Formation développeuse web/ web mobile

Dossier de projet



Magali Prost

Titre Professionnel de niveau V

DWWM

Mars 2022/ Octobre 2022

Table des matières

[Table des matières 2](#_Toc115351429)

[Introduction : 3](#_Toc115351430)

[Compétences couvertes par le projet : 3](#_Toc115351431)

[Présentation de l’entreprise 4](#_Toc115351432)

[Travail en méthodologie SCRUM 4](#_Toc115351433)

[Description de la stack de l’entreprise 5](#_Toc115351434)

[Le « clean code » 7](#_Toc115351435)

[Organisation de la team IT 7](#_Toc115351436)

[Organisation des versions 8](#_Toc115351437)

[Architecture de la solution 8](#_Toc115351438)

[Présentation du projet 9](#_Toc115351439)

[Partie Front 10](#_Toc115351440)

[Responsive 12](#_Toc115351441)

[Les tests unitaires 15](#_Toc115351442)

[Partie back 18](#_Toc115351443)

[Les fonctions asynchrones 19](#_Toc115351444)

[la méthode CRUD 22](#_Toc115351445)

[Appel de la DAO 22](#_Toc115351446)

[Les tests unitaires et d’intégration du back 23](#_Toc115351447)

[Lier le back au front 26](#_Toc115351448)

[Résultat final 26](#_Toc115351449)

[Git 27](#_Toc115351450)

[Pipeline Gitlab 28](#_Toc115351451)

[Situation de travail 29](#_Toc115351452)

[Extrait du site anglophone 30](#_Toc115351453)

[Traduction 31](#_Toc115351454)

[Sécurité 32](#_Toc115351455)

[Conclusion 33](#_Toc115351456)

[Remerciements 33](#_Toc115351457)

[Annexe 1 : organisation du service IT 34](#_Toc115351458)

[Annexe 2 : les différentes opérations de CRUD sur le projet 35](#_Toc115351459)

# Introduction :

À la suite d’une reconversion professionnelle, j’ai suivi la formation de développeuse web/web mobile au sein de l’Afpa d’Evreux. J’ai effectué mon stage en semi-présentiel dans la start-up Data Soluce dont le service IT est situé à Toulouse. J’ai été encadré par Francois-Xavier Lafont, software architect et Laurent Yague développeur sénior et scrum master.

En télétravail 3 jours par semaine, j’ai appris à travailler en équipe, mais aussi à être autonome ce qui est à mon sens une composante essentielle du métier de développeur web et web mobile.

Pendant ce stage, j’ai travaillé sur un composant react ainsi que son back-end pour afficher le nombre de projets et d’utilisateurs par plateforme. J’ai également réalisé les tests unitaires et d’intégrations de ce projet.

# Compétences couvertes par le projet :

Développer la partie front-end d’une application web ou web mobile en intégrant les recommandations de sécurité :

- Réaliser une interface utilisateur web statique et adaptable

- Développer une interface utilisateur web dynamique

Développer la partie back-end d’une application web ou web mobile en intégrant les recommandations de sécurité :

- Développer les composants d’accès aux données

- Développer la partie back-end d’une application web ou web mobile

# Présentation de l’entreprise

Data Soluce est une start-up dont le service IT est situé à Toulouse et qui travaille dans le domaine de la modélisation de projets immobiliers. Le secteur de l’immobilier et de la construction entame aujourd’hui sa transition digitale.

La mission de Data Soluce : fournir aux organisations une technologie pour permettre cette transition, pour que chaque nouveau projet soit l’occasion d’apprendre davantage et d’être encore plus précis et exhaustif pour imaginer ou gérer le projet suivant.

C’est désormais au tour de l’immobilier d’industrialiser toute la chaîne de valeur d’un bien, en réunissant sur la plateforme tous les acteurs autour d’un « double digital » optimisé sur l’ensemble du cycle de vie du bien.

# Travail en méthodologie SCRUM

La méthodologie SCRUM qu’est-ce que c’est ?

Scrum est [l'infrastructure agile](https://www.journaldunet.fr/web-tech/guide-de-l-entreprise-digitale/1443838-methode-agile-definition-comparatif-et-avantages/) la plus utilisée. A l'instar d'autres méthodes agiles, Scrum est une démarche de gestion de projet qui fait du client (ou utilisateur) le principal pilote de l'équipe en charge des développements.

Historiquement, elle est principalement mise en œuvre dans le domaine informatique, et dans celui du développement d'applications en particulier. Mais, elle est aussi utilisée de plus en plus dans d'autres domaines de l'ingénierie produit.

Scrum recommande de nommer ce qu'elle appelle un scrum master. Son rôle est de garantir la mise en œuvre du framework agile et de piloter les quatre étapes d'[un sprint scrum](https://www.journaldunet.fr/web-tech/guide-de-l-entreprise-digitale/1443836-sprint-scrum/) : plannification, daily meeting, sprint review et sprint rétrospective. Elément central pour le bon fonctionnement de l'équipe projet, le scrum master est aussi le garant de la fluidité des échanges et de la productivité du travail. A ce titre, il identifie les points de blocage et anime les brainstormings pour cerner les solutions. Enfin, il rédige le graphique d'avancement ([burndown chart ou BDC](https://fr.wikipedia.org/wiki/Burndown_chart)) qui décrit le volume de tâches restant à réaliser sur l'axe vertical et le timing projeté sur l'axe horizontal.

Chez Data Soluce, un sprint dure 2 semaines (soit 10 jours ouvrés) et ils utilisent l’application Confluence et Jira pour la planification et la traçabilité. Le premier jour du sprint, le scrum master planifie le sprint rétrospective qui présente à l’équipe de développement et au product owner, les dernières fonctionnalités, correction de bugs et actions amenées lors du sprint précédent. C’est aussi le moment de faire un bilan avec le Burndown chart et trouver des solutions d’ordre organisationnelle.

Lors du sprint planning, les product owners présentent les différentes users stories (US) et l’équipe de développement définit le temps attribué à chaque US avec un poker planner.

Ensuite, pendant les 9 jours ouvrés restants les développeurs travaillent sur les US qui leur sont attribuées et organise un daily tous les jours à 9h44 pour expliquer ce qui a été fait la veille, ce qui est prévu d’être fait ce jour et les difficultés rencontrées pour trouver une solution rapidement et atteindre les objectifs du sprint dans les meilleures conditions.

# Description de la stack de l’entreprise

Le service IT de Data Soluce utilise le système d’exploitation Linux. Nous avons essayé de lancer le projet sur mon ordinateur personnel, mais à la suite de problèmes de compatibilités et de sécurité, l’entreprise m’a prêté un ordinateur professionnel sous Linux. Grande nouveauté pour moi qui ai appris à coder sous Windows.

Toute l’équipe utilise GitLab qui est une plateforme de développement collaborative open source éditée par la société américaine du même nom. Elle couvre l'ensemble des étapes du DevOps. Se basant sur les fonctionnalités du logiciel Git, elle permet de piloter des dépôts de code source et de gérer leurs différentes versions.

L’éditeur de texte est Visual Studio Code. Nous travaillons dans des containers Docker qui est une structure de développement d’applications ouverte conçue pour faciliter le travail des DevOps et des développeurs.

Docker est une plate-forme de containerisation qui regroupe l'application et ses dépendances dans un container afin que l'application fonctionne de manière transparente dans n'importe quel environnement, qu'il s'agisse de développement, de mise en scène ou de production.

Il s'agit d'un outil conçu pour faciliter la création, le déploiement et l'exécution d'applications à l'aide de containers. Les containers Docker sont légers, alternatives aux machines virtuelles et ils utilisent le système d'exploitation hôte ce qui permet de ne pas à pré-allouer de RAM dans les containers comme on le fait dans les machines virtuelles.

Grâce à Docker, les développeurs peuvent créer, packager, livrer et exécuter des applications en toute simplicité, sous forme de containers légers, portables et autonomes, qui peuvent fonctionner pratiquement n’importe où. Les containers permettent aux développeurs de conditionner une application avec toutes ses dépendances et de la déployer en une seule unité. En fournissant des containers d’applications prédéfinis et autonomes, les développeurs peuvent se concentrer sur le code de l’application et l’utiliser sans se soucier du système d’exploitation sous-jacent ou du système de déploiement.

Dans le cadre de mon stage, j’avais accès à 4 projets : le front-end, le back-end, la Gateway et mongoDB. Pour ouvrir les containers, la ligne de commande est « docker-compose up », pour le fermer « docker-compose down », pour vérifier qu’ils fonctionnent « docker ps ».

Pour la communication interne de l’entreprise, j’avais une boite mail thunderbirds avec une adresse mail interne à l’entreprise. Nous utilisions aussi Slack pour les conversations professionnelles, les partages d’écran et les différentes questions que l’on devait poser au product owner.

Les différents langages informatiques utilisés pendant ce stage sont :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Système d’exploitation** | **Langage de développement** | **Bibliothèque FRONT-END** | **Bibliothèque**  **BACK-END** | **Serveur Http** |
| Linux  Docker | Javascript | React  Redux  Axios  Jest | Hapi  Joi  MongoDB driver  Jest | NodeJs  Nginx |

Nodejs est un outil libre codé en Javascript et orienté pour des applications en réseau. Ce n’est ni un serveur ni un framework. C’est un environnement d’exécution JavaScript pouvant être utilisé pour écrire des services côté serveur appelés API (Application Programming Interface). Il constitue une alternative aux langages serveur comme PHP, Java EE, etc.

MongoDB est une base de données NoSQL orientée document. Elle se distingue des bases de données relationnelles par sa flexibilité et ses performances. Contrairement à une base de données SQL traditionnelle, MongoDB ne repose pas sur des tableaux et des colonnes. Les données sont stockées sous forme de collections et de documents. Les documents sont des paires de valeurs / clés servant d’unité de données de base. Les collections quant à elles contiennent des ensembles de documents et de fonctions. Elles sont l’équivalent des tableaux dans les bases de données relationnelles classiques.

Jest, rendue open-source, est une librairie de test JavaScript ayant énormément gagné en popularité depuis sa mise en libre circulation. Conçue pour fonctionner aussi bien sur du JavaScript côté navigateur (front-end) que côté serveur (back-end), Jest a su s’imposer par sa rapidité d’exécution des tests, son API complète et sa facilité d’installation. Sa documentation complète et bien maintenue en fait la librairie la plus populaire pour tous les différents tests JS. Jest est sans conteste l’un des outils nodeJS les plus incontournables pour un développeur Javascript.

Redux est une implémentation dérivée de Flux qui permet de créer un Store contenant un état, réagissant à des actions dispatchées et auquel on peut souscrire pour être notifié des changements. Il permet également l'ajout de middlewares qui peuvent en quelque sorte pre-process les actions.

Axios est un client HTTP basé sur les promesses compatibles avec nodeJS et les navigateurs. Il est isomorphique (c’est à dire qu’il peut opérer dans le navigateur et dans node.js avec le même code). Côté serveur, il utilise le module natif http de node.js, et côté client (navigateur) il utilise les XMLHttpRequests.

# Le « clean code »

Le Clean Code a été développé par Robert Cecil Martin qui a inventé ce terme dans son livre « Clean Code: Rafactoring, Patterns, Testen und Techniken für sauberen Code »pour décrire un code écrit de façon propre. Les principes du Clean Code sont toutefois nettement plus anciens et ne trouvent pas leur origine dans le développement logiciel. Je vous explique ci-dessous ce qui caractérise un code propre, quels sont ses avantages et comment vous pouvez écrire du Clean Code.

Le Clean Code n’est pas un ensemble de règles strictes mais désigne plutôt une série de principes pour produire un code compréhensible de façon intuitive et facile à modifier. Compréhensible signifie dans ce cas un code immédiatement intelligible par n’importe quel développeur qualifié. Les caractéristiques suivantes augmentent la lisibilité du Clean Code :

* Le processus d’exécution de l’ensemble de l’application est logique et structuré de façon simple.
* Le rapport entre les différentes parties du code est transparent.
* La tâche ou le rôle de chaque classe, fonction, méthode et variable est immédiatement compréhensible.

Un code est facile à modifier lorsqu’il peut être facilement ajusté et complété. À l’inverse, il est également plus simple de corriger les erreurs dans le code. Par conséquent, le Clean Code est très facile à entretenir. Un code facilement modifiable comporte les attributs suivants :

* Les classes et les méthodes sont petites et, dans la mesure du possible, ont une seule et unique tâche.
* Les classes et les méthodes sont prévisibles, fonctionnent de la façon attendue et sont accessibles au grand public grâce à des API bien documentées (interfaces).
* Le code dispose de tests d'unité.

Les avantages d’une telle programmation sont évidents : le Clean Code n’est pas dépendant du développeur initial. En principe, n’importe quel programmeur peut ainsi travailler avec un tel code. Ceci permet par exemple d’éviter les problèmes survenant lorsque l’on travaille avec un legacy code. L’entretien du logiciel s’en trouve également facilité puisque les bugs peuvent être trouvés et corrigés facilement.

# Organisation de la team IT

Deux pôles : BUILD et RUN.

BUILD : équipe de développement (Scrum)

RUN : équipe de gestion de l'infrastructure (Kanban)

Laurent : Scrum master, Architecte suppléant, Devops

Ludovic : Architecte suppléant, Devops

François Xavier : Architecte référant, Devops

(Voir annexe 1 organisation de la team IT)

# Organisation des versions

Chaque composant développé pour la plateforme suit un versionning de la manière suivante.

La branche principale du répertoire git est master. Chaque développement se fait sur une branche dédiée. Lorsque nous mergeons la branche de développement, celle-ci ne doit comporter qu'un seul commit détaillant les travaux réalisés.

Tout commit est éligible à la production. Dès qu'un nouveau commit est ajouté sur master, l'environnement d'intégration continue lance automatiquement les outils de qualité. Si les résultats sont corrects, alors une version temporaire est déployée sur l'environnement continuous automatiquement.

Les outils de qualité sont :

- linter: analyse de code statique, permet de vérifier que les règles de codage sont respectées, les normes de sécurité (OWASP) aussi.

- tests automatiques : test unitaire et test d'intégration, la couverture doit être à 100%.

Lorsque le produit souhaite le déploiement des travaux réalisés, une version est créée via un tag sur la branche master. Cette version est déployée manuellement sur l'environnement de préproduction pour effectuer les dernières vérifications (TNR). Si tout est ok, alors un déploiement en production est planifié.

