

Atividade Final de Módulo: Implementação de Cadastro e Login de Usuários

Nome: Alana Vitória Maria da Silva

Curso de Extensão em Cloud

Módulo: Fundamentos em Nuvem

Diagrama do Banco de Dados:

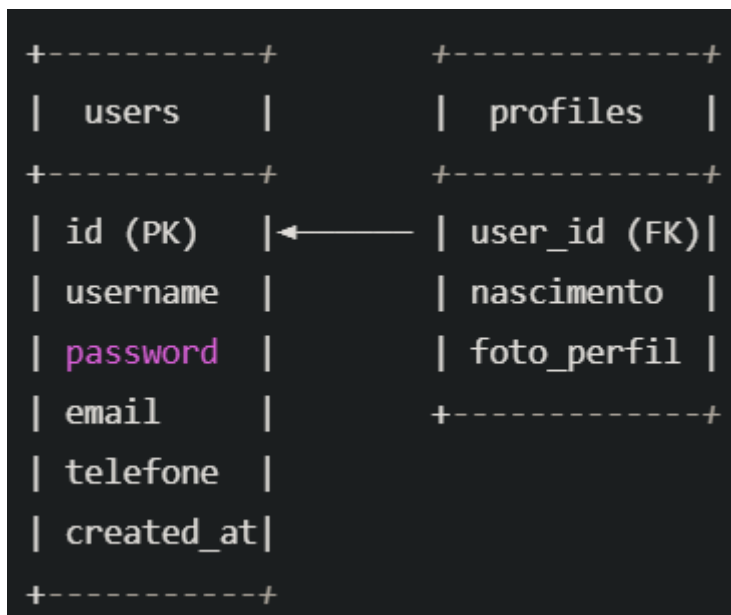


Diagrama de Classes (simples):

Class: User

- id
- username
- password
- email
- telefone
- created_at

Class: Profile

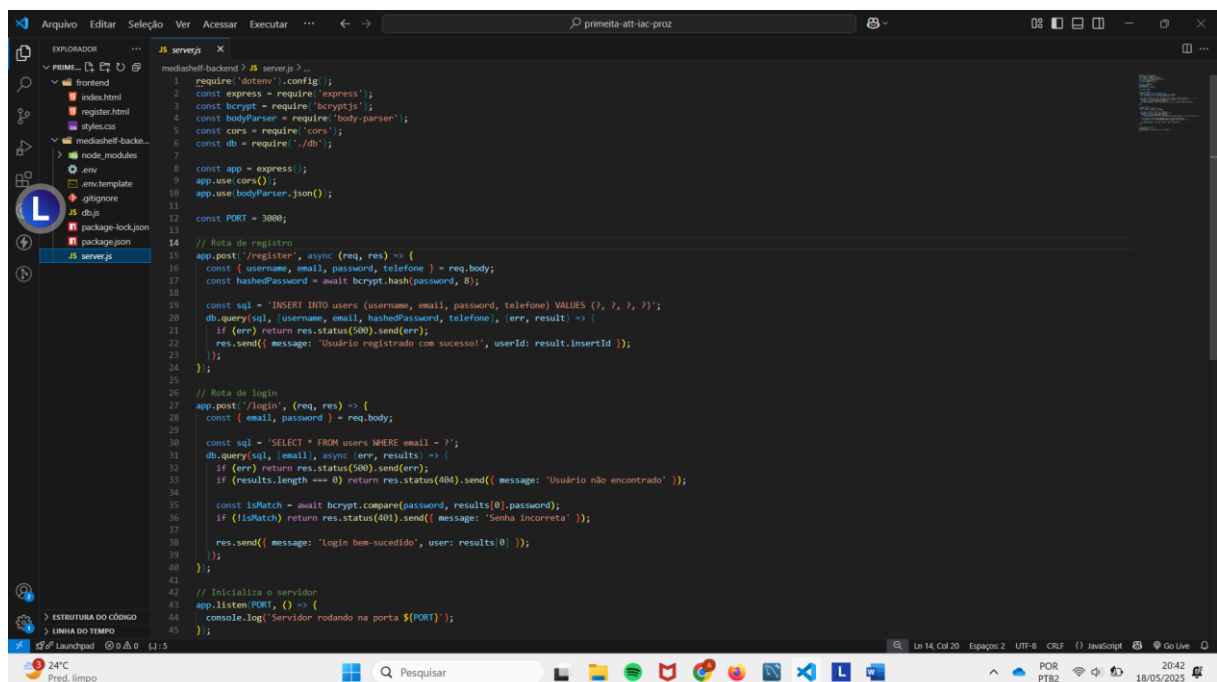
- id
- user_id
- data_nascimento
- foto_perfil

