adição de usuário

The screenshot shows a web browser window with the title 'Adicionar Usuário'. The address bar displays 'localhost:3000/add/usuario'. The form contains the following fields and a button:

- Nome:
- Telefone:
- CPF:
- Senha:
- Email:
-

The browser's taskbar at the bottom shows the date and time as 13:34 on 31/05/2023.

Rota para a página de edição de um usuário específico

```

app.get('/edit/:cod_usuario', async (req, res) => {
  const { cod_usuario } = req.params;
  try {
    const result = await pool.query('SELECT * FROM usuario WHERE cod_usuario = $1', [cod_usuario]);
    res.render('edit', { data: result.rows[0] });
  } catch (err) {
    console.error(err);
    res.send('Erro ao buscar dados');
  }
});

```

Na parte da rota “/:cod_usuario” ira ser preenchida pelo código do usuário específico quer for selecionado para edição.

Rota e lógica de edição(UPDATE) de usuário específico

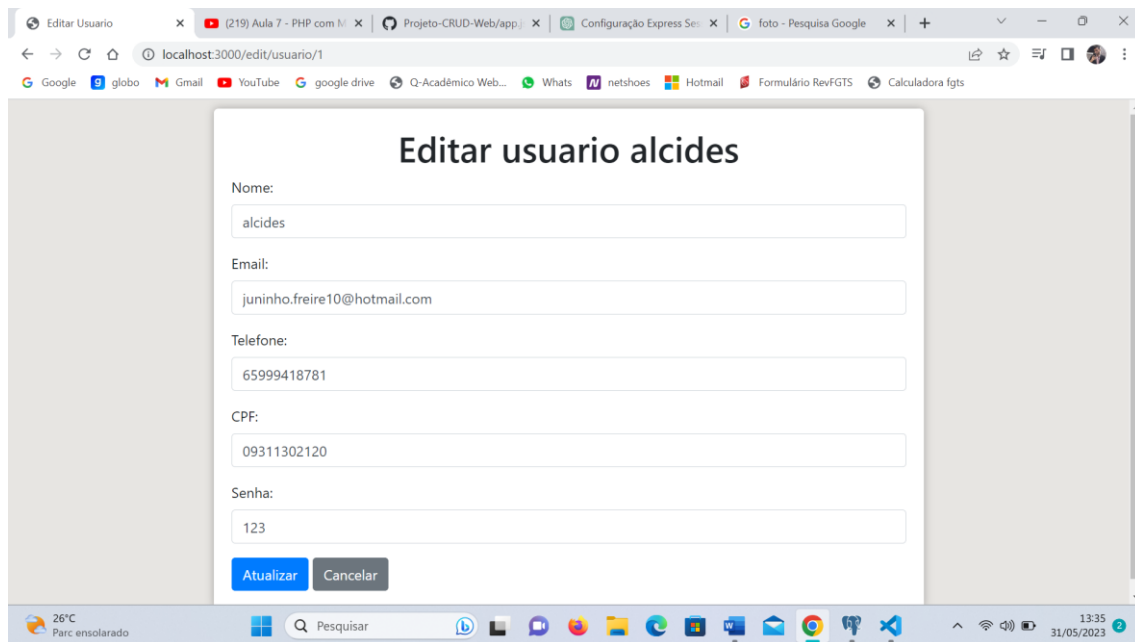
```

app.post('/edit/:cod_usuario', async (req, res) => {
  const { cod_usuario } = req.params;
  const { nome, email, telefone, senha, cpf } = req.body;
  try {
    await pool.query('UPDATE usuario SET nome = $1, email = $2, telefone = $3, senha = $4, cpf = $5 WHERE cod_usuario = $6', [nome, email,
    res.redirect('/index');
  } catch (err) {
    console.error(err);
    res.redirect('/edit/${cod_usuario}?msg=Erro ao atualizar dados');
  }
});

```

“{cod_usuario} = req.params” é feito a desestruturação para extrair o valor do código da solicitação req, assim como é feita na próxima linha, o update é feito nos campos que são passados como parâmetros e redirecionando para a página de listagem, caso ocorra um erro, será retornado o erro com uma mensagem específica

Tela de edição de usuário



The screenshot shows a web browser window with the address bar displaying 'localhost:3000/edit/usuario/1'. The page title is 'Editar usuario alcides'. The form contains the following fields and values:

- Nome: alcides
- Email: juninho.freire10@hotmail.com
- Telefone: 65999418781
- CPF: 09311302120
- Senha: 123

At the bottom of the form are two buttons: 'Atualizar' (highlighted in blue) and 'Cancelar'.

