04

Consolidando o seu conhecimento

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

- Crie a classe `JwtAutenticador`, que herdará da classe `AbstractGuardAuthenticator`
- Crie o seu construtor, injetando nele um `UserRepository` e atribua-o a um atributo privado da classe
- Alguns métodos deverão ser implementados
  - No método `supportsRememberMe`, retorne * `false`
  - No método `onAuthenticationSuccess`, retorne `null`
  - No método `onAuthenticationFailure`, retorne um `JsonResponse`, com uma mensagem de falha de autenticação para o cliente, mais o código HTTP 401
  - No método `supports`, verifique se o *path* do *request* é diferente de `/login` e retorne o booleano resultante disso
  - No método `getCredentials`, extraia o *token* do *request*, codifique-o e retorne-o, dentro de um `try-catch`, retornando `false` caso o *token* não possa ser codificado
  - No método `getUser`, se o parâmetro `credentials` não for um objeto ou se não existir a propriedade `username` dentro dele, retorne `null`. Em seguida, busque o usuário no seu repositório e retorne-o
  - No método `checkCredentials`, verifique se o parâmetro `credentials` é um objeto e se existe a propriedade `username` dentro dele. Retorne o booleano resultante disso
- Na pasta do seu projeto, abra o arquivo **config/packages/security.yaml**
  - Em **firewalls**, configure que não usuário anônimo, nem rota de *logout*. Além disso, configure o "guarda de rota" (**guard**), que utilizará o autenticador que você acabou de implementar:

```
firewalls:
dev:
    pattern: ^/(_(profiler|wdt)|css|images|js)/
    security: false
main:
    anonymous: ~
    logout: ~
    guard:
        authenticators:
            - App\Security\JwtAutenticador
```

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.