Compte Rendu TP04

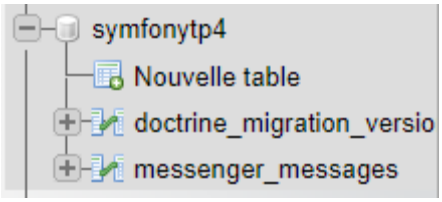
Bases de données :

1. Ajouter notre base dans le fichier. env

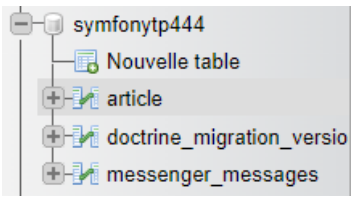
```
#
# DATABASE_URL="sqlite:///%kernel.project_dir%/var/data.db"
# DATABASE_URL="mysql://app:!ChangeMe!@127.0.0.1:3306/app?serverVersion=8&charset=utf8mb4"
DATABASE_URL="postgresql://app:!ChangeMe!@127.0.0.1:5432/app?serverVersion=15&charset=utf8"
DATABASE_URL=mysql://root:@localhost:3306/symfonyTP444?serverVersion=mariadb-10.4.11
```

2. Créer la base de données en tapant la commande : **php bin/console doctrine:database:create**

3. Base créer dans phpMyAdmin



4. Créer l'entité Article avec la commande :**php bin/console make:entity Article**
5. Faire la migration avec les commandes : **php bin/console doctrine:migrations:diff** et **php bin/console doctrine:migrations:migrate**
6. Table Article créer



Opération CRUD :

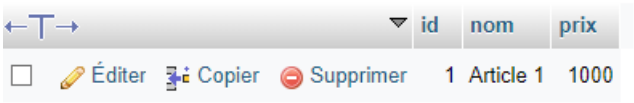
1. Ajouter la méthode save dans le fichier IndexController.php

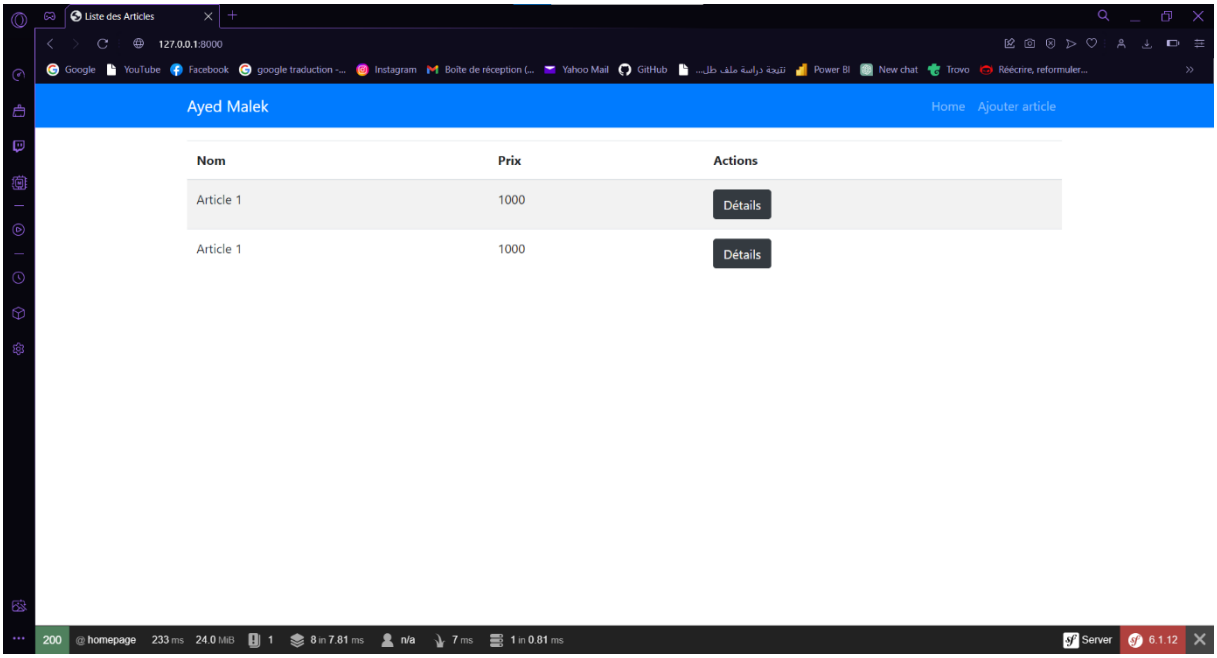
```php
public function save() {
    $entityManager = $this->entityManager->getManager();
    $article = new Article();
    $article->setNom('Article 1');
    $article->setPrix(1000);

    $entityManager->persist($article);
    $entityManager->flush();
    return new Response('Article enregisté avec id '.$article->getId());
    }
```

