Front-end :

Conception :

J’ai choisi de mettre en place une interface simple et intuitive pour l’utilisateur.

En effet mon site est composé de 8 pages principales (accueil, recettes, recherche, ajouter une recette, mes recettes favorites, mon compte, connexion et inscription).

D’autres pages sont simplement des extensions des pages principales qui permettent d’exécuter des actions (par exemple la page ‘ recette\_detail ‘ va afficher en détail un recette précise de la page ‘recettes’. Autre exemple : la page ‘mot\_de\_passe\_oublié’ accessible via un lien à partir de la page ‘connexion’ va permettre d’envoyer un lien de réinitialisation du mot de passe.)

Les 8 pages sont accessibles via une barre de navigation horizontale alors que les autres pages sont accessibles via des liens à des emplacements spécifiques.

La barre de navigation a plusieurs particularités :

- une fois connecté l’utilisation ne voit plus les liens de connexion et d’inscription, mais à la place il voit un bouton de déconnexion.

- lorsque l’utilisateur connecté a le rôle d’admin, la barre de navigation affiche un lien supplémentaire vers la page ‘Admin’.

- et pour finir un bouton permet d’alterner entre mode clair et mode sombre.

Concernant les pages principales:

- les pages Accueil, Recettes et Recherche sont accessibles à tous.

- les pages Ajouter une recette, Mes recettes favorites et Mon compte sont accessibles à tous mais lorsqu’au utilisateur n’est connecté, elles indiquent qu’il faut se connecter pour accéder à ces fonctionnalités et proposent une redirection vers la page de connexion ou la page d’inscription.

L‘ensemble des pages permet d’utiliser les fonctionnalités utilisateurs attendues :

- s’inscrire, se connecter, modifier son profil, récupérer son mot de passe.

- ajouter des recettes, les modifier, les supprimer.

- afficher les recettes, les commenter, les mettre en favoris.

- rechercher des recettes par type de cuisine, ingrédients et temps de préparation.

- gérer les permissions et les rôles pour les différentes sections du site.

- modérer les commentaires, les recettes et les utilisateurs.

Responsivité :

J’ai crée un fichier responsive.css dans lequel j’ai inséré des règles media queries pour l’affichage sur tablettes et petits écrans et pour l’affichage sur smarphones et très petits écrans.

Ces règles media queries modifient notemment :

- l’apparence et la lisibilité du menu déroulant de la barre de navigation,

- la taille des caractères,

- l’espace entre les caractères,

- la taille des images.

Accessibilité :

Afin d’assurer une navigation accessible, j’ai respecté les normes WCAG.

- un alt textuel est fourni pour toutes les images,

- les contrastes de couleurs respectent un ratio suffisant,

- une navigation intégrale au clavier est possible,

- les pages sont rédigées en langage clair et simple,

- la structure de navigation est cohérente,

- les messages d’erreur sont précis et instructif,

- des rôles ARIA ont été implémentés

Tests :

Pour les tests, j’ai utilisé Google Lighthouse pour évaluer les perfomances, l’accessibilité, les bonnes pratiques et le SEO. Cet outil m’a permis d’améliorer de nombreux points (notamment la taille des image, ce qui m’a poussé à intégrer une méthode pour compressé les images uploadées).

J’ai également utilisé l’outil de développement intégré à mon navigateur (chrome) pour tester l’affichage sur différentes tailles d’écrans.

Back-end :

Schéma fonctionnel :

Une image contenant texte, capture d’écran, Police, conception

Le contenu généré par l’IA peut être incorrect.

Base de données :

Une image contenant texte, diagramme

Le contenu généré par l’IA peut être incorrect.

Modèles de données :

CREATE TABLE `utilisateurs` (

`id` INT NOT NULL AUTO\_INCREMENT,

`nom` VARCHAR(100) NOT NULL,

`email` VARCHAR(255) NOT NULL UNIQUE,

`mot\_de\_passe` VARCHAR(255) NOT NULL,

`role` ENUM('user','admin') NOT NULL DEFAULT 'user',

`date\_inscription` DATETIME NOT NULL DEFAULT CURRENT\_TIMESTAMP,

PRIMARY KEY (`id`)

)

CREATE TABLE `bannis` (

`email` VARCHAR(100) NOT NULL,

`date\_banissement` DATETIME NOT NULL DEFAULT CURRENT\_TIMESTAMP,

PRIMARY KEY (`email`)

)

CREATE TABLE `password\_resets` (

`id` INT NOT NULL AUTO\_INCREMENT,

`email` VARCHAR(255) NOT NULL,

`token` VARCHAR(64) NOT NULL,

`expiration` DATETIME NOT NULL,

PRIMARY KEY (`id`),

)

