Compte Rendu TP07

1- Recherche Article par Nom :

1.1 Création de la classe PropertySearch.php qui a comme attribut le nom dans Entity avec le code suivant

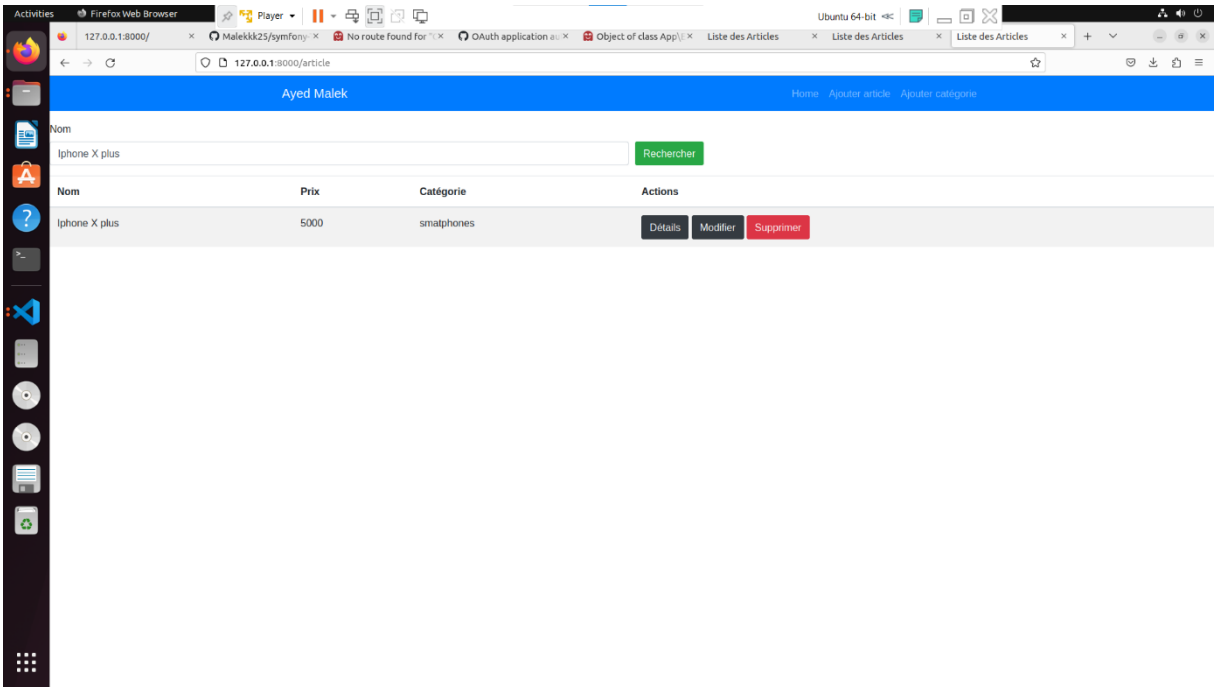
```php
<?php
namespace App\Entity;

class PropertySearch
{
 private $nom;
 public function getNom(): ?string
 {
 return $this->nom;
 }
 public function setNom(string $nom): self
 {
 $this->nom = $nom;
 return $this;
 }
}
```

1.2 Création de la formulaire PropertySearchType basé sur l'entité PropertySearch avec la commande **php bin/console make:form**
En utilisant comme entité **/App/Entity/ PropertySearch** puisqu'on la créer manuellement

1.3 Faire le filtrage par nom des articles en modifiant la fonction home de IndexController.php qui d'après le formulaire PropertySearchType prend le nom et retourne la liste des articles comme suit :

```php
 }
 public function home(Request $request)
{
$propertySearch = new PropertySearch();
$form = $this->createForm(PropertySearchType::class,$propertySearch);
$form->handleRequest($request);
$articles= [];
if($form->isSubmitted() && $form->isValid()) {
$nom = $propertySearch->getNom();
if ($nom!="")
$articles= $this->entityManager->getRepository(Article::class)->findBy(['nom' => $nom] );
else
$articles= $this->entityManager->getRepository(Article::class)->findAll();
}
return $this->render('articles/index.html.twig',[ 'form' =>$form->createView(), 'articles' => $articles]);
}
```



2- Recherche des articles dont le prix est compris entre deux valeurs

2.1 Créer la classe PriceSearch qui a comme attribut minPrice et maxPrice dans le dossier entity comme suit :

```php
<?php
namespace App\Entity;
class PriceSearch
{
 /**
 * @var int|null
 */
 private $minPrice;
 /**
 * @var int|null
 */
 private $maxPrice;

 public function getMinPrice(): ?int
 {
 return $this->minPrice;
 }
 public function setMinPrice(int $minPrice): self
 { $this->minPrice = $minPrice;
 return $this;
 }
 public function getMaxPrice(): ?int
 {
 return $this->maxPrice;
 }
 public function setMaxPrice(int $maxPrice): self
 {
 $this->maxPrice = $maxPrice;
 return $this;
 }
}
```

2.2 Créer le formulaire PriceSearchType basé sur l'entité \App\Entity\PriceSearch avec la commande **php bin/console make:form**

```php
<?php

namespace App\Form;

use App\Entity\PriceSearch;
use Symfony\Component\Form\AbstractType;
use Symfony\Component\Form\FormBuilderInterface;
use Symfony\Component\OptionsResolver\OptionsResolver;

class PriceSearchType extends AbstractType
{
    public function buildForm(FormBuilderInterface $builder, array $options): void
    {
        $builder
            ->add('minPrice')
            ->add('maxPrice')
        ;
    }

    public function configureOptions(OptionsResolver $resolver): void
    {
        $resolver->setDefaults([
            'data_class' => PriceSearch::class,
        ]);
    }
}
```

2.3 Ajouter la fonction findByPriceRange à la classe Repository/ArticleRepository qui accède à la base et trouve les articles avec le cout entre la valeur minimale et maximale puis fait l'ordre de cette liste en ordre croissant.

```php
public function findByPriceRange($minValue,$maxValue)
{
return $this->createQueryBuilder('a')
->andWhere('a.prix >= :minVal')
->setParameter('minVal', $minValue)
->andWhere('a.prix <= :maxVal')
->setParameter('maxVal', $maxValue)
->orderBy('a.id', 'ASC')
->setMaxResults(10)
->getQuery()
->getResult()
;
}
```

2.4 Ajouter la fonction ArticleParPrix dans IndexController.php qui permet d'après le formulaire PriceSearchType de prendre la valeur minimale et maximale rendre la liste des articles avec la fonction findByPriceRange du repository

```php
public function articlesParPrix(Request $request)
{

$priceSearch = new PriceSearch();
$form = $this->createForm(PriceSearchType::class,$priceSearch);
$form->handleRequest($request);
$articles= [];
if($form->isSubmitted() && $form->isValid()) {
$minPrice = $priceSearch->getMinPrice();
$maxPrice = $priceSearch->getMaxPrice();

$articles= $this->entityManager->getRepository(Article::class)->findByPriceRange($minPrice,$maxPrice);
}
return $this->render('articles/articlesParPrix.html.twig',[ 'form' =>$form->createView(), 'articles' => $articles]);
}
}
```