Université François Rabelais
Ecole d'ingénieur Polytechnique de l'Université de Tours
Département Informatique Industrielle par alternance

Projet Sécurité DII5

Rapport - Tests & Sécurité

Quentin ARCICAULT & Brandon SIMON-VERMOT

31 Janvier 2019

Sommaire

Présentation du projet

Dans le cadre du cours de sécurité que nous avons en cinquième année, de notre formation en école d'ingénieurs, à Polytech'Tours. Nous avons la possibilité d'utiliser des technologies que nous ne connaissons pas forcément dans le but de pouvoir acquérir de nouvelles connaissances, mettre en place des méthodes de tests et proposer un système sécurisé.

Ce projet a pour but de mettre à disposition un site web ou une application permettant de gérer un catalogue de produit, une liste de compte utilisateur, un panier et enfin un système de paiement.

Choix technologique

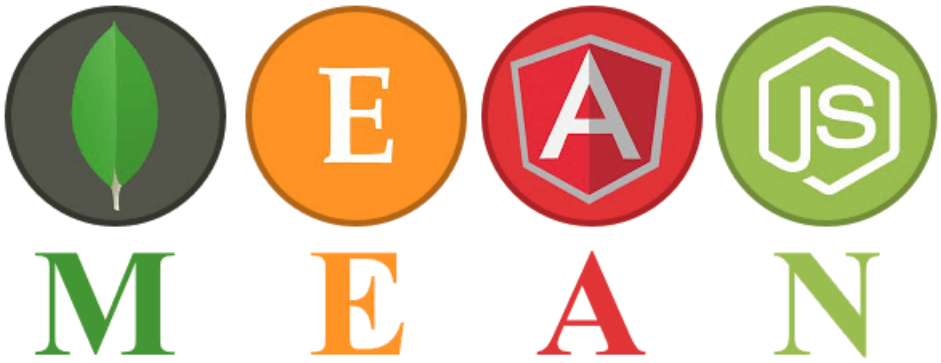
Nous avons choisi de partir, au départ, sur les choix technologiques suivants :

- Angular pour la partie front-end (ce qui sera affiché à l'utilisateur),
- Spring concernant la partie back-end (là où les données seront traitées),
- MongoDB pour le stockage de nos données, ainsi que les données utilisateurs.

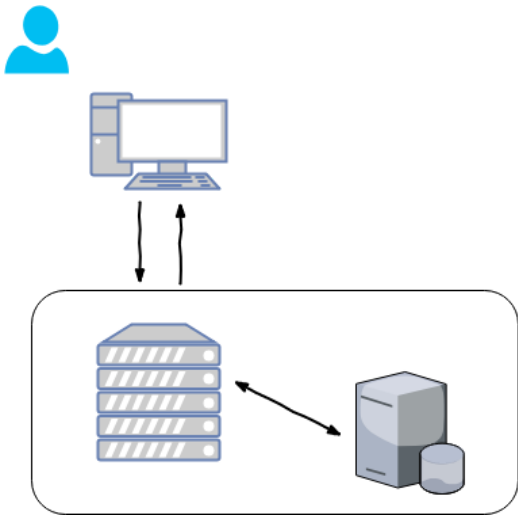
Ces choix technologiques ont été effectué car nous souhaitions avoir une première expérience avec ces deux frameworks qui sont largement utilisés, tous deux, dans l'industrie du Web. Quant à MongoDB, notre choix est fondé sur l'envie de découvrir la différence par rapport à un SGBD SQL comme mysql ou PostgreSQL.

Problèmes rencontrés

Après une première phase de prise en main, nous sommes vite retrouvés face à des problèmes de temps pour la prise en main des nouvelles technologies. Nous avons donc changé de voie pour changer la partie back-end et ainsi remplacer Spring par NodeJS + Express, afin d'utiliser notre expérience sur cette technologie et ainsi gagner du temps sur le développement du site web. Cependant aucun changement n'a été réalisé pour la partie front-end et le stockage de nos données.



Architecture



Comme nous pouvons le voir sur le schéma d'architecture ci-dessus, l'utilisateur se connecte sur le site internet de vente en ligne à travers son navigateur Internet, représenter par son ordinateur ci-dessus. Ensuite, deux choix s'offrent à lui, soit découvrir le site et son contenu, soit se connecter et avoir la possibilité de faire des achats en ligne.
Chaque échange entre le client et le serveur se faire via une API REST ci-dessus, cela peut notamment être les cas suivants :

- La connexion ou la déconnexion de l'utilisateur,
- La modification des informations du compte utilisateur,
- Le chargement des différents produits,
- L'ajout ou la suppression d'un produit dans le panier,
- La validation et l'envoie des différentes informations à travers les étapes de paiement (livraison, paiement, paiement validé ou non)

Concernant le stockage des données, nous avons choisi d'utiliser une base de données NoSQL de type MongoDB qui sera sur le même serveur que notre back-end développé avec NodeJS + Express.

Afin de faire communiquer le back-end et le front-end, nous avons choisi de créer une API REST qui est modélisée par le modèle présent dans l'archive avec ce même document (swagger.yaml).

