Application web : recherche de recettes de cuisine en fonction de leurs ingrédients

Travail de Bachelor

Oliver Richani

Juin 2023

Supervisé par :

Prof. Dr. Jacques Pasquier-Rocha
Software Engineering Group

Groupe Génie Logiciel
Département d'Informatique
Université de Fribourg (Suisse)

Table des matières

Liste des figures

Liste des codes source

1
Introduction

1.1. But du projet

Les sites internet et les applications font partie du quotidien de l'immense majorité de la population, dans un monde où quasiment tout est en voie de numérisation. Les outils utilisés, autant les programmes que les appareils sont si souvent connectés les uns aux autres qu'il en devient essentiel pour quiconque prétendrait travailler dans le domaine de l'informatique, de comprendre en détail comment un programme connecté à internet et utilisant des technologies modernes fonctionne et est mis sur pied.

Ce travail sert d'expérience concrète à la création d'une application web, ainsi qu'à la découverte et prise en main des différentes technologies mises à disposition à notre époque, servant à rendre ce type de programmes possibles.

1.2. Objectifs

L'objectif central est la création d'une application web permettant aux utilisateurs de trouver des idées de plats à cuisiner, en fonction des ingrédients qu'ils possèdent déjà et qu'ils n'avaient pas délibérément prévu d'utiliser ensemble. Ceci s'inscrit dans une idée de réduction de la consommation inutile et du gaspillage, en proposant des possibilités inopinées à partir d'ingrédients déjà en possession des utilisateurs.

Cette implémentation comprend la création d'un front-end sous forme de site web facilement utilisable par n'importe quel utilisateur, d'une base de donnée pour les recettes et d'un back-end permettant de trouver les différentes recettes possibles à partir d'ingrédients donnés par l'utilisateur.

1.3. Structure du travail

Chapitre 1 : Introduction

Contient les buts et objectifs du travail, ainsi que la structure de ce dernier avec un court résumé de chaque section.

Chapitre 2 : État de l'art

Présente plusieurs applications similaires et une comparaison technique de leurs implémentations.

Chapitre 3 : Précision des exigences

Les besoins quant à la réalisation de ce projet sont décrits dans ce chapitre, en définissant les fonctionnalités retenues, les technologies utilisées ainsi que les méthodes pour y parvenir.

Chapitre 4 : Réalisation

Ce chapitre contient le processus de création de l'application, du back-end et du front-end, ainsi que quelques problèmes rencontrés.

Chapitre 5 : Résultat

Dans ce chapitre, le résultat final est présenté et analysé avec sa documentation.

Chapitre 6 : Conclusion

Finalement, ce chapitre contient les différents ajouts et modifications potentielles qui permetteraient d'améliorer ce projet, et un résumé des différents aspects de la création de ce travail, ainsi que ma perspective sur ce qu'il m'a apporté.

2
État de l'art

2.1. Quelques applications

De nombreuses implémentations du concept ont déjà été mises en place de manière plus ou moins variée et originale. En voici quelques unes qui m'ont paru utiles à donner une idée générale de l'approche qui peut être prise.

2.1.1. EmptyMyFridge

Cette application mobile est la plus complète et complexe de toutes celles abordées dans cette section. Le but de cette application est de proposer une gestion complète de l'inventaire alimentaire de l'utilisateur. Elle contient trois sections principales : Frigo, Armoire et Congélateur. Il est possible d'y inscrire des ingrédients que l'on possède déjà, pour avoir un suivi précis de l'inventaire. Les produits qui expirent rapidement (comme les fruits et les légumes) obtiennent une approximation de leur date de péremption, à partir du moment où ils ont été ajoutés.

Il est également possible de gérer une liste de course, un profil, ainsi que les recettes disponibles. Les ingrédients peuvent être ajoutés manuellement à l'inventaire en écrivant leurs noms ou alors en scannant un code barre. La recherche est faite à travers une liste d'ingrédients communément utilisés, mais il est permis d'en ajouter dans le cas où on ne trouverait pas ce qu'on cherche. Cet ajout est propre à l'utilisateur, mais le nom de l'ingrédient est potentiellment ajouté chez tous le monde, puisque certains n'ont que leur nom disponible dans la liste générale, et l'ajout doit se faire manuellement ensuite.

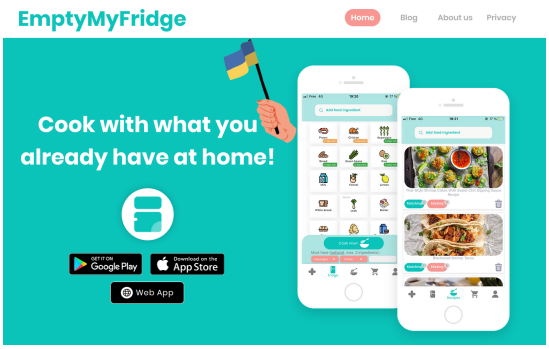

FIGURE 2.1. – Page d'accueil du site web [1]

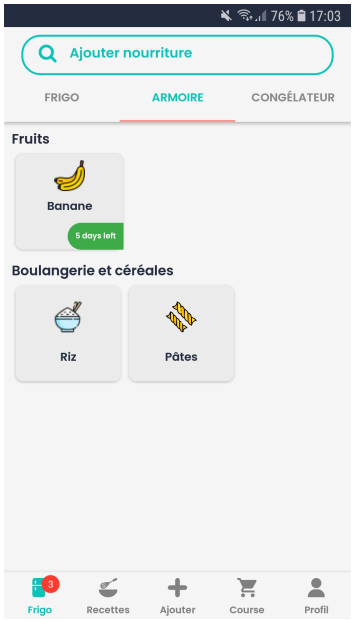

FIGURE 2.2. – Page d'accueil de l'application

Dans l'onglet "recette", il est immédiatement proposé des recettes possibles en fonction des ingrédients présents dans l'inventaire de l'utilisateur. Les ingrédients périssables sont choisis en priorité (la priorité de chaque ingrédient peut être modifiée) et il est également possible de faire une recherche de recette en fonction d'ingrédients sélectionnés manuellement.
Une recette peut être ajoutée à ses favoris et l'utilisateur peut en soumettre de nouvelles. Elles sont disponibles avec les instructions directement indiquées dans l'application et aussi en tant que lien externe vers un site web l'hébergeant (blog post) ou une vidéo de démonstration (Youtube principalement). Il est indiqué sur la recette combien d'ingrédients ont "matché" et combien il en manque pour la réaliser (il est possible de visualiser facilement lesquels), et si elle est végétarienne, vegan, etc.

L'application se veut être un gestionnaire complet des ingrédients ménagers, puisqu'en plus de proposer des recettes possibles avec ces derniers, il donne la possibilité d'en garder un suivi continu.

1. Website de EmptyMyFridge [1]

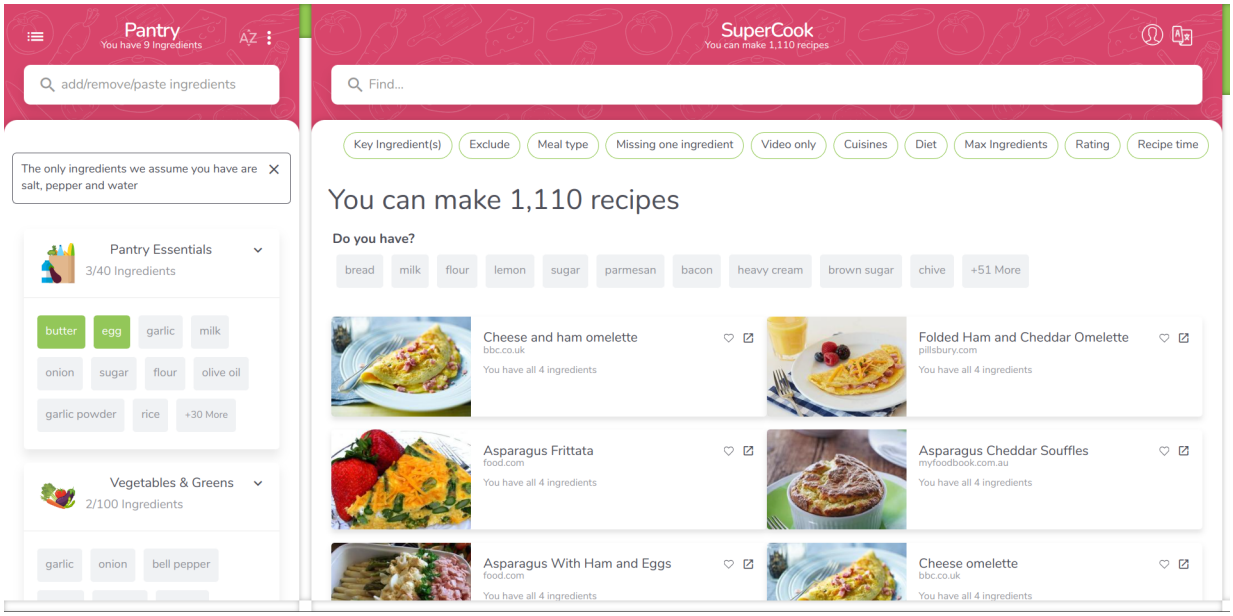

FIGURE 2.3. – Une recherche avec quelques ingrédients [2]

2.1.2. SuperCook

SuperCook est une application web permettant de trouver des recettes déjà existantes sur internet en fonction des ingrédients qu'on sélectionne. Les recettes disponibles ne sont donc que consultables sur d'autres sites internets vers lesquelles elles redirigent.

Les ingrédients avec lesquels la recherche est faite sont disponibles dans un side-pannel à gauche de le page. On peut entrer ceux que l'ont veut (une liste déroulante apparait) ou alors directement cliquer sur ceux à l'écran. Il est possible de suggérer de nouveaux ingrédients aux développeurs du site, mais pas de les inclure directement dans la recherche. Il n'est également pas possible de directement proposer de recettes manuellement sans s'inscrire, potentiellement car les recettes disponibles sont régulièrement mise à jour à partir de ce qu'il y a à disposition sur internet (avec un web scraper/crawler peut-être). Le nombre d'ingrédients manquants ou qui matchent sont indiqués sur la recette correspondante. Il est aussi proposé une grande quantité de manière de trier la pléthore de recettes disponibles, par exemple en fonction du type de repas (déjeuner, dîner, souper), de cuisine (grecque, française, asiatique, ...) ou du temps que la recette prend à concocter. Des ingrédients additionnels sont proposés pour augmenter le nombre de recherches qui matcheraient s'ils étaient sélectionnés.

Le site est assez simple d'utilisation et se caractérise par la grande quantité de possibilité de recettes à disposition, même avec très peu d'ingrédients choisis.

