Documentation technique de l’implémentation

Pour ce projet, l’implémentation repose sur un modèle client-serveur classique. Le front-end utilise HTML, CSS et JavaScript vanilla. Le back-end a été conçu avec Node.js et Express, et utilise SQLite pour la gestion des données. Les différentes fonctionnalités sont accessibles via une API REST sécurisée, permettant la gestion des utilisateurs, des recettes et des recommandations. Le projet est déployé sur Render.

Architecture technique

- **Front-end** : HTML5, CSS3, JavaScript (DOM manipulation)
- **Back-end** : Node.js avec le framework Express.js
- **Base de données** : SQLite
- **Déploiement** : Render (hébergement du serveur Node.js) & Vercel

Le projet est découpé entre une partie statique (HTML/CSS/JS) et une partie serveur qui gère les requêtes API et la communication avec la base de données.

Requêtes API

Gestion des recettes

- GET /api/recettes : Récupérer la liste de toutes les recettes
- GET /api/recettes/:id : Récupérer une recette spécifique via son identifiant
- POST /api/recettes : Ajouter une nouvelle recette (réservé aux administrateurs)
- PUT /api/recettes/:id : Modifier une recette existante
- DELETE /api/recettes/:id : Supprimer une recette

Gestion des recommandations

- GET /api/recommandations : Obtenir toutes les recommandations disponibles
- GET /api/recommandations/utilisateur/:id : Obtenir les recommandations d'un utilisateur donné
- POST /api/recommandations : Ajouter une nouvelle recommandation (réservé aux utilisateurs connectés)
- DELETE /api/recommandations/:id : Supprimer une recommandation (réservé au créateur de la recommandation)

Authentification

- POST /api/login : Connexion d'un utilisateur classique

- POST /api/admin/login : Connexion d'un administrateur
- POST /api/register : Inscription d'un nouvel utilisateur

Choix d'implémentation

Pour l'architecture technique, nous avons opté pour du HTML5, CSS3 et JavaScript vanilla, sans frameworks lourds qui auraient augmenté la taille des pages. Notre backend repose sur Node.js avec Express.js et une base de données SQLite légère, configurée pour minimiser les requêtes. En ce qui concerne le frontend, nous avons privilégié un design léger avec une interface simple sans animations inutiles, réduisant considérablement la charge des navigateurs.

Les points principaux :

- **Sécurité basique** : Contrôle du rôle utilisateur (admin ou utilisateur) côté serveur pour limiter certaines actions (ex : suppression, ajout de recette)

- **Gestion dynamique du front-end** : Le bouton "Déconnexion" et certaines options d'administration apparaissent ou disparaissent dynamiquement selon l'état de connexion de l'utilisateur

- **Manipulation DOM manuelle** : Tous les éléments du site sont mis à jour avec du JavaScript vanilla, sans frameworks supplémentaires

- **Responsive design** : Le site a été pensé pour être utilisable aussi bien sur desktop que sur mobile

- **Optimisation écologique** : Respect des bonnes pratiques Green IT : site léger, peu d'images lourdes, code simplifié

Et afin de limiter la consommation énergétique liée aux accès à la base de données, nous avons opté pour l'ouverture et la fermeture ponctuelle des connexions à chaque requête. Étant donné le faible volume de trafic attendu sur le site, l'utilisation de connexions non persistantes permet de réduire l'empreinte carbone en évitant de maintenir inutilement des connexions ouvertes.