05

Consolidando o seu conhecimento

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

- Através do console, dentro da pasta do seu projeto, instale os componentes **security**, **firebase/php-jwt** e **orm-fixtures**:

```
composer require security
composer require firebase/php-jwt
composer require orm-fixtures
```

- Ainda no console, através do **Maker**, crie um usuário, de nome `User`, salvando os seus dados no banco, com a propriedade `username` sendo única, e mantendo o padrão no restante:

```
php bin\console make:user
```

- Em seguida, crie a *migration*:

```
php bin\console make:migration
```

- E execute-as:

```
php bin\console doctrine:migrations:migrate
```

- Novamente através do **Maker**, crie uma *fixture*, de nome `UserFixtures`, salvando os seus dados no banco, com a propriedade `username` sendo única, e mantendo o padrão no restante:

```
php bin\console make:fixtures
```

- Agora, gere uma senha criptografada para o usuário, executando o comando abaixo. Digite a senha que você quiser, e a senha criptografada será retornada no campo ***Encoded password***:

```
php bin\console security:encode-password
```

- Abra a classe `UserFixtures`
  - No método `load`, instancie um `User`, atribua um *username* e uma senha (gerada acima) para ele
  - Em seguida, *persista* esse usuário e mande as alterações para o banco
- Volte ao console e, na pasta do projeto, rode as *fixtures* no banco de dados:

```
php bin\console doctrine:fixtures:load
```

- Em seguida, com o **Maker**, crie um *controller*, de nome `LoginController`:

```
php bin\console make:controller
```

- Abra a classe `LoginController`
  - Crie o seu construtor, injetando nele um `UserRepository` e um `UserPasswordEncoderInterface`, atribuindo-os a atributos privados da classe
  - No método `index`, receba o `Request` por parâmetro
  - Pegue o corpo da requisição, que será um JSON, e decodifique-a
  - Se o usuário ou a senha enviados no JSON forem nulos, retorne um `JsonResponse` com uma mensagem de erro para o cliente, com status HTTP 400
  - Busque o usuário no repositório e se ele e a sua senha não forem válidos, retorne um `JsonResponse` com uma mensagem de erro para o cliente, com status HTTP 401
  - Em seguida, crie o *token* para o usuário, usando o padrão **JWT**
  - Por último, retorne uma resposta em JSON com um array associativo do *token* criado acima

Caso já tenha feito, excelente. Se ainda não, é importante que você execute o que foi visto nos vídeos para poder continuar com a próxima aula.