2.1.3. MyFridgeFood

Ce site internet est le seul ayant opté pour une sélection des ingrédients via une liste à cocher (fig. 2.4). Cette unique méthode de sélection offre un choix très limité, une re-

2. Website de SuperCook [3]

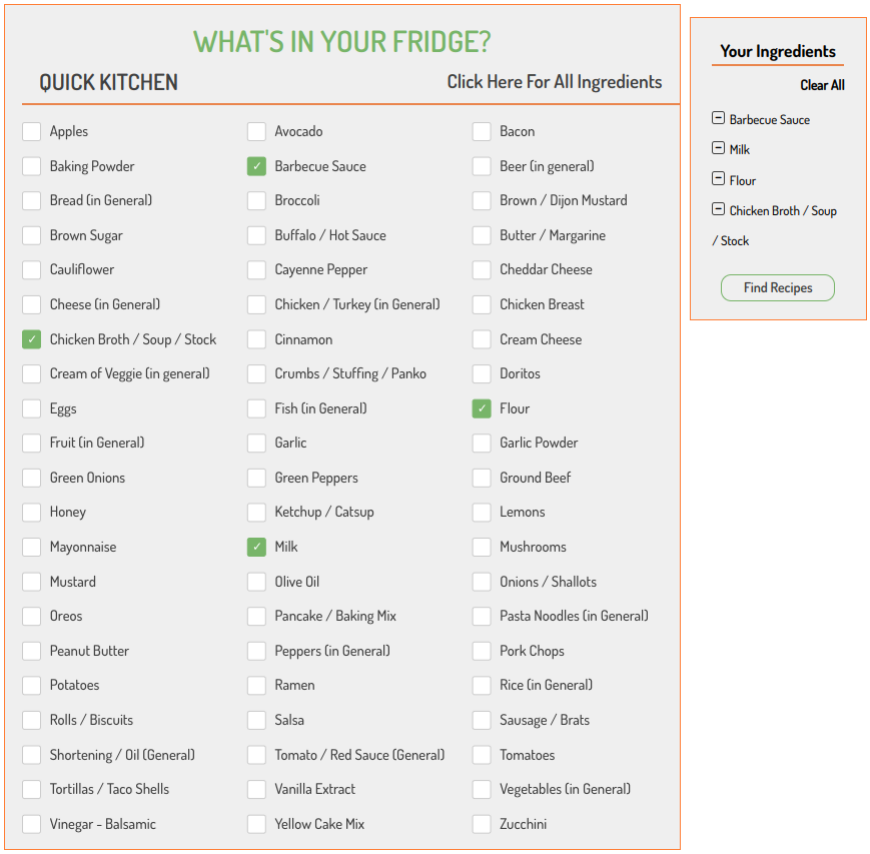

FIGURE 2.4. – Sélection des ingrédients [3]

cherche des ingrédients fastidieuse et une expérience peu efficace. L'absence d'alternative rend cet aspect plutôt maladroit. Il est possible de voir tous les ingrédients disponibles en cliquant sur "Click Here For All Ingredients" ce qui affiche une longue liste contenant les ingrédients séparés par catégories, mais ne rend pas la recherche d'un ingrédient spécifique plus simple.

Une fois les ingrédients choisis et la recherche effectuée, les résultats sont disponibles de manière claire et le temps que prend la recette ainsi que s'il manque des ingrédients ou non sont directement disponibles sur les cartes de résultat. Il y a aussi, sur la droite, quelques options de filtres à disposition, puisque le nombre de recettes proposées peut être assez facilement grand. Les recettes sont crées par des utilisateurs uniquement, et ils sont encouragés à envoyer une photo de leur plats cuisinés pour pouvoir les afficher sur le site pour la recette correspondante. Des commentaires propres à chaque recette sont égalements disponibles.

Le tout est simple et efficace, à l'exception de la méthode de recherche des ingrédients qui ne permet pas d'en trouver un précis efficacement.

3. Website de MyFridgeFood [2]

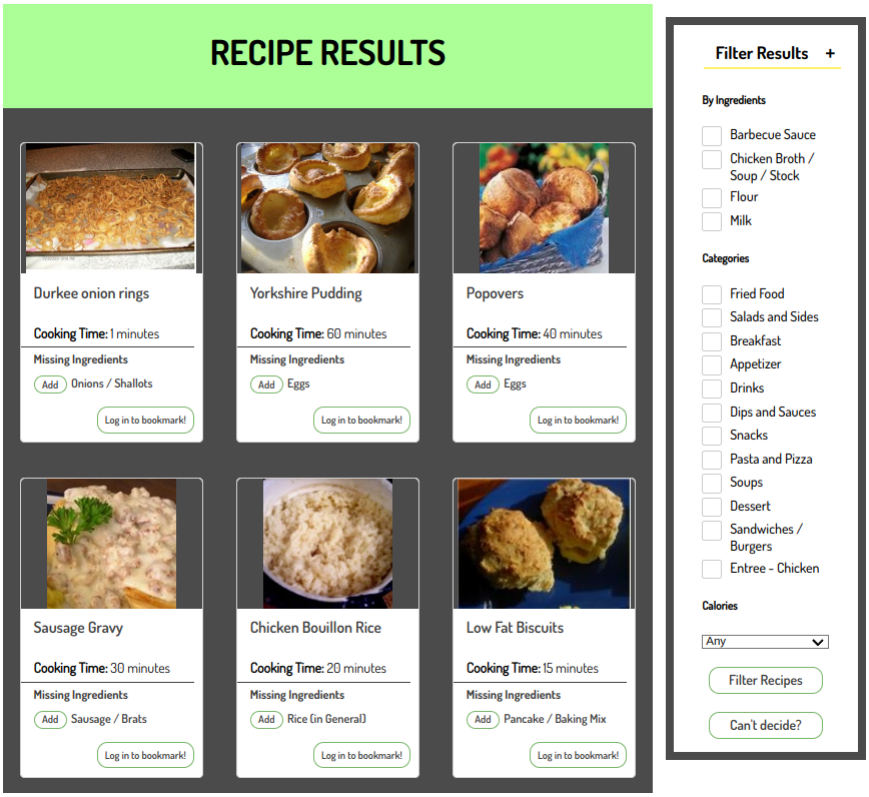

FIGURE 2.5. – Recettes résultantes

2.1.4. Tesco Recipe Finder Tool

Ce "recipe finder" proposé par Tesco fonctionne avec une recherche d'ingrédients par texte, qui propose directement une liste d'auto-complétion comme sur une majorité des applications modernes. Une fois 3 ingrédients sélectionnés, il est instantanément proposé plusieurs recettes contenant ces derniers. La liste complète des ingrédients nécéssaires est directement sous la carte de la recette, avec le temps requis pour la cuisiner. Ces ingrédients sont en gras quand ils font partis de ceux que l'utilisateur possède déjà et sont grisés quand ils sont manquants. Il est également possible de choisir un "must have" qui devra forcément se trouver dans les recettes proposées, et l'utilisateur peut choisir un ou plusieurs "tags" comme "végétarien" ou "sans produits laitiers" pour filtrer les recettes. Ces dernières sont directement proposées par Tesco, ont un "rating", contiennent énormément d'informations sur leur valeur nutritive et sont au sein de l'écosystème de leur marque, où il est possible de créer une "shopping list", commander les ingrédients chez eux, etc.

Ce design est très simple d'utilisation et très réactif puisque la recherche se fait automatiquement après l'ajout de chaque ingrédient. Ce programme sert également de valeur ajoutée à leur site internet, et veut combiner la planification des repas avec l'achat des ingrédients et leur confection, le tout directement dans leur écosystème.

4. Website de Tesco Real Food [4]

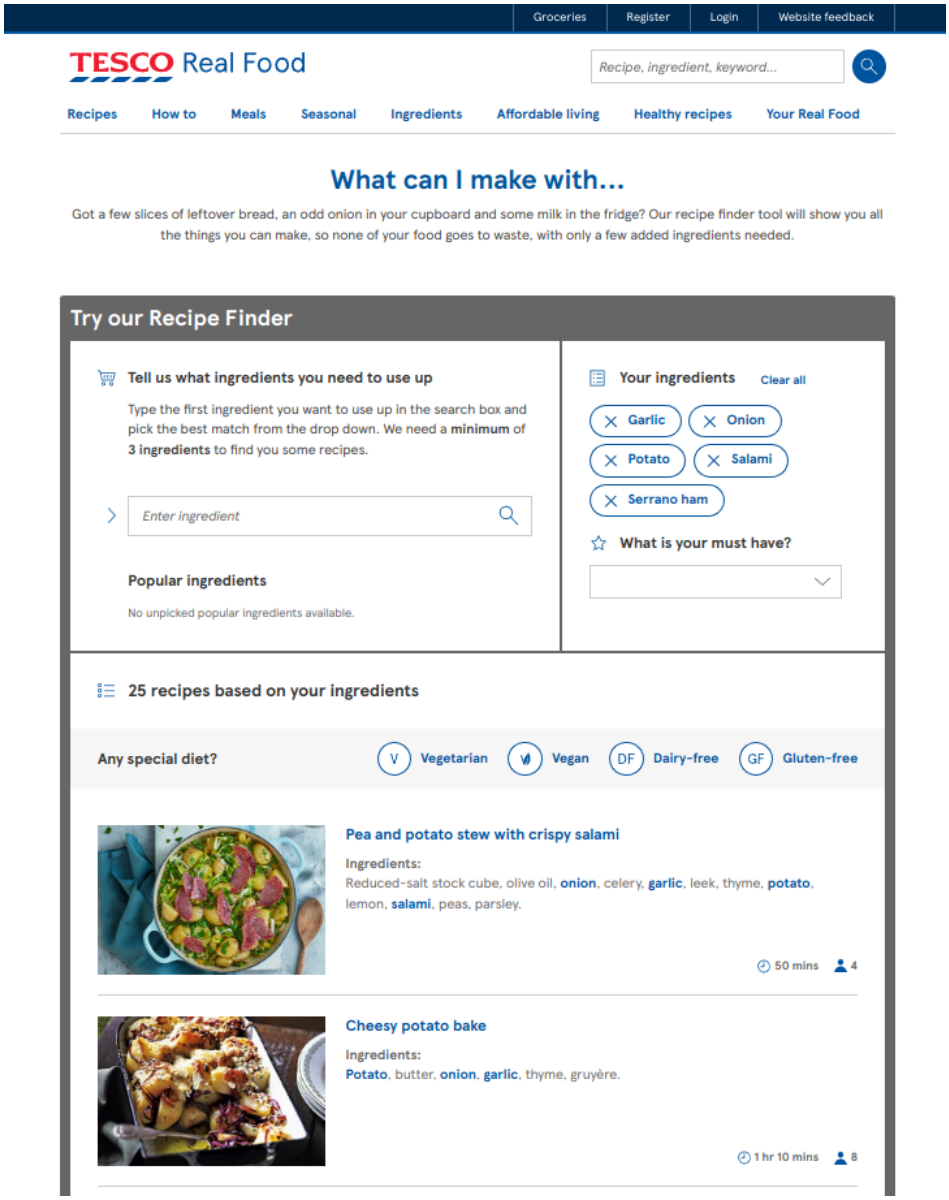

FIGURE 2.6. – Une recherche [4]

2.2. Comparaison

Les différents programmes présentés dans la section précédente ont plus ou moins le même but, mais ont souvent une approche très différente. Il est parfois possible de trouver des recettes qui sont propres au programme que l'on utilise, ou alors uniquement issues d'autres sites internets. D'autres optent pour une combinaison des deux dans le but d'avoir une diversité de contenu. Les méthodes de recherches ainsi que la présentation des résultats sont assez similaires, puisque la majorité propose la possibilité d'écrire directement un ingrédient recherché, et propose une liste déroulante avec auto-complétion pour sélectionner un ingrédient valide. Les résultats sont proposés dans des cartes où l'on peut déjà avoir une idée de quels ingrédients sont utilisés ou alors simplement s'il manque ou non des ingrédients à l'utilisateur pour qu'il puisse la réaliser.

EmptyMyFridge et le Recipe Finder de Tesco poussent le concept un peu plus loin en essayant de toucher à d'autres aspects de l'organisation des repas, comme les listes de course, ou même un tracking complet des ingrédients présents dans le domicile de l'utilisateur. Cette approche est plutôt destinée à des utilisateurs longs termes, qui prendront l'habitude d'utiliser ces programmes dans leur vie quotidienne. EmptyMyFridge rend particulièrement efficace les recherches journalières puisqu'il n'y aura pas besoin d'entrer une liste d'ingrédient à chaque fois, au prix de devoir garder un suivi rigoureux des ressources disponibles chez soi.

SuperCook et MyFridgeFood se veulent être des outils qui vont droit au but, qui est de trouver une idée de recette à cuisiner avec les ingrédients disponibles à un moment précis. Des utilisateurs irréguliers auront peut-être une meilleur expérience avec cette approche puisqu'elle ne demande que de faire une recherche après avoir entré les ingrédients voulus.

3
Précision des exigences

3.1. Fonctionnalités retenues

Dans ce chapitre, une idée générale des différentes fonctionnalités de l'application conçue est esquissée. Ces fonctionnailtés sont basées sur la compréhension des différentes approches prises par les applications présentées précédemment. Le comportement de l'application ainsi que les technologies utilisées sont ensuite présentés.

