RAPPORT DE TRAVAIL D'ÉTUDE ET DE RECHERCHE

POUR LA FIN D'ÉTUDE EN

MASTER 1 INFORMATIQUE

PARCOURS

DONNÉES ET SYSTÈMES CONNECTÉS

SESSION

2019 – 2020

À

L'UNIVERSITÉ JEAN MONNET

SAINT-ÉTIENNE

PROJET QUICK RECIPES

ROMDAN ELIAS

Table des matières

Table des figures

1. Introduction

Le projet « Quick Recipes » est réalisé dans le cadre de l'unité d'enseignement travail d'étude et de recherche abrégé en TER. Cette UE de crédits 12, consiste à effectuer un projet personnel de fin de la première année du Master à l'Université de Jean Monnet, dans mon cas il s'agit du Master Informatique parcours Données et Systèmes Connectés, d'une durée de 3 à 5 mois.

Le projet est supervisé par, le responsable de mon Master, Monsieur François JACQUENET. La supervision se présente sous forme d'envoi hebdomadaire d'un compte rendu expliquant les tâches réalisées pendant la semaine, ainsi que les objectifs fixés pour la semaine suivante, accompagné de l'envoi d'un planning illustrant la progression globale du projet. Parfois, il s'agit de poser quelques questions par rapport au projet, ou le déroulement de la soutenance, les réponses étaient toujours claires.

Le projet que j'ai choisi est centré sur le thème de la cuisine, il s'agit de bâtir de zéro un site web qui regroupe les différentes informations possibles sur les recettes de cuisine, et de les proposer gratuitement aux utilisateurs sans même le besoin d'avoir un compte enregistré sur l'application. Après l'étude des solutions existant sur le marché, cette solution se différenciera par sa navigation simple et rapide, la quantité d'informations utiles délivrées à l'utilisateur, son intelligence artificielle ainsi que la présence d'un administrateur qui supervisera les recettes.

Le projet s'est déroulé du 20 avril 2020 au 28 août 2020. Bien que le planning dessiné en début du projet soit assez différent du celui de la fin, et que certaines tâches non prioritaires soient écartées du développement pour respecter la date du rendu, les piliers de l'application sont bien mis en place.

2. Description de l'application

2.1 Expression du besoin

Nous vivons dans une époque où la majorité des échanges d'informations se passent par le réseau Internet. L'accès à l'information est devenu possible à n'importe quel individu ayant une connexion Internet et inversement, cet individu peut créer lui-même cette information et la rendre accessible au grand public. En ayant plusieurs acteurs participant à la création de l'information, ce flux ne cessera d'augmenter avec le temps.

Plus particulièrement dans le domaine de la cuisine, il est devenu facile d'avoir accès aux recettes cuisinières et d'essayer de les appliquer chez soi, sans même la nécessité d'être un chef cuisiner ou d'avoir suivi des cours particuliers en cuisine. La plupart des solutions existant actuellement sur le marché se ressemblent toutes et sont focalisées plus sur l'aspect préparation que sur les autres points, de plus l'excès dans le nombre d'éléments affichés sur une seule page, peut nuire à la navigation d'un utilisateur.

Le besoin est toujours présent d'une solution simple et rapide à naviguer, qui centralise les différentes informations disponibles sur un plat cuisiner, que ce soit l'aspect préparation, historique ou sanitaire.

2.2 Objectif

L'objectif de ce projet est de développer une solution qui permettra de répondre aux besoins cités dans l'expression du besoin et de faire en sorte que la source d'informations de cette solution soit alimentée principalement par les spécialistes de la cuisine, ainsi que de l'ajout d'un système de recommandation géré par une IA.

2.3 Solution

La solution est une application au nom de « Quick Recipes » développée dans un contexte web.

3. Fonctionnalités de l'application

Certaines fonctionnalités proposées par l'application « Quick Recipes » seront similaires aux autres solutions, tandis que d'autres le seront plus originales et par la suite, elles renforceront l'existence de cette solution. Ces fonctionnalités seront composées en trois grandes parties :

- Des fonctionnalités accessibles à n'importe quel utilisateur.
- Des fonctionnalités accessibles uniquement aux utilisateurs inscrits.
- Des fonctionnalités accessibles uniquement à l'administrateur.

3.1 Fonctionnalités accessibles au grand public

Dans le but de rendre l'application simple et rapide à naviguer, la majorité des fonctionnalités seront accessibles sans même le besoin d'être inscrit sur l'application. Ci-dessous une liste des fonctionnalités accessibles au grand public :

- Création d'un compte sur l'application.
- Recherche d'une recette de cuisine.
- Visualisation d'une recette de cuisine.
- Téléchargement d'une recette de cuisine.
- Navigation dans les recettes de cuisine.

a) Création d'un compte sur l'application

Un utilisateur non connecté aura la possibilité de s'inscrire gratuitement sur l'application en respectant les conditions suivantes :

- Saisir un pseudo non existant sur l'application.
- Saisir un pseudo entre 3 et 12 caractères.
- Saisir un pseudo sans utiliser des caractères spéciaux.
- Saisir un mot de passe entre 6 et 12 caractères.
- Accepter les règles et les conditions d'utilisation.

b) Recherche d'une recette de cuisine

Un utilisateur non connecté aura la possibilité de chercher une recette de cuisine par différents mots-clés en saisissant dans la barre de recherche :

- Le nom de la recette de cuisine.
- Un ou plusieurs ingrédients de la recette de cuisine.

Ou en sélectionnant parmi une liste aléatoire sur la page d'accueil :

- La catégorie de la recette de cuisine.
- La région de la recette de cuisine.

c) Visualisation d'une recette de cuisine

Après avoir fait son choix parmi plusieurs recettes de cuisine, un utilisateur non connecté aura la possibilité de visualiser celle choisie, comme suit :

- Le nom de la recette de cuisine.
- Les images de la recette de cuisine.
- L'histoire de la recette de cuisine.
- L'auteur de la recette de cuisine.

- La région de la recette de cuisine.
- La date de découverte de la recette de cuisine.
- Les calories de la recette de cuisine.
- La ou les catégories de la recette de cuisine.
- La méthode de préparation de la recette de cuisine.
- Les ingrédients de la recette de cuisine.

d) Téléchargement d'une recette de cuisine

Après avoir visualisé la recette, un utilisateur non connecté aura la possibilité de télécharger cette recette sous format PDF, pour qu'elle soit toujours consultable indépendamment d'une connexion Internet.

e) Navigation dans les recettes de cuisine

Un utilisateur non connecté aura la possibilité de parcourir une liste de recettes de cuisine selon la catégorie ou la région, dans le cas où il n'a pas une recette de cuisine précise à chercher.

3.2 Fonctionnalités accessibles aux utilisateurs inscrits

Un utilisateur inscrit sur l'application aura accès aux mêmes fonctionnalités qu'un non inscrit, avec quelques fonctionnalités supplémentaires :

- Connexion à son compte sur l'application.
- Accès au système de recommandation par l'IA.
- Mettre en favori des recettes de cuisine.
- Déconnexion de l'application.

a) Connexion à un compte utilisateur

Un utilisateur non connecté aura la possibilité d'accéder à son compte sur l'application en saisissant dans l'espace de connexion :

- Le pseudo correct de son compte.
- Le mot de passe correct de son compte.

b) Accès au système de recommandation par l'IA

Un utilisateur connecté aura la possibilité de bénéficier d'une assistance par l'IA pendant sa navigation sur l'application. Le but de l'IA est de permettre à l'utilisateur de trouver rapidement des recettes de cuisine en se basant sur ses goûts. Les caractéristiques et les fonctionnalités de l'IA seront détaillées dans la section « 4.6 Intelligence artificielle ».

c) Mettre en favori des recettes de cuisine

Un utilisateur connecté aura la possibilité de mettre ou enlever des recettes de cuisine en favori, ce qui lui permettra de trouver facilement ses recettes préférées dans ses prochaines visites.

d) Déconnexion de l'application

Un utilisateur connecté aura la possibilité de se déconnecter de l'application à n'importe quel moment.

3.3 Fonctionnalités accessibles à l'administrateur

Pour garantir le bon fonctionnement de l'application, certaines fonctionnalités seront accessibles uniquement à l'administrateur :

- Ajout des recettes de cuisine.
- Modification des recettes de cuisine.
- Suppression des utilisateurs.

a) Ajout des recettes de cuisine

Un administrateur aura la possibilité d'ajouter des nouvelles recettes de cuisine sur l'application en passant des fichiers textes contenant les informations des recettes de cuisine selon une structure spécifique.

b) Modification des recettes de cuisine

Un administrateur aura la possibilité de modifier ou supprimer les recettes de cuisine existantes dans l'application à travers une interface dédiée.

c) Suppression des utilisateurs

Un administrateur aura la possibilité de supprimer des utilisateurs de l'application à travers une interface dédiée.

4. Architecture du projet

4.1 Langages de programmation

Le choix des langages de programmation pour le développement du logiciel « Quick Recipes » était fait en fonction des plateformes où l'application sera disponible, de mes acquis pendant le cursus de Master 1 et des avantages que les technologies actuelles apporteront à l'application :

- Le côté client sera codé avec le Framework Angular qui sera composé du langage HTML et TypeScript. Le design sera effectué en langage CSS. Le langage JavaScript sera utilisé pour gérer quelques comportements.
- Le côté serveur sera codé en Java EE avec le Framework Spring. L'interface de programmation JPA sera utilisée pour les requêtes. Dans le cas où ces requêtes seront complexes, le langage SQL sera utilisé à la place.

- La base de données utilisée sera MySQL pour ses performances mais aussi son accès libre.
- Le style d'architecture logicielle REST sera utilisé pour établir la connexion entre le côté client et serveur à travers des API.

4.2 Environnement de développement

L'environnement utilisé dans le développement du logiciel « Quick Recipes » sera un ensemble d'outils, qui seront choisis en fonction des besoins de l'application et de mon expérience à les manipuler :

- L'OS utilisé sera Windows pour sa simplicité et le fait que les outils qui seront utilisés pendant le processus de développement sont compatibles avec ce dernier.
- L'IDE utilisé sera Visual Studio Code sous la licence MIT. Simple et rapide dans l'installation, utilisation, initialisation des projets ou encore l'ajout des dépendances.
- La gestion de versions sera effectuée avec Git sous la licence GNU GPL 2.0 ainsi que la plateforme GitHub pour héberger et gérer le code source. On gardera un historique de l'évolution du code pendant tout le projet, ce qui sera utile en cas d'un bug critique imprévisible. Pour rajouter plus de sécurité lors de codage des nouveaux modules, le système des branches sera utilisé pour séparer la version stable de celle en cours de développement.
- Le planning du projet sera effectué en diagramme de Gantt avec l'outil Gantt Project sous la licence GNU GPL. On suivra l'avancement des diverses tâches en fonction du temps, ce qui permettra d'avoir une vision sur l'évolution du projet.
- La conception des IHM sera effectuée avec le logiciel de prototypage en open source Pencil Project.
- La conception des diagrammes en langage UML sera effectuée avec le logiciel de modélisation Star UML sous la licence modifiée GNU GPL.
- Le diagramme de classes sera généré avec l'outil ObjectAid disponible dans le Marketplace de l'IDE Eclipse en open source pour sa partie Class Diagram.
- Le modèle de la base de données sera généré avec le logiciel de gestion de base de données MySQL Workbench sous la licence GPL.
- Les tests unitaires côté serveur seront effectués avec le Framework JUnit qui est adapté au développement en Java.
- Les tests unitaires côté client seront effectués avec Jasmine et Karma qui sont adaptés au développement en Framework Angular.
- Le logiciel en open source Sonar Qube sera utilisé pour la mesure de la qualité et sécurité du code.

- Le logiciel Postman sera utilisé pour tester le bon fonctionnement des API.

