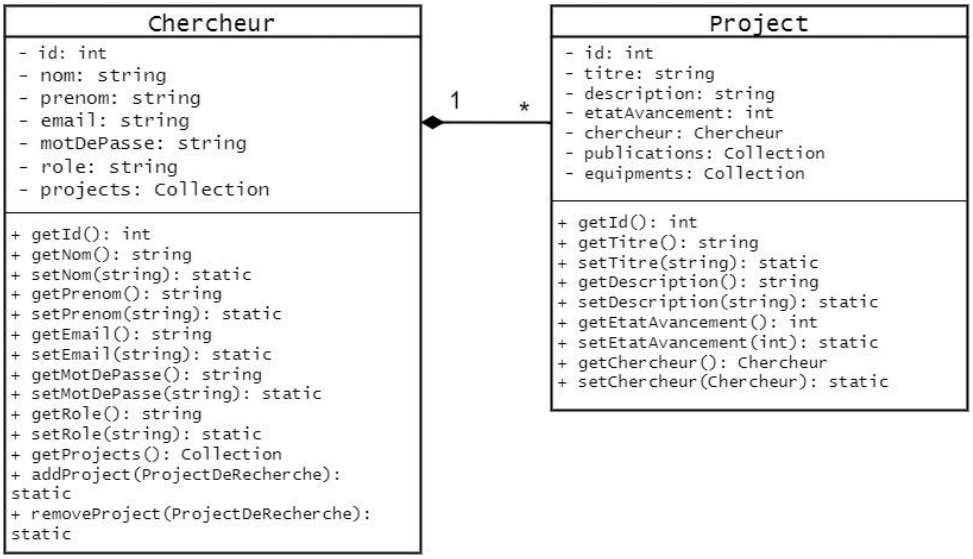
Configuration de la base de données

DATABASE_URL="mysql://root:@127.0.0.1:3306/miniProjet?serverVersion=8.0.32&charset=utf8mb4"

Create a new database as defined in your Symfony project configuration

Symfony console doctrine:database:create



Create a new user entity for the 'chercheur' role

Symfony console make:user chercheur

Generate an entity class for managing projects

Symfony console make:entity project

Create a new migration based on changes in your entity classes

Symfony console make:migration

Apply pending database migrations to synchronize the database schema

Symfony console doctrine:migrations:migrate

Establish a One-to-Many relationship, indicating that 'Project' can have multiple related entities named 'Projects'

Symfony console make:entity project

Projects / relation / Project / OneToMany

Symfony console make :fixture

```
class AppFixtures extends Fixture
{
 for ($i = 1; $i < 6; $i++) {
           $chercheur = new Chercheur();
           $chercheur->setNom($faker->lastName);
           $chercheur->setPrenom($faker->firstName);
           $chercheur->setEmail("chercheur".$i."@gmail.com");
           $plaintextPassword= "chercheurPass";
            // hash the password
            $hashedPassword = $this->passwordHasher->hashPassword(
               $chercheur,
               $plaintextPassword
            );
           $chercheur->setMotDePasse($plaintextPassword);

             for ($i = 1; $i < 5; $i++) {
              $projects = [];

               $projet = new ProjectDeRecherche();
               $projet->setTitre($faker->word());
               $projet->setDescription($faker->words(10, true));
               $projet->setEtatAvancement(random_int(0, 100));
               $projet->setChercheur($chercheur);

              $manager->persist($projet);

           foreach ($projects as $project) {
               $chercheur->addProject($project);
           }


           $manager->persist($chercheur);

}
       $manager->flush();

}
```

Load sample data fixtures into the database for testing or seeding initial data

Symfony console doctrine:load:fixture

Symfony console make:crud chercheur

Symfony console make:crud project

```
    #[Route('/{id}', name: 'app_chercheur_delete', methods: ['POST'])]
    public function delete(Request $request, Chercheur $chercheur,
EntityManagerInterface $entityManager): Response
    {
            // Now delete the Chercheur entity
            $entityManager->remove($chercheur);
            $entityManager->flush();

        return $this->redirectToRoute('app_chercheur_index', []);
    }
```

```
#[Route('/new', name: 'app_chercheur_new', methods: ['GET', 'POST'])]
public function new(Request $request, EntityManagerInterface $entityManager):
Response
{
    // Create a new Chercheur instance
    $chercheur = new Chercheur();
    // Create a form for Chercheur using ChercheurType form type
    $form = $this->createForm(ChercheurType::class, $chercheur);
    // Handle the form submission
    $form->handleRequest($request);
    // Check if the form is submitted and valid
    if ($form->isSubmitted() && $form->isValid()) {
        // Persist the new Chercheur entity to the database
        $entityManager->persist($chercheur);
        $entityManager->flush();

        // Add a flash message to indicate success
        $this->addFlash('success', 'Chercheur added successfully!');

        // Redirect to the index page after successful creation
        return $this->redirectToRoute('app_chercheur_index', []);
    }
    // Render the new Chercheur form
    return $this->render('chercheur/new.html.twig', [
        'chercheur' => $chercheur,
        'form' => $form,
    ]);
}
```

```php
class ChercheurType extends AbstractType
{
    public function buildForm(FormBuilderInterface $builder, array $options):
void
    {
        $builder
            ->add('nom',TextType::class,[
                'attr'=>[
                'class' =>'form-control',
                'id'=>'searchPublication',
                ]
            ])
            ->add('prenom',TextType::class,[
                'attr'=>[
                'class' =>'form-control',
                'id'=>'searchPublication',
                ]
            ])
            ->add('email',TextType::class,[
                'attr'=>[
                'class' =>'form-control',
                'id'=>'searchPublication',
                ]
            ])
            ->add('motDePasse',TextType::class,[
                'attr'=>[
                'class' =>'form-control',
                'id'=>'searchPublication',
                ]
            ])
            ->add('role',TextType::class,[
                'attr'=>[
                'class' =>'form-control',
                'id'=>'searchPublication',
                ]
            ])

            ->setAttributes([
                'class' => 'signup-form',
                 // Add your desired form class here
            ]);
    }
```

```php
#[Route('/{id}/edit', name: 'app_chercheur_edit', methods: ['GET', 'POST'])]
public function edit(Request $request, Chercheur $chercheur,
EntityManagerInterface $entityManager): Response
{
    // Create a form for Chercheur using ChercheurType form type and bind it to
the existing Chercheur entity
    $form = $this->createForm(ChercheurType::class, $chercheur);

    // Handle the form submission
    $form->handleRequest($request);

    // Check if the form is submitted and valid
    if ($form->isSubmitted() && $form->isValid()) {
        // Persist the updated Chercheur entity to the database
        $entityManager->flush();

        // Redirect to the index page after successful edit
        return $this->redirectToRoute('app_chercheur_index', []);
    }

    // Render the edit Chercheur form
    return $this->render('chercheur/edit.html.twig', [
        'chercheur' => $chercheur,
        'form' => $form,
    ]);
}
```

```php
#[Route('/{id}', name: 'app_chercheur_show', methods: ['GET'])]
public function show(Chercheur $chercheur): Response
{
    // Render the show Chercheur page
    return $this->render('chercheur/show.html.twig', [
        'chercheur' => $chercheur,
    ]);
}
```