3.1.1. Recettes à disposition

Les recettes misent à dispositon sur le site pour les utilisateurs doivent être insérées d'une manière ou d'une autre. L'approche choisie consiste à laisser aux utilisateurs la possiblité d'ajouter leurs propres recettes à la base de données, de manière simple, pour qu'elles constituent le total des recettes parmis lesquels une recherche est possible. N'importe qui pourrait ajouter une recette, mais le droit de les modifier ou supprimer serait accordé seulement à des administrateurs, pour éviter qu'une personne mal intentionnée ne supprime toutes les entrées.
Il sera demandé aux utilisateurs voulant ajouter une recette de fournir un titre, une description (étapes à suivre, etc.) ainsi qu'une liste d'ingrédients nécéssaires à la préparation, avec leurs quantitées respectives.

3.1.2. Collecte des ingrédients voulus

L'utilisateur voulant effectuer une recherche devrait pouvoir définir les ingrédients voulus d'une manière instinctive, tout en étant assez "guidée" pour qu'il puisse sélectionner des ingrédients valides, c'est à dire qui existent bien dans des recettes. Une méthode rendant cela possible serait de laisser l'utilisateur écrire le nom d'un ingrédient et de lui proposer une auto-complétion sous forme de liste déroulante, contenant les ingrédients existants déjà dans des recettes. Autant d'ingrédients que voulus pourraient être ajoutés sans réelles limitations.

3.1.3. Méthode de recherche

Les ingrédients entrés par l'utilisateur voulant effectuer une recherche devront permettre de trouver une recette qui les contiennent plus ou moins tous. Deux approches sont possibles : proposer les recettes ayant au moins un des ingrédients recherchés en commun, ou seulement celles contenant tous les ingrédients recherchés. La première approche donnera beaucoup plus de résultats, surtout si le nombre d'ingrédients recherchés est grand, à l'inverse de la seconde approche où avec un grand nombre d'ingrédient, l'ensemble de recettes possibles se restreint. La première approche sera donc choisie, en mettant en avant les recettes ayant le plus d'ingrédients en commun avec ceux recherchés, puisqu'il est plus intéréssant de proposer à l'utilisateur des idées de recettes en assez grande quantité pour qu'il puisse lui même faire un choix plutot qu'une recette précise et parfaite qui sera, la plupart du temps, non-existante.

3.1.4. Présentation des résultats

Les recettes finalement proposées à l'utilisateur après sa recherche seront affichée dans des cartes contenant le nombre d'ingrédients correspondants à la recherches, ainsi que tous les ingrédients présents dans la recette. Cela permet de savoir directement si beaucoup d'ingrédients sont en commun ou non avec la recherche, et d'avoir une idée desquels sont utilisés dans la recette, ainsi que ceux manquants. Les recettes affichées en premier auront la meilleure correspondance avec la recherche d'ingrédients effectuée.

3.2. Approche

3.2.1. Fonctionnement général

L'application devrait être utilisable par n'importe quel utilisateur, indépendemment de son niveau de confort avec la technologie. Le but étant de proposer des recettes concevables avec des restes ou juste des idées inspirationnelles, le site devrait être simple et sans pages et boutons superflus. L'utilisateur aura directement les différentes possibilités d'utilisation du site sous les yeux sur la page d'accueil, sous la forme d'une barre de navigation tout en haut de la page.

Le flowchart suivant (figure 3.1) montre le déroulement général d'une utilisation standard de l'application. En bleu sont les pages et actions visibles par l'utilisateur, en vert

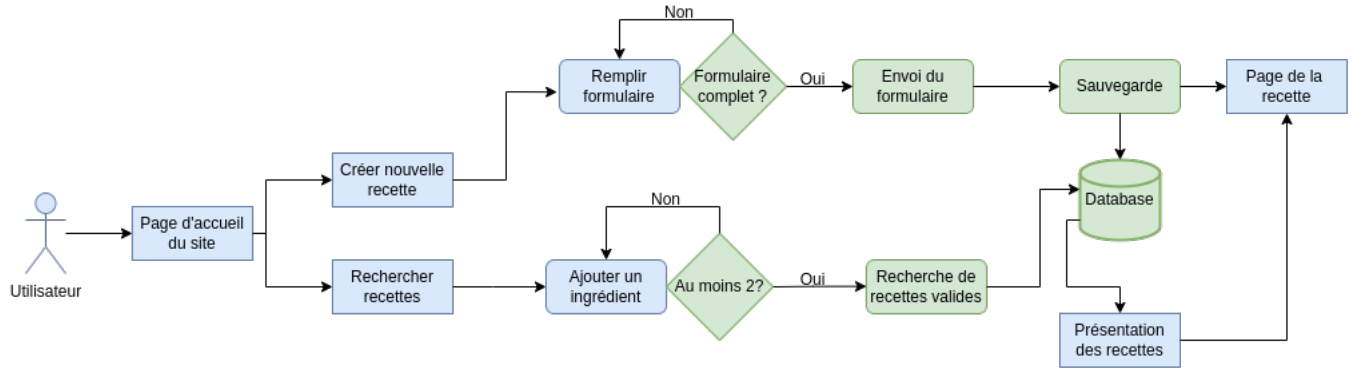

FIGURE 3.1. – Flowchart de l'application [1]

celles propres au serveur. Les boîtes avec des coins droits sont des pages, celles ayant des coins arrondis sont des actions.

3.2.2. Architecture : RESTful API

Pour réaliser ce programme, une pléthore de designs différents sont possibles. Dans le but d'exercer le concept et puisque les données entre le client et le serveur seront statiques, le choix a été de se baser sur une structure REST. Cette approche permettrait également d'accèder aux données par d'autres clients, puisque les appels effectués resteraient les mêmes, peu importe d'où ils seraient effectués.

Une "RESTful API" est une interface de programmation adhérant aux principes du "representational state transfer" qui consiste à standardiser les interactions entre les différentes utilisations d'un programme, en définissant un format de réponse fixe (ici JSON), permettant de simplifier les interactions entre différentes couches ou acteurs tels que différents clients ou des systèmes auto-gérés. Les données étant toujours formatées de la même manière et les communications étant indépendantes des états (stateless communication) il est possible de prévoir des systèmes réagissant à n'importe quel message de n'importe quel couche ou acteur sans une grande complexité. Cette base permet de créer différents clients pour accèder aux données de différentes manières, ou de créer des systèmes indépendants qui utilisent les données à disposition.

Un autre élément important est que les données sont reliées à leur URI (Uniform Resource Identifier) respective, permettant d'y accèder directement. Cela implique une structure cohérente entre les différentes pages web du site internet, pour que les ressources soient relayées adéquatement.

Pour donner un exemple concret, si cette application était déployée et utilisée en masse, grâce à cette structure d'autres applications l'utilisant pourraient être créées. Dans le cas d'un frigo connecté, une recherche automatique pourrait être faite en envoyant une requête directement au serveur sans passer par le site web (grâce à l'URI de la ressource voulue, ainsi qu'une méthode HTTP précise) pour proposer les recettes résultantes aux utilisateurs sans qu'ils aient besoin d'aller entrer les ingrédients à disposition manuelle-

1. Diagrame créé avec draw.io [5]
2. Diagrame trouvé sur LinkedIn [11]

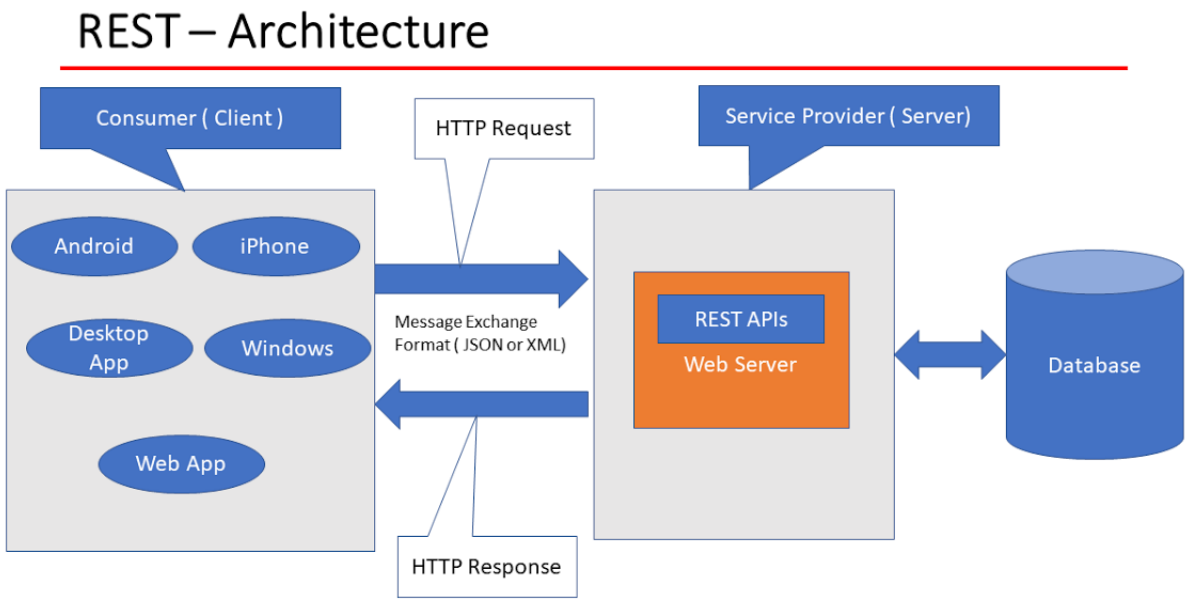

FIGURE 3.2. – Fonctionnement d'une API REST [2]

ment sur le site internet.

Pour résumer, cette architecture permet une ouverture aux évolutions possibles du programme en laissant d'autres systèmes accèder et utiliser les données, dans une structure standardisée et indépendante du client utilisé.

3.2.3. Technologies

Pour concrétiser cette application, une énorme quantité de languages, frameworks et autres technologies sont également disponibles gratuitement sur internet. Celles ayant été retenues ont toutes un rôle précis :

MongoDB

Il est évidemment nécéssaire d'avoir une base de données pour stocker les différentes recettes et ingrédients enregistrés par les utilisateurs. Le choix le plus standard serait une DB (DataBase) relationnelle fonctionnant avec des tables. Le fait est que seules des recettes avec les ingrédients respectifs doivent y être stockés et ayant toujours travaillé avec ce type de DB, l'envie d'essayer une alternative s'est avéré adaptée. Les bases de données noSQL (basées sur des documents) m'ont toujours intriguées et puisqu'il y aura que 2 tables en choisissant une DB relationnelle, il était intéressant de constater que selon la "Rule 1" d'un article [3] officel de mongoDB, il était favorable de mettre les ingrédients dans un array associé directement à la recette correspondante, plutôt que de créer un autre document (ou une deuxième table). Cette structure paraissant convenable à l'utilisation voulue, il m'est également possible de découvrir un autre paradigme de stockage

3. MongoDB Schema Design Best Practices [6]

de données. Un autre avantage est que le format de stockage dans les documents est également JSON, ce qui permettera d'utiliser directement les données avec les méthodes HTTP sans devoir les reformater.

Node.js

Node.js est un "runtime environment" pour JavaScript, permettant de faire tourner une application en continu et de l'utiliser en tant que serveur. N'ayant que peu exploré le language JavaScript, cette approche paraîssait interéssante à découvrir. De plus, grâce au "Node Package Manager" (NPM) des modules permettant le développement de programme sont directement mis à disposition dans cet écosystème. Des packages tels que Mongoose, un "object modeling tool" pour MongoDB donnant accès à des fonctions qui servent de lien facilitant les intéractions entre la base de données et le server (format JSON pour les deux entre autre), ainsi que Express.js un framework qui permet de créer rapidement une API pour les méthodes HTTP (entre autre) en définissant les actions et routes voulues sont un atout indéniable de travailler avec Node.js.