O método de atualização é acionado ao pressionar o botão atualizar

Rota para a tela de exclusão de usuário específico

```
app.get('/delete/:cod_usuario', async (req, res) => {
  const { cod_usuario } = req.params;
  try {
    const result = await pool.query('SELECT * FROM usuario WHERE cod_usuario = $1', [cod_usuario]);
    if (result.rowCount === 0) {
      return res.send('Item não encontrado');
    }
    res.render('delete', { data: result.rows[0] });
  } catch (err) {
    console.error(err);
    res.send('Erro ao buscar dados');
  }
});
```

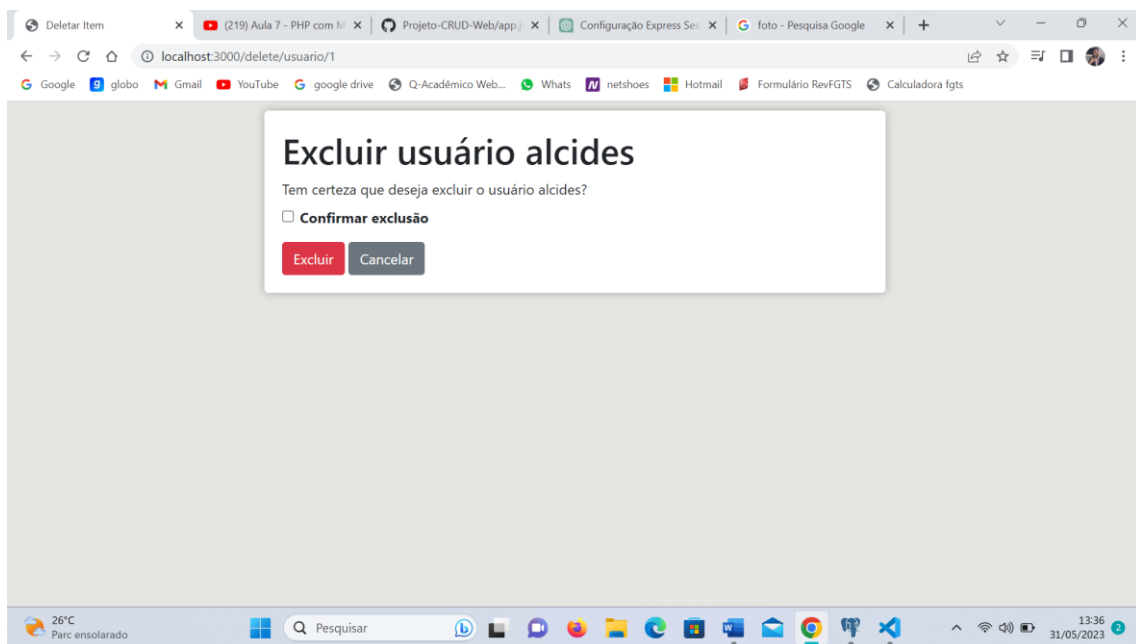
Faz o select no banco com base no id passado como parâmetro, se o resultado da consulta não retornar nada como resposta, ira retornar a mensagem de item não encontrado, caso encontre ira renderizar a tela “delete” com a listagem de resultados do banco, se ocorrer um erro na busca de dados ira retornar o erro

Rota e lógica de exclusão(DELETE) de usuário específico

```
app.post('/delete/:cod_usuario', async (req, res) => {
  const { cod_usuario } = req.params;
  try {
    await pool.query('DELETE FROM usuario WHERE cod_usuario = $1', [cod_usuario]);
    res.redirect('/index');
  } catch (err) {
    console.error(err);
    res.send('Erro ao deletar usuário');
  }
});
```

