Université Mohammed V
de Rabat

Conception et réalisation d'une application web de génération de code HTML d'une maquette

Réalisé par :
BOUTINZER Mouhcine
OUOKKI Mohamed
NIA Achraf
STITOU Nada

Encadré par :
Pr. MAHMOUD EL HAMLAOUI

Année universitaire : 2019/2020

REMERCIEMENTS

Ce travail est l'aboutissement d'un dur labeur et de beaucoup de sacrifices ; nos remerciements vont d'abord à nos familles. Je souhaite également remercier toute la communauté IT, qui ont bien voulu répondre à nos différentes questions sur le domaine dans le but de permettre la facilité de l'élaboration de ce travail. Un remerciement spécial à notre encadrant Professeur M. MAHMOUD EL HAMLAOUI qui nous a fournit les informations necéssaires pour pouvoir accomplir ce travail . En dernier lieu je remercie l'administration et le corps enseignant de l'ENSIAS qui à travers leur programme nous ont fourni des outils de qualité facilitant notre spécialisation.

Resumé

Automatiser le développement front-end est l'une des tâches très importantes. cette application utilise un modèle d'apprentissage en profondeur qui utilise un réseau neuronal convolutif (CNN) pour extraire des caractéristiques d'image des images source, qui prend des maquettes Web dessinées à la main et les convertit en code HTML de travail. Il utilise une architecture de sous-titrage d'images par tokenisation pour générer son balisage HTML à partir de maquettes de sites Web dessinées à la main.

Ce projet se veut une preuve de concept ; le modèle n'est pas encore conçu pour se généraliser à la variabilité des maquettes observées dans les filaires réels, et donc ses performances reposent sur des filaires ressemblant à l'ensemble de données de base.

Abstract

Automating front-end development is one of the very important tasks. this app uses a deep learning model that uses a convolutional neural network (CNN) to extract image characteristics from source images, which takes hand-drawn web mockups and converts them to working HTML. It uses a tokenization image captioning architecture to generate its HTML markup from mockups of hand-drawn websites. This project is intended as

a proof of concept ; the model is not yet designed to generalize to the variability of the models observed in real wireframes, and therefore its performance is based on wireframes resembling all the basic data.

Table des matières

Table des figures

Liste des tableaux

— Lien vers repository SPRING API : https://github.com/Ouokki/MockupToHtmlCss

— Lien vers repository Flask : https://github.com/Achrafnia/m2c-flask

1 Introduction

Une application web riche est un logiciel informatique dont la vocation est de fonctionner dans un navigateur web, type Firefox, Chrome, Edge, Safari, etc. Elle est communément appelée application client léger et est constituée de deux parties qui intercommuniquent : le front-end (partie visible correspondant aux pages web construites et affichées dans votre navigateur) et le back-end (partie invisible construite et déployée sur un serveur d'application tel que Tomcat [1] et réalisant les traitements lourds en vue de répondre aux requêtes soumises par le front-end).

2 Contexte général

Notre projet consiste à construire complètement une application web générée via le framework Spring Boot [2] chargé de la gestion de sa partie back-end. Nous montrerons comment enrichir cette application d'une couche cliente (ou front-end) à l'aide du framework Angular [3] dans sa version 9.

Pour réaliser les objectives de ce projet, nous avons utilisé un tas d'outils, que nous allons introduire dans cette section, Spring boot pour la partie back end, Angular pour la partie front end, flask [4] pour l'apprentissage automatique.

2.1 Objectifs du projet

Le principal objectif du projet est la génération de la forme d'un site web a partir d'une image donnée. Or faciliter la création d'une page web grace à une simple image

2.2 problématique

. Ce travail ne consiste pas à réaliser le backend et le frontend d'un site web, mais proposer un travail plus professionnel en utilisant des technologies permettant la réalisation d'un travail bien structuré ayant pour finalité l'aisance du travail commun au sein d'un groupe et le gain de temps.

2.3 Présentation des clients

Cette application est destinée a tout ceux qui souhaitent construire une page HTML sans connaissance concernant la partie codage de ce dernier.

2.4 Outils utilisés

2.4.1 Spring

FIGURE 1 – Spring

Spring est un framework open source pour construire et définir l'infrastructure d'une application Java, dont il facilite le développement et les tests.

2.4.2 Angular

FIGURE 2 – Angular

Angular est un cadriciel côté client, open source, basé sur TypeScript, et co-dirigé par l'équipe du projet « Angular » à Google et par une communauté de particuliers et de sociétés. Angular est une réécriture complète de AngularJ.

2.4.3 Flask

FIGURE 3 – Flask

Flask est un framework open-source de développement web en Python. Son but principal est d'être léger, afin de garder la souplesse de la programmation Python, associé à un système de templates.

2.4.4 Tomcat

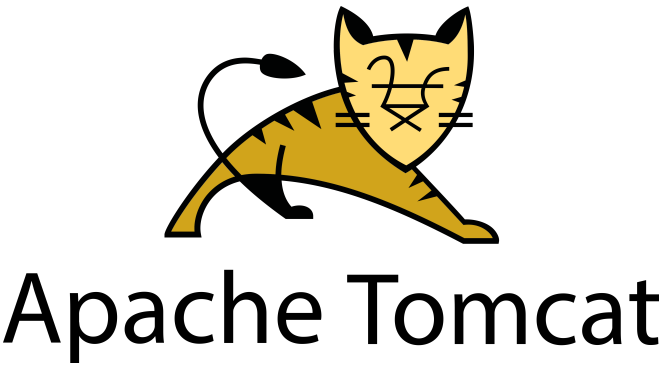

FIGURE 4 – Tomcat

Apache Tomcat est un conteneur web libre de servlets et JSP. Issu du projet Jakarta, c'est un des nombreux projets de l'Apache Software Foundation.

