25/03/2022

RAPPORT DE STAGE

Participation à l'intégration de tests end-to-end dans une application en phase de développement

BÉCART Rémy
BTS SIO OPTION SLAM
Promotion 2022

REMERCIEMENTS

Je tiens dans un premier temps à remercier Mr GUY Philippe, fondateur de Iole Solutions, de m'avoir accueilli au sein de son entreprise, et Mr HÉLIOU Aurélien, directeur générale de Iole Solutions et également mon tuteur de stage, de m'avoir accueilli chaleureusement et de m'avoir fait confiance durant ces deux mois de stage, et je le remercie également pour sa disponibilité et sa bienveillance.

Je tenais dans un second temps à remercier toute l'équipe de Iole Solutions, pour leur sympathie ainsi que pour le temps qu'ils ont consacré à me guider et à répondre à mes questions, et je les remercie également pour la bonne ambiance qu'ils ont fait régner durant tout le stage, car l'ambiance de travail est aussi vitale que l'oxygène dans l'air.

Enfin je remercie Mr Ulrich MATHON, formateur référent au Greta de Vannes Bretagne Sud, pour ses conseils et son accompagnement durant le stage.

SOMMAIRE

L'ENTREPRISE IOLE SOLUTIONS

Présentation de l'entreprise

Iole Solutions est une start-up Vannetaise créé en 2016 par monsieur GUY Philippe, innovante et en cours de développement. Iole est un terme qui désigne une embarcation bretonne. L'objectif de Iole solutions est la création et le développement d'une ERP, à destination des entreprises, regroupant tous les corps de métiers. L'idée originale vient du fondateur de Iole solutions, partant d'un constat simple :

« Il n'existe pas d'outil de gestion d'entreprise suffisamment performant sur le marché capable de répondre aux réels besoins des entreprises française et étrangère. »

Qu'est-ce qu'un ERP ?

ERP (Enterprise Ressource Planning) est un progiciel qui permet de gérer l'ensemble des processus opérationnels d'une entreprise.

- **2017** • Développement des modules de base du logiciel et du front par des stagiaires
- **2018** • Emménagement dans les locaux du PIBS et création de la start-up "Iole Solutions"
- **2019** • Agrandissement de l'équipe technique de Iole, accueil d'une responsable commercial free-lance et d'une responsable administrative et comptable
- **2020** • Consolidation de l'équipe technique de Iole par l'intégration de nouveaux développeurs
- **2021** • Délégation des pouvoirs décisionnels, nomination d'un Directeur Générale

Environnement de travail

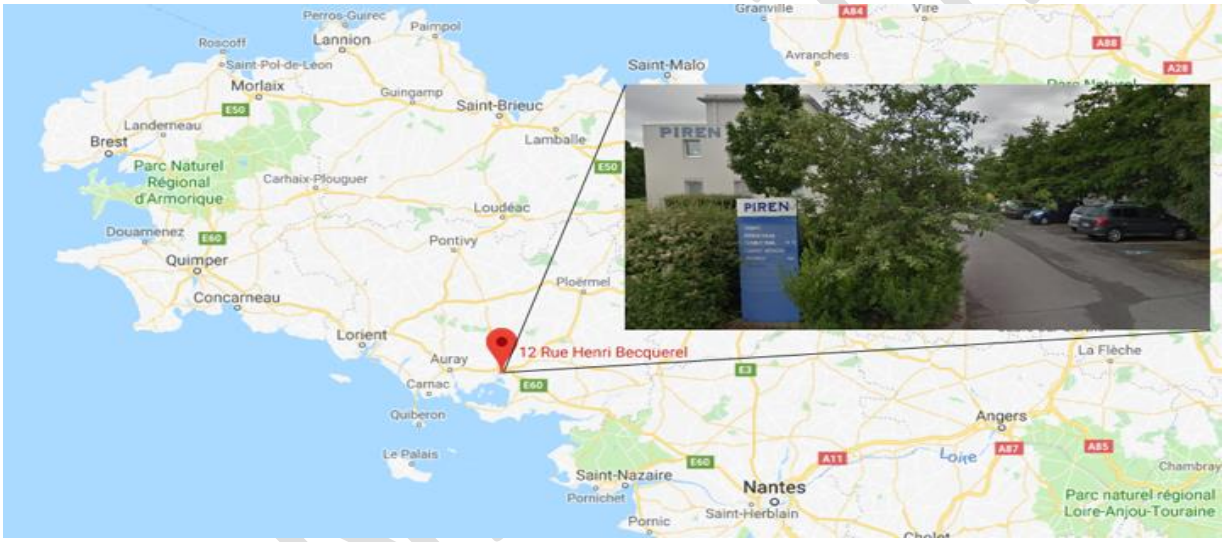
Nous travaillons dans un open space, constitué d'un ilot de 2 bureaux contenant 6 postes, ainsi que 2 autres ilots de 2 bureaux individuel contenant 2 postes chacun, pour un total de 10 postes. Iole dispose également de deux locaux indépendant de la société derrière l'open space, contenant 2 postes individuel, et dispose également d'une salle de réunion situé en face de ces locaux indépendant.

