Rapport Tests & Sécurité

Projet WeBuy application

Rédaction

Rév.	Date	Noms
1.	26/02/2022	Reboul
1.	26/02/2022	Lonjon

Révisions

Rév.	Date	Objet de la révision
1.0	26/02/2022	Création du document

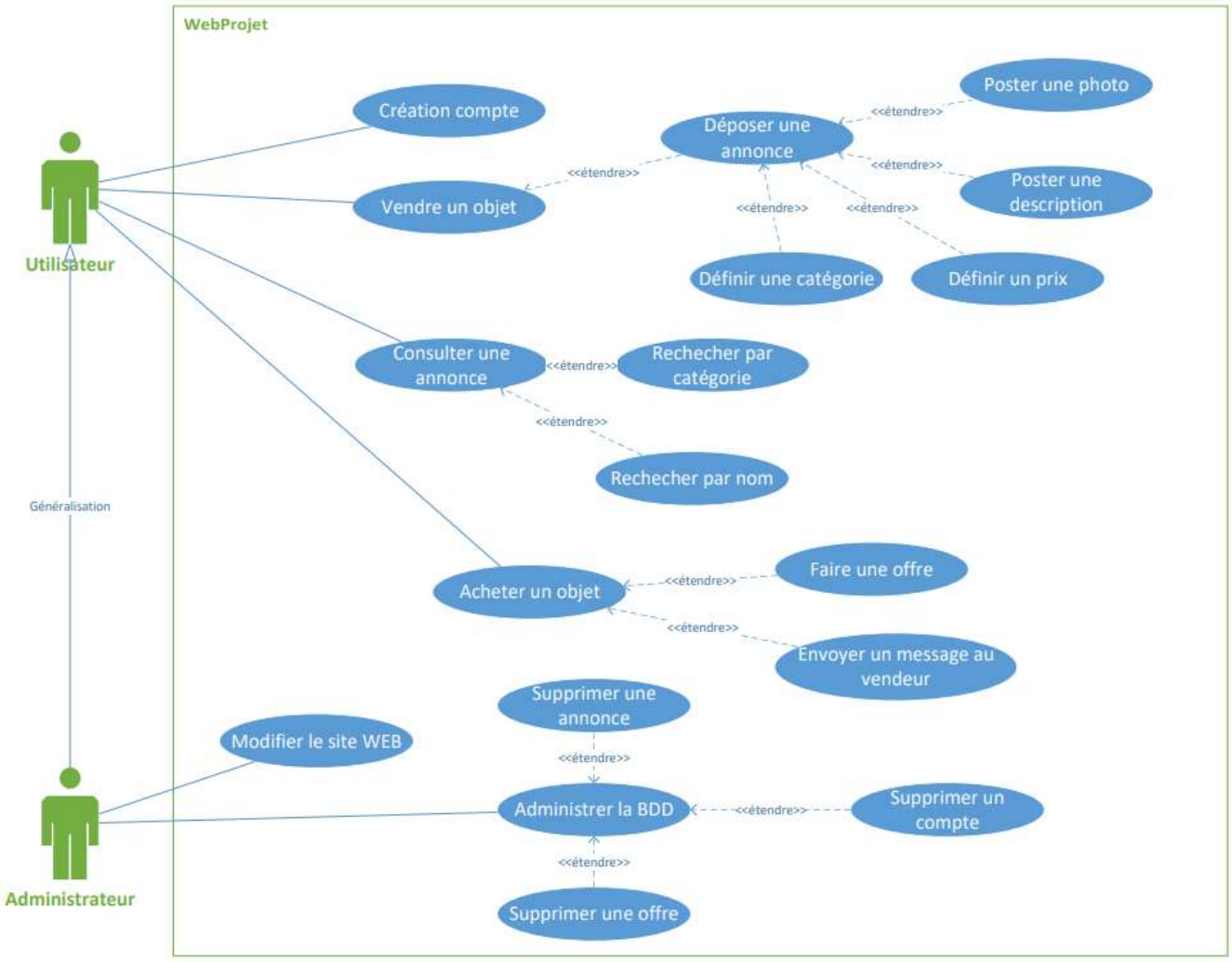
Sommaire

2. Introduction

Dans le cadre de notre unité d'enseignement d'Administration Web, nous avons pour projet la mise en place d'une application Web permettant la vente et l'achat de produits entre utilisateurs. L'objectif étant de développer un site web basé sur des technologies Node.js, Express.js et Angular.