2.4.5 primefaces

FIGURE 5 – Primefaces

PrimeFaces est une bibliothèque de composants interface utilisateur open source pour des applications JavaServer Faces.

2.4.6 Docker

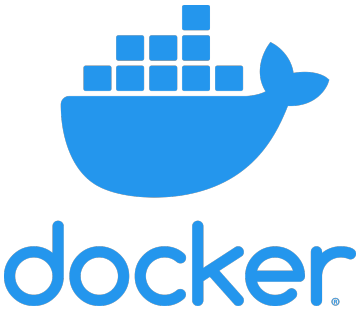

FIGURE 6 – Primefaces

Docker est un outil qui peut empaqueter une application et ses dépendances dans un conteneur isolé, qui pourra être exécuté sur n'importe quel serveur.

2.5 Conclusion

Ce chapitre est le point de départ pour la réalisation de notre projet puis il nous a permis d'avoir une vision sur son contexte général en présentant d'abord l'objectif à atteindre, les clients visés et les outils utilisés pour ce dernier. Le chapitre suivant est consacré pour l'étude conceptuelle et existentielle de notre projet.

3 Analyse et conception

Tout au long de ce chapitre, nous allons proposer et présenter un diagramme général de cas d'utilisation et un diagramme de classes, ainsi qu'un diagramme de séquence.

3.1 Introduction

La phase de modélisation et conception est la première étape du processus de développement adoptée. En effet, elle formalise et détaille ce qui a été ébauché au cours de l'étude préliminaire, et permet de dégager l'étude fonctionnelle du système. Elle permet ainsi d'obtenir une idée sur ce qui réalisera l'application proposée au cours de ce projet .

3.2 Analyse des besoins

Cette partie va servir à poser les bases du recueil des besoins du système à réaliser. Pour pouvoir clarifier les besoins des utilisateurs de notre application, nous allons présenter les besoins fonctionnels ainsi que les besoins non fonctionnels.

3.2.1 Les besoins fonctionnels

Après une étude détaillée du système, cette partie est réservée à la description des exigences fonctionnelles des différents acteurs de l'application. Ces besoins se regroupent dans le diagramme des cas d'utilisation. Notre application permet aux utilisateurs de :

— Créer un compte .
— S'authentifier .
— Editer le compte (mot de pass).
— Télèverser une image.
— Visualiser le code HTML de l'image en question.
— Télécharger le code HTML.

3.2.2 Les besoins non fonctionnels

Les besoins non fonctionnels décrivent toutes les contraintes techniques, ergonomiques et esthétiques auxquelles est soumis le système pour sa réalisation et pour son bon fonctionnement. Et en ce qui concerne notre application, nous avons dégagé les besoins suivants :

— La disponibilité : L'application doit être disponible pour être utilisé par n'importe quel utilisateur
— La convivialité de l'interface graphique : L'application doit fournir une interface conviviale et simple pour tout utilisateur, car, par le biais de celle-ci on découvrira ses fonctionnalités
— La performance : Le système doit réagir dans un délai précis, quel que soit l'action de l'utilisateur.
— La sécurité : protection des entrées et sorties.

3.3 Méthodologie utilisée

3.3.1 Introduction

Avant de créer la base de données et d'avoir un aperçu de l'ensemble du travail à effectuer, il faut réaliser une analyse du projet. Après en avoir discuté avec mon encadrant, j'ai choisi la méthode UML pour effectuer cette analyse. cette analyse est représenté par le diagramme de cas d'utilisation et le diagramme de classes .

3.3.2 Présentation d'UML

UML (Unified Modeling Language), se définit comme un langage de modélisation graphique et textuel destiné à comprendre et à définir des besoins, spécifier et documenter des systèmes, esquisser des architectures logicielles, concevoir des solutions et communiquer des points de vue. Il véhicule en particulier :

— Les concepts des approches par objets : classe, instance, classification, etc.
— Intégrant d'autres aspects : associations, fonctionnalités, événements, états, séquences, etc.

UML définit neuf types de diagrammes devisés en deux catégories :

— Diagrammes statiques (structurels) : diagramme de classe, d'objet, de composant, de déploiement et de diagramme de cas d'utilisation.
— Diagrammes dynamique (comportementaux) : diagramme d'activité, de séquence, d'état-transition et de diagramme de collaboration

3.3.3 Diagramme de cas d'utilisation

Un cas d'utilisation (Use case) « représente un ensemble de séquences d'actions réalisées par le système et produisant un résultat observable intéressant pour un acteur particulier ».
En effet, ils sont des représentations fonctionnelles du système, ils permettent de modéliser les attentes des utilisateurs afin de réaliser une bonne délimitation du système et également d'améliorer la compréhension de son fonctionnement.

