Suivi formation KRA

du 05/06/2023 au 31/07/2023

# Sommaire

[Sommaire 1](#_jcx9c7e15u3u)

[Classification 2](#_7yi624e8o8xb)

[Objet du document 2](#_egfmc08mws47)

[Versions du document 2](#_gnk96s2mkmp8)

[05/06/2023 3](#_6c0ll8oe7ut8)

[06/06/2023 3](#_a83gm9en5k5w)

[07/06/2023 3](#_j4bvd1sw9eww)

[09/06/2023 3](#_5q5lxfbqyv8o)

[12/06/2023 3](#_oigsncmewmhk)

[13/06/2023 4](#_jie965sqowvr)

[15/06/2023 4](#_j3ih4r5apz05)

[19/06/2023 4](#_fq7rtj5w4sh4)

[20/06/2023 4](#_nued5vc474d4)

[21/06/2023 4](#_bzxgpfy6qlff)

[**22/06/2023 5**](#_xoedt3b3w4ju)

[**23/06/2023 5**](#_oka7cng1aq2l)

[**27/06/2023 5**](#_h9fsnunqeb8q)

[**28/06/2023 6**](#_yrukrqxzingw)

[**29/06/2023 6**](#_i5fe859fbonj)

[**30/06/2023 7**](#_cns9awyjh0rk)

[**03/07/2023 7**](#_6h62jgo5bxxj)

[**04/07/2023 7**](#_dk7t9jb4mo1h)

[**05/07/2023 8**](#_4g1c5xpsfain)

[**06/07/2023 8**](#_p2pizdqi24rl)

[**07/07/2023 9**](#_ncx2eg4tdbgv)

[**10/07/2023 9**](#_t472m4eerbur)

[**11/07/2023 9**](#_i34nnbg6p3l5)

[**12/07/2023 10**](#_x45w1f61qnmt)

[**13/07/2023 10**](#_ewvgcdgnesld)

[**14/07/2023 10**](#_cy0o4z2iqbtk)

[**17/07/2023 10**](#_7bxve53p3d53)

[**18/07/2023 11**](#_2gy2o4l0lqrv)

[**19/07/2023 11**](#_hiojozd6is9x)

[**20/07/2023 11**](#_qw2zltpr8xtm)

[**21/07/2023 13**](#_muaps0toc5z2)

[**24/07/2023 13**](#_3juj3wja10g7)

[**25/07/2023 14**](#_nfu1xbhji7rj)

[**26/07/2023 15**](#_hfvpazke0n3k)

[**27/07/2023 15**](#_81hustoplhkr)

[**28/07/2023 15**](#_eksttjno2v3c)

[Liste des actions 16](#_en9vtxhu1wz)

[Tests Automatisés 1](#_f4uotnirrao6)

[1.1. Communication au sein de l’équipe projet 1](#_q9zgzkiiqfrx)

[1.2. Communication hors équipe projet 1](#_678i81pm6oru)

[1.3. Ressources fournies et/ou à utiliser 1](#_wv5ustq2br4w)

[5.6. Les outils 1](#_g9ivn1nxm5b6)

[5.7. Méthodes de travail en projet collaboratif 1](#_qwoob86pvvql)

[Analyse des tâches continues effectuées lors de mon stage chez Epiconcept Océan Indien : 1](#_l0gsdgqtkxbd)

[· Tâche AUTO-34 - Lancer le navigateur et accéder à l'URL de l'application : 1](#_8l24l6ran8xv)

[· Tâche AUTO-36 - Saisir les login/MDP et se connecter à l'application : 1](#_qwewg8vnxkwa)

[· Tâche AUTO-37 - Implémenter le test de connexion sur une autre application : 1](#_kuyhc3rdar8o)

[· Tâche AUTO-38 - [e2e] [NEONATLABO] Tester la redirection vers la bonne page : 1](#_l8pfei6dwedn)

[**Remerciements 1**](#_3xyx2umlh7dv)

# 

# Classification

|  |  |  |  |
| --- | --- | --- | --- |
| **Secret** | **Interne** | **Confidentiel** | **Public** |
|  | X |  |  |

# Objet du document

Ce document contient les détails des formations et connaissances transmises à Kim pendant son stage. Il sera maintenu par Kim tout au long qui s’assurera de compléter à chaque fin de journée le contenu en fonction.

# Versions du document

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Auteur** | **Notes de version** |
| 09/06/2023 | 1.0 | VEH | Initiation du document |
| 12/06/2023 | 1.1 | KRA | Formation |
| 15/06/2023 | 1.2 | KRA | Initiation aux tests automatisés |
| 19/06/2023 | 1.3 | KRA | Git & GitHub acquis |
| 20/06/2023 | 1.4 | KRA | Première tâche : AUTO-34 - Lancer le navigateur et accéder à l'URL de l'application |
| 27/06/2023 | 1.5 | KRA | 1ère tâche accomplie  2ème tâche : AUTO 36 - Saisir les login/MDP et se connecter à l'application |
| 06/07/2023 | 1.6 | KRA | 2ème tâche accomplie |
| 07/07/2023 | 1.7 | KRA | 3ème tâche : AUTO 37 - Implémenter le test de connexion sur une autre application |
| 12/07/2023 | 1.8 | KRA | 4ème tâche : AUTO-38 - [e2e] [NEONATLABO] Tester la redirection vers la bonne page |

# 

# 05/06/2023

* Présentation
* Création des comptes Googles suites
* Compte professionnels
* Google chat
* Création de compte Github

# 06/06/2023

* Formation : Méthode AGILE Scrum
* Formation Google Suite
* GitHub :
  + joindre @Epiconcept- Paris Organization
  + Ajout dans la team : “Team OI”

# 07/06/2023

* Voo4 : Noyau, utilisateurs, Groupes et rôles Voozanoo
* Passbolt : Formation et utilisation
* Git - GitHub : Commandes Git Bash

# 09/06/2023

* Entretien pour mise au point sur les acquis
* Récupération des projets sur GitHub : Epiconcept-Paris/oi-automated-tests, Epiconcept-Paris/fwk-automated-tests repository
* Visionnage des deux projets

# 12/06/2023

* Voozanoo : création groupe, rôle et utilisateur
* Documentation :
  + Tutoriel Git, GitHub
  + je me suis un peu baladé sur <https://epiconcept-paris.github.io/epidocs/users_management.html> , l’interface Voozanoo 4
* Formation : epifiles.voozanoo.net (envoie de fichiers sécurisé)

# 13/06/2023

* Git - GitHub : réussi finalement à récupérer, sauvegarder, stocker les fichiers en lignes, héberger site sur GitHub

# 15/06/2023

* Initiation aux tests automatisés
* Python, selenium, selenium base
  + Installation des librairies nécessaires
  + <https://docs.pytest.org/en/7.1.x/getting-started.html#install-pytest>
  + <https://pypi.org/project/selenium-base/>

# 19/06/2023

* Git & GitHub : création de Branch et Pull Request à faire
* Création de branche et pull request complétés

# 20/06/2023

* Création de branche et pull request complétée
* Première tâche attribuée : <https://epiconcept.atlassian.net/browse/AUTO-34>
  + AUTO-34 - Lancer le navigateur et accéder à l'URL de l'application

# 21/06/2023

* En cours de réalisation de la tâche : AUTO-34 - Lancer le navigateur et accéder à l'URL de l'application
* Problème sur l’installation de pytest

# 22/06/2023

* Problème d’installation de pytest résolu
* Attribution de la première tâche : [Auto-34 - Lancer le navigateur et accéder à l'URL de l'application](https://github.com/Epiconcept-Paris/oi-automated-tests/pull/3) :

# 23/06/2023

* Journée de détente

# 27/06/2023

* Première tâche accomplie :
  + Développement de la tâche Auto-34 : Lancer le navigateur et accéder à l'URL de l'application :

Au cours de mon stage chez Epiconcept, j'ai été attribué à la tâche Auto-34, qui consistait à développer la fonctionnalité permettant de lancer le navigateur et accéder à l'URL de l'application.