CREATE TABLE `recettes` (

`id` INT NOT NULL AUTO\_INCREMENT,

`titre` VARCHAR(150) NOT NULL,

`type\_de\_cuisine` ENUM(`cuisine\_traditionnelle`, `cuisine\_du\_monde`, `fast\_food`) NOT NULL,

`ingredients` TEXT NOT NULL,

`description` TEXT NOT NULL,

`instructions` TEXT NOT NULL,

`temps\_de\_preparation` INT NOT NULL,

`auteur\_id` INT NOT NULL,

`image\_url` VARCHAR(255) NOT NULL,

`date\_creation` DATETIME NOT NULL DEFAULT CURRENT\_TIMESTAMP,

PRIMARY KEY (`id`),

INDEX (`auteur\_id`),

CONSTRAINT `fk\_recette\_auteur`

FOREIGN KEY (`auteur\_id`)

REFERENCES `utilisateurs` (`id`)

ON DELETE CASCADE

ON UPDATE CASCADE

)

CREATE TABLE `commentaires` (

`id` INT NOT NULL AUTO\_INCREMENT,

`recette\_id` INT NOT NULL,

`user\_id` INT NOT NULL,

`contenu` TEXT NOT NULL,

`date\_publication` DATETIME NOT NULL DEFAULT CURRENT\_TIMESTAMP,

PRIMARY KEY (`id`),

INDEX (`recette\_id`),

INDEX (`user\_id`),

CONSTRAINT `fk\_commentaire\_recette`

FOREIGN KEY (`recette\_id`)

REFERENCES `recettes` (`id`)

ON DELETE CASCADE,

CONSTRAINT `fk\_commentaire\_utilisateur`

FOREIGN KEY (`user\_id`)

REFERENCES `utilisateurs` (`id`)

ON DELETE CASCADE

)

CREATE TABLE `favoris` (

`user\_id` INT NOT NULL,

`recette\_id` INT NOT NULL,

PRIMARY KEY (`user\_id`, `recette\_id`),

INDEX (`recette\_id`),

CONSTRAINT `fk\_favoris\_utilisateur`

FOREIGN KEY (`user\_id`)

REFERENCES `utilisateurs` (`id`)

ON DELETE CASCADE,

CONSTRAINT `fk\_favoris\_recette`

FOREIGN KEY (`recette\_id`)

REFERENCES `recettes` (`id`)

ON DELETE CASCADE

)

Framework :

Symfony :

J’ai utilisé plusieurs modules du framework Symfony.

- Mailer et Mime pour l’envoi du mail de réinitialisation du mot de passe :

-Pour la configuration du transport DSN SMTP défini dans un fichier.env

-Pour la création d’un objet Email, la définition de l’expéditeur, du destinataire, du sujet et du contenu html.

-Pour l’envoi du mail.

- PhpUnit pour effectuer les différents tests :

-Tests Unitaires :

1- AuthTest.php :

- Teste la fonction isLoggedIn() quand la session n’a pas été démarrée ou qu’aucun utilisateur n’est identifié. On s’attend à false.

- Teste la fonction isLoggedIn() quand un identifiant d’utilisateur est présent dans la session. On s’attend à true.

- Teste la fonction isAdmin() quand l’utilisateur est connecté mais n’a pas le rôle «admin». On s’attend à false.

- Teste la fonction isAdmin() quand l’utilisateur est connecté et a le rôle « admin ». On s’attend à true.

2- LogoutServiceTest.php :

- Teste la session qui doit être totalement vide après logout().

- Teste la redirection renvoyée après logout() qui doit correspondre à la page de connexion.

3- RecipeValidatorTest.php :

- Teste si une erreur est signalée quand le titre est vide lors de l’envoi du formulaire de dépôt de recette.

- Teste si une erreur est signalée quand l’extension de l’image n’est pas autorisée.

-Tests d’Intégration :

- AjouterRecetteTest.php :

- Vérifie que la page affiche un message d’invitation à se connecter et cache le formulaire si l’utilisateur n’est pas connecté.

- Vérifie que le formulaire affiche un message d’erreur générique lorsque les champs requis sont vides et que le formulaire reste affiché.

-Tests Fonctionnels :

- HomeFunctionalTest.php :

- Vérifie que, lorsqu’aucun utilisateur n’est en session les liens Connexion et Inscription apparaissent et le lien Déconnexion n’apparaît pas.

- Vérifie que, lorsqu’un utilisateur est connecté le bouton Déconnexion est affiché.