SIMPLON
RÉGION SUD

RAPPORT DE PROJET POUR LE TITRE PROFESSIONEL DÉVELOPPEUR WEB & WEB MOBILE

Salvatore Matraxia

Table of Contents

Presentation

Dans le cadre de ma reconversion professionnelle, je vous présente le résultat de mes efforts et de ma passion : mon blog de recettes de cuisine. Ce projet a vu le jour pendant ma formation en développement web, où j'ai acquis un ensemble complet de compétences, allant de HTML, CSS et JavaScript à PHP et MySQL, tout en développant une compréhension approfondie de l'architecture MVC.

Ce blog est bien plus qu'une simple démonstration de mes compétences techniques. Il s'agit d'une plateforme où je peux partager ma passion pour la cuisine avec d'autres passionné de la gastronomie. Au fil des années, j'ai appris à apprécier la diversité des saveurs, des techniques et des cultures culinaires, et ce blog est ma manière de donner vie à cette passion.

Dans ce rapport, je vais détailler chaque étape de la création de mon blog, que j'ai nommé "Foodies". Nous allons commencer par le processus de conception initial, en expliquant comment j'ai créé les maquettes initiales et les plans pour l'interface utilisateur. Ensuite, nous plongerons dans le développement, en mettant l'accent sur la création d'une interface utilisateur interactive et dynamique.

De plus, je vais aborder la création de la base de données qui alimente le blog, ainsi que le développement des fonctionnalités permettant d'accéder et de gérer les données. Mon objectif principal est de vous montrer comment j'ai acquis les compétences nécessaires pour répondre aux exigences de la création de ce blog, en accord avec les principes du REAC (Référentiel Emploi Activités Compétences).

Cahiers des charges

Le cahier des charges, couramment abrégé en CDC, revêt une importance cruciale dans le contexte du développement d'un projet. Il constitue un document essentiel, servant de guide fondamental pour définir les besoins et les spécifications, c'est-à-dire les éléments et les règles, associés à ce projet. L'objectif principal du CDC est de clarifier et de rendre accessibles à tous les détails et les contours du projet, tout en établissant un cadre strict pour sa réalisation. Il s'agit d'un document destiné à être partagé tant en interne qu'en externe, avec toutes les parties impliquées. Souvent, il fait office de contrat lorsqu'il s'agit de formaliser la relation entre un client et un fournisseur.

Demande du client

Apres une reconversion professionnelle, un ancien cuisinier souhaite développer son propre site internet, où il pourra partager ses propres recettes avec d'autres utilisateurs qui pourront également partager les leurs.

Le principal objectif de ce projet est de créer un site web interactif et convivial qui permettra aux utilisateurs de partager, découvrir et cuisiner des recettes de cuisine. Le site s'addres aux amateurs de cuisine, tout en permettant à l'administrateur (l'ancien cuisinier) de partager ses propres créations culinaires.

Fonctionnalité

Systeme de partage de recettes: Les utilisateurs devraient pouvoir publier leurs propres recettes en fournissant des détails tels que les ingrédients, les étapes de préparation, le temps de cuisson, le niveau de difficulté, etc.

Systeme de recherche: Les visiteurs doivent pouvoir rechercher des recettes en fonction de différents critères, tels que le type de plat, les ingrédients, la cuisine régionale, etc.

Commentaires: Les utilisateurs devraient avoir la possibilité de laisser des commentaires et des notes sur les recettes qu'ils ont essayées, créant ainsi une communauté interactive

Moderation des contenus: Un système de modération pour s'assurer que les recettes partagées sont appropriées et de qualité.

Contraintes techniques

Responsive design: Le site doit être compatible avec les appareils mobiles et tablettes pour une expérience utilisateur optimale.

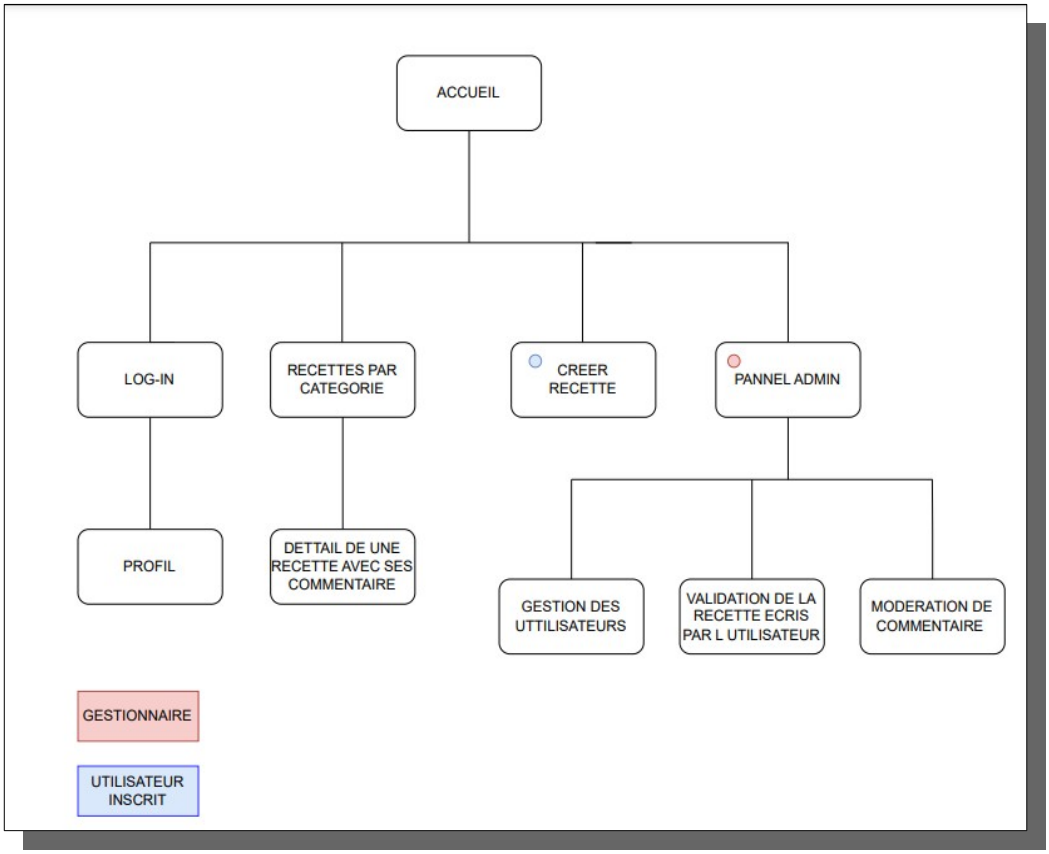
Gestion de contenu facile: Un système de gestion de contenu pour l'administrateur afin de faciliter la publication de nouvelles recettes et de contenu.

Sécurité : Étant donné que les utilisateurs fourniront des informations sensibles, comme leur adresse mail et leur mot de passe, il est primordial que le site soit sécurisé. Cela inclut l'utilisation de connexions sécurisées, la protection contre les attaques courantes et la mise en place de bonnes pratiques en matière de gestion des mots de pas

Conception

Arborescence

L'arborescence du site web décrit la structure hiérarchique de ses pages. Une légende indique si l'accès à une page est restreint à un rôle.



Maquettage

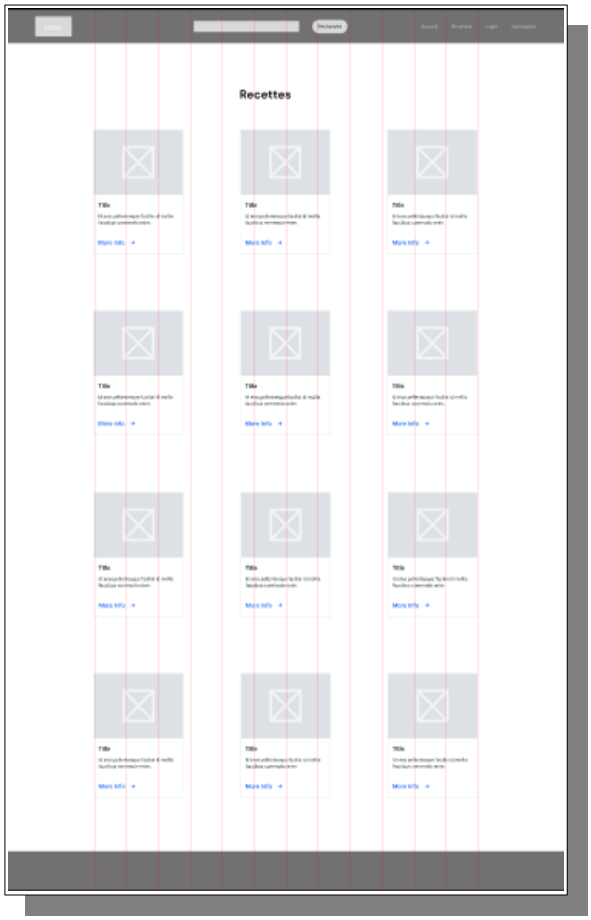
Le maquettage, dans le contexte du développement web et de la conception d'interfaces utilisateur, fait référence au processus de création de maquettes ou de prototypes visuels qui servent de représentations visuelles préliminaires d'un site web, d'une application mobile ou d'un produit numérique.

