

# Projet SymRecipe\*

Première partie

13 mai 2024

## 1 Installation et configuration

### 1.1 Prérequis

Le framework Symfony doit être installé et configuré.

Si ce n'est pas le cas, rendez-vous sur [cette page de documentation](#) pour vérifier les prérequis techniques qui se résument à deux choses :

- PHP version 8.0.2 ou plus
- Composer

Sur cette page, il est suggéré d'installer Symfony CLI donc je t'invite à le faire si ce c'est pas déjà fait.

#### 1.1.1 A votre tour



Exécutez la commande suivante pour vérifier tous les prérequis techniques :

```
symfony check:requirements
```



---

\*Emilien Gantois - <https://www.youtube.com/@developpeur.muscle>

```

PS C:\Users\alexa\symfony-projects> symfony check:requirements

Symfony Requirements Checker
=====

> PHP is using the following php.ini file:
C:\Users\alexa\UwAmp\bin\php\php-8.2.11\php.ini

> Checking Symfony requirements:

.....WW.....

[OK]
Your system is ready to run Symfony projects

Note The command console can use a different php.ini file
      than the one used by your web server.
      Please check that both the console and the web server
      are using the same PHP version and configuration.

```

## 1.2 Création du projet

### 1.2.1 A votre tour



1. Placez vous dans votre dossier de travail et exécutez la commande suivante :

```
symfony new symrecipe --webapp
```



```

Windows PowerShell
PS C:\Users\alexa\symfony-projects> symfony new symrecipe --webapp
* Creating a new Symfony project with Composer
  (running C:\ProgramData\ComposerSetup\bin\composer.phar create-project symfony/skeleton C:\Users\alexa\symfony-projects\symrecipe --no-interaction)

* Setting up the project under Git version control
  (running git init C:\Users\alexa\symfony-projects\symrecipe)

  (running C:\ProgramData\ComposerSetup\bin\composer.phar require webapp --no-interaction)

[OK] Your project is now ready in C:\Users\alexa\symfony-projects\symrecipe

PS C:\Users\alexa\symfony-projects> |

```

2. Puis, montez dans le dossier du projet (symrecipe) et exécutez la commande suivante :

```
symfony serve -d
```



```
Windows PowerShell
PS C:\Users\alexa\symfony-projects> cd .\symrecipe\
PS C:\Users\alexa\symfony-projects\symrecipe> symfony serve -d

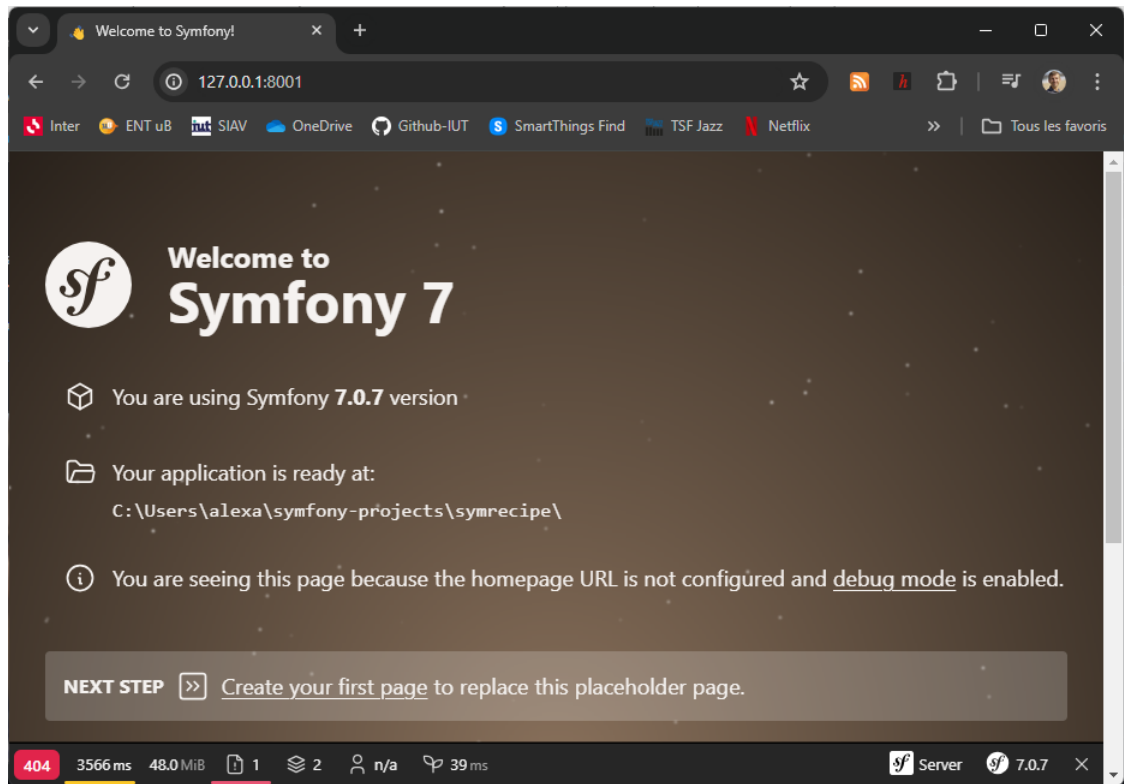
[WARNING] run "symfony.exe server:ca:install" first if you want to run the web server with TLS support, or use "--p12" or "--no-tls" to avoid this warning

[WARNING] The local web server is optimized for local development and MUST never be used in a production setup.

[OK] Web server listening
The Web server is using PHP CGI 8.2.11
http://127.0.0.1:8001

Stream the logs via symfony.exe server:log
PS C:\Users\alexa\symfony-projects\symrecipe> |
```

3. Enfin, dans un navigateur, rendez-vous à l'URL indiquée : 🖱️



Sur [github](#), créez un dépôt distant nommé « symrecipe » et reliez le référentiel local que vous venez de créer (fait automatiquement). Pour ce faire, vous pouvez suivre les consignes données par votre dépôt distant.

Enfin, n'oubliez pas de m'inviter.