Pour réaliser cette tâche, j'ai utilisé la bibliothèque Selenium, qui est un outil de test automatisé pour les applications web. J'ai également exploité le module webdriver de Selenium, qui fournit des fonctionnalités pour interagir avec les navigateurs web.

Tout d'abord, j'ai configuré le navigateur Chrome en utilisant la classe webdriver.Chrome(). Cette classe permet d'instancier un objet représentant le navigateur Chrome.

Ensuite, j'ai utilisé la variable url provenant du module vars pour spécifier l'URL de l'application à laquelle je devais accéder.

En appelant la méthode get() sur l'objet du navigateur, j'ai pu ouvrir le navigateur et charger l'URL spécifiée. Cela m'a permis d'accéder à l'application dans le navigateur et d'initialiser l'environnement de test.

Afin d'éviter la fermeture automatique du navigateur, j'ai ajouté une boucle while True pour maintenir le programme en cours d'exécution indéfiniment. Cela permet de conserver le navigateur ouvert même après avoir chargé l'URL de l'application.

En conclusion, j'ai réussi à développer la première tâche assignée (Auto-34) en utilisant Selenium et le module webdriver. J'ai pu lancer le navigateur, accéder à l'URL de l'application et maintenir le navigateur ouvert pour permettre l'exécution de tâches ultérieures. Cette fonctionnalité est essentielle pour automatiser les tests de l'application web et faciliter le processus de validation.

* Attribution de la deuxième tâche : AUTO 36 - Saisir les login/MDP et se connecter à l'application

# 28/06/2023

* Tâche en cours : AUTO 36 - Saisir les login/MDP et se connecter à l'application

Objectif de tâche accomplie avec Selenium

* Développement de la tâche avec Seleniumbase

# 29/06/2023

* Tâche : AUTO 36 - Saisir les login/MDP et se connecter à l'application

Objectif de tâche accomplie avec Seleniumbase :

Lors de l'adaptation du code de Selenium à SeleniumBase pour accomplir l'objectif de la tâche qui m'a été attribuée, j'ai effectué les modifications suivantes :

Tout d'abord, j'ai utilisé la méthode open de SeleniumBase pour accéder à l'URL souhaitée. Cette méthode simplifie l'ouverture de l'URL en remplaçant une ligne de code par self.open(url), ce qui rend le code plus clair et concis.

Ensuite, pour remplir les champs de connexion automatiquement, j'ai utilisé les méthodes type et press\_enter fournies par SeleniumBase. La méthode type m'a permis de saisir rapidement le nom d'utilisateur et le mot de passe en remplaçant plusieurs lignes de code par une seule instruction. Par exemple, au lieu d'écrire :

username\_input = driver.find\_element("css selector", "[name='login']")

username\_input.send\_keys(username)

J’ai simplement utilisé :

self.type('[name="login"]', username)

De plus, j'ai utilisé la méthode press\_enter pour simuler l'appui sur la touche Entrée et soumettre le formulaire de connexion.

Plutôt que d'utiliser time.sleep pour attendre pendant une durée fixe, j'ai utilisé la méthode wait\_for\_element de SeleniumBase. Cette méthode m'a permis d'attendre que l'élément spécifié soit présent sur la page avant de continuer l'exécution du code. Cela rend l'attente plus précise et évite d'utiliser des délais arbitraires.

Grâce à ces modifications, j'ai réussi à reproduire les fonctionnalités du code initial basé sur Selenium en utilisant SeleniumBase. J'ai ainsi pu automatiser la saisie des identifiants de connexion sur une page web et soumettre le formulaire, conformément à l'objectif de ma tâche. De plus, ces adaptations ont simplifié le code en utilisant les méthodes fournies par SeleniumBase, ce qui rend le code plus facile à maintenir et à lire.

# 30/06/2023

* Tâche accomplie : AUTO 36 - Saisir les login/MDP et se connecter à l'application
* En attente de validation de la tâche, et attente de Meeting pour la démo
* Rapport de stage

# 03/07/2023

* Tâche : AUTO 36 - Saisir les login/MDP et se connecter à l'application

La commande de lancement est : python

Nouvelle demande : commande de lancement avec “pytest”

# 04/07/2023

* Tâche : AUTO 36 - Saisir les login/MDP et se connecter à l'application

En cours : pour utilisation de “pytest” comme commande de lancement

* Erreur rencontré :   
  ERROR: usage: pytest [options] [file\_or\_dir] [file\_or\_dir] [...]

pytest: error: unrecognized arguments: --username=k.ratomahenina --password=Hotelcalifornia11..

inifile: None

rootdir: D:\MCCI SIO\Stages\Epiconcept\oi-automated-tests\SboxIo

Pytest ne reconnaît pas les arguments --username et --password tels quels.

* Nouvelle solution trouvée : Définir les variables d'environnement USERNAME et PASSWORD avant d'exécuter le script. Les définir dans PowerShell :

$env:USERNAME="username"

$env:PASSWORD="Mot de passe"

pytest -s TEST.py

# 05/07/2023

* Tâche : AUTO 36 - Saisir les login/MDP et se connecter à l'application
* Pas de solution pour avoir comme commande de lancement :

python TEST.py --username XXX --password XXX

# 06/07/2023

* Tâche : AUTO 36 - Saisir les login/MDP et se connecter à l'application
* Solution trouvée :

Définir les variables d'environnement USERNAME et PASSWORD avant d'exécuter le script par la commande de lancement pytest

$env:USERNAME="XXX"

$env:PASSWORD="XXX"

pytest -s TEST.py

* Nouvelle demande pour la tâche vu qu’il est impossible d’ajouter les arguments pour username et password dans la commande de lancement sur pytest.
* Nouvelle demande : Définir les username et password dans vars.py :

J'ai effectué une modification importante dans mon script de test automatisé afin de rendre la gestion des identifiants de connexion plus flexible. J'ai accompli cela en ajoutant l'username et le password en tant que variables dans le fichier vars.py.

Pour commencer, j'ai ouvert le fichier vars.py qui contenait déjà une variable url pour l'URL de la page à tester. J'ai simplement ajouté deux nouvelles variables : username et password. J'ai assigné les valeurs correspondantes à ces variables en utilisant les identifiants fournis : username = "XXX" et password = "XXX".