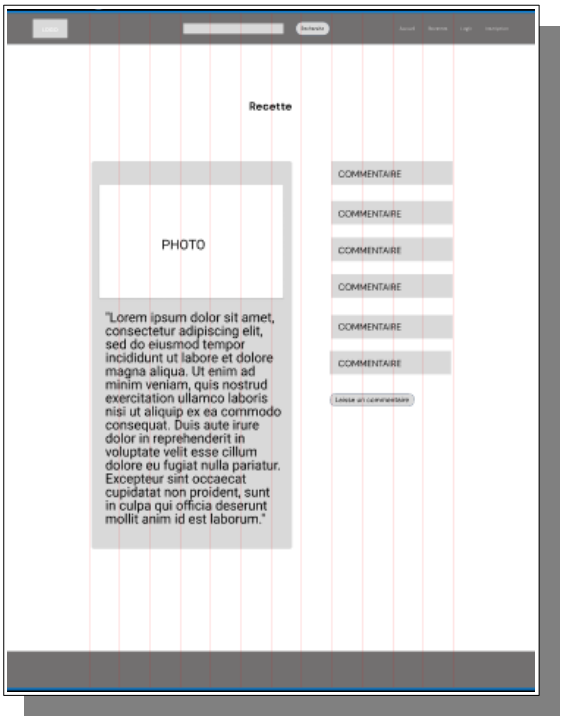
Zoning

Une maquette de zoning, également appelée wireframe, est une représentation visuelle simplifiée de l'interface utilisateur d'un site web ou d'une application. Elle est utilisée au début du processus de développement web pour planifier la disposition générale des

éléments sur une page sans se préoccuper des détails de conception graphique tels que les couleurs, les images ou les polices.

Une maquette de zoning se concentre principalement sur la structure et l'agencement des éléments, ce qui permet de définir l'emplacement des différents composants tels que les menus de navigation, les bouttons, les zones de texte, les images, etc. Elle est généralement composée de formes géométriques simples, de lignes et de textes de base pour représenter ces éléments.

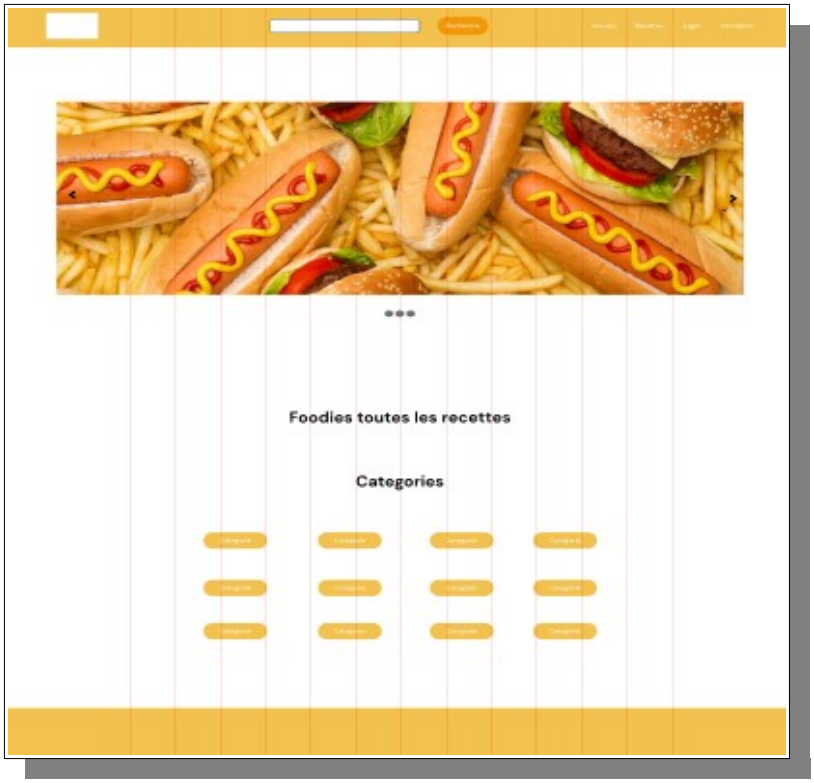


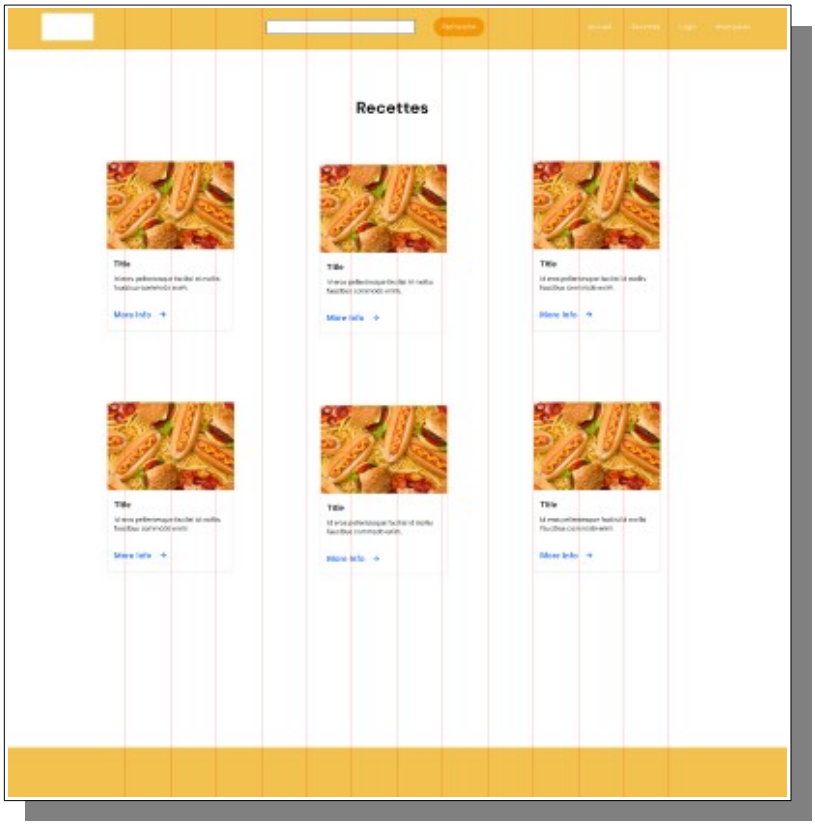


Maquette dynamique

La maquette dynamique permet aux parties prenantes de visualiser l'interface utilisateur de manière interactive. Au lieu de simplement voir des captures d'écran statiques, les utilisateurs peuvent interagir avec les éléments tels que les boutons, les menus, les liens, etc., ce qui offre une meilleure compréhension de l'expérience utilisateur.

Pour réaliser ma maquette, j'ai utilisé l'outil Figma.

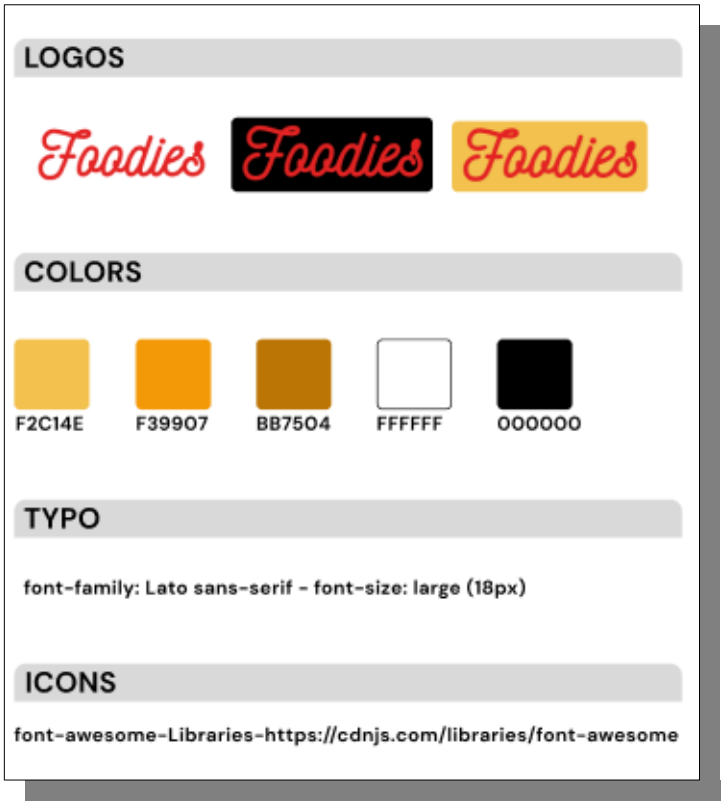

Foodies toutes les recettes
Categories


Recettes
Title
More Info

Charte graphique

La charte graphique, est le document qui définit l'ensemble des éléments visuels et graphiques utilisés pour représenter le projet. Elle sert de référence pour assurer une cohérence visuelle sur le site web.