## 2 Controller, Routing et Twig

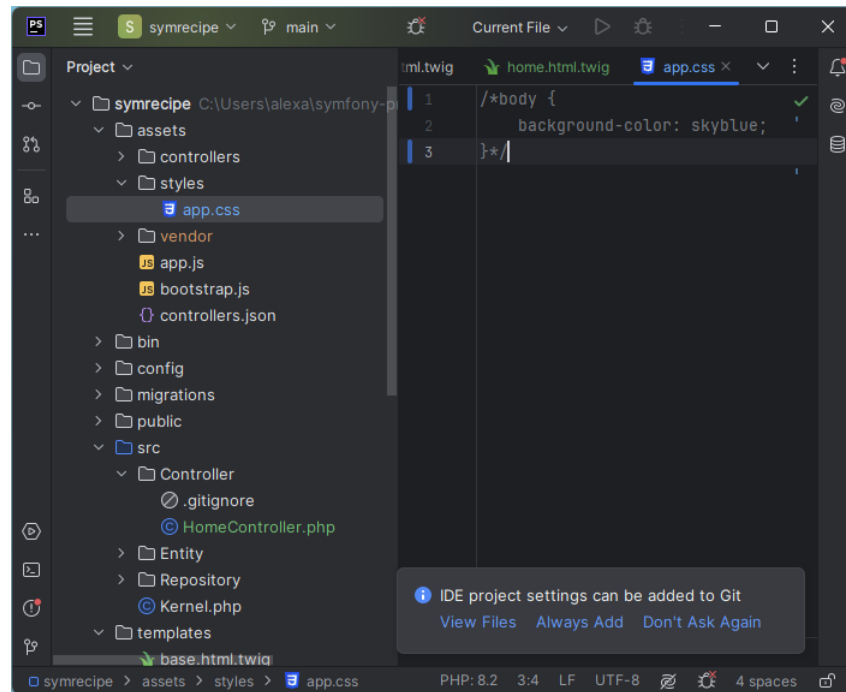
### 2.1 Première page

#### 2.1.1 A votre tour



Commencez par vous placer dans le dossier de votre projet `symrecipe`.

1. Commentez le code présent dans le fichier `asset/styles/app.css` :



2. Créez un nouveau fichier `templates/home.html.twig` avec le contenu suivant :

```
{% extends "base.html.twig" %}

{% block title %}SymRecipe - Accueil{% endblock %}

{% block body %}
    <h1>Hello SymRecipe !</h1>
{% endblock %}
```

3. Créez un nouveau fichier `src/Controller/HomeController.php` avec le contenu suivant :

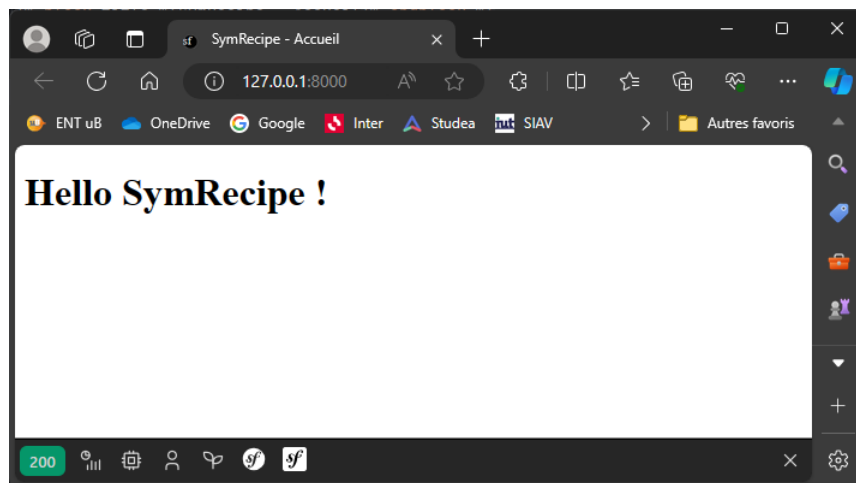
```
<?php

namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Annotation\Route;

class HomeController extends AbstractController
{
    #[Route('/', 'home.index', methods: ['GET'])]
    public function index(): Response
    {
        return $this->render('home.html.twig');
    }
}
```

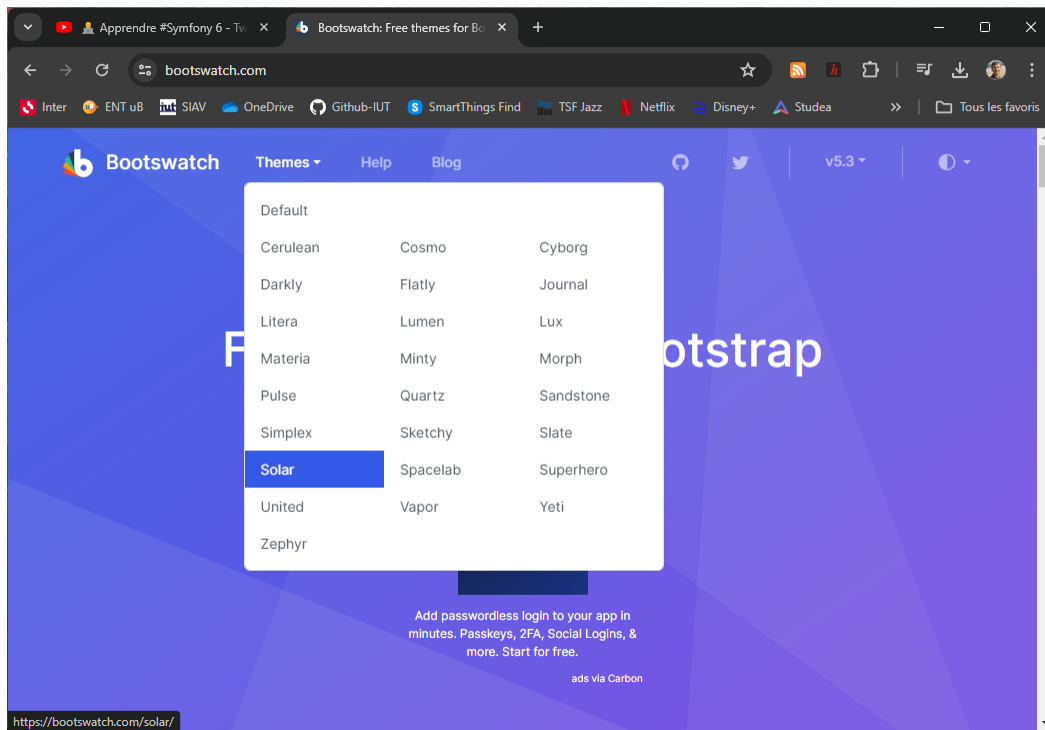
4. Testez :



Validez avec le message « Première page » et poussez !

## 2.2 Bootswatch

On va construire rapidement un design correct en utilisant [Bootswatch](#) :



Je choisis le thème **solar**. Choisissez en un autre !