Manual do Usuário:

1. Acesse a página register.html
2. Preencha os campos de nome, e-mail, senha e telefone
3. Clique em "Cadastrar"
4. Após cadastro, vá para a página index.html
5. Faça login com seu e-mail e senha
6. Se os dados estiverem corretos, você verá a mensagem de sucesso

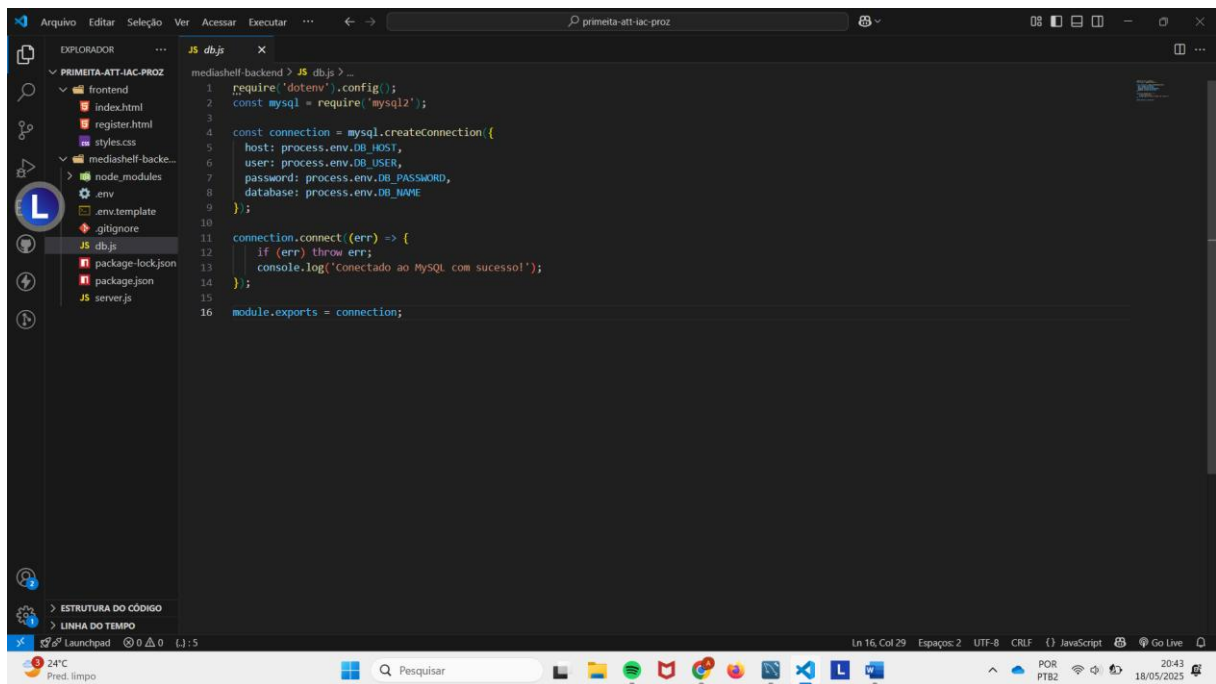
Implementação do Sistema:

