

# Authentication

1. Entité User	1
2. Composant security	1
2.1. Encoders	2
2.2. Providers	2
2.3. Firewalls	2
2.4. Access control	3
2.5. Role hierarchy	3

## 1. Entité User

Un utilisateur en base de données correspond à un objet de l'entité **User** (src/Entity/User).

On retrouvera donc une table appelée User en base de données.

Les propriétés de l'entité User correspondent aux champs de notre table User.

La classe **User** implémente l'interface **UserInterface** indispensable pour la création d'une classe utilisateurs et l'accès aux différentes fonctions liées à la gestion d'un utilisateur.

La classe **User** doit donc implémenter les méthodes présentes dans la classe **UserInterface**

Une contrainte d'unicité sur le champ **email** est définie en annotation dans l'entité.

```
/**
 * @ORM\Table("user")
 * @ORM\Entity
 * @UniqueEntity("email")
 * @ORM\Entity(repositoryClass="App\Repository\UserRepository")
 */
class User implements UserInterface
{
```

## 2. Composant security

"symfony/security-bundle": "5.2.\*"

Ce composant fournit les fichiers nécessaires et l'architecture pour sécuriser votre application.

(config/packages/security.yaml)

### 2.1. Encoders

Permet d'encoder les mots de passe des utilisateurs.

L'encodeur utilisé est **bcrypt**

```
security:
    encoders:
        App\Entity\User:
            algorithm: bcrypt
```

### 2.2. Providers

On déclare notre entité **User** et la propriété **username** en tant que provider.

On déclare également doctrine en tant qu'ORM pour la connexion à la base de données.

```
providers:
    doctrine:
        entity:
            class: App\Entity\User
            property: username
```

### 2.3. Firewalls

Le pare-feu nous permet de restreindre ou autoriser l'accès à certaines routes.

Ici

- dev
  - permet d'autoriser le chargement des pages d'assets et template du site ainsi que la barre de debug Symfony.
- main
  - indique que les utilisateurs anonymes peuvent arriver sur le chemin '/', et seront redirigés vers la page de login.
  - définit le nom de la route vers la page de login et de vérification d'authentification

```

firewalls:
  dev:
    pattern: ^/(_(profiler|wdt)|css|images|js)/
    security: false

main:
  pattern: ^/
  anonymous: ~
  form login:
    login_path: login
    check_path: login_check
    always_use_default_target_path: true
    default_target_path: /
  logout: ~

```

## 2.4. Access control

Dans cette partie vous avez la possibilité de configurer les accès aux différentes routes selon les **rôles** de vos utilisateurs

Ici

- les anonymes ont le droit d'accéder à la page de login
- les utilisateurs ayant le rôle **ROLE\_ADMIN** uniquement pourront accéder à la route **/users**
- les utilisateurs ayant le rôle **ROLE\_USER** pourront accéder à toutes les autres routes.

```

access_control:
- { path: ^/login, roles: IS_AUTHENTICATED_ANONYMOUSLY }
- { path: ^/users, roles: ROLE_ADMIN }
- { path: ^/, roles: ROLE_USER }

```

## 2.5. Role hierarchy

On définit ici la hiérarchie des rôles.

Ici, on définit que **ROLE\_ADMIN** possède tous les droits de **ROLE\_USER**.

```

role_hierarchy:
  ROLE_ADMIN: ROLE_USER

```