HTML/CSS/JavaScript/Bootstrap

HTML, CSS et JavaScript sont utilisés dans la quasi-totalité des sites web de nos jours, et Bootstrap est un "frontend toolkit" qui permet de rendre leur utilisation plus agréable pendant le design des pages web du site. Après avoir créé quelques pages web en utilisant seulement HTML et CSS, la connaissance d'un framework permettant d'ajuster les différents composants de la page de manière consistante et d'améliorer le rendu visuel m'a paru essentielle.
Ces différents outils seront la base du client web, de son design visuel et des interactions possibles entre l'utilisateur et le serveur.

4
Réalisation

Ce chapitre décrit la conception complète et chronologique de l'application, séparée entre le back-end et le front-end, ainsi que différents problèmes intéressants ayant étés rencontrés durant le travail.

4.1. Back-end

Le back-end consiste en toute la partie serveur du programme, c'est à dire ce à quoi va faire appel le client avec ses requêtes et ce qui devra calculer les réponses adéquates en donnant les données correspondantes. Il est composé d'une base de donnée, d'un "runtime environment", du code de la page web à afficher et de la définition des différentes méthodes HTTP avec lesquels des interactions sont possibles avec le client.

4.1.1. Configuration initiale

Tout d'abord, l'installation de MongoDB et la création de la base de données permettant le stockage des différentes recettes ont du être mises en place.

Une fois la base de données appelée "recipes" créée, il sera possible d'y insérer de nouvelles entrées depuis le serveur, grâce à Mongoose. Pour cela, il est nécéssaire d'installer Node.js ainsi que le Node Package Manager (NPM) pour pouvoir ajouter les différents packages voulus. L'ajout de ces package est très intuitif et se fait avec une simple commande.

```
npm install mongoose
```

Listing 4.1 – Utilisation du Node Package Manager

Les différents packages utilisés pour l'application sont :

- bootstrap
- ejs
- express
- express-mongo-sanitize
- md5
- method-override
- mongodb
- mongoose

Quand un package est installé, il est ajouté au fichier "package.json" et ses dépendances sont également ajoutées. L'utilité de chacun des packages sera expliquée en temps voulu. Un autre fichier "package-lock.json" ainsi qu'un dossier "node_modules" sont générés et servent respectivement à avoir un suivi de quels packages sont utilisés et à stocker le code nécéssaire au fonctionnement de ces derniers.

4.1.2. Structure principale

L'application a été séparée en trois dossiers ayant des fonctions différentes.

Models

Le dossier "models" contient les fichiers définissant les modèles des éléments que l'on veut insérer dans la base de donnée. Le "schema" des recettes "recipeSchema" contenu dans "recipes.js" sert de structure (ou "template") pour toutes les recettes qui vont être enregistrées. Il implique que chaque recette a plusieurs propriétés, soit le titre, des instructions, une date de création et une liste d'ingrédients. La date est automatiquement ajoutée alors que les autres éléments devront être définis. L'attribut "required" est présent pour empêcher des entrées avec des champs vides d'être enregistrées et chaque entrée a un ID unique lui étant attribué automatiquement.

```
const recipeSchema = new mongoose.Schema({

    title:{
        type: String,
        required: true
    },

    instructions:{
        type: String,
        required: true
    },

    dateOfCreation:{
        type: Date,
        required: true,
```

```
        default: Date.now
    },

    ingredients: [ingredientAndQuantitySchema]
})
```

Listing 4.2 – Définition de "recipeSchema"

Chaque ingrédient est lui-même défini pas un autre "schema", le définissant comme ayant un nom et une quantité. Cette décision a été prise pour faciliter la lecture et la compréhension de la structure.
Ces "schemas" pourront être utilisés avec Mongoose pour gérer l'ajout et la suppression d'éléments dans la base de donnée, ainsi que la recherche de différentes recettes en fonction de leurs propriétés, auxquelles il est possible d'accèder par les variables définies dans les "schemas".

```
{
    "title": "Fondue",

    "instructions":
    "1. Rub the Pot with garlic clove.
2. Cut bread into small squares.
3. Put cheese, starch, wine and lemon juice in pot and melt while stiring constantly.
4. Season with a bit of pepper and nutmeg.
5. Serve right away.
6. Stick bread squares on fondue fork one by one and stir them in the fondue.",

    "ingredients": [
                    {"name": "garlic clove", "quantity": "1"},
                    {"name": "white wine", "quantity": "4 dl"},
                    {"name": "Gruyère", "quantity": "400g"},
                    {"name": "Emmentaler", "quantity": "400g"},
                    {"name": "kirsh", "quantity": "1 shot"},
                    {"name": "lemon juice", "quantity": "1 tsp"},
                    {"name": "starch flour", "quantity": "1-2 tbsp"},
                    {"name": "white bread", "quantity": "1 loaf"},
                    {"name": "nutmeg", "quantity": "a pinch"}
                   ]
}
```

Listing 4.3 – Exemple du JSON d'une recette

Un autre modèle, "user.js" a été créé par la suite dans le but d'ajouter un administrateur au site web pouvant modifier ou supprimer des recettes. Il a été conçu dans l'optique de pouvoir ajouter d'autres utilisateurs qui seraient ou non également des admins. Chaque utilisateur est sauvegardé dans la base de donnée avec comme propriété un nom, un mot de passe et un booléen pour définir si cet utilisateur est un administrateur ou non. Pour l'instant il n'y a qu'un seul utilisateur et il est administrateur.

Routes

Ce dossier accueille les différentes "routes" par lesquels l'API communique avec le client. Ces routes sont en fait les méthodes HTTP d'un "router" défini avec le package Express.js. Grâce à cela, les méthodes HTTP peuvent être utilisées directement à travers des fonctions

JavaScript asynchrones, dans lesquels nous pouvons définir les actions requise dans chaque cas.

```
//show all recipes
router.get('/browse', async (req, res) => {
    try{

        const recipes = await Recipe.find().sort({dateOfCreation: 'desc'})
        res.render('browse.ejs', { recipes: recipes })

    } catch(err){
        res.status(500).json({ message: err.message })
    }
})


//add a recipe
router.post('/new', async (req, res) => {

    let recipe = new Recipe({
        title: req.body.title,
        ingredients: req.body.ingredients,
        instructions: req.body.instructions
    })

    try{

        let newRecipe = await recipe.save()
        res.status(201).redirect(`/recipes/recipe/${recipe.id}`)

    }catch(err){
        res.status(400).json({ message: err.message })
    }
})
//show one recipe ( using getRecipe() )
router.get('/recipe/:id', getRecipe, async (req, res) => {
    res.render('show.ejs', {recipe: res.recipe})
})


//delete a recipe ( using getRecipe() )
router.delete('/:id', getRecipe, async (req, res) => {
    try{
        await res.recipe.remove()
        const recipes = await Recipe.find().sort({dateOfCreation: 'desc'})
        res.status(200).render('adminpage.ejs', {recipes: recipes})

    } catch(err){
        res.status(500).json({message: err.message})
    }
})
```

Listing 4.4 – Quatre exemples de routes avec Express.js

Une route est définie en appelant le "router" qui est créé par une fonction d'Express.js, en choisissant une méthode HTTP et en spécifiant un URI. Les principales méthodes HTTP utilisées sont :

— GET

— POST
— PUT
— PATCH
— DELETE

Le serveur va attendre des requêtes entrantes, sur un certain URI avec une certaine méthode. Les actions qui devront être entreprises pour chaque requête sont spécifiées dans les routes définies avec Express.js.
Par exemple, un client envoyant une requête (dont les données pourront être accedées par la variable "req") GET sur l'URI "/recipes/browse" recevra une réponse (qui est contenue dans la variable "res") confectionnée par le serveur dans la route qui correspond. Dans cet exemple, le serveur utilisera la première route définie dans le Listing 4.4 où il ira chercher dans la base de donnée toutes les recettes s'y trouvant, en utilisant la méthode "find()" de Mongoose. Une fois toutes les recettes séléctionnées et enregistrées dans une variable, sa réponse "res" sera de renvoyer une page web définie dans le fichier "browse.ejs" après y avoir injecté la variable contenant les différentes recettes.

Dans le cas des deux dernières routes du Listing 4.4, une fonction intermédiaire "getRecipe()" est utilisée. Cette fonction préselectionne une recette à partir de son ID pour pouvoir y référer dans la route l'utilisant. Cela permet d'éviter du code dupliqué et redondant que certaines routes ont en commun et peuvent donc partager.
L'ajout d'une nouvelle recette se fait à travers la route définie par l'URI "/recipes/new" et la méthode POST. Depuis le site internet, il est directement vérifé que tous les champs soient remplis, mais si une requête est envoyée depuis un autre client (par exemple, le website reqbin.com [1]) cette vérification n'a pas lieu. C'est pourquoi il a été spécifié "required" dans les champs du "schema" des recettes, cela permet d'éviter des champs vides peu importe le client. L'action entreprise par le serveur sur cette route est tout simplement de créer une nouvelle recette ("new Recipe({...})" ligne 18) et de lui donner les valeurs reçues dans la requête (contenues dans la variable "req") en tant que valeurs de ses champs. Cette recette est finalement enregistrée dans la base de donnée grâce à la fonction "save()" de Mongoose, puis la réponse renvoyée au client est la page web de la recette qui vient d'être ajoutée.

Route de recherche

Le coeur de l'application se trouve dans la possiblité de trouver des recettes à cuisiner à partir d'ingrédients donnés. Cette fonctionnalité est implémentée dans la route suivante :

```
1  //search for a matching recipe
2  router.post('/home', async (req, res) => {
3
4      let ingredients = req.body.ingredients;
5
6      let matches = [];
7      let queryResponse;
8      try{
9          for (let i = 0; i < ingredients.length; i++) {
10             queryResponse = await Recipe.find({ "ingredients.name": ingredients[`${i
                   }`]});
```

1. Site utilisé pendant la conception [14]

```
11          matches.push(queryResponse);
12      }
13      matches = matches.flat(Infinity);
14
15      let result = {};
16      for(var i = 0; i < matches.length; ++i) {
17          if(!result[matches[i].title]) //counting the number occurences of each
                recipe, to know which ones contain more matching ingredients
18              result[matches[i].title] = 0;
19          ++result[matches[i].title];
20      }
21
22      result = Object.entries(result).sort((a,b) => b[1] - a[1]); //sorting results
            to diplay them from best to worse
23
24      let resultingRecipes = [];
25      for (let i = 0; i < result.length; i++){
26          const matchingRecipe = await Recipe.find({"title": result[i][0]})
27          resultingRecipes.push(matchingRecipe);
28      }
29      resultingRecipes = resultingRecipes.flat(Infinity);
30
31      res.render('searchResult.ejs', { recipes: resultingRecipes, NBmatchings: result
             })
32  }
33  catch(err){
34      console.log(err)
35  }
36 })
```

Listing 4.5 – La route effectuant la recherche de recettes à partir des ingrédients

Elle est composée de trois phases : après avoir récupéré les ingrédients envoyés par le client, via un POST sur l'URI "/recipes/home", une requête est faite à la base de donnée, par Mongoose, de retourner toutes les recettes contenant au moins un des ingrédients fournis. Cela s'effectue dans le "for loop" à la ligne 9 du Listing 4.5. Étant donné qu'il y a autant de requêtes passées à la DB que d'ingrédients fournis, il peut y avoir plusieurs fois la même recette dans le résultat contenu dans l'array "matches". Cela est utilisé dans la deuxième phase, le deuxième "for loop" à la ligne 16, où chaque occurence de chaque recette est comptabilisé et inscrit dans l'objet "result". Ce nombre pourra être affiché dans la carte de la recette associée, pour bien montrer combien d'ingrédients correspondent à ceux fournis par le client.
Finalement, après avoir ordré les titres de recettes dans l'objet "result" par ordre décroissant (pour pouvoir les afficher du meilleur "match" au moins bon) et en le transformant en array 2D (ligne 22), une dernière suite de requêtes sont faites à la DB dans le "for loop" à la ligne 25 pour avoir les recettes correspondantes aux ingrédients fournis, à partir des "title" définis plus tôt dans l'array 2D. Les variables "resultingRecipes", qui contient les recettes complètes et "result", qui contient le nombre d'occurences de chaque recettes, sont passées à la view "searchResult.ejs" pour pouvoir les afficher en tant que résultat de la recherche faite par le client.