# Architecture de la solution

La solution est une application web. Elle est composée d'un client web, d'un reverse proxy, d'un serveur de fichiers statiques, d'un serveur d'API et d'un serveur de base de données.

Le client web est appelé le frontend, c'est un client javascript se basant sur ReactJS.

Le reverse proxy est appelé ds-gateway, son rôle est d’offusquer les composants de la plateforme. Il sait rediriger les flux suivant l'url appelée.

Le serveur de fichiers statiques est appelé ds-frontend, il permet de déployer les fichiers du client web.

Le serveur d'API est appelé ds-backend, c'est un serveur javascript basé sur nodejs et hapijs.

Le serveur de base de données est appelé ds-mongodb, un serveur mongodb.

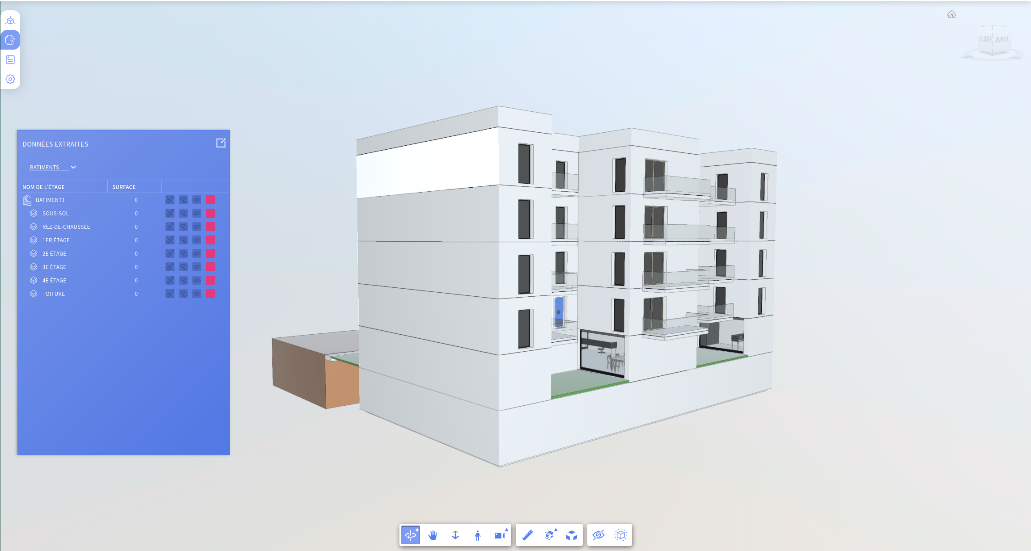
L'utilisateur se rend sur l'url de sa plateforme et télécharge les sources de l'application web. De là, l'application web communique au serveur d'api afin de faire les opérations CRUD sur les entités du domaine.

La communication entre les composants (serveurs) est sécurisée via le protocole de communication HTTPS.

# Présentation du projet

La plateforme Data Soluce permet la modélisation 3d de projets immobiliers sans installation de logiciel dédié, possède aussi une GED ou gestion électronique des documents relatifs aux projets et réunit les différents acteurs autour de leur plateforme (par exemple chef de projet, architecte, et les différents métiers du bâtiment).

Cela permet à de grands groupes immobiliers de centraliser tous leurs projets et leurs données.



Pour ma part, j’ai travaillé en front et en back, sur le dashboard plateforme. J’y ai créé un composant qui permet d’afficher un compteur d’utilisateurs et de projets par plateforme.

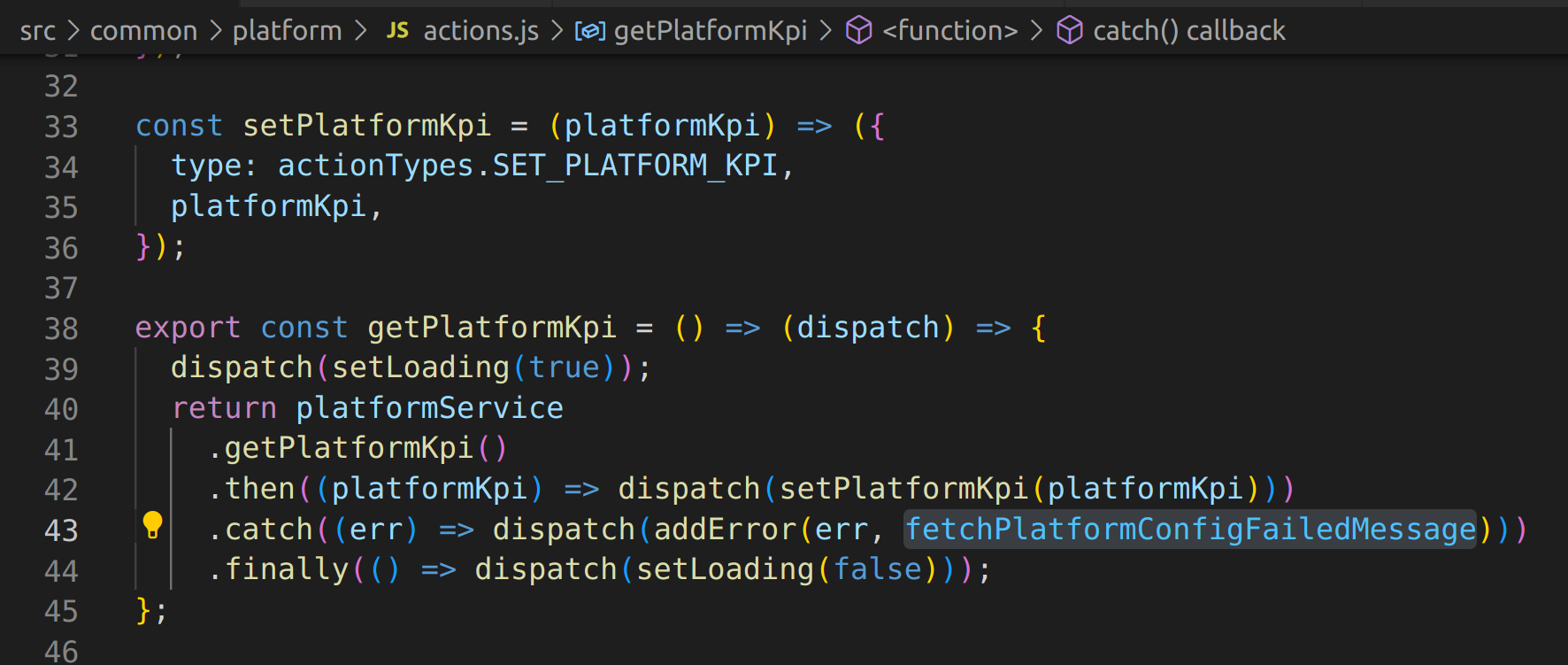
# Partie Front

J’ai commencé par la partie Front-end pour bien comprendre l’architecture du projet. Ma tâche sur la 1ere période de stage a été de concevoir un composant react qui affichait le nombre de projets et d’utilisateurs. J’ai d’abord commencé par créer une route API sans back avec des données fictives pour construire ce composant.

Une image contenant texte

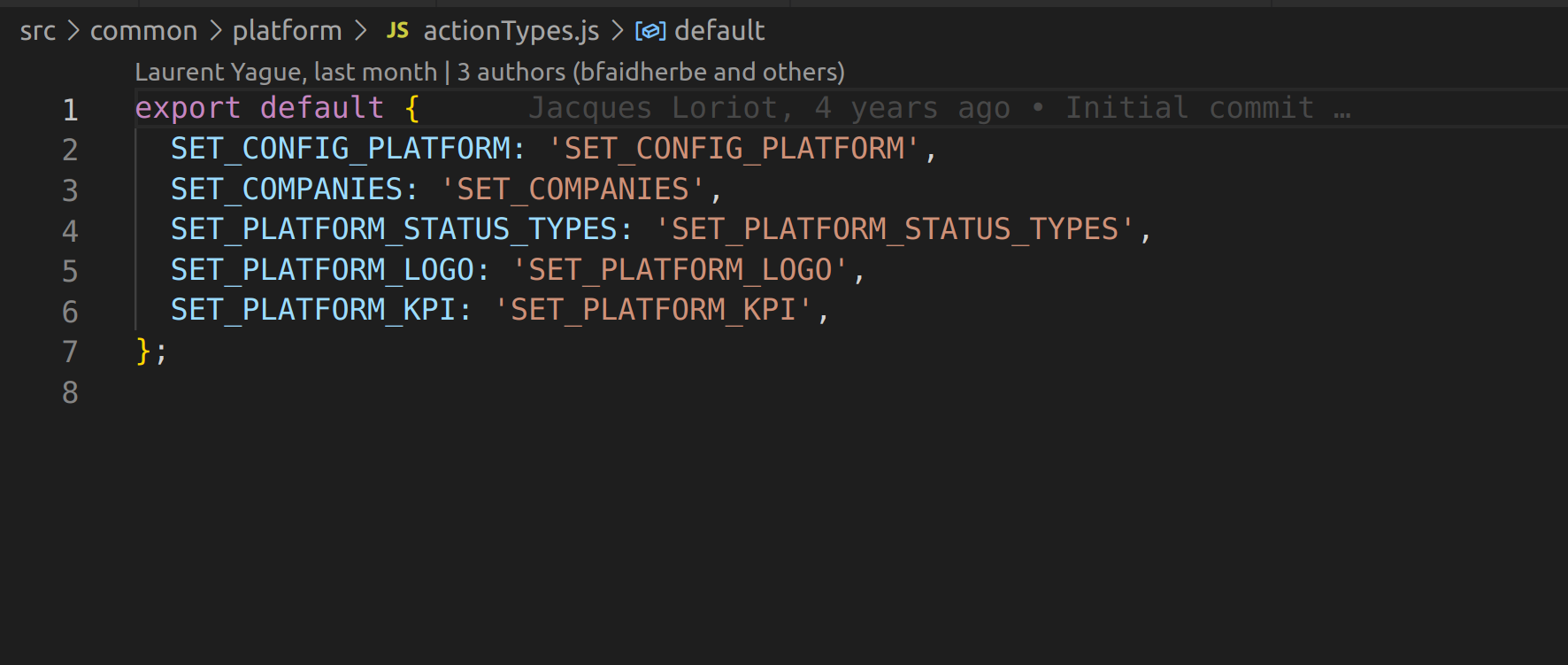
Description générée automatiquement

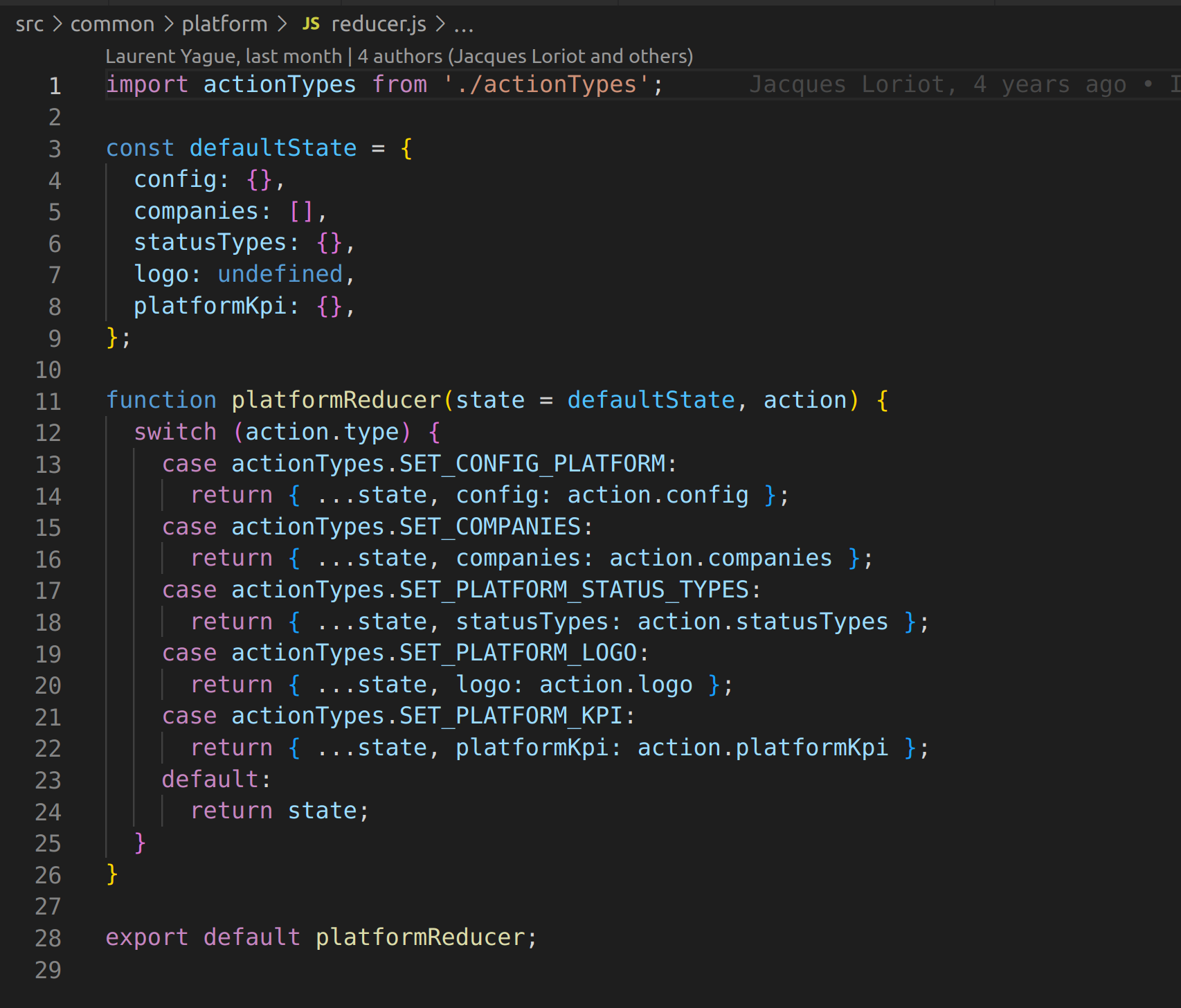
Ensuite, il a fallu créer les actions sur le state de cette plateforme KPI :



Ici, nous avons les actions get et set platformKpi. Le set permet de stocker la valeur de platformKpi et le get permet de renvoyer l’action du set si le chargement est effectué ou de renvoyer une erreur platform failed quand le chargement de la plateforme est en false.