Ao pressionar o botão para excluir, irá ser feito a exclusão do dado no banco com base no id que foi passado como parâmetro e retornará a página de listagem, caso capture um erro ira retornar o erro de deleção

Tela de exclusão do usuário



*ao marcar a checkbox e apertar o botão “deletar”, o usuário será excluído *

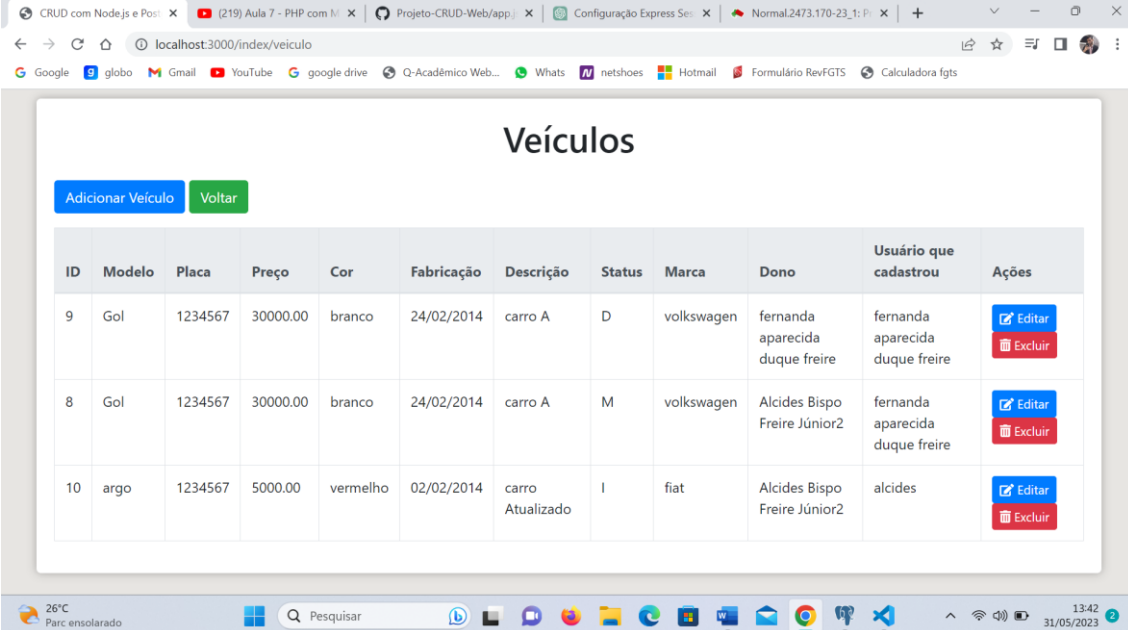
Página de listagem de veículos

```
async veiculoIndex(req, res) {
  try {
    const result = await pool.query(
      'SELECT v.cod_veiculo, v.placa, v.preco, v.cor, v.ano, v.descricao, v.status, ' +
      'mo.nome AS nomeModelo, m.nome AS nomeMarca, d.nome AS nomeDono, u.nome AS nomeUsuario ' +
      'FROM veiculo v, modelo mo, marca m, dono_veiculo d, usuario u ' +
      'WHERE v.cod_modelo = mo.cod_modelo ' +
      'AND mo.cod_marca = m.cod_marca ' +
      'AND v.cod_dono_veiculo = d.cod_dono_veiculo ' +
      'AND v.cod_usuario = u.cod_usuario '

    );

    res.render('veiculoIndex', { data: result.rows });
  } catch (err) {
    console.error(err);
    res.send('Erro ao buscar dados');
  }
},
```

Como esta é uma tabela com chave estrangeira, tive que fazer um select em campos específicos ao invés de usar *, renderizei o nome das tabelas de chave estrangeira e não o código



