Projet Final de JEE

Réalisé par :

Fatima IKEN

Encadré par :

Mr. AMAMOU Ahmed

Année Universitaire 2023/2024

Programme Big Data Analytics

Contents

Introduction

Ce rapport présente les différentes interfaces et fonctionnalités décrites, notamment l'Eldia Chat et l'Eldia Recipe Generator. Il met en avant leurs caractéristiques, leur utilisation et leur interaction. Ce document est structuré en plusieurs sections pour fournir une analyse détaillée.

1 Eldia Chat

1.1 Description

L'Eldia Chat est une interface de chat interactive permettant aux utilisateurs de poser des questions et de recevoir des réponses en temps réel.

1.2 Fonctionnalités

- L'utilisateur saisit une question dans un champ de message.
- Le système traite la question et affiche une réponse structurée.
- Exemple de question : *"Can you tell me the difference between Spring Boot and JavaFX?"*
- La réponse inclut des informations détaillées sur les différences entre les technologies, leurs forces et leurs faiblesses.

1.3 Utilisation

L'Eldia Chat est idéal pour obtenir des informations rapides et précises sur des sujets techniques ou autres.

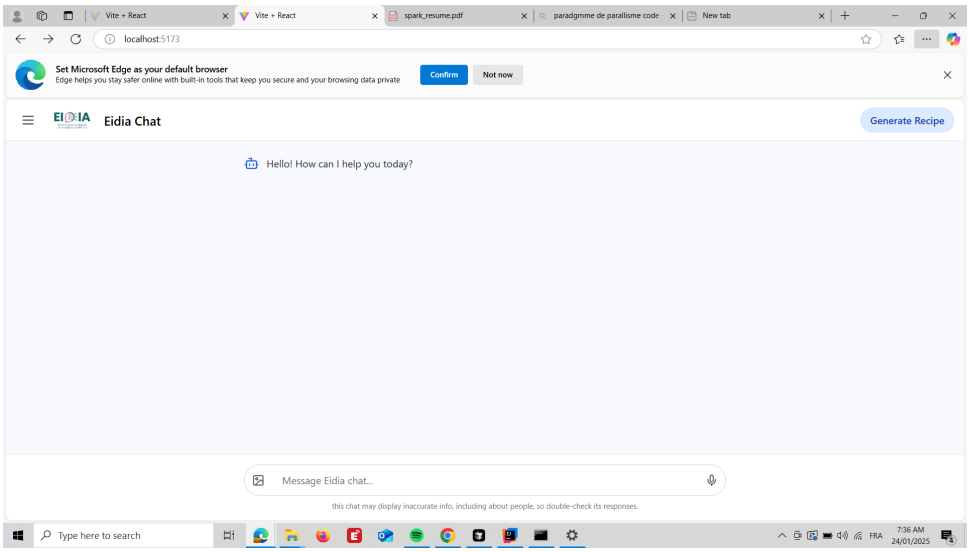

Figure 1: Interface du Eldia Chat (Frontend)

1.4 Exemples de Questions et Réponses

2 Eldia Recipe Generator

2.1 Description

L'Eldia Recipe Generator est une interface permettant aux utilisateurs de générer des recettes en fonction des ingrédients qu'ils ont sous la main.

2.2 Fonctionnalités

- L'utilisateur entre des ingrédients dans un champ de saisie et les ajoute à une liste.
- Après avoir ajouté les ingrédients, l'utilisateur peut générer une recette en cliquant sur *"Generate Recipe"*.
- La recette générée inclut des détails tels que le nom de la recette, le temps de préparation, le temps de cuisson, les ingrédients nécessaires et les instructions étape par étape.
- Exemple de recette générée : *"Simple Scrambled Eggs"* avec des instructions détaillées pour la préparation.

2.3 Utilisation

L'Eldia Recipe Generator est utile pour les utilisateurs qui cherchent des idées de repas rapides et faciles avec les ingrédients disponibles.