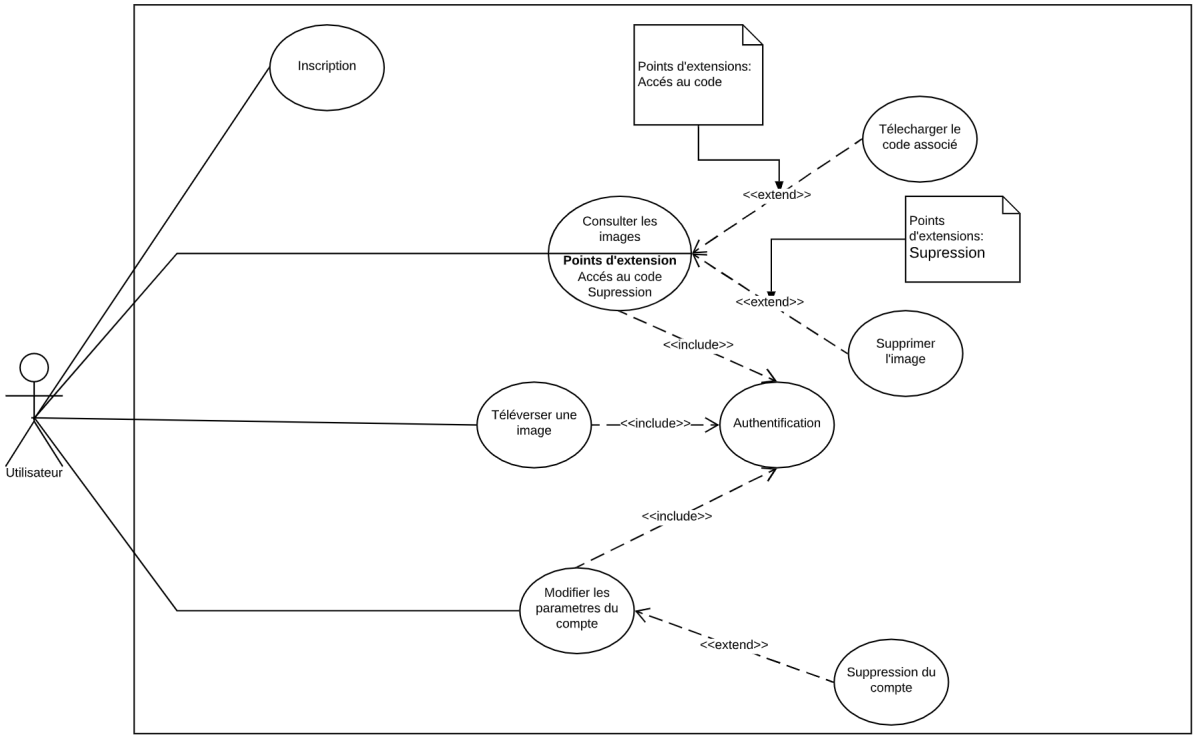

FIGURE 7 – Use case

3.3.4 Diagramme de classe

Le diagramme de classes permet de représenter l'aspect statique des structures et énumère les différentes classes du système à modéliser.

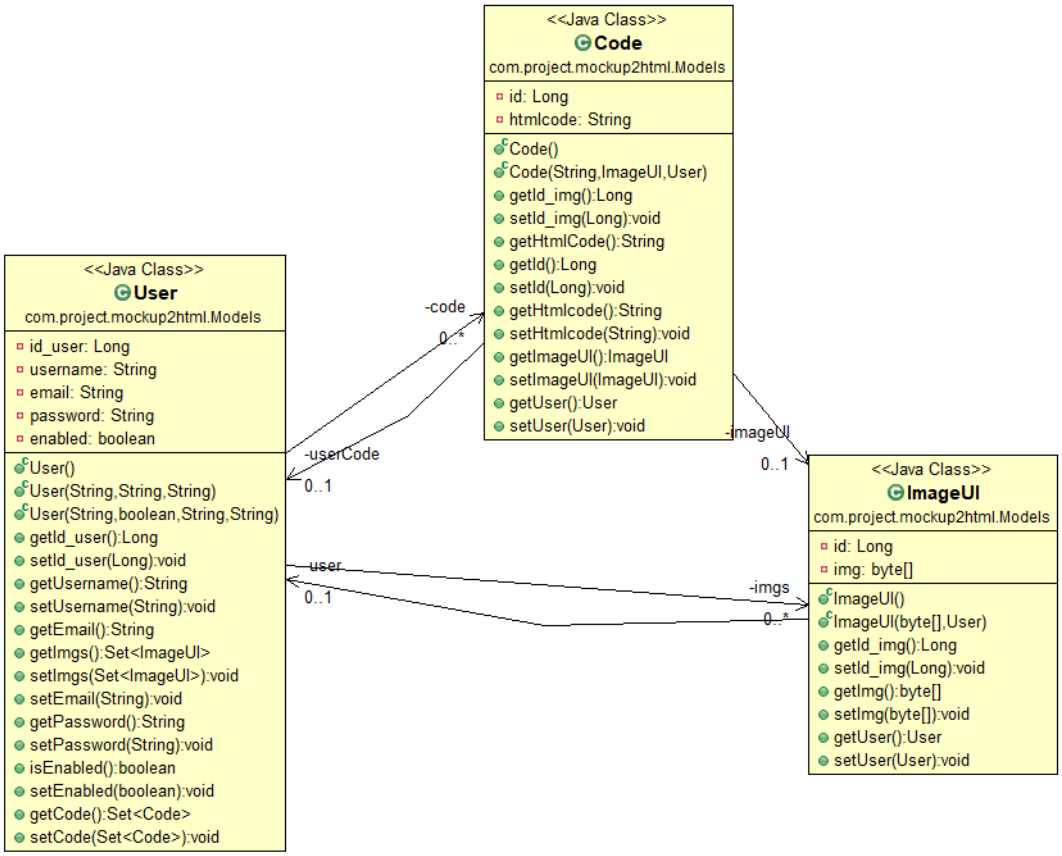

FIGURE 8 – Diagramme de classe

3.4 Conclusion

Tout au long du chapitre, nous avons bien exposé toute une analyse bien détaillée concernant la conception du projet. Le chapitre suivant serait sur la partie de la réalisation qui est inspirée sur la base de la partie de la conception.

