TP1 : Install / Controller / Routes

Nicolas Fabre / Alex Neymond (2h)

1. Installer, ou finir de configurer une instance de Symfony4 sur un environnement local.
    a. Installer SF
    b. Configurez un serveur web local vers le dossier /public de symfony
        i. Soit en utilisant un serveur web classique
        ii. Soit en utilisant le composant "server" de symfony 4 (composer require server +)
    c. A ce stade vous devez être à même d'ouvrir la page d'accueil de symfony dans un navigateur.
2. Installez les bundle "maker", "annotations" et "twig" à l'aide de composer
    a. Analysez les modifications intervenues dans le dossier /config
3. Listez les commandes du composant "console" de symfony : **php bin/console**
4. Créez votre premier controller : **FrontController**. **Manuellement**, dans le dossier src/controller. Cette classe héritera de la classe *Symfony\Bundle\FrameworkBundle\Controller\AbstractController*

FrontController.php

```php
<?php

namespace App\Controller;

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

class FrontController extends AbstractController
{
    /**
     * @Route("/", name="front_home")
     */
    public function index()
    {
        return $this->render( view: 'front/index.html.twig', [
            'controller_name' => 'FrontController',
        ]);
    }
}
```

5. Dans ce controller, créez, à l'aide d'annotation, une route pour la page d'accueil, ainsi que son template :
    d. Elle répondra sur l'url « /».
    e. Elle retourne le contenu d'un Template Twig, qui sera placé dans le dossier templates/front
    f. Cette route portera le nom de « front_home »
6. Installez Doctrine et connectez une base de donnée
7. Créer manuellement une entité App:User dans le dossier src/Entity
    g. Elle contiendra les champs :
        - id,

- Username (VARCHAR 255)
- Email (VARCHAR 255)
- Firstname (VARCHAR 255)
- Lastname (VARCHAR 255)
- JobTitle (VARCHAR 255) : « Chef de rang » « serveur »
- Enabled (boolean, not null)
- CreatedAt (DateTlime, automatiquement mis à jour lors de la création de l'entité) et UpdatedAt (DateTime, automatiquement mis à jour lors de la modification de l'entité)

b. Elle devra être persistée dans la table SQL : User

c. Renseigner l'ensemble des Getter/Setter manuellement

d. Mettre à jour le schéma de base de données et créer le Repository associé

FrontController.php | User.php

```php
<?php

namespace App\Entity;

use Doctrine\ORM\Mapping as ORM;

/**
 * @ORM\Entity(repositoryClass="App\Repository\UserRepository")
 */
class User
{
    /**
     * @ORM\Id()
     * @ORM\GeneratedValue()
     * @ORM\Column(type="integer")
     */
    private $id;

    /**
     * @ORM\Column(type="string", length=255, nullable=true)
     */
    private $username;

    /**
     * @ORM\Column(type="string", length=255, nullable=true)
     */
    private $email;

    /**
     * @ORM\Column(type="string", length=255, nullable=true)
     */
    private $firstname;
```

...

```php
    /**
     * @ORM\Column(type="datetime", nullable=true)
     */
    private $updatedAt;

    public function getId(): ?int
    {
        return $this->id;
    }

    public function getUsername(): ?string
    {
        return $this->username;
    }

    public function setUsername(?string $username): self
    {
        $this->username = $username;

        return $this;
    }

    public function getEmail(): ?string
    {
        return $this->email;
    }

    public function setEmail(?string $email): self
    {
        $this->email = $email;

        return $this;
    }
```