server.js



```
1 require('dotenv').config();
2 const express = require('express');
3 const bcrypt = require('bcryptjs');
4 const bodyParser = require('body-parser');
5 const cors = require('cors');
6 const db = require('./db');
7
8 const app = express();
9 app.use(cors());
10 app.use(bodyParser.json());
11
12 const PORT = 3000;
13
14 // Rota de registro
15 app.post('/register', async (req, res) => {
16   const { username, email, password, telefone } = req.body;
17   const hashedPassword = await bcrypt.hash(password, 8);
18
19   const sql = 'INSERT INTO users (username, email, password, telefone) VALUES (?, ?, ?, ?)';
20   db.query(sql, [username, email, hashedPassword, telefone], (err, result) => {
21     if (err) return res.status(500).send(err);
22     res.send({ message: 'Usuário registrado com sucesso', userId: result.insertId });
23   });
24 });
25
26 // Rota de login
27 app.post('/login', (req, res) => {
28   const { email, password } = req.body;
29
30   const sql = 'SELECT * FROM users WHERE email = ?';
31   db.query(sql, [email], async (err, results) => {
32     if (err) return res.status(500).send(err);
33     if (results.length === 0) return res.status(404).send({ message: 'Usuário não encontrado' });
34
35     const isMatch = await bcrypt.compare(password, results[0].password);
36     if (!isMatch) return res.status(401).send({ message: 'Senha incorreta' });
37
38     res.send({ message: 'Login bem-sucedido', user: results[0] });
39   });
40 });
41
42 // Inicialize o servidor
43 app.listen(PORT, () => {
44   console.log(`Servidor rodando na porta ${PORT}`);
45 });
```

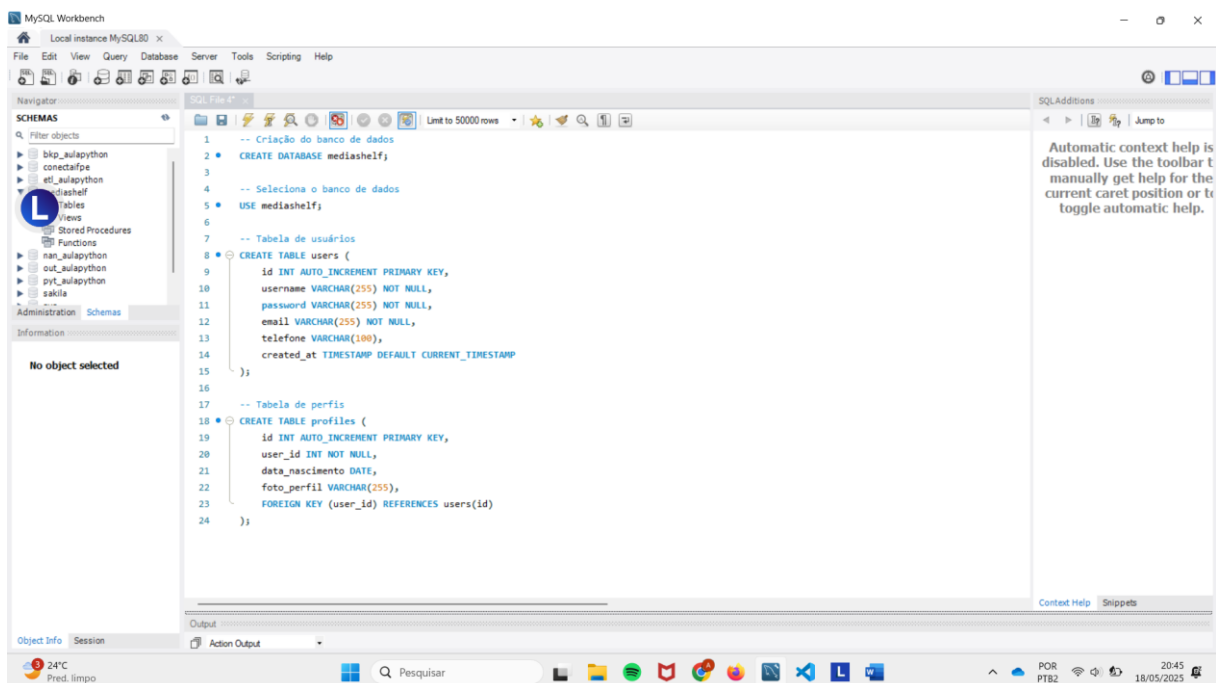
db.js



The screenshot shows a Visual Studio Code editor window with a dark theme. The Explorer sidebar on the left shows a project structure for 'PRIMEIRA-ATT-IAC-PROZ' with folders for 'frontend' and 'mediashelf-backend'. The 'db.js' file is selected in the Explorer and its content is displayed in the main editor. The code is a JavaScript file that uses the 'mysql' package to connect to a MySQL database. It sets up environment variables for host, user, password, and database, then creates a connection and logs a success message. The status bar at the bottom indicates the file is at line 16, column 29, using UTF-8 encoding and CRLF line endings.

```
1 require('dotenv').config();
2 const mysql = require('mysql2');
3
4 const connection = mysql.createConnection({
5   host: process.env.DB_HOST,
6   user: process.env.DB_USER,
7   password: process.env.DB_PASSWORD,
8   database: process.env.DB_NAME
9 });
10
11 connection.connect((err) => {
12   if (err) throw err;
13   console.log('Conectado ao MySQL com sucesso!');
14 });
15
16 module.exports = connection;
```