1.1 Ajouter le routage dans le fichier routes.yaml

```yaml
save_article:
 path: /article/save
 controller: App\Controller\IndexController::save
```

1.2 Tester le travail





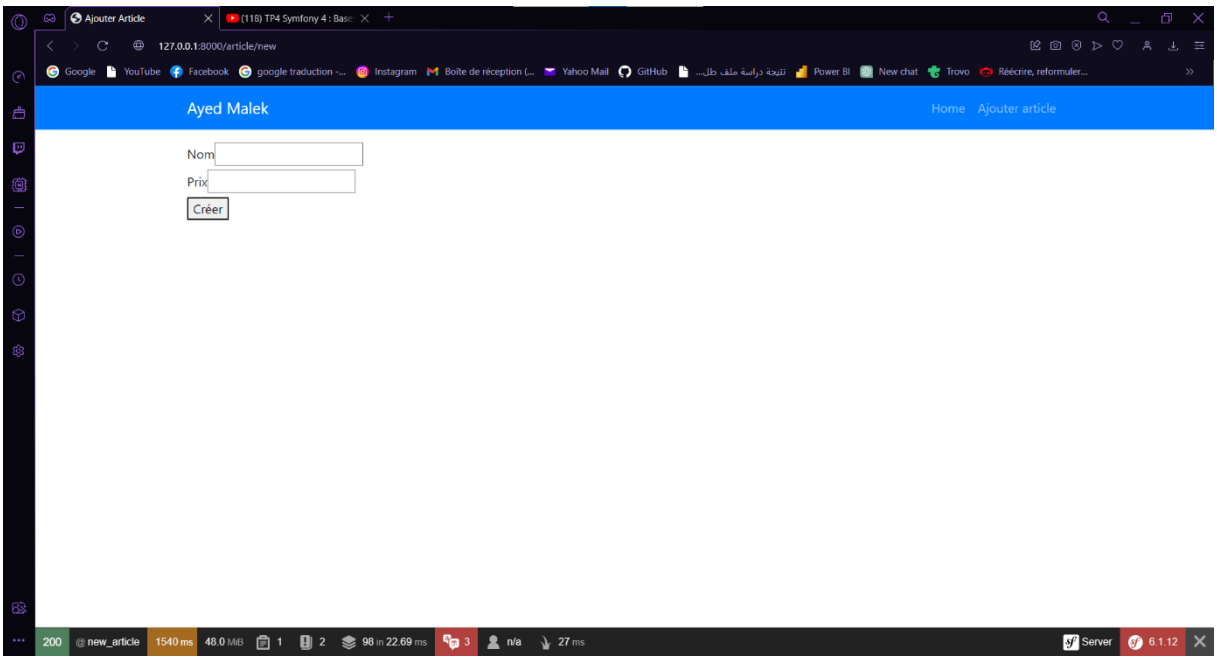
2. Ajouter la fonction qui permettent d'ajouter un nouvel article dans le fichier IndexController.php

```php
public function new(Request $request) {
        $article = new Article();
        $form = $this->createFormBuilder($article)
        ->add('nom', TextType::class)
        ->add('prix', TextType::class)
        ->add('save', SubmitType::class, array('label' => 'Créer')
        )->getForm();


        $form->handleRequest($request);

        if($form->isSubmitted() && $form->isValid()) {
        $article = $form->getData();

        $entityManager = $this->entityManager->getManager();
        $entityManager->persist($article);
        $entityManager->flush();

        return $this->redirectToRoute('article_list');
        }
        return $this->render('articles/new.html.twig',['form' => $form->createView()]);
        }
```