8. Créer, à l'aide d'annotations, dans le controller Front, une route pour la page équipe :
    h. Elle répondra sur l'url « /equipe».
    i. Elle portera le nom de « front_team »
    j. Elle retournera le contenu d'un Template Twig .
    k. Elle n'acceptera que la méthode GET
    l. Cette page listera l'ensemble des utilisateurs enregistrés dans la base de données. Sous la forme d'une liste : Nom/Prenom/Email/JobTitle.
    m. Ajouter sur la page d'accueil une ébauche de Menu Principal, contenant un lien vers la page équipe.
9. Créez votre premier controller :Admin**Controller**. En utilisant le Maker (`php bin/console make:controller`)
10. Créer, à l'aide d'annotations, dans le controller Front, un nouveau controller AdminController, et une route/action pour une page « de service » qui s'occupera d'insérer de nouveaux utilisateurs :
    n. Elle répondra sur l'url « /admin/equipe/inserer ».
    o. Elle portera le nom de « admin_team_insert »
    p. Elle n'acceptera que la méthode GET
    q. Cette action devra :
        i. Récupérer les paramètres fournis en GET : username, email, firstname, lastname, jobtitle et gérer des valeurs par défaut.
        ii. Insérer en base de données un nouvel User
    r. Une fois l'entité persistée, l'action route redirige l'utilisateur vers la page équipe.
11. Créer une entité App:Category en utilisant le maker (php bin/console make:entity)
    s. Elle contiendra les champs :
        - Id
        - Name (VARCHAR 255)
        - Image (VARCHAR 255, pour l'instant)
    t. La table en base de données sera : Category
    u. Renseigner en base de données les trois catégories suivantes :
        - Entrée
        - Plat
        - Dessert
12. Créer une entité App :Dish (ou Plat) en utilisant le maker (php bin/console make:entity)
    v. Elle contiendra les champs :
        - id,

- Name (VARCHAR 255)
- Calories (INT)
- Price (FLOAT 2 décimales)
- Image (VARCHAR 255, pour l'instant)
- Description (TEXT)
- Sticky (BOOL)
- Category (Association : Many to one vers Category)
- User (Association : Many to one vers User)

w. La table en base de données sera : Dish (ou Plat)

13. Créer, à l'aide d'annotations, dans le controller Front, une route/action pour la page « carte »

x. Elle répondra sur l'url « /carte ».

y. Elle portera le nom de « front_dishes »

z. Elle n'acceptera que la méthode GET

aa. Cette action devra :

- Lister les catégories récupérée dans la table Category, ainsi que le nombre de plats disponibles dans chacune.
- Proposer un lien vers le détail de la catégorie (voir 11 ci dessous)

14. Créer, à l'aide d'annotations, dans le controller Front, une route/action pour chaque catégorie de plat

bb. Elle répondra sur l'url « /carte/{id} ».

cc. Elle portera le nom de « front_dishes_category »

dd. Elle n'acceptera que la méthode GET et recoit en paramètre l'identifiant de la catégorie.

ee. Cette action devra :

- Vérifier l'existence de la category
- Renvoyer une Exception si la catégorie n'existe pas
- Afficher : Nom de la catégorie, liste des plats de la catégorie. Pour chacun : Nom / Prix / Calories
- Proposer un lien « retour » vers la liste des catégories (route front_dishes)

15. Créer une nouvelle Entité dans le bundle Front : Allergen

ff. Elle contiendra les champs :

- Id
- Name (VARCHAR 255)

b. Générer les Getter / Setter et la Structure Doctrine

2. Modifier les entités App :Dish et App :Allergen pour créer une relation ManyToMany

a. L'entité Dish devra alors avoir les méthode public qui permet de récupérer l'ensemble des allergènes contenus dans ce plat.

b. L'entité Allergen devra disposer du même type de méthode.