Views

Ce dossier contient les différentes "views" qui composent le site internet. Chaque "view" peut être interprêtée comme une page web du site. Ce terme vient de l'utilisation d'EJS (Embedded JavaScript), installable en tant que package Node.js et utilisé pour pouvoir mettre du code JavaScript directement "embedded" dans le code HTML de la page web, ainsi que la possiblité de "render" une page web en tant que réponse d'une route (comme à la ligne 31 du Listing 4.5, "res.render(...)") en passant des valeurs inscrites dans des variables, pour pouvoir les utiliser directement dans la view.
La majorité des valeurs passées aux views sont les recettes elles-mêmes, contenant leurs informations respectives, ou alors par exemple les ingrédients pouvant être utilisés dans une recherche par l'utilisateur. Il est possible d'accèder à ces valeurs directement au sein du code HTML, en utilisant les balises "<% ... %>".
Par exemple, le code permettant d'afficher les différentes recettes existantes dans la DB sur la view "browse.ejs", est généré par :

```
1   <div class="mx-auto" style="width: 350px;">
2       <% for(var i=0; i<recipes.length; i++) { %>
3           <div class="card" style="margin: 30px;" >
4               <a href="recipe/<%= recipes[i].id %>" >
5               <h5 class="card-header" style="margin: 10px; display: flex;justify-
                    content: center; align-items: center;"> <%= recipes[i].title %> </h5>
6               </a>
7               <div class="card-subtitle text muted mb-2" style="margin-left: 10px;">
8                   <%= recipes[i].dateOfCreation.toLocaleDateString() %>
9               </div>
10              <div class="card-text" style="margin: 10px">
11                  <% for(var j=0; j<recipes[i].ingredients.length; j++) { %>
12                  <span class="badge badge-light">
13                      <%= recipes[i].ingredients[j].name %>
14                  </span>
15                  <% } %>
16              </div>
17          </div>
18      <% } %>
19  </div>
```

Listing 4.6 – Quatre exemples de routes avec Express.js

L'array "recipes" et son contenu ont été passés à la view au moment où la view a été affichée par la réponse de la route GET "/recipes/browse" (ligne 6 du Listing 4.4) et peut être utilisé dans les balises comme expliqué précédemment. Cet array permet de lister les différents titres et ingrédients grâces à des "for loops" qui permettent également de générer du code HTML quand il se trouve au sein de la boucle.

4.2. Front-end

La partie "front-end" du programme consiste en toutes parties avec lesquels l'utilisateur peu intéragir, dans notre cas, le site internet, son apparence et utilisation.
Le site web est constitué de code HTML, CSS et Javascript, pour afficher et rendre interactives les différentes pages. J'ai également exploré les possibilités qu'offre le "toolkit"

Bootstrap, qui prétent pouvoir rendre la gestion de l'affichage des éléments présents sur les pages web simples et rendre leur aspect estétique facilement modifiable.

4.2.1. Les différentes pages

Toutes les pages ont été liées par une barre de navigation restant présente sur le haut de la page. Ce choix a été mis en place dans le but de rendre la navigation explicite, constante et accessible facilement.
La page d'accueil se trouve à l'adresse "/recipes/home", où l'ont peut directement avoir accès à l'outil de recherche de recettes. Cela étant la fonctionnalité principale du site, il a paru plutôt évident de la mettre en avant, en laissant de côté une page d'accueil plus classique.
L'autre page principale du site est la page de création de nouvelles recettes, qui se trouve à l'adresse "/recipes/new".
Une page mettant à disposition une liste complète de toutes les recettes se trouvant dans la base de donnée se trouve à l'addresse "/recipes/browse". Elle peut être utilisée comme alternative à la recherche principale du site, ou simplement pour explorer quels genres de recettes sont déjà enregistrées.

La page d'accueil et la page de création de recette sont relativement similaires, puisqu'elles ont des besoins en commun : La possibilité d'entrer des noms d'ingrédients soutenues par des propositions obtenues à partir de ceux étants déjà présents dans la DB, et un nombre d'ingrédient qui peut varier. Pour le nombre d'ingrédients un bouton "add slot" et un bouton "remove slot" ont été créés. Ils ajoutent et suppriment des champs, respectivement, dynamiquement. L'utilisateur peut donc facilement gérer le nombre d'ingrédients nécéssaire à sa recherche ou création.

Le code JavaScript des deux boutons se trouve dans une balise "<script></script>" en fin de code HTML sur les deux pages correspondantes.

```
let i = 2;

addSlot = function(){
    let container = document.getElementById("ingredientsTab");
    let element = document.createElement("div");

    element.className = "form-group"
    element.id = `slot-${i}`;
    container.appendChild(element);

    let inputIngredient = document.createElement("input");
    inputIngredient.type = "text";
    inputIngredient.required = true;
    inputIngredient.className = "form-control"
    inputIngredient.placeholder = "Ingredient"
    inputIngredient.name = `ingredients[${i}]`

    element.appendChild(inputIngredient)

    inputIngredient.setAttribute('list', "ingredients-list")

    i++;
```

```
23 };
```

Listing 4.7 – Bouton "add slot"

Le bouton "add slot" dans le code HTML appel la fonction définie dans le Listing 4.7 une fois pressé par l'utilisateur. Cette fonction génère de nouveaux éléments HTML en leur donnant les attributs nécéssaires, tout en gardant un compteur pour le nombre de nouveaux champs créés. Ce compteur est utile pour connaître l'index exacte du dernier champ, dans le but de pouvoir le supprimer si l'utilisateur le veut. Cela est possible grâce à la fonction appelée (Listing 4.8) lors de l'utilisation du bouton "remove slot" :

```
1 removeSlot = function(){
2     let container = document.getElementById("ingredientsTab");
3     let element = container.lastElementChild;
4     if(i > 2){
5         container.removeChild(element);
6         i--;
7     }
8 };
```

Listing 4.8 – Bouton "remove slot"

Encore une fois, cette fonction accède directement à l'élément HTML (ligne 3) en question pour le supprimer (ligne 5), en vérifiant tout d'abord qu'il y a bien au moins deux champs disponibles, sous la forme du compteur n'étant pas plus petit que 2 (ligne 4). Finalement, le compteur est diminué s'il y a assez de champs disponibles.

4.2.2. Bootstrap

L'installation de Bootstrap s'est faite à travers NPM, ainsi qu'en ajoutant des liens "jquery" à la fin des documents HTML dans des balises <script>, et un lien Bootstrap dans un tag <link>. Son utilisation revient à une alternative à écrire du code CSS, puisqu'il est possible de modifier directement la structure et le design d'une page web en ajoutant des attributs directement dans les balises HTML. Par exemple, le bouton "add slot" mentionné plus tôt n'est en fait qu'un tag <li>, auquel ont été ajouté des mots clefs dans sa classe :

```
1 <li class="btn btn-success my-2 my-sm-0" href="#" onClick="addSlot()">add slot</li>
```

Listing 4.9 – Rendu visuel du bouton "add slot"

Le mot clef "btn btn-success" permet de directement donner l'allure du bouton vert final. Les autres éléments présents dans la classe servent à modifier son positionnement.
Un autre exemple est la barre de navigation présente sur toutes les pages :

```
1 <nav class="navbar navbar-light bg-light">
2     <li class="navbar-brand" href="#">Recipe Finder</li>
3     <a class="btn btn-outline-success my-2 my-sm-0" href="/recipes/home">Search Recipe
          </a>
4     <a class="btn btn-outline-success my-2 my-sm-0" href="/recipes/browse">Browse
          Recipes</a>
5     <a class="btn btn-outline-success my-2 my-sm-0" href="/recipes/new">Add Recipe</a>
```

```
    <a class="btn btn-outline-secondary my-2 my-sm-0" href="/recipes/adminlogin">Admin
        </a>
</nav>
```

Listing 4.10 – Barre de Navigation du site

Les différents mots clefs présents dans les attributs "class" permettent de donner un aspect spécifique instantanément à un élément de la page.

Ce "toolkit" peut réellement s'avérer utile et puissant quand le développeur le connait bien, puisqu'il donne la possibilité de mettre en place des pages webs facilement et d'avoir un rendu visuel très rapide. Cependant, son utilisation n'est pas extrêmement facile d'accès, au point où, si l'on est déjà bien à l'aise avec du CSS, cet outil n'est pas un réel avantage puisqu'il reste assez cryptique. Dans le cas d'un développeur qui apprendrait directement son utilisation en tant que premiers pas dans la création d'une page web, il peut s'avérer être un atoût efficace.

L'auto-complétion nécéssaire aux propositions d'ingrédients donnés à l'utilisateur, ainsi que la page d'administrateur sont abordées dans la section suivante.

4.3. Problèmes rencontrés

4.3.1. Auto-complétion des ingrédients

Cette méthode est utilisée pour proposer à l'utilisateur des ingrédients déjà présents dans la base de donnée, dans les champs "ingredient" dans le cas d'une recherche de recette et pour l'ajout d'une nouvelle recette :

```
//pre-select possible ingredients to add (that are already in database)
async function getPossibleIngredients(req, res, next){

    let queryReq = await Recipe.find().select("ingredients.name -_id"); //get only
        ingredients without id of recipe

    let medium = [];
    queryReq.forEach( //removing first layer of object
      function(o) {
         medium.push(o.ingredients);
      });

    medium = medium.flat(Infinity) //removing arrays

    let ingredients = [];
    medium.forEach( //removing second layer of object
      function(o) {
         ingredients.push(o.name);
      });

    possibleIngredients = [...new Set(ingredients)]; //removing duplicates

    res.possibleIngredients = possibleIngredients
    next()
```

```
24 }
```

Listing 4.11 – Quatre exemples de routes avec Express.js

Cette fonction est toujours executée à l'appel des routes GET "/recipes/home" et GET "/recipes/new" avant le code contenu dans la route, dans le but de pouvoir proposer des ingrédients possibles quand l'utilisateur séléctionne un champ "ingredient" grâce à la balise HTML "<datalist>".
La valeur retournée par la query faite avec Mongoose est un array d'objets avec une structure beaucoup trop complexe à utiliser tel quelle, il a donc été nécéssaire de le déconstruire pour ne garder que les informations qu'il contenait. Cela se fait à travers les deux "forEach" (ligne 7 et 15 du Listing 4.11) où les valeurs internes des objets sont extraites, rendant la structure plus simple, pour finalement arriver à un simple array de valeurs qui sont les ingrédients contenus dans les recettes.
Ce problème vient peut être d'une mauvaise utilisation de Mongoose de ma part, dû à une mauvaise compréhension de comment il faudrait utiliser, rechercher ou traiter les valeurs retournées.

