Guide d'Authentification :

Le fichier permettant de configurer l'authentification se situe dans config/packages/security.yaml. Vous trouverez plus d'informations concernant ce fichier et ses différentes parties dans la documentation https://symfony.com/doc/5.4/security.html#authenticating-users.

La documentation découpe la notion d'authentification en 5 points :

- L'entité User
- Les providers
- Les firewalls
- Les Access_control
- Les Role Hierachy

1) L'entité User :

Avant de commencer il est primordial d'avoir défini une entité qui représentera l'utilisateur connecté.
Par convention on appellera cette class « User ». Pour remplir son rôle, cette entité User doit étendre l'interface « UserInterface » et donc implémenter ses différentes méthodes. Ici on implémentera l'entité User dans src/Entity/User.php

2) Les providers :

- Pour nous authentifier, un user provider va nous permettre d'indiquer où se situe les informations concernant nos utilisateurs.
- Ici, ils seront stocké dans une base de données.
- De plus, dans notre cas, on récupèrera les utilisateurs via doctrine grâce à l'entité user dont la propriété username sera utilisé pour s'authentifier sur l'application. Ici, on peut indiquer la class User car celle-ci étend l'interface « UserInterface ».

```yaml
# https://symfony.com/doc/current/security.html#loading-the-user-the-user-provider
providers:
    # used to reload user from session & other features (e.g. switch_user)
    app_user_provider:
        entity:
            class: App\Entity\User
            property: username
```

3) Les firewalls :

- Un firewall va définir comment nos utilisateurs vont être authentifiés sur certaines parties du site. Le firewall dev ne concerne que le développement ainsi que le profiler et ne devra à priori ne pas être modifié.
- Le firewall main englobe l'entièreté du site à partir de la racine défini via pattern : ^/, on indique que c'est le userprovider qui sera utilisé pour s'authentifier.
- Afin de se connecter, on définit un formulaire de connexion via form_login : ou sont indiqués le

nom des routes correspondant au formulaire, la route de vérification du login ainsi que la route vers laquelle l'utilisateur devra être redirigé par défaut après son authentification.

```yaml
# config/packagers/security.yaml
    firewalls:
        dev:
            pattern: ^/(_(profiler|wdt)|css|images|js)/
            security: false
        main:
            lazy: true
            provider: app_user_provider
            form_login:
                login_path: app_login
                check_path: app_login
                always_use_default_target_path:  true
                default_target_path:  /
            logout:
                path: app_logout
```

4) Les Access Controll :

Un access_control va définir les limitations d'accès à certaines parties du site. Dans ce cas-ci, on indique que :
- l'url /login est accessible sans authentification.
- les url commençant par /users seront accessible que par les utilisateurs ayant un ROLE_ADMIN
- les url commençant par ^/ seront accessible par les utilisateurs ayant un ROLE_USER
Selon la situation, Il est a noté que le premier cas de figure correspondant sera pris en compte.

```yaml
    # Easy way to control access for large sections of your site
    # Note: Only the *first* access control that matches will be used
    access_control:
        - { path: ^/login, roles: PUBLIC_ACCESS }
        - { path: ^/users, roles: ROLE_ADMIN }
        - { path: ^/, roles: ROLE_USER }
```

5) Les Role Hierarchy

- Un role_hierarchy permet de s'assurer qu'un utilisateur ayant un certain rôle aura automatiquement d'autres rôles.
- Ici, un utilisateur possédant le rôle « ROLE_ADMIN » aura automatiquement le rôle « ROLE_USER »

```yaml
    role_hierarchy:
        ROLE_ADMIN: ROLE_USER
```