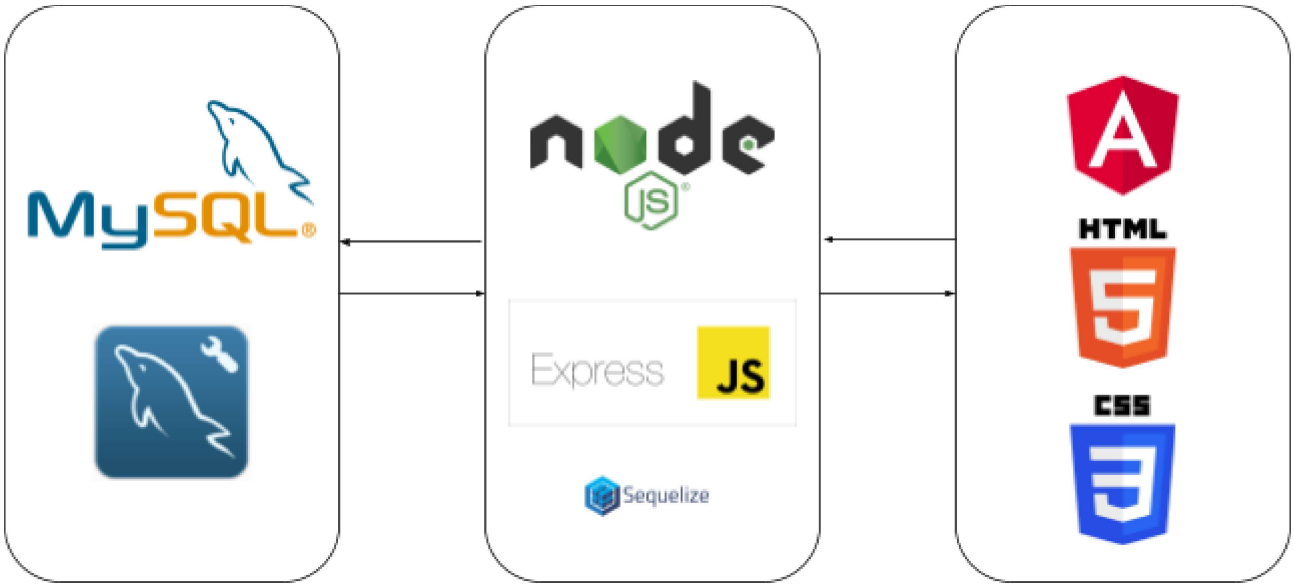
2.1. Définition des fonctions

Pour ce projet nous avons créé un diagramme Use Case, définissant les différentes fonctions disponibles sur le site Web. Ce diagramme est sert de référence pour nos spécifications ainsi que nos développements. L'expression des fonctions reste cependant faiblement détaillée dans le diagramme. L'objectif étant de dresser les grandes lignes de notre projet :



3. Environnement de travail

Pour l'environnement de travail nous avons besoin de différents outils afin de développer et agrémenter notre application de fonctionnalités tout au long du projet. Dans un premier temps on peut identifier 3 stack techniques permettant le fonctionnement de notre application :



- Notre moteur de base de données : MySQL avec MySQLWorkbench pour la visualisation de la base.

- Notre serveur qui va interagir directement avec la base et piloter les données au travers de modèles, puis communiquer les données nécessaires au front-end pour l'affichage.

- Finalement la partie front-end avec Angular, HTML & CSS permettant de proposer l'interface utilisateur et de développer tous les éléments d'interactions.

Au-delà de la stack technique permettant le développement direct du site web, on dispose également d'autres outils de support permettant d'assurer le versionning, la qualité et la sûreté du code :