En raison de la crise sanitaire que nous traversons, l'entreprise a opté pour le télétravail a raison de 3 jours par semaine, pour lutter contre la propagation du virus Covid-19. Afin de respecter la distanciation entre chaque développeur en présentiel, l'entreprise a donc divisé l'équipe développement en deux groupes : un groupe présent le lundi et le jeudi, et l'autre groupe le mardi et le vendredi, et le mercredi télétravail pour tous.

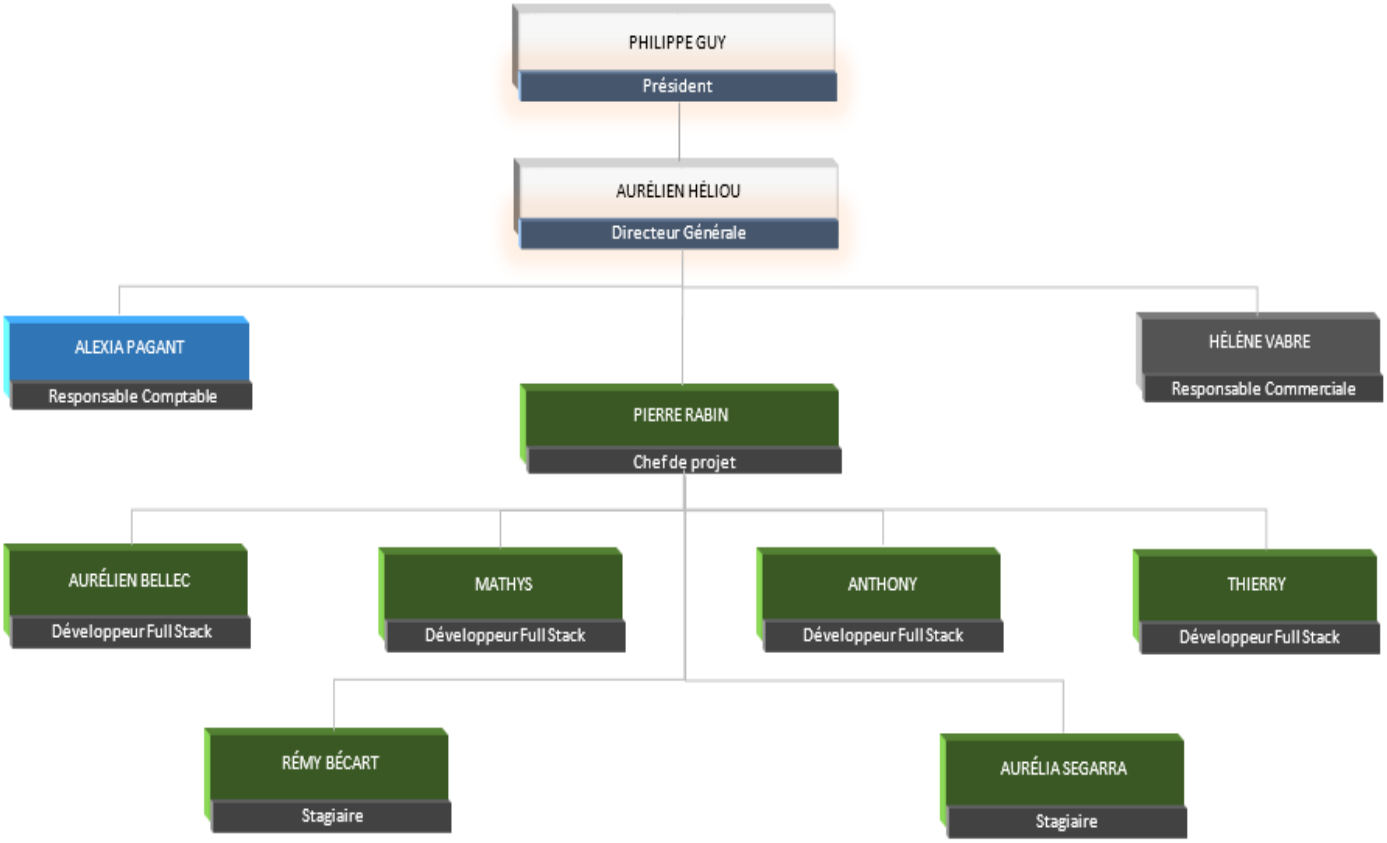
Lorsque nous sommes en télétravail, nous utilisons SoftEther VPN Client Manager, un VPN nous permettant de nous connecter en toute sécurité sur un bureau distant.

Situation géographique et juridique

Iole Solutions est une société par actions simplifiée à associé unique (SASU), créé et géré par Mr Guy Philippe. Cette petite start-up est implantée depuis juin 2018 au PIBS de Vannes (56) dans le bâtiment PIREN. Le PIBS de Vannes regroupe beaucoup de start-up innovante sous la forme d'un petit village design et connecté. C'est un lieu stratégique et accélérateur de Start-up, où Iole Solutions y a donc bien trouvé sa place.



