Rapport Application Web

Projet Club de cuisine

ATIF Hamza
CHAKHS Marieme
ETTAOUIL Oussama
ZEROUALI Imane

Parcours : Architecture, système et réseaux

Encadré par : Djomgwe Teabe Boris

2024-2025

Table des matières

1 Introduction

Ce rapport présente le développement d'une application web dédiée à la gestion d'un club de cuisine. Cette application permet aux utilisateurs de partager des recettes, participer à des événements, commenter, discuter sur un forum, et gérer leur profil. Elle repose sur une architecture full-stack combinant un frontend développé en Angular et un backend en Spring Boot, avec une base de données relationnelle embarquée. Ce document détaille l'analyse, la conception, la réalisation technique ainsi que les fonctionnalités mises en place tout au long du projet.

2 Analyse et Conception

Diagramme de classe

Ce diagramme UML représente une application de partage de recettes. Les adhérents peuvent créer des recettes, y ajouter des ingrédients et des commentaires. Ils peuvent aussi participer à des discussions via des messages. Chaque entité (recette, message, commentaire) est liée à son auteur.

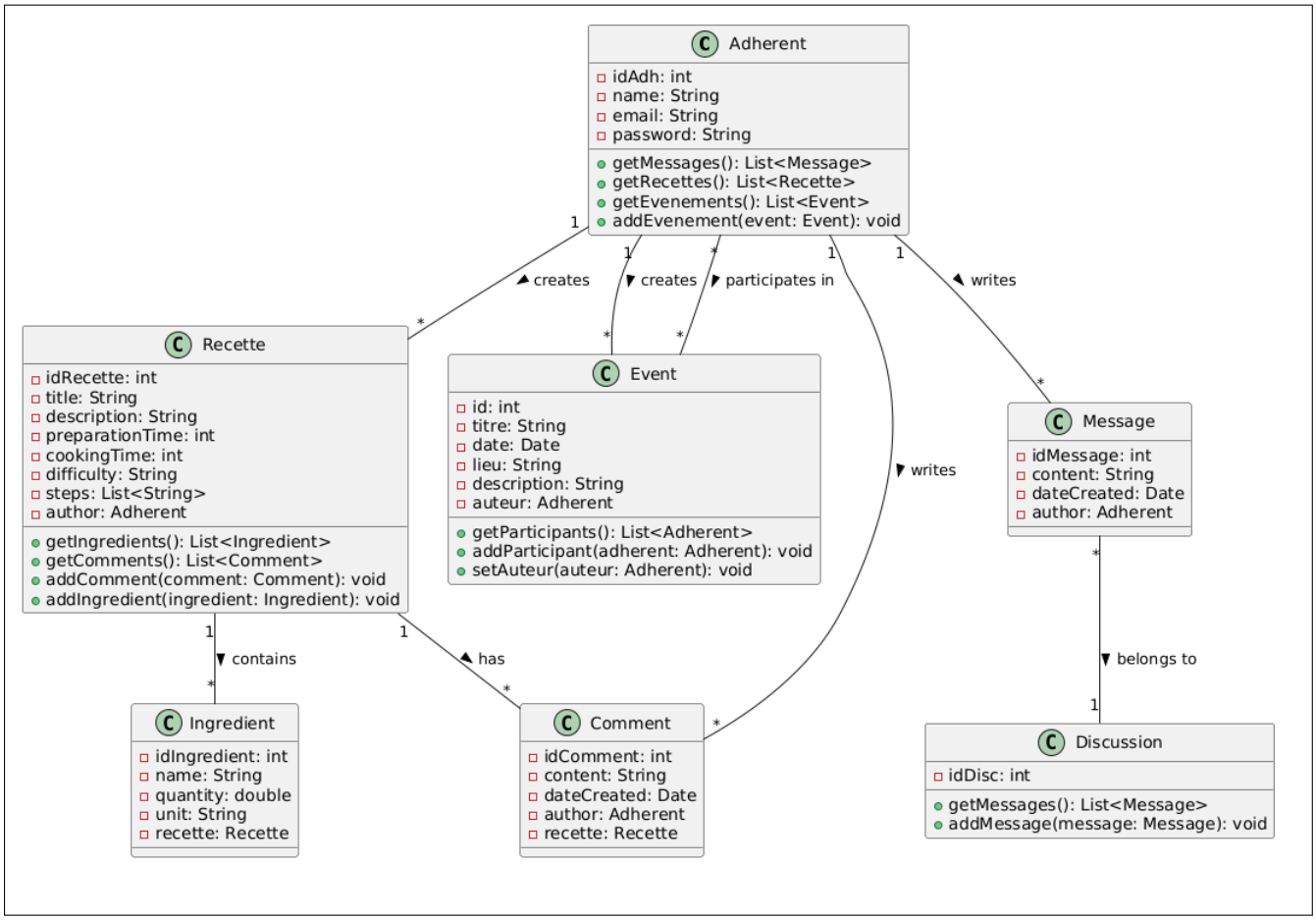

FIGURE 1 – Diagramme de classe

3 Réalisation

3.1 Technologies utilisées

Le projet repose sur une architecture web full-stack combinant :

— **Angular :** framework front-end pour créer une interface utilisateur dynamique et modulaire, avec gestion du routage et des appels HTTP.

— **Spring Boot :** framework back-end Java facilitant la création d'API REST avec gestion des entités, des routes et de la logique métier.

— **HSQLDB :** base de données relationnelle légère et embarquée, idéale pour le développement rapide et les tests.

— **Postman :** outil de test d'API REST permettant de vérifier les routes, les formats de réponse et le bon fonctionnement des échanges frontend/backend.

3.2 Fonctionnalités principales

3.2.1 Adhérents

— Inscription, connexion, mise à jour, suppression
— Récupération de la liste des adhérents ou d'un adhérent par ID

3.2.2 Recettes

— Ajout d'une recette
— Suppression d'une recette
— Récupération de toutes les recettes ou celles d'un auteur spécifique
— Gestion des ingrédients (nom, calories, quantité) et des étapes de préparation

3.2.3 Événements

— Création et suppression d'événements
— Récupération de tous les événements ou ceux d'un auteur spécifique
— Participation d'un adhérent à un événement

3.2.4 Commentaires

— Ajout d'un commentaire à une recette
— Récupération des commentaires associés à une recette

3.2.5 Forum

— Création de discussions avec titre, question et auteur
— Ajout de messages dans une discussion