La charte contient le logo, la palette de couleurs, la typographie, les images et le graphisme



BDD (base de données)

La conception du modèle de données est une étape déterminante dans tout projet de développement web impliquant le stockage et la manipulation des données. Pour mon site, j'ai conçu un modèle de données utilisant MySQL comme système de gestion.

Dictionnaire de données

Le modèle de données comprend plusieurs tables, chaque table représentant un type d'information spécifique nécessaire pour le fonctionnement du site. Voici une description de chaque table :

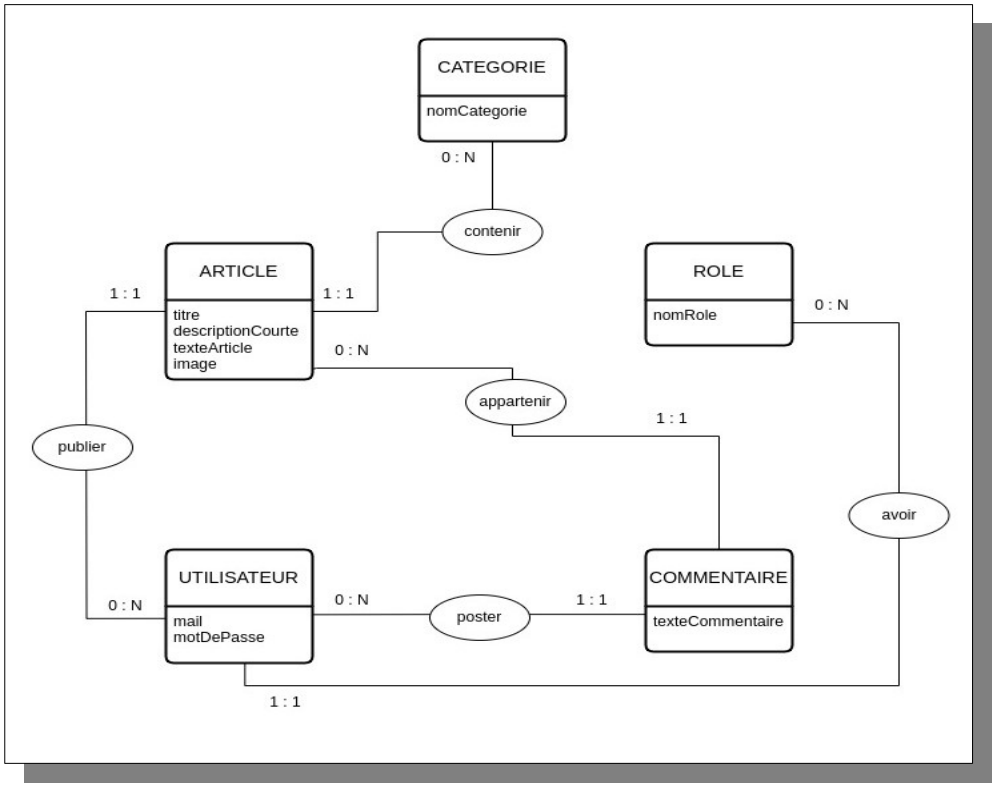
Entités et Attributs

ENTITEE	ATTRIBUE	DESCRIPTION
Article	titre descrpiton courte texte article image	Titre de l article Description de l article Le texte de l article Image de l article
Commentaire	texte commentaire	Texte des commentaires ecris par les utilisatuers
Utilisateur	nom prenom mot de passe e-mail	Nom de l utilisateur Prenom de l utilisateur Mot de passe choisi par l utilisatuer E-mail de l utilisateur
Role	membre gestionnaire	Definiton du role, gestionnaire ou utilisateur

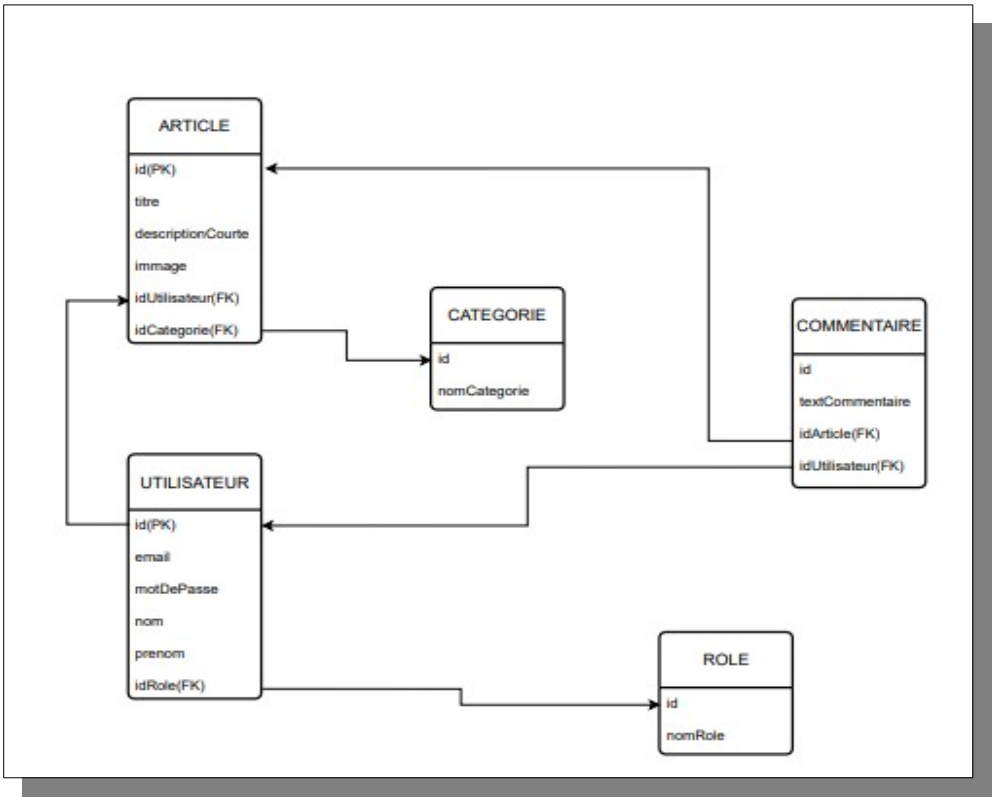
Relations et Cardinalités

Un ARTICLE est écrit par un UTILISATEUR (-) : Chaque article est associé à un seul utilisateur qui l'a écrit.
Un UTILISATEUR peut écrire plusieurs ARTICLES (0-N) : Un utilisateur peut écrire aucun, un ou plusieurs articles.
Un COMMENTAIRE est écrit par un UTILISATEUR (-) : Chaque commentaire est associé à un seul utilisateur qui l'a écrit.
Un UTILISATEUR peut écrire plusieurs COMMENTAIRES (0-N) : Un utilisateur peut écrire aucun, un ou plusieurs commentaires.
Un COMMENTAIRE concerne un ARTICLE (1-1) : Chaque commentaire est associé à un seul article auquel il se rapporte.
Un ARTICLE peut être concerné par plusieurs COMMENTAIRES (0-N) : Un article peut ne pas avoir de commentaires ou en avoir plusieurs.
Un UTILISATEUR possède un ROLE (-) : Chaque utilisateur a un rôle attribué.
Un ROLE peut être attribué à plusieurs UTILISATEURS (-N) : Un rôle peut être attribué à aucun, un ou plusieurs utilisateurs.

