Rapport de projet de JEE

par Jordan Baudin, Corentin Le Guen et Geoffrey Spaur

28 janvier 2018

Contents

1 Présentation

Ce rapport à pour but d'apporter des réponses concernant le déroulement du projet, ainsi que sur le résultat final et nos apprentissages tout au long du projet.

2 Implémentation de l'authentification

Le guide Spring

Tout d'abord, il semblait plus faisable et raisonnable d'utiliser le guide Spring sur l'authentification pour écrire sur le sujet, mais le guide a montré uniquement de simples exemples, mais aucune véritable indication de systèmes fonctionnelles, avec enregistrement d'utilisateur et connexion, sans jamais montrer de gestion de session/token. Uniquement un utilisateur rentré en dur dans le code avec un rôle.

Notre Implémentation

Le choix a donc été de gérer les utilisateurs, rôles et tokens nous-mêmes. On ajoute donc un utilisateur lors d'une requête d'enregistrement (ou d'ajout par un administrateur, voir paragraphe sur la partie Administration), on vérifie si l'email est au bon format, si le mot de passe l'est aussi (a minima 8 caractères, une lettre minuscule, une lettre majuscule et un chiffre). Ensuite, on l'enregistre avec le rôle "USER".

Lorsqu'une personne a un compte, il peut se connecter, il recevra alors un ID token, qu'il devra donner pour chacune des requêtes qu'il fera, pour vérifier qu'il est effectivement enregistré et connecté. Lorsque le cas est echéant, on vérifiera qu'il a le rôle nécessaire pour la requête faite.

Pour modifier le rôle d'un utilisateur, il sera nécessaire de passer par l'Administration, donc seul un "ADMIN" pourra modifier ce rôle.

La gestion du token se fait dans un module statique qui stocke les tokens valables en cours, si jamais le serveur tombe, tous les tokens deviennent caduques, afin d'éviter une attaque de vol de token. Le but ici est de privilégier la re-connexion de l'utilisateur à un token qui serait valable pendant 10 ans. On invalide le token lors d'une déconnexion.

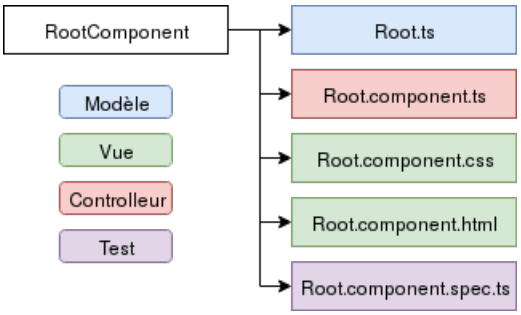
3 Implémentation du Front

Choix de la technologie

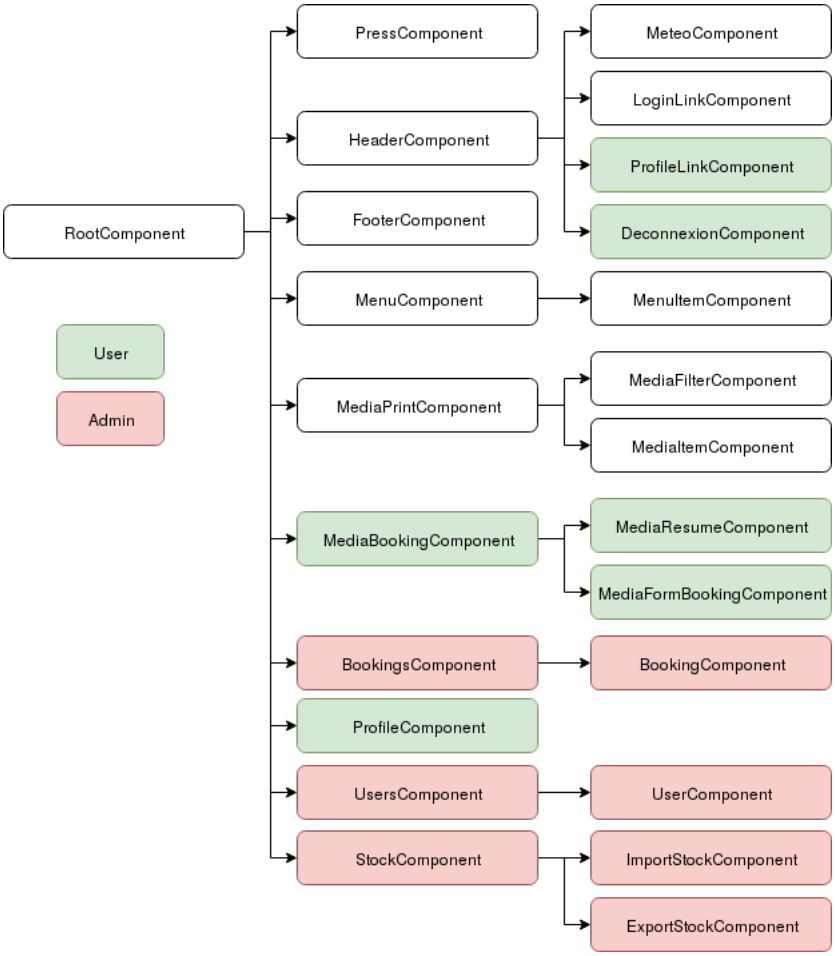
Nous avons choisi d'utiliser Angular4. Nous avons exclu Vue.js du choix des technologies car ce dernier est orienté MVVM (Modèle - Vue - Vue - Modèle), ce qui nous paraissait inapproprié au projet demandé. De plus le framework Angular4 est recommandé dans le travail en groupe car c'est un framework orienté objet.

