

ESTEBAN LE CALVEZ

Rapport de Stage

Alternance Qualité & Intégration

Postulant au titre de Concepteur Développeur Informatique

Réalisé par

LE CALVEZ Estéban

Sommaire

[Remerciements 4](#_Toc517092203)

[Introduction 5](#_Toc517092204)

[Présentation de l’entreprise 6](#_Toc517092205)

[Présentation de l’entreprise 6](#_Toc517092206)

[Organigramme 6](#_Toc517092207)

[Technologies utilisées 8](#_Toc517092208)

[Projet 13](#_Toc517092209)

[Analyse de l’existant 13](#_Toc517092210)

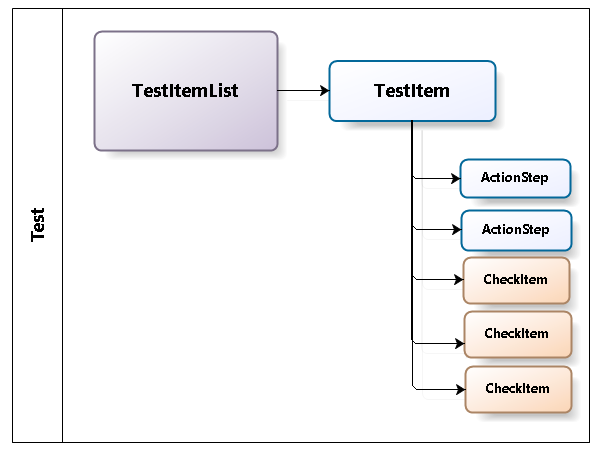
[Objectif du projet 13](#_Toc517092211)

[Fonctionnalités de Selenium 14](#_Toc517092212)

[Utilisation de Selenium chez Kurmi 15](#_Toc517092213)

[Pattern adopté 15](#_Toc517092214)

[Les tests avec Selenium 17](#_Toc517092215)

[ 17](#_Toc517092216)

# Remerciements

Je tiens à remercier toute l’équipe Rennes de **Kurmi Software** de m’avoir si bien accueilli dans leurs locaux, de m’avoir apporté leur aide quand j’en avais besoin et d’avoir toujours apporté une bonne ambiance et une envie d’aller travailler.

Je remercie **Gilles Nouais** de m’avoir recruté et aidé dans mon projet.

Je remercie mes **collègues testeurs** pour m’avoir apporté des réponses et des solutions à mes problèmes.

Je remercie l’équipe développeurs, particulièrement **Sébastien Deprez** pour m’avoir appris à utiliser Selenium et m’avoir aidé quand j’avais des questions.

# Introduction

J’ai rejoint **Kurmi Software**, société de communications unifiées basée à Cesson Sévigné, le vendredi 05 janvier 2018 en tant que Testeur logiciel/Intégrateur de tests automatisés.

Je m’occupais principalement de développer des tests automatisés qui étais déployés à chaque nouvelle version afin de tester les composants d’interface de la suite de produits Kurmi.

Au cours des missions effectués, j’ai dû apprendre à communiquer, poser des questions quand je n’y arrivais pas. Apprendre à travailler efficacement pour effectuer le travail demandé dans les temps. Et surtout apprendre un métier pour pouvoir comprendre l’application. J’ai aussi dû effectuer des recherches sur internet et plus particulièrement sur **StackOverFlow** qui est un (sinon le plus complet) forum de développement.

# Présentation de l’entreprise

## Présentation de l’entreprise

Kurmi-Software implanté depuis 2011 en tant qu’éditeur de logiciel spécialisé dans les communications unifiées est une filiale de Niji. L’entreprise est indépendante et autonome vis-à-vis de Niji.

L’entreprise compte à ce jour environ 35 salariés répartis sur 2 pôles et sites distincts. La première équipe est celle des commerciaux basés sur Paris et ils ont pour but de trouver de nouveaux clients pour Kurmi. Le second pôle situé à Cesson-Sévigné est dédié à la Recherche et Développement de la suite logiciel Kurmi.