Une fois que j'ai ajouté ces variables dans vars.py, je suis retourné à mon script de test principal, TEST.py. J'ai importé la variable url de vars.py comme je le faisais déjà auparavant. Ensuite, j'ai ajouté deux nouvelles lignes pour importer les variables username et password depuis vars.py.

Ensuite, j'ai utilisé ces variables dans la méthode test\_login() de ma classe de test MyTestClass. Au lieu de spécifier directement les identifiants de connexion dans le code, j'ai utilisé les variables username et password pour remplir les champs de connexion automatiquement.

* Tâche accomplie

# 07/07/2023

* Tâche : AUTO 36 - Saisir les login/MDP et se connecter à l'application
  + Tâche accomplie
  + Démo accomplie
* Tâche : AUTO 37 - Implémenter le test de connexion sur une autre application

# 10/07/2023

* Tâche en cours : AUTO 37 - Implémenter le test de connexion sur une autre application

# 11/07/2023

* Tâche accomplie : AUTO 37 - Implémenter le test de connexion sur une autre application

J'ai créé un fichier login\_logic.py dans le répertoire Logique pour mutualiser la logique de connexion. Dans ce fichier, j'ai défini une classe LoginLogic qui contient la méthode def\_login(), responsable de la logique de connexion. Cette méthode utilise les variables url, username et password importées depuis le fichier vars.py spécifique à chaque application.

Les fichiers TEST.py de chaque application, SboxIo et NeonatLabo, importent la classe LoginLogic du fichier login\_logic.py et créent une instance de cette classe. Ensuite, ils appellent la méthode def\_login() de l'instance pour effectuer la connexion spécifique à chaque application.

Cette approche permet d'éviter les duplications de code, car la logique de connexion est centralisée dans le fichier login\_logic.py et peut être appelée à partir des deux fichiers TEST.py.

Ainsi, si des modifications sont nécessaires dans la logique de connexion, elles peuvent être effectuées à un seul endroit, dans le fichier login\_logic.py, ce qui assure une maintenance plus facile et évite la duplication de code.

En utilisant la notion de programmation orientée objet (OOP) de Python, j'ai encapsulé la logique de connexion dans une classe réutilisable et créé des instances de cette classe dans chaque fichier TEST.py. Cela permet d'organiser et de structurer le code de manière modulaire et facilite les mises à jour et les modifications futures.

Grâce à cette approche, les tests de connexion pour SboxIo et NeonatLabo partagent la même logique de connexion, réduisant ainsi la duplication de code et permettant des modifications plus faciles et plus efficaces.

# 12/07/2023

* Tâche : AUTO-38 - [e2e][NEONATLABO] Tester la redirection vers la bonne page

Étude de réalisation de la tâche   
 Des essaies mais sans succès pour le moment

# 13/07/2023

* Tâche : AUTO-38 - [e2e][NEONATLABO] Tester la redirection vers la bonne page

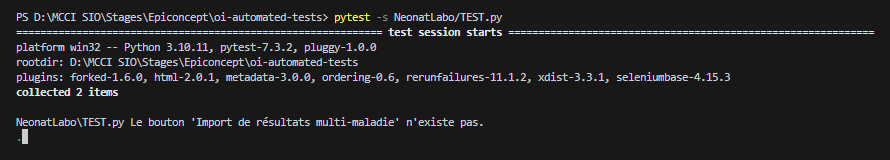
Étude de réalisation de la tâche   
 Des essaies mais sans succès pour le moment

# 14/07/2023

* Tâche : AUTO-38 - [e2e][NEONATLABO] Tester la redirection vers la bonne page :

Affichage des messages de rapport réussie

Les tests sont encore en cours

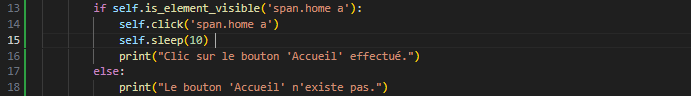


# 17/07/2023

* Tâche : AUTO-38 - [e2e][NEONATLABO] Tester la redirection vers la bonne page :

Affichage des messages de rapport réussie

Le test clique automatique sur un bouton fonctionne avec le bouton “Accueil” (pour un essai), mais non pas pour le bouton “Import de résultats”.



* En cours de réalisation du test pour le bouton “Import de résultats”

Complication sur les codes sources du bouton lui-même sur l’application

# 18/07/2023

* Tâche : AUTO-38 - [e2e][NEONATLABO] Tester la redirection vers la bonne page   
  Point avec [Vanina ELLAYAH](mailto:v.ellayah@epiconcept.io)pour la difficulté rencontrée

# 19/07/2023

* Tâche : AUTO-38 - [e2e][NEONATLABO] Tester la redirection vers la bonne page

# 20/07/2023

* Tâche : AUTO-38 - [e2e][NEONATLABO] Tester la redirection vers la bonne page
* Tâche accomplie

Dans cette nouvelle demande, l'objectif est d'automatiser une série d'actions sur une page web et de vérifier la présence d'éléments spécifiques. Voici les étapes réalisées :

Sélectionner "HBS" (en exemple) dans le bouton select : Un menu déroulant (balise select) contenant plusieurs options est présent sur la page. L'option "HBS" doit être sélectionnée dans ce menu.

Attente de 3 secondes : Après avoir sélectionné l'option "HBS", le script attend 3 secondes pour permettre à la page de se mettre à jour et d'afficher de nouveaux éléments, le cas échéant.

Vérifier la visibilité du bouton "Import des résultats" : Le script vérifie si le bouton ayant l'attribut href égal à "result/import" est visible sur la page. Si oui, le script effectue un clic sur ce bouton et attend à nouveau 3 secondes.

Affichage d'un rapport pour le bouton "Import des résultats" : Si le bouton est trouvé et cliqué avec succès, un message indiquant "Bouton 'Import des résultats' trouvé et cliqué." est affiché dans le terminal. Sinon, un message "Le bouton 'Import des résultats' n'existe pas." est affiché.

Vérifier la présence du texte "Import des résultats" sur la nouvelle page : Le script vérifie si le texte "Import des résultats" est présent sur la nouvelle page qui s'affiche après avoir cliqué sur le bouton "Import des résultats".

Affichage d'un rapport pour le texte "Import des résultats" : Si le texte "Import des résultats" est présent, il est mis en surbrillance (sélectionné comme un triple clique pour le mettre en bleu). Ensuite, un message "Le titre 'Import des résultats' est présent sur la nouvelle page." est affiché dans le terminal. Si le texte n'est pas trouvé, un message "Le titre 'Import des résultats' n'est pas présent sur la nouvelle page." est affiché.

Explication du code :

Le code est écrit en Python avec l'utilisation de la bibliothèque SeleniumBase pour l'automatisation du navigateur web. Voici comment les différentes étapes sont implémentées dans le code :

Sélectionner "HBS" dans le bouton select :

self.select\_option\_by\_text("select.form-control", "HBS")

Cette ligne de code utilise la méthode select\_option\_by\_text() fournie par la bibliothèque SeleniumBase pour sélectionner l'option "HBS" dans le menu déroulant identifié par la balise select.form-control.

