# 

# **Titre Professionnel Concepteur Développeur d’Application**

**Dossier Projet**

Présentée par Hasina RAKOTOMALALA

Formateurs et encadrant technique

Manel BENHAMOUDA

Frederic BUI

Michael NGUYEN

Centre de formation G2R

Table des matières

[**1. Remerciements** 4](#_Toc192755005)

[**2. Liste des compétences mises en œuvre** 4](#_Toc192755006)

[**3. Introduction** 5](#_Toc192755007)

[**La présentation du projet** 5](#_Toc192755008)

[**Historique et naissance de l’idée** 5](#_Toc192755009)

[**Population cible** 6](#_Toc192755010)

[**Objectifs du projet** 6](#_Toc192755011)

[***Objectif 1 : Offrir une source d’inspiration culinaire*** 6](#_Toc192755012)

[***Objectif 2 : Créer une communauté active*** 7](#_Toc192755013)

[**Analyse du marché et concurrence** 7](#_Toc192755014)

[***Présentation du marché*** 7](#_Toc192755015)

[***Analyse des concurrents*** 7](#_Toc192755016)

[**4. Cahier des charges et analyse des besoins** 8](#_Toc192755017)

[**Collecte des besoins utilisateurs** 8](#_Toc192755018)

[**Contraintes du projet et livrables attendus** 8](#_Toc192755019)

[**5. Gestion de projet** 9](#_Toc192755020)

[**Méthode Kanban pour la documentation des besoins** 9](#_Toc192755021)

[**Cycle agile d’ajustement des spécifications** 9](#_Toc192755022)

[**Suivi, planification et collaboration** 9](#_Toc192755023)

[**Approche structurée pour le déploiement** 10](#_Toc192755024)

[**Sauvegarde et tests automatisés** 10](#_Toc192755025)

[**Fonctionnalités principales couvertes par les spécifications** 10](#_Toc192755026)

[**Objectifs de qualité et conclusion** 10](#_Toc192755027)

[**6. Formalisation UML** 11](#_Toc192755028)

[**Modélisation des interactions utilisateur** 11](#_Toc192755029)

[**Modélisation des entités et du comportement** 13](#_Toc192755030)

[**7. Technologies et outils utilisés** 16](#_Toc192755031)

[**Présentation des technologies** 16](#_Toc192755032)

[**Installation et configuration des outils** 21](#_Toc192755033)

[**Justification des choix techniques** 21](#_Toc192755034)

[**8. Choix de l’architecture** 22](#_Toc192755035)

[**Choix de l'architecture MVC** 22](#_Toc192755036)

[**Interactions entre les composants** 22](#_Toc192755037)

[***Définition des relations entre les entités*** 24](#_Toc192755038)

[***Optimisation de l’indexation de la base de données*** 24](#_Toc192755039)

[**Mise en place de l’API REST** 26](#_Toc192755040)

[**9. Les maquettes et enchaînement des maquettes** 27](#_Toc192755041)

[**Conception de l’Expérience Utilisateur (UX)** 27](#_Toc192755042)

[**Conception Interface Utilisateur (UI)** 29](#_Toc192755043)

[**10. Fonctionnalités du site** 31](#_Toc192755044)

[**Gestion des recettes** 31](#_Toc192755045)

[**Sécurisation et gestion des droits des recettes** 32](#_Toc192755046)

[**Moteur de recherche et filtrage des recettes** 32](#_Toc192755047)

[**Interactions communautaires** 32](#_Toc192755048)

[**Interface utilisateur et expérience utilisateur (UX/UI)** 33](#_Toc192755049)

[**Gestion métier et services backend** 38](#_Toc192755050)

[***Exemple de code Backend*** 40](#_Toc192755051)

[**11. Sécurité et qualité du code** 41](#_Toc192755052)

[**Tests et validation** 41](#_Toc192755053)

[**Tests de sécurité et détection des vulnérabilités** 44](#_Toc192755054)

[***Sécurisation de l’application et protection des données*** 46](#_Toc192755055)

[**12. Déploiement automatisé avec Jenkinsfile** 47](#_Toc192755056)

[**Intégration continue et déploiement** 47](#_Toc192755057)

[**13. Présentation des éléments techniques** 50](#_Toc192755058)

[**14. Conclusion** 53](#_Toc192755059)

# **1. Remerciements**

Je remercie chaleureusement le centre de formation G2R pour m’avoir offert un cadre idéal où j’ai pu développer mes compétences et avancer dans mes projets. Un grand merci à l’équipe pédagogique pour son accompagnement, sa patience et ses précieux conseils. Je suis également reconnaissante envers mon maître de stage Ludovic Célérier et surtout Anh Truong, dont le soutien et les remarques constructives ont joué un rôle clé dans ma progression. Grâce à eux, j’ai pu évoluer tant sur le plan technique que personnel.

Un immense merci à mon mari, dont le soutien et les encouragements m’ont portée tout au long de cette formation. Sa confiance en moi et sa présence ont été essentiels pour me donner la force de persévérer. Ce parcours, encore en cours, aurait été bien différent sans lui.

Je tiens aussi à exprimer ma gratitude envers Manel Benhamouda, Frédéric Bui et Michael Nguyen, dont l’accompagnement a été déterminant. Leur pédagogie, leur disponibilité et leur engagement ont facilité mon apprentissage et renforcé ma motivation. Ils ont su rendre les concepts techniques plus accessibles et m’ont donné la confiance nécessaire pour avancer sereinement. Leur implication dans la réussite de leurs apprenants est inspirante, et je leur suis profondément reconnaissante pour l’impact qu’ils ont eu sur mon parcours.

Enfin, je remercie ma famille pour son soutien sans faille. Leur confiance et leurs encouragements m’ont aidée à surmonter les difficultés et à rester motivée. Ils ont toujours été là pour moi, prêts à écouter, conseiller et me donner l’énergie nécessaire pour continuer. Leurs sacrifices et leur patience ont été une source précieuse de motivation. Sans eux, je n’aurais pas pu franchir ces étapes importantes.

# **2. Liste des compétences mises en œuvre**

1. Installer et configurer son environnement de travail en fonction du projet
2. Développer des interfaces utilisateur
3. Développer des composants métier
4. Contribuer à la gestion d’un projet informatique
5. Analyser les besoins et maquetter une application
6. Définir l’architecture logicielle d’une application
7. Concevoir et mettre en place une base de données relationnelle
8. Développer des composants d’accès aux données SQL et NoSQL
9. Préparer et exécuter les plans de tests d’une application
10. Préparer et documenter le déploiement d’une application
11. Contribuer à la mise en production dans une démarche DevOps

# **3. Introduction**

**Accès au projet**

**Site en ligne :** <https://deliceketo.sytes.net>

**Code source GitHub :** <https://github.com/onjahasn/user_CDA.git>

**Code source Angular :**https://github.com/onjahasn/commentaire.git

## **La présentation du projet**

Je développe une plateforme dédiée au partage de recettes cétogènes, où les utilisateurs peuvent ajouter, modifier et supprimer des recettes tout en interagissant avec celles des autres. L’interface a été pensée pour être intuitive et simple d’utilisation.

Le site est conçu avec Symfony, qui gère à la fois le backend et le front. Les données sont stockées de manière sécurisée, avec des mesures de protection comme l’authentification et la gestion des autorisations pour garantir la confidentialité des informations. Une API RESTful assure des échanges fluides et efficaces entre le serveur et les utilisateurs, offrant ainsi une expérience optimisée.

L’objectif est de créer un espace moderne et convivial, où la communauté keto peut facilement partager et découvrir de nouvelles recettes.

## **Historique et naissance de l’idée**

Depuis un moment, je m’intéresse à l’alimentation cétogène et j’ai souvent cherché des recettes sur Internet, en passant par Facebook, Instagram, TikTok, YouTube et plusieurs blogs. Mais devoir naviguer entre toutes ces sources prend du temps et peut être compliqué.

C’est en voyant ce problème que j’ai eu l’idée de créer une application qui regrouperait toutes ces recettes au même endroit. Cette plateforme permettrait aux utilisateurs de partager facilement leurs propres recettes, de découvrir de nouvelles idées et d’échanger des conseils. En plus de rendre les recettes plus accessibles, elle encouragerait les échanges entre personnes intéressées par ce type d’alimentation.

L’objectif est de proposer un outil simple et pratique pour ceux qui veulent trouver des recettes sans passer des heures à chercher. Ce projet pourrait aider beaucoup de monde à mieux s’organiser et à partager leurs découvertes.

## **Population cible**

Le site de partage de recettes cétogènes s’adresse à un large public aux objectifs variés en matière de santé et de nutrition. Il concerne autant ceux qui veulent adopter une alimentation plus saine, réduire leur consommation de glucides ou perdre du poids, que les personnes souffrant de troubles métaboliques comme le diabète de type 2. Il peut aussi intéresser les sportifs souhaitant optimiser leurs performances grâce à une alimentation riche en bonnes graisses.

Que l’on soit débutant en cuisine ou passionné, chacun peut y trouver des idées pratiques et créatives pour cuisiner selon les principes du régime keto. La plateforme offre un espace interactif où les utilisateurs peuvent partager, modifier et commenter des recettes, échanger des conseils et découvrir de nouvelles inspirations pour une alimentation équilibrée.

## **Objectifs du projet**

### ***Objectif 1 : Offrir une source d’inspiration culinaire***

L’objectif principal du site est d’inspirer les utilisateurs en leur proposant une large sélection de recettes adaptées au régime cétogène. Ils peuvent y découvrir des plats variés, des recettes classiques aux créations plus originales, tout en apprenant à cuisiner selon les principes de cette alimentation. Chaque recette est mise en avant de manière claire et accessible, afin de faciliter la préparation des repas et encourager de bonnes habitudes alimentaires. L’idée est d’offrir un espace où chacun peut élargir ses horizons culinaires et trouver des idées simples et savoureuses pour une alimentation équilibrée.

### ***Objectif 2 : Créer une communauté active***

Le site vise aussi à créer une communauté dynamique où les utilisateurs peuvent partager leurs propres recettes et astuces. Grâce à cet espace collaboratif, chacun peut apporter sa contribution et aider les autres à adopter une alimentation plus saine. Même les personnes peu à l’aise en cuisine peuvent y trouver des idées faciles à réaliser. L’objectif est de rendre la cuisine keto accessible, ludique et variée, tout en permettant aux membres d’échanger, de s’entraider et de progresser ensemble vers un mode de vie plus sain.

## **Analyse du marché et concurrence**

### ***Présentation du marché***

Le marché de l’alimentation cétogène connaît une forte croissance, portée par l’intérêt grandissant pour les modes de vie sains et les régimes spécifiques. Riche en graisses et pauvre en glucides, ce type d’alimentation s’adresse à un public varié, des personnes cherchant à perdre du poids aux sportifs, en passant par ceux souffrant de troubles métaboliques comme le diabète. Il est également apprécié pour ses effets potentiels sur l’énergie, la satiété et la santé cognitive.

Ce marché englobe plusieurs domaines, allant des recettes et produits alimentaires spécialisés aux suppléments et services en ligne, comme les blogs et plateformes de partage de recettes. Avec l’augmentation de la demande, de plus en plus d’acteurs entrent sur ce marché pour proposer des solutions adaptées aux besoins des consommateurs. La nutrition étant un secteur en constante évolution, de nouveaux concurrents apparaissent régulièrement, chacun cherchant à apporter une approche innovante au régime cétogène.

### ***Analyse des concurrents***

Les concurrents sur le marché :

1. Diet Doctor : Un des leaders dans le domaine des régimes cétogènes, Diet Doctor propose des recettes, des plans de repas et des conseils nutritionnels. Il se distingue par ses contenus détaillés, des guides pour débutants, et des témoignages. Cependant, son modèle premium limite l'accès à certaines fonctionnalités. 2. KetoConnect : Un autre acteur majeur, KetoConnect propose des recettes variées et des ressources sur le régime cétogène. Leur approche est plus axée sur les vidéos et tutoriels, avec une forte présence sur les réseaux sociaux. Ils encouragent la personnalisation des recettes, mais le site est principalement orienté vers les anglophones.

3. MyFitnessPal : Bien que plus généraliste, MyFitnessPal permet aux utilisateurs de suivre leur alimentation, y compris des régimes comme le céto.

Cependant, il manque d’un aspect communautaire fort centré uniquement sur le régime cétogène, ce qui limite l'engagement de cette niche.

# **4. Cahier des charges et analyse des besoins**

## **Collecte des besoins utilisateurs**

Pour développer **Délices Keto**, j’ai cherché à comprendre ce que la communauté cétogène attendait réellement d’une plateforme de partage de recettes. J’ai analysé les échanges sur Facebook et Instagram, où les adeptes du régime partagent régulièrement des recettes, posent des questions sur les ingrédients et recherchent des astuces pratiques.

YouTube m’a aussi permis d’identifier des tendances et des difficultés rencontrées par les créateurs de contenu. En parcourant les commentaires sous les vidéos de recettes keto, j’ai pu repérer les interrogations fréquentes et les attentes du public, notamment sur la simplicité des préparations et la diversité des plats proposés.

Ces observations ont confirmé le besoin d’un espace structuré et interactif, où les utilisateurs pourraient facilement partager, consulter et commenter des recettes adaptées au régime cétogène. En m’appuyant sur ces retours, j’ai imaginé une plateforme qui facilite l’accès aux recettes tout en encourageant l’échange et la découverte culinaire au sein d’une communauté engagée.

## **Contraintes du projet et livrables attendus**

Le développement de Délices Keto doit prendre en compte plusieurs contraintes techniques et fonctionnelles. La plateforme doit être fluide, intuitive et sécurisée, notamment pour la gestion des utilisateurs et des recettes. L’authentification et la protection des données personnelles sont essentielles, tout comme l’optimisation des performances pour garantir une navigation agréable.

L’ajout progressif de nouvelles fonctionnalités, comme un blog pour partager des expériences et conseils, un système de commentaires et un forum de discussion, doit être bien planifié pour éviter une surcharge dès le départ. L’objectif est de créer un espace interactif où les membres peuvent échanger, s’entraider et s’inspirer. Une messagerie interne pourrait aussi être intégrée pour faciliter les échanges entre utilisateurs.

L’enjeu principal est de livrer une première version fonctionnelle avec les bases essentielles, puis d’enrichir progressivement la plateforme en améliorant l’expérience utilisateur et en ajoutant de nouvelles interactions. Tout le projet repose sur la capacité à structurer ces évolutions tout en assurant un site simple d’utilisation et adapté aux besoins des adeptes du régime cétogène.

# **5. Gestion de projet**

## **Méthode Kanban pour la documentation des besoins**

Les spécifications fonctionnelles de **Délices Keto** ont été définies à partir des attentes identifiées lors de l’analyse des besoins des futurs utilisateurs et de la communauté cétogène. Pour structurer ces spécifications et organiser efficacement le développement, j’ai utilisé un **tableau Kanban** dans **GitHub Projects**. Chaque besoin collecté a été transformé en **issue**, décrivant l’objectif visé, les critères d’acceptation, les utilisateurs cibles et les interactions attendues avec l’application. Cette organisation a facilité la traçabilité du développement et permis un suivi précis des avancées, tout en offrant la flexibilité nécessaire pour ajuster la planification au fil du projet.

## **Cycle agile d’ajustement des spécifications**

Les différentes fonctionnalités ont été classées par statut et accompagnées d’étiquettes précisant la priorité, la nature de la tâche et la complexité estimée. Ce système a permis de documenter chaque évolution de l’application et de discuter des ajustements nécessaires avec les personnes impliquées, que ce soit pour affiner une règle métier, modifier une interface ou enrichir la base de données. Grâce aux discussions intégrées aux **issues**, les spécifications ont pu être affinées progressivement, adoptant ainsi une approche **agile**, où les fonctionnalités étaient ajustées en fonction des priorités et des contraintes rencontrées.

## **Suivi, planification et collaboration**

Tout au long du développement, la gestion des tâches et des évolutions s’est appuyée sur **GitHub Projects** et le **Kanban**, facilitant la répartition du travail et la coordination des différentes phases du projet. En sollicitant régulièrement des retours de formateurs et de collègues, j’ai pu clarifier certains aspects techniques et adapter la roadmap en conséquence. Cette approche a permis d’assurer un suivi rigoureux et d’éviter les blocages en maintenant une vision claire des objectifs à atteindre.