## Organigramme

Ce pôle est constitué de plusieurs équipes.

L’équipe R&D (Recherche & Développement) développant les nouvelles fonctionnalités demandées par les clients ou suggérés par une des équipe et trouvé utile par les autres équipes,

l’équipe QA (Quality Assurance) dont je fais partie, va se charger de tester les nouvelles fonctionnalités en écrivant au préalable des cahiers de test et l’exécuter afin de voir s’il contient des bugs, auquel cas il ouvrira BugZilla pour créer un bug en indiquant toutes les informations nécessaires à l’équipe développement.

L’équipe PS (Professional Services) quant à elle, s’occupe de l’intégration de Kurmi chez le client par l’installation mais également la configuration. Chaque client a des demandes différentes et c’est à l’équipe PS d’en répondre en faisant une configuration adaptée.

L’équipe Support va s’occuper des problèmes techniques rencontré chez le client et répondre à leurs questions. Ils fonctionnent avec des tickets au préalable ouvert par le client, relatant le/les problèmes leur faisant face (dysfonctionnement, bloquant, etc.). A la suite de cela, l’équipe Support va dans le cas d’un bug le remonter à l’équipe R&D, pour dans la suite apporter une correction. Dans les autres cas, ils sont dans la capacité d’aider le client afin de résoudre le problème, car ils sont avant tout des experts en téléphonie et connaissent très bien le produit Kurmi et les configurations du client.

De ce qui est du management, il y a les 2 chefs de projets en contact avec les entreprises clientes pour connaitre les besoins et les attentes vis-à-vis de la suite logiciel Kurmi. Ainsi la tâche est attribuée à une personne du Professional Service ou du Support selon la demande. Chaque fin de semaine un mail est envoyé à tout le monde pour informer l’avancer des différents projets chez les entreprises clientes, les projets finis et ceux qui sont en attente. Différentes réunions sont organisées au sein des équipe pour parler des problèmes rencontrés et de clarifier certains points sur qui a fait quoi et comment. Frédéric Briand et Ludovic Pollet chapotent respectivement les équipes Support/PS et R&D.

Et il y a aussi toutes les 2 semaines (quand tout le monde est présent si possible) une réunion permettant à chacun d’expliquer ce sur quoi il/elle a avancé et d’ouvrir des suggestions pour la suite.

# Technologies utilisées

#### 

#### Eclipse IDE

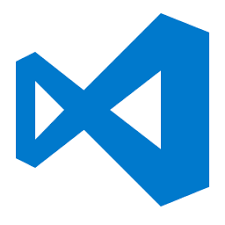
Eclipse est un projet, décliné et organisé en un ensemble de sous-projets de développements logiciels, de la fondation Eclipse visant à développer un environnement de production de logiciels libre qui soit extensible, universel et polyvalent, en s'appuyant principalement sur Java.



#### Win SCP

WinSCP est un client SFTP graphique pour Windows. Il utilise SSH et est open source. Le protocole SCP est également supporté. Le but de ce programme est de permettre la copie sécurisée de fichiers entre un ordinateur local et un ordinateur distant.

Dans mon cas : j’ai surtout utilisé Win SCP pour modifier les fichiers de clés de traduction et obtenir tous les logs quand je créais un bug.



#### Visual Studio Code

Visual Studio Code est un éditeur de code cross-platform open source gratuit.

Il supporte de base une dizaine de langages mais possède des plugins pouvant en supporter plus.

Je l’ai utilisé pour développer des tests en javascript (ce sont ces tests qui étaient sur le serveur et exécutés à chaque sortie de version).



#### SourceTree

SourceTree, logiciel développé par le groupe Atlassian propose une interface graphique pour git et Mercurial desktop. Il permet de faire toutes les commandes de git en ayant une interface graphique.