Attente de 3 secondes :

self.sleep(3)

Cette ligne de code utilise la méthode sleep() pour faire une pause de 3 secondes avant de continuer à exécuter le reste du script.

Vérifier la visibilité du bouton "Import des résultats" :

if self.is\_element\_visible("a[href='result/import']"):

self.click("a[href='result/import']")

self.sleep(3)

print("Bouton 'Import des résultats' trouvé et cliqué.")

else:

print("Le bouton 'Import des résultats' n'existe pas.")

Ces lignes de code vérifient si le bouton ayant l'attribut href égal à "result/import" est visible à l'aide de la méthode is\_element\_visible(). Si le bouton est trouvé, il est cliqué à l'aide de self.click() et une pause de 3 secondes est effectuée. Un message approprié est ensuite affiché dans le terminal en fonction du résultat.

Vérifier la présence du texte "Import des résultats" sur la nouvelle page :

if self.is\_text\_visible("Import des résultats"):

self.highlight("Import des résultats")

print("Le titre 'Import des résultats' est présent sur la nouvelle page.")

else:

print("Le titre 'Import des résultats' n'est pas présent sur la nouvelle page.")

Ces lignes de code vérifient si le texte "Import des résultats" est présent sur la nouvelle page en utilisant la méthode is\_text\_visible(). Si le texte est trouvé, il est mis en surbrillance avec self.highlight() (sélectionné comme un triple clique pour le mettre en bleu) et un message approprié est affiché dans le terminal en fonction du résultat.

En combinant ces étapes, le script est capable d'effectuer les actions requises sur la page web et de fournir un rapport dans le terminal sur la présence du bouton "Import des résultats" et du texte "Import des résultats" selon les cas.

# 21/07/2023

* Tâche : AUTO-38 - [e2e][NEONATLABO] Tester la redirection vers la bonne page
* Nouvelle demande : Travailler sur la génération de report.html pour avoir les rapports des tests
* En utilisant : self.test\_info = []

self.add\_test\_info

# 24/07/2023

* Tâche : AUTO-38 - [e2e][NEONATLABO] Tester la redirection vers la bonne page
* Tâche accomplie
* **Génération de rapports de test détaillés avec pytest-html** :

L'objectif de cette nouvelle demande est de générer des rapports de test détaillés en utilisant le plugin "pytest-html". Le scénario de test existant, implémenté dans le fichier "TEST.py", sera amélioré pour inclure les détails des rapports de test tels que les résultats (passés ou échoués) pour chaque étape du scénario.

Installation du plugin pytest-html :

Tout d'abord, nous devons nous assurer d'avoir installé le plugin "pytest-html". Cela peut être fait en exécutant la commande suivante dans le terminal :

pip install pytest-html

Mise à jour du script de test :

Le fichier "TEST.py" est modifié pour inclure l'option "--html=report.html" lors de l'exécution des tests avec pytest. Cette option permettra de générer un rapport HTML détaillé à la fin de l'exécution des tests.

Ajout des détails des rapports de test :

Le code du scénario de test est amélioré pour inclure la fonction "add\_test\_info(message, result)" qui permet d'ajouter des informations sur les étapes du test et leurs résultats (passés ou échoués) dans une liste appelée "test\_info". Cette liste stocke les détails des rapports de test au fur et à mesure de l'exécution du scénario.

Exécution des tests avec pytest-html :

Après avoir mis à jour le script de test, nous pouvons exécuter les tests en utilisant la commande suivante :

pytest -s NeonatLabo/TEST.py --html=report.html

Cette commande lancera les tests définis dans "TEST.py" et générera le rapport HTML détaillé dans le fichier "report.html".

Visualisation des détails des rapports :

Une fois les tests terminés, nous pouvons ouvrir le fichier "report.html" dans un navigateur pour visualiser les détails des rapports de test. Le rapport affiche chaque étape du test avec son résultat (passé ou échoué) et fournit une vue détaillée des résultats des tests.

Grâce à l'ajout du plugin "pytest-html" et à l'amélioration du script de test, nous sommes désormais en mesure de générer des rapports de test détaillés qui fournissent des informations complètes sur chaque étape du test et ses résultats. Cette amélioration permet une meilleure traçabilité et facilite l'analyse des résultats des tests, ce qui contribue à améliorer la qualité globale des tests automatisés.

# 25/07/2023

* Tâche : AUTO-38 - [e2e][NEONATLABO] Tester la redirection vers la bonne page
* Nouvelle demande : Afficher les noms des tests dans le rapport : “report.html”
* Je m’y prend en attribuant des Méthodes à chaque test

ex : test\_import\_results\_button, test\_login, test\_main, test\_title\_accueil\_HBS, test\_title\_import\_results

# 26/07/2023

* Tâche : AUTO-38 - [e2e][NEONATLABO] Tester la redirection vers la bonne page
* Nouvelle demande : Afficher les noms des tests dans le rapport : “report.html”
* Je m’y prend en attribuant des Méthodes à chaque test
* En cours

# 27/07/2023

* Tâche : AUTO-38 - [e2e][NEONATLABO] Tester la redirection vers la bonne page
* Nouvelle demande : Afficher les noms des tests dans le rapport : “report.html”

# 28/07/2023

* Tâche : AUTO-38 - [e2e][NEONATLABO] Tester la redirection vers la bonne page
* Nouvelle demande : Afficher les noms des tests dans le rapport : “report.html”
* Affichage des noms des tests dans le rapport : “report.html” réussie
* Tâche accomplie

# Liste des actions

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **N°** | **Action** | **Resp.** | **Date prévue** | **Statut** |
| **1** | Compléter la formation Git/GitHub | [Joshan RANLAUL](mailto:j.ranlaul@epiconcept.io)[Kim Ratomahenina](mailto:k.ratomahenina@epiconcept.io) | 12/06/2023 | Accomplie |
| **2** | Git & GitHub : création de Branch et Pull Request à faire | [v.ellayah@epiconcept.io](mailto:v.ellayah@epiconcept.io)  [Kim Ratomahenina](mailto:k.ratomahenina@epiconcept.io) | 20/06/2023 | Accomplie |
| **3** | Première tâche : AUTO-34 - Lancer le navigateur et accéder à l'URL de l'application | [Kim Ratomahenina](mailto:k.ratomahenina@epiconcept.io) | 27/06/2023 | Accomplie le 27/06/2023 |
| **4** | Tâche : AUTO-36 - Saisir les login/MDP et se connecter à l'application | [Kim Ratomahenina](mailto:k.ratomahenina@epiconcept.io) | 04/07/2023 | Accomplie le 04/07/2023 |
| **5** | Tâche : AUTO-37 - Implémenter le test de connexion sur une autre application | [Kim Ratomahenina](mailto:k.ratomahenina@epiconcept.io) | 11/07/2023 | Accomplie le 10/07/2023 |
| **6** | Tâche : AUTO-38 - [e2e][NEONATLABO] Tester la redirection vers la bonne page | [Kim Ratomahenina](mailto:k.ratomahenina@epiconcept.io) | 31/07/2023 | Accomplie le 28/07/2023 |

Étape 1 : Installation de Git

Tout d'abord, assurez-vous d'avoir Git installé sur votre ordinateur. Vous pouvez le télécharger et l'installer depuis le site officiel de Git (https://git-scm.com/).