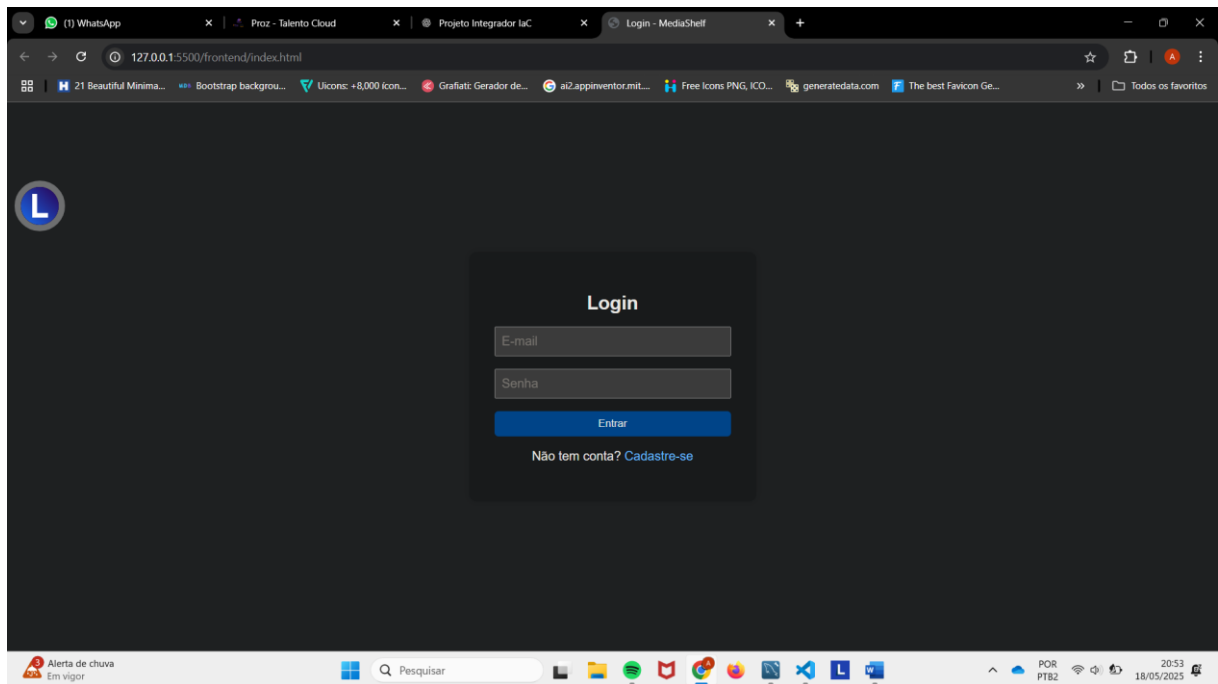
Banco MySQL:



