DOCUMENT D'ARCHITECTURE TECHNIQUE

Informatique et Gestion

Polytech Montpellier

POLYCOOKER

Présenté par Lucas NOUGUIER
le 02 Avril 2022

Sous la direction de Christophe FIORIO

Devant le jury composé de

Christophe FIORIO
Arnaud CASTELTTORT

POLYTECH MONTPELLIER

UNIVERSITÉ DE MONTPELLIER

Contents

PARTIE I

Contexte

1 Objet

PolyCooker© est un site de cuisine développé lors d'un projet Web à Polytech Montpellier en réponse à un besoin vital d'optimisation du temps passé à faire ses courses alimentaires.

2 Acteurs

Il existe différents rôles sur PolyCooker© :

- Utilisateur non connecté : il peut naviguer sur la page d'accueil et afficher des recettes. Il n'a aucun droit et l'accès aux ressources protégées est masqué par le *front* (n'apparaît pas) et interdit par le *back* (bloquera les requêtes).
- Utilisateur connecté : il a accès aux mêmes fonctionnalités que l'invité, mais peut en plus créer des recettes, accéder à sa propre page et au calendrier. L'accès aux ressources administrateurs est masqué par le *front* et interdit par le *back*.
- Utilisateur administrateur : il a les mêmes droits que l'utilisateur connecté avec en plus la possibilité de créer d'autres administrateurs et, plus tard, d'accéder à un paneau d'administration où il pourra supprimer des recettes et comptes d'autres utilisateurs.

3 Contraintes métiers

- *Scalabilité* : PolyCooker© a besoin d'être *scalable* pour supporter la montée en charge.
- Sauvergarde : Comme il s'agit d'un *cloud* de recettes, faire des sauvergardes régulièrement serait avisé car une perte majeur de données ferait perdre confiance en le service.
- Sécurité : Il s'agit d'une plateforme collaborative où chacun peu ajouter ses propres recettes (et ingrédients s'ils venaient à manquer). Il faut donc s'assurer que les personnes ont les bonnes permissions. De plus les utilisateurs pourraient être amenés à entrer des données plus personnelles selon les services rendus (numéro de téléphone...), il faut ainsi éviter qu'un utilisateur sans droit puisse accéder à la base de données et ainsi la modifier hors du cadre de l'API.
- Disponibilité : Le site aura besoin d'être actif surtout lors des périodes de repas et le week-end, quand on prévoit ses courses. Si le site est hors services en France à 15h par exemple, l'impact sur les utilisateurs sera minime.
- Portabilité : Dans la mesure où le site est une application web en JS, elle sera portable sur n'importe quel système d'exploitation. Mais il est probable qu'il soit beaucoup utilisé sur téléphone, donc le site doit s'adapter aux petits écrans.

4 Fonctionnalités

4.1 Communes

On y retrouve les principales fonctionnalités des sites de cuisine à savoir :

- Création & suppression de comptes utilisateurs
- Création & suppression de recettes
- Recherches de recettes selon les critères suivants :
  - Nom d'autheur
  - Nom de recette
  - Ingrédients
  - Saisonnalité
  - Type (entrée, dessert...)
  - Régime (omnivore...)

4.2 Spécifiques

PolyCooker© comporte d'autre fonctionnalités moins répandues qui sont :

- Création d'ingrédients dans le cas où ils n'existent pas.
- Intégration d'un calendrier (actuellement non persistant) permettant de visualiser rapidement les recettes à effectuer dans la(les) semaine(s) à venir.
- Établissement d'une liste de course en fonction des recettes entrées dans le calendrier et du nombre de personnes renseignées. Impression en *.png* disponible.

PARTIE II

Architecture

1 Rôle des tiers

1.1 Front-end

Le *front-end* va gérer tout ce qui est lié à l'affichage des pages : affichage conditionnel, affichage des résultats, formatage des données... L'architecture est la suivante :

- */src/assets* : fichiers statiques (images...)
- */src/components* : fichiers **.vue** où se trouve des parties de page. Tous à l'exception de Header.vue et Footer.vue sont appelés dans une des pages définies dans */src/views*
- */src/config* : fichier de configuration qui regroupe divers paramètres. Actuellement il ne comporte que les *end-points* de l'API
- */src/services* : ficher faisant le lien entre les actions de connexion, inscription et modification des données d'un utilisateur du front et le back
- */src/store* : fichier permettant de manipuler le stockage local de vue (Vuex)
- */src/views* : fichiers **.vue** regroupant plusieurs components (situés dans */src/components*) pour afficher le contenu principal de la page
- */src/App.vue* : fichier principal représentant la page
- */src/main.js* : fichier permettant de créer l'application VueJs

