TP 3 : Création de la vue en liste des recettes

- Créer les recettes suivantes :
    - Titre : Treipaïs
      Description : Le « Treipaïs » est une spécialité pâtissière du Limousin créé par un groupement de pâtissiers locaux. Il est composé de trois textures et de trois parfums : un fond de feuilletine, une mousse aux châtaignes, une mousse au chocolat noir, un biscuit à la noisette et un glaçage au chocolat, surmonté d'un « marron » glacé et de deux feuilles vertes en pâte d'amandes.
      Auteur : contact@lacosina.fr
    - Titre : Confit de canard
      Description : La viande est cuite pendant au moins 2 h dans la graisse chaude, entre 70 et 85 °C, puis mise en bocaux et recouverte de graisse, de telle sorte que l'air ne puisse pas entrer en contact avec elle et la détériorer.
      Auteur : contact@lacosina.fr
    - Titre : Soupe au pistou
      Description : La soupe au pistou (sopa au písto, soupo au pístou) est une soupe aux légumes d'été, avec des pâtes, servie avec du pistou, un mélange d'ail, d'huile d'olive et de basilic haché. Le terme pistou désigne, en provençal, le pilon du mortier qui sert à faire la préparation, et non pas le basilic qui se dit baseli.
      Auteur : contact@lacosina.fr
    - Titre : Tapenade
      Description : La tapenade (tapenada, tapenado) est une recette de cuisine provençale, mise au point en 1880 par un chef-cuisinier de Marseille. Elle est principalement constituée d'olives broyées, d'anchois, de thon mariné, d'herbes et de câpres (tapena en occitan, d'où son nom).
      Auteur : contact@lacosina.fr
- Vérifier la présence des données dans la base de données
- Créer le menu **Recettes** entre Accueil et Contact
- Créer le contrôleur **listerController.php** qui permet de lister toutes les recettes

```php
<?php


    // préparation de la requête d'insertion dans la base de données


    /** @var PDO $pdo **/
    $requete = $pdo->prepare("SELECT * FROM recettes");

    // exécution de la requête et récupération des données
    $requete->execute();
    $recipes = $requete->fetchAll(PDO::FETCH_ASSOC);

    require_once(__DIR__.DIRECTORY_SEPARATOR.'..'.DIRECTORY_SEPARATOR.'Views'.
DIRECTORY_SEPARATOR.'liste.php');
```

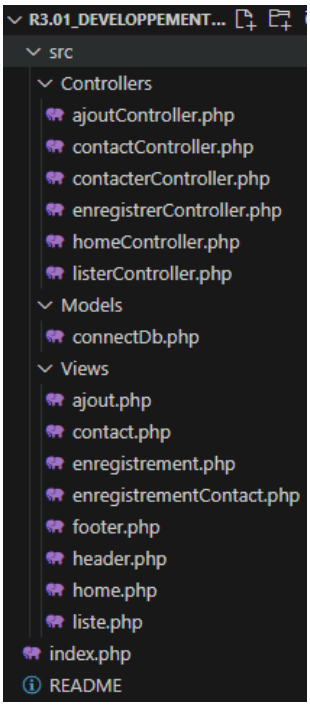
- Créer la vue **liste.php** qui affiche les listes des recettes

```php
<body>


    <h1>Recettes</h1>


    <div>
        <?php foreach ($recipes as $recipe) : ?>
            <div>
                <h2><?php echo $recipe['titre']; ?></h2>
                <p><?php echo $recipe['description']; ?></p>
                <a href="mailto:<?php echo $recipe['auteur']; ?>"><?php echo
$recipe['auteur']; ?></a>
            </div>
        <?php endforeach; ?>
    </div>

    <a href="?c=home" class="btn btn-primary">Retour à l'accueil</a>


</body>
```

- On va améliorer le visuel de la liste des recettes grâce à des « cards » Bootstrap : https://getbootstrap.com/docs/5.3/components/card/

```php
    <div class="row">
      <!-- Boucle permettant de lister les recettes -->
      <?php foreach ($recipes as $recipe) : ?>
            <div class="col-4 p-2">
                <!-- Utilisation des Cards Bootstrap -->
                <div class="card">
                    <div class="card-body">
                        <h2 class="card-title"><?php echo
$recipe['titre']; ?></h2>
                        <p class="card-text"><?php echo
$recipe['description']; ?></p>
                        Auteur : <a href="mailto:<?php echo
$recipe['auteur']; ?>"><?php echo $recipe['auteur']; ?></a>

                    </div>
                </div>
            </div>
        <?php endforeach; ?>
    </div>
```