4 Réalisation et mise en oeuvre

Dans ce dernier chapitre nous nous sommes intéressés à la partie réalisation de notre projet. Il s'agit de présenter une solution pour notre problématique.

4.1 Présentation des interfaces utilisateur

À ce stade, on entame la présentation le travail réalisé à travers des prises d'écran.

4.1.1 Primefaces

Primefaces [5] offre un nombre important de composants qui permettent de représenter l'information en fonction de la forme et du contenu de celle-ci. On peut citer entre autres :

- Cases à cocher et listes déroulantes avec option de complétion, ou choix multiples idéal pour proposer aux utiliser des options sur les formulaires
- Tableaux avec filtres, sélection facilitant l'affichage de la donnée ainsi que sa manipulation
- Layouts responsive permettant de structurer les interfaces utilisateur
- Messages de notification colorés et personnalisables

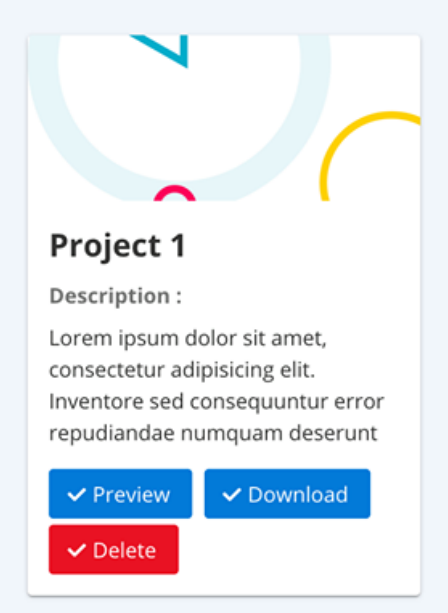

FIGURE 9 – Exemple d'application de primefaces

4.1.2 Accueil de l'application

La page d'accueil est l'entrée principale , l'utilisateur peut s'inscrire ou se connecter.

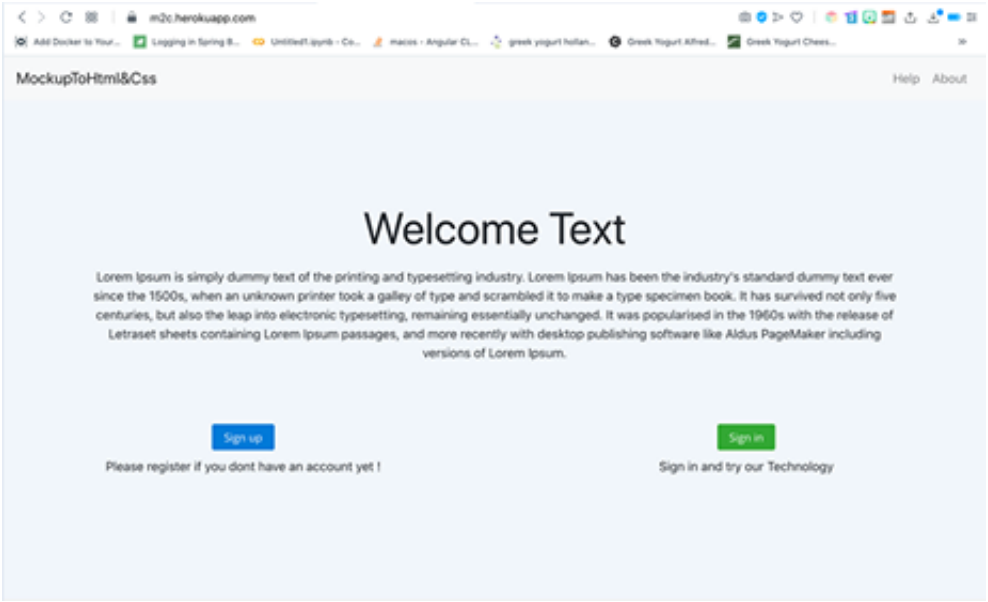

FIGURE 10 – Page d'accueil

4.1.3 Inscription

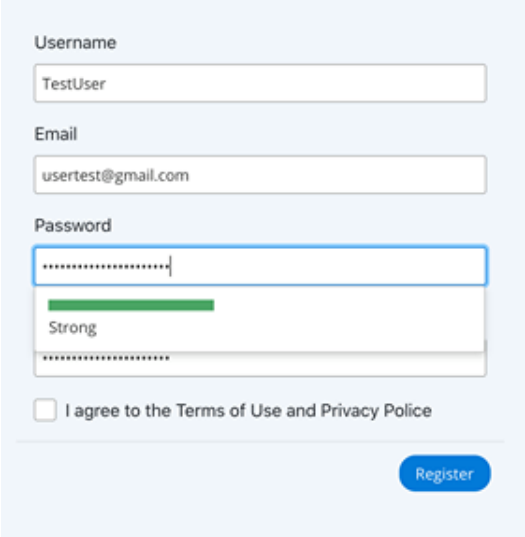

FIGURE 11 – Page d'inscription

4.1.4 Télé verser une images

L'utilisateur peut télé verser une image, pour qu'elle soit instantanément traitée.