— Récupération de toutes les discussions ou des messages d'une discussion spécifique

3.3 Frontend

Le frontend de l'application a été réalisé en Angular. L'interface est structurée en plusieurs composants réutilisables et modulaires :

— **Composant d'authentification** : permet l'inscription et la connexion des utilisateurs. L'interface s'inspire du style de Strava, avec une image de plat en fond pour rester dans le thème culinaire.
— **Page adhérent** : affiche les informations du profil, la liste des recettes partagées, et les messages du forum postés par l'utilisateur. Un bouton « Modifier » permet de mettre à jour les informations via une boîte de dialogue (modal).
— **Gestion des recettes** : composant principal affichant la liste des recettes disponibles. Il inclut une mise en page responsive et dynamique, avec récupération des données depuis le backend via des appels API.
— **Ajouter des recettes** : composant permettant aux utilisateurs authentifiés d'ajouter une nouvelle recette via un formulaire intuitif. Ce composant assure également la validation des champs et l'envoi des données au backend via des requêtes HTTP POST.

— **Detail de recettes** : composant dédié à l'affichage détaillé d'une recette sélectionnée, incluant les ingrédients, étapes de préparation, auteur, etc.

— **Page "À propos"** : section présentant le projet et ses objectifs, avec un design épuré et adapté aux écrans mobiles.

— **Page "Contact"** : formulaire simple permettant aux visiteurs de contacter le club, avec gestion de la saisie et possibilité d'envoi simulé ou réel selon les configurations.

— **Forum** : composant affichant toutes les discussions, avec leurs messages. L'utilisateur peut ajouter un message ou consulter ceux qu'il a postés via son espace personnel.

— **Événements** : liste des événements disponibles, possibilité de participation via un bouton dédié. Une fois connecté, l'utilisateur est redirigé vers sa page personnelle avec l'événement mis en évidence dans un calendrier.

— **Routage Angular** : mis en place pour relier les différentes pages, avec des guards d'authentification pour protéger les routes privées (profil, ajout de recette, etc.).

— **Services HTTP** : chaque fonctionnalité interagit avec le backend via un service Angular dédié utilisant le module `HttpClient`.

— **Composants partagés** : header, footer et sidebar ont été conçus pour être réutilisés sur toutes les pages de l'application.

3.4 Backend

Le backend de l'application est développé avec **Spring Boot**, un framework Java qui simplifie la création d'applications web en fournissant une configuration par défaut et une gestion centralisée des dépendances.