Ensuite, j’ai mis mon set platform kpi dans l’export de mon action type pour pouvoir l’importer dans le reducer.

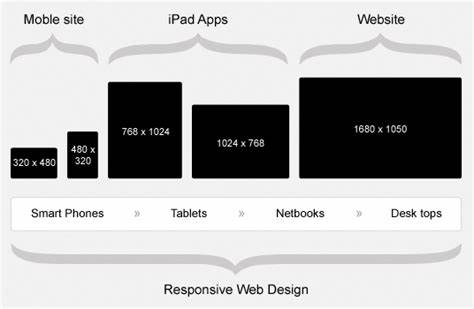




On donne à platformKpi une valeur vide dans la const defaultState et dans la fonction platformReducer on dit que state est égale defaultState, ce qui est l’état initial avant l’action mise en place. Dans ce switch, chaque action est un cas. Dans le cas de mon action, je veux qu’il retourne que platformKpi a subi l’action set\_platform\_kpi.

Il n’y a plus qu’à créer le composant.

# Responsive



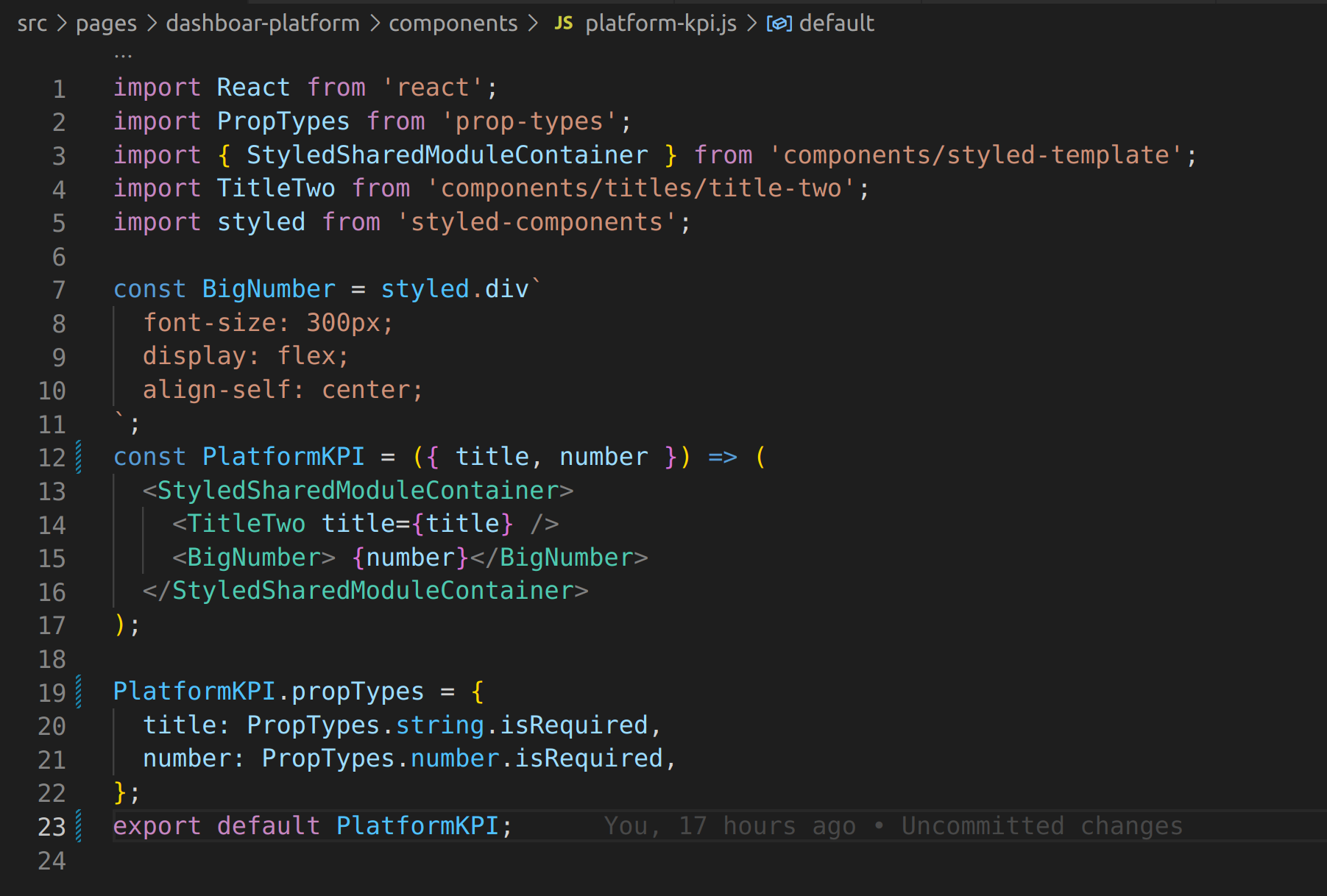
Le Responsive Design ou plus précisément le Responsive Web Design (RWD) est une technique de conception d’interface digitale qui fait en sorte que l’affichage d’une quelconque page d’un site s’adapte de façon automatique à la taille de l’écran du terminal qui le lit. Il est différent de l’Adaptative Design bien que les deux concepts aient pour but d’améliorer l’ergonomie mobile du site web.

Dans le cadre de ce projet, tous les composants sont prédéfinis dans un dossier component pour une harmonisation de tous les éléments de l’application. Ils ont été créés pour être tous responsive.

Une image contenant texte

Description générée automatiquement

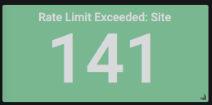
Dans ce projet il n’y a pas de fichiers .css. Ce sont uniquement des exports de composants. Je n’avais jamais vu cette manière de coder.



Dans ce projet, j’ai utilisé react. Il a donc fallu importer react pour que cela fonctionne. On importe aussi les PropTypes qui permettent de valider le type de title et number. Ici, nous voulons absolument que title soit une string et number, un nombre. Cela permettrait d’afficher le message d’erreur défini si ce n’était pas respecté.

Comme il s’agit d’un projet volumineux, et que le design était déjà défini par la UX UI designer, j’ai importé 2 éléments de styles dans mon composant (le StyledSharedModuleContainer et le TitleTwo) pour respecter la charte graphique de la plateforme.

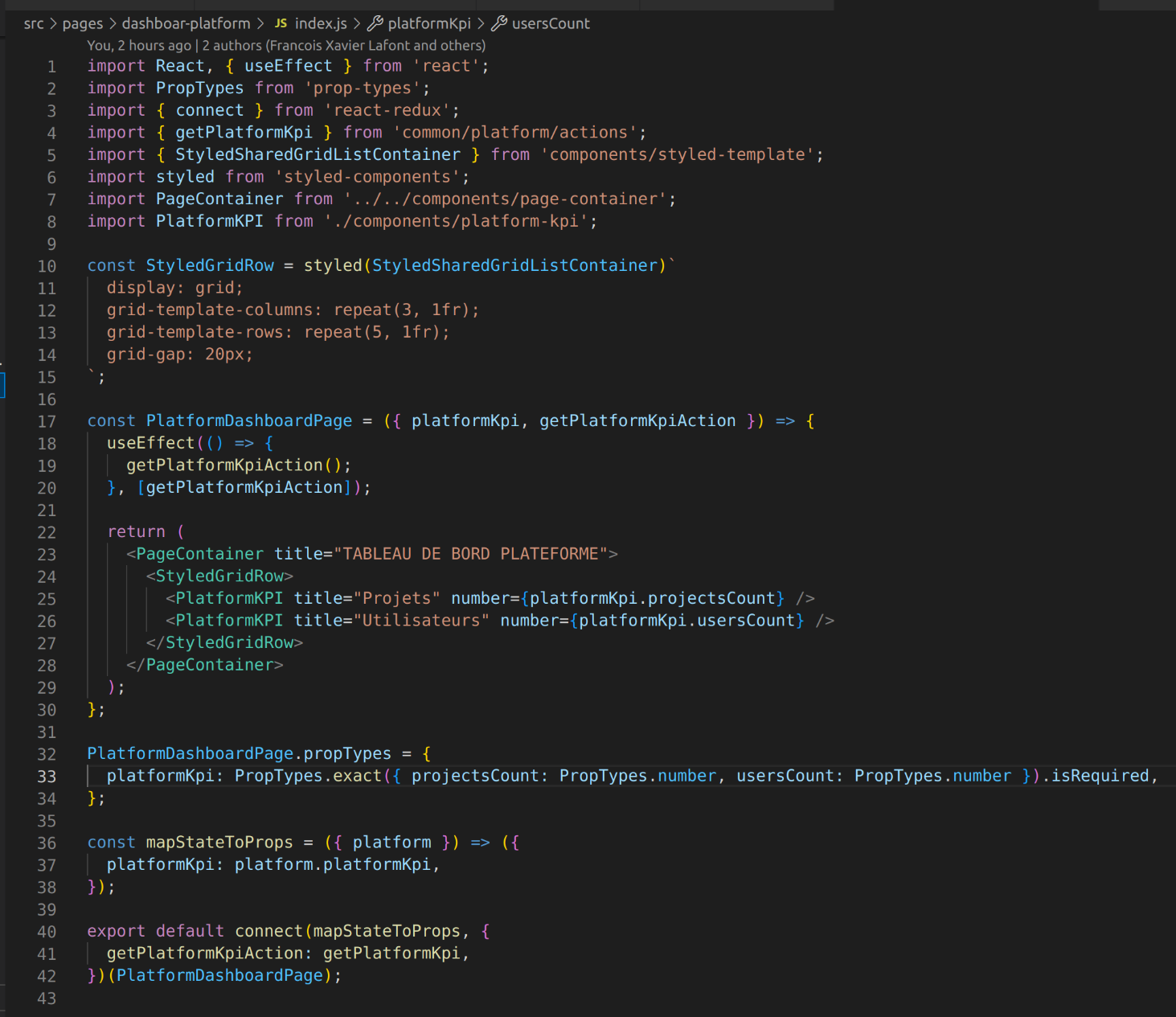
Pour le compteur, mon tuteur voulait un module ressemblant aux modules de grafana (plateforme d’observabilité open-source permettant de visualiser les métriques, les logs et les traces collectés depuis les applications pour les administrateurs système) avec le design de leur plateforme.



(Exemple de module de grafana)

J’ai donc importé styled pour pouvoir créer BigNumber. J’ai augmenté la taille de la police pour que le chiffre soit imposant et pour le centrer, je lui ai donné un display flex avec un alignement center.

Il est temps d’implémenter mon composant au sein de l’index dashboar-platform.



Ici on peut voir que j’ai modifié le style de StyledSharedGridListContainer car il ne voulait pas s’afficher sous forme de tableau. On rappelle ensuite les actions qui vont figurer sur cette page, et on décrit comment est agencée la page.

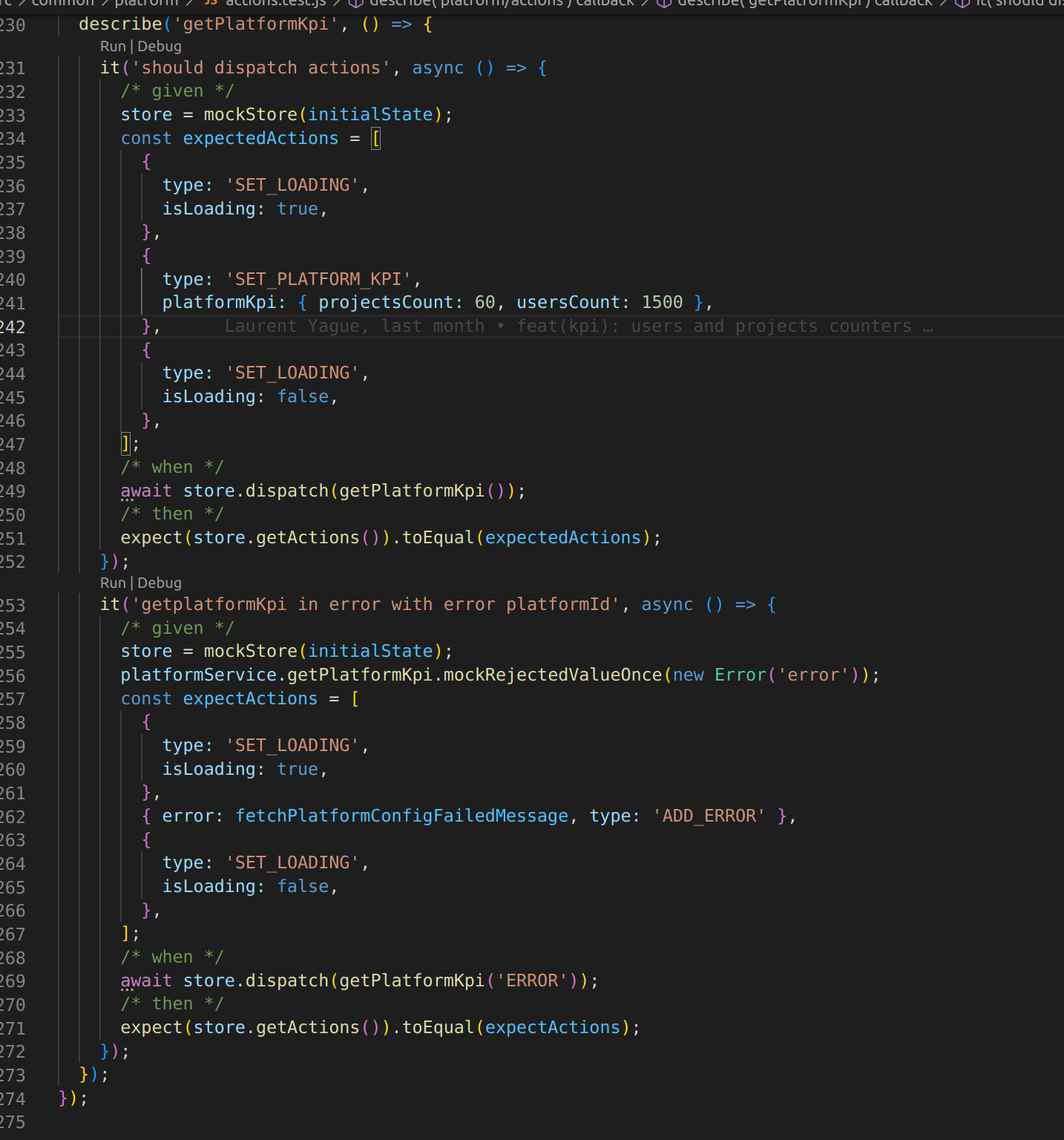
On donne le titre à notre composant et la valeur de number. On rappelle encore les propTypes.