Étape 2 : Création d'un compte GitHub

Si vous n'avez pas encore de compte GitHub, rendez-vous sur le site de GitHub (https://github.com/) et créez un compte gratuitement. Connectez-vous ensuite à votre compte.

Étape 3 : Création d'un dépôt (repository) sur GitHub

Un dépôt (repository) est l'endroit où les fichiers et les codes sont stockés sur GitHub. Pour créer un nouveau dépôt, cliquez sur le bouton "New" ou "Nouveau" sur la page principale de votre compte GitHub. Donnez un nom à votre dépôt et choisissez les options appropriées, puis cliquez sur "Create Repository" ou "Créer un dépôt".

Étape 4 : Clonage du dépôt

Le clonage du dépôt signifie que vous allez créer une copie locale du dépôt sur votre ordinateur. Pour cela, ouvrez un terminal (ou une invite de commande) sur votre ordinateur et utilisez la commande suivante :

bash :

git clone <URL du dépôt>

L'URL du dépôt est disponible sur la page principale du dépôt sur GitHub. Copiez l'URL et collez-la après la commande "git clone". Appuyez sur Entrée pour lancer le clonage du dépôt.

Étape 5 : Récupération des fichiers et des codes

Après avoir cloné le dépôt, vous disposez maintenant d'une copie locale de tous les fichiers et codes du dépôt sur votre ordinateur. Vous pouvez accéder à ces fichiers à partir du dossier créé lors du clonage.

Utilisez un éditeur de code de votre choix pour ouvrir et modifier les fichiers selon vos besoins.

Étape 6 : Mise à jour des fichiers

Si vous effectuez des modifications sur les fichiers locaux et que vous souhaitez les envoyer sur GitHub, vous devez les ajouter, les valider et les pousser vers le dépôt.

Ajouter les fichiers modifiés à la zone de préparation (staging) en utilisant la commande :

bash :

git add <nom du fichier modifié>

ou pour ajouter tous les fichiers modifiés :

bash :

git add .

Valider les modifications avec un message de commit en utilisant la commande :

bash :

git commit -m "votre message de commit"

Pousser les modifications vers le dépôt distant (GitHub) en utilisant la commande :

bash :

git push origin <nom de la branche>

ou si vous utilisez la branche principale (master) :

bash :

git push origin master

Cela mettra à jour les fichiers sur le dépôt GitHub.

**Pour créer une nouvelle branche dans Git**, vous pouvez suivre les étapes suivantes :

Assurez-vous d'être dans le répertoire de travail de votre projet Git. Vous pouvez utiliser la commande cd dans le terminal pour vous déplacer vers le répertoire approprié.

Vérifiez que vous êtes sur la branche principale (généralement appelée master ou main) en utilisant la commande suivante :

git branch

La branche actuelle sera mise en évidence avec une astérisque (\*).

Pour créer une nouvelle branche, utilisez la commande git branch suivie du nom de la nouvelle branche :

git branch <nom-de-la-branche>

Par exemple, si vous voulez créer une branche appelée "feature/nouvelle-fonctionnalite", vous pouvez utiliser la commande :

git branch feature/nouvelle-fonctionnalite

Pour basculer sur la nouvelle branche que vous venez de créer, utilisez la commande git checkout :

git checkout <nom-de-la-branche>

Par exemple :

git checkout feature/nouvelle-fonctionnalite

Vous serez maintenant sur la nouvelle branche et vous pourrez commencer à y apporter des modifications.

Si vous souhaitez combiner les deux étapes précédentes en une seule, vous pouvez utiliser la commande git checkout avec l'option -b pour créer et basculer sur une nouvelle branche en une seule commande :

git checkout -b <nom-de-la-branche>

Par exemple :

git checkout -b feature/nouvelle-fonctionnalite

**Pour créer une pull request dans Git**, suivez les étapes suivantes :

Assurez-vous que vous avez terminé votre travail sur une branche spécifique et que vous avez effectué un commit des modifications locales.

Accédez à la page principale du référentiel (repository) sur GitHub où vous souhaitez créer la pull request.

Cliquez sur le bouton "New pull request" (Nouvelle pull request). Il est généralement situé près de la partie supérieure de la page, à côté des onglets.

Sur la page "Compare changes" (Comparer les modifications), sélectionnez la branche source (la branche contenant vos modifications) et la branche cible (la branche dans laquelle vous souhaitez fusionner vos modifications). Assurez-vous de choisir les branches correctes en utilisant les menus déroulants.

Une fois que vous avez sélectionné les branches, GitHub affichera les différences entre les deux branches pour vous permettre de les examiner.

Donnez un titre significatif à votre pull request dans le champ approprié. Vous pouvez également fournir une description détaillée de vos modifications dans le champ de description.

Si vous le souhaitez, vous pouvez ajouter des commentaires supplémentaires ou des captures d'écran en utilisant les fonctionnalités disponibles sur la page de la pull request.

Lorsque vous êtes prêt à soumettre votre pull request, cliquez sur le bouton "Create pull request" (Créer une pull request).

Votre pull request sera maintenant visible sur la page du référentiel. D'autres collaborateurs peuvent examiner vos modifications, commenter, suggérer des modifications supplémentaires ou approuver la pull request.

Continuez à suivre les commentaires et les discussions sur votre pull request. Si des modifications supplémentaires sont nécessaires, vous pouvez continuer à les apporter sur la branche source, et elles seront automatiquement ajoutées à la pull request.

Une fois que votre pull request est approuvé et que vous êtes prêt à fusionner les modifications, vous pouvez cliquer sur le bouton "Merge pull request" (Fusionner la pull request) pour intégrer les modifications dans la branche cible.



# Tests Automatisés

<https://docs.pytest.org/en/7.1.x/getting-started.html#install-pytest>

<https://pypi.org/project/selenium-base/>

## 1.1. Communication au sein de l’équipe projet

La communication au sein de l'équipe d'Epiconcept Océan Indien était essentielle pour assurer la cohésion et le bon déroulement du projet. Étant donné que les membres de l'équipe étaient physiquement présents dans le même établissement, nous avons favorisé la communication verbale comme moyen privilégié d'échange d'informations. La proximité géographique a facilité les discussions informelles, les réunions spontanées et les échanges rapides pour résoudre les problèmes et prendre des décisions.

En complément de la communication verbale, nous avons utilisé divers outils de communication en ligne pour faciliter la collaboration et le partage d'informations au sein de l'équipe. L'un des principaux outils utilisés était Google Suite, qui comprend des applications telles que Google Docs, Google Sheets et Google Slides. Ces outils nous ont permis de travailler de manière collaborative sur des documents, des feuilles de calcul et des présentations en temps réel, favorisant ainsi la transparence et la productivité au sein de l'équipe.

De plus, nous avons utilisé Google Chat pour les discussions instantanées et les échanges rapides. Cela nous a permis de communiquer rapidement et efficacement, même lorsque les membres de l'équipe étaient occupés à d'autres tâches ou n'étaient pas physiquement présents.