MCD


CATEGORIE
nomCategorie
0 : N
contenir
ARTICLE
titre
descriptionCourte
texteArticle
image
1 : 1
1 : 1
0 : N
ROLE
nomRole
0 : N
appartenir
1 : 1
publier
avoir
UTILISATEUR
mail
motDePasse
0 : N
0 : N
poster
1 : 1
COMMENTAIRE
texteCommentaire
1 : 1

MLD


ARTICLE
id(PK)
titre
descriptionCourte
immage
idUtilisateur(FK)
idCategorie(FK)
CATEGORIE
id
nomCategorie
COMMENTAIRE
id
textCommentaire
idArticle(FK)
idUtilisateur(FK)
UTILISATEUR
id(PK)
email
motDePasse
nom
prenom
idRole(FK)
ROLE
id
nomRole

MPD

```sql
-- Supprime la base de données si elle existe déjà
DROP DATABASE IF EXISTS `projet_perso`;

-- Crée la base de données
CREATE DATABASE IF NOT EXISTS `projet_perso`;

USE `projet_perso`;

-- CREATION TABLE ARTICLE
CREATE TABLE article (
    id bigint(25) NOT NULL AUTO_INCREMENT PRIMARY KEY,
    titre varchar(50) NOT NULL,
    descriptionCourte varchar(300) NOT NULL,
    textArticle text NOT NULL,
    immage VARBINARY(50) NOT NULL,
    idUtilisateur bigint(20) NOT NULL,
    idCategorie bigint(20) NOT NULL,
    difficulte int(2) NOT NULL,
    dateHeure timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
    pubblication boolean DEFAULT false
);

-- ...

-- CREATION TABLE CATEGORIE
CREATE TABLE categorie (
    id bigint(25) NOT NULL AUTO_INCREMENT PRIMARY KEY,
    nomCategorie varchar(50) NOT NULL
);

ALTER TABLE article ADD CONSTRAINT `fk_article_utilisateur` FOREIGN KEY (idUtilisateur) REFERENCES utilisateur(id);
ALTER TABLE article ADD CONSTRAINT `fk_article_categorie` FOREIGN KEY (idCategorie) REFERENCES categorie(id);
ALTER TABLE commentaire ADD CONSTRAINT `fk_commentaire_article` FOREIGN KEY (idArticle) REFERENCES article(id);
ALTER TABLE commentaire ADD CONSTRAINT `fk_commentaire_utilisateur` FOREIGN KEY (idUtilisateur) REFERENCES utilisateur(id);
ALTER TABLE utilisateur ADD CONSTRAINT `fk_utlisateur_role` FOREIGN KEY (idRole) REFERENCES role(id);

```

Réalisation

Environnement de travail

Server

Server APACHE : XAMMPP (linux) , mySQL, PHP

Éditeur de texte

Vscode

Technologies

Pour développer mon site web, j'ai utilisé les technologies suivantes : HTML, CSS, PHP, SQL et JavaScript dans un architecture MVC.

HTML (Hyper Text Markup Language)

Est le langage de balisage standard utilisé pour créer la structure d'une page web. Il permet de définir la structure et le contenu de la page .

CSS (Cascading Style Sheets)

Utilisé pour la mise en page et la présentation du site web. Il permet de définir la couleur, la police, la taille, la disposition et d'autres styles visuels pour les éléments HTML.

JavaScript

Langage de programmation côté client qui permet d'ajouter de l'interactivité au site web. Utilise' pour créer des fonctionnalités dynamiques telles que des formulaires interactifs, des carrousels d'images, des menus déroulants, des effets de transition, etc.

PHP

Est un langage de programmation côté serveur utilisé pour le développement web dynamique. Il permet de créer des pages web interactives en générant du contenu dynamiquement, en se connectant à des bases de données (comme MySQL) et en traitant les formulaires soumis par les utilisateurs.

SQL

SQL (Structured Query Language) est un langage de programmation spécialement conçu pour la gestion des bases de données relationnelles. Il est utilisé pour créer, manipuler et interroger des bases de données.

Architecture du projet

MVC

MVC signifie "Modèle, Vue, Contrôleur", c'est un patron de conception concernant l'agencement du code. Le code est segmenté selon ces trois sections : le modèle contient le code qui gère la logique métier, la vue celui qui gère l'affichage, et le contrôleur gère le lien avec l'utilisateur. Favorise la séparation des taches, ce qui signifie que chaque composant a une responsabilité spécifique et ne se mélange pas avec les autres. Cela facilite la maintenance, l'extensibilité et la réutilisation du code.

Voici une brève explication de chaque composant :

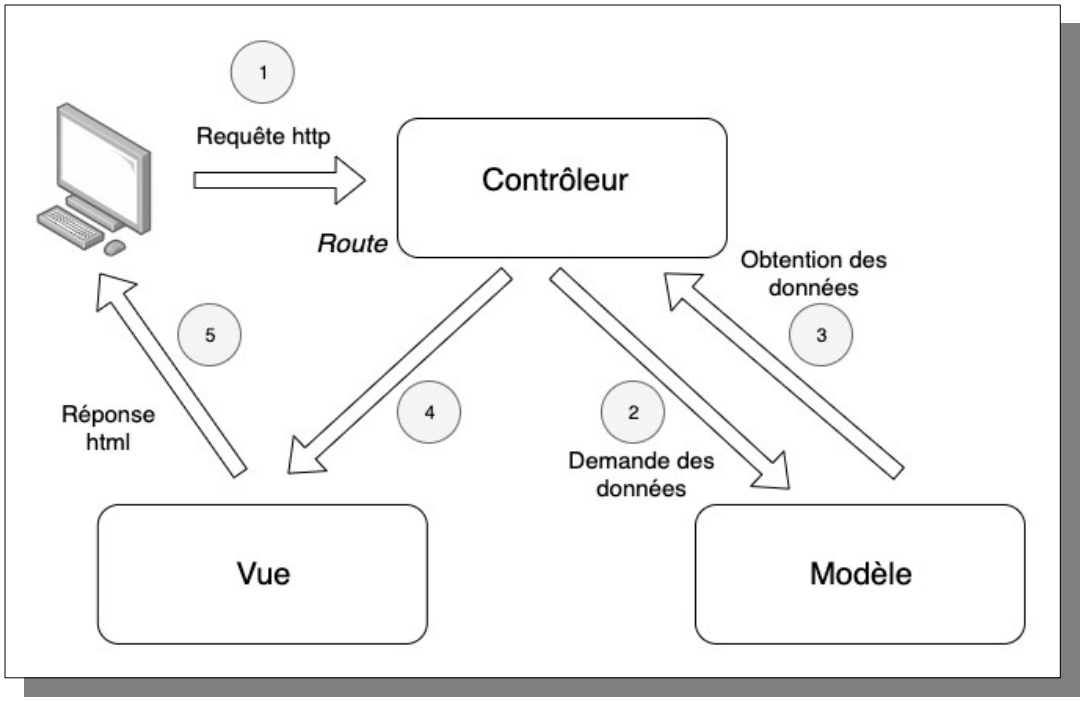
Model

Le modèle représente la couche de données de l'application. Il est responsable de la gestion des données, de la logique métier et des interactions avec la base de données. Les données

stockées dans le modèle peuvent inclure des informations des utilisateurs, des articles, des commentaires, etc.

View
La vue représente l'interface utilisateur de l'application. Elle est responsable de l'affichage des données au format HTML ou d'un autre langage de présentation. Les vues sont chargées de rendre les données du modèle de manière compréhensible pour l'utilisateur.

Controller
Le contrôleur agit comme un intermédiaire entre le modèle et la vue. Il gère les requêtes de l'utilisateur, traite les entrées de l'utilisateur, effectue les opérations nécessaires sur le modèle et choisit la vue appropriée pour afficher les résultats. Le contrôleur est responsable de la gestion de la logique de flux et de la coordination entre le modèle et la vue.



Connexion à la BDD

La connexion à une base de données en PHP est généralement effectuée en utilisant la bibliothèque PDO.

```php
<?php

/** Configuration de l'application. */
class Config
{
    /** Paramètres de connexion à la base de données. */
    const db = [
        'host' => '127.0.0.1',
        'port' => 3306,
        'dbname' => 'projet_perso',
        'user' => 'root',
        'password' => ''
    ];
}
```

```php
static function getPDO()
   {
       self::log()->info(__FUNCTION__);

       $host = Config::db['host'];
       $port = Config::db['port'];
       $dbname = Config::db['dbname'];
       $user = Config::db['user'];
       $password = Config::db['password'];
       $dataSourceName = "mysql:host=$host;port=$port;dbname=$dbname";
       $pdo = new PDO($dataSourceName, $user, $password);

       $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

       return $pdo;
   }
```

La fonction `getPDO` est une methode statique qui permet de creer et de configurer une instance de la classe PDO pour établir une connexion à la bdd MySQL en utilisant les informations de configuration stockees dans un tableau `Config::db`

Elle utilise `self::log()->info(__FUNCTION__)` pour enregistrer des informations de journalisation (logs).

Elle extrait les informations de configuration de la base de donnees a partir du tableau `Config::db` et les stocke dans des variables distinctes pour une utilisation ulterieure,

et construit une chaîne de connexion PDO appelée `$dataSourceName` en utilisant les informations extraites de la configuration. Cette chaîne de connexion contient l'hôte, le port et le nom de la base de données au format requis par PDO.

Ensuite, elle cree une instance de la classe PDO en utilisant la chaîne de connexion, l'utilisateur et le mot de passe fournis.

La méthode `setAttribute` est utilisee pour configurer le mode de gestion des erreurs de l'instance PDO. Ici, elle est configurée pour générer des exceptions PDO (`PDO::ERRMODE_EXCEPTION`) en cas d'erreur, ce qui facilite la gestion des erreurs de base de données.

Enfin, l'instance configurée de PDO est retournée par la méthode, ce qui permet de l'utiliser pour effectuer des opérations sur la base de données.