# Les tests unitaires

En programmation, le **test unitaire** est un procédé permettant de s'assurer du fonctionnement correct d'une partie déterminée d'un logiciel ou d'une portion d'un programme. Il s'agit pour le programmeur de tester un module, indépendamment du reste du programme, ceci afin de s'assurer qu'il répond aux spécifications fonctionnelles et qu'il fonctionne correctement en toutes circonstances.

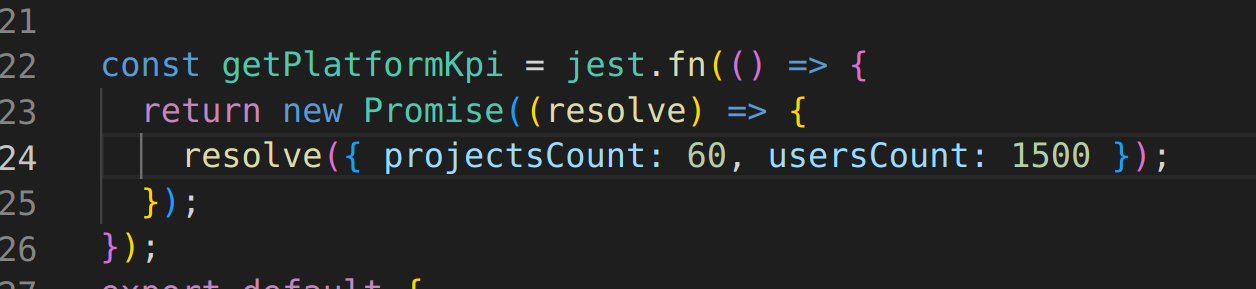
Chez Data Soluce, nous avons utilisé le framework Jest. Le test se compose ainsi :

* Describe : pour décrire la fonction que l’on va tester
* It : pour décrire le comportement que l’on attend de la fonction
* Given : donner un contexte
* When : action faite par la fonction
* Then : le résultat



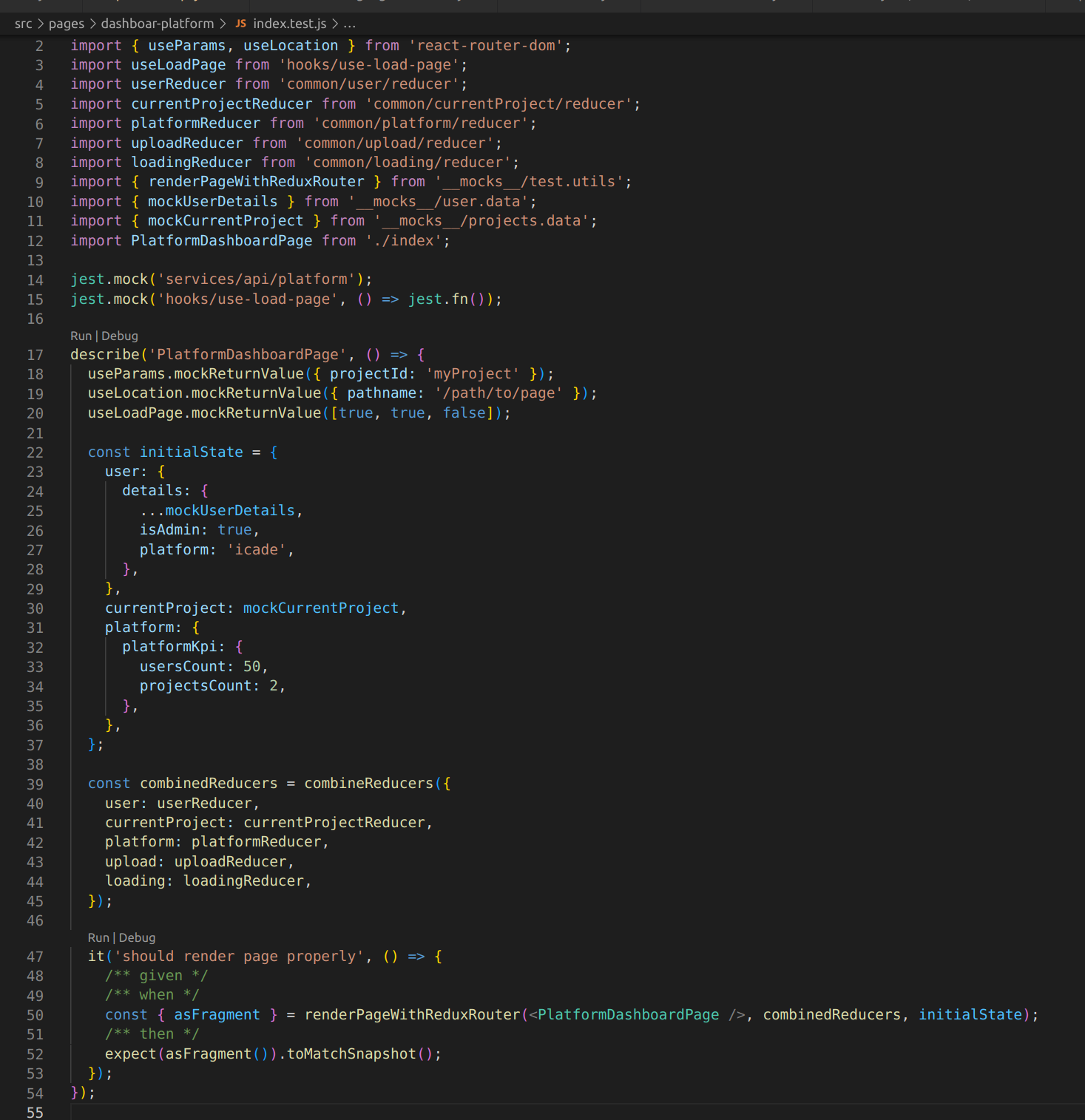
Ici, on décrit getPlatformKpi dans actions.test. On veut tester les deux conditions : si cela fonctionne et le cas d’erreur.

Dans le premier it, on lui donne une valeur fictive avec le mock définit ci-dessous. On veut vérifier que l’action fonctionne correctement.



On lui donne les actions attendues avec les valeurs du mock. Quand on envoie getPlatformKpi, on attend que ce soit égal à l’action attendue.

Enfin on décrit le cas d’erreur. Cette fois, avec la ligne 56 on lui demande de renvoyer une erreur sur la platformService. On lui redonne les actions attendues et on attend qu’il renvoie une erreur.

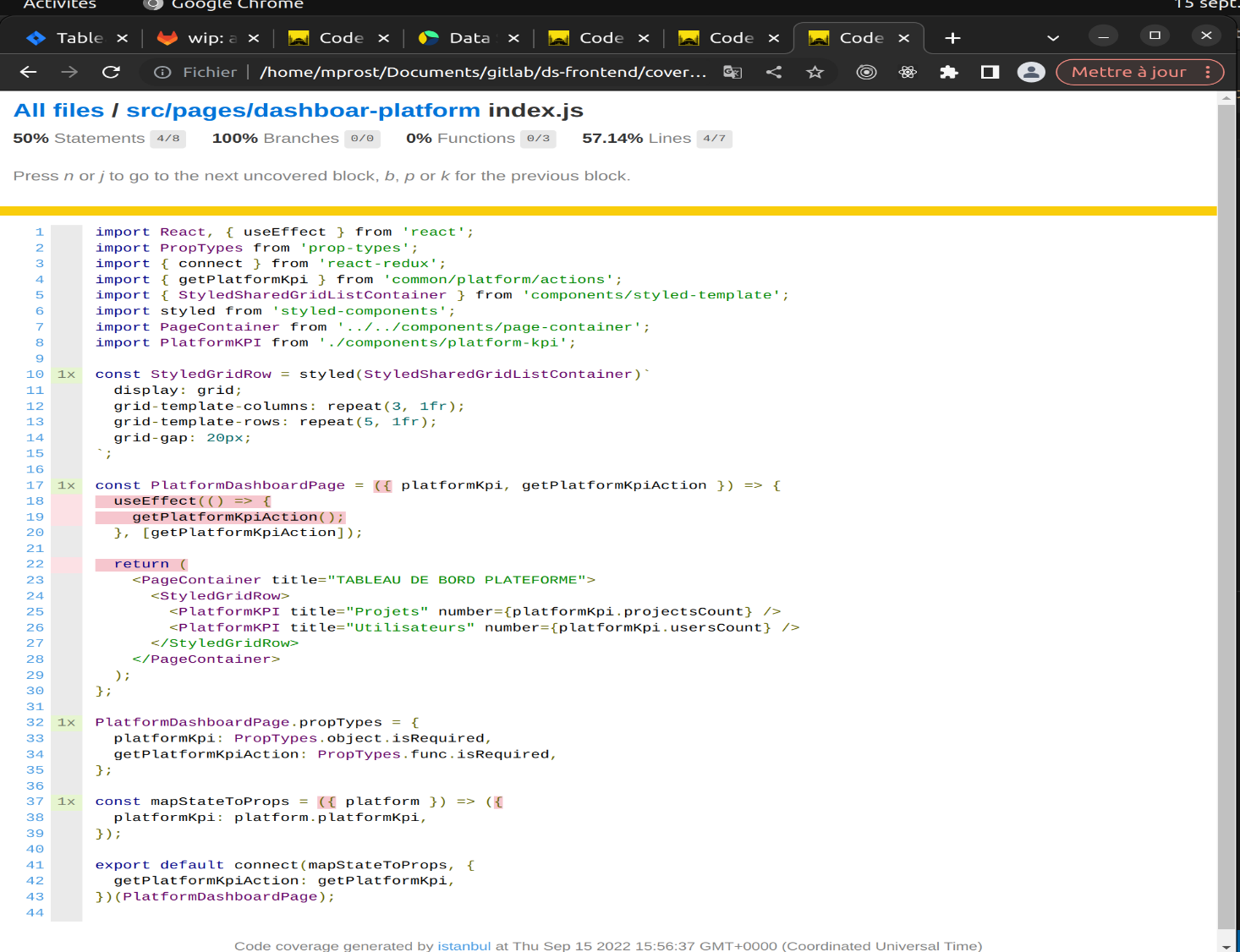


Ici, on fait le test unitaire de l’index du dashboard platform. On rappelle le mock de la plateforme. Encore une fois, on décrit la page, on appelle le mock des paramètres, de la localisation de la page et du chargement de page. On lui donne des données dans son état initial. On appelle les reducers dont on a besoin sur cette page.

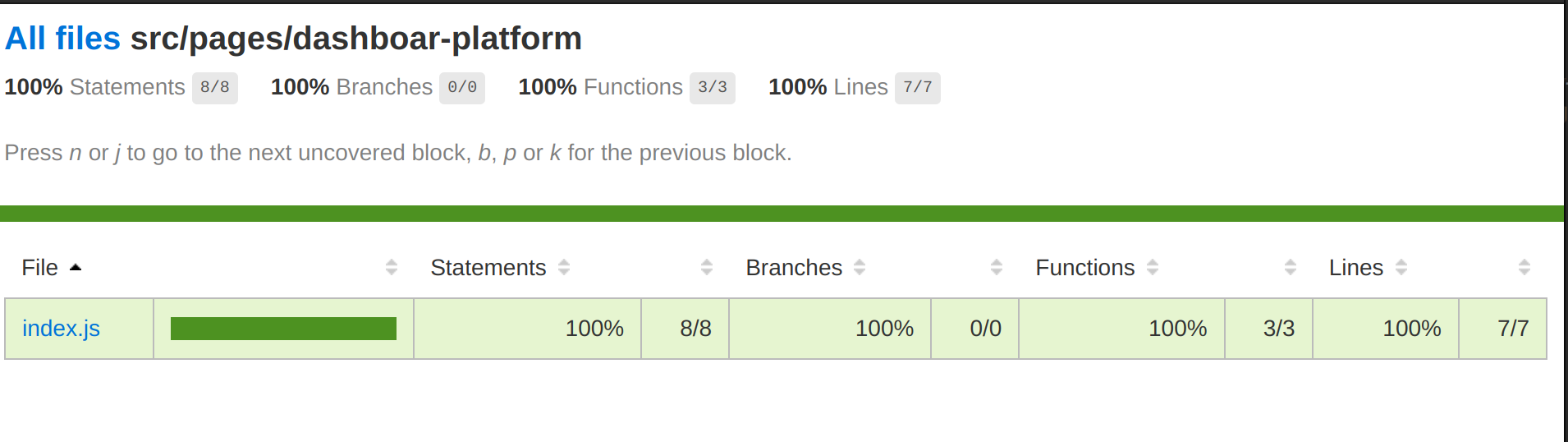
Dans le it, on dit que la platformDashBoardPage devrait rendre une page qui doit être comme le snapshot. Un**snapshot informatique, ou instantané**de**stockage, peut être décrit comme une « photo » des données stockées sur un**système de stockage de données**à un instant précis. Cela permet une récupération rapide des données en cas d’erreur, d’effacement d’un fichier ou de corruption de données liées à une défaillance d’un programme,** par exemple.

**Pour finir, il faut lancer les tests. J’ai utilisé la commande « yarn unit-tests --coverage -u ». Cela permet de lancer les tests (yarn unit-tests), vérifier la couverture des tests (--coverage) et d’actualiser le snapshot (-u pour udpate).**

On vérifie que toutes les lignes de notre code soient couvertes.



Ici, on voit que les lignes 18, 19, 22 et 37 sont encore à couvrir.



Quand tout est testé, la couverture passe à 100%.

