Justino Dos Santos Orlando Langa

Documentação da minha aplicação com alguns dos conceitos abordados na disciplina

1.Lazy Loading e Eager Loading

Implementei o Lazy Loading para otimização dos dados que são carregados apenas quando solicitados, e é ideal para economizar recursos de memória e melhorar a performance inicial da aplicação.

Implementei Eager Loading para carregar todos os dados relacionados de uma vez, minimizando o número de consultas ao banco de dados, é mais eficiente quando os dados relacionados são sempre necessários.

Como implementei:

```
$post = Post::find(1);

$comments = $post->comments;
```

2.Autenticação e Autorização (Policies)

Para autenticar os usuários, configurei um sistema baseado em JWT (JSON Web Token), ao fazer login, o usuário recebe um token seguro, que é utilizado para validar suas sessões, implementei a autenticação de dois fatores (2FA), utilizando bibliotecas que integram facilmente com aplicativos como Google Authenticator e para a autorização, utilizei Policies no Laravel para controlar o acesso a diferentes recursos da aplicação.

- **Exemplo de Implementação em Laravel:**

```
php artisan make:policy PostPolicy

public function update(User $user, Post $post)

{

   return $user->id === $post->user_id;

}
```

Definindo uma política:

```
php artisan make:policy PostPolicy

public function update(User $user, Post $post)

{

    return $user->id === $post->user_id;

}
```

3.Autenticação em dois factores (email via mailtrap)

Na minha aplicação, implementei a autenticação em dois fatores utilizando o envio de código por e-mail, configurado via Mailtrap, essa funcionalidade adiciona uma camada extra de segurança ao sistema e garante que apenas usuários autorizados tenham acesso às funcionalidades sensíveis.

- **Configuração do Mailtrap:**

Primeiro, configurei o Mailtrap no meu ambiente de desenvolvimento para capturar e-mails enviados pela aplicação, no arquivo .env, defini as variáveis necessárias para a conexão.

```
MAIL_MAILER=smtp

MAIL_HOST=smtp.mailtrap.io

MAIL_PORT=2525

MAIL_USERNAME=meu_usuario_mailtrap

MAIL_PASSWORD=minha_senha_mailtrap

MAIL_ENCRYPTION=null

MAIL_FROM_ADDRESS="no-reply@minhaapp.com"

MAIL_FROM_NAME="Minha Aplicação"
```

Para enviar a notificação:

```
$user->notify(new TwoFactorCode());
```

4.Comunicação via api

Na minha aplicação, implementei a comunicação entre diferentes serviços utilizando APIs. Isso foi essencial para integrar sistemas distintos e garantir que eles possam compartilhar dados de forma eficiente e segura.

1. **Criação da API na Aplicação**

Primeiro, desenvolvi endpoints RESTful na aplicação para expor os dados necessários. Utilizei o Laravel para criar essas rotas e controladores:

- **Exemplo de Rota API:**

Route::middleware('auth:sanctum')->get('/users', [UserController::class, 'index']);

Controlador para Manipular os Dados: No controlador, implementei os métodos para tratar requisições e retornar respostas no formato JSON:

```
namespace App\Http\Controllers;

use App\Models\User;

use Illuminate\Http\Request;

class UserController extends Controller

{

   public function index()

   {

      $users = User::all();

      return response()->json($users);

   }

}
```