2.1 Ajouter le routage de cette fonction dans le fichier routes.yaml

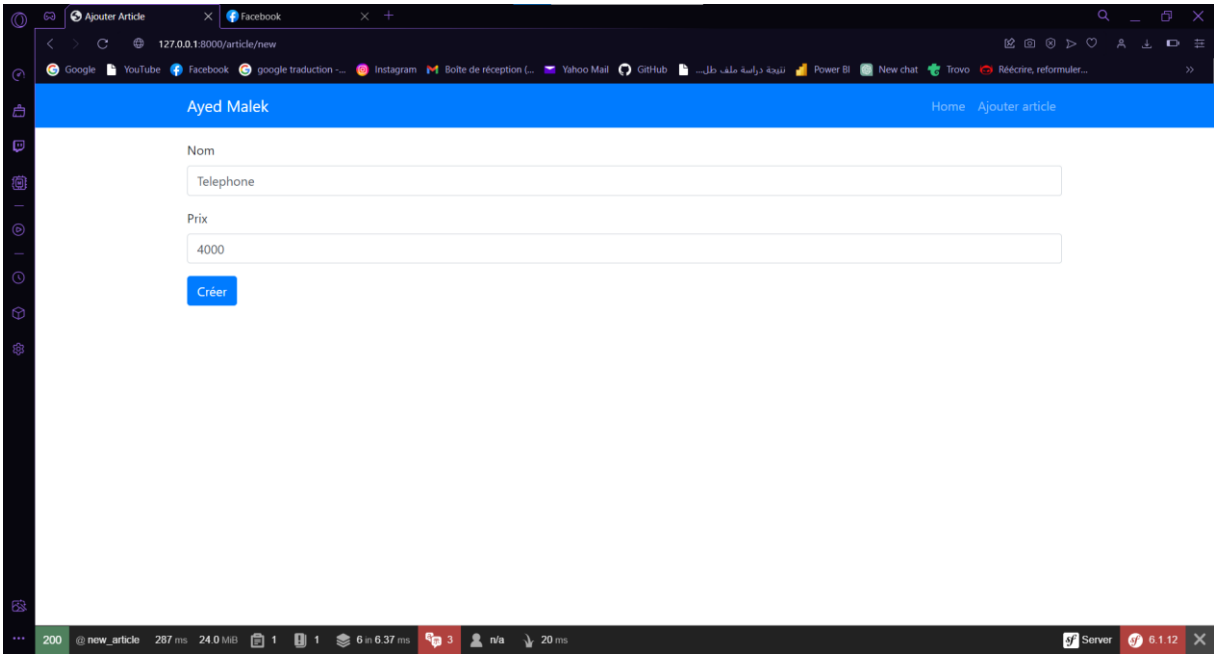
```yaml
new_article:
  path: /article/new
  controller: App\Controller\IndexController::new
```