1.2 Back-end

Le *back-end* va lui, récupérer les informations que lui aura envoyé le *front-end* et, en fonction des validations d'identité, renverra les données demandées. L'architecture est la suivante :

- */config* : fichier de configuration. Actuellement il ne comporte que l'URL depuis laquelle il accepte les requêtes
- */middleware* : fichiers contenant les diverses fonctions intermédiaires à exécuter lors des requêtes
- */routes* : contient les fichiers avec toutes les routes et les actions à faire en fonction de la requête et de la ressource demandée. Les routes et actions sont rangées par ressource.
- */services* : fichiers d'aides. Ne contient actuellement qu'une méthode de vérification des json web token.
- */index.js* : creation de l'application Node.js
- */db.js* : connexion à la base de données (utilise regex et variables d'environnement)

1.3 Base de données PostgreSQL

Effectuera les requêtes demandées

2 Architecture de déploiement

2.1 Diagramme

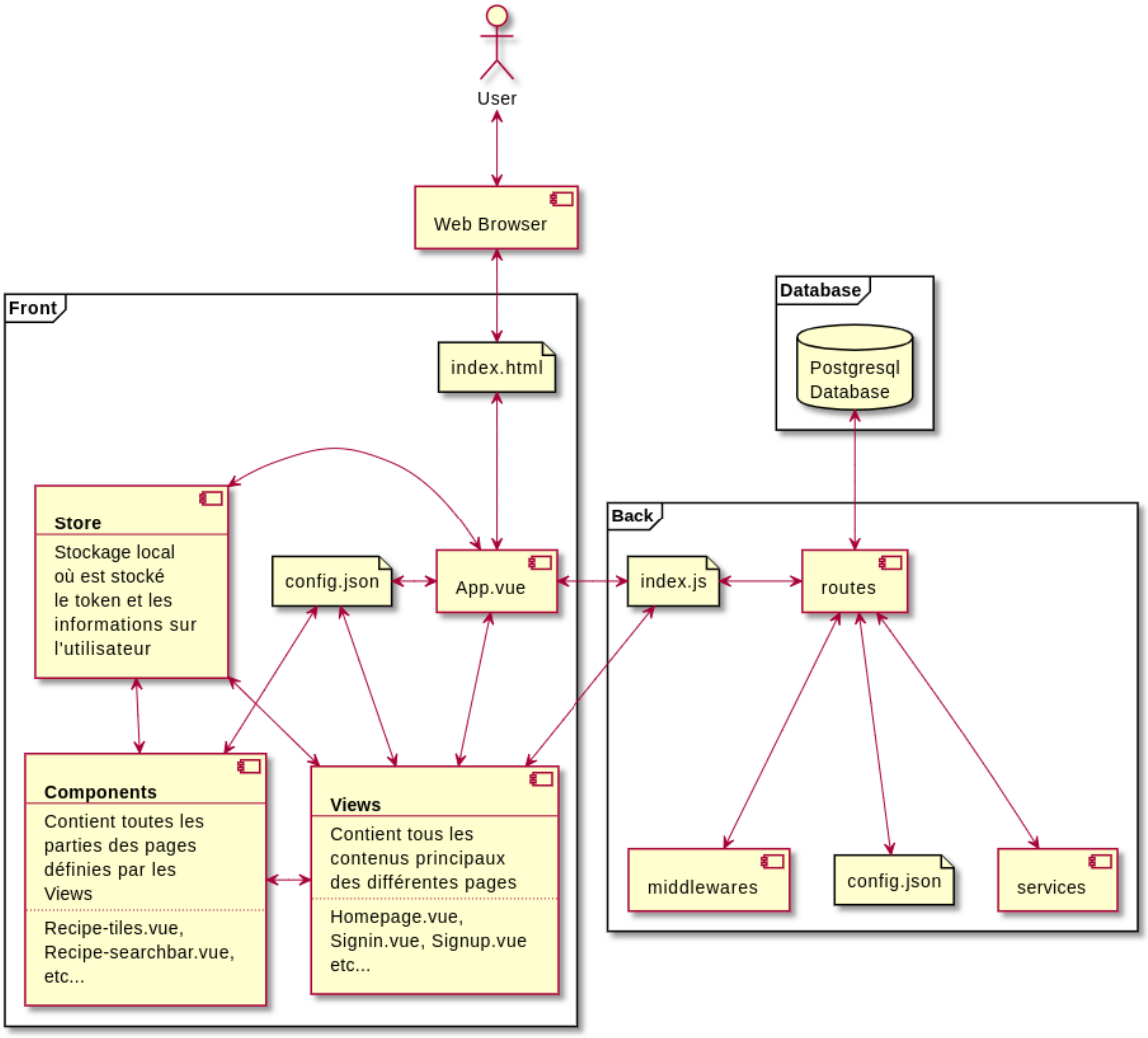


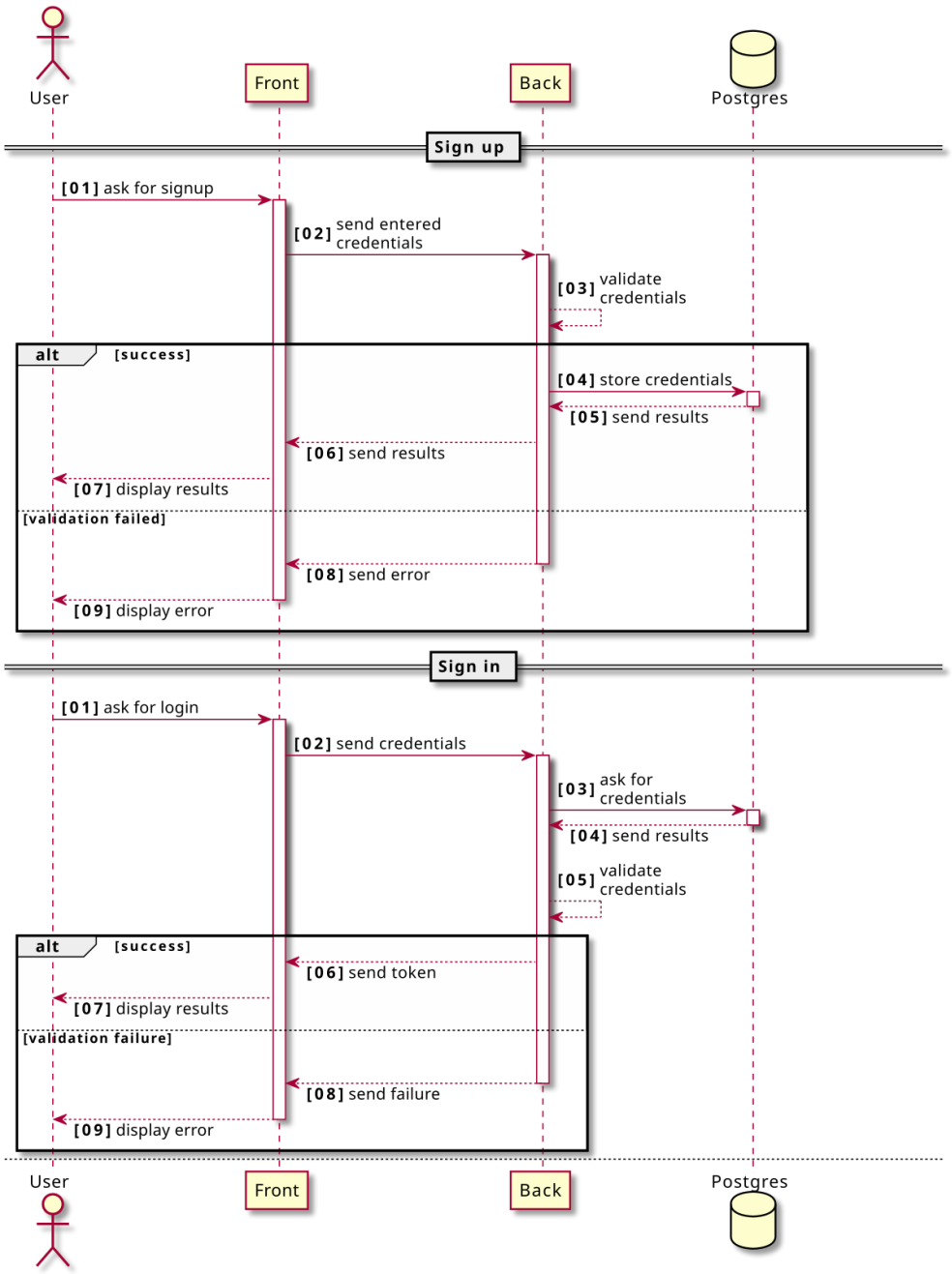
Figure 1: *Architecture du projet*

2.2 Déploiement

La plateforme de déploiement est Dokku, un PaaS relativement léger. Le déploiement a été fait sur 2 applications distinctes :