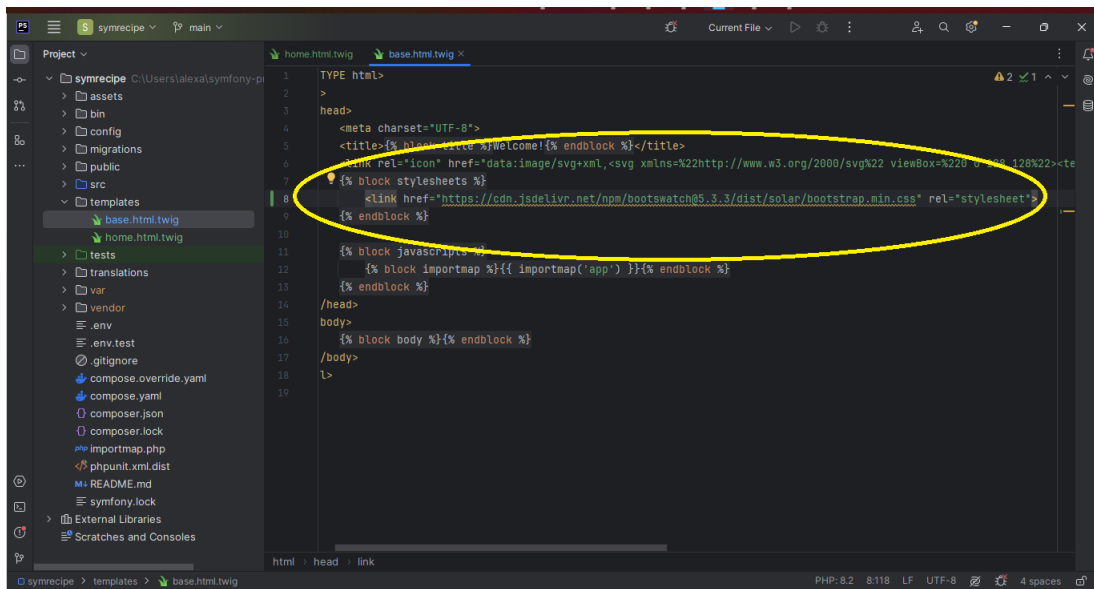
### 2.2.1 A votre tour



Ajoutez le lien suivant (en remplaçant **solar** par le nom du thème que vous avez choisi) dans le block `stylesheets` du fichier `templates/base.html.twig` :

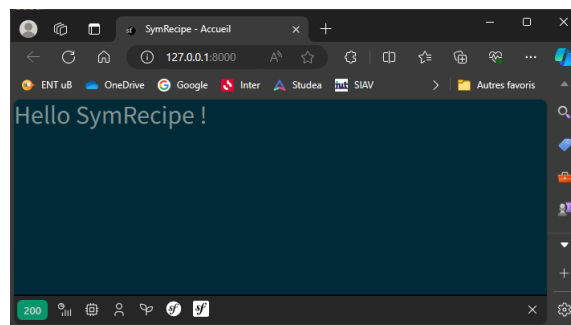
```
<link href="
https://cdn.jsdelivr.net/npm/bootswatch@5.3.3/dist/solar/bootstrap.min.css
" rel="stylesheet">
```





```
1 TYPE html>
2 >
3 <head>
4 <meta charset='\"UTF-8\"'>
5 <title>{% block title %}Welcome!{% endblock %}</title>
6 <link rel='\"icon\"' href='\"data:image/svg+xml,<svg xmlns='\"http://www.w3.org/2000/svg\"%22 viewBox='\"0 0 24 24\"%22><te
7 <link href='\"https://cdn.jsdelivr.net/npm/bootswatch@5.3.3/dist/solar/bootstrap.min.css\" rel='\"stylesheet\"'>
8 </link>
9 </head>
10 <body>
11 <div class='\"container\"'>
12 <div class='\"row\"'>
13 <div class='\"col\"'>
14 </div>
15 </div>
16 </div>
17 </body>
18 </html>
```

Puis testez :



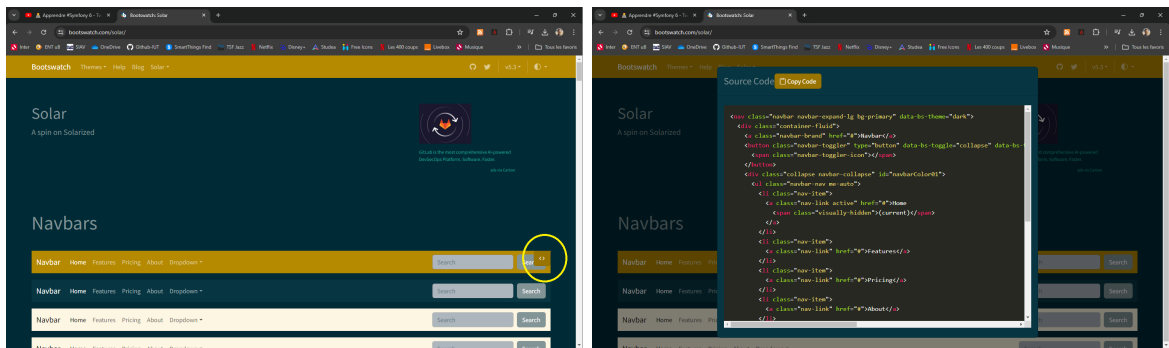
Validez avec le message « Bootswatch » et poussez !