#### Microsoft Teams

Microsoft teams est une plateforme qui contient différents espaces. Un espace de chat sur lequel nous pouvons discuter avec nos collègues, il donne la possibilité de télécharger des fichiers, des notes et des réunions.

Nous avons plusieurs « channels » pour différentes équipes, dans différents buts.



#### BugZilla

Bugzilla est une plateforme de déclaration de bugs sur une application ainsi que de communication autour de ce bug.

Nous déclarons une anomalie sur l’application, et discutons au sujet de cette dernière. Nous donnons les logs, l’environnement sur lequel ce bug a été trouvé, des images ou gif indiquant où se trouve le bug et une description de ce dernier.



#### Trello

Trello est un projet web basé sur la gestion d’applications.

Il répertorie différents tableaux sur lequel des cartes sont disposés afin de savoir qui travaille sur quoi.

Par exemple pour une équipe de développement il y aura plusieurs tableaux :

- A développer

- En cours de développement

- Développement fini

- (quand on trouve un bug) A corriger

- Correction terminée

#### 

#### Squash

Squash est une plateforme web de test permettant de déclarer :

-des exigences : on exige que tel partie de l’appli soit testée à tel endroit

-des cahiers de tests : un ensemble de pas de test à réaliser pour se dire qu’une exigence est réalisée.

- Des pas de tests : une ou plusieurs actions utilisateurs à faire pour parvenir à un résultat rapidement : Remplir les champs login et mot de passe et cliquer sur le bouton de connexion > Résultat : vous êtes connectés/vous avez un message d’erreur

#### 

#### Selenium

Selenium est un framework de test informatique développé en Java. Il permet d’automatiser des tests sur une application web en lançant un navigateur et en exécutant des tests qu’on lui ordonne d’exécuter.

On lui demande si le résultat qu’on attends est égal au résultat obtenu et on envoi tout ça dans les logs.

# Projet

## Analyse de l’existant

La mise en place de l’automatisation des tests au niveau graphique de l’application à l’aide de Selenium et Java, a été mis en place par Sébastien Deprez et ??? quelques temps avant que j’arrive.

Ils avaient mis en place un driver Selenium pour une version de Mozilla afin d’exécuter les tests automatiques sur l’interface de Kurmi.

## Objectif du projet

L’objectif du projet est la reprise et l’amélioration de l’existant.

En effet le code qu’ils avaient commencé avant que j’arrive était historique, les bonnes manières avaient été réécrite et il fallait continuer de développer des tests automatisés.

Les objectifs étaient de :

- Augmenter la couverture de code

- Pouvoir s’assurer rapidement de la non régression de fonctionnalités.

- Etre plus serein lors d’une sortie de version sur des éléments qu’ils n’auraient pas eu le temps de tester à la main

C’est pourquoi ils m’ont engagé pour développer des tests avec le framework Selenium afin d’augmenter la couverture de code.

## Fonctionnalités de Selenium

Selenium est un framework que nous avons mis en place et utilisé avec Java.

Il propose des fonctionnalités qu’un utilisateur pourrait pratiquer sur l’interface graphique d’une application afin de s’assurer du bon fonctionnement de ces dernières.

Dans mon utilisation de Selenium, il y avait aussi des méthodes créées par mes collègues auparavant. Voici une liste non exhaustive:

- Des méthodes cherchant des éléments de la page (dans le DOM) par différentes manières :

- css selector/id/ name /class (que j’ai le plus utilisé)

- xPath

- Element (div, a , p etc.)