Exemples des fonctionnalités

Inscription

La fonction d'inscription permet aux visiteurs de devenir des membres actifs de la plateforme. Les utilisateurs potentiels fournissent leurs informations personnelles : leur nom, leur adresse e-mail et un mot de passe pour creer un compte.

J'ai structuree cette fonctionnalite de la manière suivante :

```
<form id="form-inscription" class="form-inscription d-flex jc-center column mw-600 m-auto" method="post"
      action="/ctrl/inscription.php">

      <div>
          <label for="prenom">Prenom</label>
          <input id="prenom" type="text" name="prenom" minlength="3" maxlength="20" autofocus required>

          <label for="nom">Nom</label>
          <input id="nom" type="text" name="nom" minlength="3" maxlength="20" required>

          <label for="mail">E-mail</label>
          <input id="mail" type="text" name="mail" minlength="5" maxlength="319" required>

          <label for="password">Password</label>
          <input id="password" type="password" name="password" minlength="4" maxlength="20" required>

          <button type="submit">Submit</button>
      </div>

  </form>
```

Ce formulaire envoie des donnees en utilisant la méthode POST au controleur /ctrl/inscription.php

```php
<?php

require('../lib/user.php');

session_start();

// Teste si l'Utilisateur est enregistré
$isRegistred = false;

//lit les infos saisie dans le formulaire avec $_POST
$nom = $_POST['nom'];
$prenom = $_POST['prenom'];
$mail = trim(htmlspecialchars($_POST['mail']));
$passwordClear = trim(htmlspecialchars($_POST['password']));
$idRole = 2;

//creer un utilisateur que avec ces attribue
$retour = LibUser::create($nom, $prenom, $mail, $motDePasse, $idRole);
if ($retour) {
        // appel fucntion login
        $utilisateur = LibUser::login($mail, $motDePasse);
        if ($utilisateur !== null) {
                $_SESSION['nom'] = $utilisateur['nom'];
                $_SESSION['motDePasse'] = $utilisateur['motDePasse'];
                $_SESSION['utilisateur'] = $utilisateur;
                header('Location: /');
                exit;
        }

        header('Location: /ctrl/inscription-display.php');
}
```

Ce code inclut le fichier /lib/user.php que contient les fonctions et les classes nécessaires au fonctionnement du script.

On recupere les données soumises dans le formulaire à l'aide de la superglobale `$_POST`. On lui assigne egalement une valeur de 2 (utilisateur).

Faisant ensuite appel a la methde statique LibUser::create($nom, $prenom, $mail, $motDePasse, $idRole) pour créer un nouvel utilisateur dans la base de données, en passant les données du formulaire à cette méthode.

Si la creation de l'utilisateur (`$retour`) est reussie (c est a dire si `create` renvoie `t r u e`), le code continue à l'interieur du bloc `if`. Dans ce cas l'utilisateur sera connecte'.

```php
class LibUser
{
    // Bibliothèque de fonctions dédiées aux Utilisateurs
    static function create($nom, $prenom, $mail, $motDePasse, $idRole)
    {
        // hashe le mot de passe
        $passwordHashed = password_hash($motDePasse, PASSWORD_BCRYPT);



        $query = 'INSERT INTO utilisateur (nom, prenom, mail, motDePasse, idRole) VALUES';
        $query .= ' (:nom, :prenom, :mail, :motDePasse, :idRole)';

        $stmt = LibDb::getPDO()->prepare($query);
        // $stmt->bindParam(':id', $idUser);
        $stmt->bindParam(':nom', $nom);
        $stmt->bindParam(':prenom', $prenom);
        $stmt->bindParam(':mail', $mail);
        $stmt->bindParam(':motDePasse', $passwordHashed );
        $stmt->bindParam(':idRole', $idRole);
        logMsg($stmt->debugDumpParams());

        // Exécute la requête
        $successOrFailure = $stmt->execute();
        logMsg("Success (1) or Failure (0) ? $successOrFailure" . PHP_EOL);

        return $successOrFailure;
    }
```

La classe LibUser contient des fonctions liées aux utilisateurs. Cette methode est statique, ce qui signifie qu'elle peut être appelée sans avoir besoin d'instancier un objet de la classe.

Elle prend cinq paramètres en entrée : $nom, $prenom, $mail, $motDePasse, $idRole.

La méthode prepare une requte SQL pour inserer un nouvel utilisateur dans la table "utilisateur" de la base de données. La requête SQL est construite avec des paramètres nommés :nom, :prenom, :mail, :motDePasse, :idRole) pour eviter les injections SQL.

Les paramètres de la requête SQL sont liés aux valeurs passées en entrée à l'aide de la méthode bindParam . Cela permet de sécuriser la requête et d'insérer les valeurs correctement dans la base de donnee.

La requête préparée est exécutée avec $stmt->execute(), et le resultat (true si la requête a réussi, false sinon) est stocké dans la variable $successOrFailure.

```php
<?php

$pageTitle = 'Inscription';

?>

<?php include '../view/inscription-display.php' ?>
```

Ce code sert a afficher la view de la page d inscription .

Login

La fonction de connexion (login) est generalement utilisee pour verifier les informations d'identification fournies par l'utilisateur et pour autoriser l'accès à certaines parties du site web.

```php
<form id="form-login" class="form-login d-flex jc-center column mw-600 m-auto"
      method="post" action="/ctrl/login.php">

    <label for="mail">E-mail</label>
    <input id="mail" type="text" name="mail" autofocus required>

    <label for="password">Password</label>
    <input id="password" type="password" name="password" required>

    <button type="submit">Submit</button>
</form>
```

Comme dans la fonction de inscription, Ce formulaire envoie des données en utilisant la méthode POST au contrôleur /ctrl/login.php

```php
// Teste si l'Utilisateur est enregistré
$isRegistred = false;

// Lit les infos saisie dans le formulaire avec $_POST
$mail = trim(htmlspecialchars($_POST['mail']));
$motDePasse = trim(htmlspecialchars($_POST['password']));

// Cherche un utilisateur que possede cette mail et ce password
$utilisateur = LibUser::login($mail, $motDePasse);
if ($utilisateur !== null) {
    $isRegistred = true;
}

// Quand l'Utilisateur est enregistré,
// enregistre son 'username' en session et le redireige vers la page d'accueil
if ($isRegistred) {

    $_SESSION['nom'] = $utilisateur['nom'];
    $_SESSION['motDePasse'] = $utilisateur['motDePasse'];
    $_SESSION['utilisateur'] = $utilisateur;

    header('Location: /');
    exit;
}
```

`$isRegistred` est initialisee a false. Elle sera utilisee pour déterminer si l'utilisateur est enregistre.

Les données du formulaire (`$mail` et `$motDePasse`) sont extraites à partir de la superglobale `$_POST`.

`$utilisateur = LibUser::login($mail, $motDePasse);` : Cette ligne appelle la méthode statique login de la classe `LibUser` en passant l'adresse e-mail (`$mail`) et le mot de passe (`$motDePasse`) en tant que paramètres.
Si la méthode login retourne un utilisateur (`$utilisateur !== null`), cela signifie que les informations d'identification sont correctes et que l'utilisateur est considéré comme enregistré.

Si l'utilisateur est enregistré , les informations de l'utilisateur sont stockées dans la session. Cela permet à l'utilisateur d'être identifié lors de ses futures visites sur le site.

```php
static function login($mail, $motDePasse)
    {
        // Prépare la requête
        $query = 'SELECT U.id, U.nom, U.prenom, U.mail, U.motDePasse, U.idRole';
        $query .= ' FROM utilisateur AS U';
        $query .= ' WHERE U.mail = :mail';
        // $query .= ' AND U.motDePasse = :motDePasse';
        $stmt = LibDb::getPDO()->prepare($query);
        $stmt->bindParam(':mail', $mail);
        // $stmt->bindParam(':motDePasse', $motDePasse);
        logMsg($stmt->debugDumpParams());

        // Exécute la requête
        $successOrFailure = $stmt->execute();
        // logMsg("Success (1) or Failure (0) ? $successOrFailure" . PHP_EOL);

        $result = $stmt->fetch(PDO::FETCH_ASSOC);
        if ($result == false) {
            return null;
        }
        if (password_verify($motDePasse, $result['motDePasse'])) {
            return $result;
        }
        return null;

    }
```

La fonction commence par construire une requête SQL pour sélectionner des informations spécifiques de la table "utilisateur". Elle sélectionne les colonnes "id", "nom", "prenom", "mail", "motDePasse" et "idRole" de la table "utilisateur" où l'adresse e-mail (U.mail) correspond à celle fournie dans le paramètre :mail de la requête.

La requête SQL est préparée en utilisant PDO (PHP Data Objects), ce qui permet de créer une requête sécurisée et d'éviter les injections SQL. Les paramètres de la requête, tels que :mail, sont liés à des valeurs en utilisant bindParam.

