
Rapport d'activité

Bewease



Saracousti Mohamed-Amine - BTS SIO SLAM - 2021 / 2022

Présentation de l'organisation "Bewease"

Bewease propose une offre haut de gamme et innovante pour la gestion de la comptabilité matières et des échanges douaniers.

Spécialisé depuis plus de vingt ans dans l'édition de logiciels et les services associés dans le secteur des boissons, vins, alcools et spiritueux, leur objectif est de pouvoir offrir une offre haut de gamme et innovante dans le respect de leur clients et avec un niveau de service optimum.

Sommaire

- *Introduction*
- *Communication*
- *Accises*
- *TestCafe Studio*
- *Mon travail : Test de fonctionnalité*
- *Conclusion*

Introduction

Dans ce rapport, je vais vous présenter toutes les tâches que j'ai effectué durant mon stage de cinq semaines qui me permettra de valider ma première année de **BTS SIO** option **SLAM**.

Ma principale tâche durant ce stage sera de réaliser et de mettre en place des tests fonctionnels sur une application web.

Tester des fonctionnalités comme un simple bouton ou un formulaire afin de détecter les bugs comme si j'étais un client qui se rendait sur le site.

Les tests seront écrits en TypeScript qui ressemble sensiblement au JavaScript.

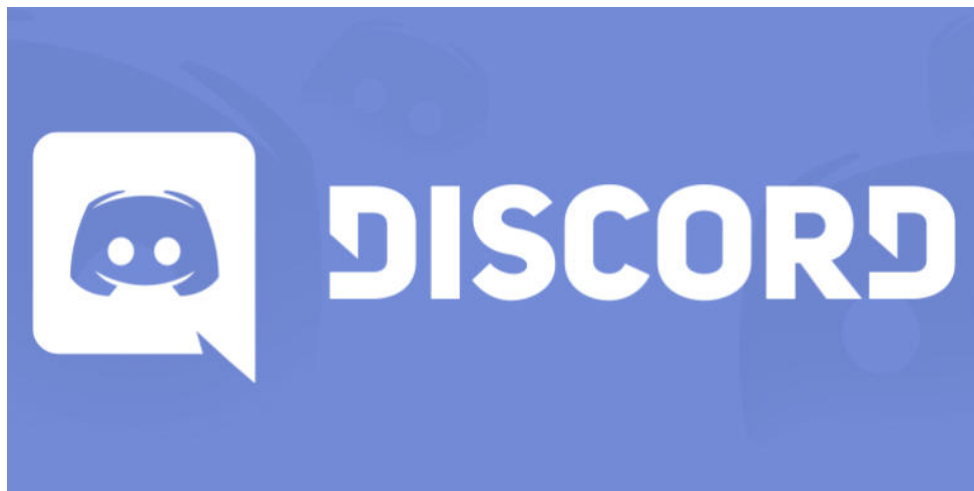
Pour me faciliter la tâche, j'ai eu à ma disposition un outil qui m'a permis de générer une grande partie du code.

J'ai donc pu m'appuyer sur la génération automatique pour mieux appréhender le code avec un logiciel qui se nomme TestCafé Studio.

Communication

Pour communiquer avec l'équipe, nous utilisons un groupe Discord en plusieurs salons où chaque employé peut communiquer avec c'est collègue selon son poste.

Ainsi, étant donné les conditions pour ce stage (travail en distanciel), il était pour moi plutôt simple de demander de l'aide à mon tuteur.



Accises



En tant qu 'entrepositaire agréé, vous devez respecter de nombreuses obligations durant l'exercice de votre activité. La première est de tenir une comptabilité matières, conformément au Code Général des Impôts et aux registres vitivinicoles.

Accises est une solution experte pour la tenue et le suivi de votre comptabilité matières. Elle vous permet de recenser les mouvements de réception, d'expéditions, de détention et de production de marchandises soumis au paiement des droits d'accise. En effet, cette taxe, contribution indirecte, concerne, entre autres, les boissons, vins, alcools et spiritueux.

Le site est développé en majorité en JavaScript et en C# avec des serveurs web IIS.

TestCafe Studio



TestCafe Studio est un outil de test frontal automatisé qui enregistre des tests visuels et a été conçu pour tester des applications Web.

TestCafe Studio enregistre vos actions et crée un test que vous pouvez rejouer ultérieurement.