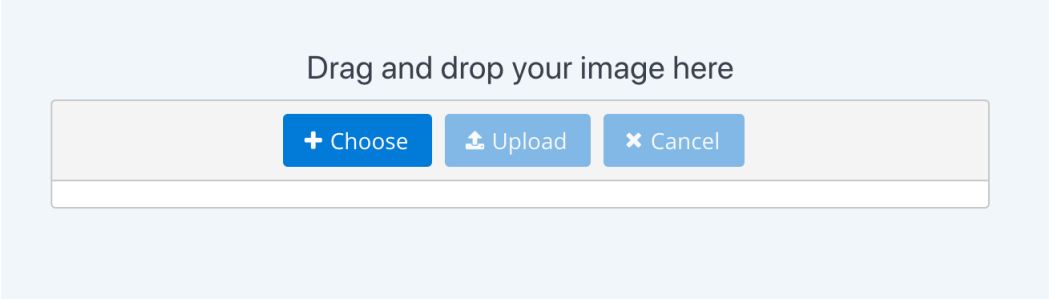

FIGURE 12 – Page de télé versement des images

4.1.5 Consulter ses projets

Après avoir sélectionné une image à partir de l'interface d'accueil, l'utilisateur pourra donc utiliser cette interface pour consulter ses projets, les télécharger ou les supprimer.

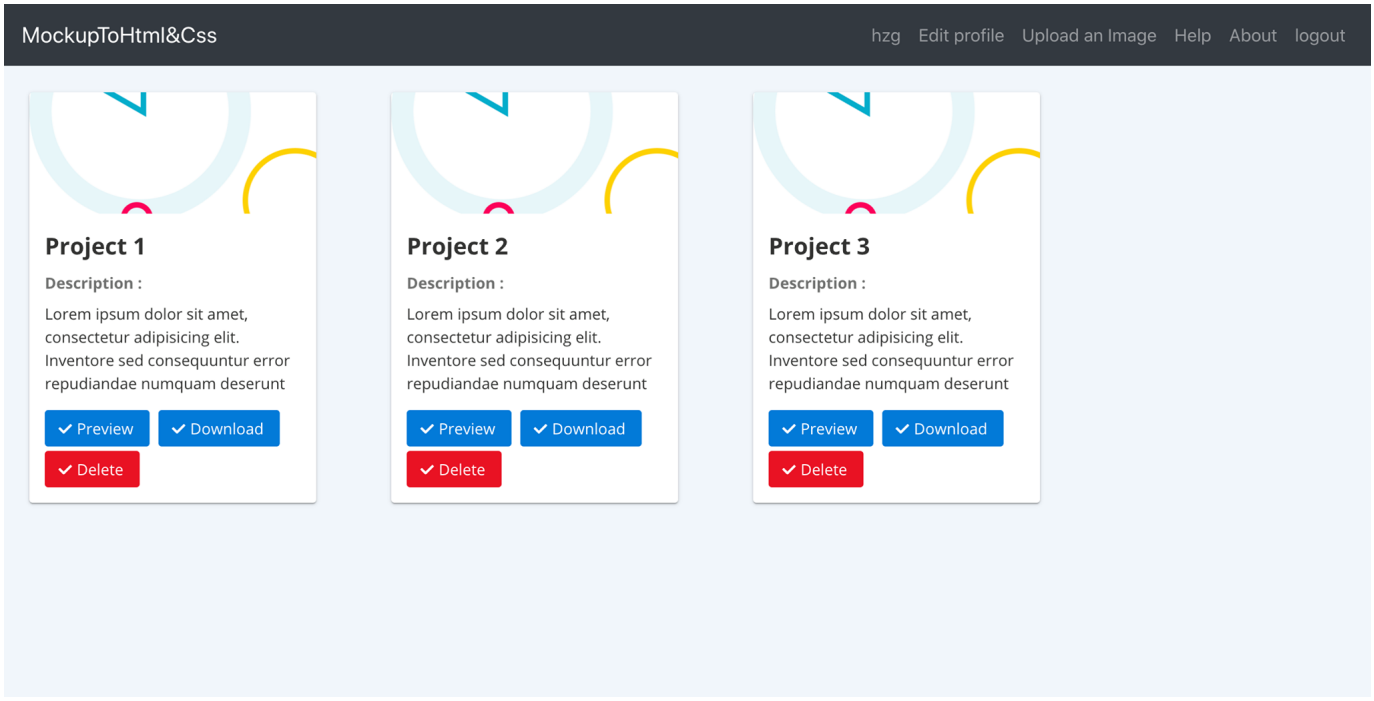

FIGURE 13 – Page de consultation des projets

4.2 Test

4.2.1 tests de bout en bout

Les tests de bout en bout [6] sont une technique utilisée pour vérifier si une application (site Internet, application mobile...) se comporte comme prévu du début à la fin. Le testeur doit se mettre dans la peau d'un utilisateur et dérouler les tests comme s'il utilisait véritablement l'outil mis à sa disposition. Cette technique permet de valider le fonctionnement du front. Mais aussi de vérifier son intégration avec le back-office et autres webservices.
L'utilisation de protractor pour angular nous à permis de réaliser ce type de test.

```
(base) mac@MacBook-Pro client % protractor conf.js
[02:27:58] I/launcher - Running 1 instances of WebDriver
[02:27:58] I/hosted - Using the selenium server at http://localhost:4444/wd/hub
Started
---------------------------------------
[02:28:17] W/element - more than one element found for locator By(css selector, button[class="ui-button ui-widget ui-state-default ui-
corner-all ui-button-text-icon-left"]) - the first result will be used
F

Failures:
1) Image Upload test  Should -------------------------
(base) mac@MacBook-Pro client % protractor conf.js
[02:30:04] I/launcher - Running 1 instances of WebDriver
[02:30:04] I/hosted - Using the selenium server at http://localhost:4444/wd/hub
Started
---------------------------------------
[02:30:17] W/element - more than one element found for locator By(css selector, button[class="ui-button ui-widget ui-state-default ui-
corner-all ui-button-text-icon-left"]) - the first result will be used
.


1 spec, 0 failures
Finished in 14.902 seconds

[02:30:21] I/launcher - 0 instance(s) of WebDriver still running
[02:30:21] I/launcher - chrome #01 passed
(base) mac@MacBook-Pro client %
```