Organigramme de l'entreprise



MISSION DE STAGE

PRÉSENTATION

Descriptif

Ma mission principale consiste à mettre en place des tests fonctionnels automatisés pour vérifier le bon fonctionnement de l'API de iole à l'aide de Cypress.

Définition

Cypress est un framework Javascript de tests end-to-end. C'est un outil open source qui consiste à tester l'ensemble de l'application en cours de développement du début à la fin pour s'assurer que le flux d'application se comporte comme prévu. Il définit les dépendances système du produit et garantit que toutes les pièces intégrées fonctionnent ensemble comme prévu. L'objectif principal des tests de bout en bout (E2E) est de tester l'expérience de l'utilisateur final en simulant le scénario de l'utilisateur réel et en validant le système testé et ses composants pour l'intégration et l'intégrité des données. Ces tests ignorent généralement la structure interne de l'ensemble de l'application et prennent le contrôle du navigateur comme un utilisateur allant sur votre application.

Objectif

Le chef de projet me transmet un fichier texte contenant à l'intérieur le descriptif des tests à réaliser sur l'API conçu par l'équipe de développeurs. L'objectif de ma mission est de m'assurer en suivant à la lettre les consignes de ce fichier que l'API fonctionne correctement en effectuant les tests, et il faudra ensuite rédiger dans un ticket sur gitea mon travail effectuer pour le faire vérifier par un développeur compétent et valider par le chef de projet pour l'incorporer sur la branche principale.

Technologies Utilisées

- **Visual Studio Code :** Éditeur de code extensible développé par Microsoft pour Windows, Linux et MacOs

- **PostgreSQL :** Système de gestion de base de données relationnelle et objet (SGBDRO)

- **Gitea :** Interface web permettant de visualiser les branches, les commits, les tags, etc… de ses projets git. Il a une utilisation proche de celle de GitHub.

- **Element :** Logicielle libre de messagerie instantanée

- **Jitsi :** Application de messagerie instantanée permettant d'effectuer des visioconférences

- **Cypress :** Outil d'automatisation de test end-to-end

- **Dokuwiki :** Moteur de wiki libre simple d'utilisation dont le but principal est de créer des documentations de toutes sortes

- **Suite Office :** Suite bureautique de propriété Microsoft regroupant une suite de logiciels comme Word, Excel, Powerpoint, OneNote, Outlook, Access et/ou Publisher

- **Soft Ether VPN Client Manager :** Logiciel client VPN et serveur VPN open source permettant une connexion sécurisée sur un bureau distant

Travail effectué

En me basant sur le fichier texte transmis par mon responsable de stage, j'ai dû dans un premier temps dans l'arborescence de fichiers VSCode créer les fichiers tests Cypress dans le dossier intégration, afin d'y insérer à l'intérieur des lignes de code permettant de tester l'API en ciblant les champs concernés pour ensuite interagir avec.

Dans un second temps, il a fallu me familiariser avec la syntaxe du langage, pour savoir comment aller chercher un champ, et comment interagir avec. Pour cela, je me suis aidé de la documentation interne de Iole expliquant bien la syntaxe des interactions les plus courantes.

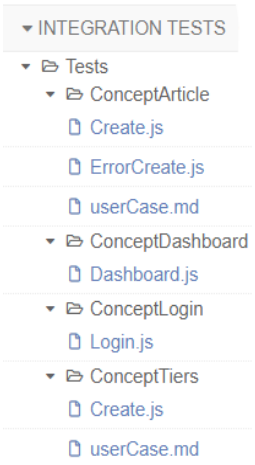
Pour pouvoir effectuer mes tests, il faut cibler les id et/ou les class des champs que l'on veut renseigner, et leur indiquer derrière l'interaction souhaité. Pour trouver les id et/ou les class que je cherche à cibler, il a fallu me connecter avec mes identifiants sur l'API pour ouvrir l'inspecteur et venir cibler à l'aide du sélecteur d'élément sur la page à inspecter l'id et/ou la « class » du champ en question, et lui indiquer ce qu'il faut taper dedans. Ensuite je copie/colle l'id et je le colle dans VSCode.

Lorsque j'ai terminé mon code, j'enregistre dans un 1er temps mon travail sur VSCode, puis j'ouvre le terminal. Dans celui-ci, je vais me diriger sur la partie front dans l'arborescence de fichiers sur laquelle j'ai écrit mon code, puis j'ouvre cypress pour tester mon code. Pour cela, il faut exécuter dans le terminal les commandes suivantes :