TestCafe Studio comprend un puissant éditeur de code avec coloration syntaxique et complétion de code intelligent. Vous pouvez modifier les scripts que vous avez enregistrés ou écrire du code à partir de zéro. Vous pouvez également convertir un test sans code en script.

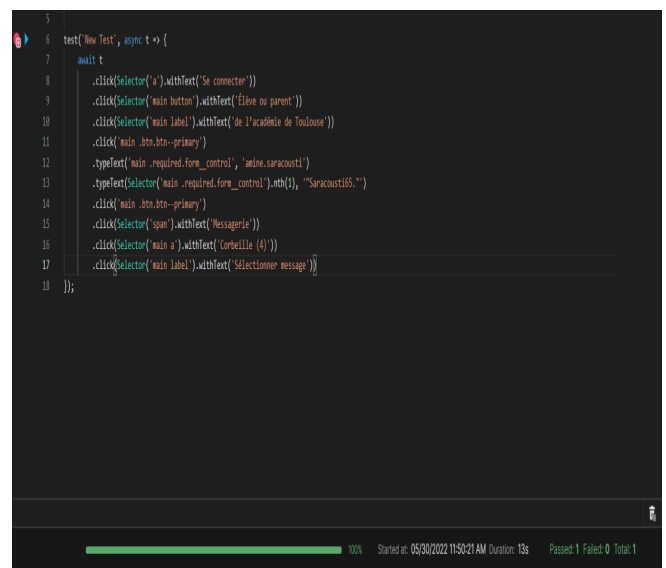
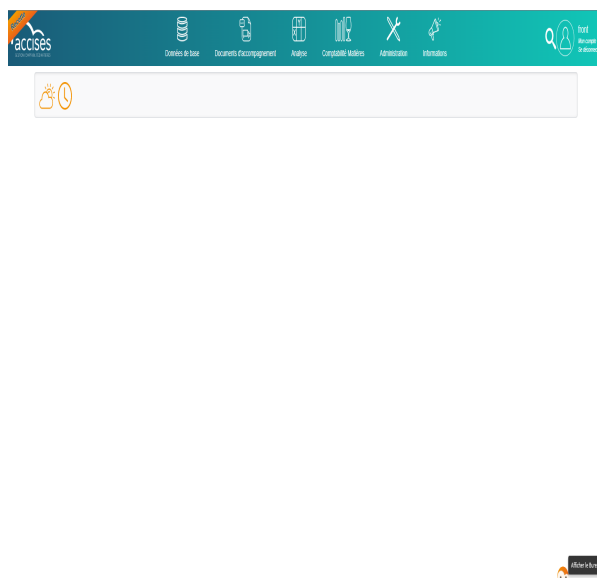
Les actions de test et les assertions utilisent des requêtes Selector pour identifier les éléments par leur classe, leurs attributs, leur ID ou d'autres propriétés.

Lorsque TestCafe Studio enregistre un test, il génère automatiquement des requêtes Selector.

Test de fonctionnalité

J'ai commencé les premiers jours par la prise en main du logiciel et du langage afin de me familiariser au langage qu'utilise l'équipe de développeurs.

J'avais donc deux outils avec moi, le produit qui est Accises et le logiciel Testcafe Studio qui me permet d'effectuer les tests.