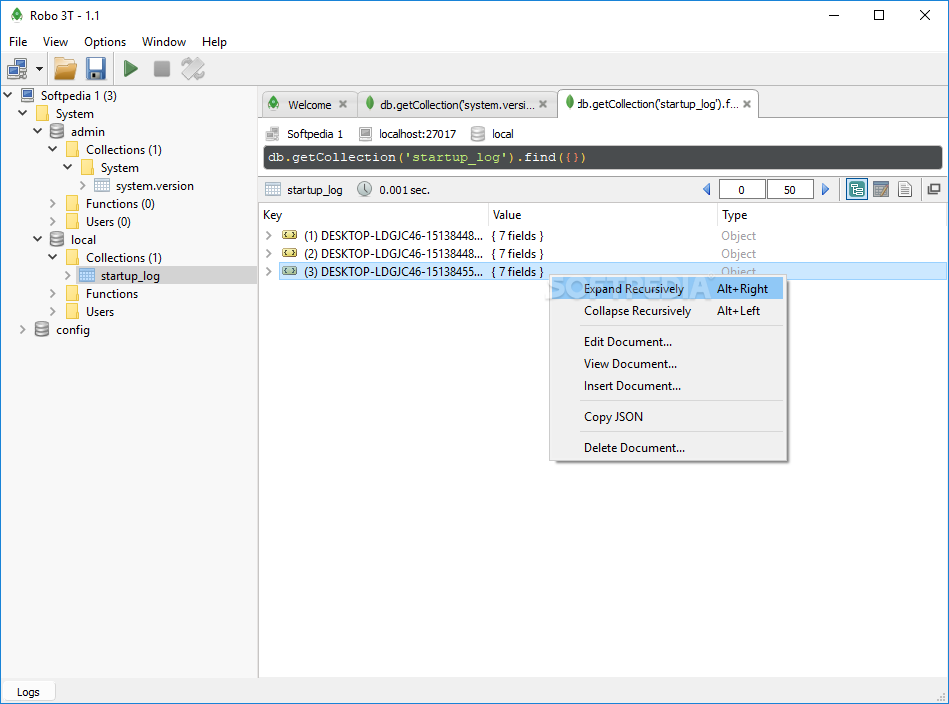
# Partie back

Data Soluce utilise mongoDB en base de données. Il a donc fallu que j’installe robo 3T pour avoir accès à la base de données et pouvoir requêter dans le shell.

Robo 3T est un outil qui vise à simplifier considérablement l'utilisation des bases de données MongoDB. Il fournit une interface graphique conviviale pour la gestion des documents et la modification de leur contenu.

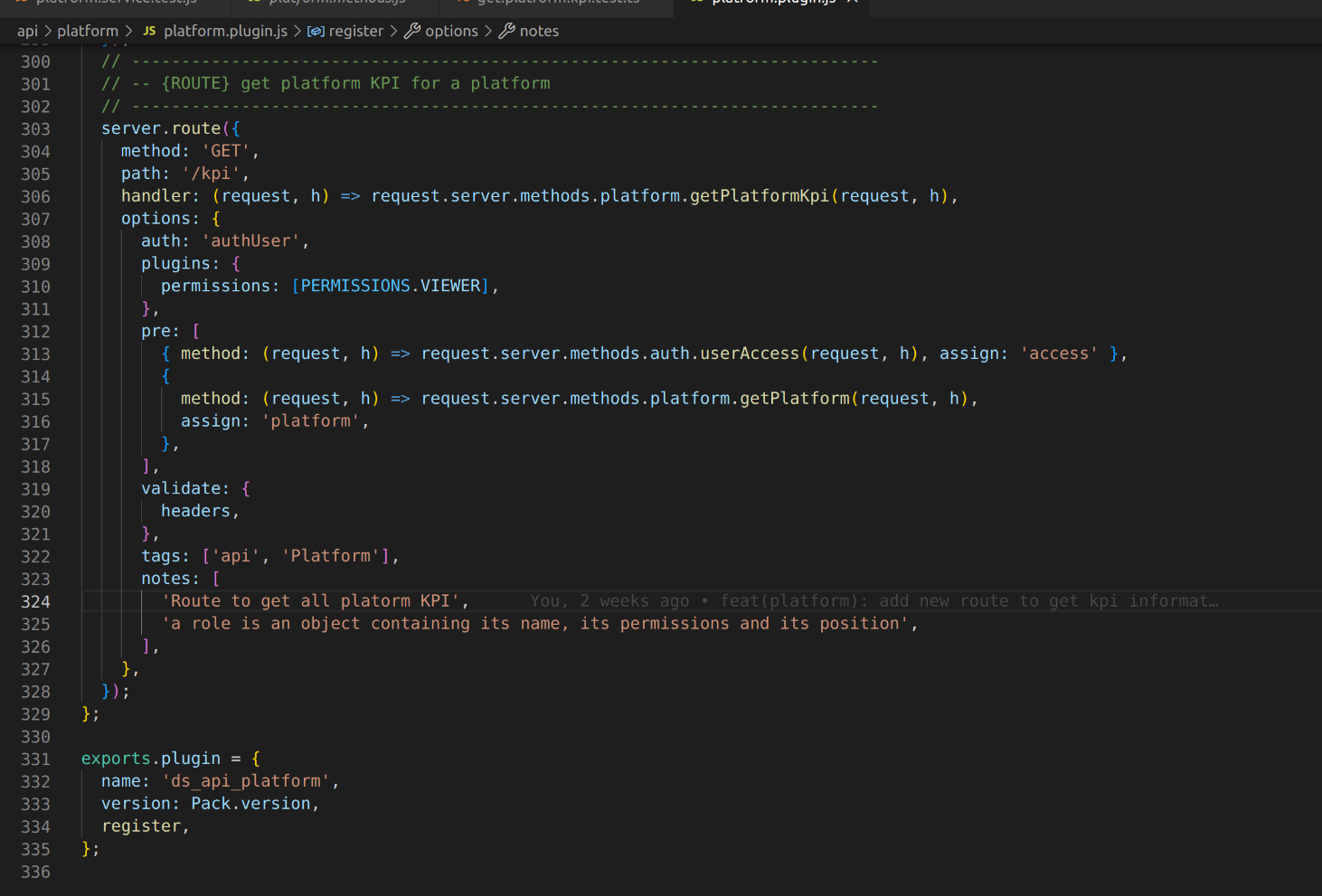
Le serveur MongoDB doit être en cours d’exécution et attendre que les connexions configurent une nouvelle connexion dans Robo 3T. Il permet d'accéder au serveur cible et de parcourir les documents stockés dans une base de données.

Robo 3T affiche tout document consulté dans un nouvel onglet, permettant une navigation rapide entre les différentes zones de la base de données. On peut ouvrir des documents et en insérer de nouveaux dans la base de données ainsi que supprimer ou modifier des documents existants.



*(Exemple de l’interface robo 3t)*

Maintenant, il va falloir créer la route dans le dossier.



Ici j’utilise la méthode GET.

**GET** est « safe », c’est-à-dire qu’une requête HTTP GET n’a pas de conséquences significatives autre que le chargement d’une ressource. GET est en outre « idempotent » ce qui signifie que plusieurs requêtes HTTP GET identiques lancées à la suite provoquent le même effet (retournent le même contenu Web) qu’une seule. Le navigateur peut donc mettre en cache la réponse à une requête GET. GET est idéale pour les accès en lecture. Et c’est en GET que le navigateur accède au serveur lorsqu’on saisit une URL dans la barre d’adresse. On lui indique son chemin. Le handler permet d’indiquer que l’on veut la méthode de la plateforme : getPlatformKpi­­­­.

J’ai aussi rajouté des options et des prérequis, pour définir quels utilisateurs peuvent accéder à ces informations (admin, viewers, chef de projets …) et le chargement de la plateforme. Sur ce screen, il n’y a que la route que j’ai créée mais toutes les routes du dossier plateforme y sont. Enfin, on exporte l’ensemble des routes.

# Les fonctions asynchrones

Une fonction asynchrone est une fonction qui s'exécute de façon asynchrone grâce à la boucle d'évènement en utilisant une promesse comme valeur de retour. Par défaut, le JavaScript est un langage synchrone, bloquant et qui ne s’exécute que sur un seul thread.

Cela signifie que :

* Les différentes opérations vont s’exécuter les unes à la suite des autres (elles sont synchrones) ;
* Chaque nouvelle opération doit attendre que la précédente ait terminé pour démarrer (l’opération précédente est « bloquante ») ;
* Le JavaScript ne peut exécuter qu’une instruction à la fois (il s’exécute sur un thread, c’est-à-dire un « fil » ou une « tache » ou un « processus » unique).

Cela peut rapidement poser problème dans un contexte Web : imaginons qu’une de nos fonctions ou qu’une boucle prenne beaucoup de temps à s’exécuter. Tant que cette fonction n’a pas terminé son travail, la suite du script ne peut pas s’exécuter (elle est bloquée) et le programme dans son ensemble parait complètement arrêté du point de vue de l’utilisateur.

Pour éviter de bloquer totalement le navigateur et le reste du script, on aimerait que ce genre d’opérations se déroule de manière asynchrone, c’est-à-dire en marge du reste du code et qu’ainsi le reste du code ne soit pas bloqué.

Cela est aujourd’hui possible puisque les machines disposent de plusieurs cœurs, ce qui leur permet d’exécuter plusieurs tâches de façon indépendante et en parallèle, que le JavaScript nous fournit des outils pour créer du code asynchrone.

L’intérêt est d’éviter un temps de latence trop long dans le chargement de la page. Le but est d’avoir une plateforme réactive.



Ici, je définis la fonction getPlatformKpi demandée dans la route. On redemande les prérequis et on déclare une constante qui attend le retour sur la plateforme du nom de celle-ci et que les utilisateurs aient un accès à la plateforme. Comme c’est une fonction asynchrone, on attend la réponse. Si on n’en a pas la fonction renvoie un message d’erreur avec le code erreur 500.

L’erreur 500 s’affiche dans le navigateur quand celui-ci n’arrive pas à accéder au serveur du site internet demandé, pour une raison qu’il ne parvient pas à identifier. On parle alors d’erreur interne du serveur ou plutôt « Internal Server Error » comme cela est signalé. Lorsque cela se produit, l’internaute n’a généralement d’autre choix que de quitter votre site et de revenir plus tard.

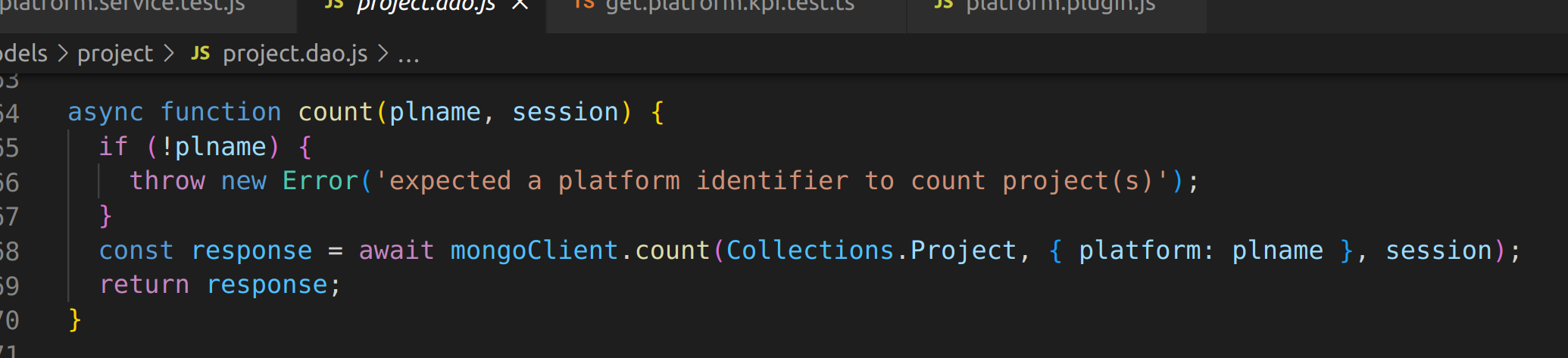
Ensuite, j’ai créé les fonctions asynchrones count dans les fichiers usersDAO et projectsDAO dans le fichier platform.service.js. Un objet d'accès aux données (en anglais data access object ou DAO) est un patron de conception (c'est-à-dire un modèle pour concevoir une solution) utilisé dans les architectures logicielles objet.

Ce modèle complète le modèle MVC (modèle-vue-contrôleur), qui préconise de séparer dans des classes les différentes problématiques :

* Des vues : charte graphique, ergonomie
* Du modèle : cœur du métier
* Des contrôleurs : tout le reste ; l'enchaînement des vues, les autorisations d'accès…

L'utilisation de DAO permet de s'abstraire de la façon dont les données sont stockées au niveau des objets métiers. Ainsi, le changement du mode de stockage ne remet pas en cause le reste de l'application. En effet, seules ces classes dites "techniques" seront à modifier (et donc à re-tester). Cette souplesse implique cependant un coût additionnel dû à une plus grande complexité de mise en œuvre.

J’ai donc créé 2 fonctions asynchrones count qui permettent de faire ma requête mongo dans les 2 dossiers qui m’intéressaient : User et Project pour appeler mes requêtes mongoDB.



Ici, une des deux fonctions asynchrones count (la deuxième est presque pareil à l’exception du message d’erreur et de la collection appelée). Si le plname est manquant (le nom de la plateforme), alors on envoie le message d’erreur qui dit qu’on attend les identifiants de la plateforme pour compter les projets (même message pour les users « to count user(s) »). Si la plateforme est présente alors on attend la réponse de mongo avec le compte de project (ou d’user pour le userDAO).