## 2.3 Barre de navigation

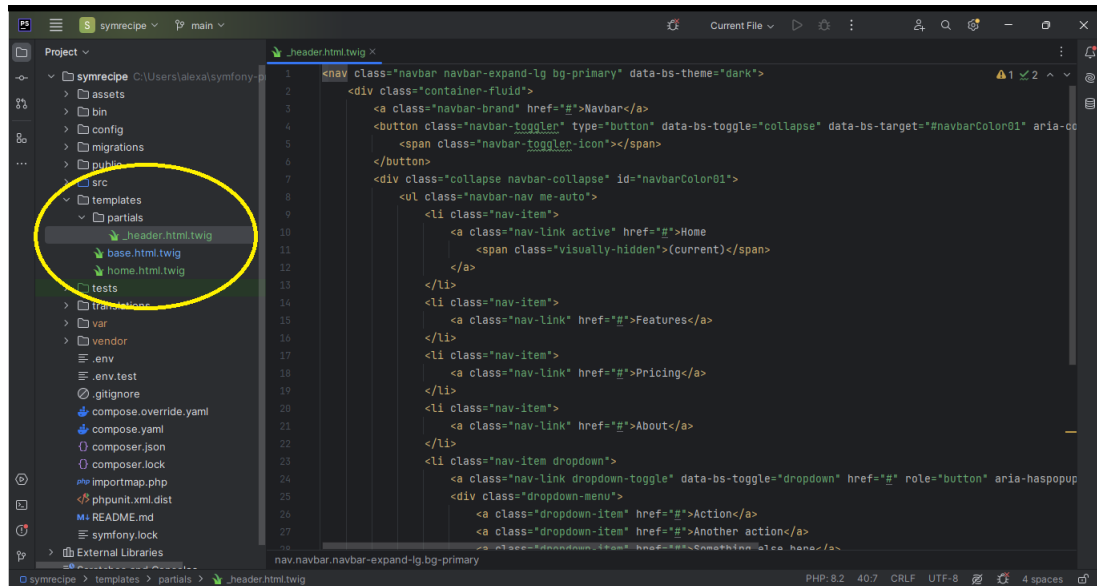
### 2.3.1 A votre tour



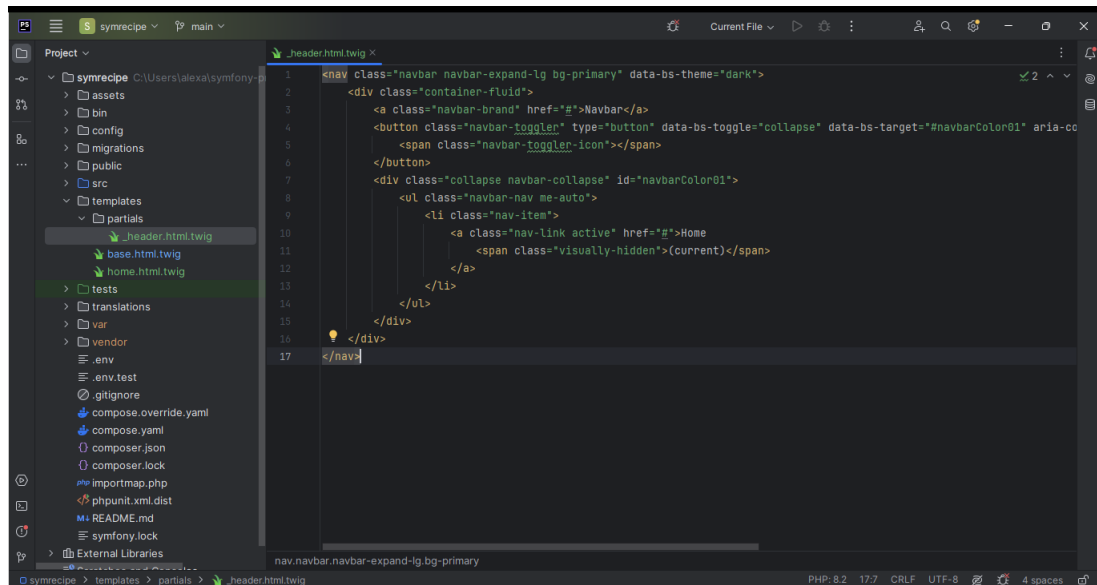
- Rendez-vous sur la page du thème Bootswatch choisi, <https://bootswatch.com/solar/> pour moi, et copiez le code de la première Navbar :



Puis collez le dans un nouveau fichier `templates/partials/_header.html.twig` :



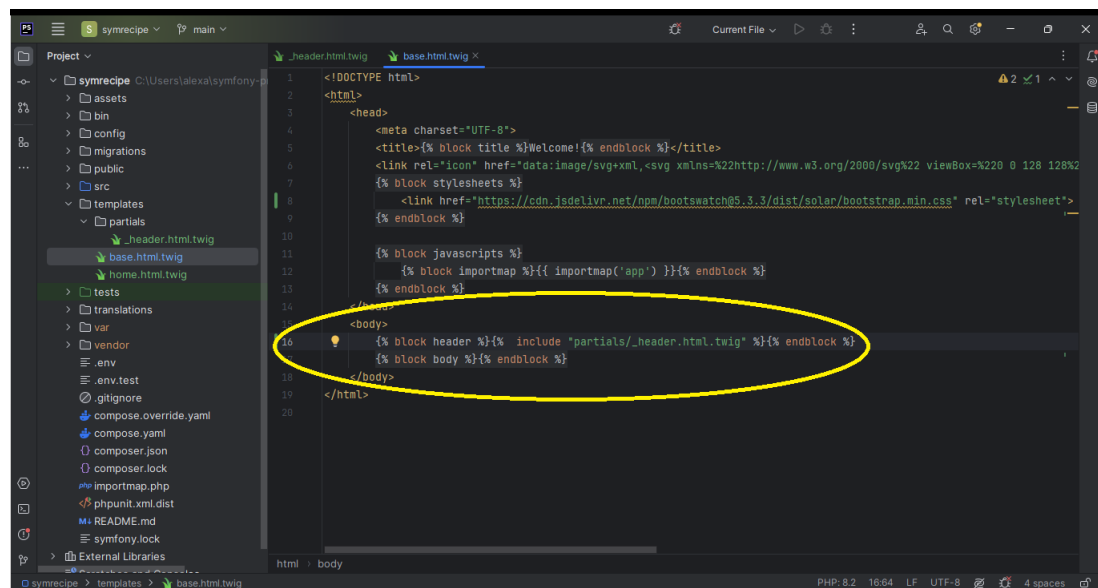
Enfin, supprimez le code inutile dans ce composant nav pour ne garder que le lien vers la page d'accueil :



- Dans le fichier `templates/base.html.twig`, créez un nouveau block header au début

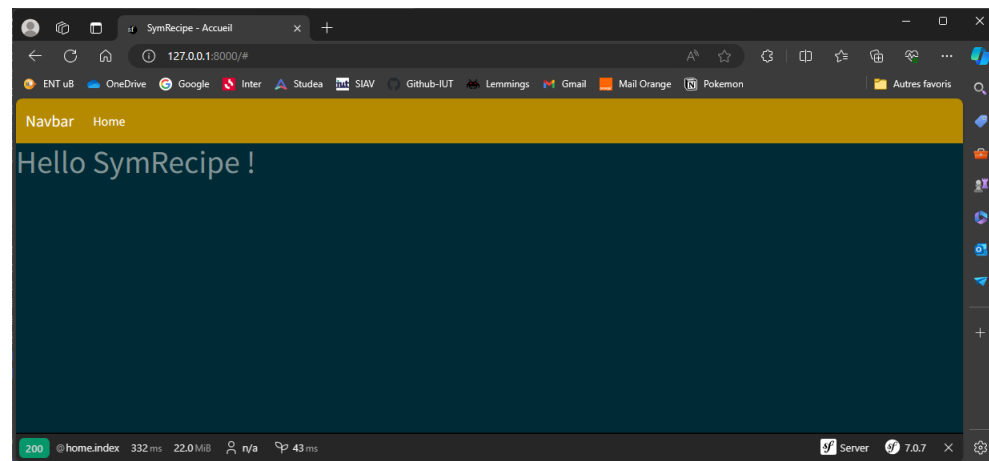


du body et insérez dans ce nouveau block header l'instruction twig :  
include "partials/\_header.html.twig" :



```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="UTF-8">
5     <title>{% block title %}Welcome!{% endblock %}</title>
6     <link rel="icon" href="data:image/svg+xml,<svg xmlns="http://www.w3.org/2000/svg"22 viewbox="220 0 128 128%2
7     {% block stylesheets %}
8     <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/solar/bootstrap.min.css" rel="stylesheet">
9     {% endblock %}
10
11     {% block javascripts %}
12     {% block importmap %}{% importmap('app') %}{% endblock %}
13     {% endblock %}
14   </head>
15   <body>
16     {% block header %}{% include 'partials/_header.html.twig' %}{% endblock %}
17     {% block body %}{% endblock %}
18   </body>
19 </html>
20
```

Rafraichissez :



Validez avec le message « Barre de navigation » et poussez !