- Le projet s'agrandit et le nombre de fichiers aussi :



Liste des fichiers du projet

- Pour avoir une meilleure visibilité des éléments du projet, nous allons regrouper certains éléments :
    - Les vues :
        - Nous allons organiser les vues par « entité » ou type d'éléments, dans notre cas, les vues associées aux recettes, aux contacts notamment
        - Une fois les dossiers créés, déplacer les fichiers et modifier les fichiers impactés
    - Les contrôleurs :
        - Nous allons organiser les contrôleurs en les transformant en classes, ainsi tout ce qui concerne les recettes sera dans une classe RecetteController.php, et de même pour ContactController.php
        - Créer le fichier **RecetteController.php** avec 3 fonctions : ajouter, enregistrer et lister

```php
<?php

class RecetteController {

    // Fonction permettant d'ajouter une nouvelle recette
    function ajouter() {
    }

    // Fonction permettant d'enregistrer une nouvelle recette
    function enregistrer() {
    }

    // Fonction permettant de lister les recettes
    function lister() {
    }

}
```

- Nous allons maintenant déplacer le code des fichiers correspondants dans cette classe

```php
<?php

class RecetteController {

    // Fonction permettant d'ajouter une nouvelle recette
    function ajouter() {
        require_once __DIR__ . DIRECTORY_SEPARATOR . '..' .
DIRECTORY_SEPARATOR . 'Views' . DIRECTORY_SEPARATOR . 'Recette' .
DIRECTORY_SEPARATOR . 'ajout.php';
    }
```

- Faire de même avec les autres fonctions (attention : il y a un piège)
- Supprimer les fichiers **ajoutController.php**, **enregistrerController.php**, et **listerController.php**
- Ensuite, modifier le fichier **index.php** pour importer cette classe, nous allons le faire au début

```php
<?php

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

    // connexion à la base de données
    require_once(__DIR__.DIRECTORY_SEPARATOR.'src'.DIRECTORY_SEPARATOR.'Models
'.DIRECTORY_SEPARATOR.'connectDb.php');
```

- De plus, actuellement, le routeur est basé sur un seul nom, exemple : **c=lister** renvoie vers la liste des recettes, mais comment ne pas le confondre avec la liste des contacts ou comment s'en rappeler ? Pas facile ! Même remarque que pour l'enregistrement des recettes et des contacts.
- Nous allons donc passer, par un système double : **c=Recette&a=lister**, l'attribut **c** renvoie vers le contrôleur et **a** renvoie vers la fonction ou l'action associée
- Prenons par exemple les actions liées aux recettes :
    - Commencer par modifier le fichier **index.php**

```php
    // mise en place de la route actuelle
    $controller = isset($_GET['c'])? $_GET['c'] : 'home';
    $action = isset($_GET['a'])? $_GET['a'] : 'index';

    // définition des routes disponibles
    switch ($controller) {
...
        // routes pour la gestion des recettes
        case 'Recette':
            $recetteController = new RecetteController();
            switch ($action) {
                case 'index':
                    $recetteController->index($pdo);
                    break;
                case 'ajouter':
                    $recetteController->ajouter();
                    break;
                case 'enregistrer':
                    $recetteController->enregistrer($pdo);
                    break;
```

Extrait de code (renommer l'action lister par index)

- Modifier les fichiers liés :
  - Le fichier **header.php** avec le menu de navigation

```php
<li class="nav-item">
                <a class="nav-link" href='?c=home'>Accueil</a>
            </li>
            <li class="nav-item">
                <a class="nav-link" href='?c=Recette&a=index'>Recettes</a>
            </li>
            <li class="nav-item">
                <a class="nav-link" href='?c=Contact&a=ajouter'>Contact</a>
            </li>
```

Extrait du menu

  - Les fichiers qui ont un lien avec les recettes :

```php
<form action="?c=Recette&a=enregistrer" method="post" enctype="multipart/form-data">
```

Exemple de modification dans le fichier d'ajout de recette

  - Créer le fichier **ContactController.php** et effectuer les mêmes opérations

```php
// routes pour la gestion des contacts
case 'Contact':
      $contactController = new ContactController();
      switch ($action) {
          case 'ajouter':
              $contactController->ajouter();
              break;
          case 'enregistrer':
              $contactController->enregister($pdo);
              break;
```