# la méthode CRUD

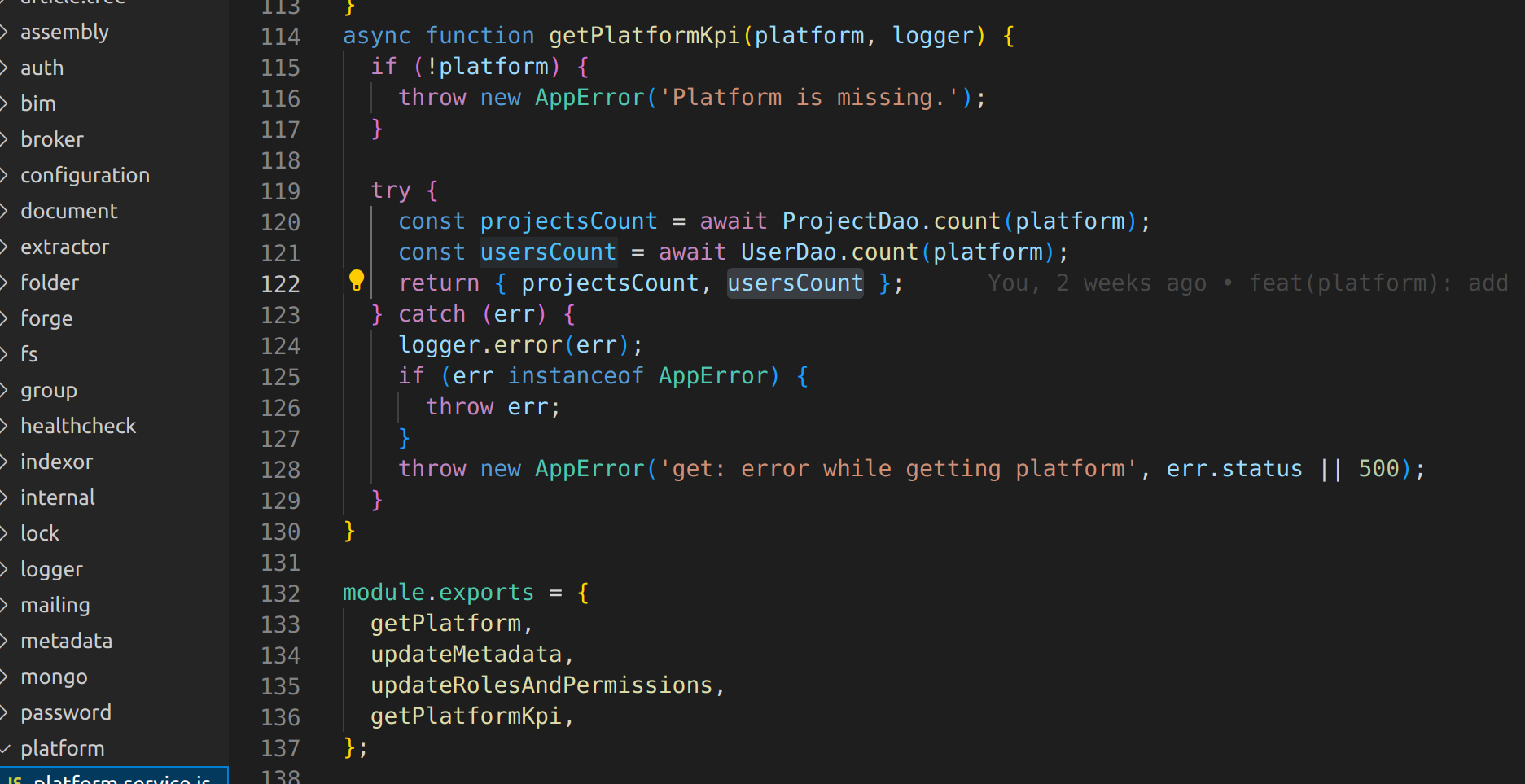
Le terme CRUD est étroitement lié avec la gestion des données numériques. Plus précisément, CRUD est un acronyme des noms des quatre opérations de base de la gestion de la persistance des données et applications :

* Create (créer)
* Read ou Retrieve (lire)
* Update (mettre à jour)
* Delete ou Destroy (supprimer)

Plus simplement, le terme CRUD résume les fonctions qu’un utilisateur a besoin d’utiliser pour créer et gérer des données. Divers processus de gestion des données sont basés sur CRUD, cependant les opérations sont spécifiquement adaptées aux besoins des systèmes et des utilisateurs, que ce soit dans la gestion des bases de données ou pour l’utilisation des applications. Ainsi, les opérations sont des outils d’accès classiques et indispensables avec lesquels les experts, peuvent par exemple vérifier les problèmes de base de données. Tandis que pour un utilisateur, CRUD signifie la création d’un compte (create), l’utilisation à tout moment (read), la mise à jour (update) ou encore la suppression (delete).

(Voir annexe 2 les différentes opérations de CRUD sur le projet)

# Appel de la DAO



Ici, on rappelle les deux fonctions count() pour ne former plus qu’une fonction à exporter. Avec le if, on renvoie une erreur si la plateforme est manquante. Ensuite on fait appel à platformDao, UserDao et la fonction qui nous intéresse (count()) avec en paramètre platform en cas d’erreur de platform que nous avons déclarée précédemment.

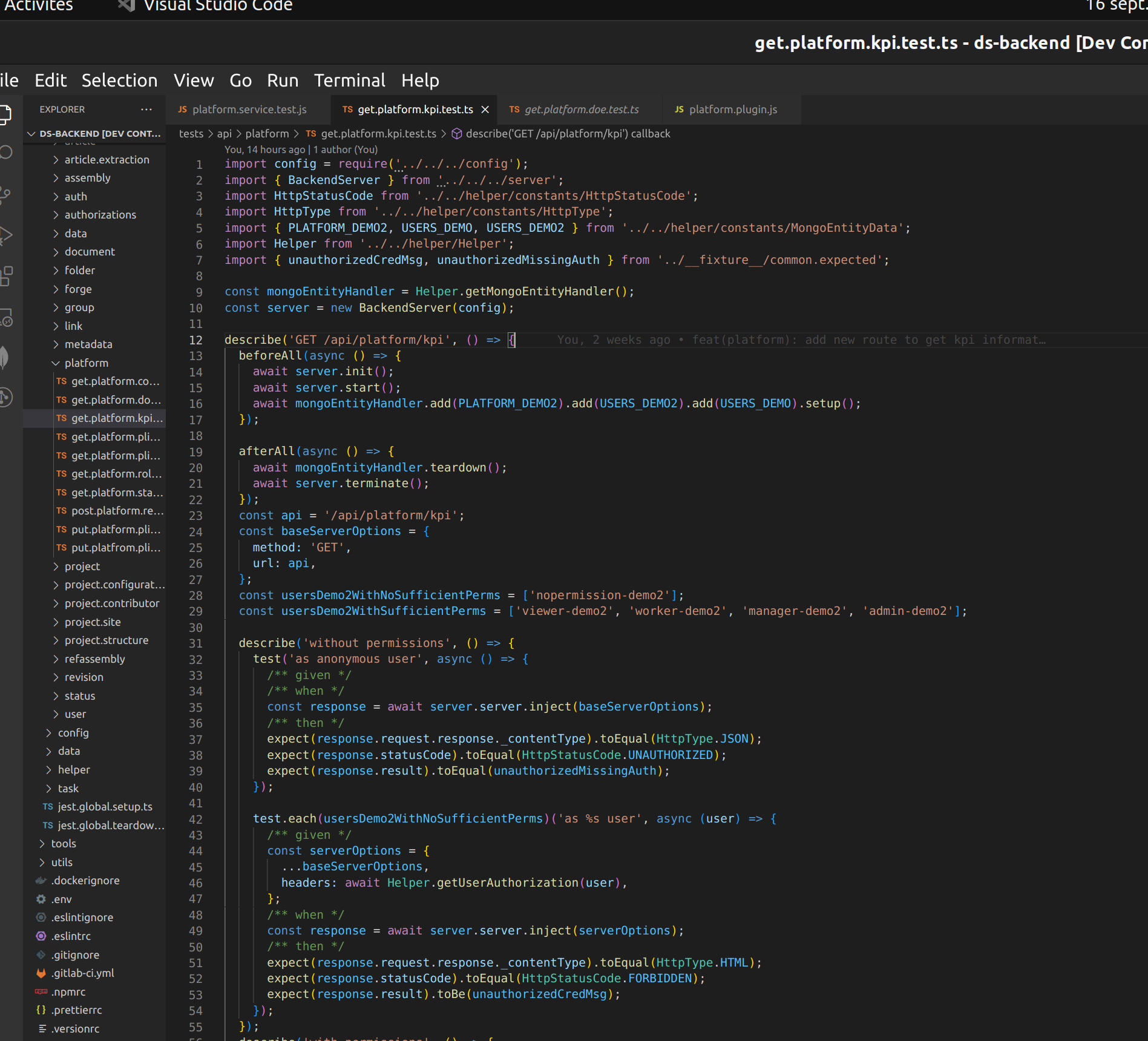
Enfin, on renvoie une erreur s’il y a un problème de connexion avec un code erreur 500.

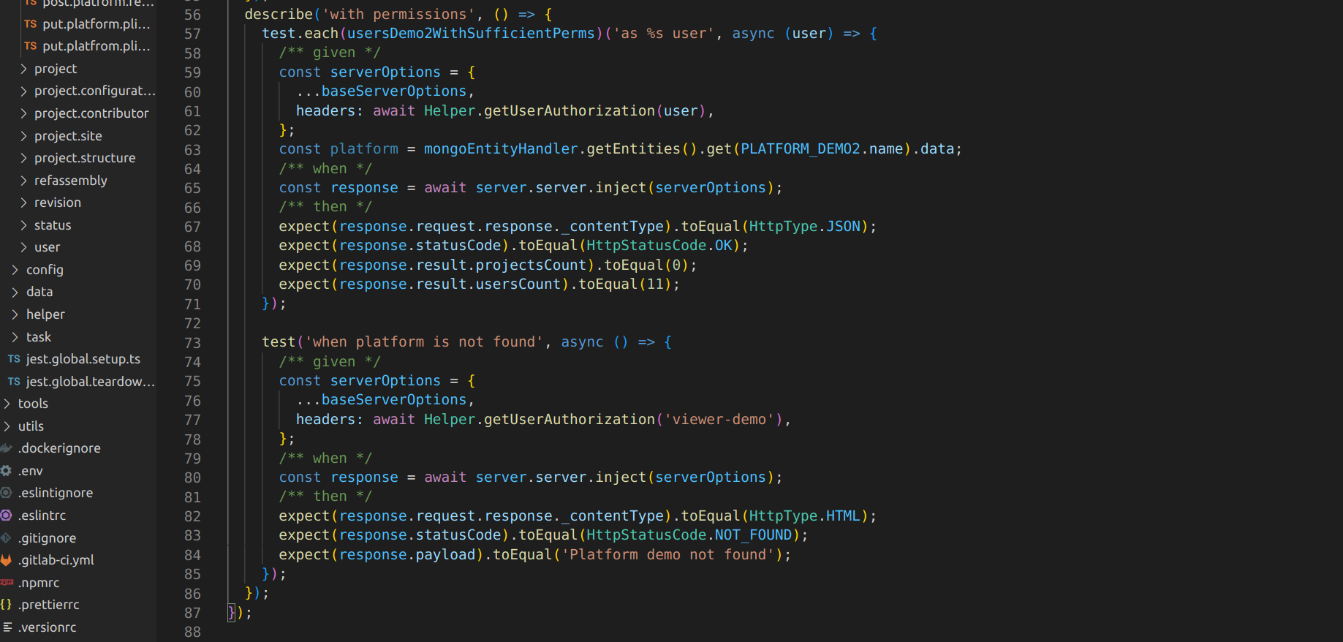
Le chemin est terminé. Il n’y a plus qu’à lier le front et le back.

# Les tests unitaires et d’intégration du back

Les tests d’intégration représentent environ 20 % du plan de test et sont plus longs et difficiles à écrire que les tests unitaires, car ils reposent sur des composants tiers.

Certains composants sont réels, d’autres sont simulés. Ces tests sont beaucoup plus fidèles que les tests unitaires. Ils permettent également de vérifier l’aspect fonctionnel, les performances et la fiabilité du logiciel.





Ceci est le test d’intégration que j’ai créé pour la route, écrit en Typescript.

Ici, on décrit la route (pour ma part, la méthode GET /api/platform/kpi). Avec le beforeAll, je lui demande de démarrer le serveur et d’ajouter la plateform demo2 et les utilisateurs de la démo et de la demo2 au tout début du test (nb : j’ai rajouté la platform démo après le screen). Avec le afterAll, je lui demande de fermer le serveur après la fin du test (c’est une fonction asynchrone donc elle s’exécutera à la fin du test).

Je déclare ensuite les constantes comme le path, la baseServerOptions et les autorisations ou non des utilisateurs. Le but étant de tester le cas de figure ou l’utilisateur a le droit ou non d’avoir accès aux données de cette route.

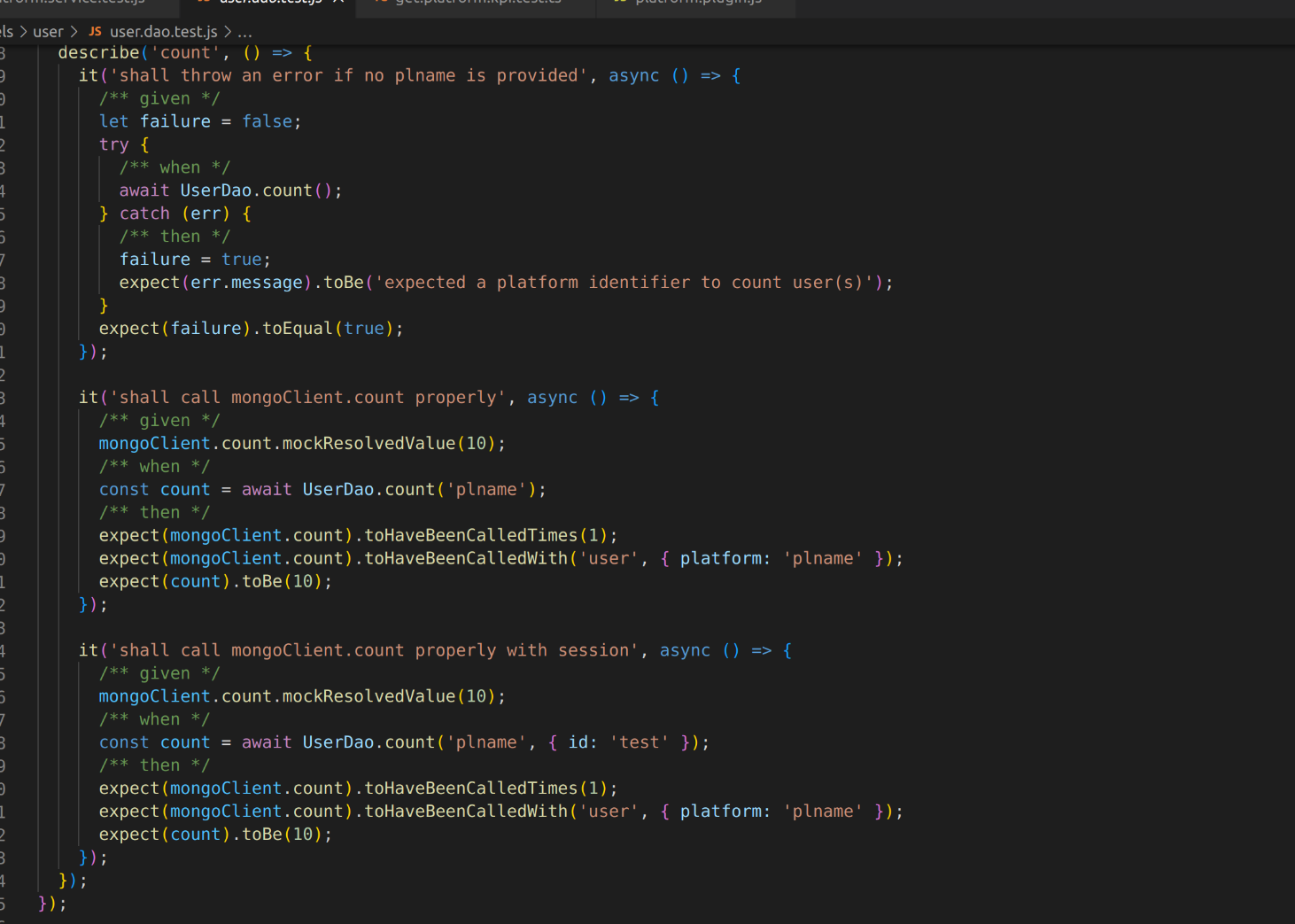
Dans le describe ‘without permissions’ d’un utilisateur inconnu, je demande de lancer le serveur avec la méthode et l’url de la route. On s’attend à ce que l’utilisateur inconnu ait un code statut unauthorised et lui envoie un message qu’il n’a pas accès à ses données.