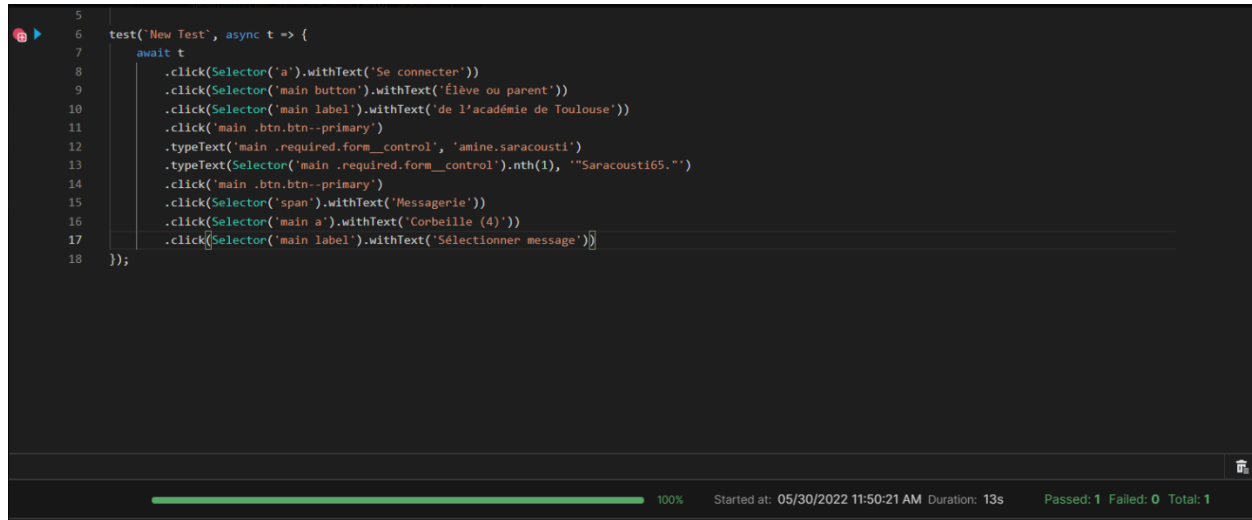


Sur TestCafe, je pouvais générer du code selon deux façons, soit je l'écrivais à la main mais cela prenait plus de temps, soit je le faisais enregistrer grâce à une fonction du logiciel (et souvent les deux).

Cela fonctionne comme un enregistrement d'écran et lorsqu'il est démarré, il écrit toutes mes actions sur le site sous forme de code en TypeScript.

Après avoir écrit tous mon code, je peux vérifier si cela fonctionne en lançant un test qui va exécuter mon code comme un robot sur le site.

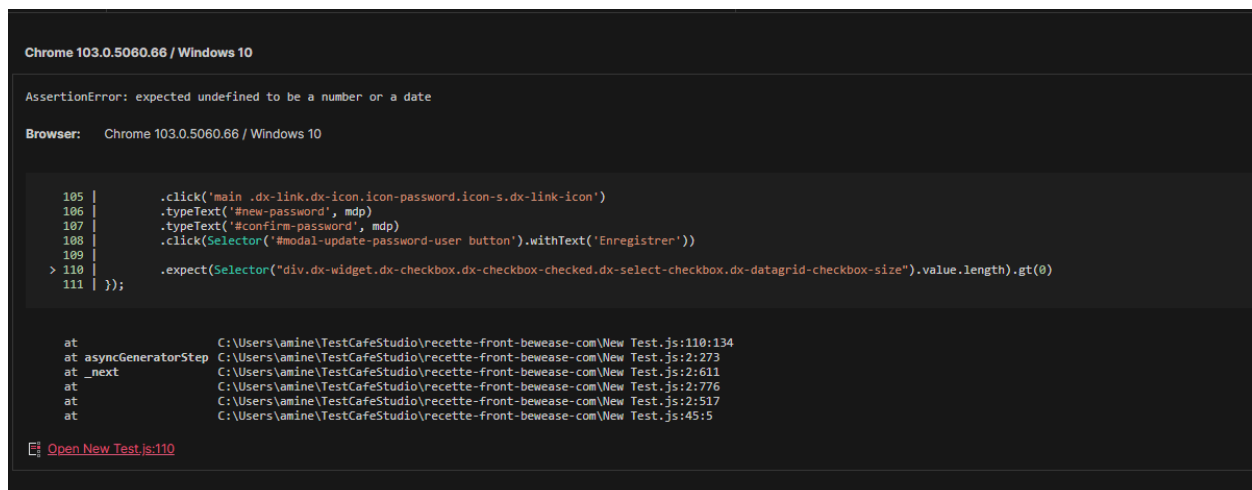
A la fin de son exécution, le logiciel m'indique si toutes les lignes de code ont été exécutés ou non.



```
5
6 test('New Test', async t => {
7   await t
8     .click(Selector('a').withText('Se connecter'))
9     .click(Selector('main button').withText('Élève ou parent'))
10    .click(Selector('main label').withText('de l'académie de Toulouse'))
11    .click('main .btn.btn--primary')
12    .typeText('main .required.form__control', 'amine.saracousti')
13    .typeText(Selector('main .required.form__control').nth(1), "Saracousti65.")
14    .click('main .btn.btn--primary')
15    .click(Selector('span').withText('Messagerie'))
16    .click(Selector('main a').withText('Corbeille (4)'))
17    .click(Selector('main label').withText('Sélectionner message'))
18  });
```