Veículos

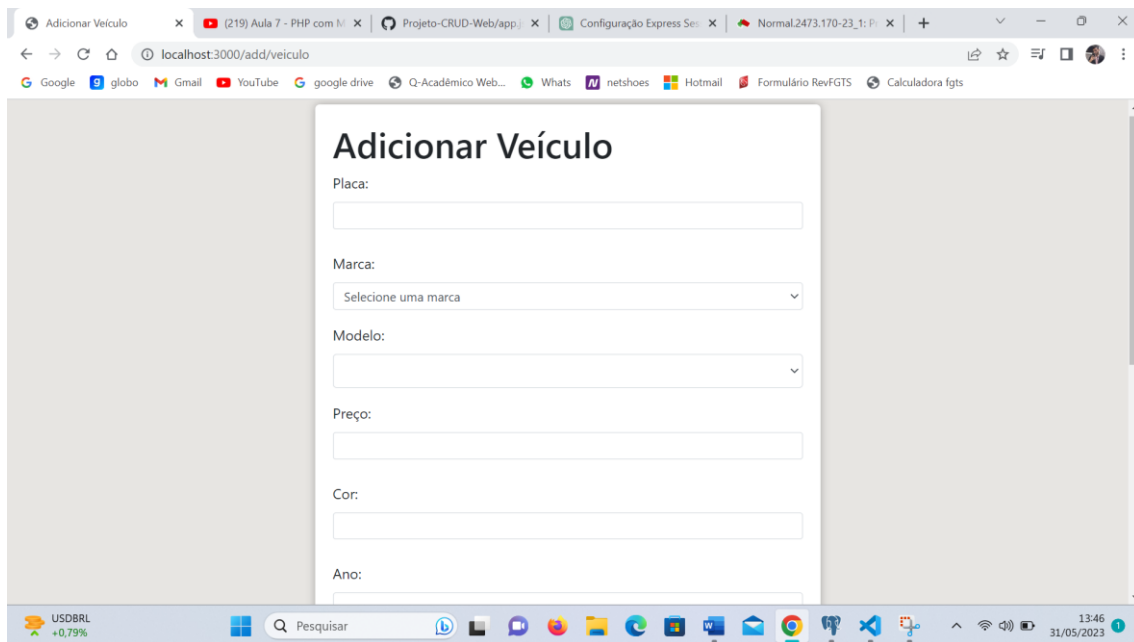
Adicionar Veículo Voltar

ID	Modelo	Placa	Preço	Cor	Fabricação	Descrição	Status	Marca	Dono	Usuário que cadastrou	Ações
9	Gol	1234567	30000.00	branco	24/02/2014	carro A	D	volkswagen	fernanda aparecida duque freire	fernanda aparecida duque freire	Editar Excluir
8	Gol	1234567	30000.00	branco	24/02/2014	carro A	M	volkswagen	Alcides Bispo Freire Júnior2	fernanda aparecida duque freire	Editar Excluir
10	argo	1234567	5000.00	vermelho	02/02/2014	carro Atualizado	I	fiat	Alcides Bispo Freire Júnior2	alcides	Editar Excluir

Neste trecho, eu faço um select para saber as marcas e os donos de veículo, o usuário eu já salvo no login e o campo de status eu defini nessas 4 opções(disponível, vendido, manutenção e indisponível), após isso, renderizo a pagina com todos os parâmetros

```
async paginaVeiculoAdd(req, res) {
  const statusOptions = ['D', 'V', 'M', 'I'];
  const marcas = await pool.query('SELECT * FROM marca');
  const usuario_nome = req.session.username;
  const dono_nome = await pool.query('SELECT * FROM dono_veiculo');

  res.render('veiculoAdd', { statusOptions, dono_nome: dono_nome.rows, marcas: marcas.rows, usuario_nome: usuario_nome });
},
```



Nesta página, fiz dois comboBox, ao selecionar a marca, já abre o comboBox dos modelos com base na marca selecionada, para isso fiz uma requisição AJAX(requisição HTTP assíncrona) para obter os modelos com base na marca selecionada e atualizar o campo de modelos dinamicamente sem precisar atualizar toda a página

Rota da minha requisição AJAX

```
async obterModelos(req, res) {
  try {
    const marcaSelecionada = req.query.marca;

    // Consulta ao banco de dados para obter o cod_marca da marca selecionada
    const marcaResult = await pool.query('SELECT cod_marca FROM marca WHERE nome = $1', [marcaSelecionada]);
    const codMarca = marcaResult.rows[0].cod_marca;

    // Consulta ao banco de dados para obter os modelos com base no cod_marca
    const modeloResult = await pool.query('SELECT * FROM modelo WHERE cod_marca = $1 ORDER BY nome', [codMarca]);
    const modelos = modeloResult.rows.map(row => row.nome); // Obtem apenas o nome dos modelos

    res.json({ modelos: modelos });
  } catch (error) {
    console.error('Erro ao obter modelos:', error);
    res.status(500).json({ error: 'Erro ao obter modelos' });
  }
},
```

Ela seleciona o código da marca com base no nome, e depois seleciona tudo do modelo com base no código da marca que é chave estrangeira da tabela modelo, retorna um JSON(arquivo de texto com dados estruturados) contendo o resultado do select.

