DOCUMENT D'ARCHITECTURE TECHNIQUE

Informatique et Gestion

Polytech Montpellier

POLYCOOKER

Présentée par Lucas NOUGUIER
le 02 Avril 2022

Sous la direction de Christophe FIORIO

Devant le jury composé de

Christophe Fiorio	[Statut jury]
Arnaud Castelltort	[Statut jury]

POLYTECH
MONTPELLIER

Contents

Part I

Technologies utilisées

CHAPTER 1

VueJs

1.1 Raison principale

J'ai choisi d'utiliser VueJs car il s'agissait d'une technologie qui avait déjà été vue lors du projet d'introduction au WEB. Pour un projet d'un peu moins de 3 semaines j'ai donc préféré exploiter ce que je connaissais déjà plutôt que de me lancer dans une nouvelle techno que je ne connaissais pas 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, j'ai directement opté pour VueJs.

1.3 Évènement

De même, 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

De plus, 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*

CHAPTER 2

NodeJs & Express

Ne souhaitant pas faire de PHP, j'ai donc fait tout mon site avec du JS. J'ai alors utilisé NodeJs en combinaison avec ExpressJs

Part II

Architecture

CHAPTER 3

Arborescence

3.1 Source

J'ai choisi de séparer les sources de mon API et du front pour faciliter la lisibilité du projet et n'avoir que les dépendances requises par l'API ou le front.

3.2 Front-end

J'ai mis l'ensemble de mes sources dans le même fichier */src*. A l'intérieur les fichiers ont été séparés par fonctionnalité :

- */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 *views*
- */src/config* : fichier de configuration qui regroupe divers paramètres. Actuellement il ne comporte que les *end-points* de l'API
- */src/materialize* : fichier regroupant l'ensemble des CSS et JS nécessaire au framework Materialize pour correctement afficher les éléments
- */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 *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

3.3 Back-end

La plupart des fichiers sont stockés à la racine dans divers dossiers :

- */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 ressources demandée. Les routes et actions sont rangées par ressources.
- */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* : fichier temporaire pour avec les informations de connexion à la base de données

CHAPTER 4

API