Architecture en components

Le framework Angular4 permet d'assembler un ensemble de composants entre eux. Ce framework permet de développer une application: *single page application*. Chaque composant Angular4 est composé de plusieurs fichiers permettant en architecture MVC:



Nous avons donc décomposé notre application avec ces composants:



Guide technique

Angular4 permet de créer des applications qui seront hébergées sur un serveur node.js. Nous devons donc dans un premier temps installer ce serveur ainsi que le gestionnaire de package lié à node.js: npm. Vous trouverez les liens de téléchargement ici. Si vous êtes sur un système Linux, il est déconseillé d'utiliser les repos officiels. En effet ces derniers ne sont pas à jour, vous ne pourrez donc pas créer de projet angular4. Téléchargez donc les binaires afin, de les inclure dans votre .bashrc. Puis vous devrez installer Angular CLI afin de pouvoir créer, modifier ou lancer votre projet:

```
$ npm install -g @angular/cli
```

Vous pouvez donc maintenant créer un nouveau projet Angular avec la commande suivante:

```
$ ng new frontend
```

Ainsi vous pourrez rajouter à loisir des composants à votre projet:

```
$ ng generate component MenuComponent
```

Pour pouvoir lancer votre application, lancez la commande:

```
$ ng serve
```

Vous pourrez ensuite parcourir votre application à l'adresse: http://localhost:4200

Difficultés techniques rencontrées

Plusieurs difficultés techniques ont été rencontré durant le développement de la partie front.

Tout d'abord l'installation d'Angular4 a posé pas mal de problème sur Debian. Le package disponible n'est pas à jour. Il a donc fallu télécharger la bonne version sur internet comme précisé plus haut.

Certains modules d'Angular4 sont dépréciés, particulièrement le module permettant d'effectuer des requêtes Ajax. En effet nombre de tutoriels sur internet sont vieux et non mises à jour. Après consultation de la documentation, nous avons utilisé le module *HttpClient* et non *Http*.

Ayant séparé le front du back, les requêtes Ajax permettant au front de contacter le back étaient bloquées par le navigateur pour des raisons de sécurité. En effet, sauf contre indication du serveur, il est interdit d'effectuer une requête Ajax vers un serveur ayant un autre domaine que le front. Nous avons donc dû configurer le serveur afin d'autoriser ces requêtes.

4 Question Bonus

REST ou SOAP ?

Concernant le choix entre REST et SOAP, il est important déjà de rappeler que REST est une architecture, tandis que SOAP est un protocole.

Ici, le projet effectue du REST en utilisant du HTML. Le REST est très orienté Application à Machine ou Homme à Machine, où on va requêter au serveur sans forcément connaître le modèle qui est derrière, les interactions sont minimales et implique des petits objets simple à manipuler. Le REST est vraiment idéal pour coordonner une API HTML ou un CGI d'application.

Le SOAP lui est vraiment fait pour les interactions Machine à Machine ou Application à Application, où les objets sont envoyés de manière totalement spécifiées, avec un contrat typé fortement, grâce au WSDL (là où le REST fonctionne avec un WADL), où la structure est spécifiée et à respecter sous un format XML.