## 2.4 Bouton d'inscription

### 2.4.1 A votre tour



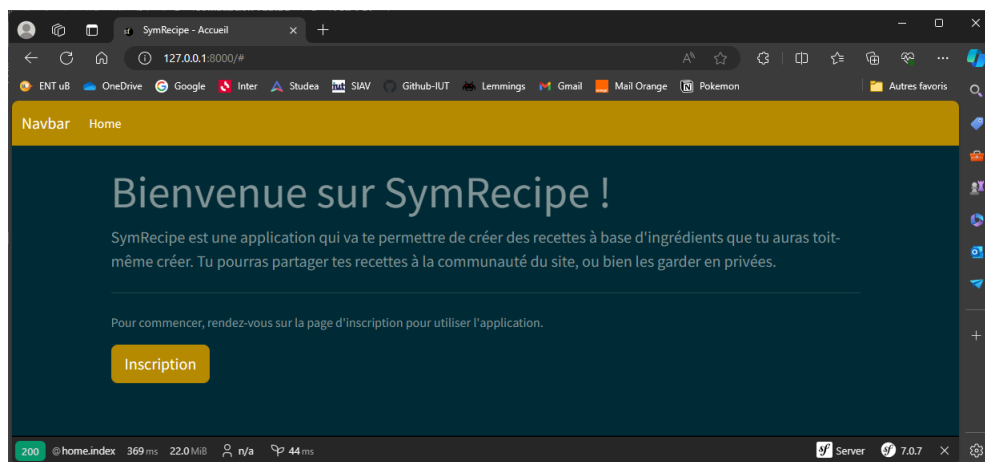
Dans le fichier `templates/home.html.twig`, insérez le code suivant dans le block `body` :

```

{% block body %}
<div class="container mt-4">
  <div class="jumbotron">
    <p class="lead">
      SymRecipe est une application qui va te permettre de créer
      des recettes à base d'ingrédients que tu auras toi-même
      créer. Tu pourras partager tes recettes à la communauté du
      site, ou bien les garder en privées.
    </p>
    <hr class="my-4">
    <p>
      Pour commencer, rendez-vous sur la page d'inscription
      pour utiliser l'application.
    </p>
    <a class="btn btn-primary btn-lg" href="#" role="button">
      Inscription
    </a>
  </div>
</div>
{% endblock %}

```

Rafraichissez :



Validez avec le message « Bouton d'inscription » et poussez !

## 3 ORM Doctrine, entité et migration

### 3.1 Configurer notre base de données

#### 3.1.1 A votre tour



Commençons par configurer Doctrine ORM dans l'application Symfony. Pour ce faire, complétez le fichier `.env` comme indiqué ci-après :

```
# .env

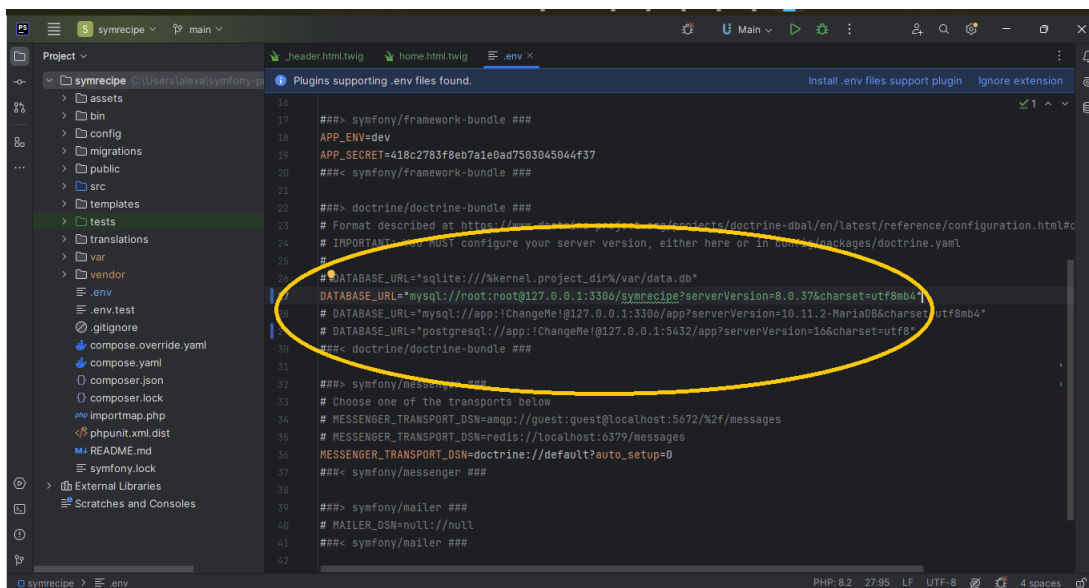
# Laissez le reste du contenu de ce fichier intact.

###> doctrine/doctrine-bundle ###
DATABASE_URL=mysql://db_user:db_password@127.0.0.1:3306/db_name
###< doctrine/doctrine-bundle ###
```

Très souvent, il faudra seulement définir le `DATABASE_URL` qui peut être de la forme suivante et qui dépend de la configuration et du type de base de données utilisé. Voici le mien :

`DATABASE_URL=`

`"mysql://root:root@127.0.0.1:3306/symrecipe?serverVersion=8.0.37&charset=utf8mb4"`



N'oubliez pas d'activer l'extension `pdo_mysql` dans votre fichier `php.ini` si ce n'est pas déjà fait.

Symfony est fourni avec une intégration de Doctrine ORM qui fournit de nombreuses commandes dans la console. Vous allez donc créer la base de données avec la commande :

```
symfony console doctrine:database:create
```



```
Windows PowerShell
PS C:\Users\alexa\symfony-projects\symrecipe> symfony console doctrine:database:create
Created database 'symrecipe' for connection named default
PS C:\Users\alexa\symfony-projects\symrecipe> |
```

Cette commande a généré et exécuté les instructions nécessaires à la création de la base de données configurée.