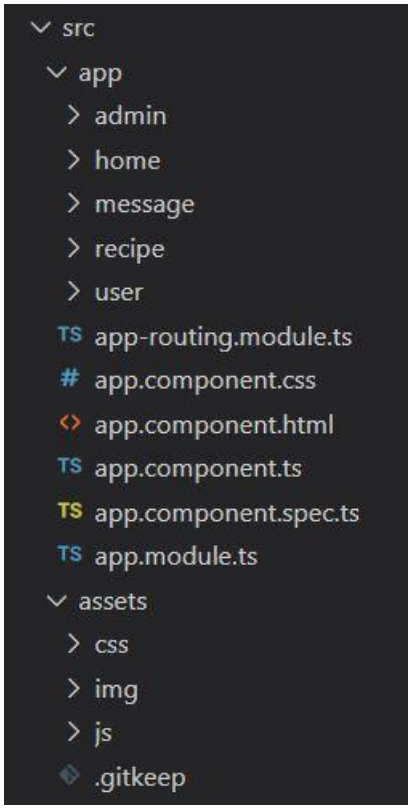
4.3 Normes et standards

Pendant le développement du logiciel « Quick Recipes », les règles ci-dessous doivent être respectées pour éviter toute anomalie pendant la compréhension et l'évolution du code :

- Utilisation du Camel Case lors du nommage des variables et des méthodes.
- Utilisation de la JavaDoc pour la documentation du code source Java.
- Utilisation des conventions de codage Java.
- Indentation du code cohérente en utilisant des espaces.
- Écriture du code et des commentaires en Anglais.
- Écriture des messages de commit en Anglais.
- Écriture des documents d'explication du projet en langage Markdown.
- Écriture des fichiers Markdown en Anglais.
- Utilisation de gitignore pour empêcher le commit des fichiers générés.
- Les ressources utilisées doivent être en open source.

4.4 Arborescence du projet

a) Côté client



Après avoir installé Angular sous sa dernière version avec la commande :

$$npm\ install\ -g\ @angular/cli$$

Le projet côté client est initialisé avec la commande :

$$ng\ new\ front$$

Les fichiers qui constituent les pages web seront créés dans le dossier :

$$front/src/app$$

Les icônes, scripts JS ou CSS de la page index, seront créés dans le dossier :

$$front/src/assets$$

Chaque component créé sera accompagné d'un service qui permet de gérer ses appels API avec le serveur.

$$front/src/app/.../service$$

- Le dossier $\boldsymbol{src/app/admin}$ contient :

$\boldsymbol{Component\ menu}$: Menu de navigation vers l'ajout ou la suppression des recettes.

$\boldsymbol{Component\ add}$: Ajout des recettes sur le serveur par l'administrateur.

$\boldsymbol{Component\ remove}$: Suppression des recettes du serveur par l'administrateur.

- Le dossier $\boldsymbol{src/app/home}$ contient :

$\boldsymbol{Component\ home}$: Gestion de la page d'accueil.

- Le dossier $\boldsymbol{src/app/message}$ contient :

$\boldsymbol{Component\ message}$: Affichage des notifications sur les pages de l'application.

- Le dossier $\boldsymbol{src/app/recipe}$ contient :

$\boldsymbol{Component\ display}$: Visualisation des informations d'une recette.

$\boldsymbol{Component\ list}$: Affichage des recettes liées à la recherche de l'utilisateur.

$\boldsymbol{Model\ recipe}$: Définition de l'objet recette.

- Le dossier $\boldsymbol{src/app/user}$ contient :

$\boldsymbol{Component\ favorite}$: Gestion des favoris d'un utilisateur.

$\boldsymbol{Component\ sign}$: Gestion de la connexion et inscription à l'application.

$\boldsymbol{Model\ user}$: Définition de l'objet utilisateur.

- Le restant des fichiers se trouvant dans le dossier $\boldsymbol{src/app}$:

$\boldsymbol{Component\ app}$: Le component par défaut de l'application qui sera remplacé par les components ci-dessus lors de la navigation sur l'application.

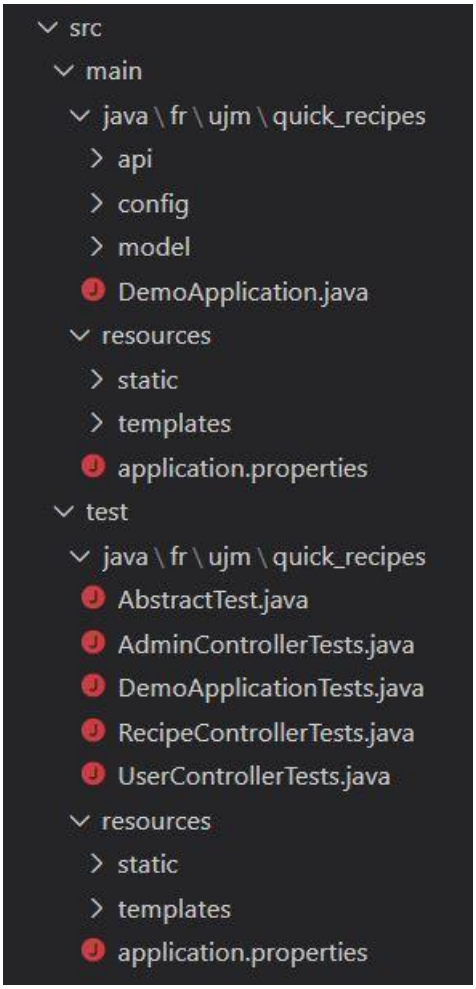
$\boldsymbol{app.module.ts}$: Déclaration et importation des components, modules et services qui constituent l'application.

$\boldsymbol{app-routing.model.ts}$: Attribution d'un chemin et d'une autorisation d'accès vers chaque component.

b) Côté serveur

L'initialisation du projet côté serveur est réalisé à travers un IDE :

- Lancement de l'IDE Visual Studio Code.
- Utilisation de l'extension Spring Initializr pour générer un projet Maven.
- Configuration des différents paramètres du projet (groupe, artéfact, ...).
- Sélection des dépendances adéquates.
- Le fichier $\boldsymbol{pom.xml}$ contient toutes les dépendances au code Java.



L'organisation des fichiers côté serveur est comme suit :

src/main/java/.../api : Contient les contrôleurs en architecture REST qui permettent de jouer l'intermédiaire entre les requêtes du client et les ressources du serveur.

src/main/java/.../config : Contient la gestion du token, le cryptage des mots de passe, les autorisations et l'authentification.

src/main/java/.../model : Contient des entités reliées aux tables de la base de données grâce au Framework d'ORM Hibernate.

src/main/ressources : Contient les images des recettes stockées sur le serveur, la configuration du serveur et sa relation avec la base de données.

src/test/java/.../quick_recipes : Contient les fichiers des tests codés sous JUnit.

src/test/ressources : Contient la configuration du serveur spécifique aux tests.

4.5 Schéma de la base de données

Table user : Regroupe toutes les informations sur les utilisateurs enregistrés sur l'application.

Table user_favorites : Regroupe la liste des recettes mises en favori par les utilisateurs. Cette table a été créée à l'aide de l'attribut JPA ***@ElementCollection*** et donc n'a pas de classe entité spécifique à elle.

Table recipe : Regroupe toutes les informations sur les recettes de cuisine.

Table category : Regroupe les catégories d'une recette (exemple sucré, salé, ...). Une recette peut avoir une ou plusieurs catégories.

Table picture : Regroupe les images d'une recette sous différents angles. Une recette peut avoir zéro, une ou plusieurs images.

Table ingredient : Regroupe les ingrédients utilisés dans la création d'une recette. Une recette peut avoir un ou plusieurs ingrédients.

Table preparation : Regroupe les méthodes de préparation d'une recette. Une recette peut avoir une ou plusieurs méthodes de préparation.

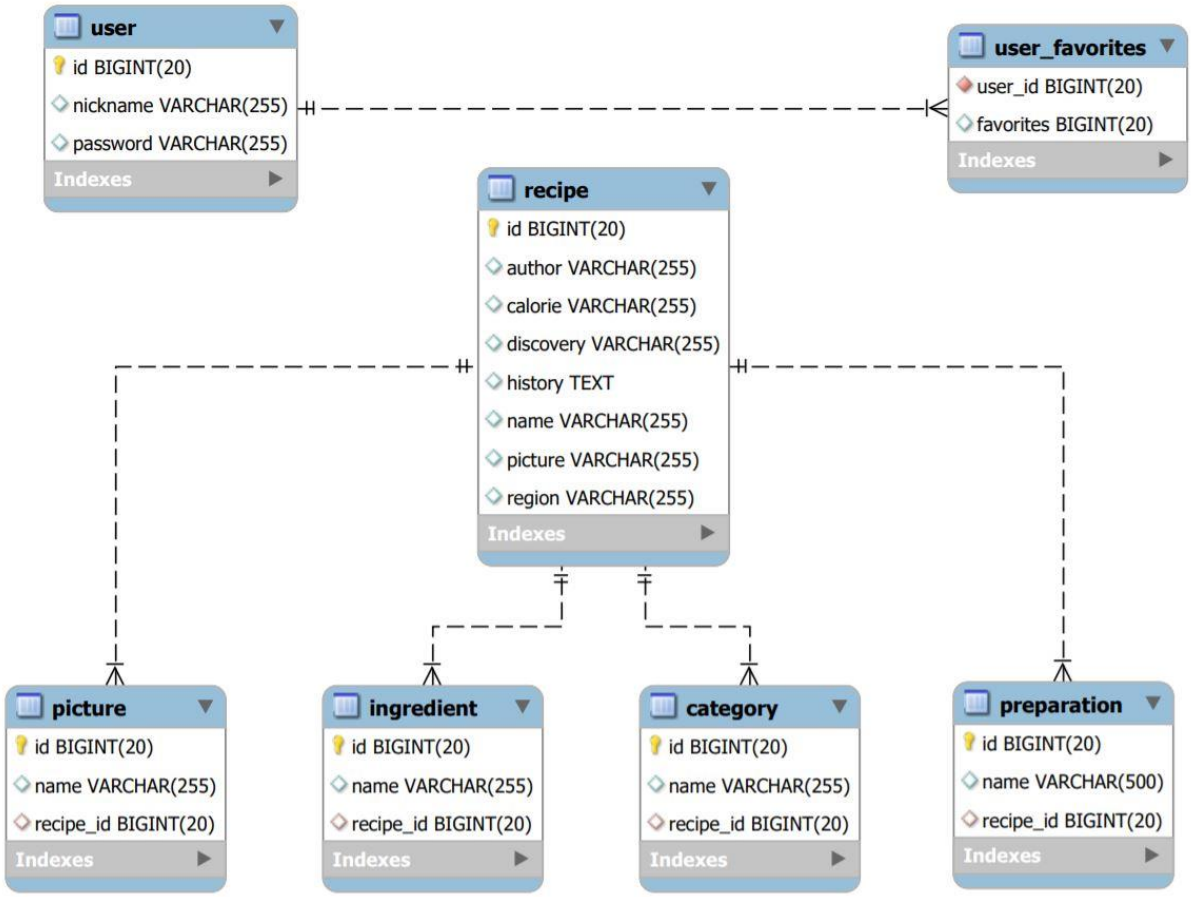


Figure 1 : Modèle de la base de données

4.6 Intelligence artificielle

a) Objectif de l'IA

Le rôle de l'IA intégré au projet est de fournir à un utilisateur connecté, une liste de recettes sélectionnées suivant ses préférences de façon fluide sans perturber sa navigation. Ce qui permet à un utilisateur de prendre connaissance des recettes dont il ignore probablement leurs existences et qui sont susceptibles de l'intéresser.

b) Utilisation de l'IA

Figure 2 : Zone d'opération de l'IA sur la page d'accueil

Dans une base de données qui contient des milliers d'entrées pour la région et les catégories des recettes, il sera impossible de tous les proposer pour l'utilisateur sur la page d'accueil. On distinguera deux méthodologies lors de la proposition :

- Pour un utilisateur non connecté, l'idée est de proposer 25 régions et catégories tirées aléatoirement depuis la base de données.
- Pour un utilisateur connecté, l'IA interviendra pour proposer 25 régions et catégories qui sont similaires aux préférences de l'utilisateur.