100% Started at: 05/30/2022 11:50:21 AM Duration: 13s Passed: 1 Failed: 0 Total: 1

C'est comme ça qu'un test est validé ou pas, et si il y a une erreur, alors la ligne et le type d'erreur nous sera indiqué.



```
Chrome 103.0.5060.66 / Windows 10

AssertionError: expected undefined to be a number or a date

Browser: Chrome 103.0.5060.66 / Windows 10

105 |         .click('main .dx-link.dx-icon.icon-password.icon-s.dx-link-icon')
106 |         .typeText('#new-password', mdp)
107 |         .typeText('#confirm-password', mdp)
108 |         .click(Selector('#modal-update-password-user button').withText('Enregistrer'))
109 |
> 110 |         .expect(Selector("div.dx-widget.dx-checkbox.dx-checkbox-checked.dx-select-checkbox.dx-datagrid-checkbox-size").value.length).gt(0)
    |               ^
111 |     });

at C:\Users\amine\TestCafeStudio\recette-front-bewease-com\New Test.js:110:134
at asyncGeneratorStep C:\Users\amine\TestCafeStudio\recette-front-bewease-com\New Test.js:2:273
at _next C:\Users\amine\TestCafeStudio\recette-front-bewease-com\New Test.js:2:611
at C:\Users\amine\TestCafeStudio\recette-front-bewease-com\New Test.js:2:776
at C:\Users\amine\TestCafeStudio\recette-front-bewease-com\New Test.js:2:517
at C:\Users\amine\TestCafeStudio\recette-front-bewease-com\New Test.js:45:5

Open New Test.js:110
```

En plus d'apprendre les lignes de base dans ce langage, j'ai aussi dû revoir les sélecteurs en CSS et Javascript qui sont primordiaux dans cette activité.

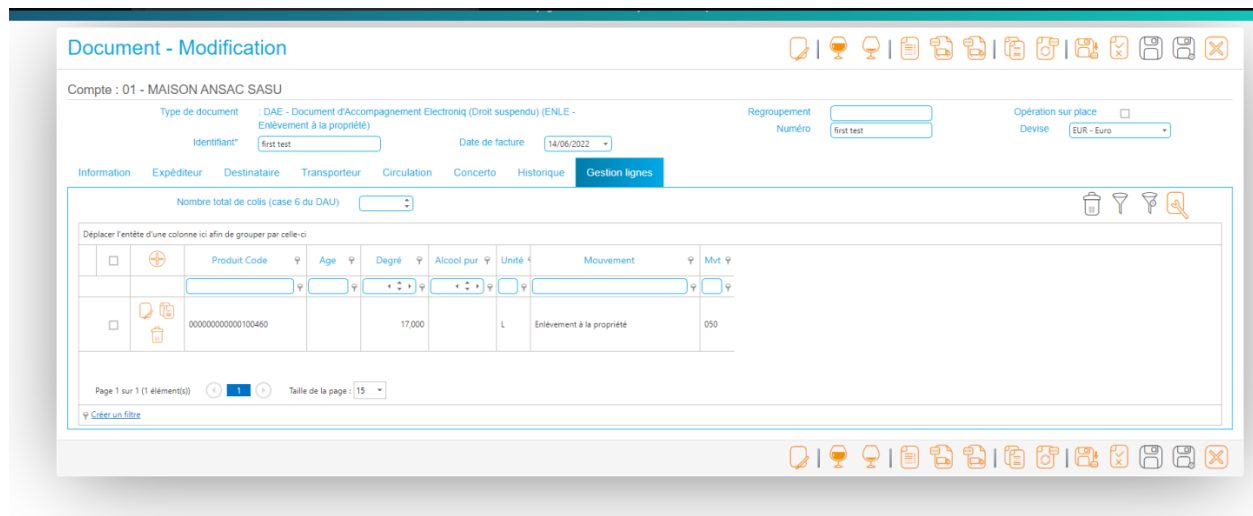
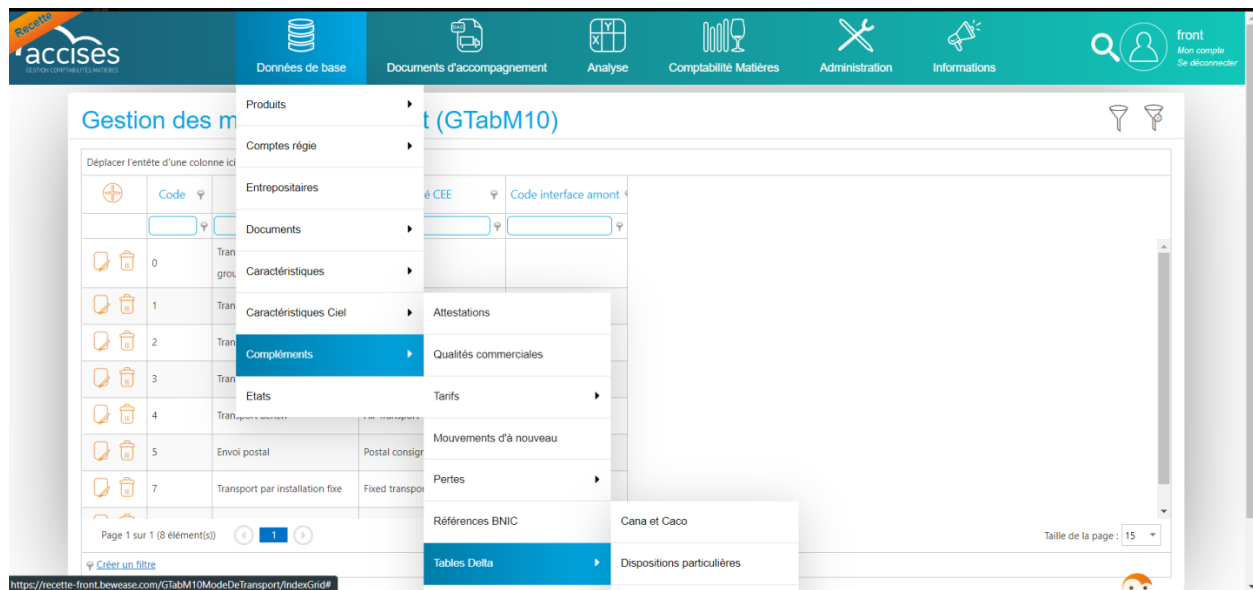
Afin de cibler un élément pour le tester, nous devons absolument trouver son sélecteur qui cible exactement l'élément choisi.