## **Approche structurée pour le déploiement**

Une fois les fonctionnalités validées, leur mise en production a suivi une approche méthodique. J’ai commencé par sélectionner un hébergement adapté, capable de garantir de bonnes performances et de supporter la montée en charge. Ensuite, j’ai automatisé le **déploiement avec Jenkins**, standardisant ainsi le processus pour éviter les erreurs manuelles. La gestion des **migrations de base de données** avec **Doctrine** et l’adaptation des configurations de production ont également été intégrées pour garantir une transition fluide.

## **Sauvegarde et tests automatisés**

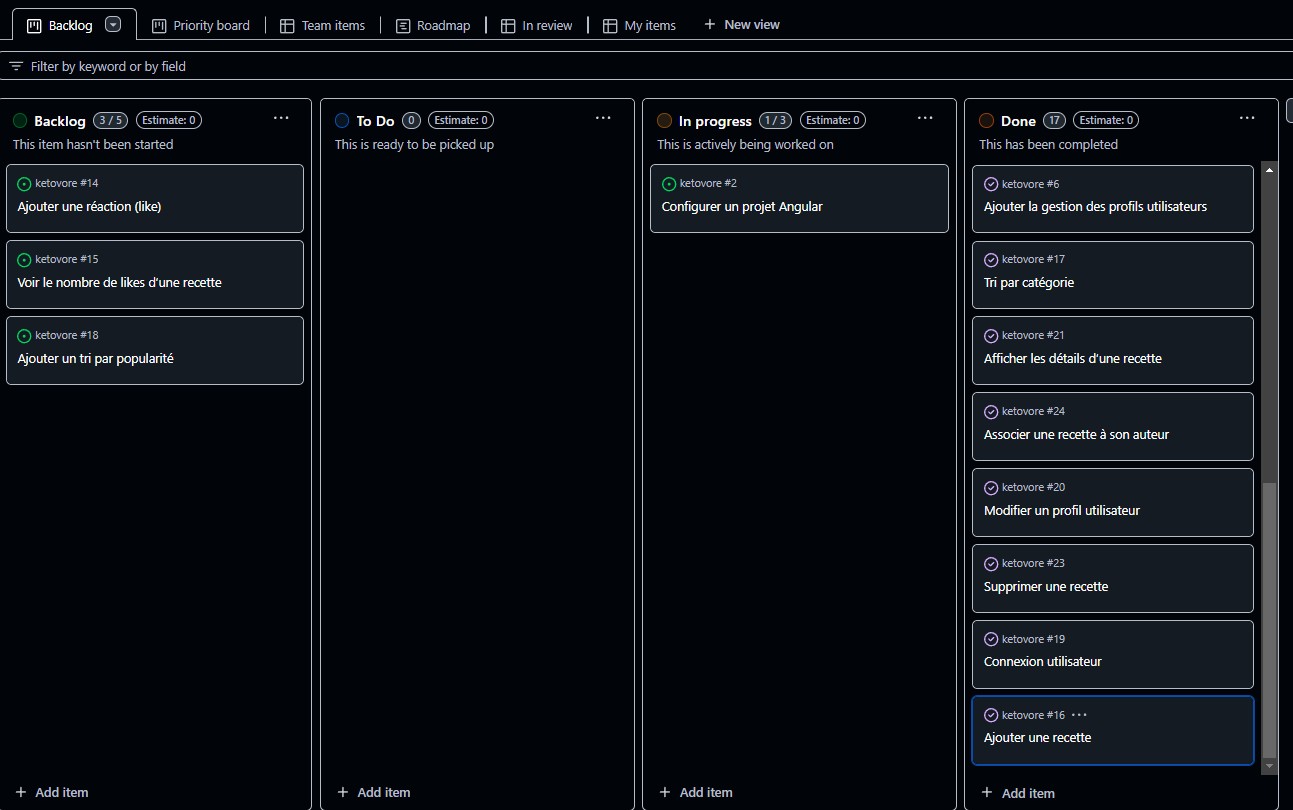
Pour sécuriser le déploiement, j’ai effectué une **sauvegarde complète** des fichiers et de la base de données avant chaque mise en production. Des **tests automatisés** ont été intégrés dans le pipeline pour valider les fonctionnalités critiques et prévenir d’éventuelles erreurs. Ces tests couvraient la conformité du code ainsi que des scénarios fonctionnels pour s’assurer du bon comportement de l’application. Une fois le déploiement terminé, des outils de **monitoring** ont été mis en place afin de détecter rapidement d’éventuelles anomalies et d’ajuster les configurations si nécessaire.

## **Fonctionnalités principales couvertes par les spécifications**

Les spécifications fonctionnelles définies au cours du projet couvrent plusieurs aspects clés de l’application, notamment la **création, la modification et la suppression de recettes**, la gestion des catégories, l’ajout de **commentaires et de notes**, l’**authentification et le contrôle des accès**, ainsi que la **recherche avancée** de recettes par filtres. Chaque fonctionnalité a été accompagnée de **critères d’acceptation précis** pour garantir une validation conforme aux attentes des utilisateurs. Les retours obtenus lors des tests ont été pris en compte pour affiner ces spécifications et améliorer continuellement l’expérience utilisateur.

## **Objectifs de qualité et conclusion**

L’ensemble de ces démarches a permis d’optimiser la gestion du projet, en assurant **une vision claire des objectifs**, **une flexibilité dans l’adaptation des spécifications** et **une meilleure collaboration** entre les différents acteurs impliqués. Grâce à cette approche structurée et agile, le développement de **Délices Keto** a progressé de manière efficace, avec un souci constant de **qualité et de stabilité**.



.

# **6. Formalisation UML**

## **Modélisation des interactions utilisateur**

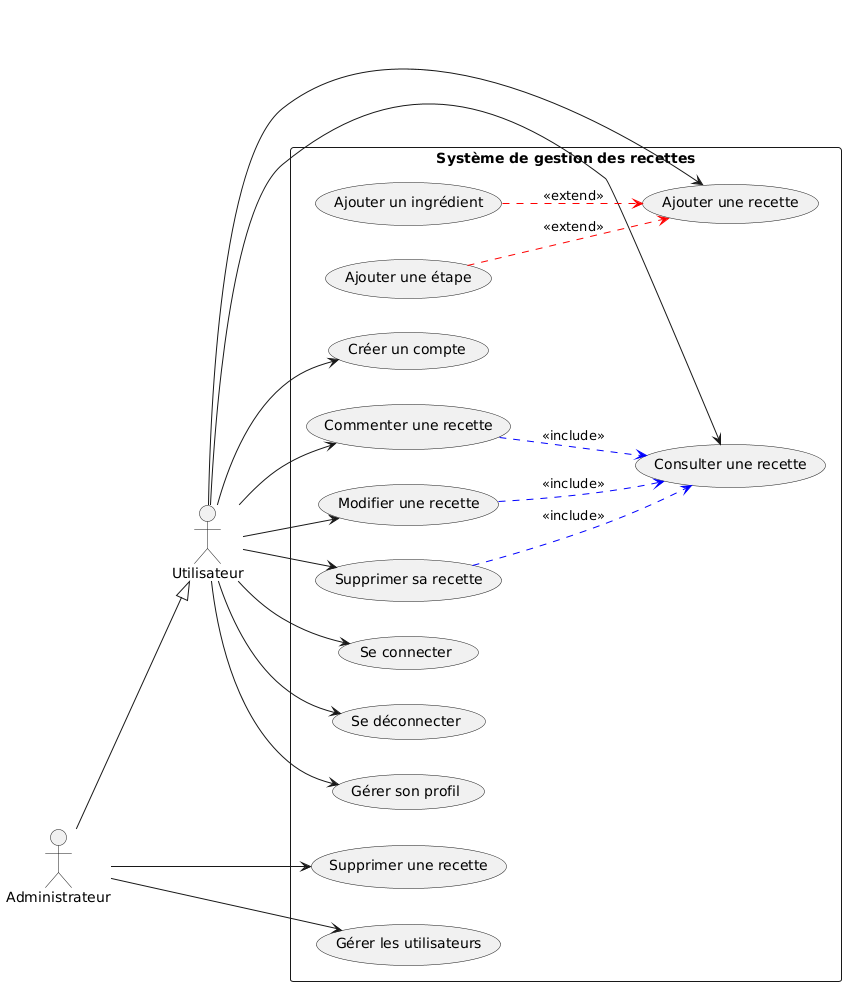
Afin de mieux comprendre le fonctionnement de l’application et ses différentes interactions, plusieurs diagrammes UML ont été réalisés. Ces diagrammes permettent de modéliser les aspects fonctionnels et structurels du projet, facilitant ainsi sa conception et son développement.

***Le diagramme de cas d'utilisation*** permet de visualiser les interactions entre les utilisateurs et le système. Il représente les différentes fonctionnalités accessibles selon le rôle de l'utilisateur.  
Dans notre projet, nous avons principalement deux types d’utilisateurs :

* Utilisateur standard (ajoute/modifie des recettes, interagit avec la communauté)
* Administrateur (gère les utilisateurs et modère le contenu)

**Ce diagramme met en avant les principales fonctionnalités offertes par l’application** :

* Un utilisateur peut s’inscrire, se connecter, créer des recettes, modifier ses propres recettes, ajouter des commentaires et noter les recettes.
* Un administrateur peut modérer les recettes et les commentaires, et gérer les utilisateurs en cas de non-respect des règles.
* Les utilisateurs peuvent rechercher des recettes grâce à un moteur de recherche avancé.
* Le système assure la gestion des rôles et permissions pour garantir la sécurité des actions.

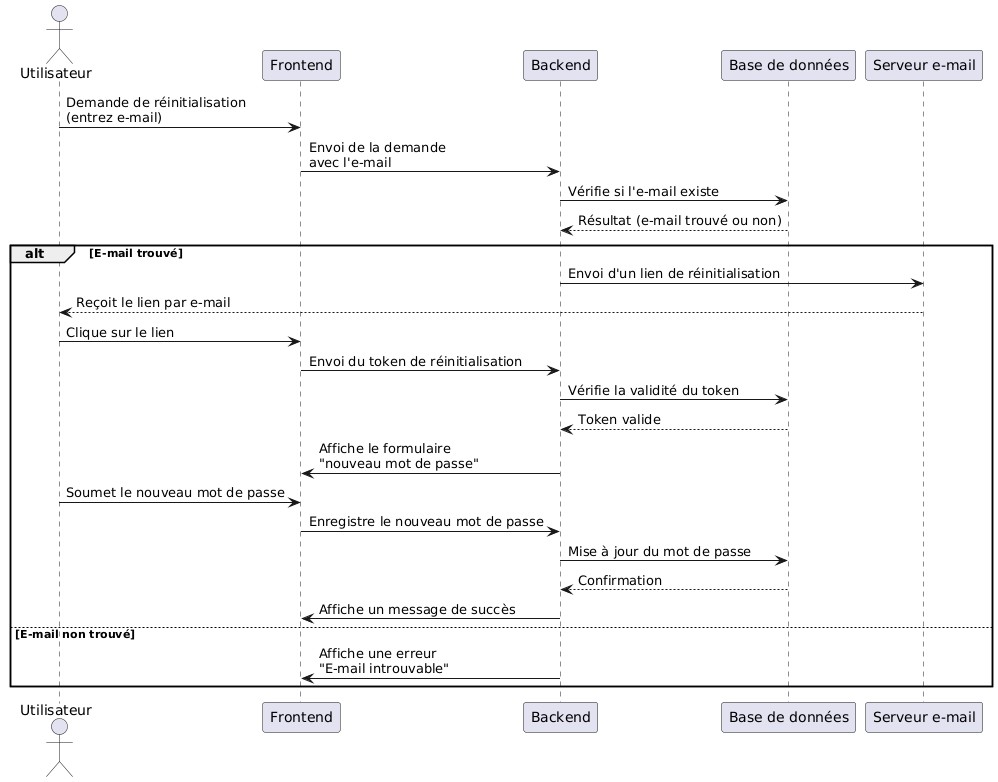


***Le diagramme de séquence*** décrit **le déroulement des échanges** entre les différents composants du système lors d’un scénario précis.  
Il montre **l'ordre des interactions** entre :

1. **L’utilisateur** (qui effectue une action)
2. **L’interface utilisateur (frontend)** (qui capture l’action)
3. **Le serveur Symfony (backend)** (qui traite la requête)
4. **La base de données** (qui stocke et récupère les données)

Ce diagramme illustre, par exemple, **le processus d’ajout d’une recette** :

* L’utilisateur remplit un formulaire et valide l’envoi.
* Le frontend **envoie la requête** au backend via une API REST.
* Le backend **valide les données** et les enregistre en base.
* Une **réponse** est renvoyée au frontend pour confirmer l’ajout.
* La nouvelle recette apparaît dans la liste des recettes de l’utilisateur.



# **Modélisation des entités et du comportement**

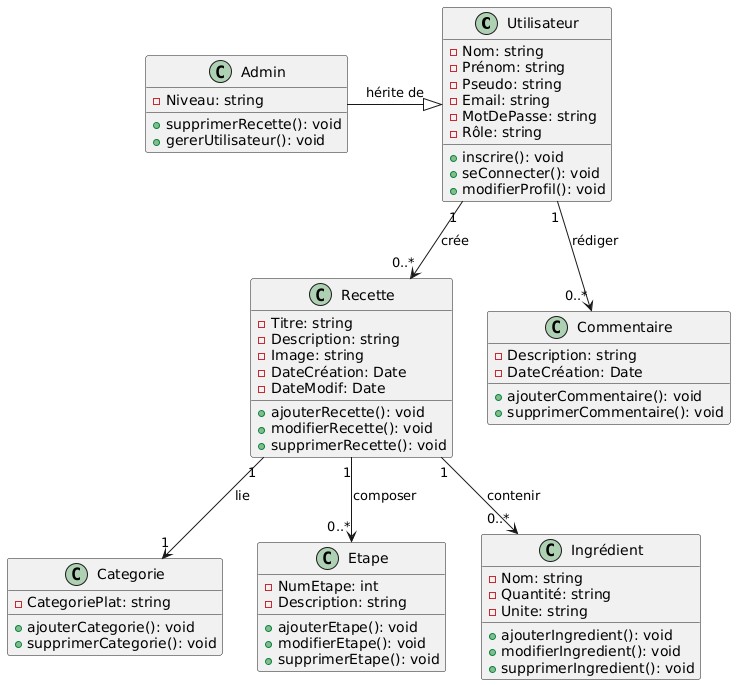
***Le diagramme de classes*** représente **les différentes entités du projet**, leurs **attributs** et leurs **relations**.  
Il permet de définir **la structure des données** et facilite l’implémentation de la base de données dans Symfony avec Doctrine ORM.

**Ce diagramme illustre les principales entités du projet et leurs relations :**

* Utilisateur : Représente les utilisateurs du site, avec leurs informations personnelles (nom, prénom, pseudo, email, mot de passe, rôle).
* Admin : Hérite de la classe Utilisateur et possède des fonctionnalités supplémentaires comme supprimer une recette et gérer les utilisateurs.
* Recette : Contient les informations essentielles d’une recette (titre, description, image, date de création, date de modification).
* Commentaire : Permet aux utilisateurs de rédiger des commentaires sur les recettes.
* Catégorie : Regroupe les recettes sous différentes catégories (ex : Entrées, Plats, Desserts).
* Étape : Décrit les étapes de préparation d’une recette.
* Ingrédient : Liste les ingrédients nécessaires pour une recette avec leur quantité et unité de mesure.

