TP 8 : Filtrer la vue des recettes selon le type de plat

Dans ce TP, nous allons mettre en place un premier système de recherche afin de filtrer les données. Nous allons profiter aussi de ce TP pour encore optimiser le code en ajouter par composer la gestion des classes par l'autoloader fourni par ce même composer qui est un gestionnaire de dépendances. Cela permettra notamment une meilleure gestion des classes et de leur appel entre les fichiers PHP. (Voir par exemple : https://pad.numerique.gouv.fr/s/RD-NVpwUj#)

- Le fichier index.php commence à devenir illisible à cause du nombre de require_once :

```php
    // import de la classe RecetteController
    require_once(__DIR__.DIRECTORY_SEPARATOR.'src'.DIRECTORY_SEPARATOR.'Controllers'.DIRECT
ORY_SEPARATOR.'RecetteController.php');

    // import de la classe ContactController
    require_once(__DIR__.DIRECTORY_SEPARATOR.'src'.DIRECTORY_SEPARATOR.'Controllers'.DIRECT
ORY_SEPARATOR.'ContactController.php');

    // import de la classe UserController
    require_once(__DIR__.DIRECTORY_SEPARATOR.'src'.DIRECTORY_SEPARATOR.'Controllers'.DIRECT
ORY_SEPARATOR.'UserController.php');

    // import de la classe FavoriController
    require_once(__DIR__.DIRECTORY_SEPARATOR.'src'.DIRECTORY_SEPARATOR.'Controllers'.DIRECT
ORY_SEPARATOR.'FavoriController.php');

    // import de la classe CommentController
    require_once(__DIR__.DIRECTORY_SEPARATOR.'src'.DIRECTORY_SEPARATOR.'Controllers'.DIRECT
ORY_SEPARATOR.'CommentController.php');
```

J'ai mal à la tête…

- Pour éviter cela, nous allons paramétrer un **autoloader**, permettant de faciliter l'appel aux classes dont nous avons besoin à l'aide du mot clé **namespace**.
- Pour cela, nous avons besoin de Composer : https://getcomposer.org/download/
    - Sur le dossier de **La Cosina**, faire la commande **composer init** en ligne de comme (par le biais de VSCode ou par le terminal)
    - Choisir de nommer le package **app/r301**
    - Valider le reste des étapes sans rien modifier
    - Modifier le fichier **composer.json** afin d'intégsrer les espaces de noms Controller et Model basés sur les chemins **src/Controllers** et **src/Models**

```
{
    "name": "app/r301",
    "autoload": {
        "psr-4": {
            "App\\R301\\": "src/",
            "App\\R301\\Controller\\": "src/Controllers/",
            "App\\R301\\Model\\": "src/Models/"
        }
    },
```

Extrait de composer.json

- Ensuite, ouvrir un contrôleur par exemple **RecetteController**

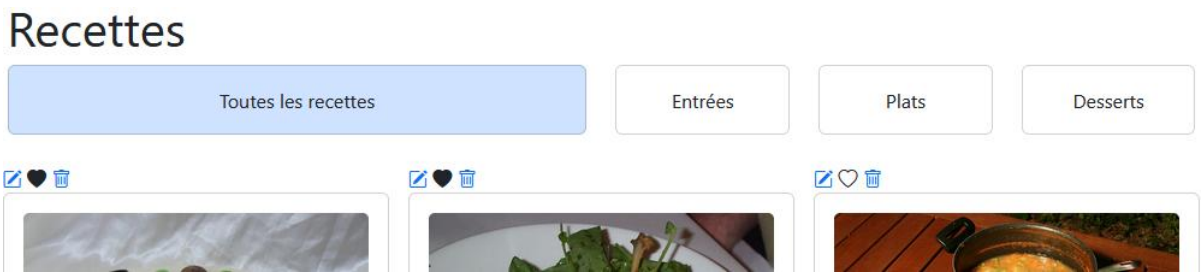
```
<?php

namespace App\R301\Controller;

class RecetteController {
```