Um trecho de código de como conseguir o código do modelo com base no nome selecionado no formulário

```
async adicionaVeiculo(req, res) {
  try {
    const { placa, preco, cor, ano, descricao, status, modelo, dono_nome, usuario_nome } = req.body;

    // Obter o código do modelo com base no nome selecionado
    const modeloSelecionado = await pool.query('SELECT cod_modelo FROM modelo WHERE nome = $1', [modelo]);
    if (modeloSelecionado.rowCount === 0) {
      return res.send('Modelo não encontrado');
    }
    const codModelo = modeloSelecionado.rows[0].cod_modelo;
```

Se não retornar nada no select, retorna um erro de modelo não encontrado, após isso salva o código do modelo em uma constante

Após fazer esse processo com os campos de chave estrangeira, ele insere na tabela os valores passados como parâmetros e retorna para a página de listagem, se ocorrer erro no insert retornará erro

```
    const result = await pool.query(
      'INSERT INTO veiculo (placa, preco, cor, ano, descricao, status, cod_modelo, cod_dono_veiculo, cod_usuario) VALUES ($1, $2, $3, $4, $5, $6, $7, $8, $9)',
      [placa, preco, cor, ano, descricao, status, codModelo, codDonoVeiculo, codUsuario]
    );

    res.redirect('/index/veiculo');
  } catch (err) {
    console.error(err);
    res.send('Erro ao adicionar dados');
  }
},
```

Renderiza a página de edição com os dados necessários

```
async paginaVeiculoEdit(req, res) {
  const statusOptions = ['D', 'V', 'M', 'I'];
  const { cod_veiculo } = req.params;
  const marcas = await pool.query('SELECT * FROM marca');
  const dono_nome = await pool.query('SELECT nome AS nomedono FROM dono_veiculo');
  const usuario_nome = req.session.username;

  try {
    const result = await pool.query(
      'SELECT v.*, m.nome AS nomeModelo, d.nome AS nomedono ' +
      'FROM veiculo v, modelo m, dono_veiculo d ' +
      'WHERE v.cod_modelo = m.cod_modelo ' +
      'AND v.cod_dono_veiculo = d.cod_dono_veiculo ' +
      'AND v.cod_veiculo = $1',
      [cod_veiculo]
    );

    res.render('veiculoEdit', { data: result.rows[0], marcas:marcas.rows, statusOptions, dono_nome:dono_nome.rows, usuario_nome:usuario_nome });
  } catch (err) {
    console.error(err);
    res.send('Erro ao buscar dados');
  }
},
```

Os valores vem preenchidos com os dados do campo selecionado para edição

Editar Veiculo

localhost:3000/edit/veiculo/9

Placa: 1234567

Modelo Atual: Gol

Marca: Selecione uma marca

Modelo:

Preço: 30000.00

Cor: branco

Ano: 24/02/2014

O modo de buscar os dados é semelhante ao modo de adição

```
async atualizaVeiculo(req, res) {
  const { cod_veiculo } = req.params;
  const { placa, preco, cor, ano, descricao, status, modelo, dono_nome, usuario_nome } = req.body;

  // Obter o código do modelo com base no nome selecionado
  const modeloSelecionado = await pool.query('SELECT cod_modelo FROM modelo WHERE nome = $1', [modelo]);
  if (modeloSelecionado.rowCount === 0) {
    return res.send('Modelo não encontrado');
  }
  const codModelo = modeloSelecionado.rows[0].cod_modelo;

  // ... resto do código ...
}
```