The screenshot shows a web application titled "Gestion des tarifs associés aux catégories". The interface includes a header with a navigation bar and a main content area with a table. The table has two columns: "Catégorie tarifaire" and "Tarif associé". The table contains several rows of data, including "005 - Tarif Rhum", "004 - Tarif Alcool", "011 - Tarif Vodka", "008 - Tarif V. Mousseux", "004 - Tarif Alcool", "010 - Tarif Cognac", and "011 - Tarif Vodka".

On the right side, the Chrome DevTools "Elements" panel is open, showing the DOM tree. The selected element is a table row with the class "dxgvDataRow_MetropolisBlue". The "Styles" panel on the right shows the default styles for this element, including "border-bottom: 1px solid #ccc", "border-right: 1px solid #ccc", "border-top: 1px solid #ccc", and "border-left: 1px solid #ccc".

Ma mission était donc de tester tous les menus du site et tous les formulaires.

Ce travail est essentiel pour l'équipe de développeurs car désormais, ils vont utiliser les tests que j'ai effectués pour tester chaque option du site après sa mise à jour.



Au fur et à mesure des tests, j'ai dû ajouter dans le code de nouvelle commande.

Les variables m'ont permis de gagner beaucoup de temps car étant donné que chaque formulaire devait contenir des données uniques, il fallait pour chaque enregistrement ajouter de nouvelles informations.

```
test('New Test', async t => {
  const identifiant = "Qfsorbt";
  const ad_email = "c0mf@grail.coom";
  await t
    .typeText('#UserName', username)
    .typeText('#Password', password)
    .click(Selector('#form-connexion button').withText('Connexion'))

    .click(Selector('#nav ul li').nth(112).find('div').nth(1).find('svg'))
    .click(Selector('#nav a').withText('Gestion des utilisateurs'))
    .click('main .dx-icon.icon-add.icon-s')

    .typeText('#identifiant', identifiant)
    .typeText('#nom', 'nm')
    .typeText('#prenom', 'pm')
    .typeText('#email', ad_email)
    .typeText('#telephone', '07589632')
    .typeText('#opérateur-amont', 'b')
    .click('#InfoGene .dx-dropdowneditor-icon')

    .typeText('#InfoGene .dx-texteditor-input', 'administrateur')
    .pressKey('down')
    .pressKey('enter')

    .click('#concerto-notif-erreurs-techniques')
    .click('#interface-donnees-notif-erreurs-techniques')
    .click('#interface-donnees-email-erreurs-techniques')
    .click(Selector('#myTab a').withText('Droits d\'accès aux comptes'))
    .click(Selector('#grid-comptes .dx-checkbox-icon').nth(1))
    .click(Selector('#myTab a').withText('Droits d\'accès aux modules'))
    .click(Selector('#TreeListDroits table tbody tr td').nth(1).find('input'))
    .click(Selector('#myTab a').withText('Informations générales'))
    .click(Selector('#modal-edit-user button').withText('Enregistrer'))

    .click(Selector('main .dx-texteditor-input').nth(1))
    .pressKey('ctrl+a')
    .typeText(Selector('main .dx-texteditor-input').nth(1), identifiant)
    .click('main .dx-link.dx-icon.icon-edit.icon-s.dx-link-icon')
    .expect(Selector('td.dx-col-33').withText(identifiant).innerText).eql(identifiant)