4.3.2. Administrateur

Le but de la page administateur mentionnée plus tôt étant de permettre aux administrateurs du site de modifier ou supprimer des recettes, elle doit évidemment être protégée par un mot de passe. Cependant, de part le paradigme REST utilisé, mettre en place une session standard n'était pas envisageable.
La solution choisie consiste à mettre à disposition la page "adminpage.ejs" à l'adresse "/recipes/adminlogin", seulement après y avoir inséré l'utilisateur et le mot de passe nécéssaires, ceux-ci étants "admin" et "password" respectivement, tant que l'application web n'est pas mise en ligne sur internet. Avant cela, l'adresse renvoie toujours la page "login.ejs", qui demande les informations de l'administrateur.

```
1  //showing the login page (only for admins for now)
2  router.get('/adminlogin', async (req, res) => {
3      res.render('login.ejs')
4  })
5
6
7  //actual login
8  router.post('/adminlogin', async (req, res) => {
9
10     //try to login by comparing with database values
11     try{
12
13         let username = req.body.username;
14         let rawPassword = req.body.password;
15
16
17         let queryRes = await User.findOne({"username": username});
18
19         if(md5(rawPassword) == queryRes.password){
20             let recipes = await Recipe.find().sort({dateOfCreation: 'desc'})
21             res.status(200).render("adminpage.ejs", {recipes: recipes});
22         }
23         else{
```

```
24              res.status(403).json({message: "Wrong password."})
25          }
26
27      }catch(err){
28          res.status(404).json({ message: "No such user." })
29      }
30  })
31
32
33  //show admin edit page
34  router.get('/edit/:id', getRecipe, getPossibleIngredients, async (req, res) => {
35      try{
36          res.render('editRecipe.ejs', {recipe: res.recipe, possibleIngredients: res.
                possibleIngredients})
37      } catch(err){
38          res.status(500).json({ message: err.message })
39      }
40  })
```

Listing 4.12 – Les routes gérant la page administrateur

Une fois les données entrées, une autre "view" est affichée (ligne 21 du Listing 4.12) mais l'adresse reste inchangée. Ceci permet de contourner le problème de session en le transformant en un nouveau, moins dérangeant : si la page est rechargée, la route reprend depuis le départ et demande à nouveau à l'utilisateur de se connecter.

Sur la view "adminpage.ejs", la liste complète des recettes enregistrée est affichée, avec des boutons "edit" et "delete" pour chacune d'elles.
Le bouton "delete" envoie une requête DELETE sur l'addresse de référrant à la recette correspondante (grâce à son ID) pour qu'elle puisse être supprimée de la DB.
Le bouton "edit" affiche une autre view, "editRecipe.ejs", avec un design identique à celle permettant de créer de nouvelles recettes. La seule différence étant que les champs de la recette choisie sont déjà remplis, grâce à l'utilisation de EJS : la recette en question est simplement passée en argument lors du "rendering" de la page (ligne 36 du Listing 4.12), il est donc possible d'afficher toutes ses informations dans les "input fields" HTML correspondants, en les générant automatiquement avec une fonction JavaScript similaire à celle permettant d'ajouter des champs vides (Listing 4.13).

```
1  let oldIngredients = <%-JSON.stringify(recipe.ingredients)%>
2
3  autoAddSlots = function(){
4      if(oldIngredients.length > 2){
5          for(let a = 2; a < oldIngredients.length; a++){
6              let container = document.getElementById("ingredientsTab");
7              let element = document.createElement("div");
8
9              element.className = "form-group";
10             element.id = `slot-${i}`;
11             container.appendChild(element);
12
13             let inputIngredient = document.createElement("input");
14             inputIngredient.name = `ingredients[${i}][name]`
15             inputIngredient.type = "text";
16             inputIngredient.required = true;
17             inputIngredient.className = "form-control";
```

```
18            inputIngredient.placeholder = "Ingredient";
19            inputIngredient.value = oldIngredients[i].name;
20
21            let inputQuantity = document.createElement("input");
22            inputQuantity.name = `ingredients[${i}][quantity]`;
23            inputQuantity.type = "text";
24            inputQuantity.required = true;
25            inputQuantity.className = "form-control";
26            inputQuantity.placeholder = "Quantity";
27            inputQuantity.value= oldIngredients[i].quantity;
28
29            element.appendChild(inputIngredient);
30            element.appendChild(inputQuantity);
31
32            inputIngredient.setAttribute('list', "ingredients-list");
33
34            i++
35        }
36    }
37 }
38
39 autoAddSlots();
```

Listing 4.13 – Fonction générant et complétant automatiquement les champs contenant les ingrédients d'une recette

La variable ayant été passée depuis la route au "rendering" est récupérée et assignée à la ligne 1, puis utilisée pour vérifier le nombre d'ingrédients (et donc de champs) devant être ajoutés, ainsi que leurs valeurs respectives (noms et quantités).

Tout cela fonctionne bien à priori, mais malheureusement il reste un problème qui empêcherait la mise en ligne du site tel quel. En effet, la sécurité ammenée par cette page admin n'est qu'artificielle : il est tout à fait possible d'envoyer une requête directement au server, sans passer par le site (comme avec CURL ou Reqbin), directement à la bonne adresse si elle est connue. La requête sera effectuée sans problème, même si elle demande de supprimer ou modifier une recette, indépendamment du statut administrateur de l'expéditeur.
Il en va de même pour la modification des recettes, il est facile de remplacer le "recipe" se trouvant à l'adresse d'une recette "/recipes/recipe/63ef86d5483529c82ef82e35" par un "edit" pour se voir ammener directement sur la page de modification à laquelle seulement les adminitrateurs sont sensés avoir accès.

```
1  //delete a recipe
2  router.delete('/:id', getRecipe, async (req, res) => {
3      try{
4          await res.recipe.remove()
5          const recipes = await Recipe.find().sort({dateOfCreation: 'desc'})
6          res.status(200).render('adminpage.ejs', {recipes: recipes})
7
8      } catch(err){
9          res.status(500).json({message: err.message})
10     }
11 })
```

Listing 4.14 – La route gérant la suppression des recettes

Une solution à cela pourrait être d'ajouter un "middleware" faisant usage d'un Json Web Token, qui est une méthode courrament utilisée pour la sécurité et les vérifications d'accès aux ressources dans ce type d'applications.
Une autre solution, plus archaïque et beaucoup moins professionnelle, serait de simplement ajouter l'obligation de fournir le mot de passe directement avec la requête en question, avec une vérification de ce mot de passe avant la ligne 4 du Listing 4.14 par exemple.

5
Résultat

Ce chapitre présente l'application web finie et son fonctionnement, tel qu'ils ont été prévus. Enfin, la documentation de l'application est brèvement explorée, pour donner la possibilité de comprendre comment utiliser les routes et fonctions principales.

5.1. Présentation de l'application

Le site internet est composé de 8 "views" différentes qui sont retournées au client selon la requête HTTP faite et traitée par une des routes.
La page d'accueil du site (Figure 5.1) a été pensée pour être directement utile et simple plutôt qu'une page d'accueil plus classique. Elle contient directement le coeur de cette application web, à savoir la recherche de recettes à partir d'ingrédients donnés. L'utilisateur est invité à entrer les ingrédients qu'il a à disposition pour directement effectuer une recherche. Quand un champ est séléctionné, une liste déroulante apparait avec des propositions d'ingrédients et se met à jour au fur et à mesure que l'utilisateur écrit.

Le résultat (Figure 5.2) d'une recherche s'affiche directement sous forme d'une liste où les titres renvoie vers les pages respectives des différentes recettes. Un indicateur est présent ("matching ingredients"), pour montrer explicitement à l'utilisateur quelles recettes ont plus d'ingrédients en commun avec leur recherche, en plus du fait qu'elles apparaissent déjà dans l'ordre.

Si l'utilisateur clique sur une recette, puis retourne en arrière dans l'historique avec la flèche "retour" du navigateur, le résultat de la recherche est toujours disponible puisque la "view" elle-même contient les résultats.

La barre de navigation reste constamment présente en haut des pages, pour assurer une navigation simple à l'utilisateur. Elle lui permet d'accèder aux autres fonctionnalités, comme parcourir toutes les recettes en cliquant sur le bouton "Browse Recipes" (Figure 5.3).

Les recettes sont affichées selon leur date d'ajout, de la plus récente à la plus vieille. Il est évidemment possible d'ouvrir la page de chaque recette en cliquant sur leur titre.