FIGURE 14 – End to end test

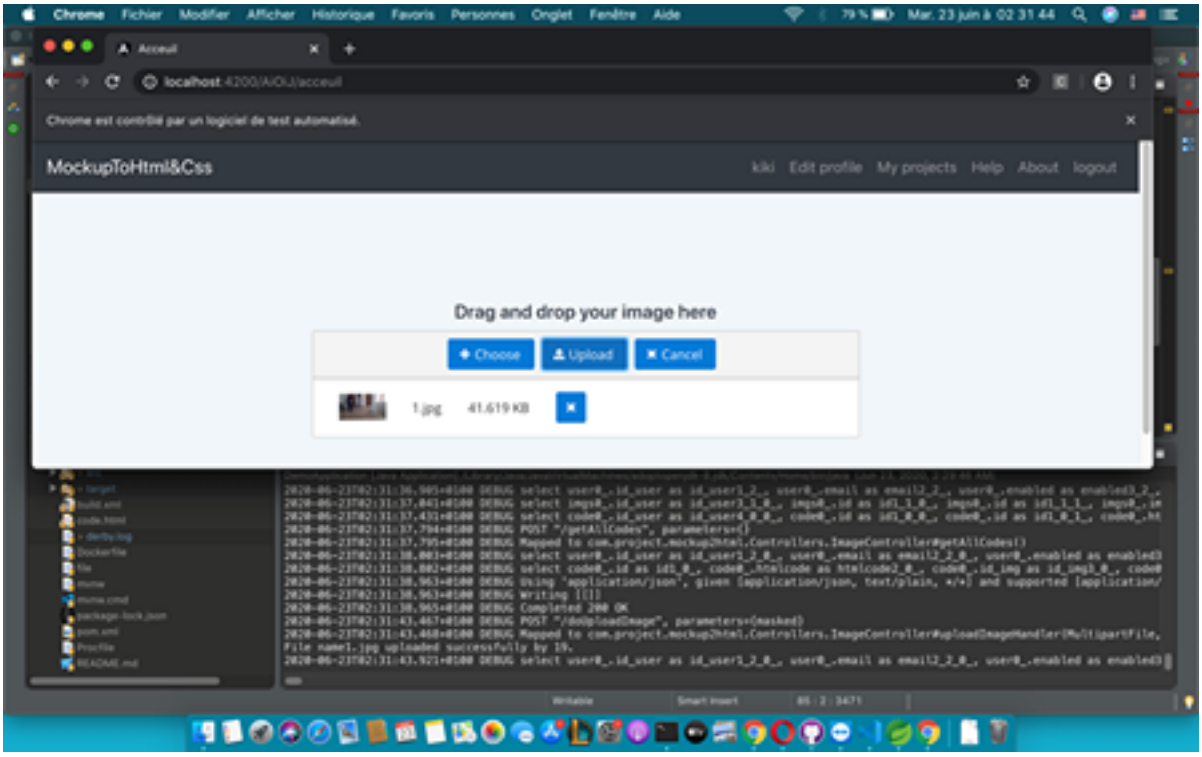

FIGURE 15 – End to end test en exécution

4.2.2 Arquillian

Arquillian [7] est une structure de test d'intégration agnostique par conteneur pour Java EE. L'utilisation d'Arquillian minimise la charge de gestion des conteneurs, des déploiements, des initialisations d'infrastructure, etc.
Tout d'abord, ShrinkWrap class fournit une API permettant de créer des fichiers déployables . Jar, . War, et ** . Ear .

Ensuite, Arquillian nous permet de configurer le déploiement de test à l'aide de l'annotation @ Deployment - sur une méthode renvoyant un objet ShrinkWrap .

4.2.3 Junit

JUnit est un framework open source pour le développement et l'exécution de tests unitaires automatisables. Le principal intérêt est de s'assurer que le code répond toujours aux besoins même après d'éventuelles modifications. Plus généralement, ce type de tests est appelé tests unitaires de non-régression.