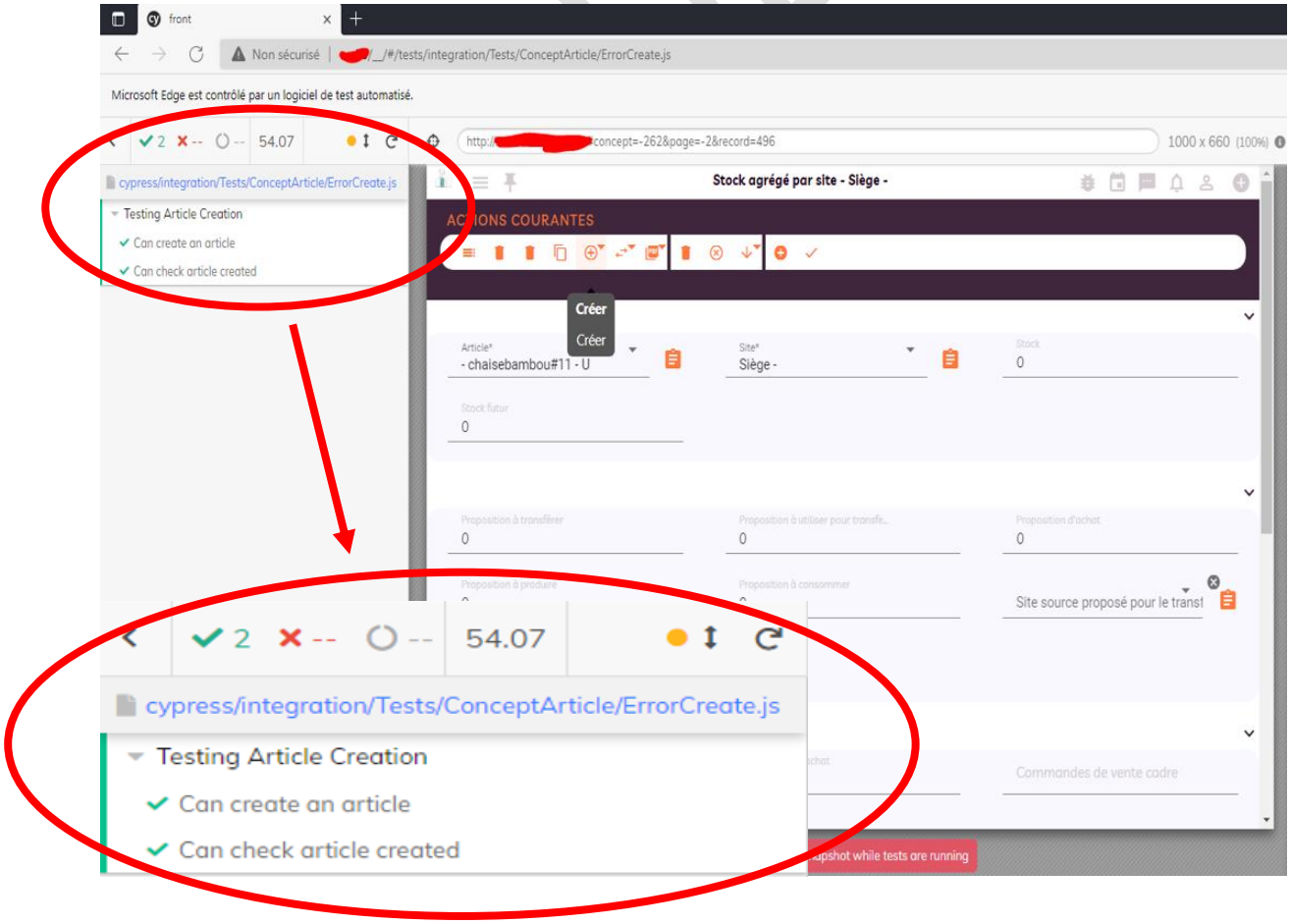
```
rbecart@srv27 MINGW64 /h/RBecart (support/cypressRemy)
$ cd front

rbecart@srv27 MINGW64 /h/RBecart/front (support/cypressRemy)
$ npm run cypress:open
```

Cela va faire apparaitre une fenêtre cypress qui fera apparaitre nos fichiers de tests, et il faudra cliquer sur l'un d'entre eux pour exécuter le test de notre code.



Si tout se passe bien, il nous mettra une encoche verte en haut de la fenêtre. Dans le cas contraire, il fera apparaitre une encoche rouge, et nous indiquera le n° de ligne concerné dans notre code contenant l'erreur, avec un lien pour aller consulter sur la documentation officielle de cypress une proposition de correction possible.