- Attribute

- Child (quand plusieurs enfants dans un element parent)

- Index (dans un tableau)

- Content (contenu dans l’élément)

-Des méthodes permettant de cliquer sur des éléments de la page :

- Click() //Clique simplement sur l’élément

- Des méthodes permettant de s’assurer que des éléments sont présents sur la page (qu’ils soient cachés de l’utilisateur ou non)

- IsPresent() //Si présent sur la page mais pas forcément visible

- isDisplayed() // Si présent et visible par l’utilisateur à l’heure ou la fonction est appelée

- Des méthodes permettant de remplir des champs (input)

-sendKeys() // Permettant d’envoyer une String quelque part (en sélectionnant l’élément // Auparavant

## Utilisation de Selenium chez Kurmi

### Pattern adopté

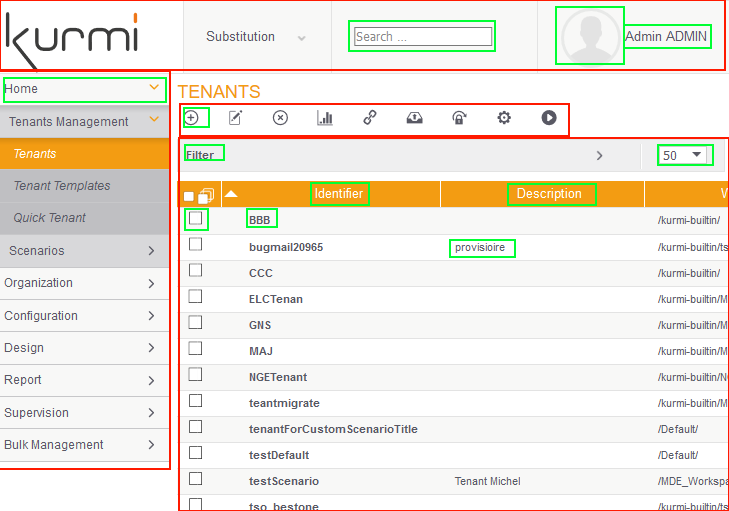
Le pattern que nous avons adopté est le Page Object.

Le pattern Page Object est un pattern compréhensible par tous qui présente une application en différentes pages, qui comprennent chacune différents composants, qui comprennent chacun différents éléments web

Les principaux éléments du pattern Page Object sont donc :

* Les PageObject
* Les ComponentObject
* Les WebElements .

Exemple concret :



Ici nous avons une page recensant les Tenants(Clients) d’une plateforme interne à Kurmi.

Eeee Composants

Eléments

Les composants sont donc un ensemble d’éléments.

C’est alors à nous de définir qui sera composant et qui sera élément.

Dans cette page, nous avons défini le menu de gauche comme un Composant et chaque sous menu/sous menu comme un élément.

Nous avons défini la liste des tenants comme un Composant et en-tête, chaque checkbox, chaque champ du tableau comme des éléments.

La barre de bouton est un composant, chaque bouton est un élément.

### Les tests avec Selenium

Chez Kurmi, pour tester l’application, nous avons décidé d’adopter le pattern de test suivant :

### 

Nous déclarons une liste de tests que l’on nomme TestItemList.

Cette liste contiendra plusieurs TestItem.

Ces test items contiendront :

* + - Une TestResultList qui est une liste de résultats
    - actionStep : plusieurs actions exécutées par Selenium
    - CheckItem : qui viendront vérifier si les action tests executés on fait ce qu’on attendait d’elles.

Un exemple avec du code que j’ai développé :

//On déclare testList

List<AbstractTestItem> testList = **new** ArrayList<>();

//On y ajoute les TestItem en passant les données demandées dans le constructeur de ce test.