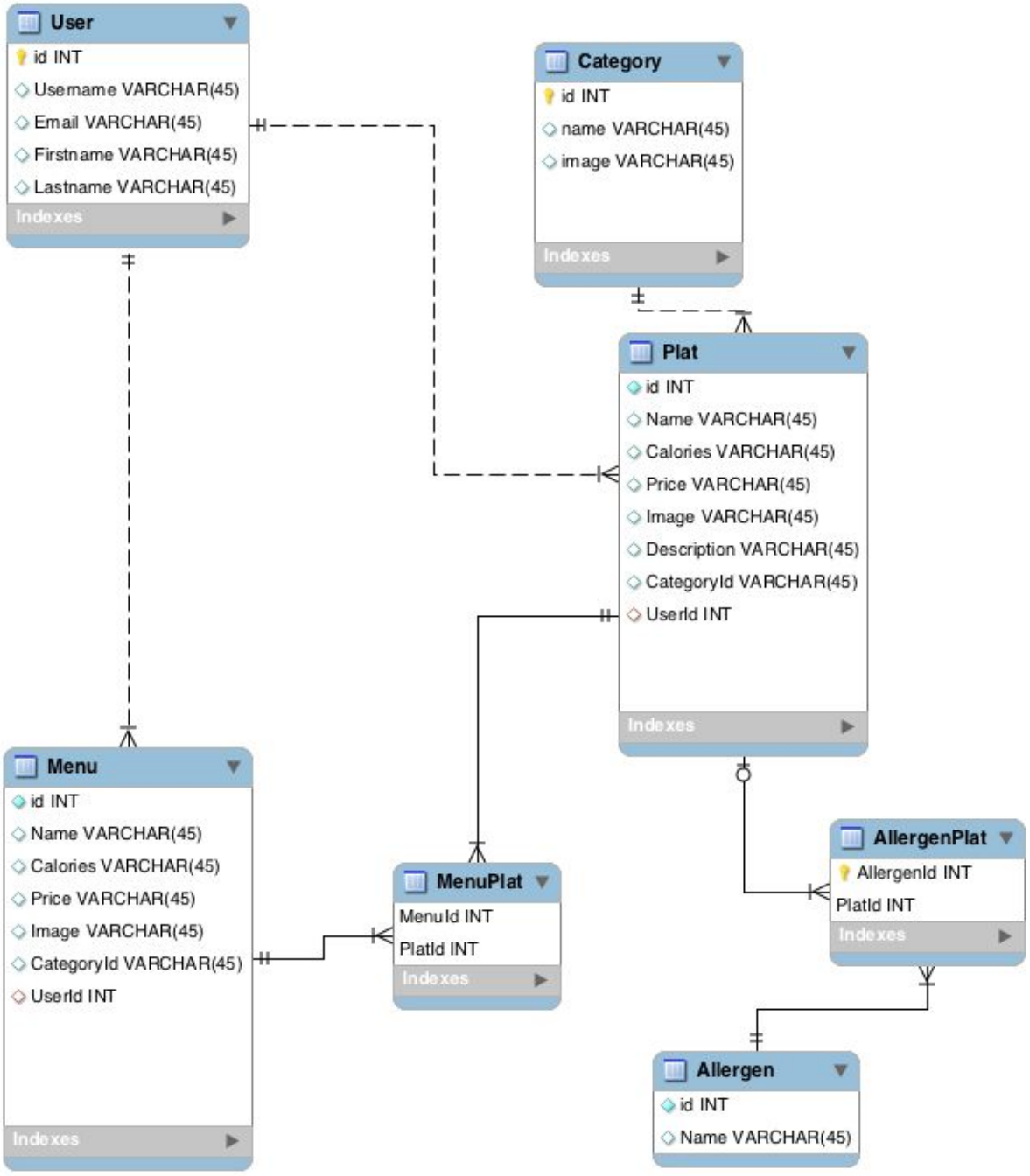
16. Sur le même principe que le point 6 : dans le controller App/Controller/AdminController créer une url de service qui se chargera d'insérer des plats et des allergènes.