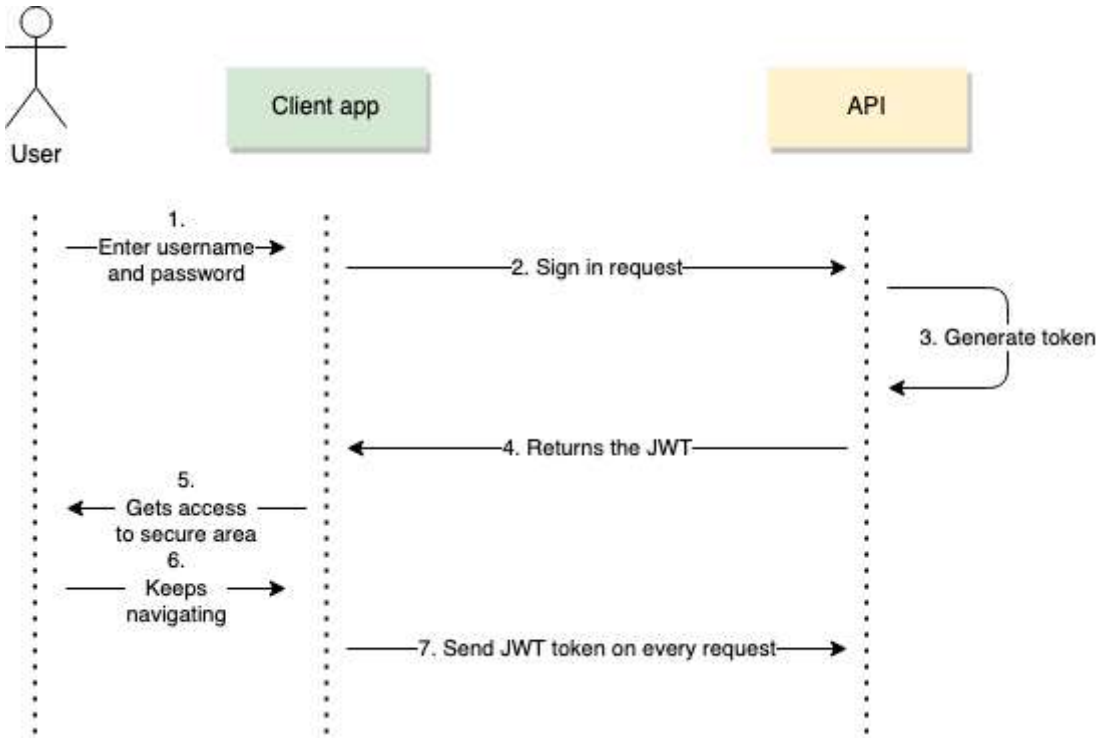
- Git avec GitHub : Le VCS le plus connu pour le développement en équipe et le versionning.

- Tests automatisés : Cypress

- IDE Integrated tools : Eslint pour la qualité de code, Prettify afin de gérer l'indentation et l'organisation des fichiers, le tout sur Visual Code.

4. Sécurité

Pour la sécurité nous avons choisi d'utiliser, le paquet "bcrypt" de nodejs ainsi qu'un json web token. Le paquet bcrypt permet l'encodage des mots de passe des utilisateurs lors des requêtes auprès de la BDD. Le token, permettra d'assurer la bonne authentification de l'utilisateur auprès des différentes pages et fonctionnalités de notre site web :