Idem pour l’utilisateur qui n’aurait pas les autorisations demandées, on s’attend à ce que cet utilisateur ait un code statut forbiden, et lui envoie qu’il n’a pas accès à ses données.

Dans la 2ème partie, on décrit les utilisateurs avec les permissions suffisantes. On ajoute avec le given des options du serveur, c’est-à-dire qu’on appelle la fonction getUserAuthorisation(user) pour vérifier l’accès. Quand on injecte cette nouvelle constante au serveur, on s’attend à ce que le statut code soit ok et que la route renvoie le résultat 0 pour projectsCount et 11 pour usersCount.

Pour finir, on teste quand la plateforme n’est pas trouvée. On s’attend à un code statut de not found et que le message d’erreur soit ‘platform demo not found’.

Il est temps de vérifier que notre test est bon. Dans le terminal de vsCode, je lance la commande « yarn integration-test » et de vérifier que ce test fonctionne correctement (c’est le cas).



Ci- dessus, je vous montre le test unitaire de la fonction count() du userDao.

Le 1er cas devrait faire remonter une erreur si aucun nom de plateforme est fourni. On nous dit qu’étant donné que failure est égale a false, quand on fait appel à la fonction count() de userDao, mais que quand il y a une erreur failure est égale à true alors on s’attend à un message d’erreur et que failure est égale à true.

Le 2ème cas démontre le bon fonctionnement de la fonction count().

Dans le given on lui donne une donnée fictive (mock) de 10. Dans le when, on fait appel à la fonction. Dans le then on attend que la fonction soit appelée une seule fois, avec le nom de la plateforme et qu’elle renvoie la valeur fictive.

Le dernier cas démontre le bon fonctionnement de la fonction avec une session. On lui rajoute juste un ID pour vérifier la condition de la session.

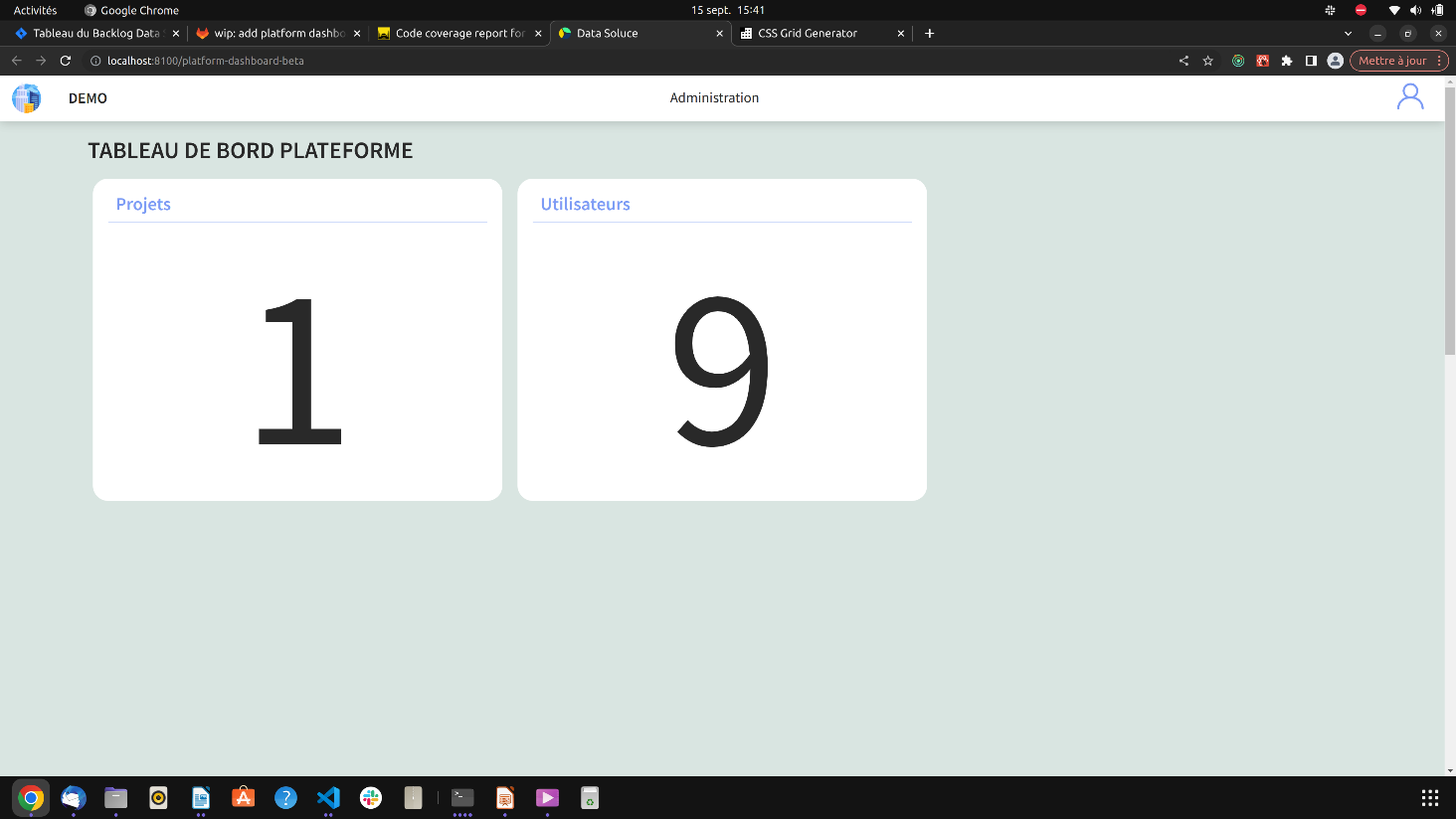
# Lier le back au front

Après quelques modifications de noms de paramètres et la modification de la route, le back a été lié au front



# Résultat final

l

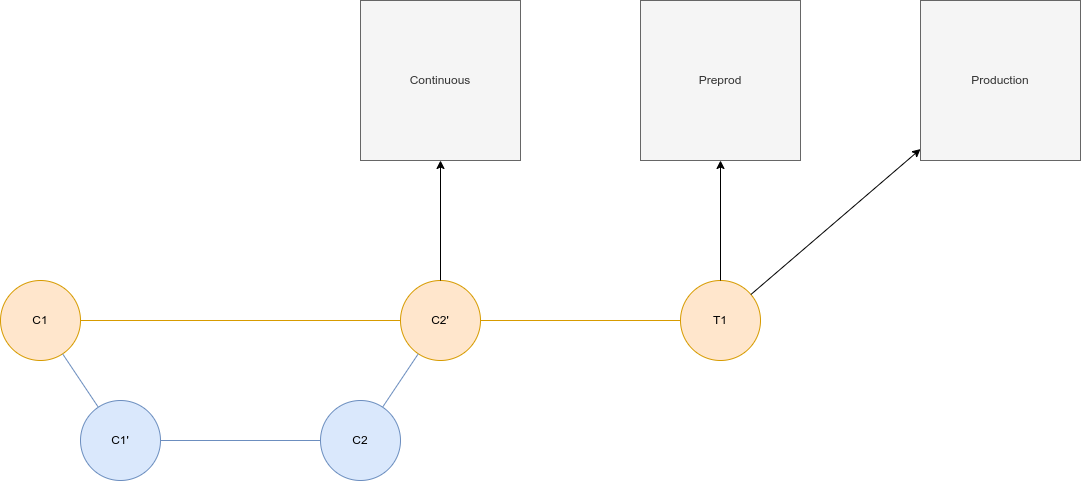


Mon travail est en pré-production dans l’attente d’une autre user story pour remplir les informations du tableau de bord.

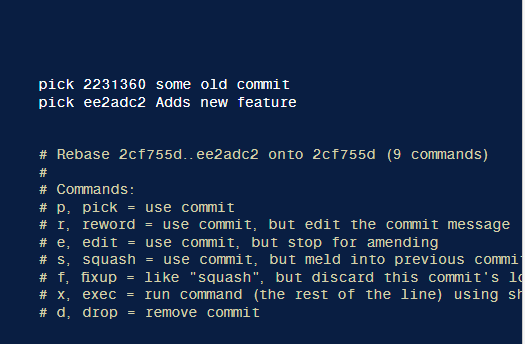
# Git

Que ce soit pour le dossier back-end ou front-end, j’ai travaillé sur ma branche git. J’ai pu faire de nombreux commit « wip » (work in progress) avec la commande git commit –m « wip » pour sauvegarder mon travail tout au long de ma progression. Chez Data soluce, chaque branche est liée à une user story. Il y a un formalisme dans l’écriture du commit et il ne faut en envoyer qu’un seul. Pour cela, il faut faire un git rebase.

Un historique git est un peu comme un arbre, avec son tronc (la branche master), ses branches et ses sous-branches. La commande git-rebase est comme une tronçonneuse : elle permet de couper une branche pour la regreffer à un autre endroit sur l'arbre.

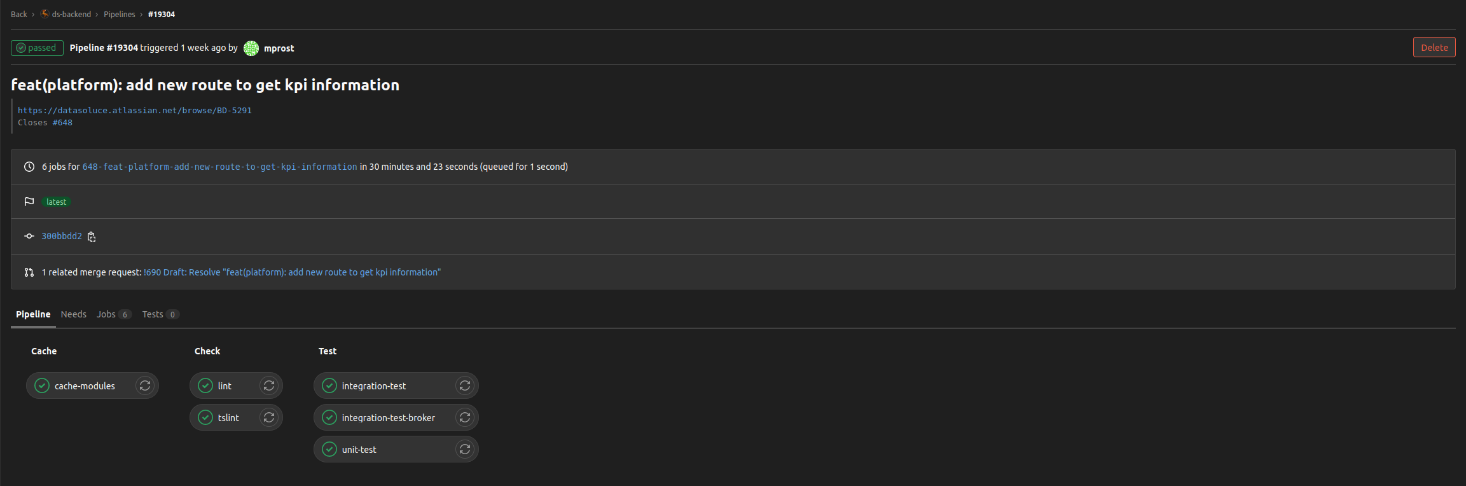


J’ai effectué la commande git rebase -i HEAD~4. Le terminal de commande affiche les commit au nombre de 4 du plus récent au plus ancien. On garde le dernier on fixup les 3 derniers. On remplace pick par f. contrôle +o pour enregistrer, entrée, contrôle+x pour sortir. Il ne reste plus qu’un commit. Après ça, j’ai fait git push -f pour forcer le push. Mon travail est resté sur la branche en attendant le commit de mon collègue.