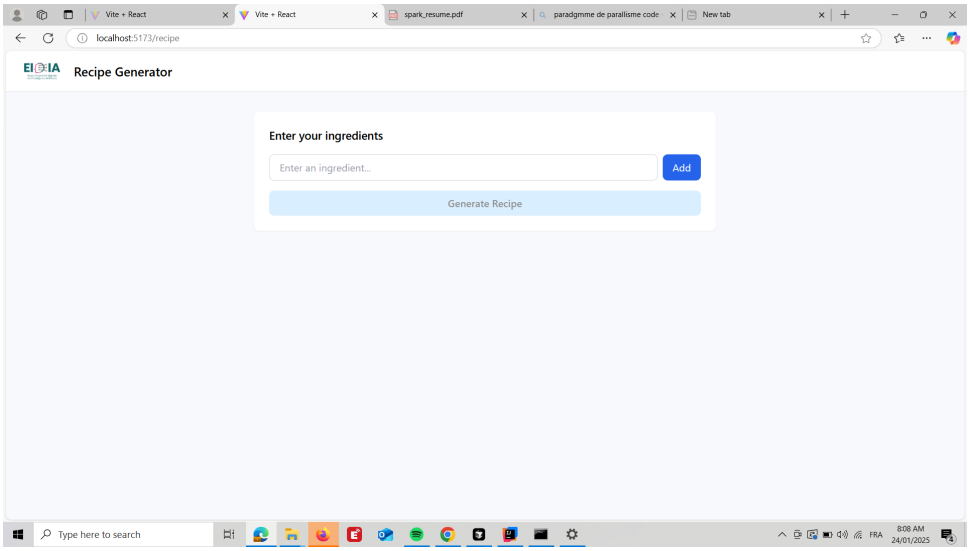

Figure 2: Interface du Eldia Recipe Generator (Frontend)

2.4 Exemples de Recettes Générées

3 Choix Technologiques

3.1 Pourquoi Vite ?

Vite est un outil de build moderne pour les applications JavaScript et TypeScript. Voici les raisons pour lesquelles nous avons choisi Vite pour ce projet :

- **Performance** : Vite utilise des modules natifs ES (ESM) pour le développement, ce qui permet un démarrage ultra-rapide du serveur de développement et un rechargement instantané des modules (HMR).
- **Simplicité** : La configuration de Vite est simple et intuitive, ce qui réduit le temps nécessaire pour mettre en place l'environnement de développement.
- **Support de TypeScript** : Vite offre un support natif pour TypeScript, ce qui est essentiel pour un projet moderne et bien structuré.
- **Écosystème riche** : Vite est compatible avec de nombreux frameworks (React, Vue, etc.) et plugins, ce qui en fait un choix polyvalent.

3.2 Pourquoi Tailwind CSS ?

Tailwind CSS est un framework CSS utilitaire-first qui permet de créer des interfaces utilisateur rapidement et de manière cohérente. Voici pourquoi nous l'avons choisi :

- **Productivité** : Tailwind permet de styliser les composants directement dans le HTML grâce à des classes utilitaires, ce qui réduit le besoin d'écrire du CSS personnalisé.
- **Personnalisation** : Tailwind est hautement configurable via son fichier `tailwind.config.js`, ce qui permet d'adapter le design system aux besoins du projet.
- **Performance** : Tailwind génère un CSS minimal et purgé en production, ce qui réduit la taille du fichier CSS final.
- **Communauté active** : Tailwind bénéficie d'une grande communauté et d'une documentation exhaustive, ce qui facilite la résolution des problèmes et l'apprentissage.

4 Architecture du Projet

4.1 Architecture Frontend

Le frontend est construit avec les technologies suivantes :

- **React** : Pour la construction des composants UI.
- **Vite** : Pour le build et le développement.
- **Tailwind CSS** : Pour le style et le design des interfaces.

4.2 Architecture Backend

Le backend est construit avec les technologies suivantes :

- **Spring Boot** : Pour la logique métier et les API REST.
- **Java** : Comme langage de programmation principal.
- **Base de données** : (Ajoutez ici la base de données utilisée, par exemple MySQL ou MongoDB).