chercheurProjects.html.twig

```
{% extends 'base.html.twig' %}
  {% block content %}
  <h1>Projects Chercheur</h1>

  {% for proj in project %}
    {% if idChercheur ==  proj.chercheur.getId() %}
      <ul>
        <li>{{ proj.titre }}</li>
        <li>{{ proj.description }}</li>
        <li>{{ proj.etatAvancement }}</li>
      </ul>
    {% endif %}
  {% endfor %}

  {% endblock %}
{% endblock %}
```

Base.html.twig

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>
            {% block title %}Welcome!
            {% endblock %}
        </title>
        {% block stylesheets %}
            <link href="{{asset('bootstrap/css/bootstrap.min.css')}}"
rel="stylesheet">
        {% endblock %}

    </head>
    <body>
        {% block body %}
            {% block content %} <a href="{{path('app_chercheur_index')}}">Home
Page</a>{% endblock %}

        {% endblock %}
    </body>
    {% block javascripts %}    {% endblock %}
        {% block footer %}  {% endblock %}
</html></body></html>
```

```php
<?php

namespace App\Controller;
use App\Form\ChercheurType;
use App\Entity\Chercheur;
use Doctrine\ORM\EntityManagerInterface;
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\Routing\Annotation\Route;
use Symfony\Component\Security\Core\Encoder\UserPasswordHasherInterface; //
Use the correct interface

class ChercheursController extends AbstractController


{

    private $em;
    public function __construct(EntityManagerInterface $em){
        $this->em = $em;
    }
    #[Route('/chercheurs', name: 'app_chercheurs')]
    public function index(): Response
    {
        $chercheurs = $this->em->getRepository(Chercheur::class)->findAll();

        return $this->render('chercheurs/index.html.twig', [
            'controller_name' => 'ChercheursController',
            'chercheurs' => $chercheurs,
        ]);
    }


    #[Route('/register', name: 'app_register')]
    public function register(Request $request, EntityManagerInterface
$entityManager): Response
    {
        $chercheur = new Chercheur();
        $form = $this->createForm(ChercheurType::class, $chercheur);

        $form->handleRequest($request);
```

```
        if ($form->isSubmitted() && $form->isValid()) {
            // Persist the user without hashing the password
            $entityManager->persist($chercheur);
            $entityManager->flush();

            // Redirect to login page or any other route
            return $this->redirectToRoute('app_login');
        }

        return $this->render('chercheurs/register.html.twig', [
            'form' => $form->createView(),
        ]);
    }


    #[Route('/create-chercheur', name: 'create-chercheur')]

    public function createChercheur(Request $request){
        $chercheur=new Chercheur();
        $form = $this->createForm(ChercheurType::class, $chercheur);
        $form->handleRequest( $request );
        if($form->isSubmitted() && $form->isValid()){
            $this->em->persist($chercheur);
            $this->em->flush();
            $this->addFlash('success', 'Chercheur created successfully.');

        }
        return $this->render('Chercheurs/chercheur.html.twig', ['form' =>
$form->createView(), 'chercheur' => $chercheur]);
        }
        #[Route('/edit-chercheur/{id}', name: 'edit-chercheur')]
    public function editChercheur(Request $request, Chercheur $chercheur)
    {
        $form = $this->createForm(ChercheurType::class, $chercheur);
        $form->handleRequest($request);

        if ($form->isSubmitted() && $form->isValid()) {
            $this->em->flush();
            $this->addFlash('success', 'Chercheur updated successfully.');


            return $this->redirectToRoute('app_chercheurs');
        }
```

```
        return $this->render('chercheurs/edit_chercheur.html.twig', ['form'
=> $form->createView(), 'chercheur' => $chercheur,]);
    }

    #[Route('/delete-chercheur/{id}', name: 'delete-chercheur')]
    public function deleteChercheur(Chercheur $chercheur)
    {
        $this->em->remove($chercheur);
        $this->em->flush();
        $this->addFlash('success', 'Chercheur Deleted successfully.');


        return $this->redirectToRoute('app_chercheurs');
    }
}
```