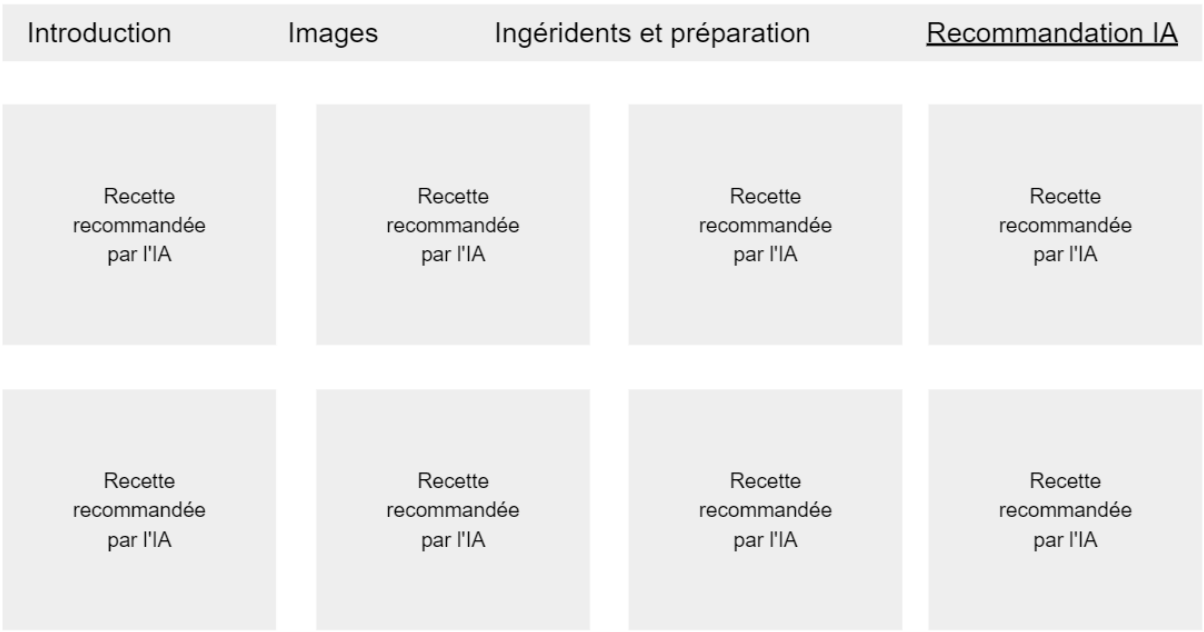


Figure 3 : Zone d'opération de l'IA sur la page de recette

La démarche est la même que le paragraphe ci-dessus :

- Pour un utilisateur non connecté, l'idée est de proposer 10 recettes tirées aléatoirement depuis la base de données.
- Pour un utilisateur connecté, l'IA interviendra pour proposer 10 recettes qui sont similaires aux préférences de l'utilisateur.

c) Etude de l'IA

Avant d'aborder la partie codage, il est important d'étudier les performances des différents algorithmes utilisés lors de l'élaboration des systèmes de recommandation par IA et de répondre à certaines questions :

- De quoi l'IA aura besoin pour fournir des résultats pertinents ?
- Quel est l'algorithme le plus adapté à notre application ?
- Quel est le langage le plus adapté pour le codage de l'IA ?
- Quelles sont les librairies disponibles en open source à utiliser ?
- Comment fonctionnera le module IA ?

Une IA de recommandation passe généralement par trois étapes afin de fournir des résultats exploitables :

- Collection des données.
- Construction d'un modèle utilisateur.
- Extraction d'une liste de recommandations.

Collection des données

La collection des données, comme son nom indique, consiste à collecter des données précises dans le but de classifier les utilisateurs selon au moins un critère. On peut traiter cet aspect de deux manières :

- Explicite : On demande à l'utilisateur d'attribuer une note (par exemple de 0 à 10) sur une recette de cuisine ou la possibilité de la mettre en favori. Cette méthode a tendance à être précise vu que l'utilisateur à une connaissance de ce qu'il réalise, en même temps s'il s'obstine à donner une note ou à utiliser le système des favoris, l'IA ne pourra jamais deviner ses préférences.
- Implicite : On suit la navigation d'un utilisateur de façon passive sans qu'il le remarque. On est sûr d'avoir des résultats à la fin au contraire de la méthode explicite, mais ces résultats peuvent ne pas être précis. Sur l'application, on comparera la fréquence de consultation des différentes recettes par un utilisateur, puis on lui suggérera des recettes qui se rapprochent de celles fréquemment consultées.

L'application « Quick Recipes » contient déjà un système qui permet à l'utilisateur de mettre des recettes en favori. Par conséquent, on se basera sur une récupération explicite des données. La source des données qui sera utilisée pour l'alimentation de l'IA, sera la liste des favoris de chaque utilisateur.

Construction d'un modèle utilisateur

Le modèle utilisateur est représenté sous forme d'une matrice qui met en relation des personnes à des objets. Plus particulièrement, notre modèle se basera sur les recettes se trouvant dans les favoris de chaque utilisateur, la matrice issue sera de cette forme :

	Utilisateur 1	Utilisateur 2	Utilisateur 3
Recette 1	1	0	0
Recette 2	0	1	0
Recette 3	0	0	1
Recette 4	1	1	1
Recette 5	0	0	0

Figure 4 : Exemple d'un modèle utilisateur

- Une matrice de taille $(\boldsymbol{n} * \boldsymbol{m})$ avec :
  - $\boldsymbol{n}$: le nombre des recettes présentes dans la base de données.
  - $\boldsymbol{m}$: le nombre des utilisateurs présents dans la base de données.

- 0 indique que la recette X ne figure pas dans les favoris de l'utilisateur Y.
- 1 indique que la recette X figure dans les favoris de l'utilisateur Y.

Extraction d'une liste de recommandations

La liste des recommandations est extraite à partir de la matrice construite grâce au modèle utilisateur via un algorithme. Bien que chaque algorithme utilise une approche différente, généralement tous les algorithmes attribuent une valeur de ressemblance à deux objets du modèle utilisateur. Plus la valeur est forte, plus la similarité est importante. En revanche, plus la valeur est faible, plus la similarité est inexistante. On distingue quartes approches à choisir lors de l'extraction d'une liste de recommandations :

- Recommandation personnalisée.
- Recommandation objet.
- Recommandation sociale.
- Recommandation hybride.

Recommandation personnalisée

Les algorithmes de ce type de recommandation se basent essentiellement sur la navigation d'un utilisateur, son historique de recherche, les articles consultés ou achetés, et les décisions prises pendant son utilisation de l'application.

L'avantage de cette approche est que tous les facteurs utilisés pour bâtir la liste de recommandations sont spécifiques à l'utilisateur et par conséquent, aucun choix externe, comme par exemple la navigation des autres utilisateurs, pourra influer la liste extraite par ces algorithmes.

Recommandation objet

Les algorithmes de ce type de recommandation focalisent sur les objets, au lieu des personnes, en construisant un profil qui caractérise chaque objet. Ce profil peut être un groupe d'attributs qui sera utilisé pour distinguer un objet d'un autre. Ensuite, ces algorithmes utiliseront le principe de corrélation d'un objet précis avec les objets précédemment consultés par l'utilisateur.

La corrélation est utilisée en probabilités et statistique pour trouver une liaison entre les variables d'un système. En recommandation objet les variables seront les profils attribués à chaque objet, et plus précisément dans notre application, les variables seront les recettes de cuisine avec deux attributs : la région et la catégorie.

Si par exemple, la liste des favoris d'un utilisateur contient plusieurs recettes indiennes, le coefficient de corrélation sera plus important et par conséquent, le système aura tendance à recommander des recettes indiennes. De la même façon pour la catégorie, si un utilisateur à plusieurs recettes sucrées dans ses favoris, le système proposera des recettes sucrées qui ne figurent pas dans les favoris de cet utilisateur.

On pourra même utiliser ce système pour les noms des recettes, on divisera tous les mots qui constituent le nom d'une recette, puis on assignera à chaque mot un poids qui déterminera son importance, enfin on cherchera la similarité des mots au poids élevé avec les autres recettes.

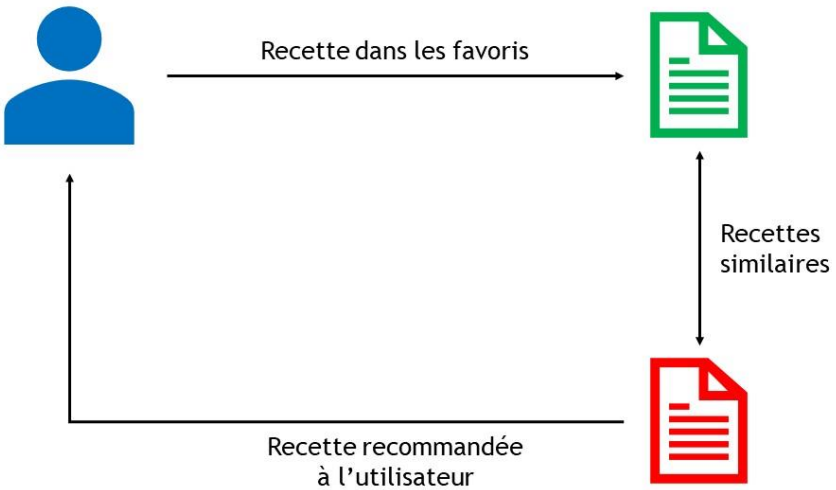


Figure 5 : Approche objet du problème (content-based filtering)

Recommandation sociale

Les algorithmes de cette approche se ressemblent à ceux de la recommandation objet, à l'exception que dans la recommandation sociale, on cherche une corrélation entre plusieurs utilisateurs ou plusieurs contenus, partant du principe que si un utilisateur X aime l'article A, B et C, et qu'un utilisateur Y aime l'article A et B. Alors, l'article C est susceptible d'intéresser l'utilisateur Y, donc on lui propose l'article C. De plus, les algorithmes de ce type n'auront besoin d'aucune information concernant la propriété des objets, ce qui nous évite d'utiliser le système des profils.

La recherche d'une corrélation entre les utilisateurs est appelée $\boldsymbol{user-centric}$, à partir d'un utilisateur, on essaie de créer un voisinage composé d'autres utilisateurs qui partagent les mêmes goûts que l'utilisateur central.

La recherche d'une corrélation entre les objets est appelée $\boldsymbol{item-centric}$, à l'inverse de la méthode précédente, on cherche une similarité entre les objets, qui sont dans notre cas, les recettes de cuisine. Généralement, la similarité s'appuiera sur le système de notation, les contenus qui sont les mieux notés auront un fort indice de similarité et inversement.

Les inconvénients posés par une telle approche résident sur la quantité des données fournies. Un tel système aura besoin de millions d'utilisateurs et d'articles pour opérer correctement, ce qui n'est pas évident lors d'un premier lancement de l'application ou si le public ciblé n'est pas assez grand. De la même manière, faire

des opérations consécutives sur une vaste quantité de données aura besoin d'une grande puissance de calcul. Pour toutes ces raisons, cette approche sera à éviter lors de la conception de l'IA de notre application.

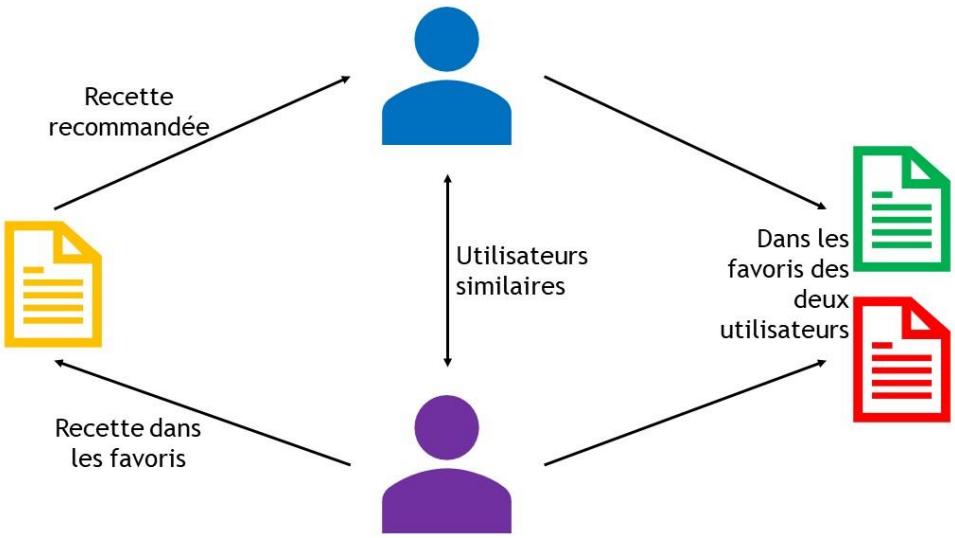


Figure 6 : Approche sociale du problème (collaborative filtering)

Recommandation hybride

La recommandation hybride combine les trois approches : personnalisée, objet et sociale. L'idée est de compenser les défauts qui se trouvent sur chacune des approches précédentes, tout en profitant des avantages proposés.