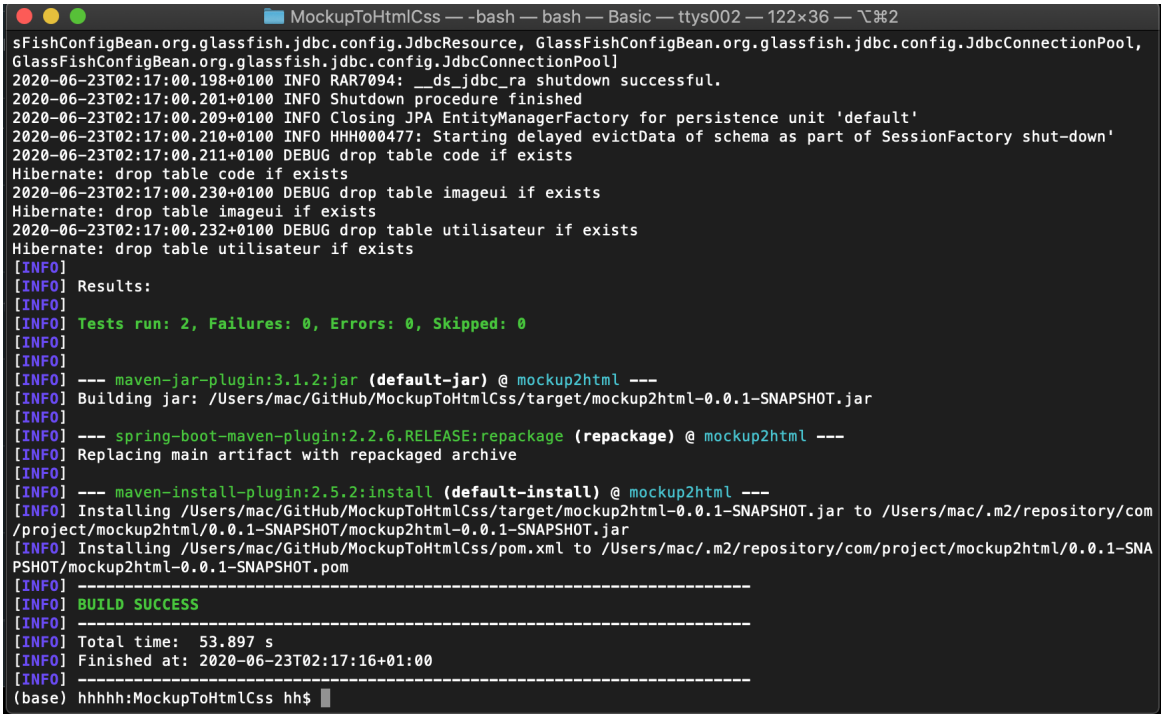

FIGURE 16 – Exécution des deux testes unitaires (Junit et arquillian)

4.2.4 Gestion de logs

La gestion de logs (LM pour Log Management) comprend une approche de la gestion de grands volumes des messages de log générés par l'ordinateur (aussi connu comme journaux d'évènements, journalisation, etc.). Aprés avoir implémenter log4j2 au niveau de spring boot, on prépare des fichier.xml décrivant les logs et leurs emplacements pour générer un fichier application.log qui contient toute la journalisation contenant les résultats des tests automatiques.

4.3 Diffusion continue

Les tests constituent un aspect important de la création de logiciels. Afin de garantir l'exécution de tests unitaires pour chaque version d'un projet, de nombreuses organisations d'ingénierie ont adopté la pratique de l'intégration continue (CI), qui implique l'utilisation d'outils comme Jenkins ou Travis CI, pour garantir que tout nouveau code fait l'objet de tests menés de manière automatique et cohérente pour détecter les éventuelles erreurs.

4.3.1 Travis

Travis CI peut déployer [8] et exécuter des tests de bout en bout dans un environnement de préproduction dans le cadre du processus de test. Le processus est défini par le fichier .travis.yml.

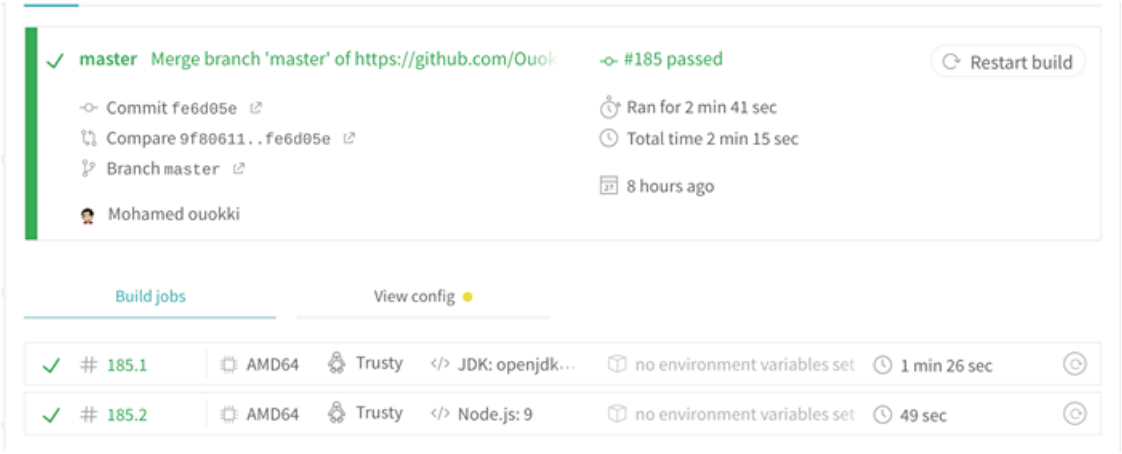

FIGURE 17 – Exécution de travis

4.4 Déploiement

4.4.1 Docker

Docker est utilisé par de très nombreuses sociétés pour différents usages. Ainsi, un des premiers usages de Docker se trouve dans la création d'environnements locaux. Il est plus simple d'utiliser Docker en local quand on travaille avec de nombreuses versions différentes des logiciels, et ainsi ne pas avoir des problèmes de compatibilité entre elles.