// Ici nous utilisons un enum pour lister les menus (MenuMapping) et nous cliquons sur le bouton qui mènera // à la liste des divisions

testList.add(**new** GoToMenuAndCheckTitleTestItem(

"Initialisation du test", MenuMapping.***LIST\_DIVISIONS***,

***EXPECTED\_TITLE\_DIVISIONS\_LIST***));

testList.add(**new** DivisionsPage(

"KUP\_GUI\_Administrator\_02\_02-01 - Divisions Page")

);

On va ensuite dans le testItem DivisionsPage :

//On cree la liste de resultat TestResultList

TestResultList resultList = **new** TestResultList(getId());

//On y ajoute ensuite des tests

//Ici nous vérifions que le bouton Create est affiché.

resultList.add(check(listDivisionPage.isActionCreateDiplayed(),

MESSAGE\_CREATE, listDivisionPage, **true**));

//Ici nous verifions que le bouton Delete est affiché.

resultList.add(check(listDivisionPage.isActionDeleteDiplayed(),

MESSAGE\_DELETE, listDivisionPage, **true**));

[…]

//On retourne cette liste pour obtenir en logs des chaines de chaque test  
//et voir quels tests ont été exécutés avec succès (résultat attendu bon)   
// et quels tests non.

**return** resultList.generateGlobalResult(listDivisionPage);

### Les différents tests exécutés

Pour chaque tests executés, je devais réfléchir à toutes les possibilités possibles.   
Par exemple lors d’une création, je devais savoir quels champs n’etaient pas censés être corrects en les entrant (login trop long, email incorrect etc.).

Je vais donc lister ici l’ensemble des tests que j’ai développés par ensemble de page.

#### Les tests sur les super administrateurs.

Sur cet ensemble de page, il y avait :   
 -Une liste avec tous les super administrateurs

-Une page de création

-Une page d’information

-Une page de modification

-2 pages de suppressions (l’une depuis la liste en cochant une ou plusieurs chekcbox, l’autre depuis la page d’information)

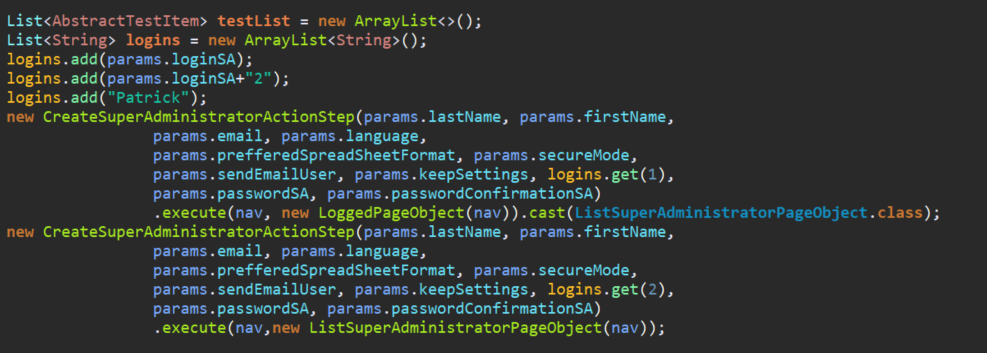
Sur chaque page je devais vérifier que tous les composants étaient présents.

J’ai donc commencé par la page de liste. Le menu de gauche et la bannière étant présents sur toutes les pages, je n’avais pas besoin de le vérifier (c’était vérifié à chaque test. Il s’agissait d’une PageObject parente contenant ces deux éléments.)

Chaque test commence de l’écran de connexion de l’application, et fait appel à un autre test qui va nous connecter en tant qu’administrateur.

Création

La première chose que j’avais à faire était de créer un jeu de test. Pour cela j’ai appelé deux fois une ActionStep de création de super administrateur en lui passant les paramètres que je souhaitais lui passer.



Cette ActionStep :

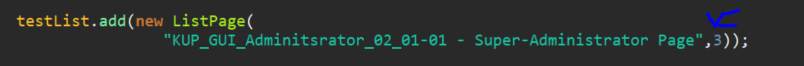
-Clique sur le menu des super administrateurs

-Clique sur le bouton de création

-Remplit les différents champs du formulaire avec les paramètres passés

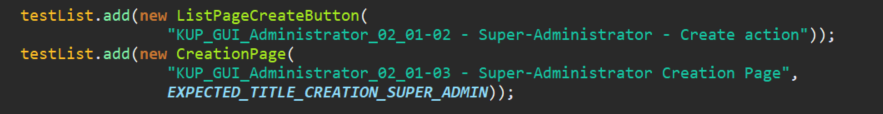
-Clique sur le bouton de création du super administrateur.