Cette approche est actuellement utilisée par les grandes entreprises comme Amazon, Netflix et Google. Le système de recommandation est devenu un enjeu majeur sur le marché, étant donné qu'il participe à une partie importante du chiffre d'affaires des entreprises commerciales. (Voir bibliographie [1])

d) Codage de l'IA

Choix de l'algorithme

La majorité des fonctionnalités proposées sur l'application sont accessibles au grand public sans le besoin d'avoir un compte. L'échantillon d'utilisateurs qu'on aura ne sera pas assez important pour faire fonctionner correctement la recommandation sociale ou hybride. Tandis que la recommandation personnalisée est concentrée sur la navigation de l'utilisateur, les données récupérées seront moins précises dans certaines situations. Par élimination, la recommandation objet sera la plus adaptée à utiliser dans notre application.

Choix du langage

Le langage Python sera utilisé pour le codage de l'IA, vu l'existence des librairies qui permettent de coder un système de recommandation.

Choix des librairies

Parmi les librairies disponibles et qui permettent d'implémenter la recommandation objet, il existe la libraire RAKE, abréviation de Rapid Automatic Keyword Extraction. L'idée derrière l'utilisation de cette librairie, est d'extraire le nom, les catégories et la région d'une recette qui se trouve dans les favoris d'un utilisateur, puis de chercher une similarité avec les autres recettes qui se trouvent dans la base de données, et enfin de proposer une liste de recommandations sans inclure la recette qui se trouve déjà en favori. On aura besoin d'utiliser en plus : La librairie Numpy pour manipuler les matrices construites par le modèle utilisateur. La librairie Pandas pour analyser les données qui représentent toutes les recettes enregistrées sur l'application sous format csv. La librairie Scikit-Learn pour gérer l'apprentissage automatique.

Implémentation du module

- Importation des librairies choisies. (Voir bibliographie [2])
- Extraction d'uniquement l'attribut nom, catégories et région du fichier csv.
- Utilisation de la librairie RAKE pour supprimer les majuscules, les espaces et la ponctuation des attributs extraits.
- Les attributs extraits seront combinés pour construire un sac de mots.
- Transformation du sac de mots en une matrice avec $\boldsymbol{fit\_transform}$.
- Comparaison de la similarité des recettes avec $\boldsymbol{cosine\_similarity}$.
- Création d'une fonction qui prend en paramètre les recettes du favori d'un utilisateur, et retourne les dix recettes les plus similaires à ces recettes.