Annexes

```
# Création d'un article :
*DEBUT :*
ATTENTION 2 tests à faire donc 2 fiches à créer

## 1er TEST :
### Connexion à iole
    - identifiant : [redacted]
    - Mot de passe : [redacted]

### Création :
    - Menu
    - Gestion des Articles et des Stocks
    - Créer
    - Articles
        - saisir les informations obligatoires :
            - Nom
        - Saisir les informations complémentaires :
            - Code article
            - Unité (liste de valeur à U-Unit par defaut)
            - Prix unitaire
            - Type de TVA
            - Mots cles de recherche
            - Produit Abstrait : False => vérifier que le booléen "tenue en stock" est a true
        - Saisir les sous tableaux :
            - Tarification achat-vente
                - Bien définir des plages de début et de fin ainsi q'un tarif HT et le type de commande
            - Taxe de l'article
            - Usage et TVA de l'article
### Vérification
    - Menu
    - Gestion des Articles et des Stocks
    - Stock par article
    - RECHERCHER
        - Rechercher l'article créé
        - S'assurer que le stock de l'article est bien agrégé à 0

FIN du 1er test
```

```
{
  "env": {
    "dashboard": "/iole#page=-1",
    "createTiers": "/iole#concept=-133&page=-2",
    "displayTiers": "/iole#concept=-133&page=-3",
    "displayOneTiers": "/iole#concept=-133&page=-2&record=71",
    "createArticle": "/iole#concept=-41&page=-2",
    "displayArticles": "/iole#concept=-263&page=-3",
    "displayOneArticles": "/iole#concept=-41&page=-2&record=148",
    "displayStock" : "/iole#concept=-262&page=-2&record=496"
  }
}
```

Ci-dessus les données au format JSON les variables contenant les URL des différentes pages.

```
/// <reference types="Cypress" />

describe("Testing Article Creation", () => {

    // Connexion à Iole :
      beforeEach(() => {
        cy.login('[redacted]', '[redacted]');
      });

  it("Can create an article", () => {
    cy.visit(Cypress.env("workspace") + Cypress.env("dashboard")); // Chargement page d'accueil de Iole

    // Cliquer sur :
    cy.get('top-app-bar').click({force: true}); // Menu
    cy.get('navigation-drawer').click({force: true}); // Gestion des articles et des stocks
    cy.get('navigation-drawer').click({force: true}); // Créer
    cy.get('navigation-drawer').click({force: true}); // Articles
    cy.visit(Cypress.env("workspace") + Cypress.env("createArticle")); // Chargement page création
d'articles

    // Saisir les informations obligatoires :
    cy.get('#iole_input_text_3').click().clear().type('chaisebambou#15'); // Nom de l'article

    // Saisir les informations complémentaires :
    cy.get('#iole_input_text_4').click().clear().type('015'); // Code article

    cy.get('#iole_input_select_-346 > .mdc-select__anchor > .mdc-select__selected-text').click(); //
Unité (liste de valeur à U-Unit par défaut)
    cy.get('#iole_input_text_7').clear().type('40'); // Prix unitaire

    cy.get('#iole_input_select_-3105 > .mdc-select__anchor > .mdc-select__selected-text').click(); //
Type de TVA
    cy.get('[data-value="-1"]').click({force: true});
    cy.get('#iole_input_text_41').clear().type('chaise'); // Mot clé de recherche

    cy.get('[data-visibility="40"] > .switch > label.iole > .lever').click({force: true}); // Tenue en
stock
    cy.get('#-\\33 112').check({force: true}); // Tenue en stock cocher
```

```
  it("Can check article created", () => {
    cy.visit(Cypress.env("workspace") + Cypress.env("dashboard")); // Page d'accueil de Iole

    // Cliquer sur :
    cy.get('top-app-bar').click({force: true}); // Menu
    cy.get('navigation-drawer').click({force: true}); // Gestion des articles et des stocks
    cy.get('navigation-drawer').click({force: true}); // Stock par article
    cy.visit(Cypress.env("workspace") + Cypress.env("displayArticles")); // Chargement de la page stock par article

    cy.wait(5000); // Attente de 5 secondes pour laisser le temps à la page de se charger, sinon le test ne fonctionne pas à cause du temps de réponse
    cy.get(':nth-child(2) > .mdc-tab__ripple').click(); // RECHERCHER

    cy.get('#iole_input_text_1').clear().type('chaisebambou{enter}'); // Saisie du nom de l'article créé dans la barre de recherche puis entrée

    cy.get(':nth-child(1) > .tabulator-frozen > .material-icons').click(); // Click sur l'icône bloc-note de l'article en question
    cy.visit(Cypress.env("workspace") + Cypress.env("displayOneArticles")); // Chargement de la page des informations de l'article créé

    cy.wait(5000); // Attente de 5 secondes pour laisser le temps à la page de se charger, sinon le test ne fonctionne pas à cause du temps de réponse
    cy.get(':nth-child(1) > .tabulator-frozen > .material-icons').click({multiple: true}); // scroller tout en bas pour rejoindre la rubrique stock d'articles par site et cliquer sur chaque icône bloc-note de chaque site pour vérifier les stocks

    cy.visit(Cypress.env("workspace") + Cypress.env("displayStock")); // Chargement de la page stock de l'article du site en question

    cy.wait(5000); // Attente de 5 secondes pour laisser le temps à la page de se charger, sinon le test ne fonctionne pas à cause du temps de réponse
    cy.get('#iole_input_text_24').should('have.value', '0'); // S'assurer que le stock de l'article est bien agrégé à 0
  });
```

En résumé

J'ai effectué au total 5 tests fonctionnels sur la partie comptabilité de l'API de Iole :

- Test du login
- Test du dashboard
- Test d'une création de tiers
- 2 tests sur la création d'un article (dont 1 avec tenue en stock)

Difficulté rencontrées :

J'ai rencontré quelques erreurs dans mes tests, notamment des erreurs de syntaxe. J'ai eu d'autres erreurs un peu plus complexe, comme des URL qui n'étaient pas bon, des id incorrect, des interactions qui ne s'exécutaient pas, ou encore des problèmes de latences sur les serveurs de Iole faisant ainsi échoué mes tests. Le plus difficile était d'identifié et de comprendre la cause de ces erreurs, afin de pouvoir les corriger.

