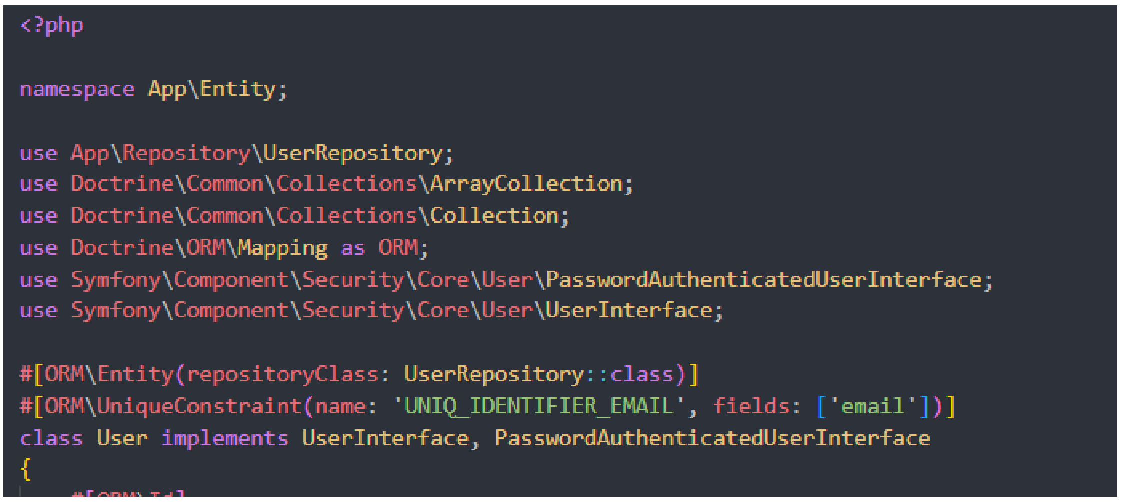
Implémentation of authentification

User entity

The user is represented by the AppBundle Entity User class. A uniqueness constraint is applied to the email attribute so as not have duplicates.



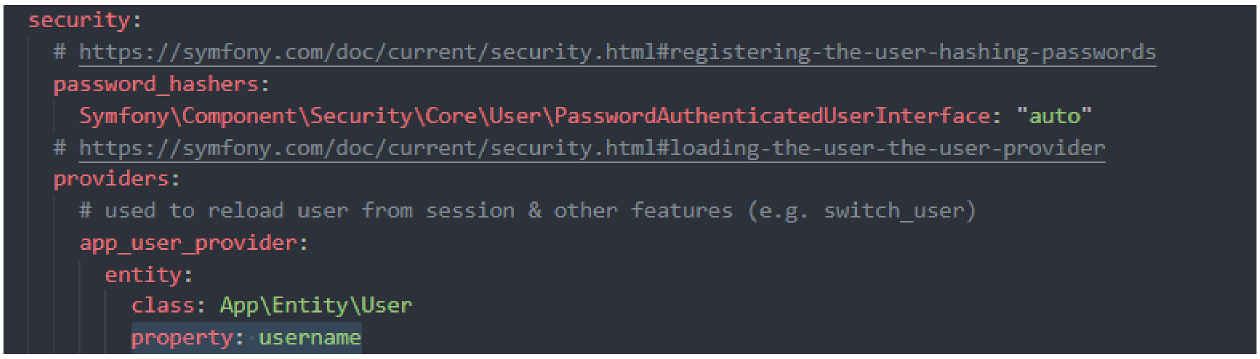
```php
<?php

namespace App\Entity;

use App\Repository\UserRepository;
use Doctrine\Common\Collections\ArrayCollection;
use Doctrine\Common\Collections\Collection;
use Doctrine\ORM\Mapping as ORM;
use Symfony\Component\Security\Core\User\PasswordAuthenticatedUserInterface;
use Symfony\Component\Security\Core\User\UserInterface;

#[ORM\Entity(repositoryClass: UserRepository::class)]
#[ORM\UniqueConstraint(name: 'UNIQ_IDENTIFIER_EMAIL', fields: ['email'])]
class User implements UserInterface, PasswordAuthenticatedUserInterface
{
```

The security file

Secutity regarding authentification is configured in the file config/packages/security.yaml . For more informations https://symfony.com/doc/6.4/security.html

Providers

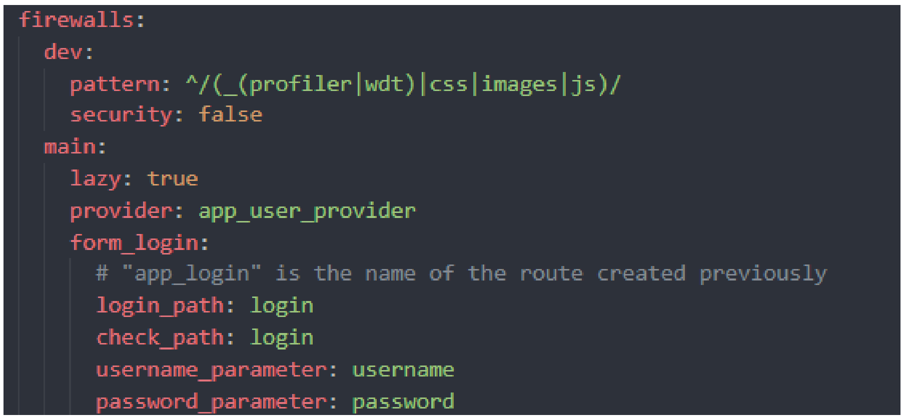
A provider will allow us to indicate where is the information that we want to use to authenticate the user, we say that we will retrieve users through the entity User whose property username will be used to authenticate on the site.



```yaml
security:
  # https://symfony.com/doc/current/security.html#registering-the-user-hashing-passwords
  password_hashers:
    Symfony\Component\Security\Core\User\PasswordAuthenticatedUserInterface: "auto"
  # 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
```

Firewall

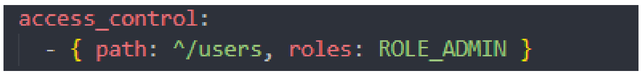
A firewall is designated to prevent an unauthenticated user from accessing certain parts of the site. To authenticate, a form accessible to the login route is used:



```yaml
firewalls:
  dev:
    pattern: ^/(_(profiler|wdt)|css|images|js)/
    security: false
  main:
    lazy: true
    provider: app_user_provider
    form_login:
      # "app_login" is the name of the route created previously
      login_path: login
      check_path: login
      username_parameter: username
      password_parameter: password
```

Securing URL patterns (access_control)

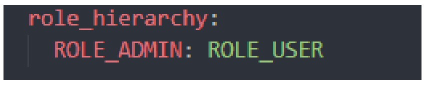
The most basic way to secure part of your app is to secure an entire URL pattern in security.yaml.



```yaml
access_control:
  - { path: ^/users, roles: ROLE_ADMIN }
```

Hierarchical Roles

Instead of giving many roles to each user, you can define role inheritance rules by creating a role hierarchy:



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