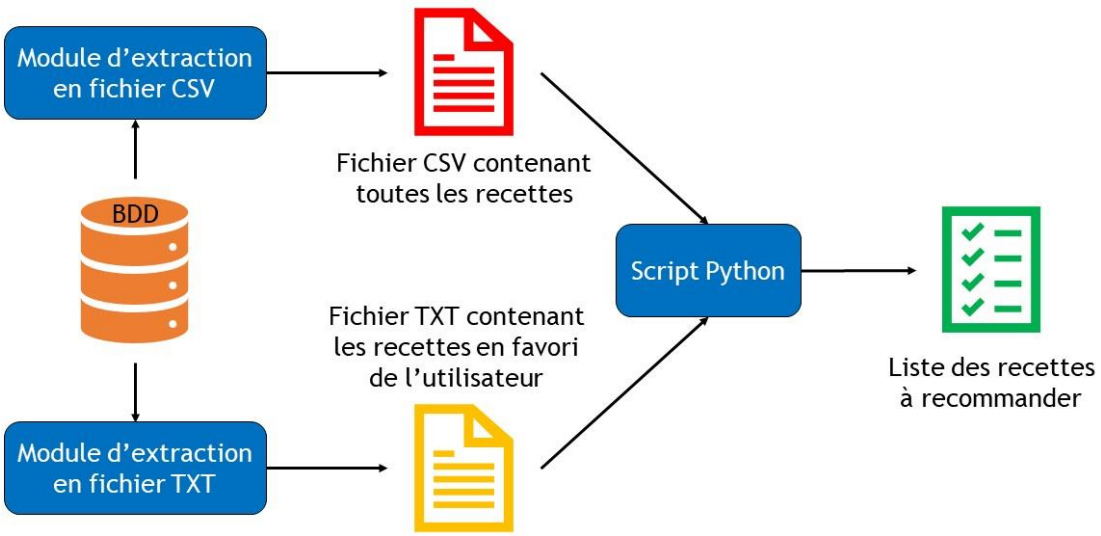


Figure 7 : Fonctionnement du module IA

5. Modélisation

5.1 Diagrammes de cas d'utilisation

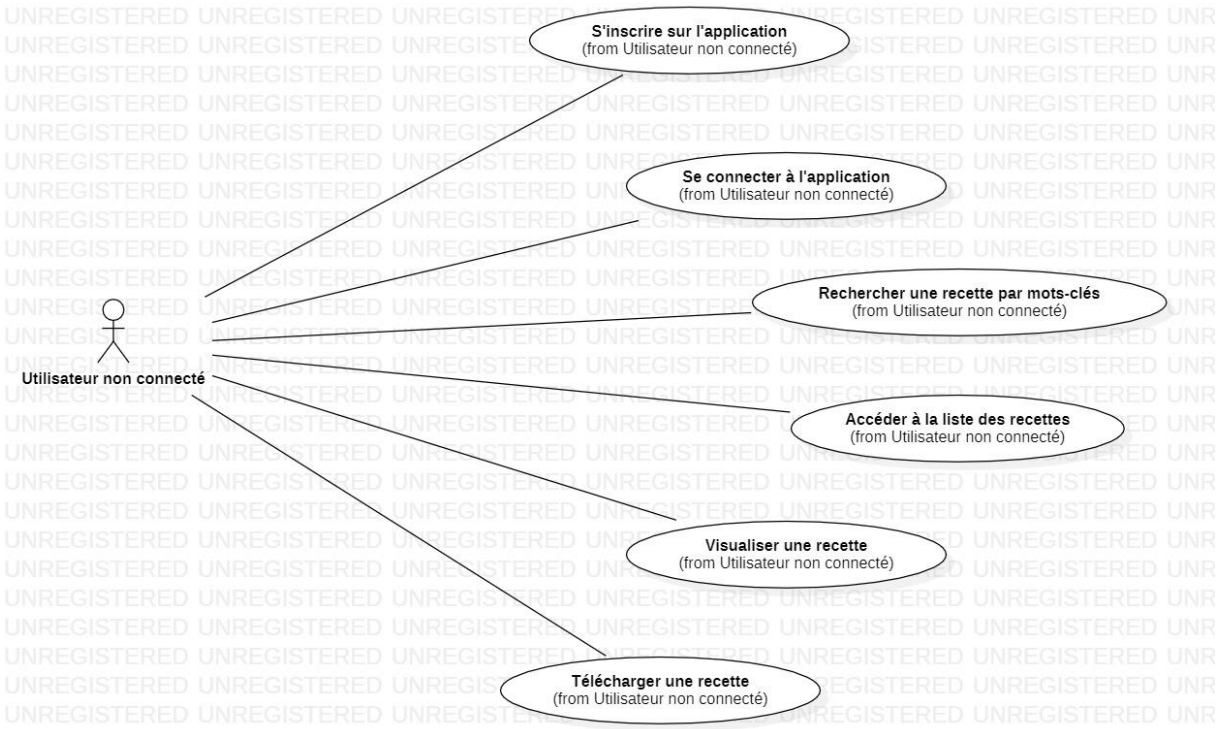


Figure 8 : Cas d'utilisation pour un utilisateur non connecté

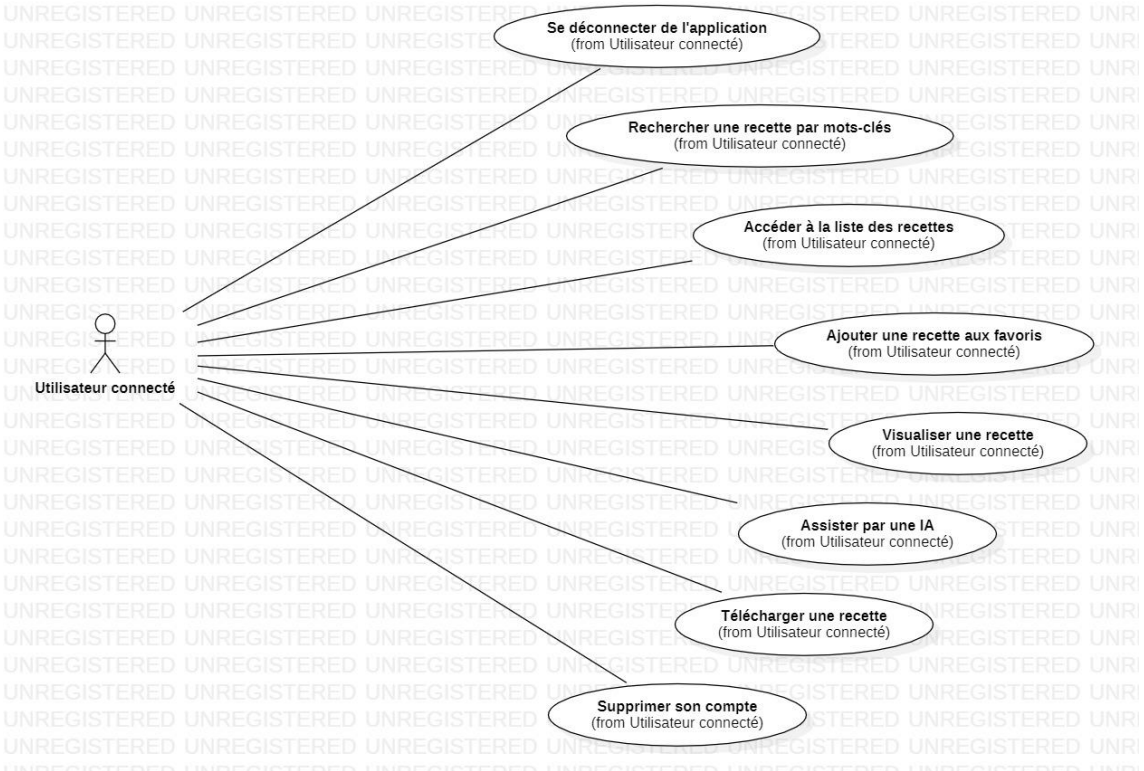


Figure 9 : Cas d'utilisation pour un utilisateur connecté

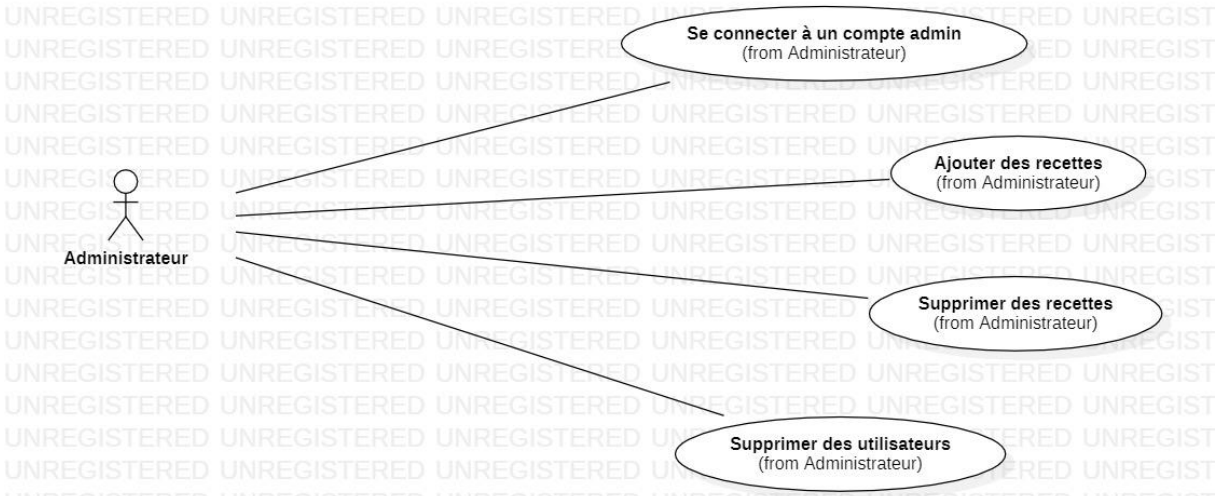


Figure 10 : Cas d'utilisation pour un administrateur

5.2 Diagrammes d'activité

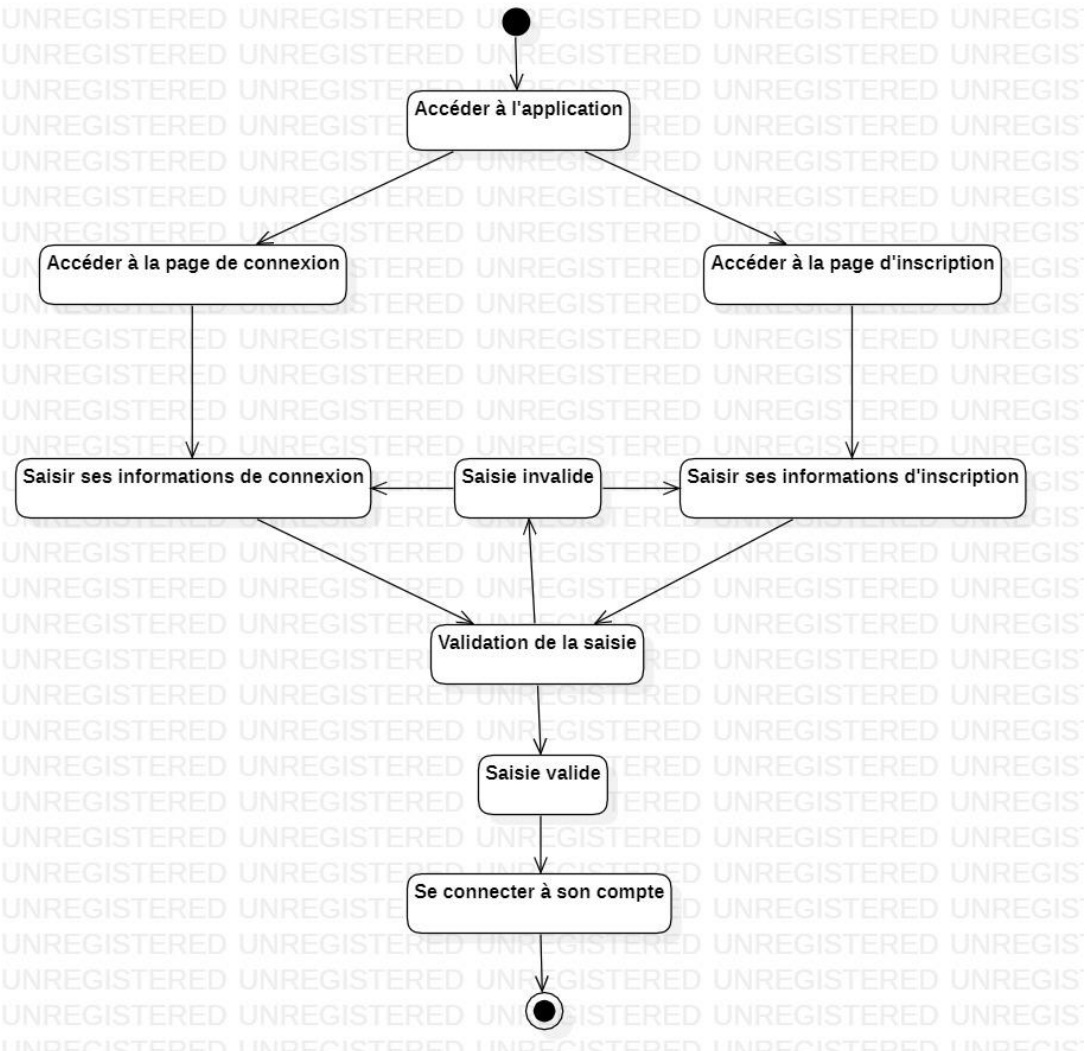


Figure 11 : Activité de connexion à l'application

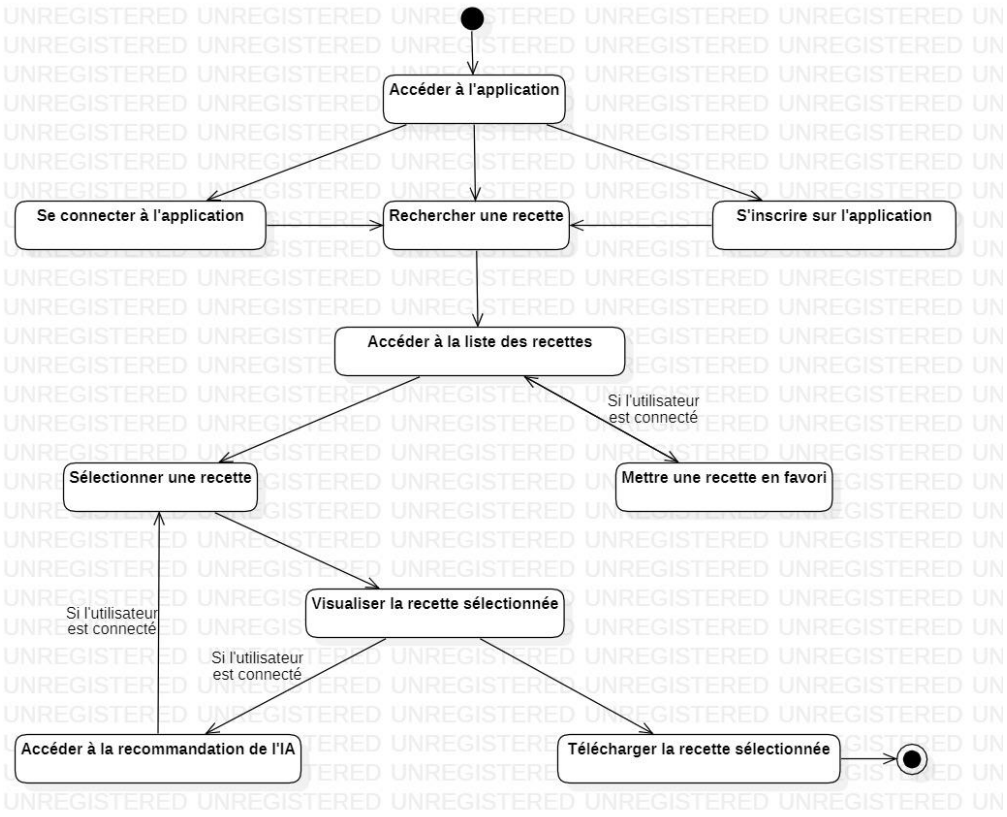


Figure 12 : Activité de recherche de recette

5.3 Diagrammes de séquence

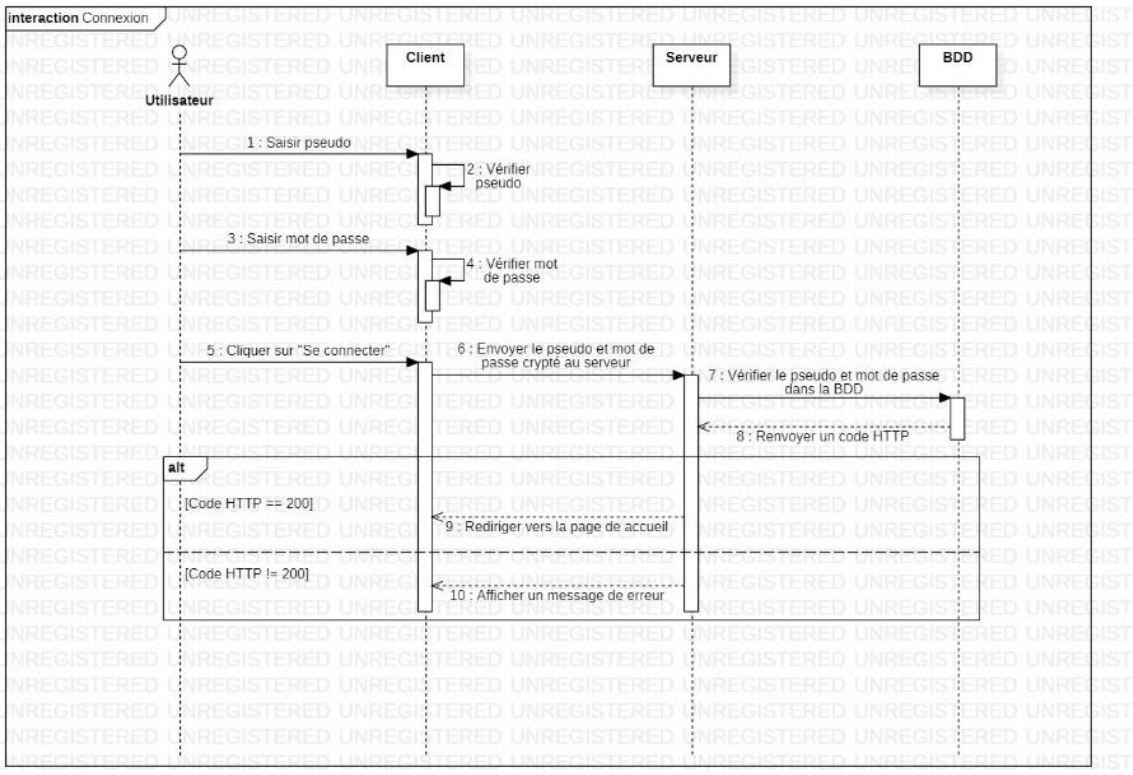


Figure 13 : Séquence de connexion

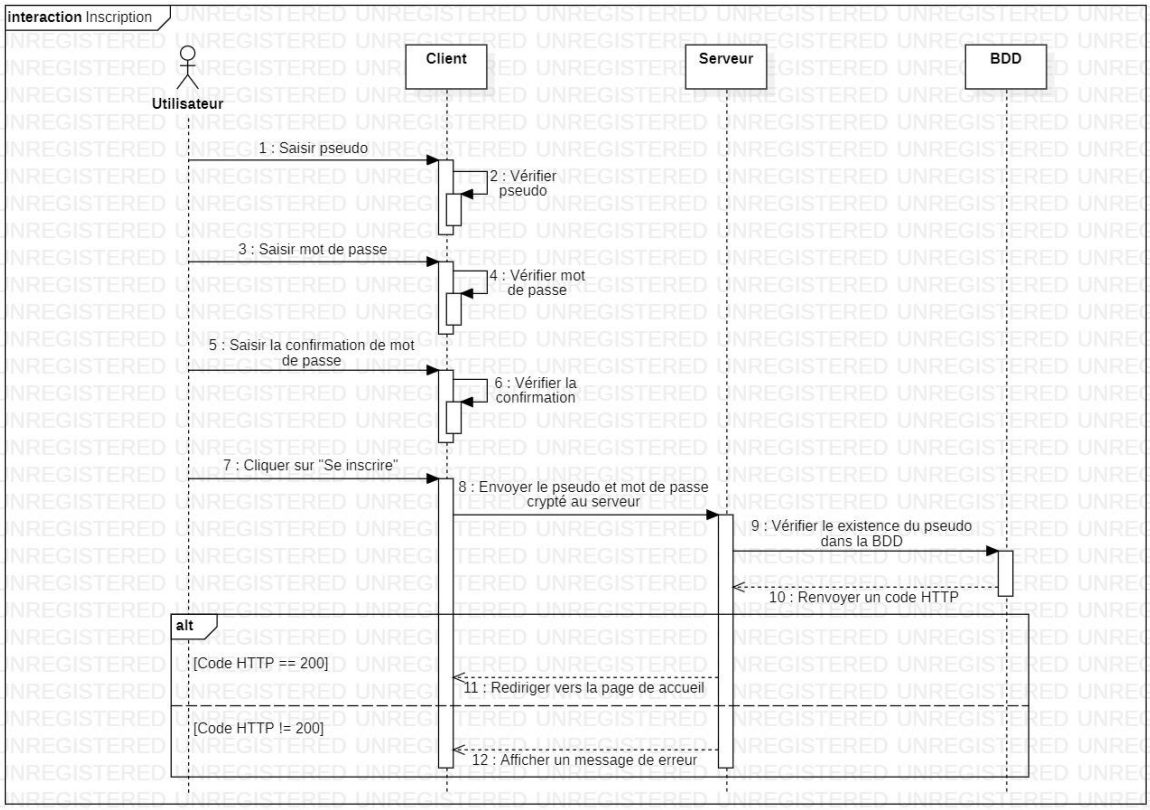


Figure 14 : Séquence d'inscription

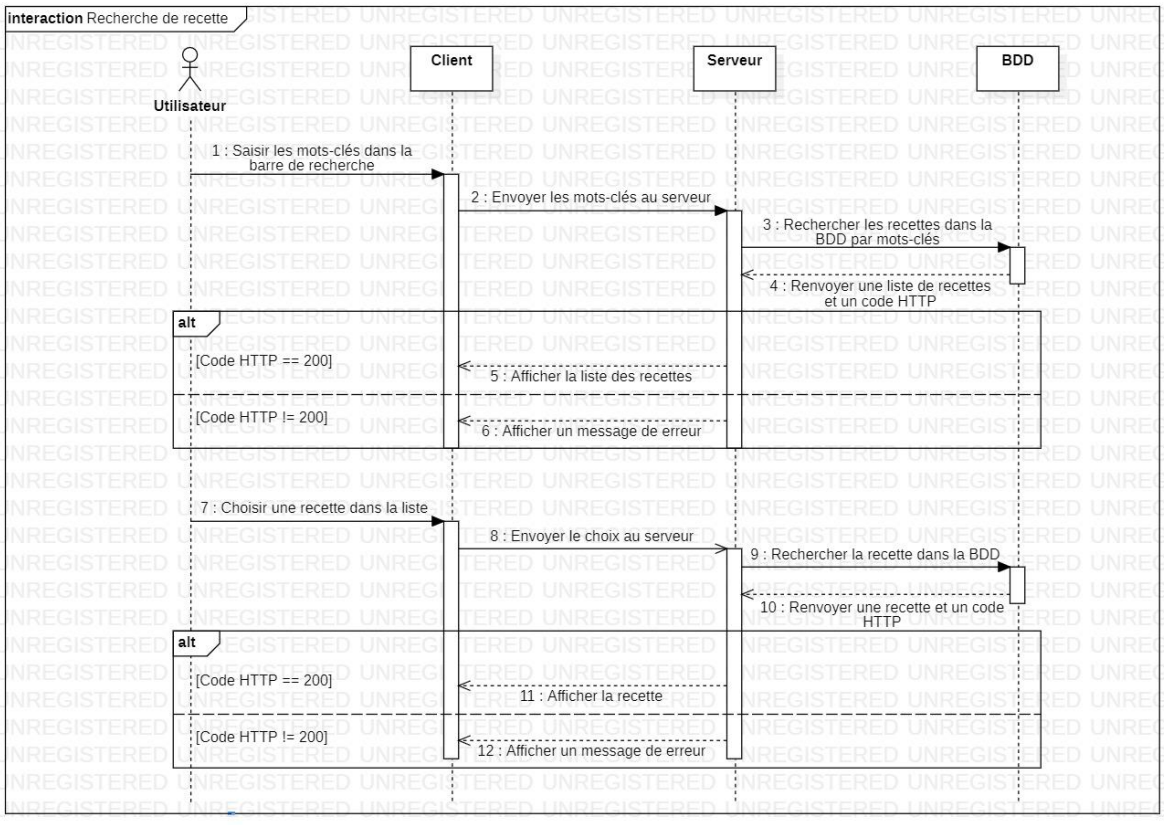


Figure 15 : Séquence de recherche de recette

5.4 Diagrammes de classes

Figure 16 : Classes du package « api »

UserController : Gestion des requêtes en relation avec les utilisateurs.

RecipeController : Gestion des requêtes en relation avec les recettes.