Vérifiez donc que votre base de données a bien été créée :

```
MySQL 8.0 Command Li
mysql> show databases;
+-----+
| Database |
+-----+
| elevage  |
| information_schema |
| mon_super_projet |
| mysql    |
| performance_schema |
| symrecipe |
| sys      |
| test-webapp |
+-----+
8 rows in set (0.00 sec)

mysql> |
```

## 3.2 Création d'une première entité

### 3.2.1 A votre tour



1. Créons notre première entité avec la commande :

```
php bin/console make:entity
```

Puis répondez aux questions comme indiqué dans la capture d'écran suivante :

```
Windows PowerShell
PS C:\Users\alexa\symfony-projects\symrecipe> php bin/console make:entity

Class name of the entity to create or update (e.g. OrangePopsicle):
> Ingredient

Add the ability to broadcast entity updates using Symfony UX Turbo? (yes/no) [no]:
>

created: src/Entity/Ingredient.php
created: src/Repository/IngredientRepository.php

Entity generated! Now let's add some fields!
You can always add more fields later manually or by re-running this command.

New property name (press <return> to stop adding fields):
> name

Field type (enter ? to see all types) [string]:
>

Field length [255]:
>

Can this field be null in the database (nullable) (yes/no) [no]:
>

updated: src/Entity/Ingredient.php

Add another property? Enter the property name (or press <return> to stop adding fields):
> price

Field type (enter ? to see all types) [string]:
> float

Can this field be null in the database (nullable) (yes/no) [no]:
>

updated: src/Entity/Ingredient.php

Add another property? Enter the property name (or press <return> to stop adding fields):
> createdAt

Field type (enter ? to see all types) [datetime_immutable]:
>

Can this field be null in the database (nullable) (yes/no) [no]:
>

updated: src/Entity/Ingredient.php

Add another property? Enter the property name (or press <return> to stop adding fields):
>

Success!

Next: When you're ready, create a migration with php bin/console make:migration

PS C:\Users\alexa\symfony-projects\symrecipe> |
```

Remarquez qu'à la toute fin, Symfony nous dit que nous sommes maintenant prêts pour créer une migration avec la commande `php bin/console make:migration`. Nous y reviendrons.

Pour l'instant, vérifiez que la commande `php bin/console make:entity` a bien créé les deux fichiers suivants :

- `src/Entity/Ingredient.php`
- `src/Repository/IngredientRepository.php`

2. Effectuez maintenant une migration avec la commande :

```
php bin/console make:migration
```



```
Windows PowerShell
PS C:\Users\alexa\symfony-projects\symrecipe> php bin/console make:migration
created: migrations/Version20240508095750.php

Success!

Review the new migration then run it with php bin/console doctrine:migrations:migrate
See https://symfony.com/doc/current/bundles/DoctrineMigrationsBundle/index.html
PS C:\Users\alexa\symfony-projects\symrecipe>
```

Puis dans le répertoire migrations de votre projet symrecipe, vérifiez que vous avez un nouveau fichier (nommé `Version20240508095750.php` chez moi). Il s'agit d'une traduction de notre entité `Ingredient` en langage SQL.

3. Après avoir vérifié que la requête SQL de création de la table `Ingredient` est conforme aux spécifications de l'entité `Ingredient`, nous allons enfin exécuter cette migration dans la base données symrecipe qui est toujours vide jusqu'à présent :

```
php bin/console doctrine:migrations:migrate
```



```
Windows PowerShell
PS C:\Users\alexa\symfony-projects\symrecipe> php bin/console doctrine:migrations:migrate

WARNING! You are about to execute a migration in database "symrecipe" that could result
in schema changes and data loss. Are you sure you wish to continue? (yes/no) [yes]:
>

[notice] Migrating up to DoctrineMigrations\Version20240508095750
[notice] finished in 208.6ms, used 22M memory, 1 migrations executed, 2 sql queries

[OK] Successfully migrated to version: DoctrineMigrations\Version20240508095750

PS C:\Users\alexa\symfony-projects\symrecipe>
```

Vérifiez que vous avez maintenant 3 tables dans votre base de données symrecipe et contrôlez la conformité de la structure de la table `Ingredient` :

```
MySQL 8.0 Command Line Cli
mysql> show tables;
+-----+
| Tables_in_symrecipe |
+-----+
| doctrine_migration_versions |
| ingredient             |
| messenger_messages      |
+-----+
3 rows in set (0.00 sec)

mysql> desc ingredient;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id     | int  | NO   | PRI | NULL    | auto_increment |
| name   | varchar(255) | NO | | NULL    |
| price  | double | NO | | NULL    |
| created_at | datetime | NO | | NULL    |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.01 sec)

mysql> |
```



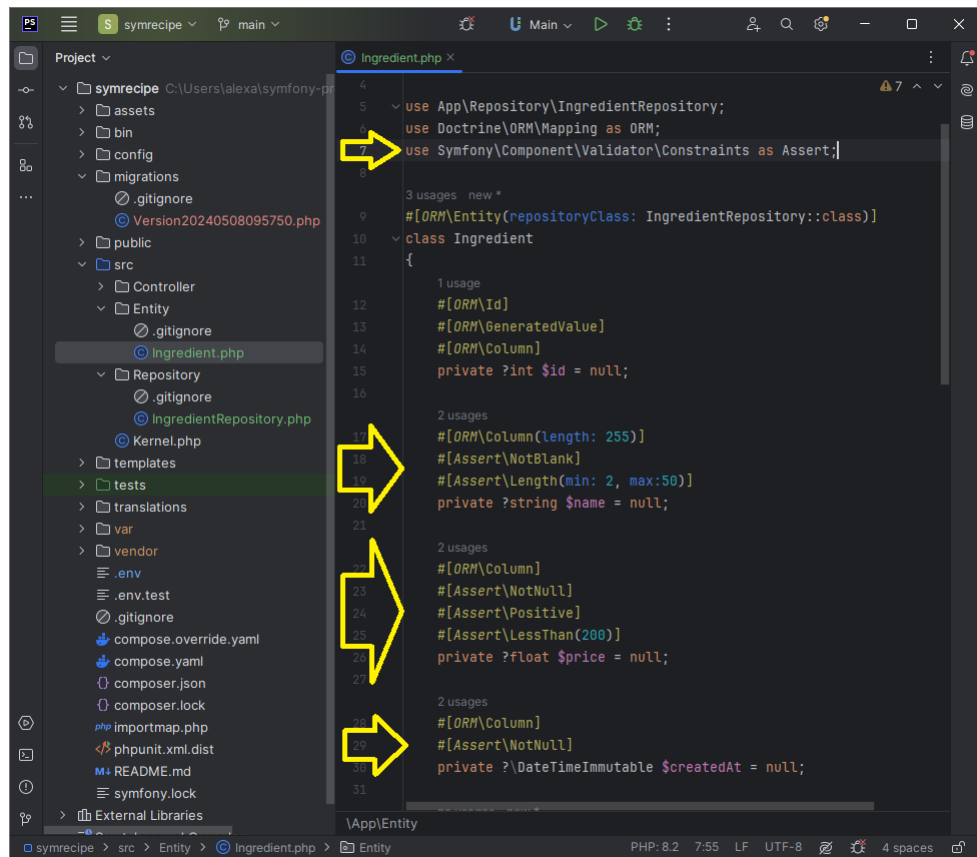
Validez avec le message « Création de l'entité Ingredient » et poussez !

### 3.3 Validation des données

#### 3.3.1 A votre tour



- Complétez votre entité **Ingredient** avec les contraintes suivantes :



- Puis ajoutez le constructeur suivant :

```
<?php
// src/Entity/Ingredient.php

// ...

public function __construct()
{
    $this->createdAt = new \DateTimeImmutable();
}

// ...
```

Le `\` devant `DateTimeImmutable()` évite d'avoir à importer le `DateTimeImmutable`.