**Relations entre les entités :**

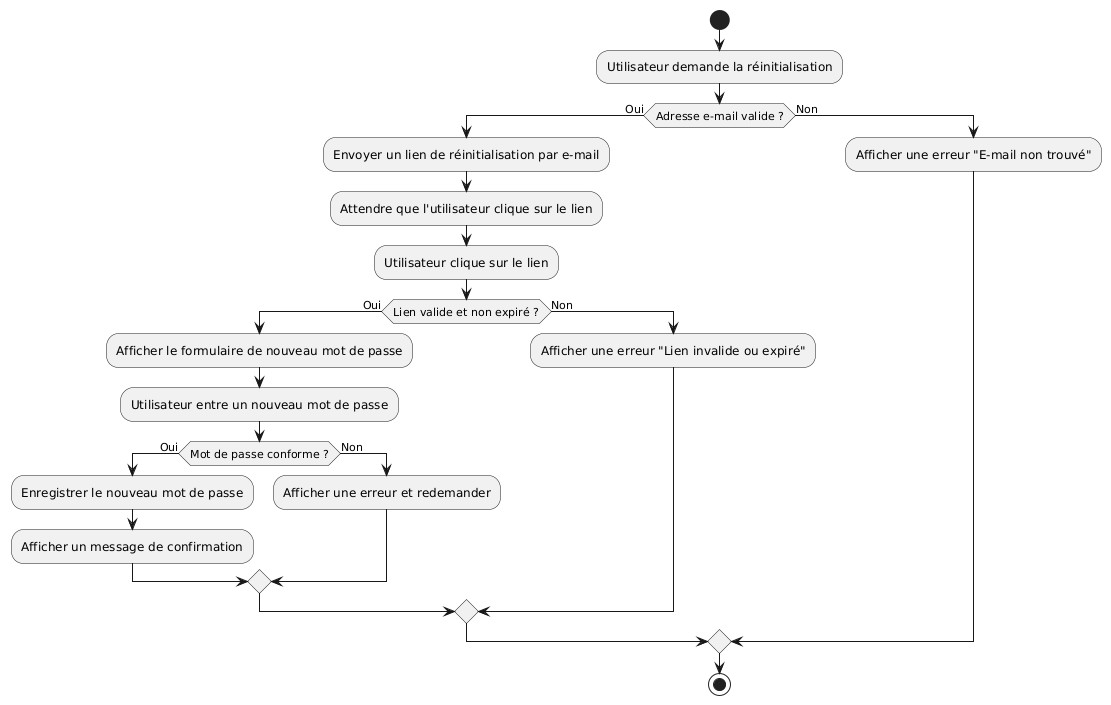
* Un Utilisateur peut créer plusieurs Recettes.
* Un Utilisateur peut rédiger plusieurs Commentaires.
* Une Recette appartient à une seule Catégorie, mais une Catégorie peut contenir plusieurs Recettes.
* Une Recette est composée de plusieurs Étapes et peut contenir plusieurs Ingrédients.



***Le diagramme d’activité*** représente **le déroulement des actions** dans le projet.  
Il est utilisé pour **modéliser un processus spécifique**, comme l’inscription d’un utilisateur ou la validation d’une recette.

**Ce diagramme décrit** **le processus de réinitialisation du mot de passe** :

1. L’utilisateur demande une réinitialisation.
2. **Vérification de l’adresse e-mail** : Si elle est valide et existe en base, un **lien de réinitialisation est envoyé**. Sinon, un message d’erreur est affiché.
3. L’utilisateur reçoit un email et **clique sur le lien de réinitialisation**.
4. **Vérification du lien** : Si le lien est valide et non expiré, le formulaire de saisie du nouveau mot de passe est affiché. Sinon, un message d’erreur s’affiche.
5. L’utilisateur entre un **nouveau mot de passe**.
6. **Vérification de la conformité du mot de passe** :
   * Si le mot de passe respecte les critères de sécurité, il est **enregistré** et un message de confirmation est affiché.
   * Sinon, un message d’erreur s’affiche et l’utilisateur doit recommencer.



Ces modèles UML fournissent une vision claire et précise de la structure et du comportement de l’application, garantissant ainsi une bonne compréhension du projet par toutes les parties prenantes.

# **7. Technologies et outils utilisés**

## **Présentation des technologies**

***Symfony pour le backend***

Pour le projet, le choix des outils répond à des critères de performance, de scalabilité et de sécurité. Symfony a été retenu pour son architecture robuste, ses bonnes pratiques intégrées, et ses nombreuses fonctionnalités facilitant le développement rapide d'applications web complexes.

Il offre également une excellente gestion des entités avec Doctrine et une prise en charge native des API RESTful, ce qui permet une organisation optimale des échanges de données entre le frontend et le backend.

C’est un Framework PHP open source, puissant et flexible, conçu pour simplifier le développement d'applications web. Il s’appuie sur l’architecture MVC (Model-View-Controller), qui organise le code en trois couches : le modèle (logique métier et données), la vue (affichage avec Twig) et le contrôleur (gestion des requêtes et des réponses). Symfony respecte également les standards PSR (PHP Standards Recommandations),

comme PSR-4 pour le chargement automatique des classes, garantissant un code propre et interopérable. Avec des outils comme le gestionnaire de routes, le moteur de services et Doctrine pour l’ORM, il facilite la création d’applications sécurisées, performantes et évolutives. Une documentation complète, une communauté active et une compatibilité avec les meilleures pratiques du web en font un choix incontournable pour les projets de toutes tailles.

***Angular pour le frontend***

Angular a été privilégié pour le frontend en raison de sa capacité à créer des interfaces utilisateur dynamiques et réactives. Il permet de structurer le code proprement grâce à son système de composants réutilisables, ce qui facilite l'évolution de l'application.

***MariaDB pour la base de données***

MariaDB (MySQL) a été choisi pour sa fiabilité, ses performances éprouvées en environnement production, et sa compatibilité avec l’ORM Doctrine. Son modèle relationnel structuré permet une gestion efficace des données, notamment pour organiser les recettes, utilisateurs et commentaires du site.

***MongoDB pour le suivi des visiteurs***

MongoDB est une base de données NoSQL orientée documents, utilisant un format BSON (une extension de JSON). Contrairement aux bases relationnelles, elle stocke les données sous forme de documents flexibles, facilitant la scalabilité et la manipulation des données sans schéma rigide.

MongoDB est particulièrement adaptée aux applications nécessitant des performances élevées et une grande évolutivité, comme les systèmes de Big Data ou les applications cloud.

Pour assurer un suivi efficace des visiteurs sur le site, j’ai intégré **MongoDB** comme base NoSQL dédiée au stockage des logs de navigation. Contrairement à MariaDB, qui impose une structure rigide, MongoDB permet de stocker et manipuler les données sous forme de **documents JSON**, offrant ainsi une meilleure flexibilité et des performances accrues pour l’enregistrement des visites en temps réel.

À chaque passage sur le site, les informations essentielles comme la **page consultée** et l’**horodatage de la visite** sont enregistrées. Ces données peuvent ensuite être exploitées pour analyser la fréquentation, identifier les pages les plus populaires et adapter l’ergonomie du site en fonction des habitudes des utilisateurs. L’analyse des pics de trafic permet également d’optimiser les ressources serveur pour garantir de meilleures performances.

Dans ce projet, **MariaDB et MongoDB sont complémentaires**. MariaDB stocke les données structurées comme les utilisateurs et les recettes, tandis que MongoDB est utilisé pour les logs de navigation. Cette approche hybride garantit à la fois **une organisation stable et une gestion efficace des données dynamiques** liées à l’activité des visiteurs.

***Visual Studio Code***

Visual Studio Code (VS Code) est un éditeur de code open-source et multiplateforme développé par Microsoft.

Léger et puissant, il est conçu pour prendre en charge un large éventail de langages de programmation tels que JavaScript, Python, PHP, C++, et bien d'autres. Sa flexibilité repose sur une vaste bibliothèque d'extensions, qui permet d'ajouter des fonctionnalités comme l'intégration de Git, le débogage en temps réel ou le support pour des Framework spécifiques. VS Code offre une expérience de développement optimisée grâce à des outils intégrés tels que l'autocomplétions intelligente (IntelliSense), la coloration syntaxique et un terminal intégré. Compatible avec Windows, MacOs et Linux, il s'adapte à tous les environnements. Son interface personnalisable et sa communauté active en font l'un des éditeurs de code les plus populaires auprès des développeurs, qu’ils soient débutants ou expérimentés.

***GitHub pour la gestion de version***

GitHub est utilisé pour la gestion de version et la collaboration entre développeurs. Chaque modification du code est suivie via un système de branches, permettant une intégration continue fluide et sécurisée.

***Jenkins pour le déploiement***

Pour assurer une intégration continue et un déploiement automatisé, Jenkins a été mis en place. Cet outil permet d’automatiser les tests et les mises en production, garantissant ainsi une livraison rapide et fiable en limitant les erreurs humaines.

***SonarQube pour l’analyse de code***

Enfin, SonarQube est utilisé pour un contrôle rigoureux de la qualité du code. Cet outil détecte les failles de sécurité, les mauvaises pratiques et les problèmes de maintenabilité. En couplant SonarQube avec Jenkins, on s'assure d'une qualité constante du code avant chaque déploiement, garantissant ainsi la pérennité et la sécurité du projet​.

***HTML (HyperText Markup Language)*** est leLangage standard utilisé pour structurer le contenu des pages web. Il permet de définir l'organisation et la hiérarchie des éléments, comme les titres, paragraphes, images, liens, listes, tableaux, et bien d'autres. Composé de balises, HTML offre une structure lisible pour les navigateurs, qui interprètent et affichent les pages pour les utilisateurs. Il est le socle de tout site web, souvent utilisé conjointement avec CSS pour le style et JavaScript pour les interactions. Facile à apprendre, HTML est indispensable pour créer des pages web fonctionnelles, qu’il s’agisse de projets simples ou de sites complexes nécessitant une base solide.

***CSS (Cascading Style Sheets)*** est un langage de feuille de style utilisé pour décrire la présentation visuelle des pages web écrites en HTML. Il permet de contrôler l’apparence des éléments, notamment les couleurs, les polices, les marges, les espacements, les alignements et les tailles. Grâce à CSS, les développeurs peuvent séparer le contenu (HTML) de la mise en forme, rendant le code plus structuré et facile à maintenir. CSS offre des fonctionnalités avancées, comme les animations, les transitions, et les médias queries, qui permettent de créer des designs responsifs s'adaptant aux différents écrans (mobile, tablette, ordinateur). Son système de cascade et d’héritage facilite la gestion des styles, en attribuant des priorités selon leur emplacement. Essentiel pour le développement web, CSS est utilisé dans tous les sites modernes pour créer des interfaces attractives et interactives.

***TypeScript (TS)*** est un langage de programmation open-source développé par Microsoft, qui étend JavaScript en y ajoutant un système de typage statique. Conçu pour améliorer la robustesse et la maintenabilité des applications, TypeScript permet aux développeurs de définir les types des variables, fonctions et objets, réduisant ainsi les erreurs courantes dès la phase de développement grâce à une vérification anticipée. Compatible avec JavaScript, TypeScript est transcompilé en JS standard, ce qui lui permet de fonctionner sur tous les navigateurs et environnements prenant en charge JavaScript. Il introduit également des fonctionnalités modernes comme les interfaces, les classes, les modules et la compatibilité avec les dernières normes ECMAScript. Adopté dans de nombreux projets, notamment Angular, TypeScript offre une expérience de développement optimisée grâce à son intégration avec des IDE comme Visual Studio Code, proposant l’autocomplétions et la documentation en temps réel. Il est devenu un outil incontournable pour le développement d’applications web complexes et évolutives.

***Bootstrap*** est un Framework CSS open-source largement utilisé pour créer des interfaces web modernes, responsives et esthétiques. Développé initialement par Twitter, il offre une collection d’outils prêts à l’emploi, tels que des grilles flexibles, des composants préconçus (boutons, cartes, barres de navigation, etc.) et des utilitaires CSS pour simplifier le développement front-end. Avec son système de grille basé sur Flexbox, Bootstrap facilite la conception de sites adaptés à tous les écrans, du mobile au bureau, selon l’approche "mobile first". En plus des styles CSS, il inclut des composants JavaScript interactifs comme des modals, des carrousels et des dropdowns, qui enrichissent l’expérience utilisateur. Son installation simple et sa documentation claire en font un outil accessible aux débutants tout en offrant des fonctionnalités puissantes pour les développeurs expérimentés. Bootstrap est un choix incontournable pour accélérer le développement de projets web élégants et professionnels.

***MySQL*** est un système de gestion de bases de données relationnelles open-source, réputé pour sa fiabilité, ses performances et sa simplicité. Créé en 1995, il est basé sur le langage SQL (Structured Query Language) pour interagir avec les données. MySQL est largement utilisé pour gérer des bases de données dans des applications web, des systèmes de gestion de contenu et des applications métiers. Il offre des fonctionnalités avancées telles que la gestion des transactions, le support des clés étrangères, des requêtes complexes, et la réplication de données pour assurer la haute disponibilité. Compatible avec plusieurs systèmes d'exploitation, MySQL s'intègre facilement dans des environnements variés, notamment avec PHP dans le cadre du développement web. Grâce à une communauté active et une documentation complète, MySQL reste un choix populaire, que ce soit pour des projets simples ou des applications complexes nécessitant une gestion robuste des données.

***PhpMyAdmin*** est un outil open source écrit en PHP, conçu pour gérer facilement les bases de données MySQL et MariaDB via une interface web conviviale. Il permet d’exécuter diverses opérations sur les bases de données sans avoir à écrire de commandes SQL complexes, bien que l'exécution de requêtes personnalisées reste possible. Parmi ses fonctionnalités principales, on trouve la création, la modification et la suppression de bases, de tables, de colonnes ou d'index. Il offre également des outils pour importer, exporter et sauvegarder les données, ainsi que pour gérer les utilisateurs et leurs permissions. PhpMyAdmin est multilingue et s'intègre aisément avec différents serveurs web, ce qui le rend accessible pour les développeurs et les administrateurs système. Son interface intuitive et sa robustesse en font l'un des outils les plus populaires pour la gestion de bases de données, surtout pour les projets utilisant MySQL ou MariaDB.

***PHP (Hypertext Preprocessor)*** est un langage de programmation open source, principalement utilisé pour le développement web. Créé en 1995, il s’exécute côté serveur et permet de générer du contenu dynamique, comme des pages web interactives. Facile à apprendre, il est largement adopté grâce à sa flexibilité et à sa compatibilité avec de nombreux serveurs et systèmes d’exploitation. PHP est utilisé dans des Frameworks populaires comme Symfony et Laravel, ainsi que dans des systèmes de gestion de contenu comme WordPress et Drupal. Sa syntaxe intuitive, combinée à une vaste bibliothèque de fonctions, permet de développer rapidement des applications robustes. Au fil des versions, PHP a considérablement évolué, intégrant des fonctionnalités modernes comme la programmation orientée objet, la gestion des erreurs avancée et des performances accrues.

Aujourd’hui, PHP reste un pilier incontournable du développement web.

## **Installation et configuration des outils**

Pour démarrer mon projet, j’ai installé les outils nécessaires en suivant une approche méthodique. Tout d’abord, j’ai téléchargé et installé Visual Studio Code (VSCode), un éditeur de code très populaire et polyvalent. Je me suis rendu sur le site officiel de VSCode, où j’ai choisi la version compatible avec mon système d’exploitation, puis je l’ai installé en suivant les instructions fournies. Ensuite, j’ai configuré VSCode en ajoutant des extensions utiles pour mon projet, comme PHP Intelephense pour le développement Symfony, ainsi que Prettier pour le formatage automatique du code. J’ai également installé le terminal intégré pour exécuter directement les commandes de Symfony et gérer mon projet plus efficacement. J’ai ensuite installé PHP, Composer, et Symfony CLI pour configurer l’environnement backend. J’ai vérifié leurs installations en exécutant les commandes php -v, composer -v, et symfony -v. Enfin, j’ai créé un dossier de projet et utilisé symfony new pour initialiser la structure de base, prêt à commencer le développement.

## **Justification des choix techniques**

Le choix de Symfony et Angular repose sur plusieurs critères :

* Symfony : un Framework robuste avec une gestion avancée des services, une compatibilité avec Doctrine ORM et un support long terme.
* Angular : un Framework frontend structuré, permettant le développement d’interfaces modulaires et réactives.
* MariaDB : une base de données relationnelle performante, offrant un bon support de réplication et une intégration native avec Symfony
* **MongoDB est utilisé pour stocker et analyser les données des visiteurs**. Grâce à son architecture distribuée, il permet de gérer efficacement de **grands volumes de logs de visite** et d’offrir des insights sur l’utilisation du site. De plus, il facilite **l’intégration avec Symfony** grâce à son API riche et ses drivers compatibles.

# **8. Choix de l’architecture**

## **Choix de l'architecture MVC**

Pour Délices Keto, j’ai opté pour l’architecture MVC (Modèle-Vue-Contrôleur) afin de structurer le code de manière claire et évolutive. Le Modèle gère les données et la logique métier via Doctrine ORM, qui simplifie les interactions avec la base de données et l’organisation des entités comme les utilisateurs, recettes et commentaires.