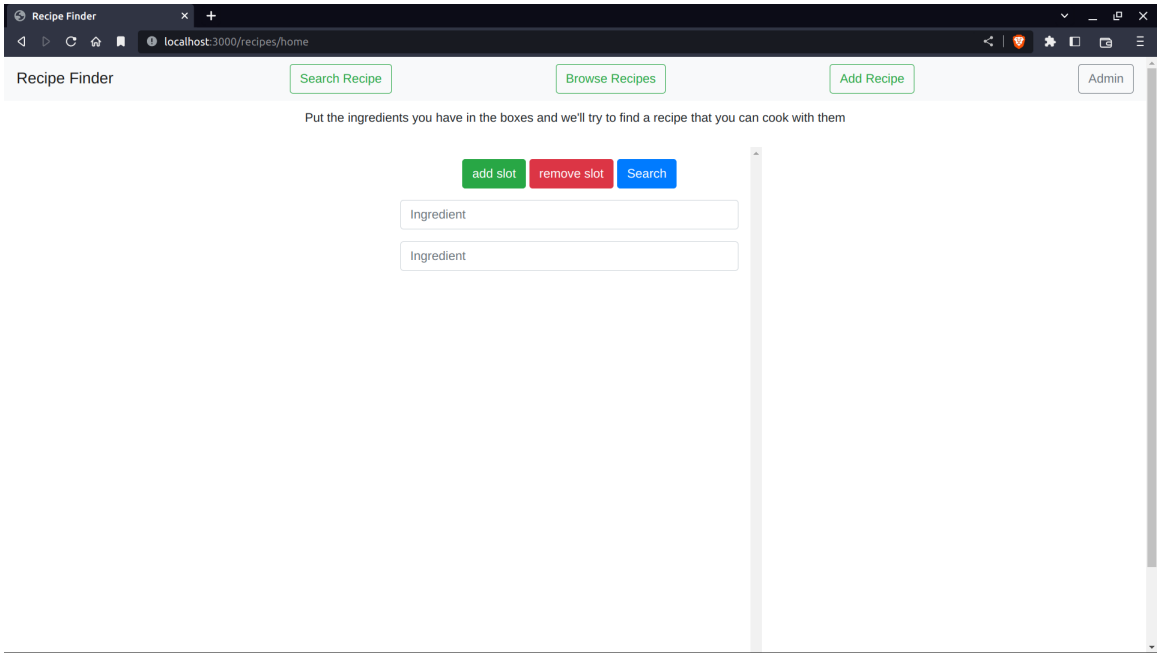

FIGURE 5.1. – Page d'accueil du site

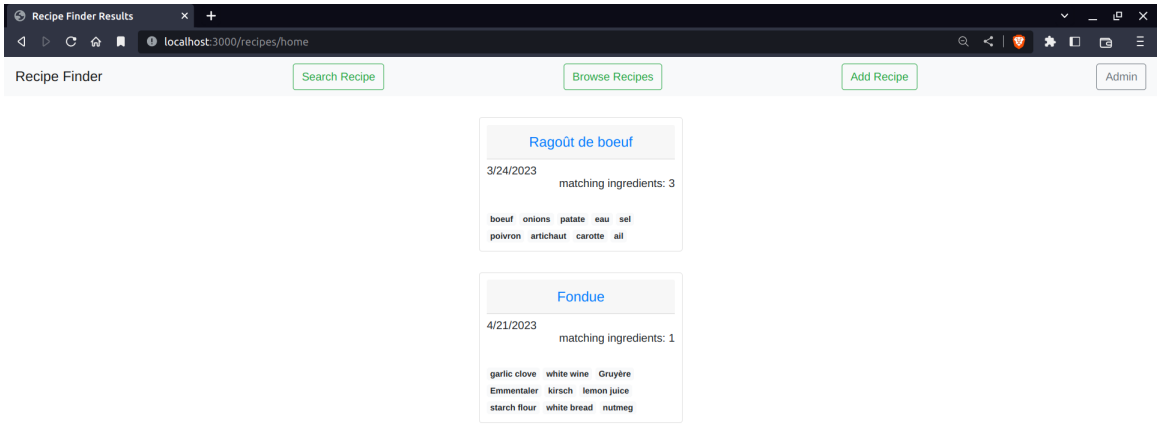

FIGURE 5.2. – Résultat d'une recherche

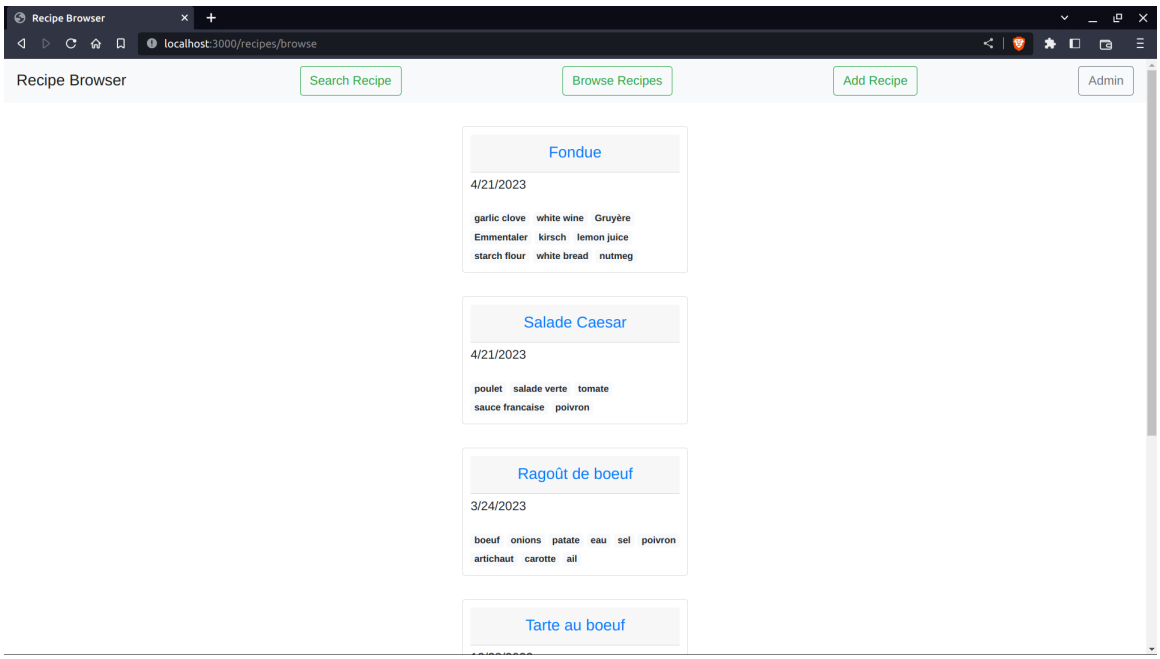

FIGURE 5.3. – Liste de toutes les recettes

La page d'ajout de recette (Figure 5.4) quant à elle, est renvoyée en cliquant sur le bouton "Add Recipe".

Là également, une fois un champ d'ingrédient séléctionné, une liste déroulante apparait avec des propositions. Le champ de texte pour les instructions ne force pas une structure et laisse les utilisateurs mettre en page leurs instructions pour une recette comme ils le souhaites, par exemple en numérotant les étapes ou en ajoutant des précisions ça et là.

Une fois une recette ajoutée, l'utilisateur est redirigé directement vers sa page (Figure 5.5).

Les ingrédients et leurs quantités sont mis en avant de manière claire, et les instructions sont présentes en dessous, en essayant de garder la structure mise en place par l'utilisateur (retours à la ligne, etc).

La page d'administrateur, renvoyée en cliquant sur le bouton "Admin", donne tout d'abord une "view" demandant un "username" et un mot de passe (Figure 5.6). Les utilisateurs normaux qui cliqueraient dessus devraient comprendre directement que ce "login" ne leur est pas destiné.

Une fois les informations entrées, une autre view prend la place du "login" et affiche toutes les recettes ainsi que des boutons pour les modifier et supprimer (Figure 5.7).

Un administrateur pourra donc supprimer une recette simplement en cliquant sur un bouton, ou la modifier, sur la page prévue à cet effet (Figure 5.8)

Cette page contient déjà les valeurs de la recette correspondante et permet de les modifier comme bon leur semble. La modification prendra effet instantanément et la date de création restera inchangée pour garder un suivi de quelles recettes ont été ajoutées en premier.