## 3.4 Les Fixtures

Les [fixtures](#) vont nous permettre de générer un jeu de fausses données.

Grâce à *Symfony Flex*, on a juste à dire à *Composer* d'installer le *bundle* `orm-fixtures` pour qu'il s'occupe de tout.

### 3.4.1 A votre tour





1. Dans votre terminal, exécutez la commande suivante :

```
composer require --dev orm-fixtures
```



```
Windows PowerShell
PS C:\Users\alexa\symfony-projects\symrecipe> composer require --dev orm-fixtures
./composer.json has been updated
Running composer update doctrine/doctrine-fixtures-bundle
Loading composer repositories with package information
Updating dependencies
Lock file operations: 2 installs, 0 updates, 0 removals
- Locking doctrine/data-fixtures (1.7.0)
- Locking doctrine/doctrine-fixtures-bundle (3.6.0)
Writing lock file
Installing dependencies from lock file (including require-dev)
Package operations: 2 installs, 0 updates, 0 removals
- Downloading doctrine/doctrine-fixtures-bundle (3.6.0)
- Installing doctrine/data-fixtures (1.7.0): Extracting archive
- Installing doctrine/doctrine-fixtures-bundle (3.6.0): Extracting archive
Generating autoload files
114 packages you are using are looking for funding.
Use the 'composer fund' command to find out more!

Symfony operations: 1 recipe (0852b59520540458039765428ddcb69774)
- Configuring doctrine/doctrine-fixtures-bundle (>=3.0): From github.com/symfony/recipes:main
Executing script cache:clear [OK]
Executing script assets:install public [OK]
Executing script importmap:install [OK]

What's next?

Some files have been created and/or updated to configure your new packages.
Please review, edit and commit them: these files are yours.

No security vulnerability advisories found.
Using version ^3.6 for doctrine/doctrine-fixtures-bundle
PS C:\Users\alexa\symfony-projects\symrecipe> |
```

2. Ensuite, ouvrez le fichier `src/DataFixtures/AppFixtures.php` nouvellement créé et constatez qu'il y a 2 lignes de code commentées, données pour l'exemple :

```
Project
├── routes.yaml
├── services.yaml
├── migrations
├── .gitignore
├── Version20240508095750.php
├── public
├── src
│   ├── Controller
│   ├── DataFixtures
│   │   ├── AppFixtures.php
│   │   ├── Entity
│   │   │   ├── .gitignore
│   │   │   ├── Ingredient.php
│   │   │   ├── .gitignore
│   │   │   ├── IngredientRepository.php
│   │   │   ├── Kernel.php
│   │   ├── templates
│   │   ├── tests
│   │   ├── translations
│   │   ├── var
│   │   ├── vendor
│   │   ├── .env
│   │   ├── .env.test
│   │   ├── .gitignore
│   │   ├── compose.override.yaml
│   │   ├── compose.yaml
│   │   ├── composer.json
│   │   └── composer.lock
├── templates
├── tests
├── translations
├── var
├── vendor
├── .env
├── .env.test
├── .gitignore
├── compose.override.yaml
├── compose.yaml
├── composer.json
└── composer.lock

AppFixtures.php
1 <?php
2
3 namespace App\DataFixtures;
4
5 use ...
6
7 new *
8 class AppFixtures extends Fixture
9 {
10     new *
11     public function load(ObjectManager $manager): void
12     {
13         // $product = new Product();
14         // $manager->persist($product);
15         $manager->flush();
16     }
17 }
18
```

Nous allons les adapter à notre contexte.

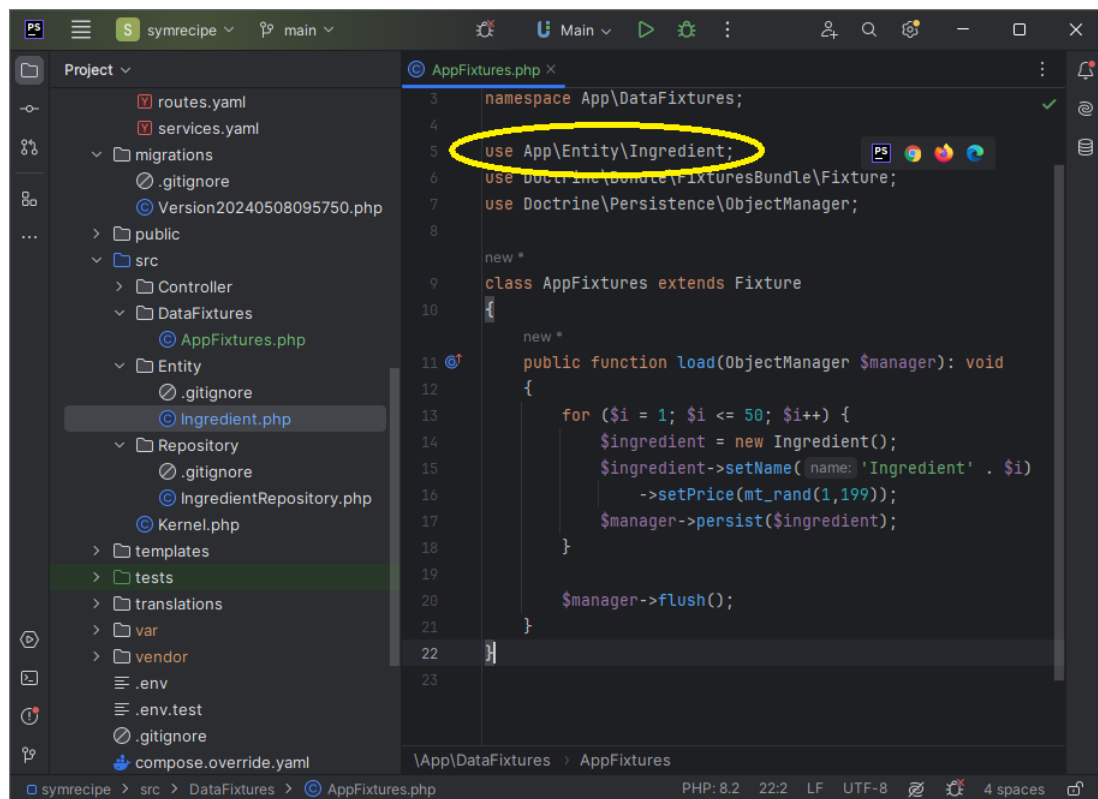
Remplacez donc des deux lignes commentées par celles-ci :

```
<?php
// src/DataFixtures/AppFixtures.php
// ...

for ($i = 1; $i <= 50; $i++) {
    $ingredient = new Ingredient();
    $ingredient->setName('Ingredient' . $i)
        ->setPrice(mt_rand(1,199));
    $manager->persist($ingredient);
}

// ...
```