AdminController : Gestion des requêtes en relation avec l'administrateur.

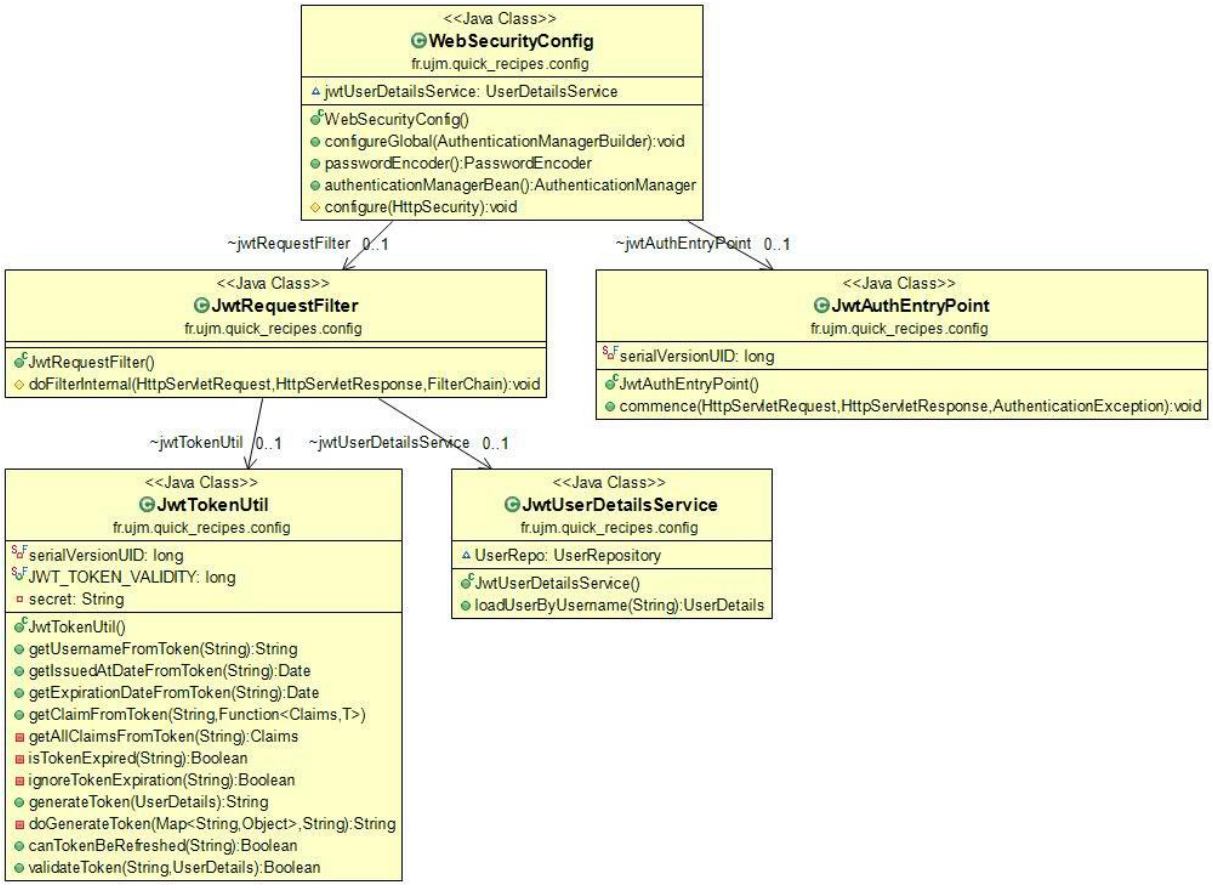


Figure 17 : Classes du package « config »

WebSecurityConfig : Attribution d'une autorisation aux API.

JwtRequestFilter : Vérification du token lors des appels API.

JwtAuthEntryPoint : Gestion des accès non autorisés.

JwtTokenUtil : Génération du token.

JwtUserDetailsService : Gestion de l'authentification.

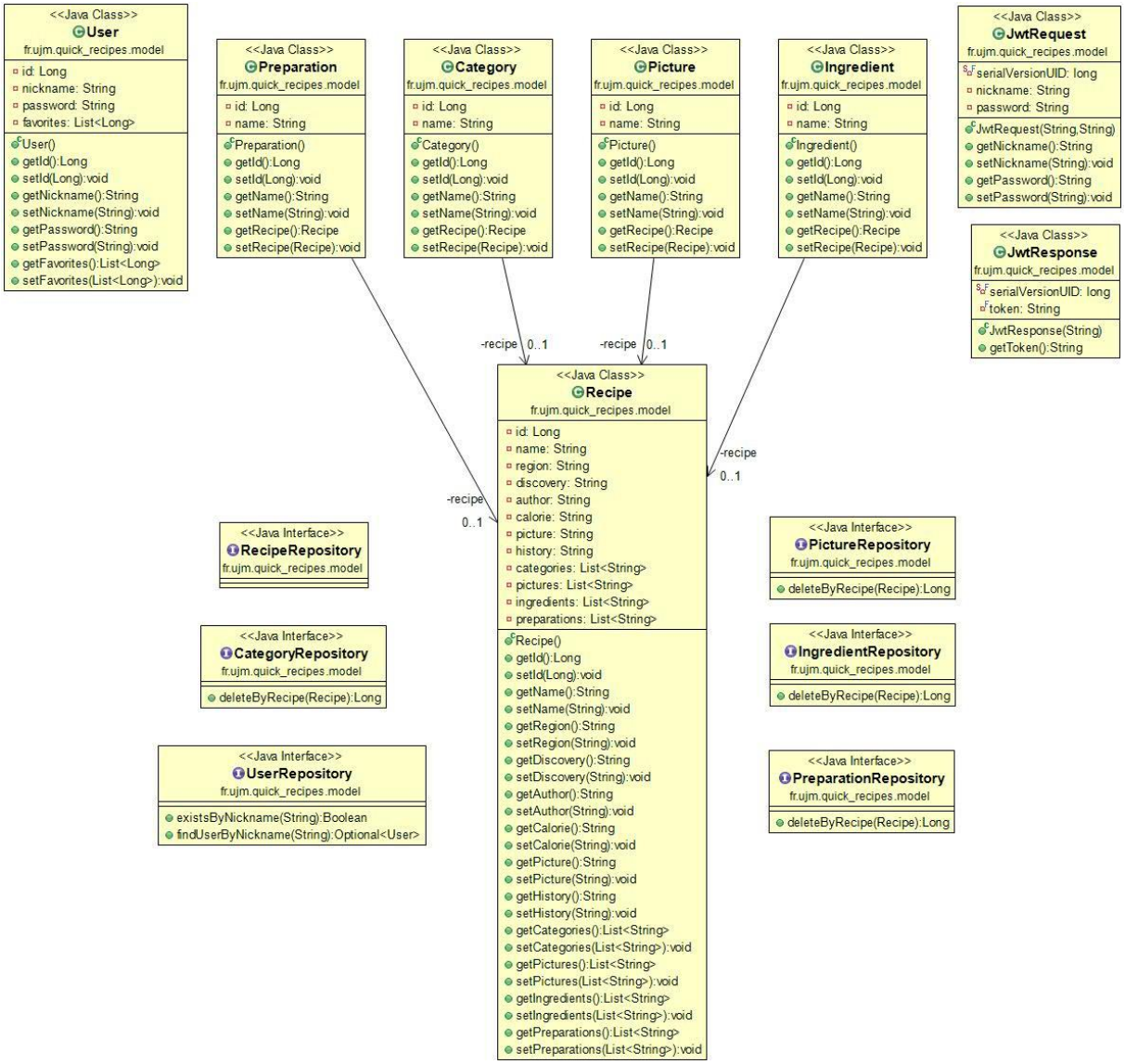


Figure 18 : Classes du package « model »

Recipe : Entité pour définir l'objet recette.

Category : Entité pour définir l'attribut catégorie d'une recette.

Picture : Entité pour définir l'attribut image d'une recette.

Ingredient: Entité pour définir l'attribut ingrédient d'une recette.

Preparation : Entité pour définir l'attribut préparation d'une recette.

User : Entité pour définir l'objet utilisateur.

JwtRequest : Classe pour définir l'objet utilisateur lors de l'authentification.

JwtResponse: Classe pour définir l'objet token lors des appels API.

Les classes se terminent par ***Repository*** sont utilisées par Hibernate pour générer les tables, dans la base de données, qui correspondent à chaque entité.

