1 Projeto Laravel - Álbum de Fotos

1.1 Criando o Projeto

```
composer create-project laravel/laravel AlbumFotos
```

Após a criação do projeto, vamos instalar o NPM e habilitar o bootstrap:

Acessando o terminal na pasta criada para o projeto, execute:

- composer require laravel/ui - -dev
- php artisan ui bootstrap
- npm install
- npm run dev

1.2 Atualizar vite.config.js

É preciso alterar o arquivo *vite.config.js* e adicionar o caminho bootstrap 5 e remover resources/css/app.css

vite.config.js > default > plugins > input

```
import { defineConfig } from 'vite';
import laravel from 'laravel-vite-plugin';
import path from 'path'

export default defineConfig({
    plugins: [
        laravel({
            input: [
                'resources/js/app.js',
            ],
            refresh: true,
        }),
    ],
    resolve: {
        alias: {
            '~bootstrap': path.resolve(__dirname, 'node_modules/bootstrap'),
        }
    },
});
```

1.3 Importar Bootstrap 5 SCSS na pasta JS

É necessário importar o caminho SCSS bootstrap 5 em resources/js/app.js

resources > js > app.js

```
import './bootstrap';
import '../sass/app.scss'
```

1.4 Instalando jQuery

- npm install jquery - -save-dev

Depois de instalar o jQuery, precisamos importá-lo no arquivo resources/js/bootstrap.js.

1.5 Atualizar bootstrap.js

resources/js/bootstrap.js

resources > js > bootstrap.js

```
1  import * as Popper from '@popperjs/core'
2  window.Popper = Popper
3
4  import $ from 'jquery';
5  window.$ = $;
6
7  import 'bootstrap';
8
9  /**
10  * We'll load the axios HTTP library which allows us to easily issue requests
11  * to our Laravel back-end. This library automatically handles sending the
12  * CSRF token as a header based on the value of the "XSRF" token cookie.
13  */
14
15 import axios from 'axios';
16 window.axios = axios;
17
18 window.axios.defaults.headers.common['X-Requested-With'] = 'XMLHttpRequest';
19
```

1.6 Configurando *.env*

Edite o arquivo **.env** que está na raiz do seu projeto, informando: DB_DATABASE ; DB_USERNAME ; DB_PASSWORD

- DB_CONNECTION=mysql
- DB_HOST=127.0.0.1
- DB_PORT=3306
- DB_DATABASE=AlbumFotos
- DB_USERNAME=USUARIO_COM_PERMISSAO_DE_ACESSO
- DB_PASSWORD=SENHA_DO_USUARIO

2 Criando o Model

- php artisan make:model Foto -m

Edite o arquivo database » migrations » [timestamp]create_fotos_table, informando os atributos que deverão ser criados na tabela do BD.

```
1  public function up()
2  {
3    Schema::create('fotos', function (Blueprint $table) {
4        $table->bigIncrements('id');
5        $table->string('email');
6        $table->string('mensagem');
7        $table->string('arquivo');
8        $table->timestamps();
9    });
10 }
```

Execute o comando no terminal:

- $ php artisan migrate

O próximo passo será alterarmos nosso model acrescentando os atributos de nossas tabelas:

app/Models/**Foto.php**

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Foto extends Model
{
    use HasFactory;
    protected $fillable = ['email', 'mensagem', 'arquivo'];
}
```

3 Armazenamento dos arquivos que serão recebidos

Para armazenar as fotos que serão enviadas, vamos criar uma pasta **imagens** em: /**storage**/**app**/-**public**/ e em seguida criar o link para a pasta **public** do projeto:

- php artisan storage:link

4 Controller

- php artisan make:controller ControladorFoto -r

4.1 Implementando as funcionalidades do Controller

Primeiro devemos indicar o caminho para o Model a ser utilizado por esse controlador, e também utilizaremos as funcionalidades da classe **Storage**;

```php
use Illuminate\Support\Facades\Storage;
use App\Models\Foto;
```

Em seguida implementaremos o método **index**.

```php
public function index()
{
  $posts = Foto::all();
  return view('index', compact('posts'));
}
```

A implementação do método **store**.

este método será responsável pelo envio dos dados ao Model para inclusão no BD.

```php
public function store(Request $request)
{
  $path = $request->file('arquivo')->store('imagens', 'public');
  $post = new Foto();
  $post->email = $request->input('email');
  $post->mensagem = $request->input('mensagem');
  $post->arquivo = $path;
  $post->save();
  return redirect('/');
}
```

A implementação do método **destroy**.