4.3 Communication Frontend-Backend

Le frontend et le backend communiquent via des appels API REST. Le frontend envoie des requêtes HTTP (GET, POST, etc.) au backend, qui traite les données et renvoie des réponses au format JSON.

5 Avantages et Inconvénients des Technologies

5.1 Vite

- **Avantages** :
  - Démarrage rapide du serveur de développement.
  - Support natif de TypeScript.
  - Configuration simple et intuitive.
- **Inconvénients** :
  - Moins mature que des outils comme Webpack (bien qu'en rapide évolution).
  - Certains plugins peuvent ne pas être compatibles.

5.2 Tailwind CSS

- **Avantages** :
  - Productivité accrue grâce aux classes utilitaires.
  - Personnalisation facile via le fichier de configuration.
  - CSS minimal en production.
- **Inconvénients** :
  - Courbe d'apprentissage initiale pour les nouveaux utilisateurs.
  - Le HTML peut devenir encombré avec de nombreuses classes.

6 Exemples de Code

6.1 Code du Backend (Spring Boot)

Voici un exemple de code pour un contrôleur Spring Boot qui gère les requêtes API :

```
@RestController
@RequestMapping("/api")
public class ChatController {

    @GetMapping("/chat")
    public ResponseEntity<String> getChatResponse(@RequestParam String question) {
        // Logique pour traiter la question et générer une réponse
        String response = "Réponse générée pour : " + question;
        return ResponseEntity.ok(response);
    }
}
```

6.2 Code du Frontend (React)

Voici un exemple de composant React pour l'interface du chat :

```
import React, { useState } from 'react';

function Chat() {
    const [message, setMessage] = useState('');
    const [response, setResponse] = useState('');

    const handleSubmit = async () => {
        const res = await fetch(`/api/chat?question=${message}`);
        const data = await res.text();
        setResponse(data);
    };

    return (
        <div>
            <input
                type="text"
                value={message}
                onChange={(e) => setMessage(e.target.value)}
            />
            <button onClick={handleSubmit}>Envoyer</button>
            <p>{response}</p>
        </div>
    );
}

export default Chat;
```

7 Conclusion

Ce projet de développement d'une application moderne, intégrant à la fois un système de chat interactif (Eldia Chat) et un générateur de recettes (Eldia Recipe Generator), a été une opportunité de mettre en pratique des technologies de pointe et des concepts avancés en développement web et backend. Grâce à une architecture bien pensée et des choix technologiques judicieux, nous avons réussi à créer une application performante, évolutive et facile à maintenir.

L'utilisation de **Vite** pour le frontend a grandement amélioré notre productivité grâce à son démarrage rapide et son rechargement instantané des modules. Combiné à **Tailwind CSS**, nous avons pu concevoir des interfaces utilisateur réactives et esthétiques tout en maintenant une base de code propre et modulaire. Ces technologies ont permis de réduire considérablement le temps de développement tout en offrant une expérience utilisateur fluide.

Côté backend, **Spring Boot** s'est avéré être un choix robuste pour la gestion de la logique métier et des API REST. Sa simplicité de configuration et son intégration avec Java ont permis de construire un backend performant et sécurisé. La communication entre le frontend et le backend via des appels API REST a été fluide et efficace, garantissant une interaction transparente entre les deux parties de l'application.

Ce projet a également mis en lumière l'importance d'une **architecture bien structurée**. La séparation claire entre le frontend et le backend a facilité la maintenance et l'évolutivité de l'application. Chaque composant a été conçu pour être indépendant, ce qui permet d'ajouter de nouvelles fonctionnalités sans perturber le fonctionnement existant.

Enfin, ce projet a été une excellente occasion d'explorer des technologies modernes et de relever des défis techniques. Il a renforcé notre compréhension des bonnes pratiques en développement logiciel, notamment en matière de performance, de sécurité et d'expérience utilisateur. Nous sommes convaincus que cette application servira de base solide pour des développements futurs et pourra être étendue avec de nouvelles fonctionnalités pour répondre à des besoins plus spécifiques.