La Vue, assurée par Twig et Angular, offre une interface fluide et réactive, facilitant la personnalisation et l’affichage des contenus. Le Contrôleur gère les requêtes et les réponses en assurant la communication entre le Modèle et la Vue. Cette séparation améliore la maintenance du projet et permet l’intégration de fonctionnalités avancées comme l’authentification et les APIs RESTful.

## **Interactions entre les composants**

Les contrôleurs traitent les requêtes des utilisateurs et interagissent avec Doctrine ORM pour récupérer ou modifier les données. Les repositories permettent d’exécuter des requêtes spécifiques sans surcharger la logique métier. Les services centralisent les traitements complexes pour éviter la redondance du code.

Les réponses envoyées au frontend sont formatées en JSON grâce au Serializer de Symfony, facilitant l’échange avec Angular. La sécurité repose sur JWT pour l’authentification et pour la gestion des permissions.

Cette organisation garantit un projet bien structuré, flexible et facile à faire évoluer.

**Conception de la base de données**

***Définition des entités***

Pour concevoir la structure de la base de données, j'ai commencé par analyser les besoins fonctionnels du projet afin d'identifier les entités principales et leurs relations. Le projet nécessitait de gérer des utilisateurs, des recettes, des catégories, des commentaires et des ingrédients.

L'entité Utilisateur permet de stocker les informations des utilisateurs, comme leur nom, leur email et leur rôle.

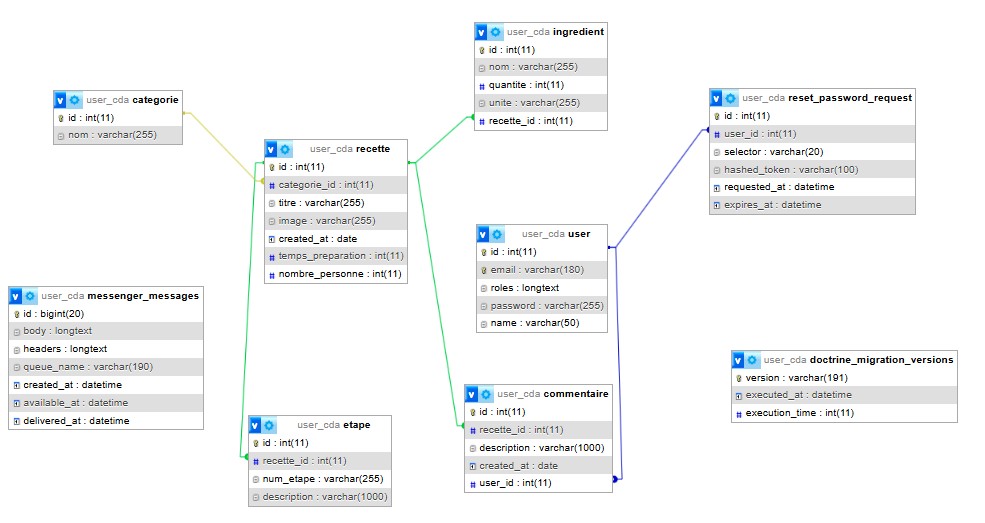
L'entité Recette représente les recettes avec des champs comme le titre, la description, et un lien vers une catégorie et son auteur.

L'entité Catégorie organise les recettes en différentes thématiques comme entrées, plats et desserts.

L'entité Commentaire permet aux utilisateurs de commenter les recettes en incluant l'identifiant de l'auteur et celui de la recette.

Enfin, l'entité Ingrédient liste les composants de chaque recette en incluant une relation avec la recette associée.

***Schéma MCD/MPD***



# ***Définition des relations entre les entités***

Les relations entre les entités ont été établies pour garantir une structure cohérente et optimiser les requêtes.

Un **utilisateur** et ses **recettes** sont liés par une relation **1-n** : un utilisateur peut créer plusieurs recettes, mais chaque recette appartient à un seul utilisateur (user\_id).

Les **catégories** et les **recettes** suivent aussi une relation **1-n** : une catégorie peut contenir plusieurs recettes, mais une recette appartient à une seule catégorie (categorie\_id).

Les **recettes** et leurs **étapes** sont en relation **1-n** : une recette peut avoir plusieurs étapes, mais chaque étape est liée à une seule recette (recette\_id).

Les **ingrédients** et les **recettes** respectent également une relation **1-n** : une recette contient plusieurs ingrédients, mais chaque ingrédient est associé à une seule recette (recette\_id).

Les **commentaires** et les **recettes** sont en relation **1-n** : une recette peut recevoir plusieurs commentaires, mais chaque commentaire est rattaché à une seule recette (recette\_id).

Enfin, les **utilisateurs** et les **commentaires** sont aussi liés par une relation **1-n** : un utilisateur peut écrire plusieurs commentaires, mais chaque commentaire appartient à un seul utilisateur (user\_id).

Ces relations **1-n** permettent une organisation claire des données et facilitent la gestion des requêtes tout en maintenant l’intégrité du système.

# ***Optimisation de l’indexation de la base de données***

Pour améliorer les performances des requêtes, j'ai utilisé des techniques d’indexation adaptées aux besoins du projet. J’ai ajouté des index sur les colonnes fréquemment utilisées dans les requêtes WHERE ou dans les clauses JOIN. Par exemple, les colonnes user\_id dans la table des recettes et recette\_id dans la table des commentaires ont été indexées pour accélérer les recherches et l’établissement des relations entre ces tables.

J’ai également mis en place des index uniques sur certaines colonnes nécessitant des valeurs distinctes, comme les emails des utilisateurs dans la table des utilisateurs. Cela garantit l’intégrité des données tout en améliorant la vitesse des requêtes sur ces champs.

Enfin, j’ai testé les performances générales de la base de données en exécutant des requêtes sur les données les plus utilisées. Ces analyses m'ont permis d’optimiser ou d’ajouter des index supplémentaires afin d’assurer une bonne réactivité du système, même en cas d’augmentation du volume des données stockées.

***Contraintes mis en place sur les enregistrements***

Pour garantir une base de données cohérente et évolutive, j’ai utilisé Doctrine, intégré à Symfony, ainsi que son système de migrations. Cela permet de suivre, appliquer et versionner les modifications du schéma de manière sécurisée.

J’ai défini les **entités** en précisant les types de colonnes, les **relations entre les tables** (OneToMany, ManyToOne, etc.) et les **contraintes** comme les clés primaires et étrangères. Par exemple, l’entité **Recette** possède une clé primaire auto-incrémentée (@Id @GeneratedValue) et une relation ManyToOne avec **Utilisateur**, assurant que chaque recette appartient à un seul utilisateur.

Les **clés étrangères** sont définies à l’aide de @JoinColumn, avec des options comme onDelete="CASCADE", permettant de supprimer automatiquement les recettes d’un utilisateur supprimé. Cela garantit l'intégrité des relations et évite les données orphelines.

Les modifications du schéma sont gérées via les **migrations Doctrine**. Après toute modification des entités, j’ai utilisé php bin/console make:migration pour générer un fichier de migration contenant les instructions SQL nécessaires, puis php bin/console doctrine:migrations:migrate pour les appliquer. Ce système assure que les changements sont effectués dans le bon ordre et facilite la gestion des mises à jour sur différents environnements (développement, test, production).

Grâce à cette approche, la base de données reste toujours synchronisée avec le code, tout en évitant les conflits et en garantissant l’intégrité des données sur l’ensemble du projet.

***Utilisation de Doctrine ORM pour la gestion des données***

Pour gérer les données relationnelles, j’ai utilisé **Doctrine** comme ORM dans mon projet Symfony. Cet outil permet de manipuler la base de données en travaillant directement avec des **entités PHP**, rendant l’interaction plus intuitive.

J’ai d’abord défini les **entités** représentant les tables, en utilisant des annotations comme @Entity, @Table et @Column pour préciser les types de données, relations et contraintes. Par exemple, l’entité **Recette** contient des propriétés comme id, titre et dateCreation, ainsi que des relations OneToMany avec les étapes et les ingrédients. Doctrine génère automatiquement les tables et met à jour la structure de la base avec des commandes comme php bin/console doctrine:schema:update.

Grâce à **QueryBuilder** et **DQL**, j’ai pu écrire des requêtes complexes sans manipuler directement du SQL. Par exemple, pour récupérer les recettes d’un utilisateur, une simple requête dans un **repository** suffit, ce qui simplifie le développement et évite les erreurs SQL.

Doctrine gère aussi la **persistance des données**. Pour ajouter une recette, il suffit d’instancier l’entité, de définir ses propriétés et d’exécuter $entityManager->persist($recette) suivi de $entityManager->flush(), assurant ainsi une gestion fiable des enregistrements.

Enfin, le **lazy loading** permet de charger uniquement les données nécessaires, optimisant l’accès aux relations complexes. En utilisant Doctrine, j’ai pu améliorer la lisibilité du code, simplifier la maintenance et garantir des performances optimales.

## **Mise en place de l’API REST**

L’API REST de Délices Keto suit les principes REST pour assurer une communication fluide entre le frontend et le backend. Chaque ressource est accessible via une URL unique, comme /api/recettes pour les recettes et /api/utilisateurs pour les utilisateurs, avec les méthodes HTTP standards : GET pour récupérer, POST pour créer, PUT pour modifier et DELETE pour supprimer. Les échanges se font en JSON, et les statuts HTTP (200, 201, 400, 404, etc.) indiquent clairement le résultat des opérations.

L’API repose sur le principe de statelessness, où chaque requête est indépendante, facilitant la scalabilité et simplifiant la gestion côté serveur. Les endpoints sont définis selon les conventions REST, comme /api/recettes/{id} pour accéder à une recette spécifique. Pour assurer la compatibilité future, la version de l’API est intégrée dans les URLs, par exemple /api/v1/recettes.

***Sécurité et gestion des erreurs***

L’authentification repose sur des tokens JWT, et les accès sont contrôlés selon les rôles des utilisateurs. Une validation stricte des données empêche les failles potentielles. En cas d’erreur, Symfony gère automatiquement les exceptions avec des réponses adaptées (400 pour une requête invalide, 403 pour un accès refusé, etc.). Le système de logs intégré permet d’analyser les erreurs et d’optimiser la gestion des incidents.

***Implémentation des appels RESTful***

Le backend, développé avec Symfony, expose les endpoints via des contrôleurs définissant les routes et les actions spécifiques. Les données sont sérialisées en JSON grâce au Serializer de Symfony, et validées avec le Validator avant traitement.

Côté frontend, les requêtes API sont gérées via HttpClient d’Angular, qui permet d’envoyer et de recevoir des données de manière asynchrone. Des tests ont été réalisés pour garantir la cohérence et la robustesse des échanges, assurant une interaction fluide et sécurisée entre le frontend et le backend.

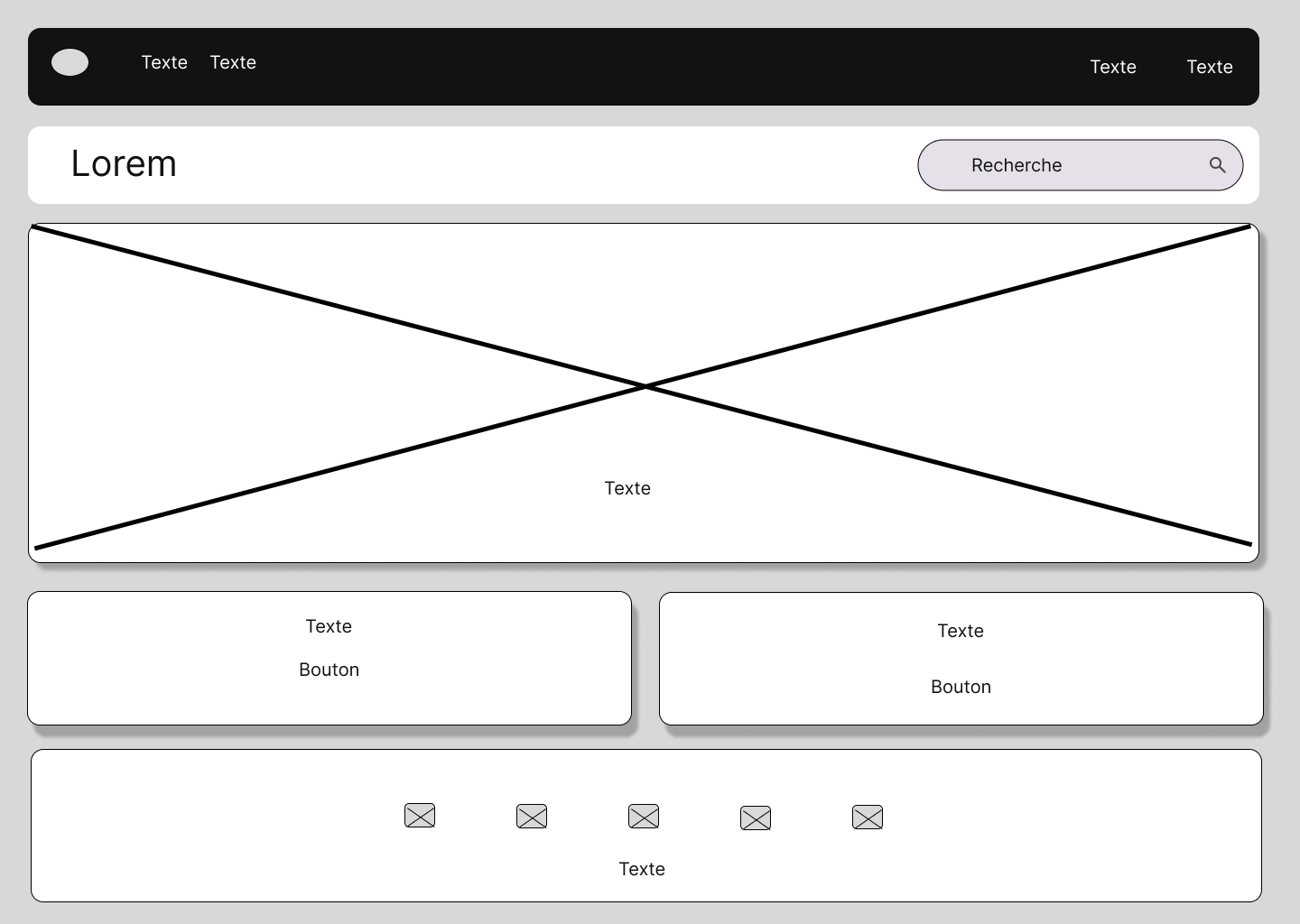
# **9. Les maquettes et enchaînement des maquettes**

## **Conception de l’Expérience Utilisateur (UX)**

***Wireframes de l’Expérience Utilisateur***

Avant de me lancer dans la création de maquettes haute-fidélité, j’ai commencé par concevoir des **wireframes** plus ou moins détaillés en utilisant Figma. Il s’agissait d’établir la structure globale du site et l’emplacement des différents éléments : barre de navigation, champ de recherche, blocs de recettes populaires et récentes. Ces schémas m’ont permis de vérifier rapidement la cohérence d’ensemble et de m’assurer que la navigation resterait intuitive, avec un point d’entrée clair dès la page d’accueil. J’ai également opté pour une grille d’images lorsqu’il s’agissait d’afficher la liste des recettes, afin de rendre la consultation plus agréable et de mettre en avant la dimension visuelle, essentielle dans un projet de cuisine.

Cette première phase m’a apporté la flexibilité nécessaire pour me concentrer sur l’ergonomie et l’expérience utilisateur, sans me laisser distraire par des choix de couleurs ou de typographies définitives. Les wireframes ont ainsi servi de base solide pour la suite, puisqu’une fois leur cohérence validée, j’ai pu passer à un design plus précis, en incorporant la palette chromatique et le style souhaité.