Gestion des contacts

- Enfin, les modèles :
  - Nous allons transformer le fichier connectDb.php en une classe que nous appellerons Database.php

```php
<?php
Class Database {

    private $host = 'localhost';
    private $db_name = 'lacosina';
    private $username = 'root';  // changez si nécessaire
    private $password = '';      // changez si nécessaire
    private $conn;

    public function getConnection() {
        $this->conn = null;
        try {
            $this->conn = new PDO("mysql:host=" . $this->host . ";dbname=" .
$this->db_name, $this->username, $this->password);
            $this->conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
        } catch (PDOException $exception) {
            echo "Erreur de connexion : " . $exception->getMessage();
        }
        return $this->conn;
    }
}
```

La classe Database

- Continuer avec la classe **Recette**

```php
<?php

require_once __DIR__ . 'Database.php';

Class Recette {
    private $conn;

    // Constructeur
    function __construct() {
        $database = new Database();
        $this->conn = $database->getConnection();
    }
```

- Le constructeur de la classe permet d'avoir la connexion à la base de données
- Les fonctions **find..** que l'on va créer nous permettront de créer les requêtes de sélection souhaitées
    - **findAll** pour lister toutes les recettes

```php
    public function findAll() {
        $query = "SELECT * FROM recettes";
        $stmt = $this->conn->prepare($query);
        $stmt->execute();
        return $stmt->fetchAll(PDO::FETCH_ASSOC);
    }
```

    - **find** pour une recette avec $id comme argument

```php
    public function find($id) {
        $query = "SELECT * FROM recettes WHERE id = '$id'";
        $stmt = $this->conn->prepare($query);
        $stmt->execute();
        return $stmt->fetch(PDO::FETCH_ASSOC);
    }
```

- **findBy** pour une recherche multicritère avec un tableau $params comme argument

```
public function findBy(array $params) {
        $query = "SELECT * FROM recettes WHERE ". implode(' AND ', 
array_map(function($key) {
            return "$key = :$key";
        }, array_keys($params)));

        $stmt = $this->conn->prepare($query);
        foreach ($params as $key => $value) {
            $stmt->bindValue(":$key", $value);
        }
        $stmt->execute();
        return $stmt->fetchAll(PDO::FETCH_ASSOC);
    }
```

  - Il reste à ajouter 3 fonctions : ajout, modification et suppression

```
    public function add($titre, $description, $auteur, $image) {
        $query = "INSERT INTO recettes (titre, description, auteur, image, date_creation) 
VALUES (:titre, :description, :auteur, :image , NOW())";
        $stmt = $this->conn->prepare($query);
        $stmt->bindParam(':titre', $titre);
        $stmt->bindParam(':description', $description);
        $stmt->bindParam(':auteur', $auteur);
        $stmt->bindParam(':image', $image);
        $stmt->execute();
        return $this->conn->lastInsertId();
    }

    public function update($id, $titre, $description, $auteur, $image) {
        $query = "UPDATE recettes SET titre = :titre, description = :description, auteur = 
:auteur, image = :image WHERE id = :id";
        $stmt = $this->conn->prepare($query);
        $stmt->bindParam(':id', $id);
        $stmt->bindParam(':titre', $titre);
        $stmt->bindParam(':description', $description);
        $stmt->bindParam(':auteur', $auteur);
        $stmt->bindParam(':image', $image);
        return $stmt->execute();
    }

    public function delete($id) {
        $query = "DELETE FROM recettes WHERE id = :id";
        $stmt = $this->conn->prepare($query);
        $stmt->bindParam(':id', $id);
        $stmt->execute();
        return $stmt->rowCount() > 0;
    }
```

  - Il reste à intégrer cette classe dans le contrôleur et modifier le code afin d'intégrer les modifications apportées par la classe Recette

```
// connexion à la base de données
require_once(__DIR__.DIRECTORY_SEPARATOR.'..'.DIRECTORY_SEPARATOR.'Models'.DIR
ECTORY_SEPARATOR.'Recette.php');
```

```php
    private $recetteModel;

    public function __construct() {
        $recetteModel = new Recette();
        $this->recetteModel = $recetteModel;
    }
```

Au début de la classe, il faut initialiser le modèle Recette

```php
$recipes = $this->recetteModel->findAll();
```

```php
$recipe = $this->recetteModel->find($_GET['id']);
```

Effectuer tous les remplacements nécessaires

- Il faut ensuite enlever l'appel à PDO ou $pdo qui sont obsolètes.
- Puis, il faut modifier le fichier index pour enlever l'appel à connectDb.php pour laisser uniquement l'appel à RecetteController.php
- Faire de même avec le modèle Contact
- Enfin supprimer le fichier connectDb.php qui n'a plus d'utilité.