Solutions apportées :

- **Pour les URL incorrect :** j'ai simplement changé le contenu de la variable concerné dans le JSON en indiquant le bon URL.

- **Pour les erreurs d'id :** j'ai simplement ciblé à l'aide de l'open selector playground de Cypress le champ concerné sur l'API, afin d'obtenir le bon id.

- **Pour les problèmes d'interaction :** j'ai eu un souci de clic sur le menu déroulant de l'API pour ma création d'article, l'interaction ne se faisait pas. J'ai trouvé sur le site officiel de Cypress une solution pour corriger ce problème, en forçant simplement le clic sur le champ en question.

- **Pour les problèmes de latence :** La base de données sur laquelle j'étais connecté avait des problèmes de lenteur, empêchant ainsi de charger rapidement les pages de l'API. Lorsque je choisissais dans le menu le champ « création d'articles », la page se chargeait au bout de 5 secondes, et au-delà de 4 secondes de temps de chargement, cypress considère que le délai de chargement de la page est dépassé, et engendrera une erreur faisant ainsi échoué mon test. Pour remédier à ce problème, j'ai simplement rajouté un « cy.wait » de 5000 millisecondes (5 secondes), pour laisser le temps à la page de charger.

METHODOLOGIE AGILE

L'approche agile est un ensemble de pratique appliqué lors d'un projet. Cette approche met le client au centre du système, elle favorise des échanges réguliers entre le client et l'équipe de développement. En effet réduire les intermédiaires entre les concepteurs et le client permet à ce dernier de communiquer ses retours et de redéfinir le cahier des charges plus régulièrement.

Daily Meeting

Chaque matin lorsque nous nous installons sur notre poste de travail, le premier réflexe est de se connecter sur Element et de se rendre dans la salle de réunion public pour se retrouver avec tous les autres développeurs. Le chef de projet donne le coup d'envoi de la réunion, et les développeurs s'expriment à tour de rôle.

L'objectif de cette réunion spéciale consiste à expliquer pour chacun ce que nous avions fait la veille, de ce qu'il nous reste à faire, de faire part des difficultés rencontrés, et éventuellement réclamer de l'aide lorsque nous sommes bloqués, ou encore expliquer que nous avons aidé un collègue à le débloquer d'une tâche complexe.

Après que tous les développeurs se sont exprimés, le chef de projet récapitule ce qui a été dit par chacun, et s'organise pour aller voir les développeurs ayant besoin d'aide, ou en attente d'un nouveau ticket.

Procédure de validation

Lorsque j'ai terminé la tâche confiée par le chef de projet, il faut envoyer mon travail effectué de mon dépôt local sur le dépôt distant en effectuant une merge_request (MR), qui engendrera un ticket sur Gitea, afin que celui-ci soit contrôlé par un développeur compétent, et obtenir son approbation pour incorporer notre travail sur la branche principale définitivement, ou nous le retourner par un ticket de retour pour modifier si nécessaire notre code. Afin d'avoir un suivi performant, il est nécessaire d'indiquer le n° du ticket et de décrire ce que l'on a fait dans la MR.

Mais avant de faire cette action, il faut aller sur la branche principale et récupérer les derniers changements qui ont été envoyées pour ensuite revenir sur ma branche de travail et effectuer la MR. Pour cela, les principales commandes à connaitre pour la MR dans l'ordre sont les suivantes :

- **git checkout <branchePrincipale>**
  → Bascule sur la branche <branchePrincipale> qui correspond au dépôt distant

- **git pull**
  ➔ Récupère les données qui ont été envoyées au dépôt distant

- **git checkout <maBranche>**
  → Bascule sur la branche <maBranche> correspondant à mon dépôt local

- **git add .**
  → Ajoute le contenu de mes fichiers édité ou modifié sur mon dépôt local