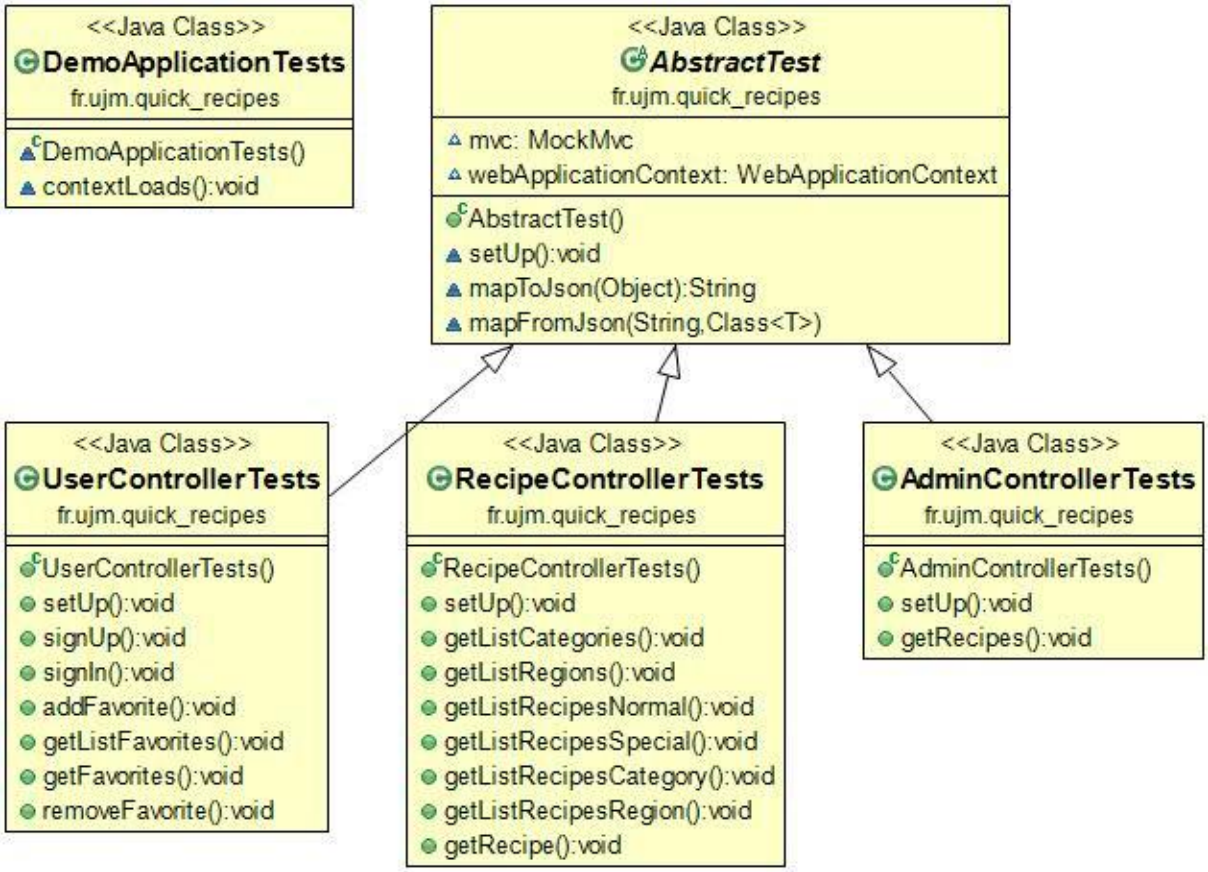


Figure 19 : Classes du package des tests

AbstractTest : Construction de l'environnement des tests.

UserControllerTests : Tests réalisés sur le contrôleur des utilisateurs.

RecipeControllerTests : Tests réalisés sur le contrôleur des recettes.

AdminControllerTests : Tests réalisés sur le contrôleur de l'administrateur.

6. Maquettes

6.1 Page d'accueil

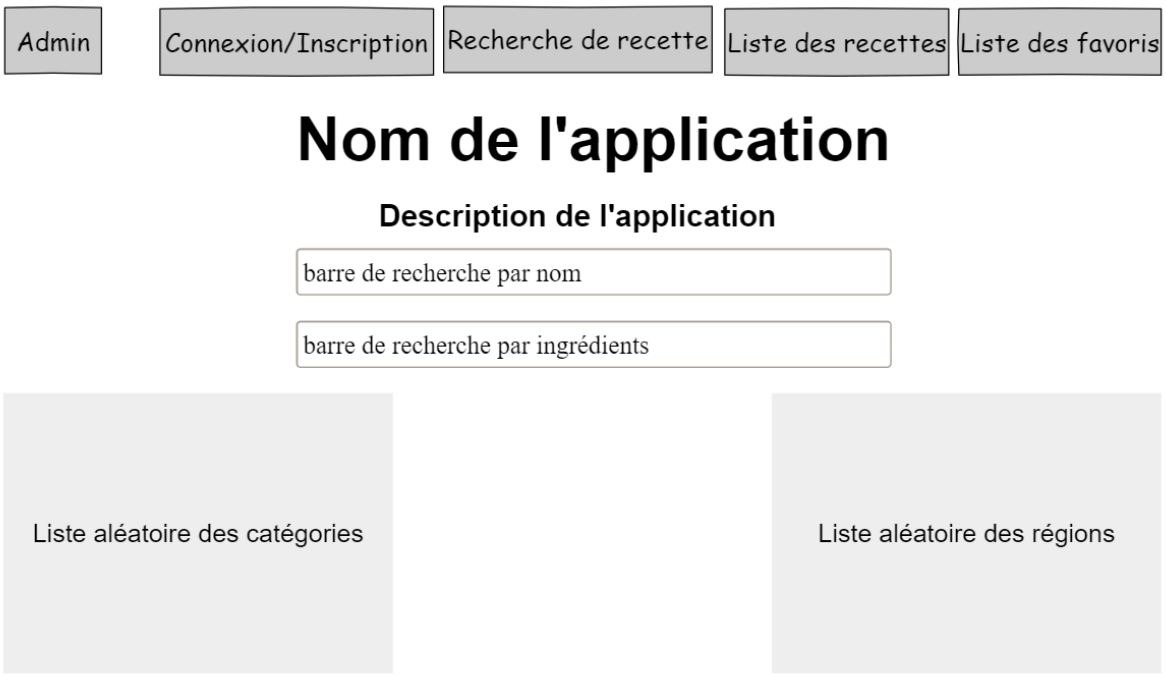


Figure 20 : Interface de la page d'accueil

Le design de la page d'accueil était réalisé en se basant sur un Template Bootstrap et adapté en fonction de l'environnement de l'application.

6.2 Page d'inscription et de connexion

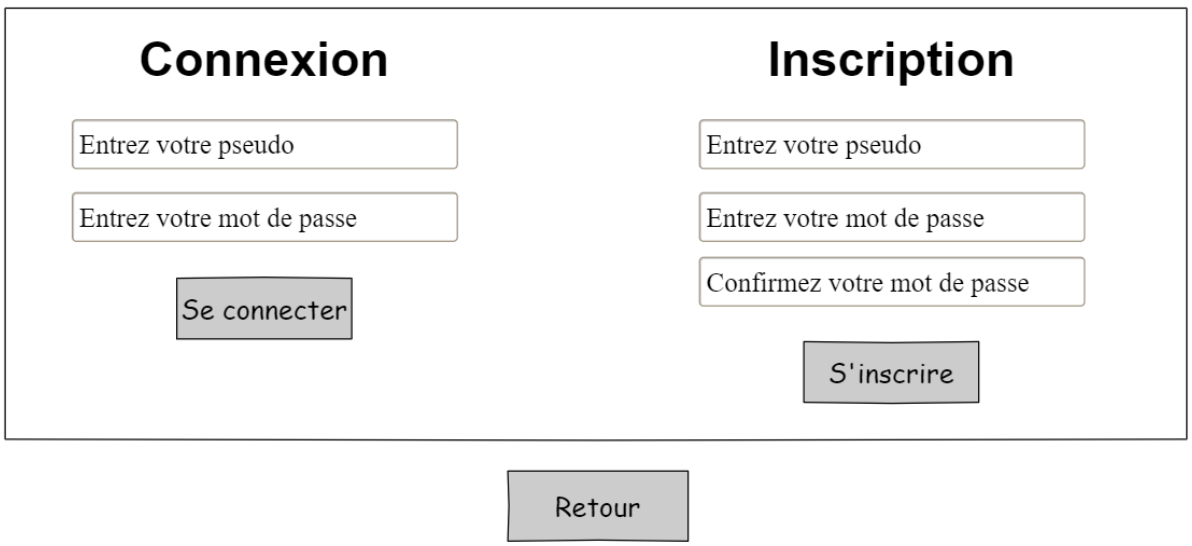


Figure 21 : Interface de la page d'inscription et de connexion

Pour une utilisation simple et rapide de l'application, l'inscription et la connexion sont mises sur la même page.

6.3 Page de visualisation de la liste des recettes

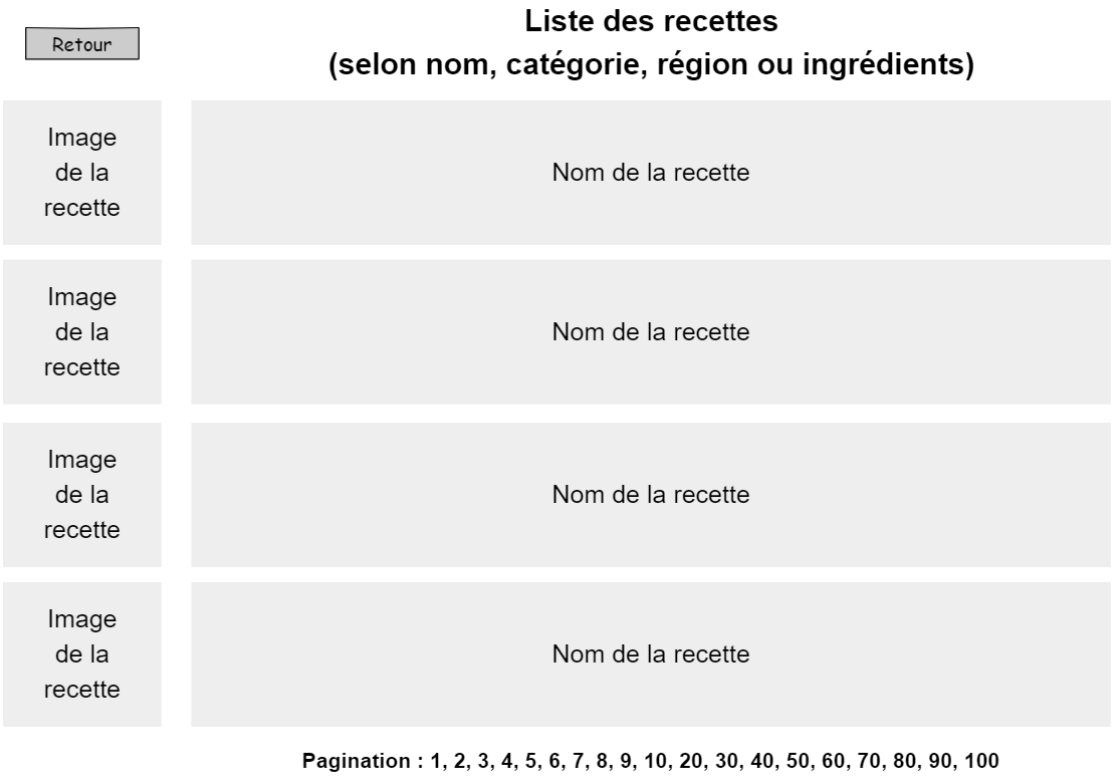


Figure 22 : Interface de visualisation de la liste des recettes

6.4 Page de visualisation d'une recette

Figure 23 : Interface de visualisation d'une recette (Introduction)

Figure 24 : Interface de visualisation d'une recette (Images)

Figure 25 : Interface de visualisation d'une recette (Ingrédients et préparation)

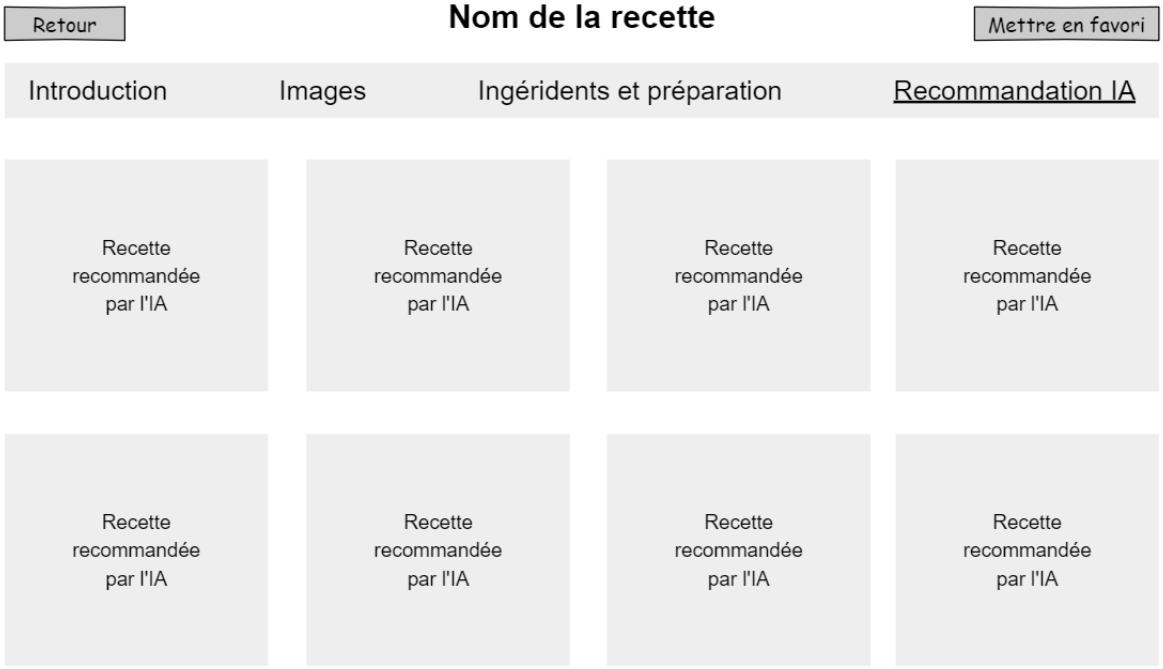


Figure 26 : Interface de visualisation d'une recette (Recommandation IA)

Le défilement entre les composants de l'interface de visualisation d'une recette, se fait de façon fluide sans le rechargement de la page.