## 1.2. Communication hors équipe projet

Le projet chez Epiconcept Océan Indien nécessitait également des communications avec des personnes en dehors de l'équipe projet, notamment des responsables basés en France. Pour faciliter ces échanges internationaux, nous avons utilisé des outils de communication en ligne tels que Google Meet. Cela nous a permis d'organiser des réunions virtuelles pour discuter de sujets spécifiques, partager des informations et collaborer à distance.

Par exemple, j'ai eu une formation sur l'utilisation de Passbolt avec Bryan, consultant en support et maintenance basé en France. Cette formation s'est déroulée via une réunion en ligne, où nous avons pu échanger des informations et des connaissances essentielles pour la mise en œuvre réussie de Passbolt dans le projet.

De même, j'ai participé à une réunion d'initiation aux tests automatisés avec Mariem HAMROUNI, qui est également basée en France. Nous avons utilisé Google Meet pour échanger des idées, discuter des meilleures pratiques en matière de tests automatisés et explorer les outils disponibles pour améliorer la qualité du développement de l'application.

## 1.3. Ressources fournies et/ou à utiliser

Pendant mon stage chez Epiconcept Océan Indien, j'ai utilisé mon propre laptop personnel pour effectuer les tâches et le développement de l'application. Cela m'a permis de bénéficier d'un environnement de travail familier et de disposer des outils et des logiciels nécessaires pour mener à bien mes missions.

En termes de ressources fournies par l'entreprise, j'ai eu accès au réseau interne, ce qui m'a permis de me connecter aux dossiers contenant les informations du service approvisionnement. De plus, j'ai utilisé des applications et des outils tels que Microsoft Access pour travailler sur l'application existante.

## 5.6. Les outils

La société Epiconcept Océan Indien utilise une variété d'outils pour soutenir ses activités et optimiser ses opérations. Voici une liste des principaux outils utilisés par l'entreprise :

Google Suite : Epiconcept Océan Indien utilise la suite d'outils G Suite de Google, comprenant Gmail, Google Drive, Google Docs, Google Sheets, Google Slides, Google Calendar, etc. Ces outils permettent la collaboration en temps réel, le partage de documents et la gestion des calendriers.

Jira : Jira est un outil de gestion de projet largement utilisé par Epiconcept Océan Indien. Il permet de suivre les tâches, les problèmes, les demandes de fonctionnalités et les rapports de bogues, facilitant ainsi la coordination et la gestion des projets.

Voozanoo : Voozanoo est une plateforme de développement d'applications web utilisée par Epiconcept Océan Indien. Elle offre des fonctionnalités de développement rapide d'applications (RAD) et permet de créer des applications personnalisées selon les besoins spécifiques de l'entreprise.

Passbolt : Passbolt est un gestionnaire de mots de passe open source utilisé par Epiconcept Océan Indien. Il permet de stocker en toute sécurité les mots de passe et les informations d'identification, facilitant ainsi la gestion et le partage des informations sensibles.

Git et GitHub : Epiconcept Océan Indien utilise Git comme système de contrôle de version pour la gestion du code source de ses projets. GitHub est utilisé comme plateforme de dépôt Git, facilitant la collaboration entre les développeurs et le suivi des modifications du code.

VSCode : VSCode (Visual Studio Code) est un éditeur de code source léger et puissant utilisé par Epiconcept Océan Indien. Il offre des fonctionnalités avancées d'édition, de débogage et d'intégration avec d'autres outils, ce qui en fait un choix populaire parmi les développeurs.

MySQL : MySQL est un système de gestion de base de données relationnelle utilisé par Epiconcept Océan Indien pour stocker et gérer les données des applications et des projets.

Selenium : Selenium est un ensemble d'outils d'automatisation des tests utilisé par Epiconcept Océan Indien pour tester les applications web. Il permet de créer et d'exécuter des scripts de tests automatisés, garantissant ainsi la qualité et la fiabilité des logiciels développés.

Seleniumbase : Seleniumbase est une extension de Selenium qui offre des fonctionnalités supplémentaires pour les tests automatisés. Epiconcept Océan Indien utilise cet outil pour renforcer ses capacités de test et améliorer l'efficacité des tests automatisés.

## 5.7. Méthodes de travail en projet collaboratif

La méthode Agile Scrum est une approche de gestion de projet adoptée par la société Epiconcept Océan Indien pour la réalisation de ses projets de développement de logiciels. Cette méthodologie favorise la collaboration, la flexibilité et la livraison itérative des fonctionnalités, ce qui permet à l'équipe de s'adapter efficacement aux besoins changeants du projet

Scrum repose sur des principes clés qui sont mis en pratique chez Epiconcept Océan Indien. Voici un aperçu détaillé des éléments fondamentaux de la méthode Agile Scrum tels qu'ils sont appliqués au sein de l'entreprise :

Équipe Scrum : L'équipe est auto-organisée et interfonctionnelle, rassemblant des membres ayant des compétences complémentaires nécessaires à la réalisation du projet. Chaque membre de l'équipe est responsable de la planification, de la réalisation et de la livraison des fonctionnalités. Cette approche favorise la collaboration et permet une répartition équilibrée des responsabilités au sein de l'équipe.

Product Owner : Le rôle du Product Owner est crucial dans la méthodologie Scrum. Il représente les intérêts des parties prenantes et définit les exigences du projet. Le Product Owner est responsable de la création et de la gestion du backlog des fonctionnalités à développer, en priorisant les éléments en fonction de la valeur métier. Il joue un rôle clé dans la définition des objectifs et la planification des Sprints.

Sprints : Les Sprints sont des itérations de travail d'une durée fixe. Chez Epiconcept Océan Indien, les Sprints commencent et se renouvellent chaque mardi. Chaque Sprint a une durée typique de deux semaines, permettant à l'équipe de se concentrer sur un ensemble spécifique de fonctionnalités à réaliser et de s'engager à les livrer à la fin du Sprint. Cette approche favorise une gestion efficace du temps et des ressources.

Planification du Sprint : Au début de chaque Sprint, l'équipe tient une réunion de planification pour sélectionner les éléments du backlog à réaliser pendant le Sprint. Les fonctionnalités sont décomposées en tâches plus petites et des estimations de temps sont faites pour chaque tâche. Cette étape permet de définir clairement les objectifs du Sprint et de répartir les responsabilités au sein de l'équipe.

Daily Scrum : Chaque jour pendant le Sprint, l'équipe se réunit lors d'une réunion quotidienne de courte durée, appelée Daily Scrum. L'objectif de cette réunion est de partager les progrès réalisés, discuter des défis rencontrés et coordonner les activités à venir. Cela favorise la transparence et la communication au sein de l'équipe, permettant ainsi de suivre l'avancement du projet de manière efficace.

Revue de Sprint : À la fin de chaque Sprint, l'équipe organise une revue de Sprint pour présenter les fonctionnalités développées au Product Owner et aux parties prenantes. Cette revue permet de recueillir des commentaires, d'identifier les ajustements nécessaires et de valider la progression du projet. Les discussions et les retours d'expérience recueillis lors de cette revue contribuent à l'amélioration continue du processus de développement.