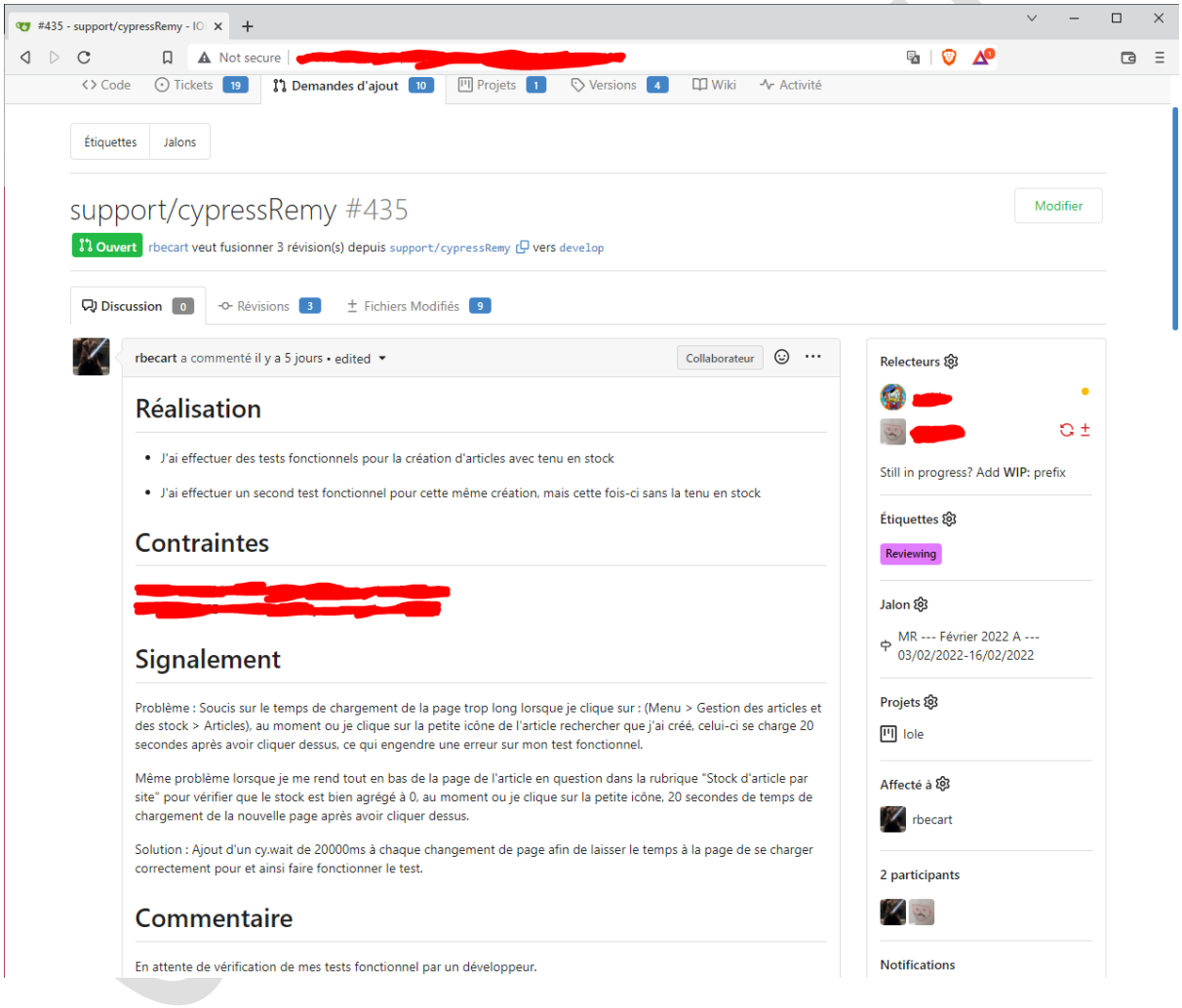
- **git commit –m « mon message »**
  → Enregistre les modifications effectuer sur mon dépôt local avec un message permettant de préciser ce que l'on a fait ou les changements apporter sur nos fichiers

- **git merge <branchePrincipale>**
  → Fusion de mon travail sur la branche principale

- **git push**
  → Envoie le commit sur le dépôt distant

Après avoir exécuter dans le terminal les commandes git pour la MR, il faut aller sur Gitea pour faire une nouvelle demande d'ajout, et ainsi créer un ticket. Pour cela, aller sur l'onglet « demande d'ajout » et cliquer sur « nouvelle demande ».

Une fenêtre de discussion va apparaitre. Sur celle-ci, indiquer la réalisation du travail, les contraintes (la base de données utilisé), un signalement pour expliquer les difficultés rencontrées, et un commentaire pour demande de vérification par un développeur.



Avant de valider la demande pour créer le ticket de contrôle, il faut demander à un des développeurs de l'équipe si on peut le rattacher en relecteur de mon ticket pour contrôler mon travail, et obligatoirement rattacher d'office le chef de projet en supplément qui contrôlera et validera mon ticket.

Renseigner également le nombre d'heure que l'on a passé pour le développement de la tâche, et également dans la partie « jalon » la période sur laquelle nous avons travaillé, puis valider la demande d'ajout.