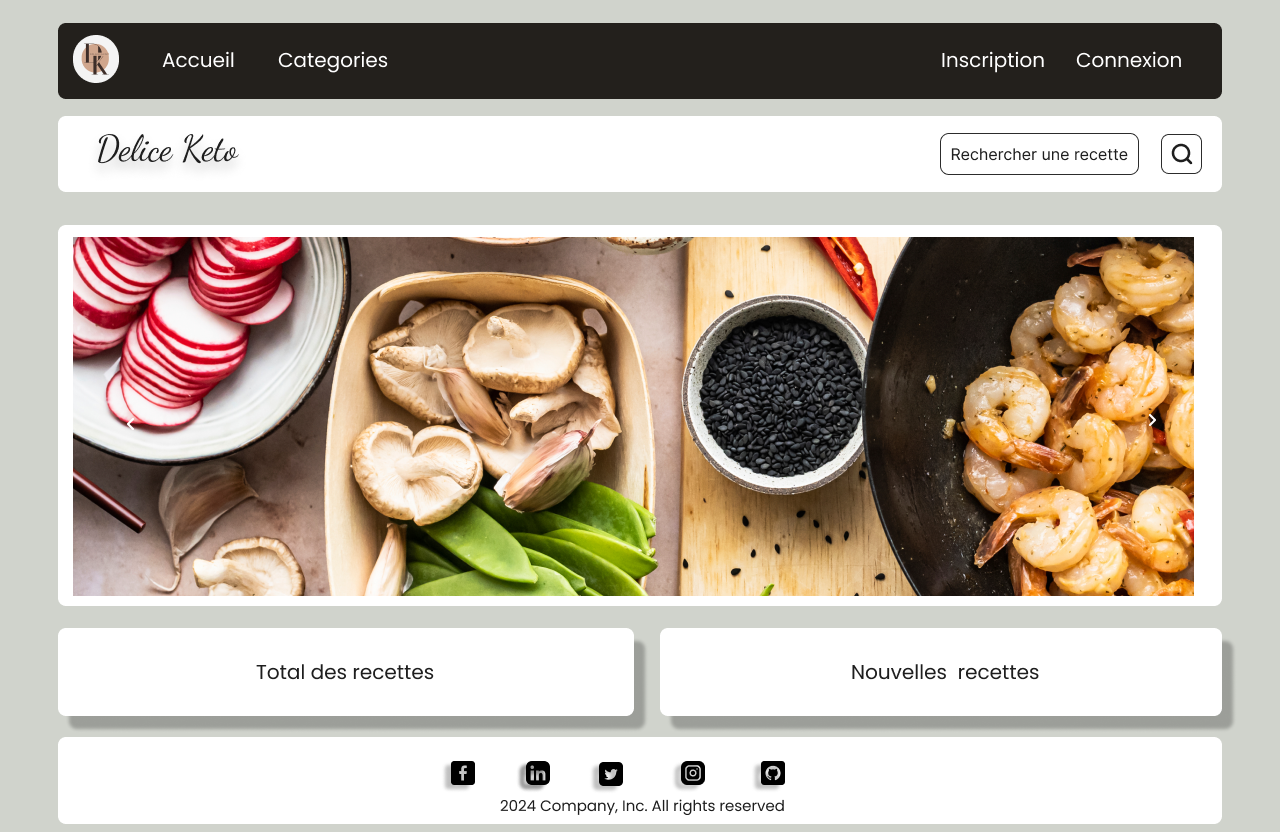


***Création des maquettes et validation de l’interface***

Pour concevoir les maquettes de ma plateforme de partage de recettes cétogènes dans le cadre de ma certification, j’ai suivi une approche progressive avec **Figma**. J’ai d’abord réalisé une première version en intégrant les couleurs de mon logo pour assurer une identité visuelle cohérente. Cette version incluait une **page d’accueil** avec une barre de navigation, une section de recherche, ainsi que des blocs pour les **recettes populaires** et les **nouvelles recettes**. Une **page de détail** affichait la photo, les ingrédients, les étapes et une section commentaires.

Cependant, après avoir terminé cette première version, j’ai constaté que les couleurs du logo alourdissaient l’interface et nuisaient à la lisibilité. J’ai donc revu la **palette de couleurs**, optant pour des **tons plus neutres et apaisants**, afin de mieux mettre en valeur les contenus essentiels et d’offrir une navigation plus fluide.

Tout au long du processus, j’ai testé les maquettes pour garantir leur **cohérence et ergonomie**. Bien que je n’aie pas eu de parties prenantes officielles, j’ai recueilli des retours auprès de mes proches et collègues pour affiner l’interface. Grâce à ces ajustements successifs, j’ai pu aboutir à une **interface harmonieuse et intuitive**, optimisée pour une meilleure expérience utilisateur.



## **Conception Interface Utilisateur (UI)**

***Typographie***

La typographie de mon site web repose sur une **combinaison harmonieuse de styles** pour offrir une expérience visuelle **élégante et agréable**.

Le **corps du texte** utilise la police **Poppins**, une **sans-serif moderne**, qui apporte une touche **contemporaine et lisible** à l’ensemble. Certains éléments spécifiques, comme **le titre principal de la barre de recherche**, adoptent la police **Dancing Script**, ajoutant une **dimension artistique et manuscrite** grâce à son effet **cursif et ses ombrages subtils**.

Les tailles et poids des polices sont soigneusement **ajustés selon le contexte**. Les **titres et éléments interactifs (boutons, liens)** adoptent une mise en **gras** pour **se démarquer et attirer l’attention**, tandis que les **textes des formulaires et des cartes de contenu** privilégient une **lisibilité sobre et élégante**.

Le **design global** est renforcé par des **effets subtils**, tels que **des ombres portées, des espacements optimisés et des animations au survol** (par exemple, **les boutons changent de couleur et deviennent gras** lorsqu’ils sont activés).

Ces **choix typographiques** s’intègrent parfaitement à une **structure visuelle cohérente et adaptative**, offrant **une navigation fluide**, même sur des **écrans plus petits grâce aux styles responsive**​.

***Couleurs utilisées***

Pour l’élaboration de mon site, j’ai choisi une **palette de couleurs harmonieuse et réfléchie**, afin d’offrir **une expérience visuelle agréable**, tout en restant en accord avec **l’objectif de ma plateforme**.

Initialement, j’ai utilisé **les couleurs de mon logo**, notamment des **tons plus vifs** pour refléter **l’identité visuelle du projet**. Cependant, après plusieurs essais, j’ai décidé d’opter pour une **palette plus douce et apaisante** afin de **mieux mettre en valeur les contenus et les images des recettes**.

La **version finale du site** présente une **dominante de couleurs neutres**, comme **le beige clair et le blanc cassé** pour les **arrière-plans**, apportant une **sensation de clarté et de simplicité**. Le **noir** a été utilisé pour le **texte et les éléments principaux**, assurant un **contraste optimal** pour une **meilleure lisibilité**.

Des **accents en gris doux** soulignent les **zones secondaires**, comme **les bordures et les placeholders**, afin de **maintenir une hiérarchie visuelle équilibrée**.

Enfin, les **images colorées des recettes** jouent un **rôle clé**, en ajoutant une **touche de dynamisme** et en rendant l’ensemble **harmonieux et accueillant**​.

***Conception du logo***

Mon logo est conçu pour **refléter l’univers de la cuisine cétogène** tout en étant **moderne et épuré**. Il se compose des **lettres "D" et "K"**, entrelacées harmonieusement pour **symboliser "Délices Keto"**, le tout placé sur un **cercle beige** qui évoque **la douceur et l’équilibre**, des valeurs fondamentales de ce mode de vie.

La **typographie** est **fine et élégante**, tandis que la **palette de couleurs**, mêlant **noir et beige**, transmet une impression de **professionnalisme et de naturalité**.

Pour le créer, j'ai utilisé **Canva**, un **outil de design graphique intuitif**. J'ai commencé par **explorer plusieurs idées** en testant **différentes compositions et typographies**. Une fois le concept défini, j'ai **ajusté les proportions des éléments** pour créer un **design équilibré et visuellement agréable**.

J'ai sélectionné **une palette minimaliste** qui s’harmonise avec **l’identité du projet**, puis **intégré des formes géométriques** pour **renforcer l’aspect moderne**.

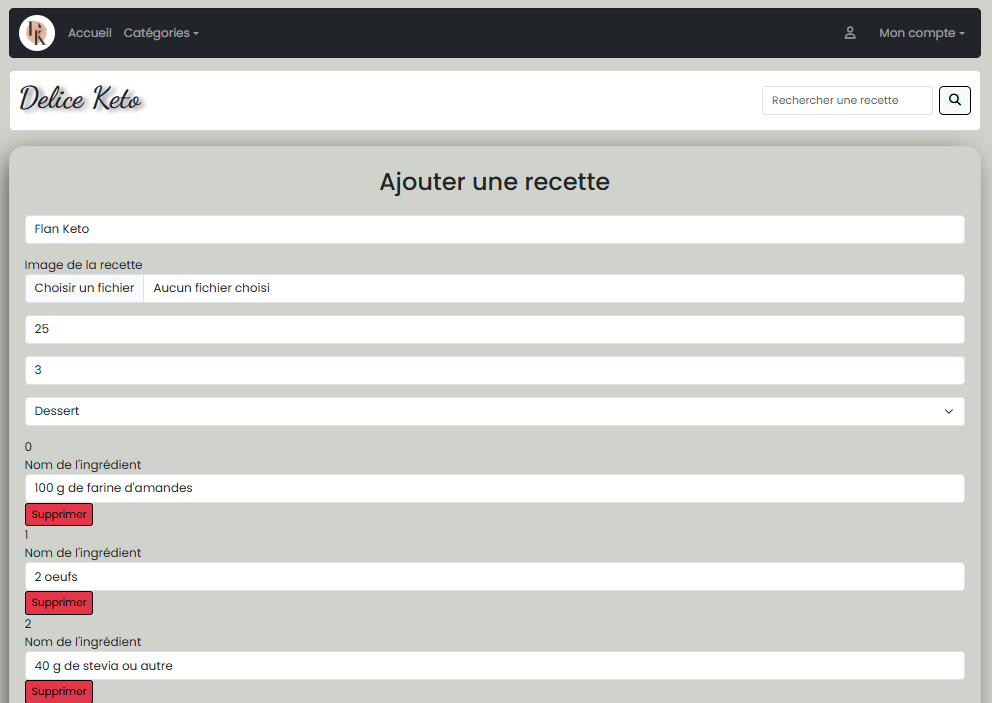
Enfin, j’ai **optimisé le logo** pour qu’il soit **adaptable à divers supports**, tels que **les réseaux sociaux ou le site web**. Cette démarche garantit **une identité visuelle forte et cohérente** pour **"Délices Keto"**​.



# **10. Fonctionnalités du site**

## **Gestion des recettes**

Les utilisateurs ont la possibilité d'ajouter, modifier et supprimer leurs propres recettes cétogènes. Cette fonctionnalité leur permet de personnaliser leur espace en ajoutant des recettes adaptées à leurs préférences alimentaires et en mettant à jour leur contenu si nécessaire.  
Les recettes peuvent être organisées par catégories et affichées sous forme de liste ou de fiches détaillées, avec des images, des ingrédients et des étapes de préparation.



## **Sécurisation et gestion des droits des recettes**

Une fonctionnalité essentielle de mon site internet consacré à l'alimentation cétogène est la gestion des recettes par les utilisateurs, leur offrant la possibilité de modifier uniquement les recettes qu'ils ont eux-mêmes ajoutées.

Cette fonctionnalité assure une personnalisation tout en maintenant la sécurité et la cohérence des données. Chaque utilisateur, après s'être authentifié, peut ajouter ses propres recettes cétogènes, les éditer ou les supprimer, selon ses besoins. Cependant, il ne peut modifier que ses propres créations, garantissant ainsi que les recettes partagées par d'autres utilisateurs restent intactes et protégées contre des modifications non souhaitées.

Cette gestion est rendue possible grâce à une base de données sécurisée où chaque recette est associée à un utilisateur spécifique via un identifiant unique. De plus, bien que les utilisateurs ne puissent pas accéder directement à la base de données, ils peuvent consulter les recettes partagées par la communauté, les commenter, et les noter, contribuant ainsi à un environnement d’échange d’idées sans compromettre la structure et l'intégrité des recettes. Cette fonctionnalité assure un équilibre entre flexibilité individuelle et sécurité des contenus partagés.

## **Moteur de recherche et filtrage des recettes**

Pour faciliter l’accès aux recettes, un moteur de recherche avancé a été intégré. Il permet aux utilisateurs de rechercher des recettes selon plusieurs critères, comme les ingrédients disponibles, le temps de préparation, ou encore les catégories spécifiques (petit-déjeuner, dîner, desserts cétogènes).  
Les résultats s’affichent en temps réel grâce à un système de filtrage dynamique, offrant ainsi une expérience utilisateur fluide et efficace.

## **Interactions communautaires**

Afin d’encourager le partage et l’échange entre les membres, les utilisateurs peuvent commenter, noter et enregistrer leurs recettes préférées.

Commentaires : Chaque utilisateur peut laisser des avis sur une recette, partager des conseils ou poser des questions.

Système de notation : Un système de notation par étoiles permet d’évaluer les recettes les plus appréciées par la communauté.

Favoris : Un utilisateur peut enregistrer ses recettes favorites dans un espace personnel pour y accéder rapidement plus tard.

## **Interface utilisateur et expérience utilisateur (UX/UI)**

***Espace utilisateur***

L’espace utilisateur offre une interface intuitive et épurée où chaque membre peut personnaliser son expérience sur le site. Après inscription, un profil utilisateur est généré, permettant la gestion des informations personnelles et des préférences.

Les utilisateurs peuvent sauvegarder des recettes favorites afin d’y accéder plus rapidement et de retrouver facilement les plats qu’ils souhaitent préparer. Une section spécifique regroupe toutes les recettes publiées par l’utilisateur, offrant ainsi une vision claire de ses contributions à la communauté.

L’espace utilisateur inclut également un système de notifications, permettant d’être informé des nouveaux commentaires, des réactions à ses recettes ou encore des tendances culinaires du moment. Cette fonctionnalité améliore l’engagement des membres et facilite les interactions au sein de la plateforme​.

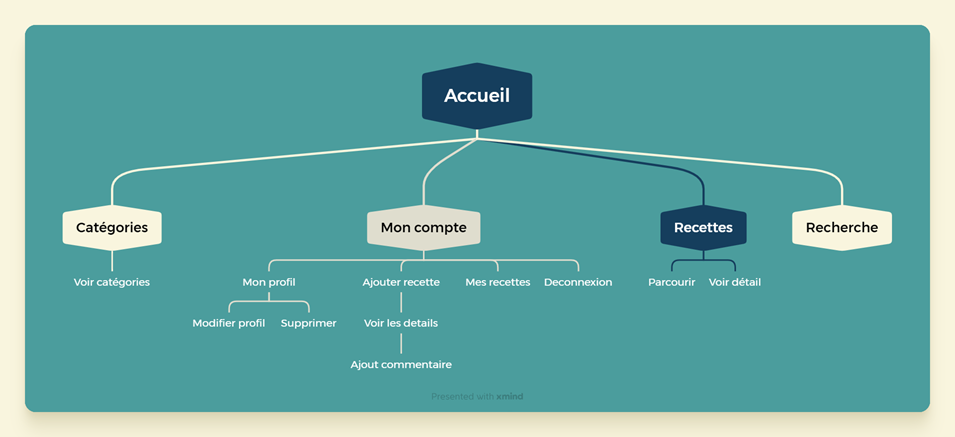
Le rôle **Utilisateur** du site "Délices Keto" permet à chaque membre inscrit de participer activement à la communauté tout en profitant des fonctionnalités disponibles.

Les utilisateurs peuvent **créer et partager des recettes**, en détaillant les ingrédients, les étapes et en ajoutant des photos pour les rendre plus attrayantes. Ils peuvent **explorer et rechercher des recettes**, en accédant à une bibliothèque organisée et en utilisant un moteur de recherche pour trouver des idées adaptées à leurs envies ou besoins spécifiques.

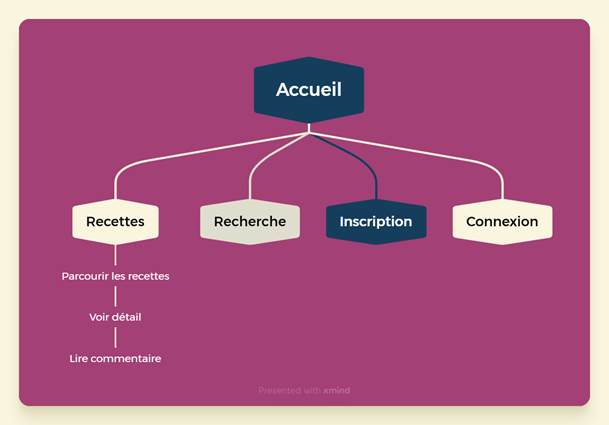
Ils ont également la possibilité d’**interagir avec la communauté** en commentant les recettes, posant des questions, partageant des conseils et donnant leurs impressions après avoir testé un plat. Les utilisateurs peuvent aussi **évaluer les recettes**, en attribuant des mentions "j’aime" afin d’aider les autres à repérer les meilleures créations.