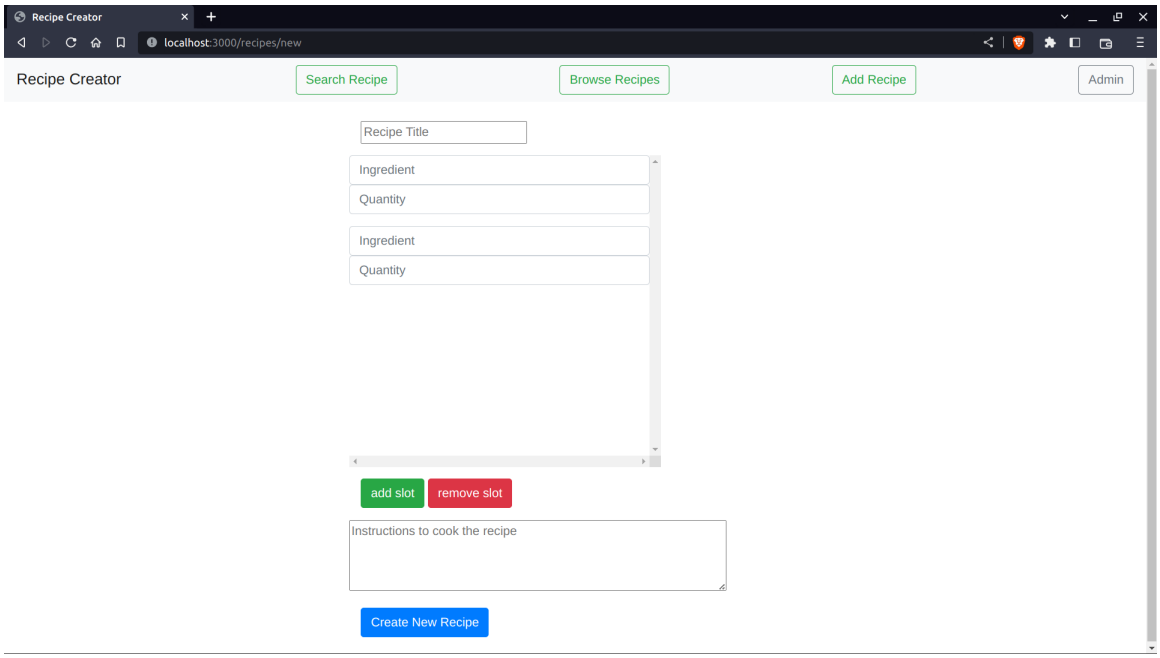

FIGURE 5.4. – Formulaire à remplir pour ajouter une recette

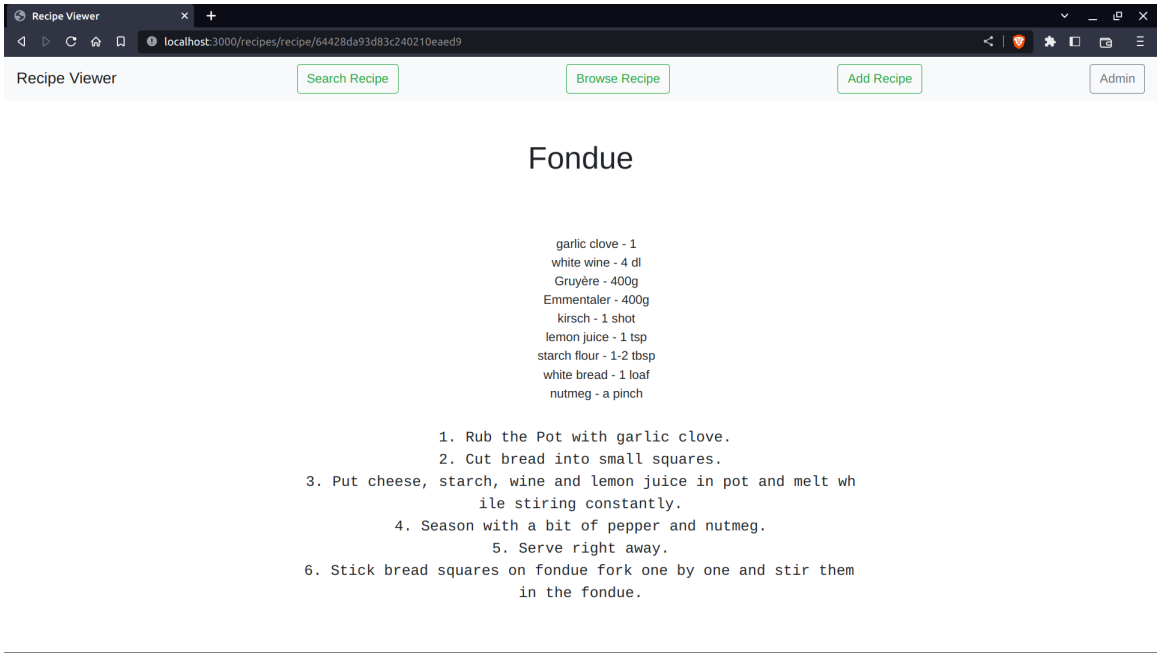

FIGURE 5.5. – Page d'une recette

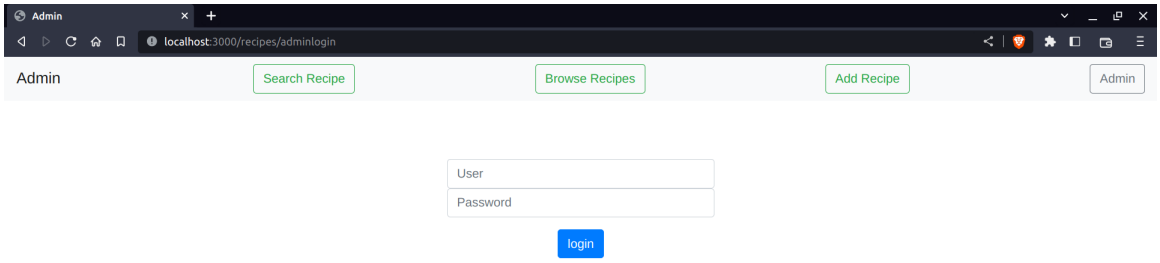

FIGURE 5.6. – Page du login pour les administrateurs

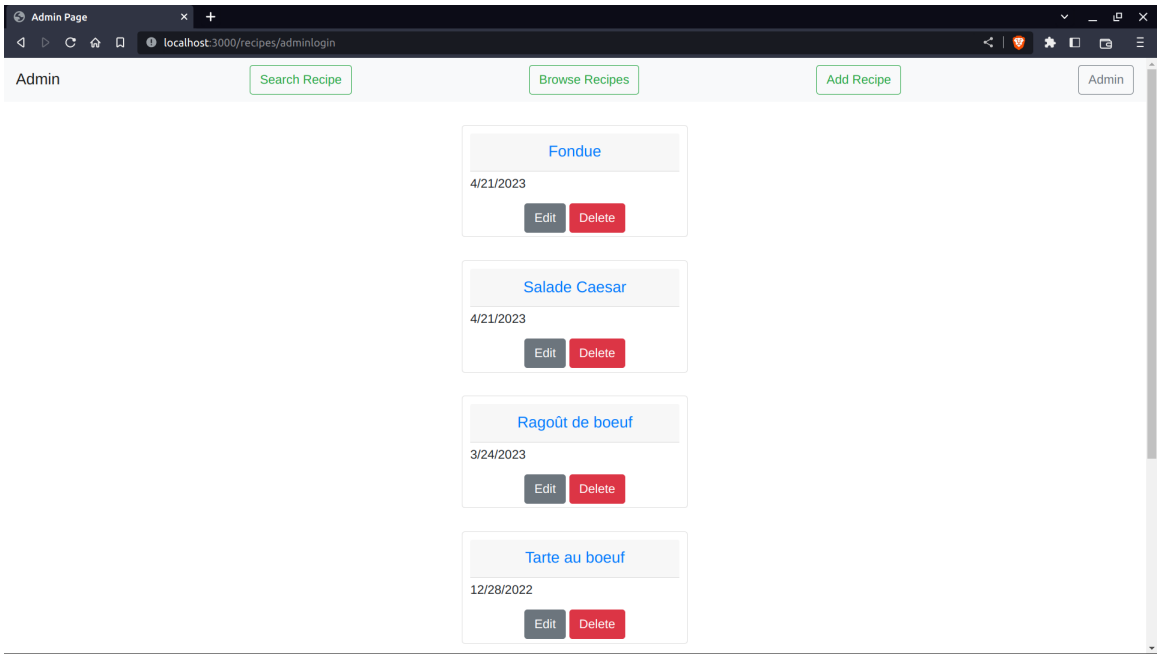

FIGURE 5.7. – Page permettant la modification et suppression des recettes

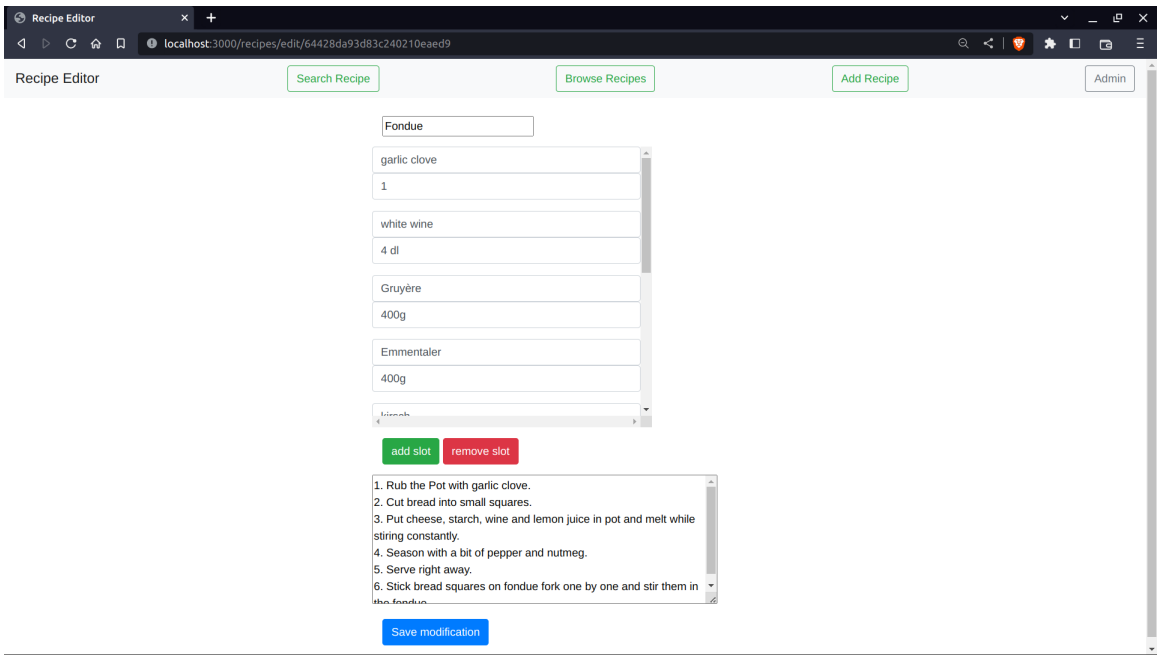

FIGURE 5.8. – Page pour modifier les entrées d'une recette

5.2. Documentation

Cette section contient la documentation des différentes fonctions et routes utilisées par le programme, se trouvant toutes dans le fichier "routes/recipes.js".

Nom	Type	Méthode HTTP	Description
recipes/browse	route	GET	Cette route renvoie une page contenant toutes les recettes présentes dans la DB.
recipes/recipe/ :id	route	GET	Cette route renvoie la page d'une recette, à partir de son ID.
recipes/ :id	route	DELETE	Cette route supprime une recette à partir de son ID. Ne devrait être utilisée que par un administrateur.
recipes/new	route	GET	Route renvoyant la page permettant d'enregistrer une nouvelle recette.
recipes/new	route	POST	Route créant une nouvelle recette et l'ajoutant à la DB. Elle attend un JSON avec les informations de la recette à ajouter.

recipes/home	route	GET	Route renvoyant la page d'accueil du site.
rrecipes/home	route	POST	Route effectuant la recherche d'une recette à partir d'ingrédients. Les noms des ingrédients sont attendus dans une liste dont le nom est "ingredients". Elle renvoie une page contenant les résultats
recipes/edit/ :id	route	GET	Route affichant la page de modification d'une recette destinée aux admins.
recipes/edit/ :id	route	POST	Permet aux admins de modifier une recette à partir de son ID. Est utilisée comme un PUT grâce au package "method-override". Elle attend un JSON complet de la recette en question et renvoie vers sa page.
recipes/newuser	route	POST	Permet de créer un nouvel utilisateur (fonctionnalité pas réellement implémentée) dont des administrateurs. Elle attend un JSON contenant un utilisateur et une key "permission" avec la valeur "granted". Elle retourne le JSON du nouvel utilisateur.
recipes/adminlogin	route	GET	Renvoie la page de login pour administrateur.
recipes/adminlogin	route	POST	Cette route gère une tentative de login pour admin. Elle attend un "username" et un "password" déjà présents dans la DB. Elle renvoie à la page destinée aux admins, si le login est validé.

getRecipe(req, res, next)	fonction	Cette fonction est utilisée comme middleware pour récuper une recette selon son ID. Elle est utilisée par plusieurs routes utilisant un ID dans leur adresse. Il suffit de l'ajouter en tant que paramètre de la route et de référer à la recette séléctionnée par "res.recipe".
getPossibleIngredients(req, res, next)	fonction	Cette fonction récupère tous les ingrédients existants dans la DB. Elle est utilisée par plusieurs routes proposant des suggestions d'ingrédients. Il suffit de l'ajouter en tant que paramètre à une route pour accèder aux ingrédients disponibles, se trouvant dans la variable "res.possibleIngredients".

6 Conclusion

6.1. Extension et améliorations

Le résultat du travail sur le site internet permet de partager et rechercher des recettes sur la platforme mise à disposition pour n'importe quel utilisateur. Mais l'idée sous-jacente au site était que sa conception respectant une architecture de API REST permette d'avoir une interface plus générale pour accèder aux données de manière différente que celle prévue pour les utilisateurs standards. Cette API n'a pas été mise en place aussi entièrement que prévu, certainement car le développement n'a pas été assez distinct entre le site et cette dernière.

Une réelle séparation pourrait être faite au niveau du serveur, où l'API offrirait la possibilité d'envoyer des requêtes depuis d'autres client sans faute, et où le site serait donc traité en tant que client standard qui utiliserait pleinement l'API comme les potentiels autres. Ce changement rendrait l'organisation et le fonctionnement du programme beaucoup plus structuré et clair, mais ajouterait aussi plus de polyvalente et une extension des possibilités de son utilisation.

Au niveau du site en lui même, si l'amélioration précédente était mise en place il serait assez simple de mettre en place des sessions et des comptes pour les utilisateurs et les administrateurs. Cela premetterait de régler le problème d'accès illégitime aux pages restreintes tout en ajoutant des fonctionnonalités de personalisation pour les utilisateurs. Ces ajouts sont monnaie courrante sur les sites internets de nos jours et il parait donc assez sensé de les mettre en place si le programme était mis en ligne sur internet.

Une autre petite amélioration pourrait être au niveau de la structure du code et de l'arbre de fichiers : séparer les routes gérant des parties différentes du site dans des fichiers indépendants, ainsi que séparer celles propres au site et celles propres aux données, à défaut de complètement séparer l'implémentation du site de la partie API. À ce niveau également, une sécurité avancée pourrait être mise en place, pour éviter des exploitations de bugs ou de design mal pensés.

6.2. Conclusion finale

Ce travail a permis la création d'une REST API mêlée à un site internet permettant de l'utiliser, dans le but de permette à des utilisateurs de créer et consulter des recettes de cuisine. La consultation des recettes peut se faire via une fonction de recherche utilisant les ingrédients déjà présents chez l'utilisateur pour afficher des recettes les utilisants, par ordre de pertinence.

Dans un premier temps, après avoir traité les objectifs de ce travail, les différentes applications déjà existantes ont été explorées en fonction de leurs fonctionnalités et approches. Ensuite, les différentes étapes menant à la conception du site internet donnant accès aux fonctionnalités mises en place sur le serveur ont été décrites pour enfin présenter le résultat final. Les différents problèmes rencontrés et améliorations possibles ont été abordés à la fin du travail.

Ce travail m'a permis de me familiariser avec des concepts et technologies que je n'avais jamais utilisés, particulièrement les REST APIs, MongoDB et Node.js. Ces connaissances sont omniprésentes dans le contexte d'internet et il est donc précieux d'avoir de l'expérience concrète avec.

Je n'avais également jamais travaillé sur un projet de cette taille, seul qui plus est. Il est très probable que je fasse face à des projet similaires dans le futur, je suis donc très satisfait d'avoir déjà acquis quelques connaissances générales, surtout au niveau de la supervision du programme dans son ensemble et de tous les aspects qu'il couvre.

Ce travail m'a beaucoup apporté et a été agréable tout au long de sa complétion, y compris les différents problèmes rencontrés qui constituent des étapes inévitables et nécéssaires à surmonter pour réellement gagner des compétences et les assimiler durablement.

A License of the Documentation

Copyright (c) 2023 Oliver Richani.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation ; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.

The GNU Free Documentation Licence can be read from [18].

Bibliographie

[1] Application pour recettes.
https://www.emptymyfridge.com/ (dernière consultation le 13 mars, 2023).

[2] Application pour recettes.
https://myfridgefood.com/ (dernière consultation le 13 mars, 2023).

[3] Application pour recettes.
https://www.supercook.com/#/desktop (dernière consultation le 13 mars, 2023).

[4] Application pour recettes.
https://realfood.tesco.com/what-can-i-make-with.html (dernière consultation le 13 mars, 2023).

[5] Outil pour créer des schémas.
https://www.draw.io/ (dernière consultation le 13 mars, 2023).

[6] MongoDB Schema Design Best Practices.
https://www.mongodb.com/developer/products/mongodb/mongodb-schema-design-best-practices/ (dernière consultation le 13 mars, 2023).

[7] Documentation BootStrap.
https://getbootstrap.com/docs/5.3/getting-started/introduction/ (dernière consultation le 13 mars, 2023).

[8] Documentation HTML.
https://developer.mozilla.org/en-US/docs/Web/HTML/Reference (dernière consultation le 13 mars, 2023).

[9] Exemples de désinfection de user input.
https://happycoding.io/tutorials/java-server/sanitizing-user-input (dernière consultation le 13 mars, 2023).

[10] Tutoriel pour utiliser BootStrap.
https://github.com/ishfulthinking/Creating-a-Website-with-Bootstrap (dernière consultation le 13 mars, 2023).

[11] Understanding REST architecture.
https://www.linkedin.com/pulse/understanding-rest-architecture-gabriel-gitonga (dernière consultation le 13 mars, 2023).

[12] Website utilisé comme inspiration.
https://github.com/ishfulthinking/Creating-a-Website-with-Bootstrap (dernière consultation le 13 mars, 2023).

[13] Website utilisé comme inspiration.
https://github.com/WebDevSimplified/Markdown-Blog (dernière consultation le 13 mars, 2023).

[14] Outil de testing pour RESTful API.
https://reqbin.com/ (dernière consultation le 13 mars, 2023).

[15] Documentation Express.js.
https://expressjs.com/en/4x/api.html (dernière consultation le 13 mars, 2023).

[16] Documentation Node.js.
https://nodejs.org/dist/latest-v18.x/docs/api/ (dernière consultation le 13 mars, 2023).

[17] Express.js and Node.js introduction.
https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/Introduction (dernière consultation le 13 mars, 2023).

[18] Free Documentation Licence (GNU FDL).
http://www.gnu.org/licenses/fdl.txt (dernière consultation le 13 mars, 2023).

[19] Repository de l'application.
https://github.com/Oli-Mopy/recipes-finder (dernière consultation le 22 mai, 2023).