Les tickets

Le chef de projet s'occupe alors de répartir les tickets aux différents développeurs. Les tickets possèdent un statut pouvant être ouvert, clos, en cours ou nouveau. Il est également assigné à un développeur. On retrouve également d'autres informations sur le ticket :

- **La priorité (du moins important au plus urgent) :**

  COOL : Priorité de développement faible

  Waiting : Priorité de développement normal / Développement en pause

  WARM : Priorité de développement importante

  HOT : Priorité de développement très importante

  BURN : Priorité de développement maximale ! Interrompt les priorités inférieures

- **Le type :**

  feature : Ajout ou modification de fonctionnalité majeure

  hotfix : Ajout ou modification de fonctionnalité mineure

  support : Ajout ou modification d'outil et de fonctionnalité interne

- **La partie concerné (Back ou Front)**

- **Le thème :**

  Pennylan : Intégration API comptable Pennylan

  Functional Test : Test Fonctionnel

  Input : Développement des inputs en front

  Requestor : Création de requêtes

  Invoice : Génération de facture

  Print : Génération de PDF

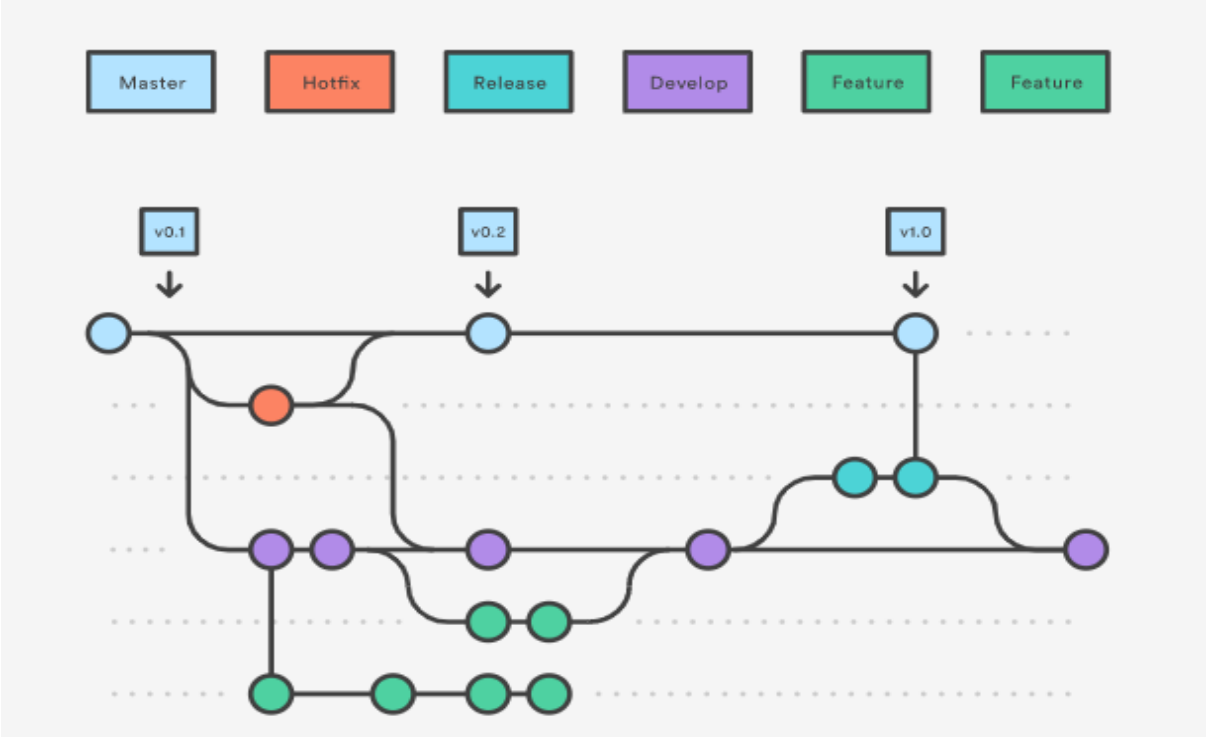
- **La mission à réalisé**

Ces informations permettent aux développeurs de connaître précisément la mission à effectuer, et lorsqu'un développeur se voit assigné plusieurs tickets cela lui permet de s'organiser en traitant en priorité les tickets les plus importants.

Cet organisation par ticket permet d'avoir un suivi précis de l'activité de chaque développeur, de l'avancement des chantiers en cours et le suivi des correctifs par rapport au retour client.

Les branches

L'organisation des branches est structurée selon le gitflow workflow. Il existe des branches principales Master (versions stables de l'application) et Develop (état actuel de l'application). A chaque ticket correspond une nouvelle branche, celle-ci est souvent supprimée une fois le ticket réalisé. Le schéma[1] suivant illustre l'organisation des branches.



CONCLUSION

Ce stage m'a permis de découvrir le quotidien d'un développeur en entreprise, et de découvrir de nouvelles technologies dont je n'avais pas encore connaissance, notamment la découverte de Gitea, mais également Cypress.

Ces 10 semaines d'apprentissage en milieu professionnelle ont été très enrichissante. Elles m'ont permis de prendre davantage confiance en moi, et d'approfondir certaines notions du métier qui n'étaient pas encore comprise en cours, et m'a permis d'avoir une vision plus clair du métier sur le terrain.

Enfin cette expérience m'a permis d'acquérir une belle expérience, courte mais intense, qui me servira tout au long de ma futur carrière de développeur.