Rétrospective de Sprint : Après la revue de Sprint, l'équipe tient une rétrospective pour évaluer les performances du Sprint précédent et identifier des opportunités d'amélioration. Cette étape permet à l'équipe de réfléchir aux succès et aux défis rencontrés, d'identifier les bonnes pratiques à maintenir et d'explorer des pistes d'amélioration pour les Sprints futurs. Cette démarche d'apprentissage continu permet à l'équipe de s'améliorer constamment et d'optimiser ses performances.

Chez Epiconcept Océan Indien, les Sprints suivent un calendrier spécifique. Les Sprints commencent et se renouvellent chaque mardi, avec une durée de deux semaines. Lorsque j'ai rejoint Epiconcept le 5 juin, nous étions en plein Sprint K (les noms des sprint sont définis alphabétiquement), qui avait commencé le 30 mai. Pendant mon stage, je travaillerai jusqu'au 31 juillet, en plein Sprint O. Cette période de travail au sein des Sprints m'a permis de m'intégrer pleinement à l'équipe et de participer activement au développement des fonctionnalités assignées.

En travaillant selon la méthode Agile Scrum, j'ai pu contribuer au développement des fonctionnalités assignées, participer aux réunions quotidiennes et aux revues de Sprint, et suivre le flux de travail itératif de l'équipe. Cette approche m'a permis de développer des compétences essentielles en matière de collaboration, de gestion du temps et de résolution de problèmes, tout en favorisant une communication transparente et une adaptation efficace aux changements.

La méthode Agile Scrum adoptée par Epiconcept Océan Indien favorise une approche collaborative et adaptative dans la réalisation des projets. Elle permet à l'équipe de s'ajuster aux changements, de fournir des résultats de haute qualité de manière itérative et d'assurer la satisfaction des parties prenantes tout au long du processus de développement. Cette méthodologie a démontré son efficacité dans la gestion des projets chez Epiconcept Océan Indien et contribue au succès continu de l'entreprise.

## Analyse des tâches continues effectuées lors de mon stage chez Epiconcept Océan Indien :

#### · Tâche AUTO-34 - Lancer le navigateur et accéder à l'URL de l'application :

La première tâche qui m'a été attribuée lors de mon stage chez Epiconcept Océan Indien consistait à développer la fonctionnalité permettant de lancer le navigateur et accéder à l'URL de l'application. Pour réaliser cette tâche, j'ai utilisé la bibliothèque Selenium, qui est un outil de test automatisé pour les applications web, ainsi que le module webdriver de Selenium, qui fournit des fonctionnalités pour interagir avec les navigateurs web.

Dans un premier temps, j'ai configuré le navigateur Chrome en utilisant la classe webdriver.Chrome(). Cette classe permet d'instancier un objet représentant le navigateur Chrome, ce qui m'a permis de contrôler le navigateur de manière automatisée.

Ensuite, j'ai utilisé une variable url provenant du module vars pour spécifier l'URL de l'application à laquelle je devais accéder. En appelant la méthode get() sur l'objet du navigateur, j'ai pu ouvrir le navigateur et charger l'URL spécifiée. Cela m'a permis d'accéder à l'application dans le navigateur et d'initialiser l'environnement de test.

Une difficulté que j'ai rencontrée lors de cette tâche était liée à l'installation de pytest, une bibliothèque de test pour Python. Cependant, après avoir résolu ce problème, j'ai pu terminer avec succès cette tâche. J'ai également pu maintenir le navigateur ouvert en utilisant une boucle while True, ce qui m'a permis d'exécuter des tâches ultérieures sans fermer le navigateur à chaque fois.

Grâce à cette première tâche, j'ai pu acquérir une expérience pratique de l'utilisation de Selenium et du module webdriver pour automatiser le lancement du navigateur et l'accès à l'URL de l'application. Cela m'a également permis de mieux comprendre les défis liés aux tests automatisés et à la configuration des environnements de test.

#### · Tâche AUTO-36 - Saisir les login/MDP et se connecter à l'application :

La deuxième tâche que j'ai réalisée était de développer la fonctionnalité permettant de saisir les identifiants de connexion (login et mot de passe) et de se connecter à l'application automatiquement. Cette tâche était cruciale car elle permettait d'automatiser la phase de connexion, ce qui est essentiel pour réaliser des tests automatisés de manière efficace.

Pour accomplir cette tâche, j'ai utilisé la bibliothèque SeleniumBase, qui est une extension de Selenium permettant d'effectuer des actions de test plus facilement et avec un code plus concis. J'ai également utilisé le fichier vars.py pour stocker les identifiants de connexion en tant que variables.

Dans un premier temps, j'ai utilisé la méthode open de SeleniumBase pour accéder à l'URL de l'application. Cette méthode simplifie l'ouverture de l'URL en remplaçant plusieurs lignes de code par une seule instruction, ce qui rend le code plus clair et plus facile à lire.

Ensuite, j'ai utilisé les méthodes type et press\_enter fournies par SeleniumBase pour remplir automatiquement les champs de connexion avec les identifiants fournis. La méthode type m'a permis de saisir rapidement le nom d'utilisateur et le mot de passe en utilisant une seule instruction, ce qui a considérablement réduit la quantité de code nécessaire.

Une difficulté que j'ai rencontrée lors de cette tâche était de trouver un moyen de spécifier les identifiants de connexion (username et password) lors de l'exécution du script de test. Après quelques essais, j'ai réalisé que je pouvais définir les variables d'environnement USERNAME et PASSWORD avant d'exécuter le script de test. Cela m'a permis de passer les identifiants en tant que variables d'environnement au lieu de les spécifier directement dans la commande de lancement du script.

En réalisant cette tâche, j'ai pu développer une approche plus modulaire et réutilisable pour la saisie des identifiants de connexion, en utilisant les fonctionnalités fournies par SeleniumBase. Cela m'a également permis de mieux comprendre l'importance des variables d'environnement pour la configuration des tests automatisés.

#### · Tâche AUTO-37 - Implémenter le test de connexion sur une autre application :

La troisième tâche que j'ai entreprise était de développer le test de connexion pour une autre application, en l'occurrence "NeonatLabo". L'objectif était d'automatiser une série d'actions sur la page web de cette application et de vérifier la présence d'éléments spécifiques.

Pour réaliser cette tâche, j'ai créé un fichier login\_logic.py dans le répertoire "Logique" pour mutualiser la logique de connexion. Dans ce fichier, j'ai défini une classe LoginLogic qui contenait la méthode def\_login(), responsable de la logique de connexion. Cette méthode utilisait les variables url, username et password importées depuis le fichier vars.py spécifique à chaque application.

J'ai ensuite importé la classe LoginLogic du fichier login\_logic.py dans les fichiers TEST.py de chaque application, SboxIo et NeonatLabo. En créant une instance de cette classe, j'ai pu appeler la méthode def\_login() pour effectuer la connexion spécifique à chaque application.