Enfin, chaque utilisateur a le contrôle total sur ses propres publications, puisqu'il peut **modifier ou supprimer ses recettes** en cas de corrections ou mises à jour. Ce rôle favorise une participation active et encourage le partage d’idées, renforçant l’esprit communautaire autour du régime cétogène​.

**Arborescence pour l’utilisateur**



**Arborescence pour un visiteur non enregistré**



***Espace administrateur***

Le rôle **Administrateur** sur le site "Délices Keto" est essentiel pour garantir le bon fonctionnement de la plateforme, maintenir une expérience utilisateur de qualité et assurer un environnement sécurisé et respectueux pour tous les membres de la communauté.

Les administrateurs supervisent les **inscriptions et peuvent gérer les comptes utilisateurs**, notamment en suspendant ou supprimant ceux qui ne respectent pas les règles de la plateforme. Ils veillent à ce que chaque membre contribue positivement à la communauté.

Ils assurent également la **modération des contenus**, en surveillant les recettes, commentaires et autres publications des utilisateurs. Ils peuvent supprimer ou modifier des publications qui ne respectent pas les lignes directrices, comme des recettes non cétogènes ou des commentaires offensants.

Une autre responsabilité importante des administrateurs est la **validation des recettes**. Pour garantir la qualité du contenu publié, ils peuvent approuver ou rejeter les nouvelles recettes proposées avant qu'elles ne soient visibles par la communauté.

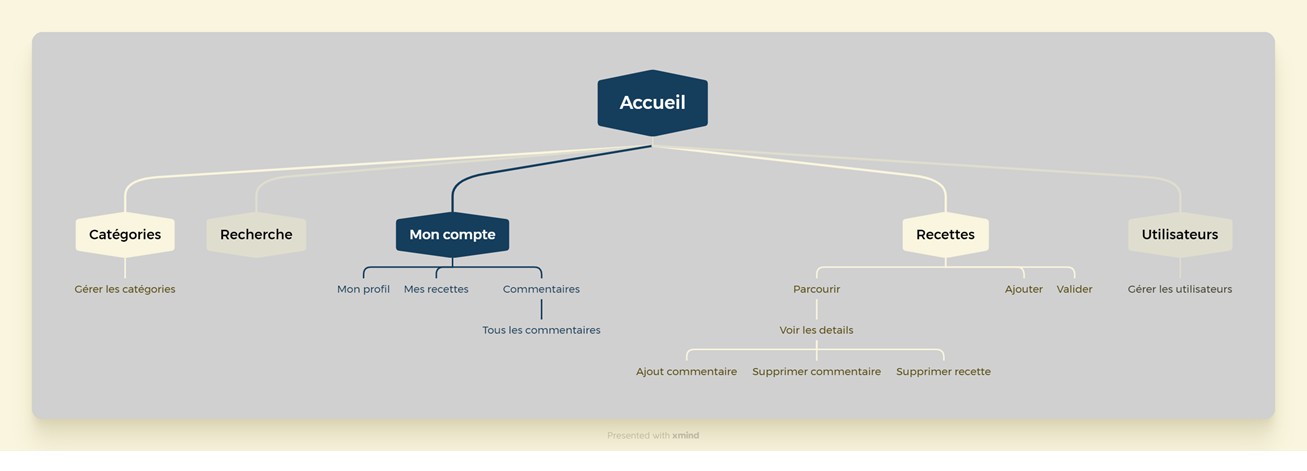
Les administrateurs gèrent aussi **l'organisation des catégories**, en ajoutant, modifiant ou supprimant des catégories pour structurer efficacement les recettes et en faciliter la recherche.

Sur le plan technique, ils assurent la **supervision du site**, signalant ou résolvant les éventuels problèmes techniques, et veillent à ce que la plateforme reste performante et accessible.

Enfin, les administrateurs réalisent une **analyse des activités** du site, en utilisant des outils dédiés pour surveiller les statistiques, comme le nombre d'inscriptions, les recettes publiées ou encore les tendances d’utilisation. Ces données leur permettent d’adapter les fonctionnalités du site aux besoins des utilisateurs et d’améliorer continuellement l’expérience offerte.

En résumé, le rôle de l’administrateur est **le pilier organisationnel et technique de la plateforme**, garantissant une expérience fluide et collaborative pour tous​.

**Arborescence de l'administrateur**



***Conception responsive et validation***

Pour garantir une expérience utilisateur optimale sur tous les types d’appareils, j’ai adopté une approche responsive en combinant plusieurs techniques de conception adaptative.

**Utilisation de Bootstrap et des media queries**

J'ai utilisé **Bootstrap**, un Framework CSS basé sur une grille fluide et flexible, pour simplifier l’adaptation de l’interface aux différentes tailles d’écran. Grâce à ses classes utilitaires, comme col-sm, col-md et col-lg, la mise en page s’ajuste automatiquement en fonction de la résolution de l’appareil, allant des smartphones aux écrans larges.

En complément, j’ai utilisé des **media queries CSS personnalisées** pour affiner certains styles selon la taille de l’écran. Cela m’a permis d’ajuster la taille des polices, les marges et les espacements pour améliorer l’ergonomie sur mobile. Les **menus de navigation** ont été adaptés en **hamburger menu** sur les petits écrans, garantissant une navigation fluide et intuitive.

Pour les médias, j'ai utilisé la classe img-fluid de Bootstrap pour rendre les images entièrement responsives. Cela permet un chargement plus rapide et une qualité adaptée à chaque périphérique.

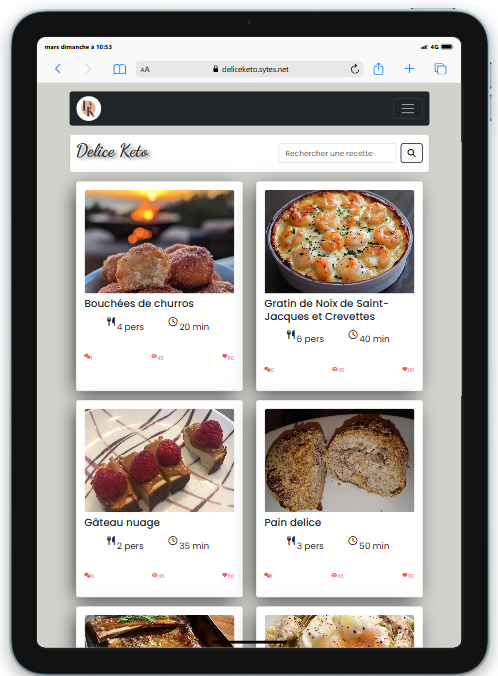
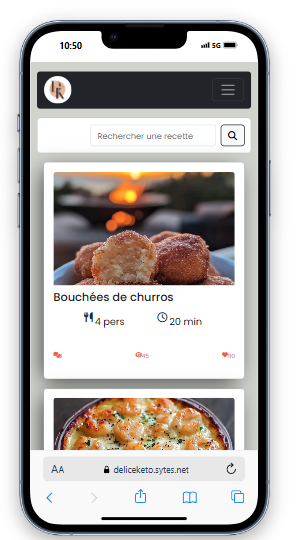
**Tests et validation de l'affichage responsive**

Pour vérifier l’efficacité de la conception responsive, j’ai effectué des **tests manuels** sur différents appareils réels, notamment **des smartphones, tablettes et ordinateurs**. Ces tests ont permis de détecter d’éventuelles incohérences dans l’affichage et d’apporter des ajustements en conséquence.

J’ai également utilisé les **outils de développement intégrés aux navigateurs**, comme le **mode responsif de Chrome et Firefox**, afin de simuler plusieurs résolutions d’écran sans nécessiter d’appareil physique.

Enfin, des tests ont été réalisés sur plusieurs navigateurs courants (Chrome, Firefox, Edge) afin de garantir une **expérience utilisateur homogène** et éviter des incompatibilités d’affichage liées à certaines configurations spécifiques.

**Affichage sur mobile et écrans larges**



***Implémentation du composant Commentaire (Angular)***

Pour concevoir et implémenter le composant d'interface utilisateur Commentaire avec Angular, j'ai utilisé une approche modulaire pour afficher les commentaires récupérés depuis une API et permettre des actions comme la suppression. Ce composant a été créé à l'aide de la commande Angular CLI ng generate component commentaire, générant une structure comprenant les fichiers nécessaires (HTML, TypeScript, CSS). Côté HTML, j'ai conçu une interface simple pour afficher une liste de commentaires. Les commentaires sont affichés dans un tableau, comprenant le contenu du commentaire, l'auteur et des boutons pour effectuer des actions, comme supprimer un commentaire. Pour un rendu responsif, j'ai utilisé les classes CSS de Bootstrap, garantissant une apparence cohérente sur tous les écrans. Côté TypeScript, le composant récupère les commentaires via un service Angular utilisant HttpClient. Une méthode getCommentaires() a été définie dans le service pour envoyer une requête GET à l’API et récupérer les commentaires sous forme de tableau. Ces données sont ensuite affichées dynamiquement dans le composant grâce à la directive \*ngFor. Pour les actions, comme la suppression, j'ai ajouté un bouton associé à chaque commentaire. Lors du clic, une méthode est appelée pour envoyer une requête DELETE à l’API, avec l’identifiant du commentaire. Après une suppression réussie, la liste des commentaires est mise à jour en temps réel pour refléter les modifications. Enfin, j'ai testé le composant en simulant des réponses API et en vérifiant que les commentaires s’affichaient correctement et que les actions, comme la suppression, fonctionnaient comme prévu. Ce composant constitue une base solide, extensible pour intégrer d'autres fonctionnalités à l'avenir.

## **Gestion métier et services backend**

***Implémentation des règles et de la logique métier côté serveur***

Pour implémenter les règles et la logique métier côté serveur de mon application, j'ai utilisé **Symfony**, un Framework offrant un cadre structuré et puissant.

J'ai commencé par **créer des entités** pour modéliser **les objets principaux du projet**, comme **les utilisateurs, les recettes, les commentaires et les catégories**. Chaque entité est **directement liée à une table dans la base de données MySQL**, permettant une **manipulation fluide des données** grâce à **l'ORM Doctrine**.

Les **contrôleurs ont été définis** pour gérer **les actions principales**, telles que **la création, la mise à jour, la suppression et la consultation des recettes ou des commentaires**.

Cependant, pour garantir une **séparation claire des responsabilités**, j'ai externalisé **la logique métier dans des services dédiés**.

Par exemple, **lors de la soumission d'une recette**, le contrôleur déléguait **au service** la tâche de **valider les données d’entrée**, comme **la vérification des champs obligatoires (titre, ingrédients, étapes) ou des formats spécifiques**. En cas d’erreur, **le service renvoyait des messages explicites** au contrôleur, qui **les transmettait ensuite à l’utilisateur via une réponse structurée**.

Pour gérer **les autorisations**, j'ai utilisé **le composant de sécurité de Symfony**. Des **contrôles d'accès ont été intégrés** au niveau **des actions critiques** pour garantir que **seuls les utilisateurs autorisés puissent modifier ou supprimer des recettes**.

J'ai défini **des rôles et des permissions spécifiques**, tels que **les rôles "auteur" et "administrateur"**, en utilisant **des annotations comme @IsGranted directement dans les contrôleurs**.

Enfin, des **tests unitaires et fonctionnels** ont été écrits pour **vérifier que chaque règle métier était appliquée correctement** et que **les résultats correspondaient aux attentes**.

Ces tests ont couvert **des cas standards et des cas limites**, garantissant **la robustesse et la fiabilité du système**.

Cette approche **structurée et méthodique** m’a permis de **concevoir une logique métier maintenable, évolutive et adaptée aux besoins de l’application**​.

***Développement des services backend***

Pour développer des **services backend** pour gérer **la logique métier** dans mon projet, j'ai adopté **une approche modulaire** en m'appuyant sur les fonctionnalités offertes par **Symfony**.

Tout d'abord, j'ai créé **des services dédiés dans le dossier src/controller**, chacun étant responsable **d'une partie spécifique de la logique métier**, comme **la gestion des utilisateurs, des recettes ou des commentaires**.

Chaque service a été conçu pour **encapsuler les règles métier** et **éviter de les dupliquer dans les contrôleurs**.

Par exemple, **un service dédié à la gestion des recettes** contenait **des méthodes pour valider les données avant leur enregistrement**, vérifier les **permissions d'accès**, et **exécuter les opérations nécessaires sur la base de données via Doctrine**.

J'ai utilisé **l'injection de dépendances** pour intégrer **d'autres services nécessaires**, comme **les repositories pour interagir avec les entités**, ou **des composants comme le logger pour tracer les actions importantes**.

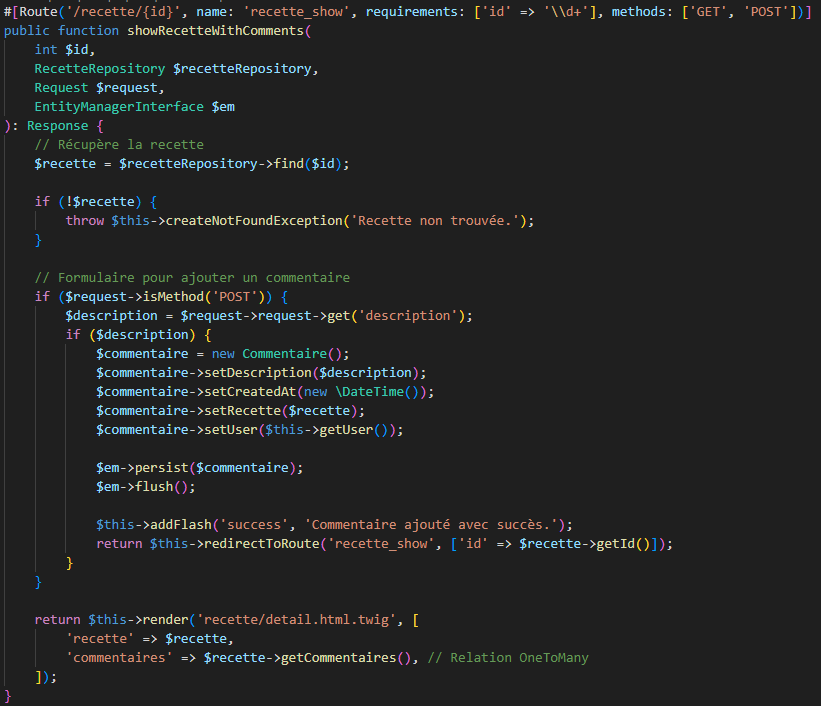
Cela a permis de **maintenir un code propre, réutilisable et facile à tester**, tout en **facilitant la maintenance future et l'évolution du projet**.

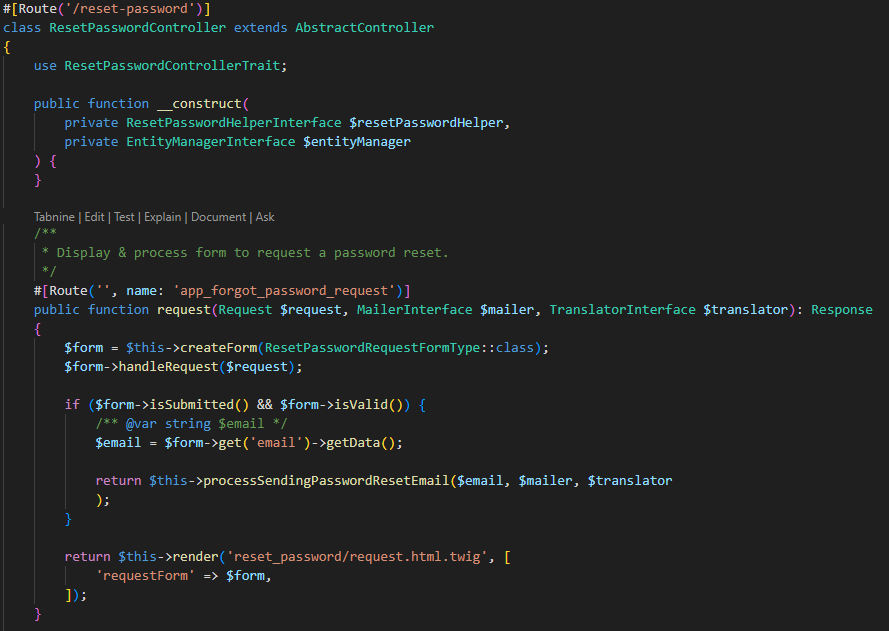
De plus, les services ont été **structurés pour rester indépendants des contrôleurs**, ce qui facilite leur **réutilisation dans différents contextes**, comme **des commandes console, des appels API ou des tests unitaires**.