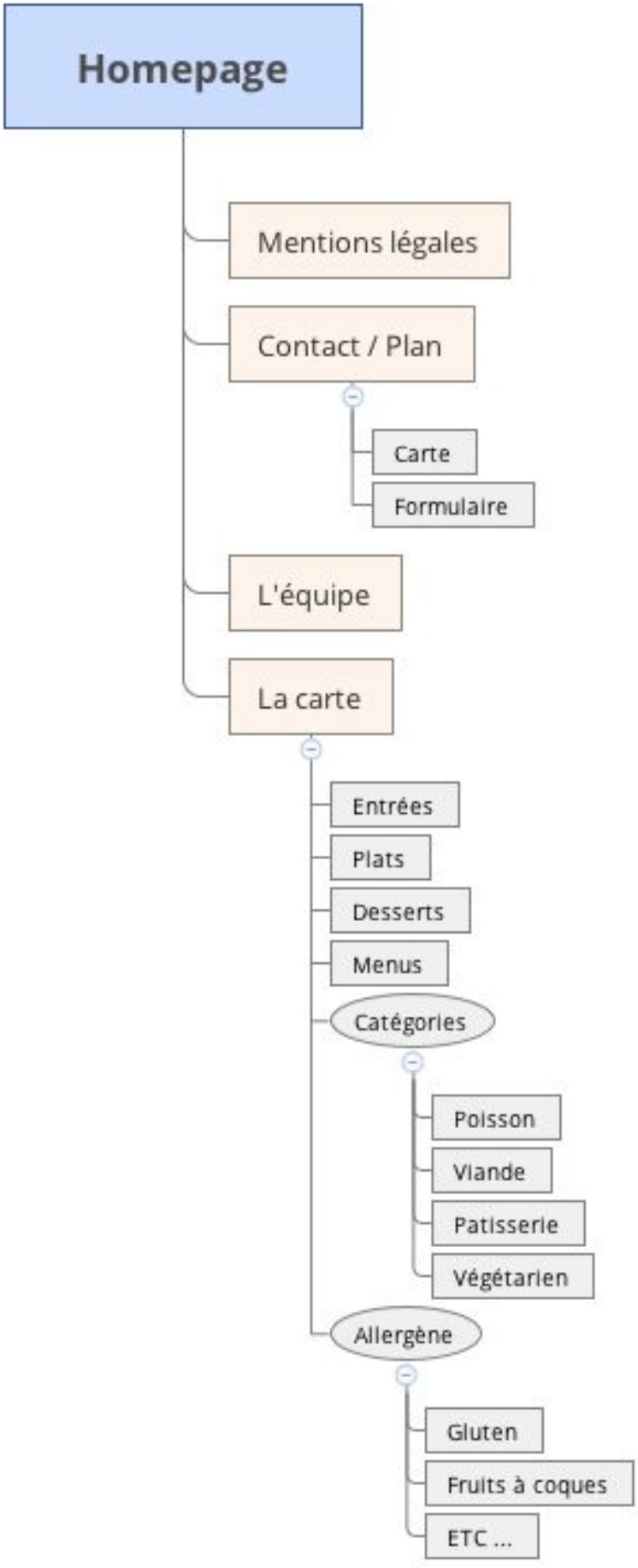
c. L'action chargera le fichier Json fourni

d. Elle insèrera en base de données les allergènes / plats qui n'existent pas, et updatera ceux qui existent.

e. A la fin du traitement renverra sur la page dishes

17. Ajouter sur les pages /carte/{id} la liste des allergènes contenus dans chaque plats


User
id INT
Username VARCHAR(45)
Email VARCHAR(45)
Firstname VARCHAR(45)
Lastname VARCHAR(45)
Indexes
Category
id INT
name VARCHAR(45)
image VARCHAR(45)
Indexes
Plat
id INT
Name VARCHAR(45)
Calories VARCHAR(45)
Price VARCHAR(45)
Image VARCHAR(45)
Description VARCHAR(45)
CategoryId VARCHAR(45)
UserId INT
Indexes
Menu
id INT
Name VARCHAR(45)
Calories VARCHAR(45)
Price VARCHAR(45)
Image VARCHAR(45)
CategoryId VARCHAR(45)
UserId INT
Indexes
MenuPlat
MenuId INT
PlatId INT
Indexes
AllergenPlat
AllergenId INT
PlatId INT
Indexes
Allergen
id INT
Name VARCHAR(45)


Homepage
Mentions légales
Contact / Plan
Carte
Formulaire
L'équipe
La carte
Entrées
Plats
Desserts
Menus
Catégories
Poisson
Viande
Patisserie
Végétarien
Allergène
Gluten
Fruits à coques
ETC ...