- Modifier enfin le fichier **index.php** afin d'intégrer les nouvelles classes, en introduisant par le **require** de l'**autoload** :

```
    require 'vendor/autoload.php';

    use App\R301\Controller\RecetteController;
    use App\R301\Controller\ContactController;
    use App\R301\Controller\UserController;
    use App\R301\Controller\FavoriController;
    use App\R301\Controller\CommentController;

    // import de la classe RecetteController
    require_once(__DIR__.DIRECTORY_SEPARATOR.'src'.DIRECTORY_SEPARATOR.'Controllers'.DIRECTORY_SEPARATOR.'RecetteController.php');

    // import de la classe ContactController
    require_once(__DIR__.DIRECTORY_SEPARATOR.'src'.DIRECTORY_SEPARATOR.'Controllers'.DIRECTORY_SEPARATOR.'ContactController.php');

    // import de la classe UserController
    require_once(__DIR__.DIRECTORY_SEPARATOR.'src'.DIRECTORY_SEPARATOR.'Controllers'.DIRECTORY_SEPARATOR.'UserController.php');

    // import de la classe FavoriController
    require_once(__DIR__.DIRECTORY_SEPARATOR.'src'.DIRECTORY_SEPARATOR.'Controllers'.DIRECTORY_SEPARATOR.'FavoriController.php');

    // import de la classe CommentController
    require_once(__DIR__.DIRECTORY_SEPARATOR.'src'.DIRECTORY_SEPARATOR.'Controllers'
```

- Retournons à notre objectif initial du TP :
- Ajouter le champ **type_plat VARCHAR (100) NOT NULL** dans la table **recettes**
- Modifier le modèle en conséquence
- Modifier les formulaires d'ajout et de modification des recettes afin d'ajouter un **input** de type **select** avec 3 valeurs possibles
    - Valeur : entree (Entrée)
    - Valeur : plat (Plat)
    - Valeur : dessert (Dessert)
- Modifier aussi l'action d'enregistrement de la recette pour ajouter cette nouvelle donnée

- Modifier les recettes par le formulaire de modification :
    - Treipaïs : Dessert
    - Confit de canard : Plat
    - Soupe au pistou : Entrée
    - Tapenade : Entrée
    - Salade landaise : Entrée
- Modifier le fichier qui affiche les recettes (liste.php) :
    - Il faut ajouter un div avec des Cards Bootstrap, qui vont afficher les types de recettes :



    - Par défaut, le filtre **Toutes les recettes** doit être sélectionné (fond : classe = **bg-primary-subtle**)
    - Au passage de la souris sur les différents types de plats, le fond (classe = **bg-primary-subtle**) doit être mis et la souris doit devenir un pointer.
    - Au clic, la liste des recettes doit s'actualiser sans rafraîchir toute la page :
        - Mettre un **id** sur la **div** qui contient la liste des recettes (**listeRecettes**)
        - Dans le contrôleur **RecetteController.php**, il faut modifier l'action **index**, en prenant en compte un paramètre **filtre** (**$_GET['filtre']**), puis modifier la requête SQL en conséquence, si besoin, tester une URL avec le filtre **plat** : **?c=Recette&a=index&filtre=plat**.
        - Adapter le fichier **recipes.js** en conséquence à l'aide d'un **fetch**, dont le résultat doit être du **html** qu'il faut ensuite filtrer grâce au **Parser** afin de pouvoir cibler une **div** particulière :

```
// Mettre à jour la liste des recettes avec le filtre par fetch qui renvoie
une réponse en html
fetch('?c=Recette&a=index&filtre=' + filterValue)
.then(response => response.text()) // Récupère le texte de la réponse
.then(html => {
          // Parse le HTML pour créer un document DOM
          let parser = new DOMParser();
          let doc = parser.parseFromString(html, 'text/html');

          // Sélectionner le div avec une classe ou un ID spécifique
          let divContent = doc.querySelector('#listeRecettes'); // Par
exemple, un div avec l'ID "listeRecettes"

          // Change le contenu de la div listeRecettes avec le HTML récupéré
filtré sur l'id listeRecettes
          document.getElementById('listeRecettes').innerHTML =
divContent.innerHTML; // Affiche le HTML dans la div recipes
```

Extrait de code

- Une fois le résultat du filtre, remarquer que les évènements JS à l'intérieur du div (**listeRecettes**) ne fonctionnent plus (le fond, le clic sur la recette, le clic sur le cœur pour définir un favori), pour résoudre cela :
    - Créer une fonction JS qui contient les évènements du détail de la recette :

```
function actionsRecipe() {

        // Sélectionne toutes les recettes avec la classe 'recipefav'
        let recipefav = document.querySelectorAll('.recipefav');
...
}
```

Extrait de code

    - Y mettre les instructions et évènements liés
    - Faire appel à cette fonction après le chargement du **DOM** et aussi après la mise à jour de la div **listeRecettes**.