3.4.1 Structure générale

L'application repose sur un **contrôleur principal unique**, nommé `Facade`, qui regroupe l'ensemble des endpoints REST. Ce contrôleur centralise les opérations CRUD liées aux différentes entités fonctionnelles du projet.

3.4.2 Entités principales

Les entités représentent les modèles de données du système. Elles sont mappées à la base de données à l'aide de JPA (Java Persistence API). Voici les principales entités utilisées :

— **Adherent** : utilisateur de la plateforme (nom, prénom, email, mot de passe, etc.).

— **Recette** : composée d'un nom, d'une photo, d'ingrédients, d'étapes, de catégories, et d'un auteur.

— **Ingredient** : utilisé dans une recette, avec un nom, une quantité et un nombre de calories.

— **Event** : événement avec un titre, une date, un lieu, une description et un auteur.

— **Comment** : commentaire laissé par un adhérent sur une recette.

— **Discussion** : sujet de forum avec un titre, une question et un auteur.

— **Message** : réponse postée dans une discussion.

3.4.3 Repositories

Chaque entité est associée à un **repository** qui étend l'interface `JpaRepository`. Cela permet de bénéficier automatiquement des opérations CRUD (Create, Read, Update, Delete) sans implémentation manuelle.
Exemples de repositories :

- `AdherentRepository`
- `RecetteRepository`
- `EventRepository`
- `CommentRepository`
- `DiscussionRepository`
- `MessageRepository`

3.4.4 Base de données

Le projet utilise **HSQLDB** (HyperSQL DataBase), une base de données relationnelle embarquée.

- **Type** : base en mémoire, adaptée pour les tests et le développement.
- **Scripts** : les fichiers de configuration et d'initialisation sont présents dans le répertoire `db/`.

Elle est automatiquement configurée au lancement de l'application via Spring Boot.

3.4.5 API REST

Le backend expose une **API REST complète** accessible via `http://localhost:8080`. Chaque ressource dispose de ses propres routes pour gérer les opérations de type CRUD. Exemples d'actions possibles :

- `POST /adherents/inscription` : inscription d'un nouvel utilisateur
- `GET /recettes` : récupération de toutes les recettes
- `POST /evenements/ajout` : création d'un événement
- `POST /recettes/ajout-avec-image` : ajout d'une recette avec image
- `GET /discussions/{id}/messages` : récupération des messages d'une discussion

Le backend est configuré pour accepter les requêtes provenant du frontend Angular via l'annotation `@CrossOrigin(origins = "http://localhost:4200")`.

3.5 Code source

Le code source complet de ce projet est disponible sur le dépôt GitHub suivant :

`https://github.com/AtifHamzaN7/Projetweb.git`

Ce dépôt contient l'ensemble des fichiers du backend (Spring Boot), du frontend (Angular), ainsi que les scripts de base de données et les ressources nécessaires au bon fonctionnement de l'application.

4 Axes d'amélioration

- **Sécurité** :
  - Mettre en place une authentification (via JWT ou sessions) pour protéger les endpoints sensibles.
  - Chiffrer les mots de passe dans la base de données (par exemple avec BCrypt) afin de garantir la confidentialité des données utilisateurs.
- **Gestion des modifications et suppressions** : permettre aux utilisateurs de modifier ou supprimer leurs propres commentaires et messages du forum, avec contrôle d'accès approprié.
- **Validation des entrées** : appliquer des règles strictes sur les données saisies par l'utilisateur (adresse email valide, mot de passe sécurisé, contrôle des champs obligatoires).
- **Gestion des erreurs** : retourner des messages d'erreur explicites et structurés côté API pour faciliter le débogage et améliorer l'expérience utilisateur.
- **Pagination** : ajouter la pagination sur les listes longues (recettes, discussions, événements, etc.) pour améliorer les performances et l'ergonomie.
- **Tests automatisés** : développer des tests unitaires et d'intégration afin d'assurer la robustesse et la fiabilité du backend.

5 Conclusion

Ce projet d'application web pour un club de cuisine nous a permis de mettre en pratique l'ensemble des compétences acquises en développement full-stack. Grâce à l'utilisation de technologies modernes comme Angular et Spring Boot, nous avons conçu une plateforme fonctionnelle permettant aux utilisateurs de partager des recettes, interagir via un forum et participer à des événements. Ce travail nous a également sensibilisés aux bonnes pratiques d'architecture logicielle, d'organisation en équipe et de gestion de projet.