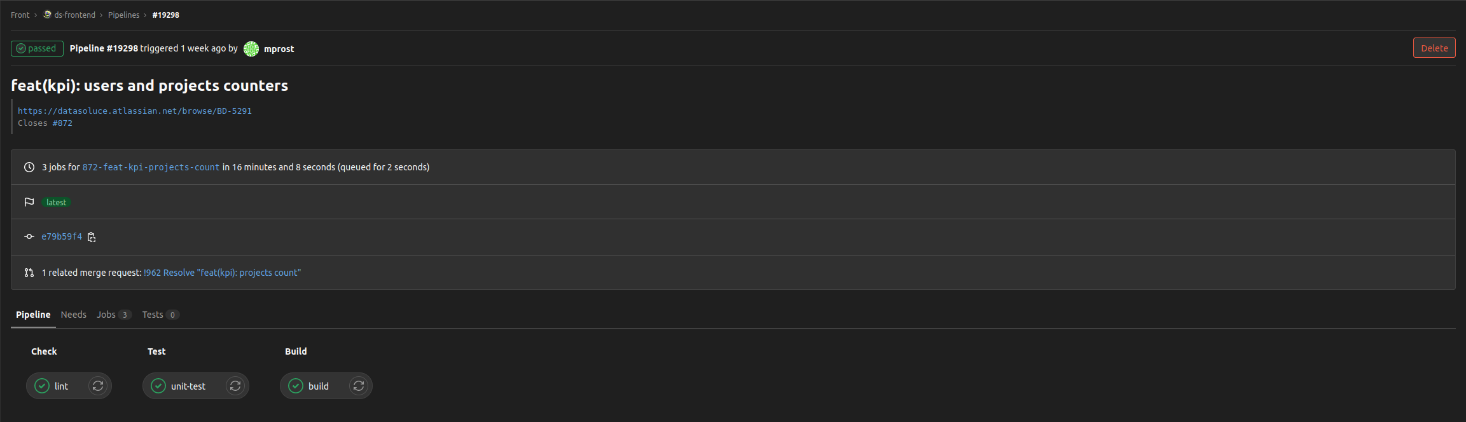


# Pipeline Gitlab

Après avoir push notre travail, il faut passer les pipelines.



(Pipeline coté back)



(Pipeline coté front)

Chez Data soluce il y a 3 pipelines à passer. Le code sans faute, le eslint et tslint et les tests unitaires et d’intégration. ESLint est un plugin permettant d’uniformiser la mise en forme du code entre les différents développeurs contribuant au projet en javascript. Idem pour tslint mais au niveau de typescript.

Dès que je faisais un commit, celui-ci passait le test des pipelines. Si un des pipelines étaient failed je ne pouvais pas merger sur la branche master. Souvent mes tests unitaires en local étaient bons mais ne passaient pas au pipeline (soit il manquait un bout de code non couvert soit un problème de mock, ou de snapshots). Mais grâce à la console d’erreur de gitlab, je pouvais trouver facilement mon erreur.

Si le pipeline Eslint, Tslint étaient failed, il fallait lancer la commande yarn prettier dans le terminal pour harmoniser le code.

Quand tous les pipelines étaient verts, le push pouvait être fusionné sur la branche master.

# Situation de travail

Lors du développement du projet, j’ai dû me documenter afin d’apprendre de nouveaux langages ou pour approfondir ceux dont j’avais déjà connaissance.

Dans le monde de l’informatique en particulier dans le développement web, la quasi-totalité des documentations sont en anglais. Nous pouvons en trouver en français mais l’anglais est de rigueur surtout pour les technologies et frameworks populaires. C’est pourquoi il faut un niveau d’anglais minimum, je dirais que j’ai un niveau d’anglais technique c’est-à-dire que je comprends les documentations et tout ce qui a trait à l’informatique. Cependant, je ne pourrais pas tenir une conversation complète sans la moindre difficulté avec quelqu’un qui parle anglais.

Pour ma part, la recherche s’est portée sur les routes Hapi.

Mon tuteur m’avait envoyé le lien https://hapi.dev/ pour bien comprendre le fonctionnement des routes sur l’application.

# Extrait du site anglophone

J’ai tuilisé le site : <https://hapi.dev/tutorials/expresstohapi/?lang=en_US> pour bien comprendre comment faire une route.



# Traduction

Les routes in Hapi sont appelées dans un ordre spécifique, donc vous n’aurez jamais de problème avec deux routes qui rentrent en conflit avec une autre. Les routes sont appelées du plus spécifique au moins spécifique. Par exemple une route avec un chemin ‘/home’ sera appelée avant ‘/{any\*}’.

Regardons comment installer une route basique avec Hapi :

Express :

Hapi :

Pour créer une route, Express a la structure « app.METHOD(PATH, HANDLER) » et Hapi a la structure « server.route({METHOD,PATH,HANDLER}) ». La METHOD, le PATH et le HANDLER passe dans le serveur Hapi comme un objet. Comme vous pouvez le voir, pour retourner une chaine de caractères en Express, vous appelez ‘res.send()’, tandis qu’avec Hapi, vous retournez simplement une chaine de caractère.

Methods :

Hapi peut utiliser toutes les méthodes de route qu’Express, excepté HEAD. Hapi a aussi la possibilité d’utiliser de multiples méthodes sur un seul objet route. Par exemple :

Pour utiliser toutes les méthodes disponibles comme avec Express ’app.all()’, utilisez « method ‘\*’ ».

Path :

Comme avec Express, l’option path avec Hapi doit être une chaine de caractères, qui peut aussi contenir des paramètres. Les paramètres dans Express sont précédés par « : » comme « ‘/users/ :usersId’ ». Dans Hapi, vous mettrez le paramètre dans les accolades, comme « path : ‘/usesrs/{usersId}’ ».

# Sécurité

Pour la sécurité, Data Soluce a mis en place les mécanismes et procédures suivantes :

**- Communication chiffrée entre les composants (https)**

L’abréviation HTTPS signifie « Hypertext Transfer Protocol Secure », qui se traduit par « Protocole hypertexte de transmission sécurisée ». Le protocole de transmission est en quelque sorte le langage par lequel le client - en règle générale le navigateur - et le serveur Web se comprennent. Le HTTPS est la version du protocole de transmission qui opère en transmission chiffrée.

Le HTTPS remplit deux fonctions :

La communication entre le client Web et le serveur Web est chiffrée. Afin d’empêcher qu’un tiers non autorisé « écoute » la communication en prenant par exemple connaissance du trafic en réseau WLAN.

Le serveur Web est authentifié par le fait qu’en tout début de communication, un certificat est envoyé au client Web pour attester de la crédibilité du domaine. Cette mesure contribue à combattre les tromperies résultant de faux sites Web.

**- Configuration du Content Security Policy niveau serveur**

Une Content Security Policy (CSP) ou stratégie de sécurité du contenu permet d'améliorer la sécurité des sites web en permettant de détecter et réduire certains types d'attaques, dont les attaques XSS (en-US) (Cross Site Scripting) et les injections de contenu. Ces attaques peuvent être utilisées dans divers buts, comme le vol de données, le défacement de site ou la diffusion de malware.

L'un des objectifs de CSP est la réduction et le rapport d'attaques XSS (injections de contenu). Les attaques XSS exploitent la confiance que les navigateurs ont dans le contenu reçu des serveurs. Des scripts malveillants peuvent être exécutés par le navigateur d'une victime parce que le navigateur fait confiance au serveur qui lui envoie des données même quand le contenu ne vient pas de là où il semble venir.

**- Toute donnée provenant d'un formulaire est validée pour garantir la cohérence des données**

Elles sont validées par leur type à l’entrée et à la sortie du serveur.

**- Tous les composants et les systèmes de la plateforme sont monitorés avec la suite ELK et des alertes sont mises en place sur des logs clés**

Toutes ces données sont analysées par l’administrateur système. La suite ELK est un acronyme utilisé pour décrire une suite qui comprend trois projets populaires : Elasticsearch, Logstash et Kibana. Souvent appelée Elasticsearch, la suite ELK vous permet d'agréger les journaux de tous vos systèmes et applications, de les analyser, et de créer des visualisations pour la surveillance des applications et de l'infrastructure, un dépannage plus rapide, l'analytique de la sécurité, etc.

**- L'authentification et les autorisations suivent le schéma OAuth implicit. Data Soluce va évoluer vers un schéma plus recommandé dans les mois qui viennent**

OAuth, abréviation de « Open Authorization », est un protocole standard ouvert permettant une autorisation API sécurisée. Afin d’éviter tout risque d’interception de ces données par des tiers et leur utilisation abusive, une autorisation des transferts de données effectués entre ces API est par conséquent nécessaire.

# Conclusion

Ce stage a été une révélation pour moi, autant géographiquement qu’au niveau de ma pratique professionnelle. J’ai vu différents métiers dans la sphère du développement (product owner, UX/UI designer, administrateur réseaux …). J’ai changé de système d’exploitation (de windows à linux). J’ai travaillé sur des technologies que je ne connaissais pas ou très peu. Je pense que le maître mot de ce stage a été adaptabilité.

Au début de ce projet, j’avais l’appréhension de ne pas être à la hauteur car j’avais beaucoup de choses à apprendre sur la réalité du métier et je débutais sur la stack de l’entreprise. J’ai beaucoup aimé tout ce que j’apprenais grâce à l’accueil de l’équipe. J’ai suivi des tutoriels et des cours sur internet, tous intéressants, qui m’ont permis aussi d'améliorer ma compréhension de l’anglais à l’oral. Puis au fur et à mesure, j’ai vu une progression dans la compréhension du code durant ces 8 semaines.

J’ai souvent été cloué devant des écrans à chercher, lire, apprendre, encore et encore, je pense que c’est aussi là que se situe mon travail. Certes je produis des lignes et des lignes de codes. Malgré ces déboires, j’ai pu apprendre à utiliser toutes mes compétences et connaissances pour résoudre mes problèmes et gagner en autonomie. Ce sont juste des expériences formatrices. J’ai suivi ce stage avec envie et passion... De plus, savoir que mon travail va être envoyé en production est une grande satisfaction pour moi.

Le métier de développeur est enrichissant, et les projets sur lesquels on travaille sont encore plus intéressants. Cela nous donne envie d’apprendre et la qualité de notre travail s’en ressent.

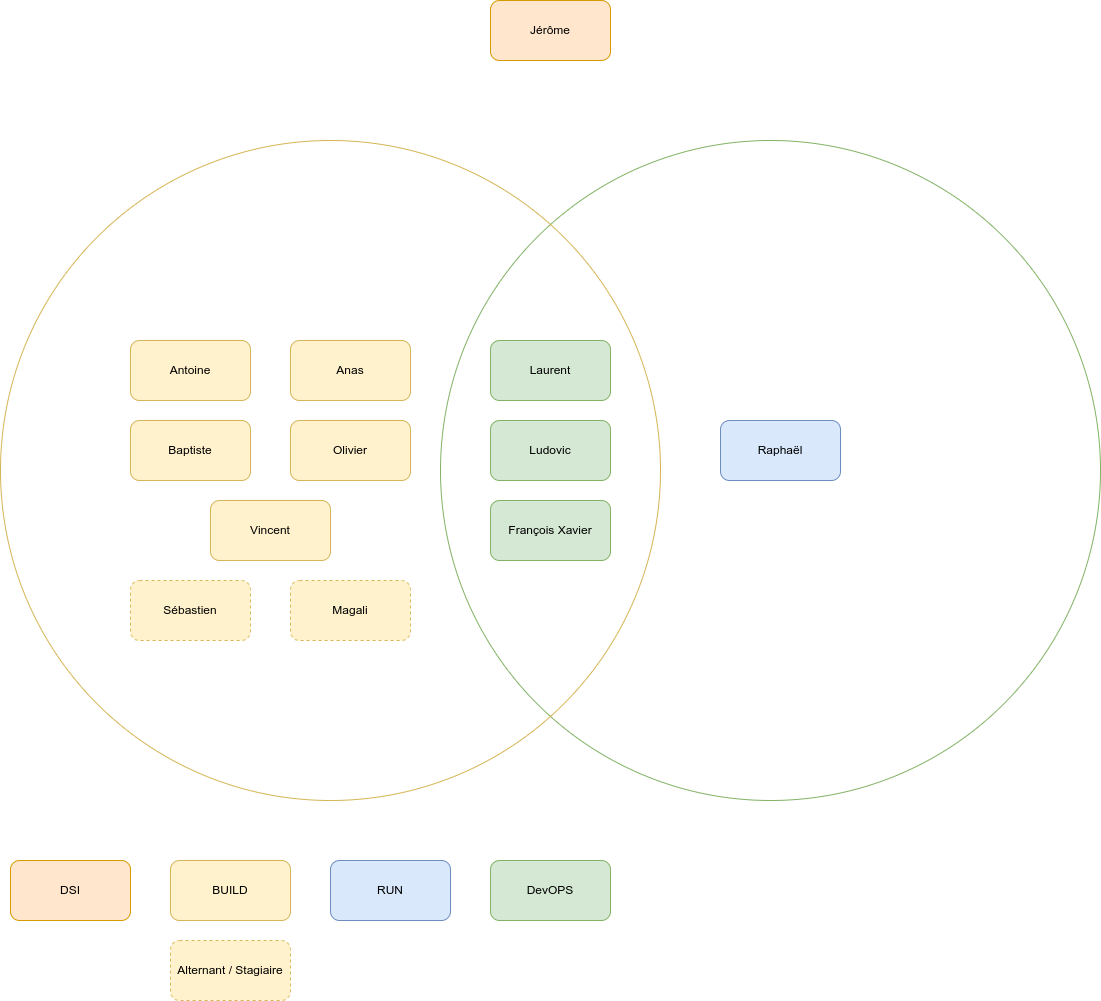
J’ai appris non seulement à coder proprement, mais j’ai acquis une méthodologie de travail. Ce stage m’a conforté dans l’idée que ma reconversion professionnelle était la bonne, et je suis décidée à continuer dans cette voie en me dirigeant vers la formation de conceptrice, développeuse d’applications en alternance.

# Remerciements

Je remercie :

* L’Afpa pour ma formation
* Mon formateur Guillaume Delacroix
* L’équipe de Data Soluce et plus particulièrement mon tuteur pendant ce stage Francois-Xavier Lafont pour leur pédagogie et leur professionnalisme

# Annexe 1 : organisation du service IT



# Annexe 2 : les différentes opérations de CRUD sur le projet

Une image contenant texte

Description générée automatiquement

Cette requête se traduirait en SQL par : INSERT INTO table VALUES ('valeur 1', 'valeur 2', ...).

Une image contenant texte

Description générée automatiquement

Cette requête se traduirait en SQL par : UPDATE table

SET nom\_colonne\_1 = 'nouvelle valeur'

WHERE condition ;

Une image contenant texte

Description générée automatiquement

Cette requête se traduirait en SQL par : DELETE FROM `Project`

WHERE ‘projectId’ = 1 ;