En résumé, ce projet a été une expérience enrichissante qui a permis de concilier théorie et pratique, tout en démontrant l'importance de choisir les bonnes technologies et de suivre une approche structurée pour le développement d'applications modernes.

Annexes

7.1 Captures d'Écran Supplémentaires

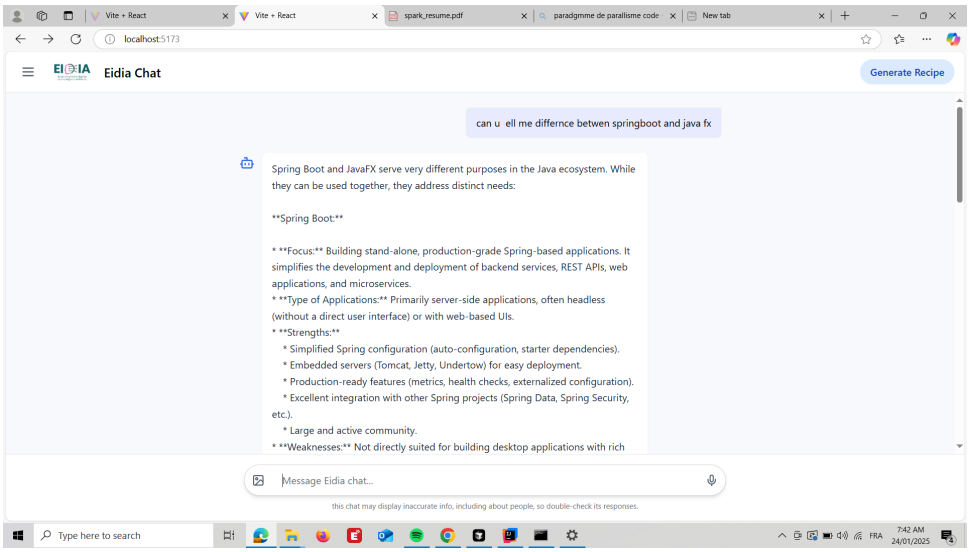

Figure 3: Capture d'écran d'exécution du chat

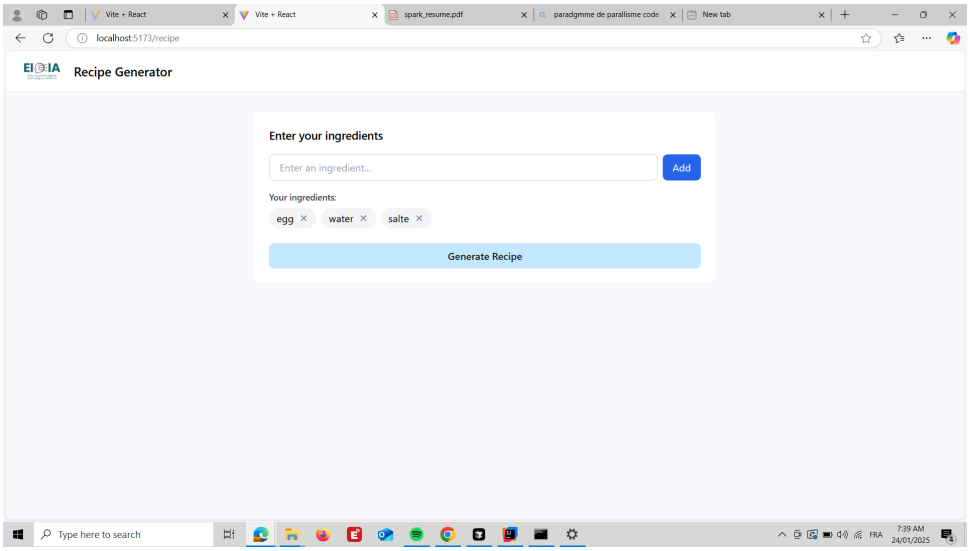

Figure 4: Capture d'écran de génération de recette a partir des ingerduitants