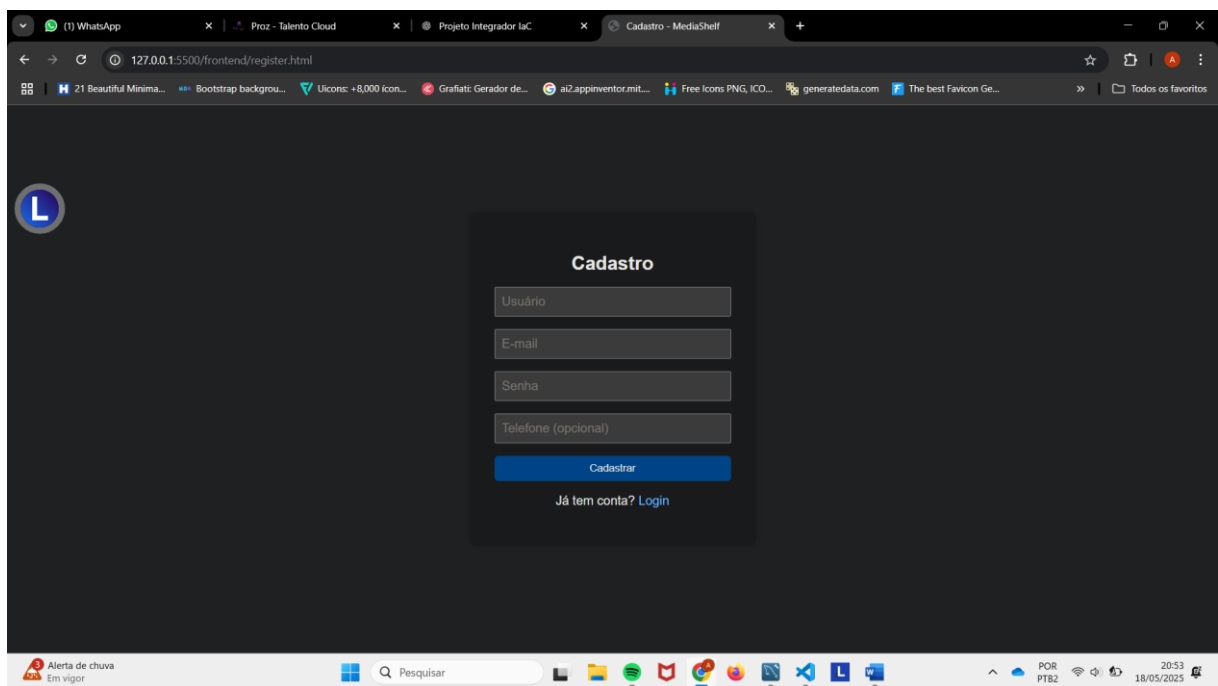
The screenshot shows the MySQL Workbench interface. The 'Schemas' sidebar on the left shows a list of databases, including 'bkp_alapaython', 'conexaofipe', 'etl_alapaython', 'mediashelf', 'nan_alapaython', 'out_alapaython', 'pyt_alapaython', and 'sakila'. The 'mediashelf' database is selected. The main editor displays an SQL script for creating a database and tables. The script includes comments in Portuguese and SQL code for creating the 'mediashelf' database, selecting it, and creating two tables: 'users' and 'profiles'. The 'users' table has columns for id, username, password, email, telefone, and created_at. The 'profiles' table has columns for id, user_id, data_nascimento, and foto_perfil, with a foreign key constraint on user_id. The status bar at the bottom indicates the file is at line 24, column 1, using UTF-8 encoding and CRLF line endings.

```
1 -- Criação do banco de dados
2 CREATE DATABASE mediashelf;
3
4 -- Seleciona o banco de dados
5 USE mediashelf;
6
7 -- Tabela de usuários
8 CREATE TABLE users (
9   id INT AUTO_INCREMENT PRIMARY KEY,
10  username VARCHAR(255) NOT NULL,
11  password VARCHAR(255) NOT NULL,
12  email VARCHAR(255) NOT NULL,
13  telefone VARCHAR(100),
14  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
15 );
16
17 -- Tabela de perfis
18 CREATE TABLE profiles (
19   id INT AUTO_INCREMENT PRIMARY KEY,
20   user_id INT NOT NULL,
21   data_nascimento DATE,
22   foto_perfil VARCHAR(255),
23   FOREIGN KEY (user_id) REFERENCES users(id)
24 );
```

Login:



Cadastro:



Terminal:

```
16 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

dope0@Alana_Vitoria MINGW64 ~/OneDrive - MSFT/Documents/primeita-att-iac-proz/mediashelf-backend
$ node server.js
Servidor rodando na porta 3000
Servidor rodando na porta
3000
Conectado ao MySQL com sucesso!

dope0@Alana_Vitoria MINGW64 ~/OneDrive - MSFT/Documents/primeita-att-iac-proz/mediashelf-backend
$ node server.js
Servidor rodando na porta 3000
Conectado ao MySQL com sucesso!
█
```