La fonction utilise `logMsg($stmt->debugDumpParams())` pour enregistrer les paramètres de la requête préparée à des fins de débogage.

La requête préparée est exécutée avec `$stmt->execute()`, et le résultat de l'exécution est stocké dans la variable `$successOrFailure`.

On utilise `$stmt->fetch(PDO::FETCH_ASSOC)` pour récupérer la première ligne de résultats de la requête sous forme de tableau associatif.

La fonction `fetch` est une méthode utilisée dans l'extension PDO pour récupérer une seule ligne de résultats d'une requête SQL sous forme d'un tableau associatif ou d'un objet.

Si la requête renvoie un résultat, la fonction utilise `password_verify` pour comparer le mot de passe fourni (`$motDePasse`) avec le mot de passe haché stocké dans la base de données

(`$result['motDePasse']`). Si les mots de passe correspondent, la fonction retourne les informations de l'utilisateur sous forme de tableau associatif. Sinon, elle retourne à nouveau `null`.

Affichage des articles

Pour structuré la fonction que permet aux visiteurs de visualiser les articles publie' sur la plateforme j'ai procédé de la manière suivante

```php
<?php session_start(); ?>
<?php $pageTitle = 'Recettes' ?>

<?php require_once('../lib/article.php'); ?>

<?php
if (array_key_exists('utilisateur', $_SESSION) && isset($_SESSION['utilisateur'])) {
    $idUtilisateur = $_SESSION['utilisateur']['id'];
}
?>

<?php
// Inclure le fichier contenant la fonction readAll() et la connexion à la base de données

// Appel de la fonction readAll() pour récupérer tous les articles
$articles = LibArticle::readArtPub($idUtilisateur);

?>
<?php include '../view/article.php' ?>
```

`<?php $articles = LibArticle::readArtPub($idUtilisateur); ?>` : Cette ligne appelle la fonction `readArtPub()` de la classe `LibArticle` (qui est incluse depuis le fichier `'article.php'`) pour récupérer tous les articles. L'ID de l'utilisateur est utilisé comme argument, la fonction peut récupérer des articles spécifiques à un utilisateur.

```php
static function readArtPub($idUtilisateur)
  {
      // Prépare la requête
      $query = 'SELECT ART.id, ART.idUtilisateur, ART.titre, ART.descriptionCourte,';
      $query .= ' ART.textArticle, ART.immage, ART.difficulte, ART.dateHeure, ART.pubblication, ART.idCategorie';
      $query .= ' FROM article ART';
      $query .= ' WHERE ART.pubblication = 1 or ART.idUtilisateur = :idUtilisateur';
      $query .= ' ORDER BY ART.titre ASC';
      $stmt = LibDb::getPDO()->prepare($query);
      $stmt->bindParam(':idUtilisateur', $idUtilisateur);
      // Exécute la requête
      $successOrFailure = $stmt->execute();
      //  logMsg("Success (1) or Failure (0) ? $successOrFailure" . PHP_EOL);

      $result = $stmt->fetchAll(PDO::FETCH_ASSOC);
      return $result;
  }
```

La fonction prend un paramètre $idUtilisateur qui représente l'ID de l'utilisateur pour lequel les articles sont récupérés.

La requête SQL est préparée dans la variable $query. Cette requête sélectionne plusieurs colonnes de la table "article" en fonction de certaines conditions. Voici un aperçu des éléments de la requête :

SELECT ART.id, ART.idUtilisateur, ART.titre, ...: Sélectionne les colonnes spécifiées de la table "article".

FROM article ART: Spécifie la table "article" avec l'alias "□ □ □".

WHERE ART.pubblication = or ART.idUtilisateur = :idUtilisateur: Définit les conditions pour filtrer les articles. Il sélectionne les articles qui ont pubblication égal à 1 (peut être publié) ou qui appartiennent à l'utilisateur spécifié par son ID (□□□ □□□□□□□□□).

ORDER BY ART.titre ASC: Ordonne les résultats par ordre croissant du champ "titre".

Enfin, la fonction renvoie le résultat, qui est une liste d'articles sous forme d'un tableau associatif.

```
1
2  <?php foreach ($articles as $article) { ?>
3      <article class="art p-1 mw-350 f-1-300">
4          <header class="art__header">
5              <picture>
6                  <img src="https://source.unsplash.com/1600x900?meal&<?= $article['titre'] ?>" alt="Lorem"
7                      width="600">
8              </picture>
9              <h2>
10                 <?= $article['titre'] ?>
11             </h2>
12             <p>
13                 <?= substr($article['descriptionCourte'], 0, 100) . '...' ?>
14             </p>
15             <p>
16                 <small>Difficulté :
17                     <?php for ($i = 1; $i <= $article['difficulte']; $i++) { ?>
18                         ⭐
19                     <?php } ?>
```

J'ai génère une liste d'articles à partir d'un tableau `$articles` en utilisant une boucle `foreach` qui itère à travers un tableau d'articles stocké dans la variable `$articles`. À chaque itération, un article individuel est extrait du tableau et affiché.

```php
<?php
                        // Divise la chaîne de texte $article['textArticle'] en un tableau de lignes
                        //  en utilisant des caractères de retour à la ligne comme délimiteurs.
                        $linesText = explode(PHP_EOL, $article['textArticle']);
                        $linenb = 0;
                        foreach ($linesText as $line) {
                            $linenb++;
                            if ($linenb > 6 ) {
                                break;
                            }
                            ?>
                            <?= $line ?><br>
                        <?php } ?>
```

Création d'un article

La création d'un article dans le blog implique plusieurs étapes, notamment la collecte des données du formulaire, la validation des données, l'insertion des données dans la base de données et l'affichage du résultat

```php
<form id="form-article" method="post" action="/ctrl/create-article.php">

            <div>
                <br>
                <label for="title">Titre de votre recette: </label>
                <input id="title" type="text" placeholder="Titre" name="titre" autofocus>
                <br>
                <label for="description">Description: </label>
                <input id="description" type="text" placeholder="Description" name="descriptionCourte">
                <br>
                <label for="text">Texte de votre recette: </label>
                <textarea id="text" placeholder="Texte" name="textArticle" rows="5" cols="33"></textarea>


                <br>
                <label for="difficulte">Difficulte: </label>
                <input id="difficulte" type="number" placeholder="Texte" name="difficulte" min="1" max="5">
                <br>
                <label for="categorie">Categorie recette: </label>
                <select id="categorie" name="idCategorie">
                    <?php foreach ($listCategories as $categorie): ?>
                        <option value="<?= $categorie['id'] ?>"><?= $categorie['nomCategorie'] ?></option>
                    <?php endforeach; ?>
                </select>

                <br>
                <label for="image">Image</label>
                <input id="image" type="text" placeholder="Votre image" name="immage">
                <hr>
                <button type="submit">Creer</button>

            </div>

        </form>
```

Une fois que l'utilisateur a rempli ces champs et cliquee sur le bouton "Creer", le donnees du formulaire sont envoyee au serveur via la methode POST

```php
<?php require_once('../lib/article.php');

session_start(); ?>


<?php $pageTitle = 'Create Article' ?>



<?php
$idUser = $_SESSION['utilisateur']['id'];
$title = $_POST['titre'];
$description = $_POST['descriptionCourte'];
$text = $_POST['textArticle'];
$picture = $_POST['immage'];
$idCategorie = $_POST['idCategorie'];
$difficulte = $_POST['difficulte'];


// Appel de la fonction create() pour creer des articles
$articles = LibArticle::create($idUser, $title, $description, $text, $picture, $idCategorie, $difficulte);


header('Location: /ctrl/article.php');
?>
```

Ce code recuper les donnees du formulaire

idUser : L'ID de l'utilisateur connectee , qui est obtenu a partir de la session.

title : Le titre de l'article.

description : La description courte de l'article.

text : Le texte complet de l'article.

picture : L'URL de l'image associée à l'article.

idCategorie : L'ID de la catégorie à laquelle l'article appartient.

difficulte : Le niveau de difficulte de l'article.

LibArticle::create($idUser, $title, $description, $text, $picture, $idCategorie, $difficulte); : Cette ligne fait appelle a la fonction create() de la classe LibArticle pour creer un nouveau article en utilisant les donnees extrait du formulaire. Les parametres de la fonction sont passes avec les valeurs récupérées précédemment.

```php
static function create($idUser, $title, $description, $text, $picture, $idCategorie, $difficulte)
    {
        $query = 'INSERT INTO article (idUtilisateur, titre, descriptionCourte, textArticle, immage, idCategorie, difficulte) VALUES';
        $query .= ' (:idUtilisateur, :titre, :descriptionCourte, :textArticle, :immage, :idCategorie, :difficulte)';

        $stmt = LibDb::getPDO()->prepare($query);
        $stmt->bindParam(':idUtilisateur', $idUser);
        $stmt->bindParam(':difficulte', $difficulte);
        $stmt->bindParam(':idCategorie', $idCategorie);
        $stmt->bindParam(':titre', $title);
        $stmt->bindParam(':descriptionCourte', $description);
        $stmt->bindParam(':textArticle', $text);
        $stmt->bindParam(':immage', $picture);
        logMsg($stmt->debugDumpParams());

        // Exécute la requête
        $successOrFailure = $stmt->execute();
        logMsg("Success (1) or Failure (0) ? $successOrFailure" . PHP_EOL);

        return $successOrFailure;
    }

```

La fonction create construit la requête SQL d'insertion en utilisant la syntaxe □ □ □ □ □ □ □ □ □ □. La requête insère des valeurs dans les colonnes de la table article qui correspondent aux données de l'article qu'on souhaite créer.