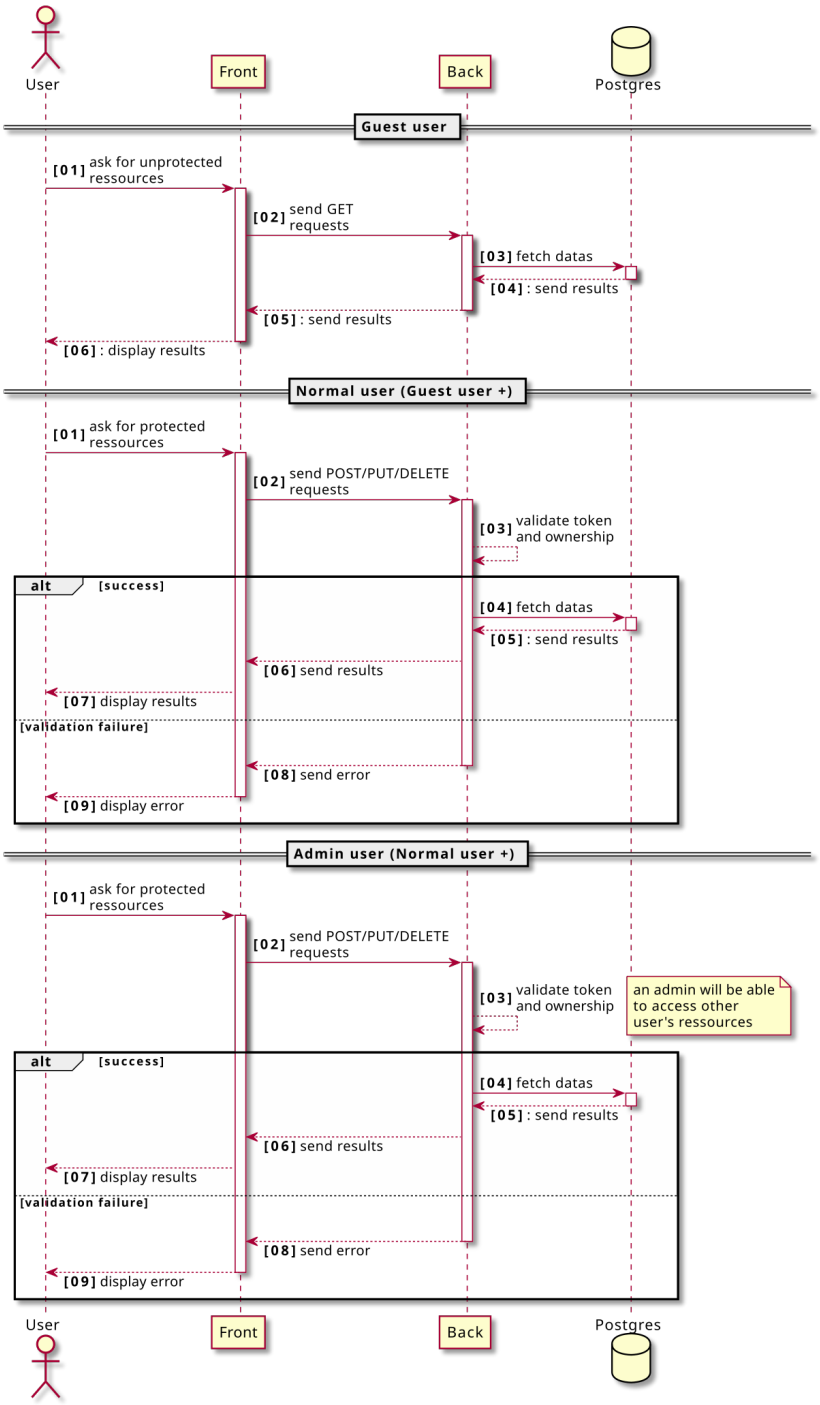
- polycookerapi : API de PolyCooker
- polycooker-cli : *Front-end* de PolyCooker

3 Flux d'information



(a) *Par type de connexion*

Figure 2: *Description des flux de données en fonction des cas d'utilisations*



(b) *Par utilisateur*

Figure 2: *Description des flux de données en fonction des cas d'utilisations*

PARTIE III

Détails des choix et explications

1 VueJs

1.1 Raison principale

J'ai choisi d'utiliser VueJs car il s'agissait d'une technologie qu'on a déjà vue lors du projet d'introduction au WEB. Pour un projet d'un peu moins de 3 semaines, j'ai préféré utiliser ce que je connaissais déjà plutôt que de me lancer dans une nouvelle technologie, pour éviter de perdre trop de temps lors de la prise en main. Si le projet avait été plus long, j'aurai probablement regardé un peu plus les autres possibilités.