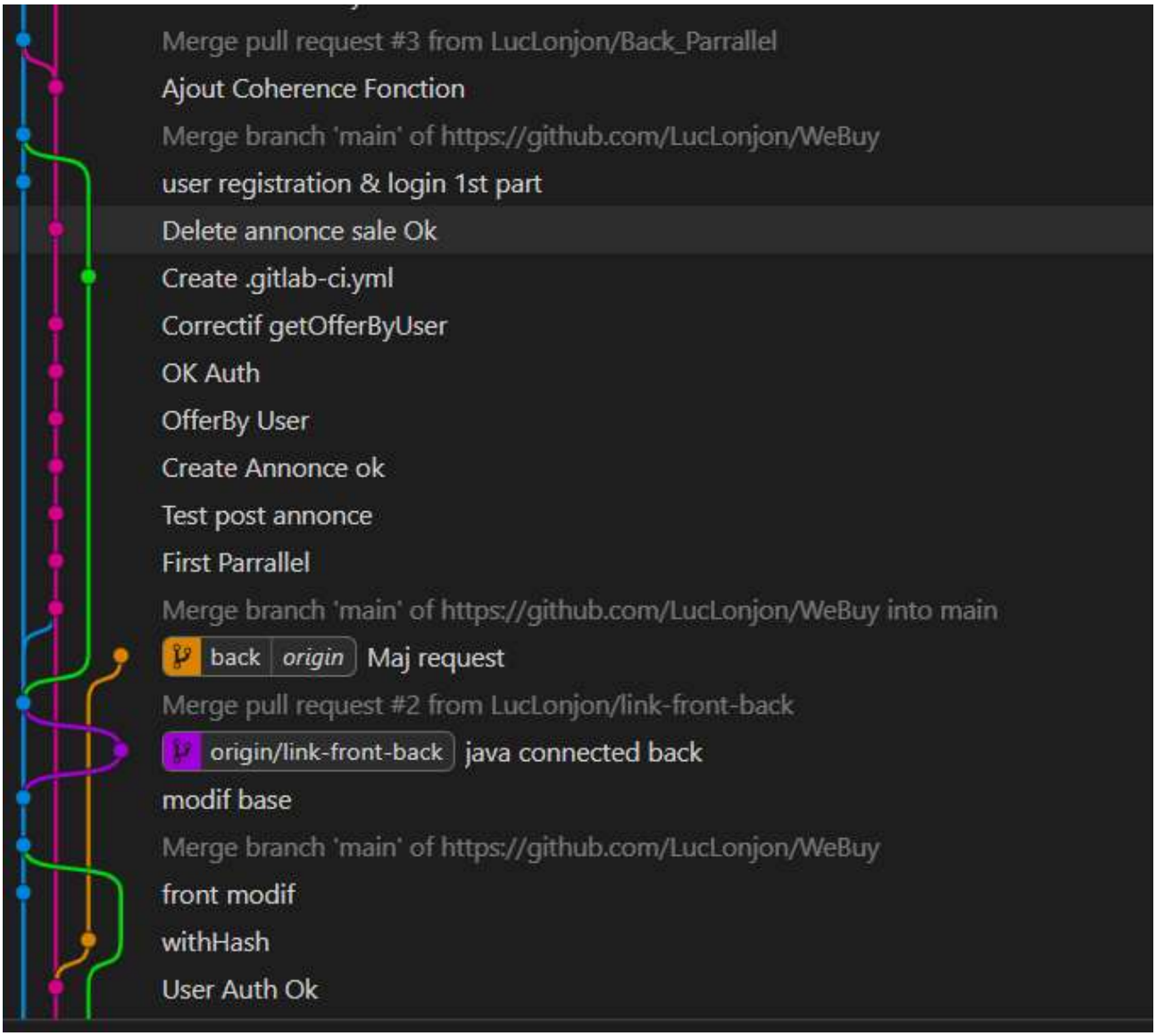


Ces différentes vérifications seront réalisées par des fonctions "Middleware" comprenant aussi bien :

- La gestion des erreurs d'authentification
- La vérification de la bonne authentification

5. Organisation GitHub

Pour notre projet comme mentionné précédemment nous avons choisi d'utiliser GitHub . Afin de faciliter la lecture des différentes branches et push réalisés nous avons décidé d'utiliser GitGraph sur VScode. Cette extension permet de visualiser graphiquement les différents changements sur notre repository GIT. Cet outil nous à permis de régler de nombreux problèmes. GitGraph ci-dessous :



Nous avons décidé d'utiliser une architecture avec deux branches :

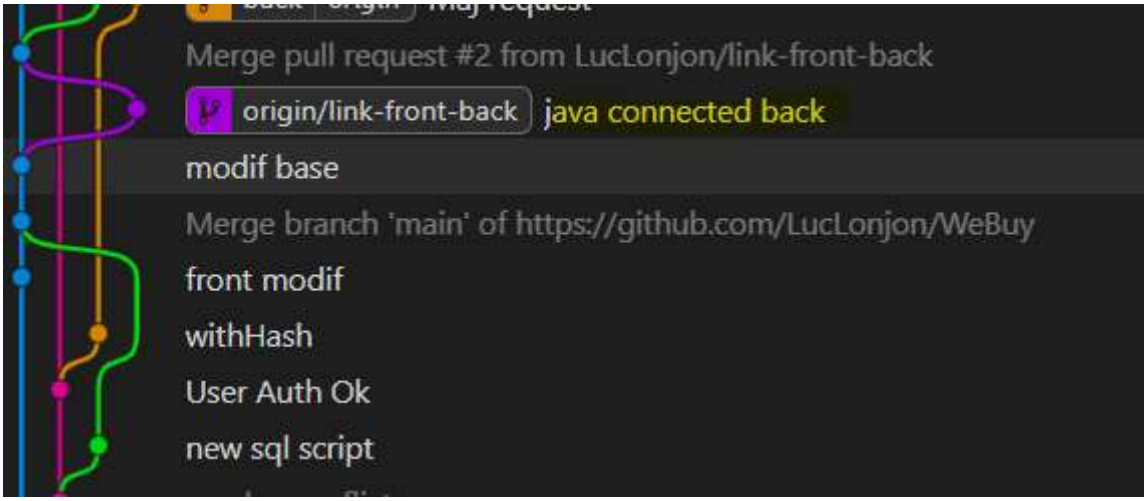
- Une FrontEnd
- Une BackEnd

Nous avions défini à des points précis des merges de la Branch BackEnd sur la BranchFront (main). Cette architecture n'était pas la plus simple à maintenir, mais nous étions rigoureux avec des

commits et des push réguliers. Au début du projet nous avons voulu adopter un virage technologique en changeant de stack technique pour le back.