    .expect(Selector("td.tr.dx-row.dx-data-row.dx-column-lines>td.dx.col-col-33").getAttribute("value")).eql(identifiant)
});
```

```
.expect(Selector('td.dx-col-33').withText(identifiant).innerText).eql(identifiant)
```

Cette commande (`.expect`) est parmi les plus importantes de tout mon code car elle me confirme que le test à bien fonctionné en vérifiant les données dans le formulaire.

Elle va valider le test.

En l'occurrence, sur l'exemple ci-dessus, la commande veut dire : sur le sélecteur "td.dx-col-33" est ce qu'il y a bien l'identifiant qui est connu grâce à la variable.

Mes premiers test était plutôt simple et je commenter chaque block :

```
54 test('produits de regroupement', async t => {
55     await t
56     //Page de connexion
57     .typeText('#UserName', 'front')
58     .typeText('#Password', 'T6QqHp56w5UyvA2!')
59     .click(Selector('#form-connexion button').withText('Connexion'))
60     //Menu deroulant
61     .click('#nav .section-icon')
62     .click(Selector('#nav a').withText('Produits'))
63     .click(Selector('#nav span').withText('Produits de regroupement'))
64     //Creer un nouveau formulaire
65     .click('main .icon-add.icon-s')
66     //Remplir le formulaire
67     .typeText('main .dxeEditArea_MetropolisBlue.dxeEditAreaSys', '973')
68     .typeText(Selector('main .dxeEditArea_MetropolisBlue.dxeEditAreaSys').nth(1), 'j')
69     .click('main .dxeButton.dxeButtonEditButton_MetropolisBlue')
70     .click(Selector('main td').withText('Compte 6 mois').nth(2))
71     .click(Selector('main button').withText('Enregistrer'))
72     //Chercher le code dans le filtre
73     .click('main .dxeEditArea_MetropolisBlue.dxeEditAreaSys')
74     .pressKey('ctrl+a')
75     .pressKey('backspace')
76     .typeText('main .dxeEditArea_MetropolisBlue.dxeEditAreaSys', '973')
77     .pressKey('enter')
78     //Commande expect
79     .expect(Selector('td.dxic').withText('973').innerText).eql('973')
80 });
```

J'ai dû ajouter et modifier d'autres lignes de code dans les tests au fur et à mesure. Par exemple, chaque formulaire doit contenir des données uniques donc il fallait écrire la commande suivante afin d'effacer les précédentes données :

```
.pressKey('ctrl+a')  
.pressKey('backspace')
```

Ensuite, j'ai aussi dû faire en sorte qu'à la fin du test, il soit supprimé afin de ne pas encombrer la base de données étant donné que ce ne sont que des tests qui ont pour seul but de tester la fiabilité des formulaires et du site.

Conclusion

Pour conclure, ce stage fut mon premier pas dans le monde professionnel du développement web.

J'ai énormément progressé en Javascript et TypeScript que je ne maîtriser pas du tout avant.

J'ai aussi appris à travailler au même rythme que mes collègues en utilisant des méthodes de travail comme notamment la méthode Agile qui me permet de me fixer des objectifs court terme.

J'ai aussi dû apprendre à bien communiquer car il fallait tout le temps tenir au courant mon tuteur de mes avancées et de mes problèmes à travers des réunions quotidiennes et par message.