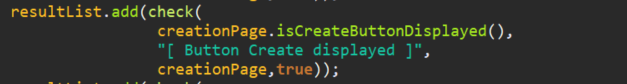
-Vérifie qu’on arrive bien sur la liste des super administrateurs (sinon, c’est que quelque chose s’est mal passé).

Après la création de ces 2 Super administrateurs, je pouvais commencer mes tests. J’ai donc commencé par vérifier que le nombre de super administrateurs présents sur cette page étaient corrects.

Ici, j’attends 3 super administrateurs. (les deux créés plus celui de base)

Le test va ensuite cliquer sur le bouton de création et vérifier que le titre de la page sur laquelle on arrive est bon, et que tous les éléments sont présents.

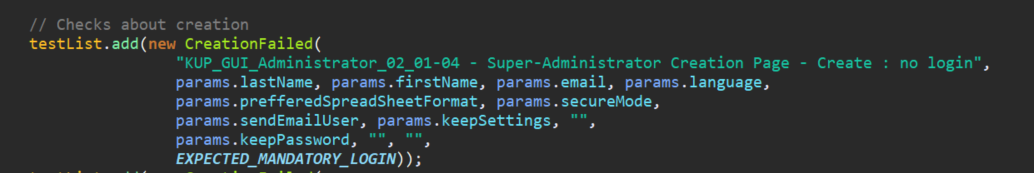




Résultat en console quand tout est OK.



J’ai ensuite testé les messages d’erreurs dans le cas où l’utilisateur n’entrait pas des choses corrects dans les champs.



On passe en paramètres les valeurs des champs qui vont être remplit dans le formulaire ainsi que le message d’erreur attendu. (EXPECTED\_MANDATORY\_LOGIN)

Les resultats en console sont les suivants.

DONE\_OK : CreationFailed : [ At least one error message must be : {The parameter 'Login of the administrator' is mandatory} result are : {The parameter 'Login of the administrator' is mandatory} ]

DONE\_OK : CreationFailed : [ At least one error message must be : {The parameter 'Password of the administrator' is mandatory} result are : {The parameter 'Password of the administrator' is mandatory} ]

DONE\_OK : CreationFailed : [ At least one error message must be : {The parameter 'Lastname of the administrator' cannot exceed 64 characters} result are : {The parameter 'Lastname of the administrator' cannot exceed 64 characters} ]

DONE\_OK : CreationFailed : [ At least one error message must be : {The parameter 'Firstname of the administrator' cannot exceed 64 characters} result are : {The parameter 'Firstname of the administrator' cannot exceed 64 characters} ]

DONE\_OK : CreationFailed : [ At least one error message must be : {The parameter 'E-mail of the administrator' is not in the correct format} result are : {The parameter 'E-mail of the administrator' is not in the correct format} ]

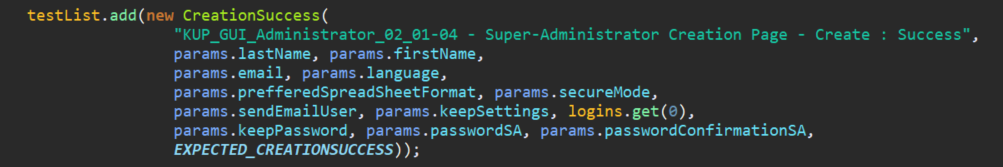
DONE\_NOT\_OK : CreationFailed : [ At least one error message must be : {BDD\_UNIQUE\_CONSTRAINT\_VIOLATION\_NATES\_ADMIN\_USERID} result are : {The value "admin" already exists in the database.} ]

La console renvoi :

-DONE\_OK quand le message attendu est le bon.

