12

Consolidando o seu conhecimento

Chegou a hora de você seguir todos os passos realizados por mim durante esta aula:

1) Na linha de comando, na pasta do projeto, execute o comando para instalar o `firebase/php-jwt`:

```
composer require firebase/php-jwt
```

2) Ainda na linha de comando, gere a *migration* para a tabela de usuários e gere o *seeder* para usuário:

```
php artisan make:migration criar_tabela_usuarios --create_usuarios
php artisan make:seeder UsuarioSeeder
```

3) Abra a *migration* gerada e altere o método `up` para adicionar as colunas `email` e `password`:

```
public function up()
{
    Schema::create('usuarios', function (Blueprint $table) {
        $table->bigIncrements('id');
        $table->string('email')->unique();
        $table->string('password');

        $table->timestamps();
    });
}
```

4) Na linha de comando, dentro do projeto, rode a *migration* para gerar a tabela do usuário:

```
php artisan migrate
```

5) Abra a classe `User` e adicione o atributo `$table`, para definir o nome da tabela:

```
protected $table = 'usuarios';
```

Além disso, tire o elemento `name` do array `$fillable` (deixe apenas `$email`).

6) Abra a classe `UsuarioSeeder`, para criar um usuário de teste. Altere o método `run` para:

```
public function run()
{
    \App\User::create([
        'email' => 'teste@email',
        'password' => Hash::make('senha')
    ]);
}
```

7) Abra a classe `DatabaseSeeder` para chamar o `UsuarioSeeder`. Altere apenas o método `run()`:

```
public function run()
{
    $this->call('UsuarioSeeder');
}
```

8) Na linha de comando, chame o *seeder* para inserir o usuário no banco:

```
php artisan db:seed
```

9) Agora, ajuste o provedor de autenticação. Para tal, abra a classe `AuthServiceProvider` e altere o método `boot()` para usar o cabeçalho `Authorization` e decodificar o *token* JWT.

O método completo ficará assim:

```
public function boot()
{
    // Here you may define how you wish users to be authenticated for your Lumen
    // application. The callback which receives the incoming request instance
    // should return either a User instance or null. You're free to obtain
    // the User instance via an API token or any other method necessary.

    $this->app['auth']->viaRequest('api', function (Request $request) {
        if (!$request->hasHeader('Authorization')) {
            return null;
        }
        $authorizationHeader = $request->header('Authorization');
        $token = str_replace('Bearer ', '', $authorizationHeader);
        $dadosAutenticacao = JWT::decode($token, env('JWT_KEY'), ['HS256']);

        return User::where('email', $dadosAutenticacao->email)
             ->first();
    });
}
```

10) Repare que foi usado um `JWT_KEY`. Para definir a chave, abra o arquivo `app/.env` e adicione uma nova propriedade:

```
JWT_KEY=uma_chave_muito_secreta123
```

11) Agora, habilite o *provider* e *facedes* do Laravel. Para tal, abra o arquivo `bootstrap/app.php` e descomente as seguintes linhas:

```
$app->withFacades();
```

E:

```
$app->register(App\Providers\AuthServiceProvider::class);
```

Além disso, ainda no arquivo `bootstrap/app.php`, descomente o *middleware* padrão:

```
$app->routeMiddleware([
    'auth' => App\Http\Middleware\Authenticate::class,
]);
```

12) Ainda falta habilitar o *middleware* no arquivo de rotas. Para tal, abra o arquivo `routes/web.php` e adicione `'middleware' => 'autenticador'` no primeiro grupo.

Para a sua comparação, a linha inteira ficará da seguinte forma:

```
$router->group(['prefix' => 'api', 'middleware' => 'autenticador'], function () use ($router) {
```

Ainda no mesmo arquivo, adicione a rota de login:

```
$router->post('/api/login', 'TokenController@gerarToken');
```

13) Por fim, crie um novo *controller*, chamado `TokenController`, na pasta `app/Http/Controllers`. Ele irá gerar o *token*:

```
<?php
namespace App\Http\Controllers;

use App\User;
use Firebase\JWT\JWT;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;

class TokenController extends Controller
{
    public function gerarToken(Request $request)
    {
        $this->validate($request, [
            'email' => 'required|email',
            'password' => 'required'
        ]);
        $usuario = User::where('email', $request->email)->first();

        if (is_null($usuario)
            || !Hash::check($request->password, $usuario->password)
        ) {
            return response()->json('Usuário ou senha inválidos', 401);
        }

        $token = JWT::encode(
            ['email' => $request->email],
            env('JWT_KEY')
        );

        return [
            'access_token' => $token
        ];
    }
```

```
}
```

14) Teste a geração do *token*, usando a URI `/api/login`, e enviando o e-mail e senha como JSON no corpo da requisição `POST`:

```
{
    "email" : "teste@email",
    "password": "senha"
}
```

Caso já tenha feito, excelente. Se ainda não, é importante que você execute o que foi visto nos vídeos para poder continuar com os próximos cursos que tenham este como pré-requisito.