Node js -> Spring Boot

C'est pourquoi au sein de notre Git on peut voir un moment de divergence :



Cet essai étant avéré non concluant, nous avons supprimé la branche.

Enfin, nous avons une baisse de modifications sur le repository à partir de janvier et de début février notamment dû à la fin de notre PFE et du départ en Irlande. Nous avons retravaillé sur le projet à notre retour en France comme en attestent les PUSH.

6. Tests

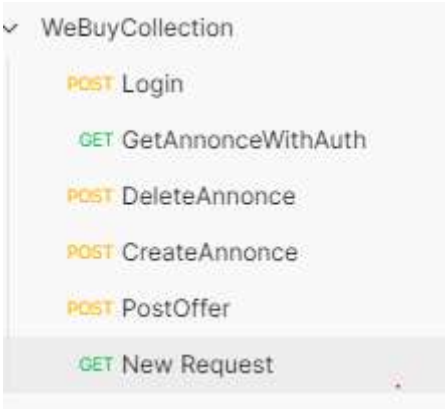
6.1. Méthode de tests

Au sein de notre projet nous avons occulté pendant une grande partie les tests automatisés ainsi que la qualimétrie de notre code. Nous étions obnubilés par le déploiement des nouvelles features pour le site. La majorité de nos tests réalisés était des tests unitaires sur Postman pour notre API Rest car son non fonctionnement était bloquant pour nous deux. Nous étions cependant au courant de l'importance de tests de façon approfondie nos fonctions afin de vérifier d'éventuels trous dans la raquette. La majorité des erreurs étaient donc souvent trouvées par l'autre binôme lors du test Front end + Back end. Nous avons cependant changé notre fusil d'épaule avec l'intégration de cypress puis de sonarqube. Leurs utilisations bien que limitées par nos compétences et le temps accordé nous ont montré la puissance de ses outils.

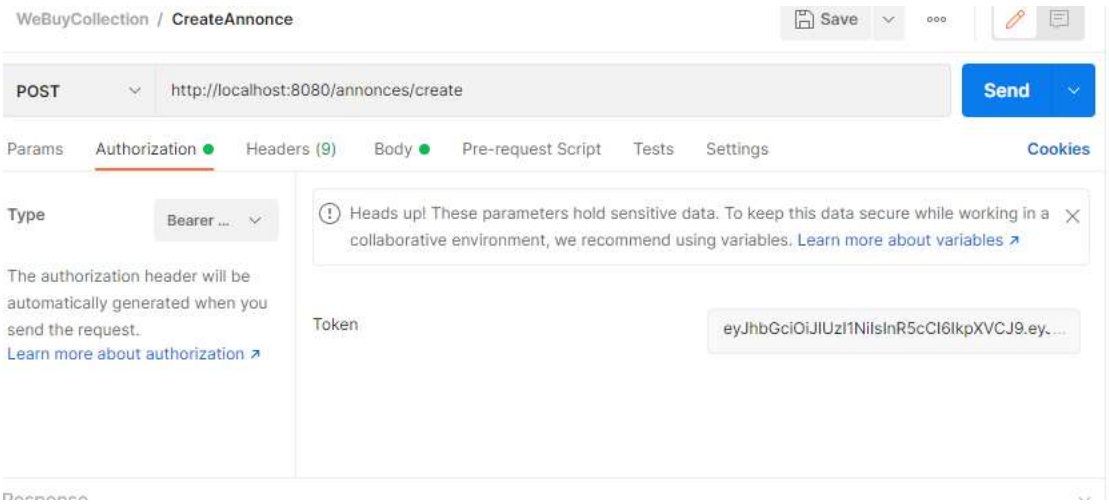
6.2. Tests manuel et unitaire avec Postman