Ils ont été **testés rigoureusement** pour **garantir leur fiabilité et leur conformité aux règles métier définies**.

Enfin, j'ai **vérifié que chaque service respectait les exceptions et renvoyait des messages clairs en cas d'erreur**, offrant ainsi **une expérience utilisateur optimale**​.

## ***Exemple de code Backend***





# **11. Sécurité et qualité du code**

## **Tests et validation**

Pour tester la logique métier et garantir son bon fonctionnement, j'ai mis en place une stratégie rigoureuse basée sur des **tests unitaires, des tests fonctionnels et des scénarios spécifiques**.

***Tests unitaires***

J’ai commencé par écrire des tests unitaires en utilisant **PHP Unit**, le Framework de test standard pour Symfony. Les tests unitaires permettent de vérifier que **chaque fonction ou méthode** se comporte correctement de manière isolée, sans dépendre d'autres parties du système.

Tout d'abord, j’ai installé PHP Unit via **Composer** et configuré un fichier phpunit.xml pour définir l’environnement de test, comme une base de données dédiée ou des configurations spécifiques.

Ensuite, pour chaque **fonctionnalité clé**, j’ai créé des classes de test dans un répertoire dédié (tests/). Par exemple, pour tester une méthode dans un service qui **calcule le score d’une recette en fonction de plusieurs critères**, j’ai écrit un test qui **appelle cette méthode avec des données simulées et vérifie que le résultat est correct**.

Chaque test utilise les **assertions de PHP Unit** comme assertEquals (), assertTrue () ou assertCount () pour comparer le résultat attendu avec celui obtenu.

Pour les classes ayant des **dépendances**, j’ai utilisé des **objets simulés (mocks)** grâce à des outils intégrés dans PHP Unit, permettant de remplacer ces dépendances et de me concentrer uniquement sur la logique de la méthode testée.

J’ai également configuré le **pipeline CI/CD avec Jenkins** pour exécuter ces tests **automatiquement à chaque commit**. Si un test échoue, le pipeline est interrompu, ce qui garantit que les nouvelles modifications ne cassent pas les fonctionnalités existantes. Cette automatisation, combinée aux tests unitaires bien structurés, m’a permis de **valider chaque composant individuel**, de **détecter rapidement les erreurs** et de garantir **la stabilité et la fiabilité du code**.

***Tests d’intégration***

Ensuite, j’ai défini des tests d’intégration pour vérifier les **interactions entre les différents modules**, comme les services, les contrôleurs et la base de données.

Par exemple, j’ai testé les **requêtes API** en envoyant des données et en validant les **réponses et les modifications dans la base de données**. Ces tests utilisent une **base de données temporaire** configurée via **Docker**, garantissant un environnement **isolé et reproductible**.

J’ai également testé des **cas limites** pour vérifier la robustesse du système. Cela incluait des scénarios avec des **données inattendues**, des **accès non autorisés**, ou des **charges importantes**. Ces tests ont permis d’**identifier et de corriger des failles potentielles avant la mise en production**.

***Tests fonctionnels***

Pour valider les scénarios utilisateurs, j’ai simulé des cas réels, comme :

* **L’inscription d’un utilisateur**,
* **La connexion et l’authentification**,
* **L’ajout et la suppression d’une recette**,
* **L’ajout d’un commentaire et sa modification**.

Ces tests ont été réalisés en **interagissant directement avec les points d’entrée de l’application (API ou interface utilisateur)**.

Pour les **tests fonctionnels et de bout en bout**, j’ai utilisé **des requêtes HTTP simulées** avec des outils comme curl et des **scripts automatisés** pour vérifier les réponses des différentes fonctionnalités.

***Automatisation et couverture de test***

J’ai intégré tous ces tests dans un **pipeline CI/CD via Jenkins**. Chaque **push sur le dépôt Git** déclenche automatiquement l’**exécution des tests** et génère **des rapports de couverture avec PHP Unit et SonarQube**.

Pour automatiser l'exécution des tests après chaque modification du code, j'ai utilisé **Jenkins** comme outil d'intégration continue. J'ai commencé par **installer Jenkins** sur un serveur dédié et configuré un **pipeline basé sur un fichier Jenkinsfile** pour gérer les étapes de **build, test et déploiement**.

Dans le **pipeline CI/CD**, j’ai défini une **étape spécifique** pour l’exécution des tests. Après chaque **commit** ou **pull request** sur la branche principale ou de développement, **Jenkins déclenche automatiquement un job**. Ce déclenchement est configuré via un **webhook** avec le dépôt Git, garantissant que **chaque modification est immédiatement analysée**.

Lors de l’étape de test, Jenkins utilise **PHP Unit** pour exécuter les tests unitaires et d’intégration. Les résultats sont **collectés au format JUnit**, permettant à Jenkins de générer **des rapports détaillés** et de les afficher dans l’interface utilisateur.

J’ai également intégré un **seuil de qualité** : si **certains tests échouent**, le **pipeline est arrêté immédiatement** et une **notification est envoyée** via e-mail ou un outil comme **Slack**.

Cette configuration garantit que **toutes les modifications sont validées automatiquement**, détecte rapidement **les erreurs et empêche le déploiement de code instable**, tout en **améliorant la qualité globale du projet**.

***Tests d’intégration et fonctionnels automatisés***

Pour automatiser les tests d'intégration et fonctionnels et éviter les régressions, j'ai intégré ces étapes **dans le pipeline CI/CD configuré avec Jenkins**.

Pour les **tests d'intégration**, j'ai écrit **des scripts de test personnalisés en PHP avec PHP Unit**, vérifiant les **interactions entre les différents modules et services** de l'application.

Ces tests utilisent une **base de données de test isolée**, configurée grâce à **Docker Compose**, pour garantir un environnement propre. Dans le fichier **Jenkinsfile**, j'ai ajouté **une étape qui exécute ces tests automatiquement après la phase de build**, permettant de vérifier des **cas réels** tels que la **validité des données renvoyées par les API** ou **la mise à jour correcte des enregistrements**.

Pour les **tests fonctionnels**, j'ai conçu des scripts basés sur **des scénarios utilisateurs clés**, comme **l'inscription, la gestion des recettes et les commentaires**. Ces tests fonctionnent en **interagissant directement avec l'application via des requêtes HTTP simulées avec des outils comme curl**.

Les résultats sont analysés pour **valider le bon fonctionnement des fonctionnalités essentielles**.

J’ai configuré **Jenkins** pour **exécuter ces tests à chaque commit ou pull request sur les branches principales**. Si un test échoue, **le pipeline est stoppé immédiatement pour éviter tout déploiement de code instable**.

Les **logs et rapports générés** permettent une **analyse rapide** et facilitent la correction des erreurs.

Cette stratégie garantit **un code fiable, réduit les risques de régression et simplifie la maintenance à long terme**.

## **Tests de sécurité et détection des vulnérabilités**

En plus des tests unitaires et fonctionnels réalisés avec PHPUnit, j’ai mené une série de **tests de sécurité** afin d’identifier et de corriger d’éventuelles vulnérabilités dans l’application. Ces tests ont ciblé plusieurs types de menaces courantes comme les **injections SQL**, les attaques **XSS (Cross-Site Scripting)** et les mauvaises configurations de sécurité.

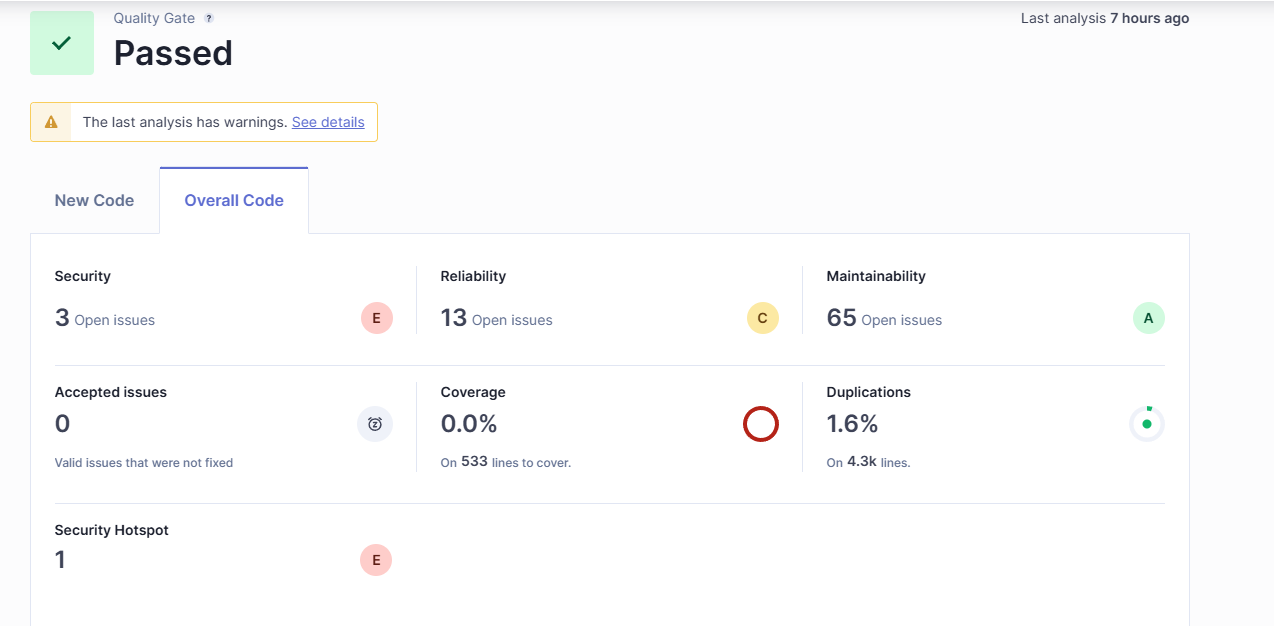
**Méthodologie des tests**

* **Injection SQL** : J’ai simulé des requêtes malveillantes dans les **formulaires d’authentification et de recherche** afin de vérifier si des **données non filtrées** pouvaient être injectées dans la base de données. L’utilisation systématique de **requêtes préparées avec l’ORM Doctrine** a permis de bloquer ces attaques​.
* **XSS (Cross-Site Scripting)** : Des scripts JavaScript malveillants ont été injectés dans les **champs de saisie utilisateur** (commentaires, descriptions de recettes) pour tester la capacité du site à **échapper et filtrer correctement les entrées**​.
* **Analyse automatisée avec SonarQube** : J’ai intégré **SonarQube** dans le pipeline CI/CD pour analyser le code source et identifier **les failles de sécurité**. Cet outil détecte les mauvaises pratiques, comme les injections SQL non protégées ou des failles XSS potentielles​.

**Résultats des tests et corrections appliquées**

Grâce aux **mesures de protection mises en place**, les tests n’ont révélé **aucune vulnérabilité exploitable** :  
✔ **Protection contre les injections SQL** via l’utilisation de **requêtes préparées avec Doctrine**.  
✔ **Sécurisation des entrées utilisateur** en filtrant et validant toutes les données avant leur traitement.  
✔ **Mécanisme de protection CSRF intégré à Symfony**, empêchant les attaques de type falsification de requêtes​.  
✔ **Intégration des tests de sécurité dans le pipeline CI/CD**, permettant de détecter **automatiquement** toute régression avant le déploiement​.

Ces actions garantissent que l’application est **conforme aux normes de sécurité actuelles**, offrant une **protection avancée contre les attaques courantes** et assurant une **expérience utilisateur sécurisée**.



Contrôle qualité

***Gestion des accès et permissions***

Pour définir et gérer les rôles et permissions dans mon application, j'ai utilisé le composant de sécurité de Symfony, connu pour sa flexibilité et sa robustesse dans la gestion des accès. J'ai d'abord défini plusieurs rôles distincts, notamment **"utilisateur"** et **"administrateur"**, chacun ayant des droits spécifiques. Par exemple, les utilisateurs peuvent consulter les recettes et laisser des commentaires, les modérateurs peuvent valider ou supprimer des commentaires, et les administrateurs disposent d'un contrôle total sur l'application, y compris la gestion des utilisateurs et des contenus.

La configuration des contrôles d'accès a été réalisée au niveau des routes et des contrôleurs en utilisant des annotations pour restreindre certaines actions selon les rôles. En complément, j'ai ajouté une logique de vérification des permissions directement dans les services pour m'assurer que les utilisateurs respectent bien les règles d'accès définies, même en cas de modification non prévue des requêtes.

Pour renforcer la sécurité et simplifier l'expérience utilisateur, j'ai implémenté un système d'authentification basé sur des sessions sécurisées, tout en prévoyant une compatibilité future avec des jetons pour des interactions API. Cette approche claire et méthodique garantit une gestion précise des rôles et permissions, tout en maintenant un haut niveau de sécurité et une flexibilité pour les futures évolutions du projet​.

### ***Sécurisation de l’application et protection des données***

Pour garantir une architecture orientée sécurité sur notre site, nous avons mis en œuvre plusieurs pratiques essentielles, couvrant la communication sécurisée, l’authentification, la gestion des données sensibles et la protection contre les attaques courantes. Voici les détails :

**1. Sécurisation des communications avec SSL**  
J'ai configuré un certificat SSL avec Let’s Encrypt pour assurer que toutes les communications entre les utilisateurs et notre site sont chiffrées. Cela garantit que les données sensibles, telles que les identifiants de connexion ou les informations personnelles, ne peuvent pas être interceptées par des tiers malveillants. Le site utilise le protocole HTTPS, vérifié par un certificat délivré par Let’s Encrypt, une autorité de confiance.

**2. Authentification sécurisée**  
Nous avons mis en place un système d'authentification robuste en utilisant les pratiques suivantes :

* Hachage des mots de passe : Les mots de passe des utilisateurs sont stockés sous forme de hachage avec l’algorithme sécurisé bcrypt, rendant impossible leur récupération en clair, même en cas de violation.

**3. Protection contre les injections SQL**

* ORM Doctrine : Toutes les interactions avec la base de données passent par l’ORM Doctrine, qui utilise des requêtes préparées, éliminant ainsi le risque d’injections SQL.
* Validation des entrées utilisateur : Les données saisies par les utilisateurs sont systématiquement validées et nettoyées à l’aide des contraintes de Symfony Validator.

**4. Gestion des sessions**Les sessions utilisateur sont sécurisées par des identifiants uniques et chiffrés. Nous avons également configuré les cookies de session avec les attributs HttpOnly (empêche l'accès JavaScript) et Secure (exige HTTPS).

**5. Protection contre les attaques CSRF**Symfony fournit un mécanisme intégré pour protéger les formulaires contre les attaques Cross-Site Request Forgery (CSRF). Chaque formulaire inclut un jeton CSRF unique, qui est validé avant l’exécution de toute action critique.

**6. Tests de sécurité continus**  
Nous utilisons des outils comme SonarQube pour effectuer des scans réguliers et corriger les vulnérabilités détectées. Ces tests couvrent les failles courantes comme les injections SQL, les attaques XSS et les mauvaises configurations.

Ces pratiques, combinées à une surveillance active et à une sensibilisation continue, assurent un haut niveau de sécurité pour nos utilisateurs et leurs données.

**Veille technologique et cybersécurité**  
Dans ma **veille en cybersécurité**, je suis particulièrement attentif à plusieurs ressources qui offrent des informations actualisées et pertinentes sur la sécurité informatique.

**1. TLDR** : Cette newsletter condensée est l’une de mes principales sources d’information. Elle me permet de rester informé des dernières évolutions en **technologie et cybersécurité** sans être submergé par un volume excessif d’informations. Les articles sont courts et précis, ce qui facilite une veille quotidienne efficace.

**2. Daily.dev** : J’ai installé cette application qui centralise les **meilleures nouvelles du développement** en provenance de **Dev.to, Medium, et d’autres blogs techniques influents**. Elle couvre également les **vulnérabilités récentes, les failles de sécurité, et les meilleures pratiques** en matière de développement sécurisé.