N'oubliez pas l'instruction `use App\Entity\Ingredient;` en début de fichier :



- Enfin, passez la commande suivante pour créer le jeu d'ingrédients dans la base de données :

```
php bin/console doctrine:fixtures:load
```



```
Windows PowerShell
PS C:\Users\alexa\symfony-projects\symrecipe> php bin/console doctrine:fixtures:load
Careful, database "symrecipe" will be purged. Do you want to continue? (yes/no) [no]:
> yes
> purging database
> loading App\DataFixtures\AppFixtures
PS C:\Users\alexa\symfony-projects\symrecipe> |
```

Vérifiez côté base de données :

```
MySQL 8.0 Command Line Cl
mysql> select count(*) from ingredient;
+-----+
| count(*) |
+-----+
|      50 |
+-----+
1 row in set (0.01 sec)

mysql> select * from ingredient limit 5;
+-----+-----+-----+-----+
| id | name       | price | created_at |
+-----+-----+-----+-----+
| 1 | Ingredient1 | 199   | 2024-05-08 14:37:26 |
| 2 | Ingredient2 | 58    | 2024-05-08 14:37:26 |
| 3 | Ingredient3 | 92    | 2024-05-08 14:37:26 |
| 4 | Ingredient4 | 199   | 2024-05-08 14:37:26 |
| 5 | Ingredient5 | 109   | 2024-05-08 14:37:26 |
+-----+-----+-----+-----+
5 rows in set (0.00 sec)

mysql> |
```

## 3.5 FakerPHP

On va également installer et utiliser [FakerPHP](#) qui va nous permettre de générer des faux noms, des fausses adresses e-mail, etc. toujours dans le but de rendre les données fictives plus concrètes.

### 3.5.1 A votre tour



1. Commencez par installer le paquet :

```
composer require --dev fakerphp/faker
```



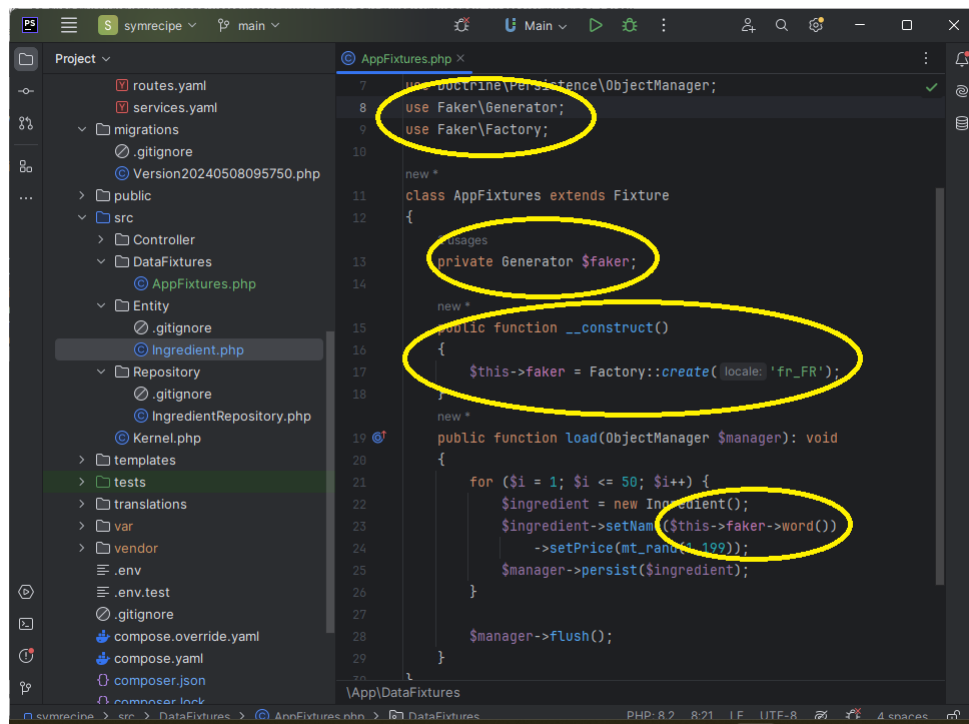
```
Windows PowerShell
PS C:\Users\alexa\symfony-projects\symrecipe> composer require --dev fakerphp/faker
./composer.json has been updated
Running composer update fakerphp/faker
Loading composer repositories with package information
Updating dependencies
Lock file operations: 1 install, 0 updates, 0 removals
- Locking fakerphp/faker (v1.23.1)
Writing lock file
Installing dependencies from lock file (including require-dev)
Package operations: 1 install, 0 updates, 0 removals
- Downloading fakerphp/faker (v1.23.1)
- Installing fakerphp/faker (v1.23.1): Extracting archive
Generating autoload files
114 packages you are using are looking for funding.
Use the 'composer fund' command to find out more!

Run composer recipes at any time to see the status of your Symfony recipes.

Executing script cache:clear [OK]
Executing script assets:install public [OK]
Executing script importmap:install [OK]

No security vulnerability advisories found.
Using version ^1.23 for fakerphp/faker
PS C:\Users\alexa\symfony-projects\symrecipe> |
```

2. Puis modifiez la classe AppFixtures :



```
AppFixtures.php
7 use Doctrine\Persistence\ObjectManager;
8 use Faker\Generator;
9 use Faker\Factory;
10
11 new *
12 class AppFixtures extends Fixture
13 {
14     private Generator $faker;
15
16     new *
17     public function __construct()
18     {
19         $this->faker = Factory::create('fr_FR');
20     }
21
22     new *
23     public function load(ObjectManager $manager): void
24     {
25         for ($i = 1; $i <= 50; $i++) {
26             $ingredient = new Ingredient();
27             $ingredient->setName($this->faker->word());
28             $ingredient->setPrice(mt_rand(1, 100));
29             $manager->persist($ingredient);
30         }
31
32         $manager->flush();
33     }
34 }
```

3. Enfin, repassez la commande `php bin/console doctrine:fixtures:load` et vérifiez les modifications du jeu d'ingrédients dans la base de données :

```
MySQL 8.0 Command Line Cl  X + - □ X

mysql> select count(*) from ingredient;
+-----+
| count(*) |
+-----+
|      50 |
+-----+
1 row in set (0.00 sec)

mysql> select * from ingredient limit 5;
+----+-----+-----+-----+
| id | name      | price | created_at |
+----+-----+-----+-----+
| 51 | accusamus | 157   | 2024-05-08 15:47:14 |
| 52 | voluptatibus | 197   | 2024-05-08 15:47:14 |
| 53 | quidem    | 104   | 2024-05-08 15:47:14 |
| 54 | officia   | 90    | 2024-05-08 15:47:14 |
| 55 | molestiae | 52    | 2024-05-08 15:47:14 |
+----+-----+-----+-----+
5 rows in set (0.00 sec)

mysql> |
```



| Validez avec le message « Fixtures et FakerPHP » et poussez !