-DONE\_NOT\_OK et arrête l’exécution du test quand le message attendu n’est pas le bon. (c’est souvent le cas quand en local on a un message, mais que sur le serveur où seront exécutés les tests en a un différent)

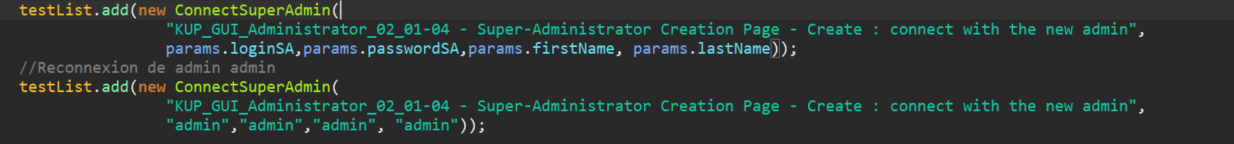
Après avoir testé qu’on avait les bons messages d’erreur quand une création ne réussis pas ; on test que la création marche bien quand on rentre les bons paramètres.



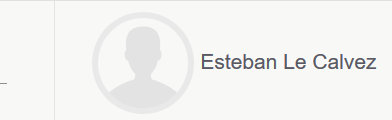
Ce test est sensiblement le même mais on arrivera sur la page avec la liste des super administrateurs et un message de succès nous disant que l’utilisateur est créé.

Connexion de l’utilisateur créé

Après avoir créé avec succès le super administrateur, on s’assure que ce n’est pas qu’un message vert nous disant qu’il est créé et on test de le connecter.



Ce test commence sur la page de login et s’attend à arriver sur une page loguée de l’application. Il vérifiera dans la bannière si le nom et prénom du super administrateur sont corrects.

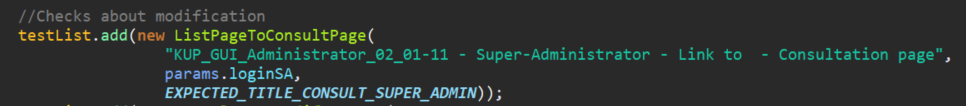


On reconnecte ensuite le super administrateur par défaut pour continuer les tests.

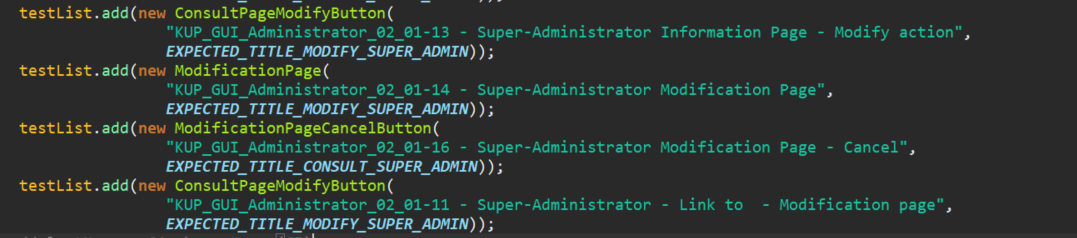
Modification

Une fois s’être assuré que la création des super administrateurs fonctionne bien, nous passons à la modification.

La première étape est de se rendre sur le super administrateur que nous voulons modifier afin de le modifier.



On passe donc en paramètre le login du Super administrateur afin que le test clique dessus et vérifie l’intitulé de la page.

On clique ensuite sur le bouton de modification. 

Ici on vérifie qu’on arrive bien sur une page de modification avec tous les champs présents, à la manière de la création.

Une fois sur la page de modification, on vérifie le bon fonctionnement du bouton Cancel en cliquant dessus et et vérifiant qu’on arrive sur la bonne page, puis on retourne sur la page de modification pour commencer les tests sur les modifications.

A la manière de la création, on va vérifier qu’en entrant des choses incohérentes dans les champs du formulaire, on obtient des messages d’erreur.