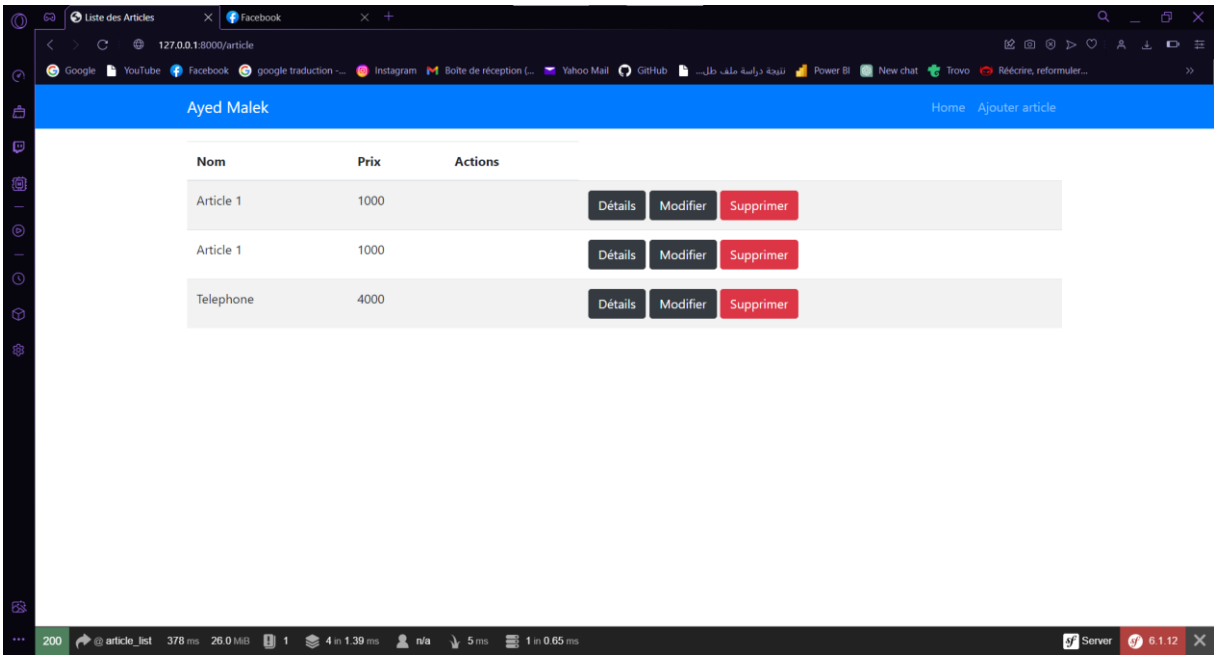
2.2 Tester le travail



3. Ajouter Bootstrap à l'application dans le fichier twig.yaml

```yaml
twig:
    default_path: '%kernel.project_dir%/templates'
    form_themes: ['bootstrap_4_layout.html.twig']
```





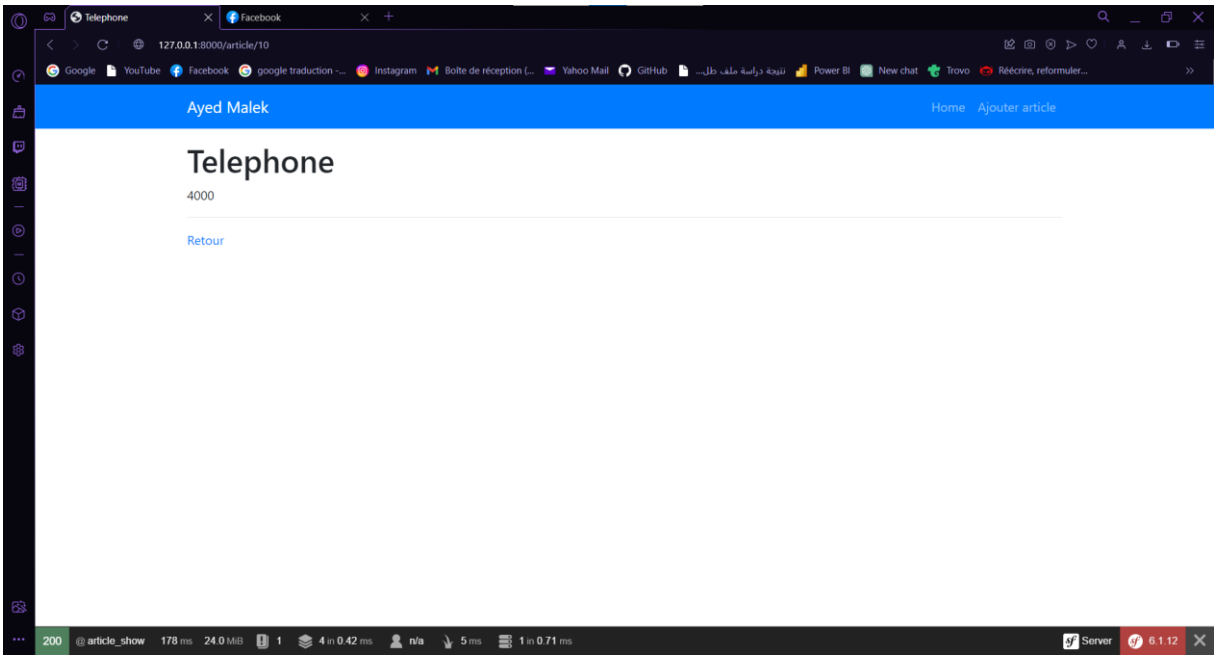
4. Ajouter la fonction qui affiche les details du produit dans le fichier IndexController.php

```
public function show($id) {
        $article = $this->entityManager->getRepository(Article::class)->find($id);
        return $this->render('articles/show.html.twig', array('article' => $article));
            }
```

4.1 Ajouter le routage de cette fonction dans routes.yaml

```yaml
article_show:
  path: /article/{id}
  controller: App\Controller\IndexController::show
```