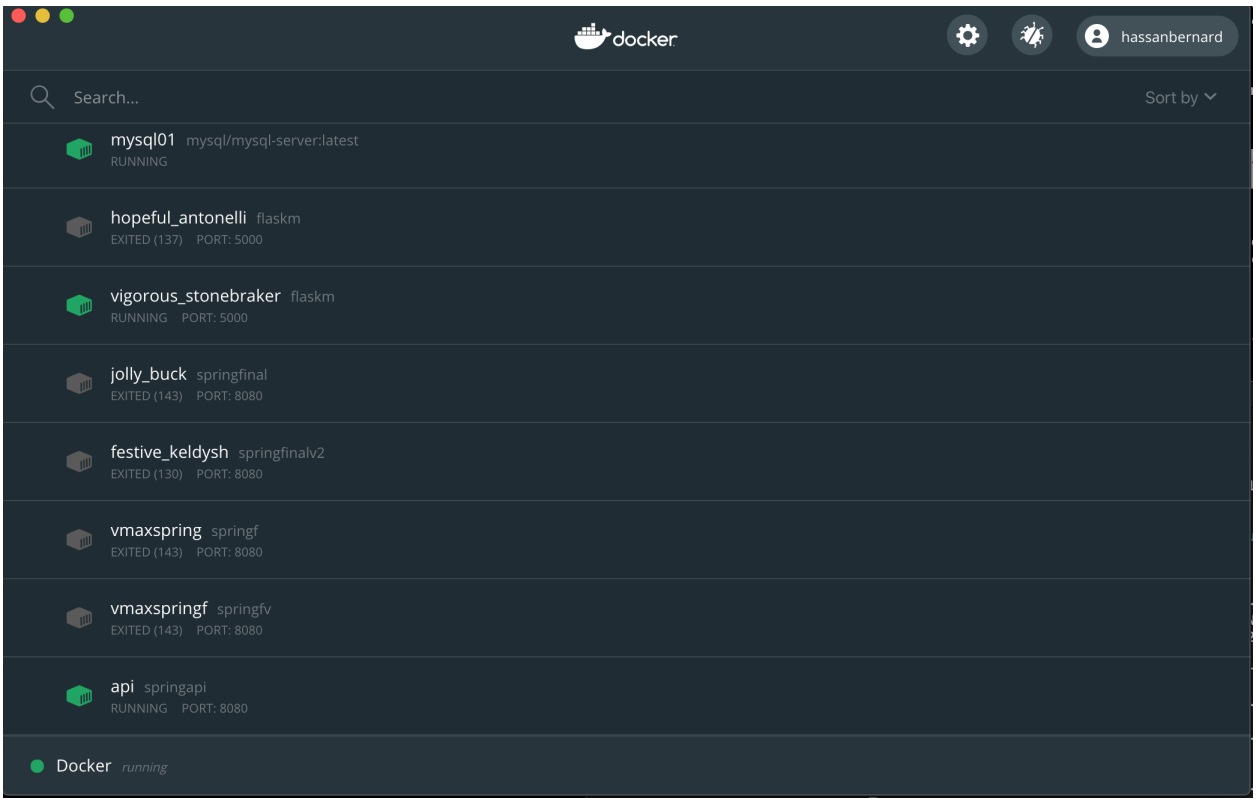

FIGURE 18 – Docker container

4.4.2 Heroku

Heroku est une plateforme en tant que service (PaaS) permettant de déployer des applications sur le Cloud très facilement.

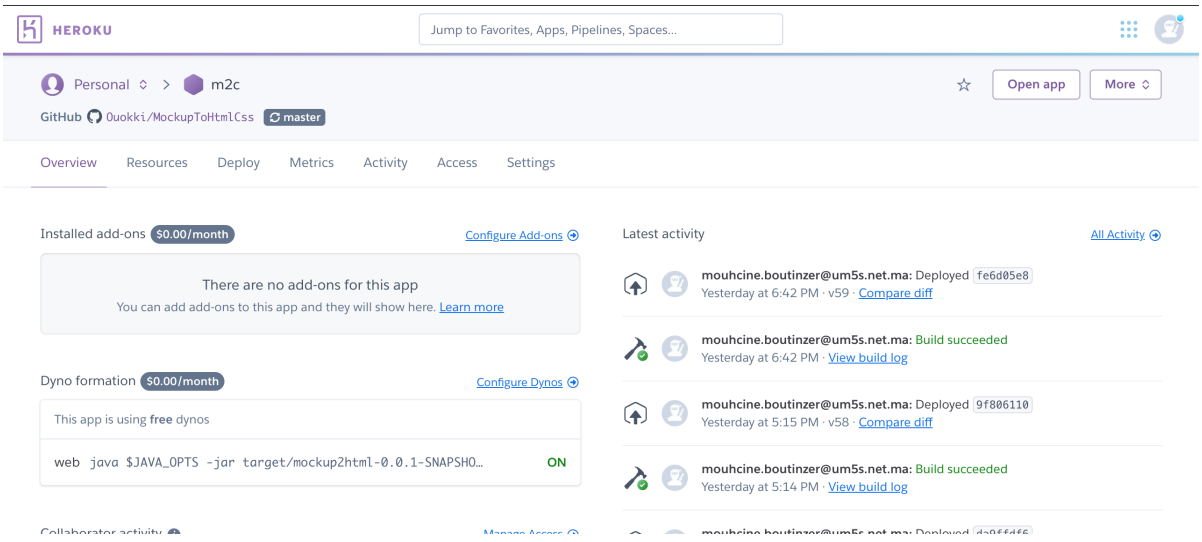

FIGURE 19 – Déploiement sur heroku

4.5 Conclusion

Dans ce chapitre, on a expliqué tout ce qui concerne la partie la plus importante de notre projet, où on a bien exposé la réalisation et la présentation du travail .

5 Conclusion et perspectives

Les principaux objectifs fixés à l'origine du projet étant atteints, nous pouvons affirmer que ce projet est une réussite. Nous sommes donc entièrement satisfaits du travail accompli tout au long du développement de ce projet ainsi que du résultat obtenu. L'apprentissage du Java Entreprise Edition et d'autres logiciels et librairies sert comme une préparation pour notre prochaine intégration dans le monde du travail.

6 Bibliographie

[1] http://tomcat.apache.org/index.html

[2] https://spring.io/projects/spring-framework

[3] https://angular.io/

[4] https://flask.palletsprojects.com/en/1.1.x/

[5] https://www.primefaces.org/primeng/

[6] https://www.protractortest.org/#/tutorial

[7] https://www.codeflow.site/fr/article/arquillian

[8] https://docs.travis-ci.com/user/deployment