```php
public function destroy($id)
{
  $post = Foto::find($id);
  if(isset($post)){
```

```
        $arquivo = $post->arquivo;
        Storage::disk('public')->delete($arquivo);
        $post->delete();
    }
    return redirect('/');
}
```

4.2 Criando nossa view

Criaremos a view principal do nosso sistema: em /resources/views **index.blade.php**

```
<!doctype html>
<html lang="pt-br">
<head>
    <meta name="viewport" content="width=device-width, initial-scale=1,
        shrink-to-fit=no">
    <meta charset="utf-8">
    <meta name="csrf-token" content="{{ csrf_token() }}">
    @vite(['resources/js/app.js'])
    <title>Álbum</title>
    <style>
        body { padding: 20px; }
        .navbar { margin-bottom: 20px; }
        :root { --jumbotron-padding-y: 10px; }
        .jumbotron {
          padding-top: var(--jumbotron-padding-y);
          padding-bottom: var(--jumbotron-padding-y);
          margin-bottom: 0;
          background-color: #fff;
        }
        @media (min-width: 768px) {
          .jumbotron {
            padding-top: calc(var(--jumbotron-padding-y) * 2);
            padding-bottom: calc(var(--jumbotron-padding-y) * 2);
          }
        }
        .jumbotron p:last-child { margin-bottom: 0; }
        .jumbotron-heading { font-weight: 300; }
        .jumbotron .container { max-width: 40rem; }
        .btn-card { margin: 4px; }
        .btn { margin-right: 5px; }
        footer { padding-top: 3rem; padding-bottom: 3rem; }
        footer p { margin-bottom: .25rem; }
    </style>
</head>
<body>

    <header>
      <div class="navbar navbar-dark bg-dark shadow-sm">
          <a href="#" class="navbar-brand d-flex align-items-center">
            <strong>Meu álbum de fotos</strong>
          </a>
      </div>
    </header>

    <main role="main">

      <section class="jumbotron text-center">
        <div class="container">
          <h1 class="jumbotron-heading">Escreva aqui seu novo Post</h1>
          <form method="POST" action="/" enctype="multipart/form-data">
```

```
50            @csrf
51            <div class="form-group text-left">
52              <label for="email">Endereço de e-mail</label>
53              <input type="email" class="form-control" id="email" name="
                    email" placeholder="nome@dominio.com">
54            </div>
55            <div class="custom-file">
56              <input type="file" class="custom-file-input" id="arquivo" name
                    ="arquivo">
57              <label class="custom-file-label" id="RotuloArquivo" for="
                    arquivo">Escolha um arquivo</label>
58            </div>
59            <div class="form-group text-left">
60              <label for="mensagem">Sua mensagem</label>
61              <textarea class="form-control" id="mensagem" name="mensagem"
                    rows="3"></textarea>
62            </div>
63            <p>
64              <button type="submit" class="btn btn-primary my-2">Enviar</
                    button>
65              <button type="reset" class="btn btn-secondary my-2">Cancelar</
                    button>
66            </p>
67          </form>
68        </div>
69      </section>
70
71      <div class="album py-5 bg-light">
72        <div class="container">
73          <div class="row">
74            @foreach($posts as $post)
75                <div class="col-md-4">
76                  <div class="card mb-4 shadow-sm">
77                  <img class="card-img-top figure-img img-fluid rounded" src
                        ="/storage/{{$post->arquivo}}">
78                    <div class="card-body">
79                      <p class="card-text">{{$post->email}}</p>
80                      <p class="card-text">{{$post->mensagem}}</p>
81                      <div class="d-flex justify-content-between align-items
                            -center">
82                        <div class="btn-group">
83                          <form method="POST" action="/{{$post->id}}">
84                            @csrf
85                            <input type="hidden" name="_method" value="
                                delete">
86                            <button type="submit" class="btn btn-sm btn-
                                outline-danger">Apagar</button>
87                          </form>
88                        </div>
89                      </div>
90                    </div>
91                  </div>
92                </div>
93            @endforeach
94          </div>
95        </div>
96      </div>
97
98    </main>
99
100   <footer class="text-muted">
```

```
      <div class="container">
        <p class="float-right">
          <a href="#">Voltar</a>
        </p>
        <p>©2023 CEFET-MG Unidade Varginha</p>
        <p>Técnico em Informática</p>
        <p>Aplicações Web II</p>
      </div>
    </footer>
</body>
</html>
```

4.3 Definindo nossas rotas

Em /routes está localizado o arquivo **web.php** que deve ser alterado.

```
<?php

use Illuminate\Support\Facades\Route;

Route::get('/', [App\Http\Controllers\ControladorFoto::class, 'index']);
Route::post('/', [App\Http\Controllers\ControladorFoto::class, 'store']);
Route::delete('/{id}', [App\Http\Controllers\ControladorFoto::class, '
   destroy']);
```