6.5 Page de visualisation des recettes en favori

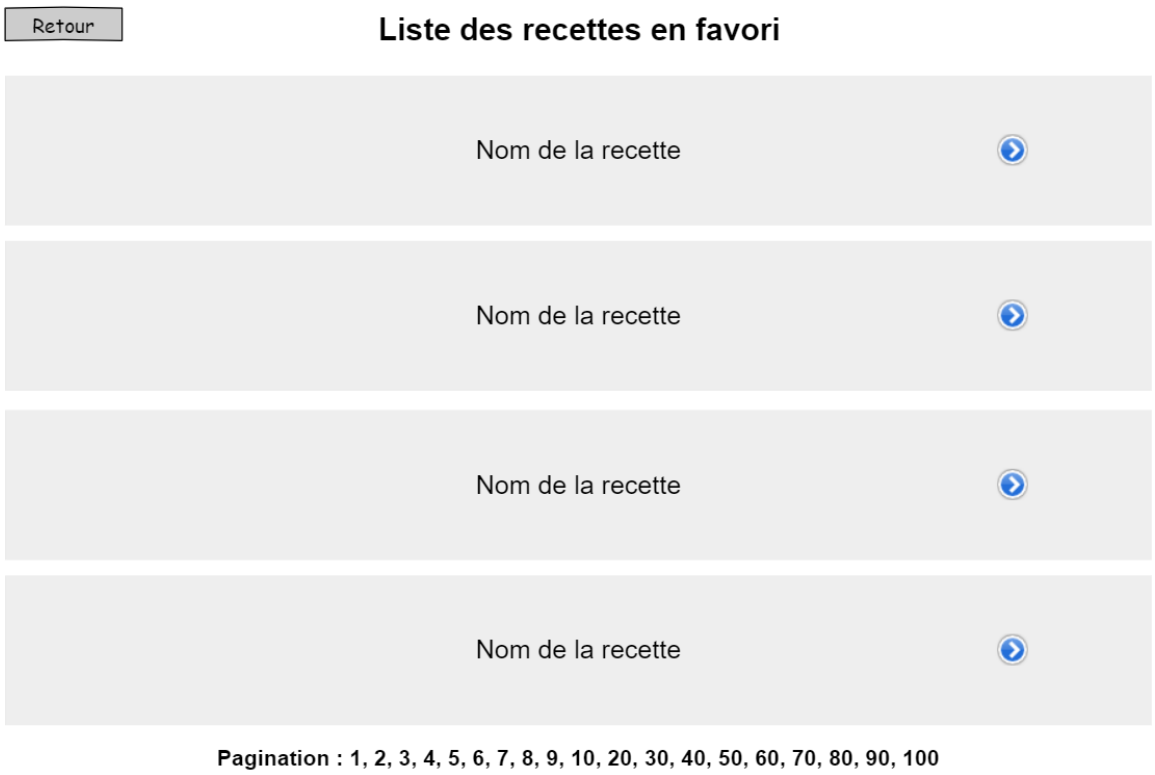


Figure 27 : Interface de visualisation des recettes en favori

6.6 Page d'ajout des recettes

Figure 28 : Interface d'ajout des recettes (Admin)

6.7 Page de suppression des recettes

Figure 29 : Interface de suppression des recettes (Admin)

7. Synthèse

7.1 Déroulement du projet

Nom	Date de début	Date de fin
Phase 1 : Initialisation du projet	20/04/2020	22/05/2020
Description de l'application	20/04/2020	21/04/2020
Fonctionnalités de l'application	22/04/2020	24/04/2020
Choix des technologies	27/04/2020	28/04/2020
Conception de la base de données	29/04/2020	01/05/2020
Construction du projet (client/serveur)	04/05/2020	05/05/2020
Rédaction du rapport	06/05/2020	08/05/2020
Conception IHM	11/05/2020	15/05/2020
Modélisation UML	18/05/2020	22/05/2020
Phase 2 : Codage du projet	25/05/2020	24/07/2020
Fonctionnalités prioritaires	25/05/2020	26/06/2020
Phase de développement	25/05/2020	19/06/2020
Phase de test	22/06/2020	26/06/2020
Fonctionnalités supplémentaires	29/06/2020	24/07/2020
Phase de développement	29/06/2020	17/07/2020
Phase de test	20/07/2020	24/07/2020
Phase 3 : Finalisation du projet	27/07/2020	14/08/2020
Intégration de l'application	27/07/2020	31/07/2020
Rédaction du rapport	03/08/2020	07/08/2020
Préparation de la soutenance	10/08/2020	14/08/2020

Figure 30 : Planning à l'état du 15/05/2020

Nom	Date de début	Date de fin
Phase 1 : Initialisation du projet	20/04/2020	22/05/2020
Description de l'application	20/04/2020	21/04/2020
Fonctionnalités de l'application	22/04/2020	24/04/2020
Choix des technologies	27/04/2020	28/04/2020
Conception de la base de données	29/04/2020	01/05/2020
Construction du projet (client/serveur)	04/05/2020	05/05/2020
Rédaction du rapport	06/05/2020	08/05/2020
Conception IHM	11/05/2020	15/05/2020
Modélisation UML	18/05/2020	22/05/2020
Phase 2 : Codage du projet	25/05/2020	07/08/2020
Fonctionnalités prioritaires	25/05/2020	03/07/2020
Phase de développement	25/05/2020	03/07/2020
Phase de test	29/06/2020	03/07/2020
Fonctionnalités supplémentaires	06/07/2020	31/07/2020
Phase de développement	06/07/2020	24/07/2020
Phase de test	27/07/2020	31/07/2020
Intégration de l'application	27/07/2020	07/08/2020
Phase 3 : Finalisation du projet	03/08/2020	28/08/2020
Rédaction du rapport	03/08/2020	21/08/2020
Intégration IA	03/08/2020	21/08/2020
Préparation de la soutenance	24/08/2020	28/08/2020

Figure 31 : Planning à l'état du 21/08/2020

La phase 1 du projet s'est globalement déroulé comme prévu vu que cette partie concerne principalement l'initialisation du projet, la conception et la modélisation, la prise de certaines décisions, et le début de la rédaction du rapport. Il est rare de rencontrer des imprévus dans ces situations.

La phase 2 du projet est la plus longue du projet, elle s'est terminée le 07/08 au lieu du 24/07. Environ deux semaines de retard que prévu dû à des difficultés rencontrées lors de la partie de codage. Heureusement, que le planning initial du projet a pris assez de marge pour gérer les tâches qui prennent plus de temps que prévu.

La phase 3 du projet s'est adaptée au retard causé par la phase précédente sans que la durée de ses tâches soit affectée. L'intégration de l'application s'est rajoutée à la phase 2, ce qu'a permis à la phase 3 de gagner une durée supplémentaire qui s'est partagée entre l'étude de l'IA et la finalisation du rapport.

7.2 Difficultés rencontrées

Comme dans la majorité des projets, on a tendance à rencontrer différents types de difficultés au fil de la progression sur le projet. L'objectif est de trouver une alternative à chaque problème sans toucher aux fondamentaux du projet. Dans notre cas, la plupart des difficultés sont intervenues durant la phase de codage :

- L'outil de conception des IHM est assez limité dans ses ressources, vu sa licence en open source. Les interfaces dessinées avec cet outil ont été modifiées pour s'adapter aux interfaces réalisées sous Angular. Pourtant, logiquement c'est à la partie de codage de s'inspirer de celle de conception.
- Les outils de modélisation génèrent des images non vectorielles. Plus on agrandit ces images, plus leur qualité se dégrade, ce qui nuit à la lecture des petits caractères.
- La génération d'une grande quantité d'images et les introduire à la base de données pour les tests en masse était hors de portée. Une alternative était d'ajouter la même image à toutes les recettes.
- La génération des données pour les tests en masse était effectuée avec le logiciel libre Mockaroo. Certaines données générées comme le nom, l'histoire et la méthode de préparation des recettes sont non réalistes.
- L'application utilise le modèle ORM dans la conception de sa base de données, cette couche engendre une limitation dans l'écriture des requêtes SQL. En effet, nos requêtes doivent être adaptées au langage HQL, les expressions régulières (Regex) et les méthodes comme LIMIT et OFFSET sont inconnus au langage HQL. Par conséquent, il s'agit de trouver des alternatives comme l'implémentation manuelle et l'ajout des Regex à la configuration du serveur ou l'utilisation des méthodes spécifiques au langage HQL.
- Les interfaces de l'application sont construites d'un mix entre plusieurs fichiers CSS. Parfois, il est assez compliqué d'appliquer le style voulu sur un élément, vu qu'il sera écrasé par un autre style plus prioritaire.

- Le lancement des scripts JS à l'intérieur des components d'Angular demande une démarche assez spécifique à effectuer, qui n'est pas toujours la meilleure des solutions.
- Conservation du planning initial ainsi que le développement de toutes les fonctionnalités prévues pour l'application.

7.3 Acquis

Bien que ce projet ait connu quelques rebondissements, les objectifs principaux ont été atteints et j'ai réussi à gagner de l'expérience sur certains aspects :

- Développement en environnement Angular.
- Écriture des requêtes en langage HQL.
- Utilisation du parseur DOM.
- Adaptation du planning en fonction de l'évolution du projet.

7.4 Quantité de code

	Front	Back	Test	Total
Nombre de package	5	3	1	9
Nombre de classe et d'interface	17	23	5	45
Lignes de code	1 597	1 873	293	3 763

Figure 32 : Statistiques de codage

Le calcul des lignes de code est l'addition du code lui-même, des commentaires, des imports et des espaces entre les méthodes. Pour la partie front, uniquement les fichiers qui se trouvent dans le dossier $\boldsymbol{app}$ ont été comptabilisés, à l'exception des fichiers $*.\boldsymbol{spec}$ générés automatiquement.

8. Conclusion

Le but ultime derrière le projet était de regrouper les connaissances que j'ai apprises tout au long de ma formation, en première année de Master Informatique, dans une seule application personnelle.

Les UE comme Programmation Web Avancée et Projet Intégré, m'ont aidé dans tous ce qui concerne le codage en Angular et Spring ainsi que l'utilisation des API REST. Mes acquis dans les UE comme Document Numérique et Interopérabilité, ont permis l'implémentation du module d'ajout des recettes dans la base de données à travers des fichiers XML. La réflexion sur le module d'IA est rendue plus simple à réaliser grâce aux UE comme Introduction à l'Intelligence Artificielle et Machine Learning.

Enfin, les interactions que j'ai eues dans les projets réalisés précédemment en groupe, m'ont donné une certaine expérience qu'a contribué dans le développement du produit.

Glossaire

JPA : Java Persistence API, une interface de programmation Java qui permet de relier les classes d'une application aux tables d'une base de données à travers des annotations.

REST : Representational State Transfer, un style d'architecture logicielle à respecter lors de la création des web services.

API : Application Programming Interface, un ensemble de fonctionnalités qui sert d'intermédiaire entre les requêtes des clients et les ressources du serveur.

OS : Operating System, un ensemble de programmes qui exploite les périphériques d'un ordinateur.

IDE : Integrated Development Environment, un ensemble d'outils et d'extensions qui permet d'accélérer le processus de développement.

ORM : Object-Relational Mapping, un type de programme informatique qui permet de simuler une base de données orientée objet à partir d'un programme applicatif et d'une base de données relationnelle.

HQL : Hibernate Query Language, un langage similaire au langage SQL, à la différence qu'il est utilisé pour interroger une base de données orientée objet.

Bibliographie

[1] PodcastScience, les algorithmes de recommandation, disponible sur :

Lien vers la ressource

[2] KDnuggets, content-based recommender using NLP, disponible sur :

Lien vers la ressource