Fiabilité

Afin d'augmenter la fiabilité de notre application, nous avons décidé de mettre en place des tests unitaire côté back-end et des tests End-to-End côté front-end, via l'outil Selenium.

Aussi, nous avons choisi de mettre en place SonarQube pour tester la qualité de notre code et pouvoir l'améliorer.

Sécurité

En terme de sécurité, nous avons choisi pour l'instant de mettre en place un token d'authentification (Json Web Token). Celui-ci sera ensuite envoyé à l'utilisateur, qui l'enverra à son tour dans ces requêtes pour prouver à son tour son identité.

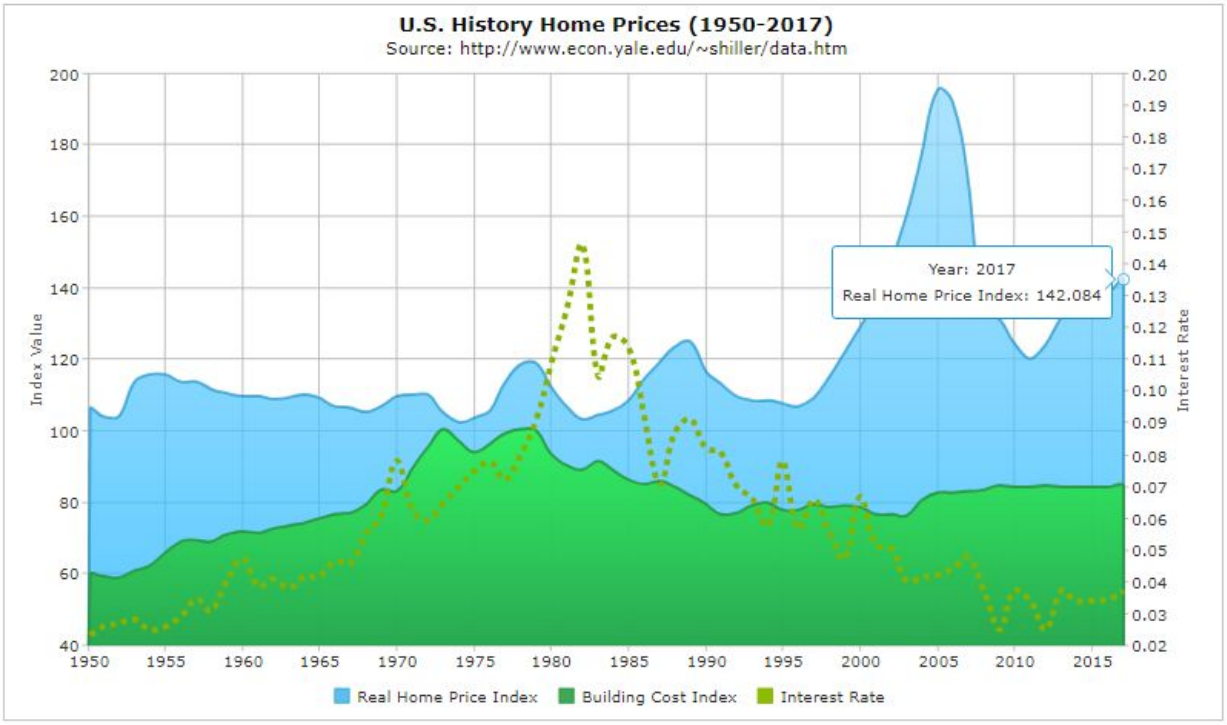
Afin d'assurer une couche de sécurité supplémentaire durant les transactions, nous mettons en place "Let's Encrypt" qui permet d'activer le protocole SSL et ainsi imposer l'utilisation de certificat et rendre les requêtes illisibles (attaque Man-In-The-Middle par exemple).

Améliorations possibles

Avec la contrainte de temps, sans oublier les différentes tâches en dehors de ce cours, nous n'avons pas pu s'accorder beaucoup de temps sur l'optimisation du code, ce qui en fait une amélioration possible.

Il est tout à fait possible de voir plus de fonctionnalités à ajouter à notre site Web, par exemple pour aider les vendeurs du site web, ils pourraient avoir une page qui recense tous les indicateurs de stock, commandes, d'utilisateurs, etc. sous forme de graphique ou/et tableau.

Ci-dessous vous pouvez retrouver un exemple de graphique, réalisable grâce à la feature 'jQWidgets Chart' disponible pour Angular 7 :



Conclusion

Comme nous l'avons indiqué, un manque de temps ne nous a pas permis de prendre en main correctement Spring mais aussi Angular en partie. Nous avons décidé de changer pour une technologie avec laquelle nous avons plus d'expérience afin de pouvoir s'assurer un back-end fonctionnel mais renforcer nos efforts sur Angular.

Pour conclure, ce projet est intéressant afin qu'on puisse mettre plus d'importance sur la partie des tests unitaire, que ça soit la partie front-end ainsi que pour le back-end. La liberté sur le choix technologique peut être intéressante mais aussi un problème qui peut apparaître au cours du projet.