1.2 Réactivité

J'ai beaucoup apprécié la réactivité de Vue et la facilité avec laquelle on pouvait créer des éléments dans le DOM (via le *v-for* notamment). Comme je savais que j'allais avoir un affichage par tuile qui correspondrait au résultat d'une requête, cela m'a conforté dans mon choix.

1.3 Évènement

Il était extrêmement facile de gérer les différents évènements (clics de boutons, changement/chargement de page...) et grâce au stockage local de Vue (Vuex), adapter le contenu de la page en fonction de l'utilisateur était très simple (en faisant néanmoins les vérifications d'identité dans le *back*)

1.4 Router

Le router de Vue (Vue Router) permet de facilement changer un seul composant sans avoir à recharger toute la page, ce qui me fut utile pour éviter d'avoir à regarder les composants *Header* & *Footer*

2 NodeJs & Express

Ne souhaitant pas faire de PHP, j'ai donc fait tout mon site en Javascript. J'ai alors utilisé NodeJs pour la création du serveur. Pour gérer l'API j'ai choisi ExpressJs, un framework open-source permettant de gérer facilement les routes et d'analyser automatiquement les requêtes HTTP (ce qui m'aura permis d'éviter de recoder tout le routage).

3 Contraintes métiers

3.1 Scalabilité

PolyCooker© utilise le cache pour les fonctionnalités d'autocomplétion, qui à terme représenteront de grosses requêtes (notamment lors de la création de recette)

Comme les requêtes deviendront très lourdes au fur et à mesure que les utilisateurs vont rajouter des recettes et ingrédients, une limite (actuellement statique) a été fixée à 25 résultats par requête. À l'avenir cette limite pourrait évoluer mais devra rester relativement faible du fait de la quantité d'information à envoyer (dont les images...)

Pour récupérer les styles et scripts de Materialize, PolyCooker© passe par les CDNs, permettant ainsi un temps de chargement plus faible et une meilleure expérience utilisateur.

3.2 Sécurité

L'application utilise des Json Web Tokens pour la vérification d'identité. Pour les requêtes GET il n'y a pas de vérification car pas d'altération de la base de données, et ceci permet aux visiteurs de faire le tour du site avant d'éventuellement s'inscrire.

En revanche, dès qu'une methode autre que GET est lancé (POST, PUT, DELETE), le token est vérifié dans le *back* avant d'effectuer la requête sur PostgreSQL. Donc un utilisateur non connecté ne pourra pas créer de recettes / ingrédients et un utilisateur non administrateur ne pourra pas accéder à la fonctionnalité de création d'administrateurs ou de suppression de comptes (autre que le sien).

Tous les mots de passe sont stockés encryptés et ne quitte pas la base de données, même sous forme de json web token.

De plus, pour chaque requête SQL paramétrée, un pré-formattage a lieu pour prévenir les insertions.

3.3 Disponibilités

Le site est déployé sur Dokku, plateforme qui le maintiendra actif et qui peut prendre en charge les sauvegardes automatiques de la base de données. Donc les contraintes de disponibilités peuvent être respectée.

PARTIE IV

Autres

1 Risques

L'application utilise des json web tokens et pour toute action entraînant une modification de la base de donnée, le token de l'utilisateur est vérifié pour s'assurer de son identité. Mais si le token est volé, alors une personne malveillante pourrait avoir accès à la base de données. Pour limiter les risques, les tokens ont actuellement une durée de validité de 6h.

2 Perspectives

2.1 Expérience utilisateur

- Possibiliter d'ajouter des images.
- Associer à des utiliateurs des paramètres par défault pour les recherches (un végétarien ne voudra jamais prendre de viande, il est donc inutile de lui en proposer).
- Regrouper en haut des étapes de la recettes des contenus compressibles, où il y aurait les détails de certaines étapes plus techniques suceptibles de ne pas être connues du grand public, en fonction des mots-clés présents dans les étapes.

2.2 Fonctionnalités

- Un calendrier persistant aligné sur un calendrier réel.
- Un système de notation avec commentaire des recettes.
- Une plateforme type réseau social où le thème serait exclusivement culinaire.
- Une détermination automatique de la saison associé à la recette grâce à la saison des ingrédients contenus.
- Des catégories alimentaires (fruits, légumes, viandes…) et des recherches par catégories avec une détermination automatique du régime d'une recette en fonction des catégories présentes.
- La réplication du Nutri-Score grâce à des informations détaillées sur les ingrédients (pourcentage de sucre, graisse, vitamines…) avec la possibilité de proposer des plats en fonction de ceux déjà mis dans le calendrier pour avoir un régime équilibré.

2.3 Économiques

- Revendre des données anonymisé sur ce qui est le plus consommé selon la localisation.