Dans le cadre de notre projet nous avons utilisé de façon répétée Postman. Comme évoqué dans la partie "Méthode de tests", nous avons créé une collection Postman permettant de tester les API Rest de notre projet. Ce point était important, il était la

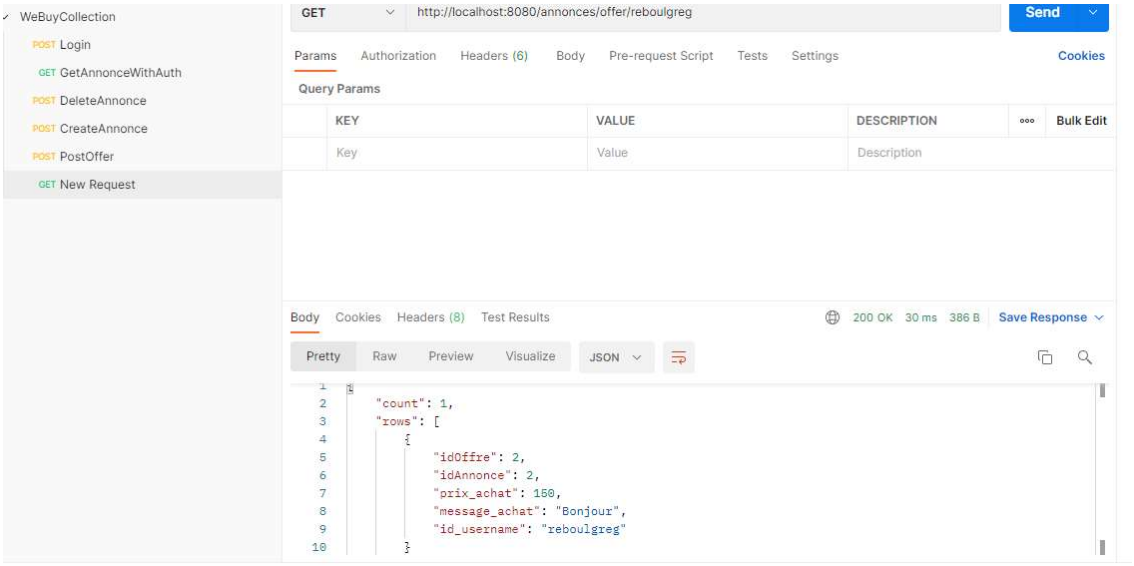
jonction entre nos deux développements. Nous n'avons jamais validé une API rest sans l'avoir testé avec plusieurs jeux de données avec Postman. Voici ci-dessous notre collect Postman :



Grâce à Postman nous avons pu tester de manière simplifiée et rapide le bon fonctionnement de l'authentification de nos utilisateurs par un Bearer Token. En effet, comme vous pouvez le voir ci-dessous, PostMan met à disposition de nombreux paramètres. Ces paramètres, dont le Bearer Token, ont été vraiment utiles dans notre projet. Ils ont permis de définir les erreurs simples que nous pouvions avoir lors de nos tests.



Enfin, le retour des données demandées par Postman étant affichées de façon lisible, il permet de valider facilement les incohérences entre notre base de données et la requête exécutée. Voici un exemple ci-dessous illustrant les données retournées par Postman pour les offres postées par un Utilisateur.

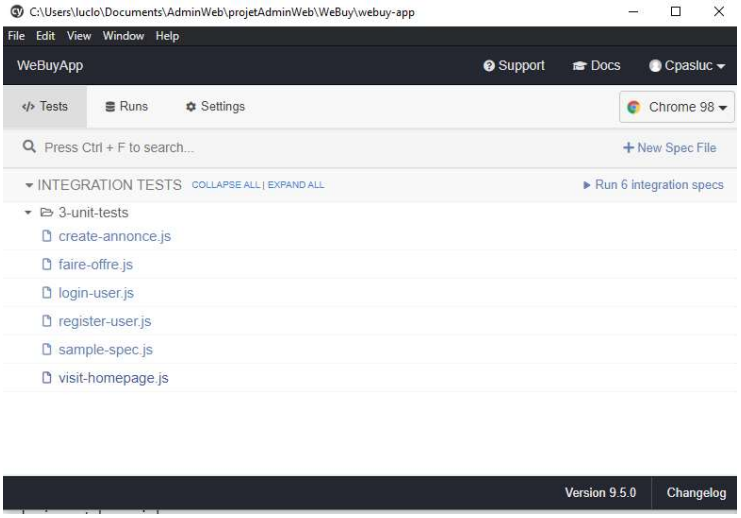


Pour conclure sur Postman, cet outil de test à grandement servi à tester les fonctionnalités BackEnd avant qu'elles soient mises à disposition du FrontEnd. Nous avions choisi de développer chacun séparément une partie du site. "Postman était donc notre interface à nous deux".