**3. Krebs on Security** : Je consulte régulièrement ce site qui propose **des enquêtes approfondies sur les cyberattaques et les failles de sécurité** dans les entreprises et applications populaires.

Grâce à cette **veille technologique active**, je peux suivre les **dernières tendances en cybersécurité** et **adapter les bonnes pratiques** à mes projets, garantissant ainsi une approche **proactive et sécurisée** du développement.

# **12. Déploiement automatisé avec Jenkinsfile**

## **Intégration continue et déploiement**

Pour automatiser le déploiement de l’application, j’ai mis en place un **pipeline CI/CD** défini dans un **Jenkinsfile**. Ce fichier décrit les différentes étapes nécessaires pour déployer l’application de manière efficace et reproductible.

Le **pipeline Jenkins** est déclenché automatiquement à chaque **commit** ou **pull request** sur une branche spécifique du projet, garantissant ainsi un déploiement fluide et sans intervention manuelle.

Le **Jenkinsfile** est structuré en plusieurs étapes essentielles :

1. **Préparation de l’environnement** : Récupération du code source depuis le dépôt Git.
2. **Build de l’application** : Compilation et installation des dépendances nécessaires au bon fonctionnement du projet.
3. **Déploiement sur le serveur** : L’application est copiée et mise en production directement depuis Jenkins en utilisant des commandes adaptées (comme scp pour le transfert des fichiers ou ssh pour exécuter les commandes de déploiement sur le serveur distant).

Grâce à cette approche, **le processus de déploiement est entièrement automatisé**, réduisant ainsi les erreurs humaines et garantissant une mise en production rapide et fiable.

**Analyse de la qualité du code avec SonarQube**

Pour analyser la qualité du code dans une démarche DevOps, j’ai utilisé SonarQube en l’intégrant dans mon pipeline CI/CD via Jenkins. Tout d'abord, j'ai installé SonarQube en tant que service Docker pour centraliser l’analyse. Ensuite, j’ai configuré un fichier sonar-project.properties dans le projet afin de définir les paramètres nécessaires, comme le nom du projet, la clé unique, le chemin des fichiers sources et les exclusions éventuelles. Dans le pipeline, j’ai ajouté une étape dédiée à l’analyse de code après le build et avant les tests. Pour cela, j’ai utilisé le scanner SonarQube, un outil en ligne de commande, pour envoyer les métriques du code source à l’instance SonarQube. Ce processus permet de détecter des problèmes comme des bugs, des vulnérabilités, des duplications de code et des mauvaises pratiques. Les règles d’analyse sont basées sur des standards prédéfinis pour le langage utilisé et peuvent être personnalisées selon les besoins du projet. Les résultats de l’analyse sont accessibles dans l’interface web de SonarQube, avec des graphiques et des rapports détaillés sur les problèmes détectés. Pour une démarche proactive, j’ai configuré des seuils de qualité : si des critères comme le taux de couverture des tests ou le niveau de dette technique ne sont pas atteints, le pipeline échoue, empêchant ainsi le déploiement.

En intégrant SonarQube dans le pipeline DevOps, j’ai pu automatiser la surveillance de la qualité du code, améliorer sa maintenabilité et réduire les risques d'introduire des erreurs en production

**Maintenance et évolutions futures**

***Planification et suivi du projet***

Pour planifier le déploiement de mon application en production, j'ai suivi une approche structurée comprenant plusieurs étapes essentielles. Tout d'abord, j'ai réalisé une évaluation des besoins techniques, en choisissant un serveur d'hébergement adapté avec des ressources suffisantes pour garantir des performances optimales et une capacité à gérer la montée en charge.

Ensuite, j'ai mis en place un pipeline de déploiement automatisé en utilisant Jenkins, ce qui a permis de standardiser et de simplifier le processus. J'ai configuré des scripts pour automatiser la création et la mise à jour des conteneurs Docker, la migration de la base de données avec Doctrine, et la gestion des configurations spécifiques à l'environnement de production.

Pour minimiser les risques, j'ai effectué une sauvegarde complète de la base de données et des fichiers du projet avant le déploiement. J'ai également intégré des tests automatisés dans le pipeline pour valider les fonctionnalités critiques et détecter les régressions. Ces tests incluaient des vérifications de conformité du code et des tests fonctionnels ciblant les cas d’utilisation principaux.

Après le déploiement, j'ai surveillé de près l'application à l'aide d'outils de monitoring pour détecter d'éventuelles anomalies et ajuster les configurations si nécessaire. Cette planification méthodique a permis d'assurer un déploiement fiable et sans interruption pour les utilisateurs finaux​.

***Documentation du déploiement***

Pour documenter les étapes nécessaires au déploiement de l'application, j'ai créé un guide détaillé et structuré. Ce document couvre toutes les phases, des prérequis à la mise en production, afin de garantir un déploiement fluide et reproductible.

J’ai commencé par lister les prérequis techniques, comme l’installation des outils nécessaires (Git, Composer, Apache, MariaDB, PHP) et la configuration de l’environnement serveur. J’ai également détaillé les configurations essentielles, notamment la création des fichiers .env, la gestion des clés API, et les paramètres de connexion à la base de données.

Ensuite, la documentation inclut les commandes pour installer les dépendances avec Composer et configurer l’application en mode production. J’ai décrit les étapes pour exécuter les migrations de la base de données avec Symfony (php bin/console doctrine:migrations:migrate) et pour initialiser les données si nécessaire.

Avant le déploiement en production, j’ai ajouté des instructions pour exécuter les tests automatisés avec PHPUnit et analyser les résultats SonarQube afin d’identifier d’éventuelles anomalies.

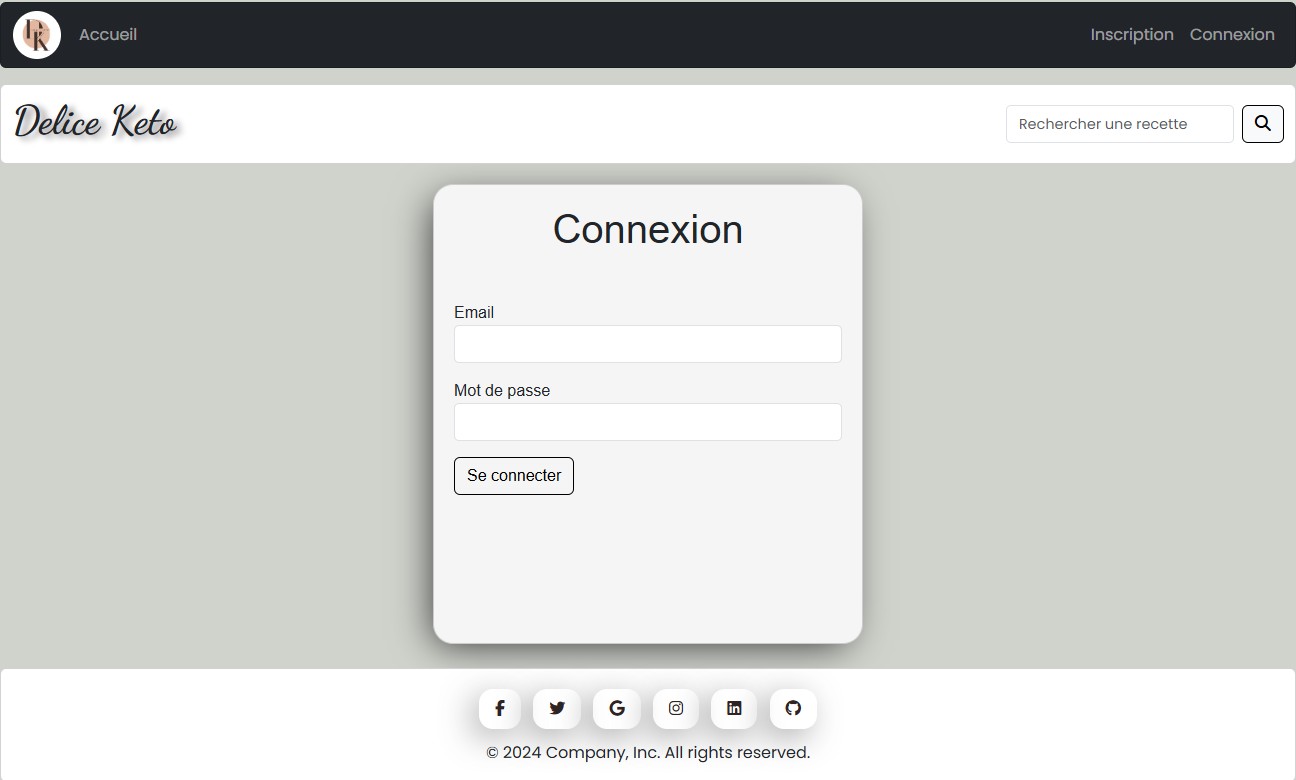
Enfin, la documentation explique comment récupérer la dernière version du code depuis GitHub, redémarrer le serveur Apache, et effectuer des vérifications post-déploiement pour s’assurer du bon fonctionnement de l’application.

# **13. Présentation des éléments techniques**

**Captures d’écran des interfaces**

Les captures d’écran suivantes permettent d’illustrer les différentes pages du site et leurs fonctionnalités :

* **Page d’accueil** : Présente un aperçu des recettes les plus populaires et permet une navigation rapide.
* **Page de connexion** : Interface utilisateur permettant aux membres de se connecter de manière sécurisée.

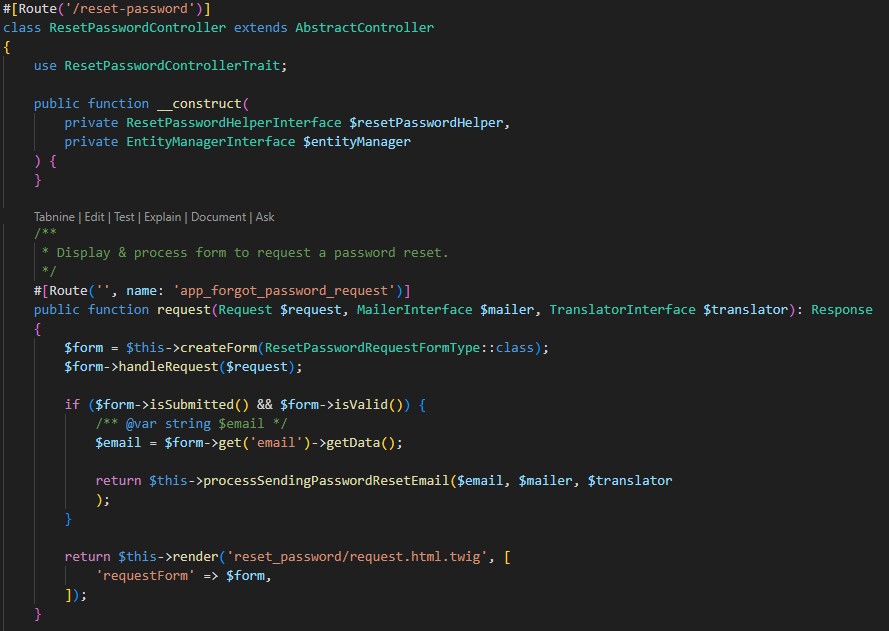


* **Espace utilisateur** : Affichage des recettes enregistrées, des interactions et de l’historique des activités.
* **Page de gestion des recettes** : Interface dédiée à l’ajout, la modification et la suppression des recettes personnelles.
* **Page de consultation des recettes** : Présentation détaillée d’une recette avec ses **ingrédients, étapes et commentaires des utilisateurs​.**

**Extraits de code**

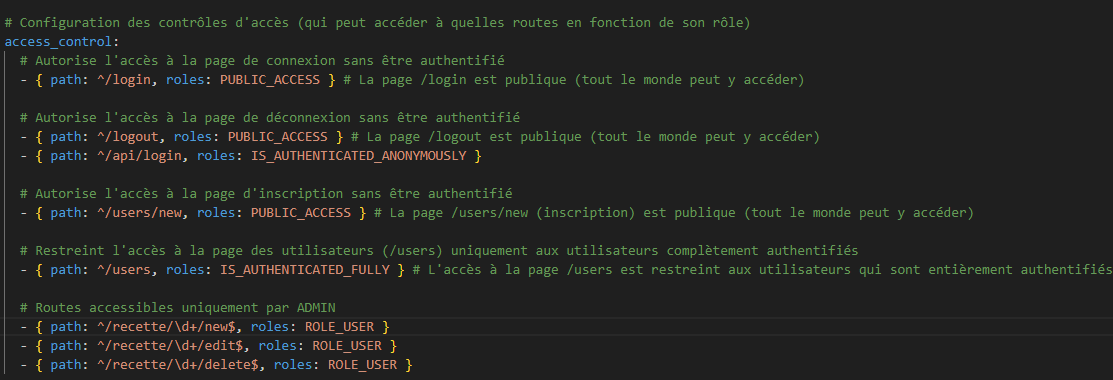
***Composants métier***

Pour intégrer et valider les formulaires utilisateurs dans mon application, j'ai utilisé les outils fournis par Symfony, qui simplifient considérablement la gestion des formulaires.  
J'ai commencé par définir des classes de formulaire dans le dossier src/Form, où chaque classe représente une structure spécifique, comme le formulaire de création de recette, d'inscription ou d'ajout de commentaire. Dans ces classes, j'ai configuré les champs nécessaires, en précisant leur type (TextType, TextareaType, etc.), leurs options (comme les contraintes de longueur ou de format), et leurs labels pour une meilleure lisibilité.  
Symfony offre un composant de validation puissant, que j'ai utilisé pour définir des contraintes telles que NotBlank, Length ou Email directement dans les entités ou dans les classes de formulaire.  
Pour l'intégration dans les vues, j'ai utilisé le moteur de templates Twig, qui rend la génération des formulaires intuitive grâce à des fonctions dédiées comme form\_start, form\_widget, et form\_end. Cela m’a permis d’afficher les formulaires avec un rendu uniforme tout en personnalisant leur apparence selon les besoins.  
La validation des données s'effectuait automatiquement côté serveur lors de la soumission des formulaires. En cas d'erreur, Symfony renvoyait des messages clairs associés aux champs concernés, facilitant la correction par l’utilisateur.  
Enfin, j'ai testé chaque formulaire en simulant des cas courants et des scénarios limites pour m'assurer de leur bon fonctionnement, garantissant ainsi une expérience utilisateur fluide et fiable​.



***Composants d’accès aux données***

***Composants de sécurité***



# **14. Conclusion**

Avec le projet "Délices Keto", j’ai pu mettre en pratique l’ensemble des compétences acquises durant ma formation, aussi bien sur le plan technique que méthodologique. La création de cette plateforme collaborative a demandé une réflexion approfondie sur l’architecture logicielle, la conception de la base de données, l’implémentation d’interfaces intuitives et la mise en place d’un environnement de développement fiable et sécurisé. Grâce à Symfony, Angular, Docker, Jenkins et SonarQube, j’ai bâti un écosystème complet favorisant la qualité du code, l’intégration continue et la modularité.

Ce projet m’a également permis de mesurer l’importance de la gestion de projet et de la documentation, que ce soit en définissant clairement les objectifs et besoins, en assurant un suivi rigoureux via Kanban, ou encore en détaillant chaque étape de conception et de déploiement. L’expérience m’a confortée dans l’idée qu’un projet solide repose sur un bon équilibre entre la technicité et la collaboration, tant avec les utilisateurs qu’avec les équipes impliquées.

Au-delà de l’aspect purement technique, "Délices Keto" répond à une communauté en pleine expansion, souhaitant partager des recettes et s’entraider autour du régime cétogène. Les fonctionnalités de partage, de commentaires et de gestion des contenus ouvrent la voie à une véritable plateforme d’échange. De futures évolutions sont déjà envisagées pour enrichir l’expérience utilisateur, notamment grâce à l’ajout d’un blog, d’un forum ou encore de fonctionnalités sociales plus poussées.

En somme, ce projet marque une étape clé dans mon parcours : il illustre non seulement la mise en œuvre de compétences variées, mais aussi la volonté de proposer un service à valeur ajoutée, évolutif et accessible à une large audience. Je reste convaincue qu’avec le temps et l’implication d’une communauté active, "Délices Keto" continuera de s’améliorer, tout en valorisant le partage et la créativité dans l’univers de la cuisine cétogène.