Método de UPDATE

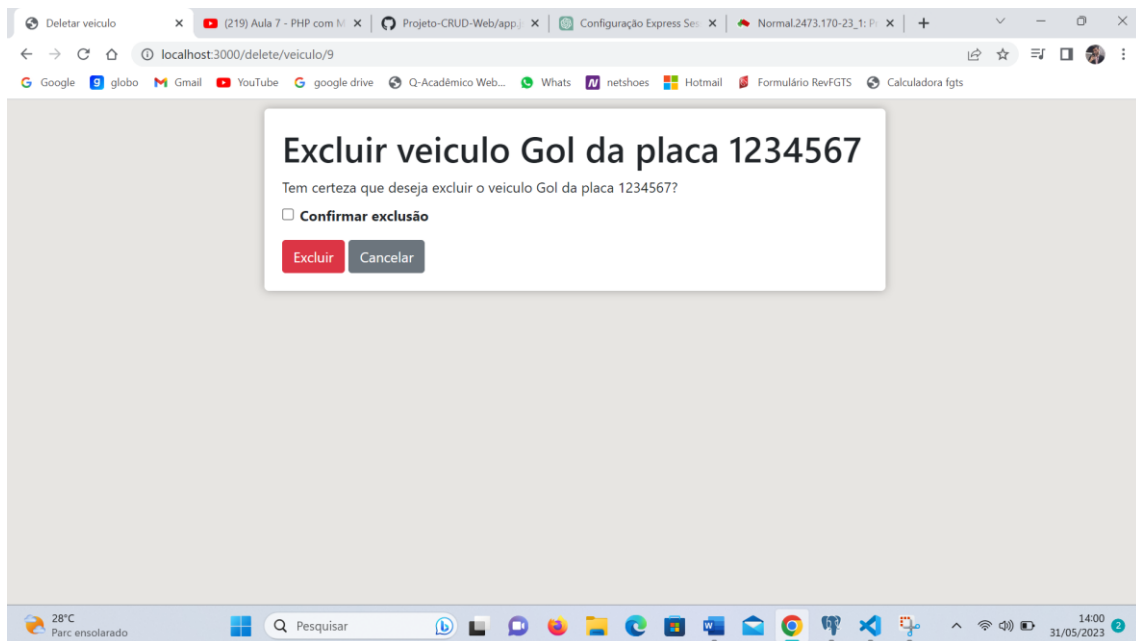
```
try {
  await pool.query(
    'UPDATE veiculo SET placa = $1, preco = $2, cor = $3, ano = $4, descricao = $5, status = $6, cod_modelo = $7, cod_dono_veic'
    [placa, preco, cor, ano, descricao, status, codModelo, codDonoVeiculo, codUsuario, cod_veiculo]
  );

  res.redirect('/index/veiculo');
} catch (err) {
  console.error(err);
  res.redirect('/edit/veiculo/${cod_veiculo}?msg=Erro ao atualizar dados');
},
```

Página de exclusão de veículo

```
async paginaVeiculoDelete(req, res) {
  const { cod_veiculo } = req.params;
  const nomeDoModelo = await pool.query(
    'SELECT v.*, m.nome AS nomeModelo ' +
    'FROM veiculo v, modelo m ' +
    'WHERE v.cod_modelo = m.cod_modelo ' +
    'AND v.cod_veiculo = $1',
    [cod_veiculo]
  );
};
```

Eu renderizo ela com alguns campos(tudo do veiculo e o nome do modelo) para melhorar na visualização do usuário



A página de exclusão mostra o nome do modelo e a placa para confirmar a exclusão do dado no banco

Método de exclusão de veículo, ele exclui comparando o código passado como parâmetro na tabela, retorna para a página de listagem e se capturar algum erro, retornará um erro

```
async deletaVeiculo(req, res) {  
  const { cod_veiculo } = req.params;  
  
  try {  
    await pool.query('DELETE FROM veiculo WHERE cod_veiculo = $1', [cod_veiculo]);  
    res.redirect('/index/veiculo');  
  } catch (err) {  
    console.error(err);  
    res.send('Erro ao deletar veículo');  
  }  
}  
};
```

projeto foi dividido em aproximadamente 17 páginas , superando 1900 linhas de código