Ensuite, elle prépare la requête SQL en utilisant la méthode prepare de l'objet PDO. La requête contient des paramètres nommés qui seront liés aux valeurs réelles avant l'exécution de la requête.

Elle utilise la méthode bindParam pour lier chaque paramètre nommé de la requête à une valeur fournie en tant qu'argument de la fonction. Par exemple, :idUtilisateur est lié à la valeur de $idUser, :titre est lié à $title, etc..

Exécute la requête préparée en utilisant la méthode execute de l'objet PDO. Cette étape effectue réellement l'insertion des données dans la base de données.

```php
<?php require_once('../lib/article.php'); ?>

<?php

$pageTitle = 'Creer Article';


//list les categories pour alimentatire la liste des choix dans le formulaire
$listCategories = LibArticle::listCategorie();

?>

<?php include '../view/create-article-display.php' ?>
```

Afficchage des articles cree.

Commenter un article

Pour commenter un article, j'ai ajoute' une fonctionnalité qui permet aux utilisateurs de saisir leur commentaire dans le formulaire sur la page de l'article-single.

```
<form id="form-commentaire" method="post" action="/ctrl/commentaire.php">
                    <br>
                    <label for="commentaire">Commentaire: </label>
                    <input id="commentaire" type="text" placeholder="text" name="textCommentaire" autofocus required>
                    <br>
                    <input type="hidden" name="idArticle" value="<?= $article['id'] ?>">
                    <button type="submit">creer</button>
                </form>
```

Les données du formulaire sont récupérées à l'aide de la superglobale `$_POST` :

`$text` : Le contenu du commentaire saisi par l'utilisateur.

`$idArticle` : L'ID de l'article auquel le commentaire est associé.

`$idUser` : L'ID de l'utilisateur qui a soumis le commentaire, extrait de la session.

Appelle de la méthode statique `LibArticle::create_comment($text, $idArticle, $idUser)` pour ajouter un commentaire à un article.

Après avoir ajouté le commentaire, le code redirige l'utilisateur vers la page de l'article en cours en utilisant `header("Location: /ctrl/article-single.php?id=$idArticle")`.

```php

//lit les infos saisie dans le formulaire avec $_POST
$text = $_POST['textCommentaire'];
$idArticle = $_POST['idArticle'];
$idUser = $_SESSION['utilisateur']['id'];



// Appel de la fonction readAll() pour récupérer tous les articles
$articles = LibArticle::create_comment($text, $idArticle, $idUser);

//redirige utlisateur vers l article courant
header("Location: /ctrl/article-single.php?id=$idArticle");
```

```php
static function create_comment($text, $idArticle, $idUser)
    {
        $query = 'INSERT INTO commentaire (textCommentaire, idArticle, idUtilisateur) VALUES';
        $query .= ' (:textCommentaire, :idArticle, :idUtilisateur)';

        $pdo = LibDb::getPDO(); // Assuming you have a function to get the PDO connection

        $stmt = $pdo->prepare($query);
        $stmt->bindParam(':textCommentaire', $text);
        $stmt->bindParam(':idArticle', $idArticle);
        $stmt->bindParam(':idUtilisateur', $idUser);

        // Execute the query and handle any errors
        $successOrFailure = $stmt->execute();

        return $successOrFailure;
    }
```

Une requête SQL d'insertion INSERT INTO est préparée pour ajouter les données du commentaire dans la table commentaire. La requête inclut les valeurs à insérer en utilisant des paramètres nommés (:textCommentaire, :idArticle, :idUtilisateur).

Suppression d'un article

Pour supprimer un article de la base de données, faut exécuter une requête SQL de suppression (DELETE) en utilisant l'ID de l'article qu'on souhaite supprimer en utilisant PHP et PDO .

```php
<?php if ($idRole == 1 || $utilisateur['id'] == $article['idUtilisateur']) { ?>

                <div>
                    <a href="../ctrl/deletArticle.php?id=<?= $article['id'] ?>">delete article</a>
                </div>
            <?php } ?>
```

Si le rôle de l'utilisateur ($idRole) est égal à 1 (l'utilisateur a un rôle de administrateur). Si l'ID de l'utilisateur actuel ($utilisateur['id']) est égal à l'ID de l'utilisateur qui a créé l'article ($article['idUtilisateur']). Si on est le gestionnaire ou l auteur de l article redirige', on as le droit de supprimer ou modifier l article.

```php
<?php

require_once('../lib/article.php');

$pageTitle = 'Articles';

$id = $_GET['id'];

// Appel de la fonction delete pour effacer l article
$ret = LibArticle::deleteCommentaires($id);
$ret = LibArticle::delete($id);


// redirige l utilisateur a la page recette
header('Location: /ctrl/article.php');
include '../view/article.php' ?>
```

On récupère l'ID de l'article à supprimer à partir des paramètres GET de l'URL en utilisant `$_GET['id']`.
Ensuite, on appelle deux fonctions statiques de la classe LibArticle pour effectuer la suppression :
`LibArticle::deleteCommentaires($id)`: Cette fonction supprime les commentaires associés à l'article spécifié par son ID.

`LibArticle::delete($id)`: Cette fonction supprime l'article en utilisant l'ID.

Après avoir effectué la suppression, il redirige l'utilisateur vers la page des articles en utilisant header(`'Location: /ctrl/article.php'`).

Panneau d'administration

Le panneau d'administration que j'ai cree est l'interface réservée aux administrateurs du sitre pour la gestion des utilisateurs, des articles, des commentaires, etc.

J'ai concu un tableau qui liste les articles du site avec les informations de l'utilisateur qui les a créés, leur titre et leur statut de publication.

```php
//list les categories pour alimentatire la liste des choix dans le formulaire
$rows = LibArticle::listArtComm();
```

```php
static function listArtComm()
    {
        // Prépare la requête
        $query = 'SELECT ';
        $query .= ' ART.id AS idArticle';
        $query .= ', ART.titre';
        $query .= ', U.nom';
        $query .= ', U.id AS idUtilisateur';
        $query .= ', ART.descriptionCourte';
        $query .= ', ART.dateHeure';
        $query .= ', ART.pubblication';
        $query .= ' FROM article AS ART';
        $query .= ' INNER JOIN utilisateur AS U ON ART.idUtilisateur = U.id';
        $query .= ' WHERE ART.dateHeure > date_sub(curdate(), interval 30 day)';
        $stmt = LibDb::getPDO()->prepare($query);


        // Exécute la requête
        $successOrFailure = $stmt->execute();
        //  logMsg("Success (1) or Failure (0) ? $successOrFailure" . PHP_EOL);

        $result = $stmt->fetchAll(PDO::FETCH_ASSOC);
        return $result;
    }
```

Cette fonction effectue une requête SQL pour récupérer des informations sur les articles, y compris le nom de l'utilisateur qui les a crees, la date de creation, le titre, la description courte, et le statut de publication.

La requete utilise une jointure interne (INNER JOIN) entre la table "article" et la table "utilisateur" sur la colonne "idUtilisateur". Cela permet de lier chaque article à l'utilisateur qui l'a créé.

Une clause WHERE est utilisée pour filtrer les articles en fonction de la date. Cela est fait en comparant la colonne "dateHeure" avec la date actuelle moins 30 jours (date_sub(curdate(), interval 30 day)).

Les résultats de la requête sont récupérés à l'aide de $stmt->fetchAll(PDO::FETCH_ASSOC) et renvoyés en tant qu'ensemble de données associatif.

```php
<table>
        <tr>
            <th>user</th>
            <th>titre</th>
            <th>publication</th>
        </tr>

        <?php foreach ($rows as $row) { ?>
            <tr>
                <td>
                    <?= $row['nom'] ?>
                </td>

                <td>
                    <a href="../ctrl/article-single.php?id=<?= $row['idArticle'] ?>">
                        <?= $row['titre'] ?>
                    </a>
                </td>

                <td>
                    <?= $row['pubblication'] ?>
                </td>
            </tr>
        <?php } ?>
    </table>
```

Affichage du tableau administrateur.