Une difficulté que j'ai rencontrée lors de cette tâche était de gérer les spécificités de chaque application, notamment les éléments de l'interface utilisateur et les comportements attendus. Cependant, en utilisant la programmation orientée objet (OOP) de Python, j'ai pu encapsuler la logique de connexion dans une classe réutilisable et créer des instances de cette classe dans chaque fichier TEST.py. Cela a permis d'organiser et de structurer le code de manière modulaire, facilitant ainsi les mises à jour et les modifications futures.

Grâce à cette tâche, j'ai pu approfondir mes connaissances en programmation orientée objet et en gestion de la logique de test de manière réutilisable. J'ai également acquis une meilleure compréhension des différences entre les applications et de l'importance de gérer les spécificités de chacune dans les tests automatisés.

#### · Tâche AUTO-38 - [e2e] [NEONATLABO] Tester la redirection vers la bonne page :

La quatrième tâche que j'ai réalisée était de tester la redirection vers la bonne page sur l'application NeonatLabo. L'objectif de cette tâche était d'automatiser une série d'actions sur une page web et de vérifier la présence d'éléments spécifiques.

Pour accomplir cette tâche, j'ai effectué des essais pour automatiser le clic automatique sur un bouton spécifique ("Accueil" et "Import de résultats") et pour vérifier la présence d'un texte spécifique ("Import des résultats") sur la nouvelle page qui s'affiche après le clic.

J'ai rencontré des difficultés lors de cette tâche, notamment dans la gestion des codes sources du bouton "Import de résultats" sur l'application NeonatLabo. Cependant, en utilisant les fonctionnalités de la bibliothèque SeleniumBase, j'ai réussi à vérifier la visibilité du bouton et à cliquer automatiquement dessus si nécessaire. J'ai également pu vérifier la présence du texte "Import des résultats" sur la nouvelle page et afficher un rapport approprié dans le terminal en fonction du résultat.

Une amélioration que j'ai apportée lors de cette tâche était l'utilisation du plugin "pytest-html" pour générer des rapports de test détaillés. Cela m'a permis d'inclure les détails des rapports de test tels que les résultats (passés ou échoués) pour chaque étape du scénario, facilitant ainsi l'analyse des résultats des tests.

Grâce à cette tâche, j'ai pu approfondir mes compétences en automatisation de tests web en utilisant SeleniumBase et en générant des rapports de test détaillés avec pytest-html. J'ai également amélioré ma capacité à résoudre des problèmes liés à la gestion des éléments de l'interface utilisateur sur des applications web complexes.

Les tâches continues que j'ai réalisées lors de mon stage chez Epiconcept Océan Indien m'ont permis d'acquérir une expérience précieuse en matière de tests d'automatisation pour les applications web. J'ai pu développer des compétences pratiques en utilisant des bibliothèques telles que Selenium et SeleniumBase, ainsi qu'en manipulant les environnements de test et les identifiants de connexion. J'ai également renforcé ma compréhension de la programmation orientée objet (OOP) et de la structuration du code pour rendre les tests plus modulaires et réutilisables.

Ces tâches m'ont également confronté à des défis techniques, tels que la gestion des spécificités de chaque application et la résolution de problèmes liés aux éléments de l'interface utilisateur. Cependant, en faisant preuve de détermination et en utilisant les ressources disponibles, j'ai pu surmonter ces difficultés et atteindre les objectifs fixés.

Suite à la demande d'affichage des noms des tests dans les rapports de tests dans report.html, j'ai ajouté des noms de méthode explicites pour chaque test dans le fichier TEST.py. Cela permettra de rendre les rapports de test plus clairs et informatifs, en indiquant le nom de chaque test exécuté.

Voici la mise à jour des codes avec les noms de méthode pour chaque test :

import sys

import os

sys.path.insert(0, os.path.abspath(os.path.join(os.path.dirname(\_\_file\_\_), "..")))

import pytest

from seleniumbase import BaseCase

from base.login import Login as NeonatLogin

class NeonatLaboTestClass(BaseCase):

def \_\_init\_\_(self, \*args, \*\*kwargs):

super(NeonatLaboTestClass, self).\_\_init\_\_(\*args, \*\*kwargs)

self.test\_info = [] # Initialisation ici pour stocker les détails du test

def test\_main(self):

self.test\_login()

self.test\_title\_accueil\_HBS()

self.test\_import\_results\_button()

self.test\_title\_import\_results()

def test\_login(self):

login = NeonatLogin(self)

login.login()

self.add\_test\_info("Saisie et entré du login", "passed")

self.sleep(15)

def test\_title\_accueil\_HBS(self):

if self.is\_text\_visible("Accueil"):

self.select\_option\_by\_text("select.form-control", "HBS")

self.add\_test\_info("Vérification du titre 'Accueil' et sélection de 'HBS'", "passed")

self.sleep(3)

def test\_import\_results\_button(self):

if self.is\_element\_visible("a[href='result/import']"):

self.click("a[href='result/import']")

self.add\_test\_info("Vérification du bouton 'Import des résultats'", "passed")

def test\_title\_import\_results(self):

# Vérifier si le titre 'Import des résultats' est présent sur la nouvelle page

if self.is\_text\_visible("Import des résultats"):

self.add\_test\_info("Vérification du titre 'Import des résultats' sur la nouvelle page", "passed")

def add\_test\_info(self, message, result):

self.test\_info.append((message, result))

self.sleep(3)

if \_\_name\_\_ == "\_\_main\_\_":

# Lancer les tests avec pytest-html et générer le rapport HTML

pytest.main(["-s", "NeonatLabo/TEST.py", "--html=report.html"])

# Remerciements

Je tiens tout d'abord à exprimer ma profonde gratitude envers l'équipe d'Epiconcept pour m'avoir offert l'opportunité de réaliser mon stage au sein de leur entreprise. Je suis reconnaissant envers toutes les personnes qui m'ont accueilli, soutenu et guidé tout au long de cette expérience enrichissante.

Mes remerciements vont tout particulièrement à mon tuteur, Vanina Ellayah, Team Manager. Son expertise, son soutien et sa bienveillance ont été d'une valeur inestimable tout au long de mon stage. Ses conseils éclairés, sa capacité à résoudre les problèmes et sa passion pour son travail ont contribué à rendre mon expérience de stage des plus gratifiantes.

Je tiens également à remercier l'ensemble de l'équipe d'Epiconcept pour leur bienveillance, leur collaboration et leur esprit d'équipe. Leur accueil chaleureux et leur partage de connaissances ont grandement facilité mon intégration au sein de l'entreprise.

Enfin, je n'oublie pas de remercier ma famille, mes amis, la MCCI Business SChool et toutes les personnes qui m'ont soutenu tout au long de ce stage. Leur encouragement et leur soutien indéfectible ont été des moteurs essentiels dans la réalisation de cette expérience.

Ce stage chez Epiconcept a été une véritable opportunité d'apprentissage et de développement professionnel, et je suis reconnaissant envers toutes les personnes qui ont contribué à sa réussite. Leur confiance en moi et leur soutien ont été des éléments clés dans mon parcours.

Je souhaite à l'entreprise Epiconcept une continuité de succès dans ses activités et je suis honoré d'avoir pu faire partie de cette organisation exceptionnelle.