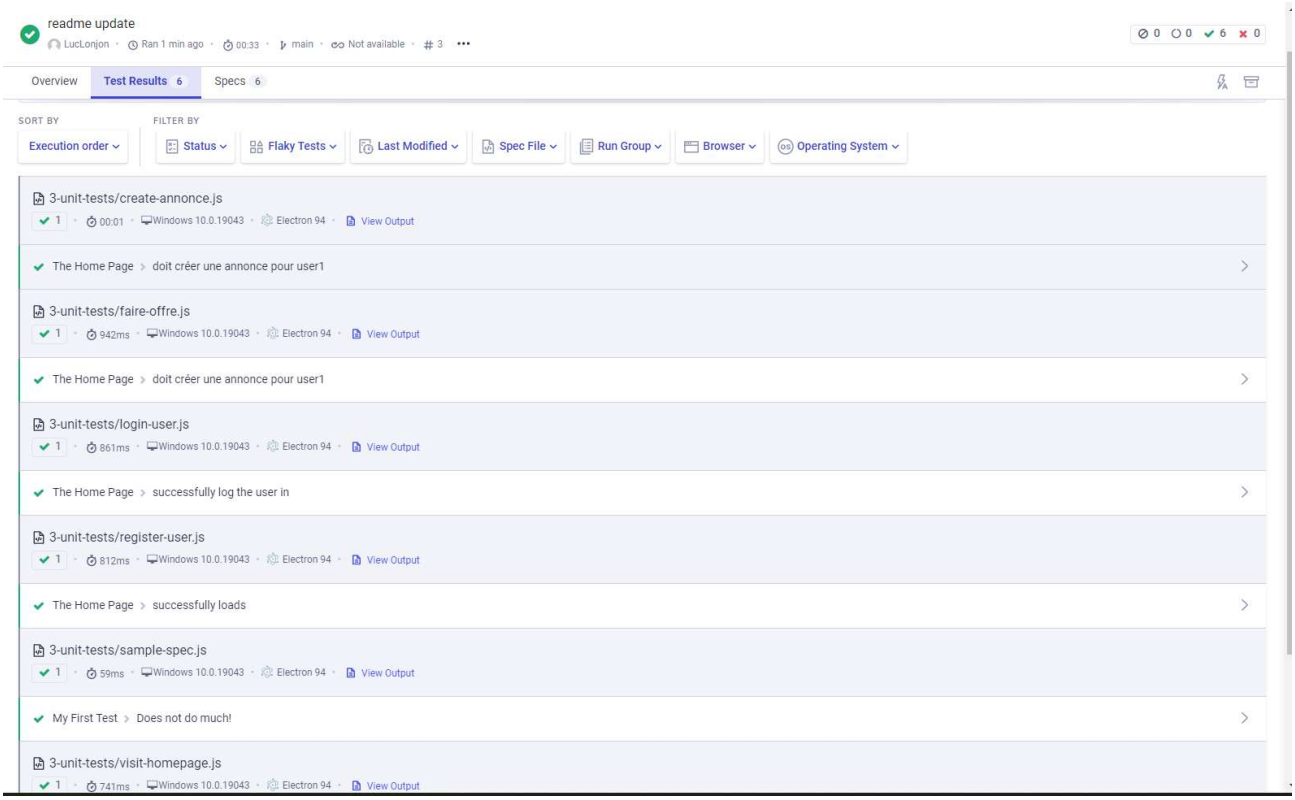
6.3. Tests automatisés avec cypress

Nous n'avions jamais utilisé cypress auparavant et c'est avec grande surprise que nous avons appris à l'utiliser. La prise en main est assez simple nous avons pu mettre en place des tests automatisés rapidement grâce à cet outil.