Exemple d’utilisation javascript : carrousel

Dans mon application, j'ai ajouté une fonctionnalité qui affiche mes recettes dans un carrousel de manière aléatoire.

```
<script>
  console.log("script slider loaded");
  let slideIndex = 1;
  showSlides(slideIndex);

  // Next/previous controls
  function plusSlides(n) {
    showSlides(slideIndex += n);
  }

  // Thumbnail image controls
  function currentSlide(n) {
    showSlides(slideIndex = n);
  }

  function showSlides(n) {
    let i;
    let slides = document.getElementsByClassName("mySlides");
    let dots = document.getElementsByClassName("dot");
    if (n > slides.length) { slideIndex = 1 }
    if (n < 1) { slideIndex = slides.length }
    for (i = 0; i < slides.length; i++) {
      slides[i].style.display = "none";
    }
    for (i = 0; i < dots.length; i++) {
      dots[i].className = dots[i].className.replace(" active", "");
    }
    slides[slideIndex - 1].style.display = "block";
    dots[slideIndex - 1].className += " active";
  }

</script>
```

`console.log("script slider loaded");` : Cette ligne affiche un message dans la console pour indiquer que le script du carrousel a été chargé avec succès.

`let slideIndex = 1 ;` : Cette variable slideIndex est utilisée pour suivre la diapositive actuellement affichée. Elle est initialisée à 1, ce qui signifie que la première diapositive est affichée au chargement de la page.

`showSlides(slideIndex);` : Cette ligne appelle la fonction showSlides pour afficher la diapositive initiale (la première diapositive).

`function plusSlides(n);` : Cette fonction est appelée lorsque l'utilisateur clique sur les boutons "Suivant" ou "Précédent" pour naviguer entre les diapositives. Elle prend un

argument n, qui indique le nombre de diapositives à avancer (positif pour aller en avant, négatif pour revenir en arrière).

`function currentSlide(n)` : Cette fonction est appelée lorsque l'utilisateur clique sur l'un des points de pagination (dots) pour accéder directement à une diapositive spécifique. Elle prend un argument n qui correspond au numéro de la diapositive à afficher.

`function showSlides(n)` : Cette fonction principale gère l'affichage des diapositives en fonction de la valeur de slideIndex. Elle cache toutes les diapositives, puis affiche la diapositive correspondante à l'index actuel. De plus, elle met à jour les classes des points de pagination pour indiquer quelle diapositive est active.

Sécurité

Htmlspecialchars

La fonction `htmlspecialchars` en PHP a pour rôle de transformer des caractères spéciaux en entités HTML, principalement dans le but de renforcer la sécurité en prévenant les risques liés à l'injection de code HTML ou JavaScript non sécurisé dans les pages web.

```php
// Lit les infos saisie dans le formulaire avec $_POST
$mail = trim(htmlspecialchars($_POST['mail']));
$motDePasse = trim(htmlspecialchars($_POST['password']));
```

PDOStatement::bindParam

L'utilisation de PDO (PHP Data Objects) et de la méthode bindParam pour lier des paramètres à votre requête SQL est une pratique pour prévenir les attaques par injection SQL. Cela permet d'éviter que des données malveillantes ne soient directement incorporées dans les requêtes SQL. Au lieu de cela, les valeurs des paramètres sont liées de manière sécurisée à la requête, ce qui les rend plus sûres.

```php
$query = 'INSERT INTO utilisateur (nom, prenom, mail, motDePasse, idRole) VALUES';
        $query .= ' (:nom, :prenom, :mail, :motDePasse, :idRole)';

        $stmt = LibDb::getPDO()->prepare($query);
        $stmt->bindParam(':nom', $nom);

        ...
```

Au lieu d'incorporer directement les valeurs dans la requête SQL, ce code utilise des paramètres nommés (par exemple, :nom, :prenom, :mail, etc.) dans la requête. Cela permet de préparer la requête sans les données réelles, ce qui évite les attaques par injection SQL.

La méthode prepare de PDO est utilisée pour préparer la requête. La requête préparée est stockée dans la variable $stmt.

Les paramètres nommés de la requête SQL sont liés à des variables PHP réelles à l'aide de la méthode bindParam. Cela signifie que lorsque la requête est exécutée, les valeurs de ces paramètres seront automatiquement remplacées par les valeurs des variables PHP correspondantes. Cela garantit que les données sont insérées de manière sécurisée.

Par exemple, $stmt->bindParam(':nom', $nom) signifie que la valeur de $nom sera insérée dans la colonne "nom" de la table "utilisateur" lorsque la requête sera exécutée.

Mot de passe hashé

```php
$passwordHashed = password_hash($motDePasse, PASSWORD_BCRYPT);
```

Avant d'insérer le mot de passe dans la base de données, il est haché à l'aide de la fonction password_hash. Cela sécurise le stockage du mot de passe en le transformant en une chaîne de caractères hachée irréversible. Lors de la vérification ultérieure du mot de passe, de norme, on utilise password_verify pour comparer le mot de passe haché stocké avec le mot de passe saisi par l'utilisateur.

Référencement

Titre de Page Unique (<title>) :

Est utilisée pour déterminer le titre de la page HTML, qui est affiché dans l'onglet du navigateur. Lorsque cette variable est définie, est utilise' pour personnaliser le titre de chaque page du site web.

Description de Page Unique

Chaque page devrait également avoir une balise <meta> avec l'attribut "name" défini sur "description". Cette balise <meta> devrait contenir une brève description du contenu de la page. Cela aidera les moteurs de recherche à afficher des extraits pertinents dans les résultats de recherche.

```
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="/asset/style.css">

    <meta name="description" content="Découvre et partage ta passion pour la cuisine!">
    <title>Foodies - Home</title>
</head>
```

Utilisation de l'Attribut "alt" des Balises <img> : Chaque image sur le site web a un attribut "alt" qui décrit brièvement le contenu de l'image. Cela améliore l'accessibilité pour les utilisateurs ayant des lecteurs d'écran et renforce le référencement.

Utilisation Maximale des Balises Sémantiques : Utilisez des balises HTML sémantiques comme <header>, <nav>, <article>, <section>, <footer>, etc., pour structurer le contenu de votre site de manière logique. Cela aide les moteurs de recherche à comprendre la hiérarchie de votre contenu.

```
<picture>
    <img src="asset/farandolle.jpg" alt="tous les specilaite">
</picture>
```

Balisage HTML Valide : Vérification du code à l'aide du validateur W3C pour assurer sa conformité aux normes HTML, garantissant ainsi un meilleur référencement

Optimisation du Chargement des Images : Réduisez la taille des images en utilisant des formats adaptés (comme le WebP) et des outils de compression d'images. Utilisez également des images responsives pour une expérience utilisateur optimale sur tous les appareils.

Minification des Ressources CSS et JS : Minifiez vos fichiers CSS et JavaScript pour réduire leur taille. Cela accélère le chargement de la page.

Minimiser les Échanges Réseaux Externes (CDN) : Réduire le nombre de demandes externes, notamment celles vers des réseaux de distribution de contenu (CDN), pour accélérer le chargement de la page.

Connexion Sécurisée HTTPS : Obtenez un certificat SSL pour activer la connexion sécurisée HTTPS sur votre site web. Cela garantit la sécurité des données de vos utilisateurs et est également favorable au référencement.

Conclusion

En conclusion, je tiens à souligner que la création et le développement d'un projet web ou d'une application ont été pour moi un parcours passionnant et stimulant. Tout au long de ce processus, j'ai réalisé à quel point il est crucial de planifier avec soin, de concevoir de manière réfléchie et de mettre en œuvre de manière méticuleuse.

Les décisions relatives aux fonctionnalités, à l'architecture, à la sécurité et à d'autres aspects clés ont été au cœur de mon expérience. J'ai appris que ces choix sont essentiels pour atteindre les objectifs du projet.

En outre, il est primordial de rester flexible et de s'adapter aux évolutions des besoins des utilisateurs et du marché. Les projets web et les applications sont en constante évolution, et je comprends maintenant l'importance de la maintenance et des mises à jour régulières pour assurer leur efficacité continue.

En fin de compte, le succès de ce projet a été le fruit de l'engagement de toute l'équipe, d'une vision claire et d'une exécution solide. Que ce soit pour un site web, une application mobile, une plateforme de commerce électronique ou tout autre projet, je suis convaincu que la planification minutieuse et l'attention aux détails sont essentielles pour créer une solution qui répond aux besoins des utilisateurs et atteint les objectifs fixés.

Je tiens à exprimer ma profonde gratitude envers mes formateurs, leur patience, leur dévouement et la richesse de leurs connaissances qu'ils nous ont transmises tout au long de cette formation ont été inestimables. Cela a été une expérience d'apprentissage précieuse, et je suis reconnaissant d'avoir eu l'opportunité de travailler avec des professionnels aussi dévoués et compétents. Merci infiniment pour votre soutien et votre guidance.