4.2 Tester la méthode



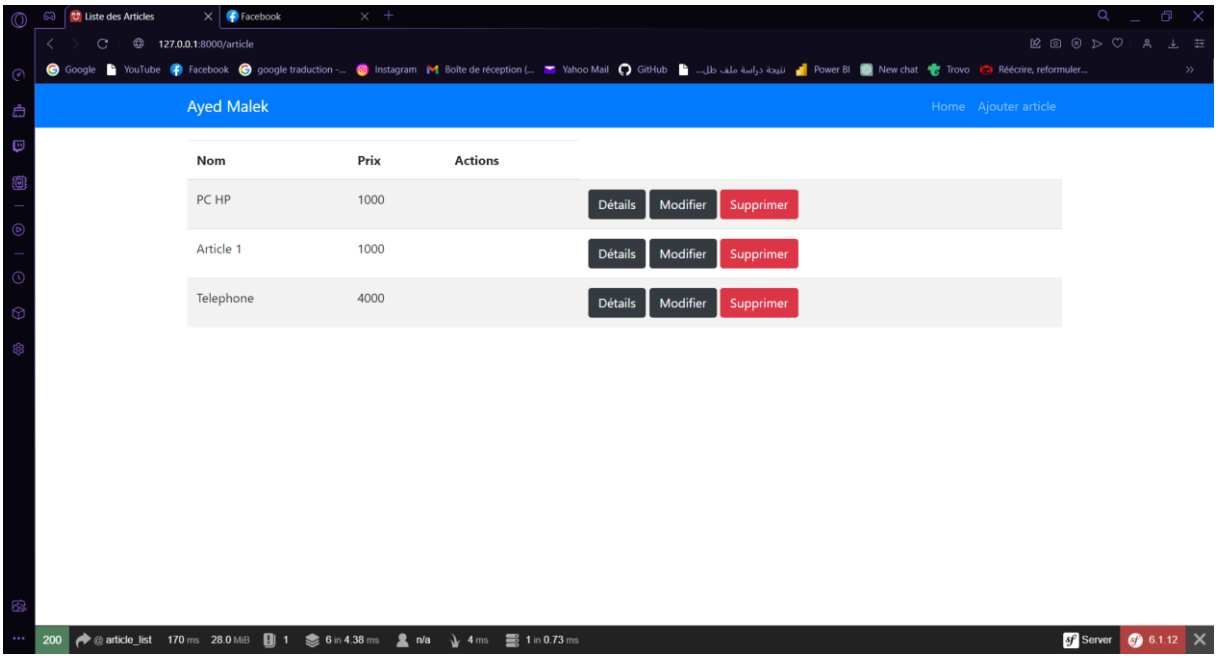
5. Ajouter la fonction update d'un article dans le fichier IndexController.php

```php
public function edit(Request $request, $id) {
        $article = new Article();
        $article = $this->entityManager->getRepository(Article::class)->find($id);

        $form = $this->createFormBuilder($article)
                ->add('nom', TextType::class)
                ->add('prix', TextType::class)
                ->add('save', SubmitType::class, array(
                'label' => 'Modifier'
                ))->getForm();

                $form->handleRequest($request);
                if($form->isSubmitted() && $form->isValid()) {

                $entityManager = $this->getDoctrine()->getManager();
                $entityManager->flush();

                return $this->redirectToRoute('article_list');
                return $this->render('articles/edit.html.twig', ['form' => $form->createView()]);
                }
            }
```

5.1 Ajouter le routage de cette fonction dans routes.yaml

```yaml
edit_article:
  path: /article/edit/{id}
  controller: App\Controller\IndexController::edit
```

5.2 Tester le Travail



6. Ajouter la fonction de suppression d'un article dans le fichier IndexController.php

```php
public function delete(Request $request, $id) {
    $article = $this->entityManager()->getRepository(Article::class)->find($id);

    $entityManager = $this->entityManager()->getManager();
    $entityManager->remove($article);
    $entityManager->flush();

    $response = new Response();
    $response->send();
    return $this->redirectToRoute('article_list');
    }
```

6.1 Ajouter le routage de cette fonction dans routes.yaml

```
delete_article:
 path: /article/delete/{id}
 controller: App\Controller\IndexController::delete
```

6.2 Tester le travail