Nous avons essayé de mettre en place une automatisation de test sur les fonctionnalités les plus basiques tel que ci-dessous :



Nous avons également testé la fonctionnalité de run des tests avec le rapport :



Toutes les fonctionnalités basiques ont permis des tests passants. De plus nous avons aussi essayé de créer des erreurs afin d'observer les informations et l'analyse faites par Cypress, comme ci-dessous :

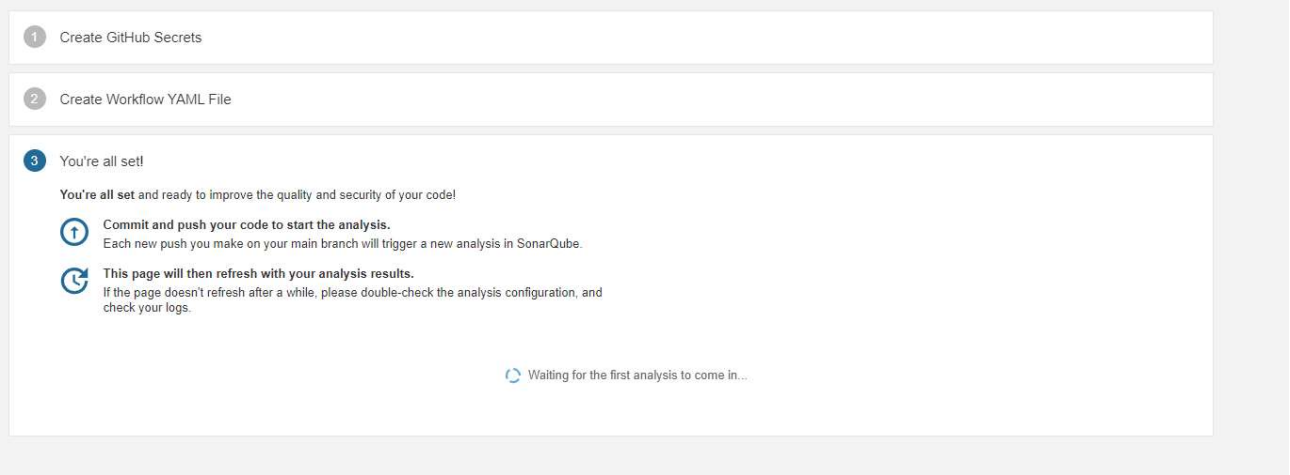
Nous avons pu observer que cypress est outil très puissant et capable d'apporter une force d'analyse de fonctionnalités très intéressante.

7. Rapport SonarQube

Pour SonarQube nous avons tentés de l'utiliser de deux façons différentes :

- Utilisation de la sonde sonarQube en local sur notre projet
- Utilisation en remote directement sur le repository

Malheureusement il semblerait que l'analyse de sonar ne se termine jamais, nous n'atteignons jamais la phase du rapport malgré la connexion au repository :



8. Problèmes rencontrés

8.1. ESlint + Prettify

Nous avions décidé d'utiliser Eslint pour la qualimétrie de notre code plus précisément pour l'uniformité de notre Code. Cependant malgré nos tentatives d'essais nous n'avons jamais réussi à mettre en place totalement EsLint. En effet, nous avons suivi de nombreux tutoriels permettant de paramétrer les deux outils ensemble au sein de notre projet et ide.

```
{
    "env": {
        "browser": true,
        "commonjs": true,
        "es2021": true
    },
    "extends": [
        "standard"
    ],
    "parser": "@typescript-eslint/parser",
    "parserOptions": {
        "ecmaVersion": "latest"
    },
    "plugins": [
        "@typescript-eslint"
    ],
    "rules": {
    }
}
```

Nous avions choisi comme référence de lintage :

```
✓ How would you like to define a style for your project? …
❯ Use a popular style guide
  Answer questions about your style
  Inspect your JavaScript file(s)
```

Qui se base sur les standards du GitHub de Airbnb, malheureusement nous n'avons jamais réussi à implémenter totalement EsLint et Prettify malgré notre bonne volonté. Le Formatting automatique après sauvegarde n'a donc jamais fonctionner.

9. Conclusion

Pour conclure à travers ce projet, nous avons découvert les méthodes tests ainsi que les outils pour les réaliser. Nous sommes conscients que notre méthode de test et l'organisation de celle-ci n'était pas optimale (Réalisation de tests à la fin projet). Cependant nous avons compris la puissance d'outils comme cypress et les possibilités données par EsLint et SonarQube bien que nous n'avons pas pu les intégrer ces deux derniers. Enfin, nous avons via ce cours découvert un standard de cahier de recettes permettant pour nos futurs projets d'avoir un modèle de référence afin de dérouler nos tests.