Le génie pour l'industrie

RAPPORT TECHNIQUE
PRÉSENTÉ À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
DANS LE CADRE DU COURS GTI795 PROJET DE FIN D'ÉTUDES EN LOG-TI

Remplacement de UISpec4J comme cadriciel de test

ALEXIS BERTRAND TCHAMANBE
TCHA12118008

DÉPARTEMENT DE GÉNIE LOGICIEL ET DES TI

Professeur-superviseur

Alain APRIL

MONTRÉAL, 20 DECEMBRE 2017
AUTOMNE 2017

REMERCIEMENTS

Je remercie tous ceux qui m'ont de près ou de loin aidé à réaliser ce projet. Il s'agit particulièrement de :

— Alain April, pour son orientation technique et pédagogique

— Mathieu Dupuis, pour son assistance technique

— Tous les membres de ma famille pour leur soutien moral et psychologique.

Remplacement de UISpec4J comme cadriciel de test

ALEXIS BERTRAND TCHAMANBE
TCHA12118008

RÉSUMÉ

Présentement dans le cours LOG240 utilise le cadriciel UISpec4J pour effectuer des tests boite noire dans les séances de laboratoire. Ce cadriciel est désuet, n'est plus supporté et ne reflète pas la réalité en entreprise, d'où la nécessité de trouver un cadriciel populaire largement utilisé dans le domaine des tests applicatifs. De plus, plusieurs étudiants ont de la difficulté à différencier les tests boites noires des tests boite blanche, car dans les deux laboratoires, les étudiants ont le code source de l'application tester. Ce problème académique serait inexistant si l'application testée était une application web hébergée sur un serveur distant. Dans le cadre de ce projet de fin d'études, l'objectif est de réévaluer les conditions de ce laboratoire. Ceci nous a permis de proposer une application Web Open source hébergé sur un serveur distant, ainsi qu'un cadriciel de test automatisé, permettant d'implémenter l'automatisation des différents cas de tests.

TABLE DES MATIÈRES

Page

LISTE DES TABLEAUX

Page

LISTE DES FIGURES

Page

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

SDK: Software Development Kit

ETS : École de Technologie Supérieure

LOG240 : Cours de Tests et maintenance à l'ETS pour les étudiants en génie logiciel

INTRODUCTION

L’assurance qualité est une étape essentielle dans le cycle de développement des applications. [9] Elle est un ensemble d'activités planifiées et systématiques de toutes les actions nécessaires pour fournir une assurance suffisante qu'un logiciel soit conforme aux exigences établies. L’automatisation des tests applicatifs est de plus en plus importante dans l’exercice de l’assurance qualité. Plusieurs outils existent de nos jours pour faciliter l’automatisation des tests applicatifs. Le choix d’un outil approprié est le facteur critique dans la réussite de cet exercice. Dans la formation des étudiants en Génie logiciel à l’ÉTS, ils doivent réussir le cours LOG240 Tests et maintenance, qui les initie à l’application des différents types de tests selon les principes du génie logiciel. Ce cours utilise le cadriciel UISpec4J pour effectuer des tests boite noire dans ses séances de laboratoire. Ce cadriciel n'est plus supporté et n’est pas utilisé en industrie, d’où la nécessité de proposer un nouveau cadriciel de test plus récent et utilisé dans l’industrie. Selenium est un outil populaire qui permet d’automatiser les tests des applications web. De plus, il est utilisé dans plusieurs entreprises comme cadriciel d’automatisation des tests. De ce fait l’objectif de ce projet sera de remplacer UISpec4J par Selenium comme outil d’automatisation des tests pour les laboratoires du cours LOG240. De plus, il est question de trouver une application Web open source, pour remplacer l’ancien logiciel (FinanceJ) qui ne permet pas de faire des test Web. Après avoir trouvé une application open source appropriée pour ce projet, nous allons proposer un guide d’utilisation de l’application, ainsi que l’intégration de Selenium à JUnit.

CHAPITRE 1

DÉPLOIEMENT D'UNE APPLICATION WEB OPEN-SOURCE

1.1 Évaluation des Applications Web Open-source

1.1.1 Présentation des applications :

Dans cette étape, il est question de trouver une application web open source, permettant d'implémenter au moins dix cas d'utilisations. Dans nos différentes recherches, nous avons trouvé plusieurs applications, mais très peu avaient une logique d'affaire respectant les exigences du client, à savoir :

- Une application Web faite suivant les bonnes pratiques (MVC, Springs, Hybernate…), et dans un langage de développement requis par le projet (compte tenu des environnements déployés à l'ÉTS, il est essentiel que l'application soit en Java ou C#)
- Une application qui implémenter au moins dix cas d'utilisations pertinents.

Les applications Java en particulier sont difficiles à déployer dans certains cas surtout s'il n'y a pas une documentation claire. Nous avons téléchargé plusieurs applications Java que nous n'avons pas pu déployer en raison du manque de documentation, et l'utilisation des Frameworks très complexes.

La plupart des applications open source sur internet sont soient incomplète, soit mal conçue.

Parmi les applications open source que nous avons pu déployer, nous avons retenu préliminairement quatre et de ces quatre, nous allons sélectionner l'application que nous jugeons appropriée pour simuler un environnement d'apprentissage idéal dans le cadre du cours LOG240.

- **L'application Java EE appelé Doctor's Online**

Langage de développement : Java

Contexte : L'application permet de gérer une clinique. Les patients peuvent créer un compte, prendre des rendez-vous en ligne et le docteur peut consulter les rendez-vous des patients. L'administrateur peut créer et supprimer les utilisateurs.

Qualité du code : Mauvais, aucun patron de conception

Décision : Rejet, car les cas d'utilisation ne sont pas bien implémentés et la qualité du code n'est pas bonne.

Lien de l'application: http://www.java2s.com/Open-Source/Java_Free_Code/Web_Application/Download_BookMyDoc_Free_Java_Code.htm

Le tableau ci-dessous nous en dit plus sur le fonctionnement de l'application.

Cas d'utilisations	Description	Fonctionnement
Authentification (3 cas)	Permet à un Docteur, un Patient ou un admin de s'authentifier.	Fonctionnel
Créer d'un docteur	Permet à un admin de créer un Docteur	Non implémenté
Supprimer un docteur	Permet à un admin de supprimer un Docteur	Partiellement implémenté
Supprimer un patient	Permet à un admin de supprimer un patient	Partiellement implémenté
Demander un rendez-vous	Permet à un patient de prendre un rendez-vous	Non implémenté
Supprimer un rendez-vous	Permet à un patient de supprimer un rendez-vous	Non implémenté
Laisser un feedback	Permet à un patient de laisser un feedback	Non implémenté
Consulter son courriel	Permet à un patient ou à un Docteur de consulter son courriel	Partiellement implémenté
Envoyer un courriel	Permet à un patient ou à un Docteur d'envoyer un courriel	Partiellement implémenté
Consulter les feedback	Permet à un Docteur de consulter	

	les feedbacks	
Demander un départ	Permet à un Docteur de demander un départ	Partiellement implémenté

Tableau 1.1 Fonctionnalités de l'application

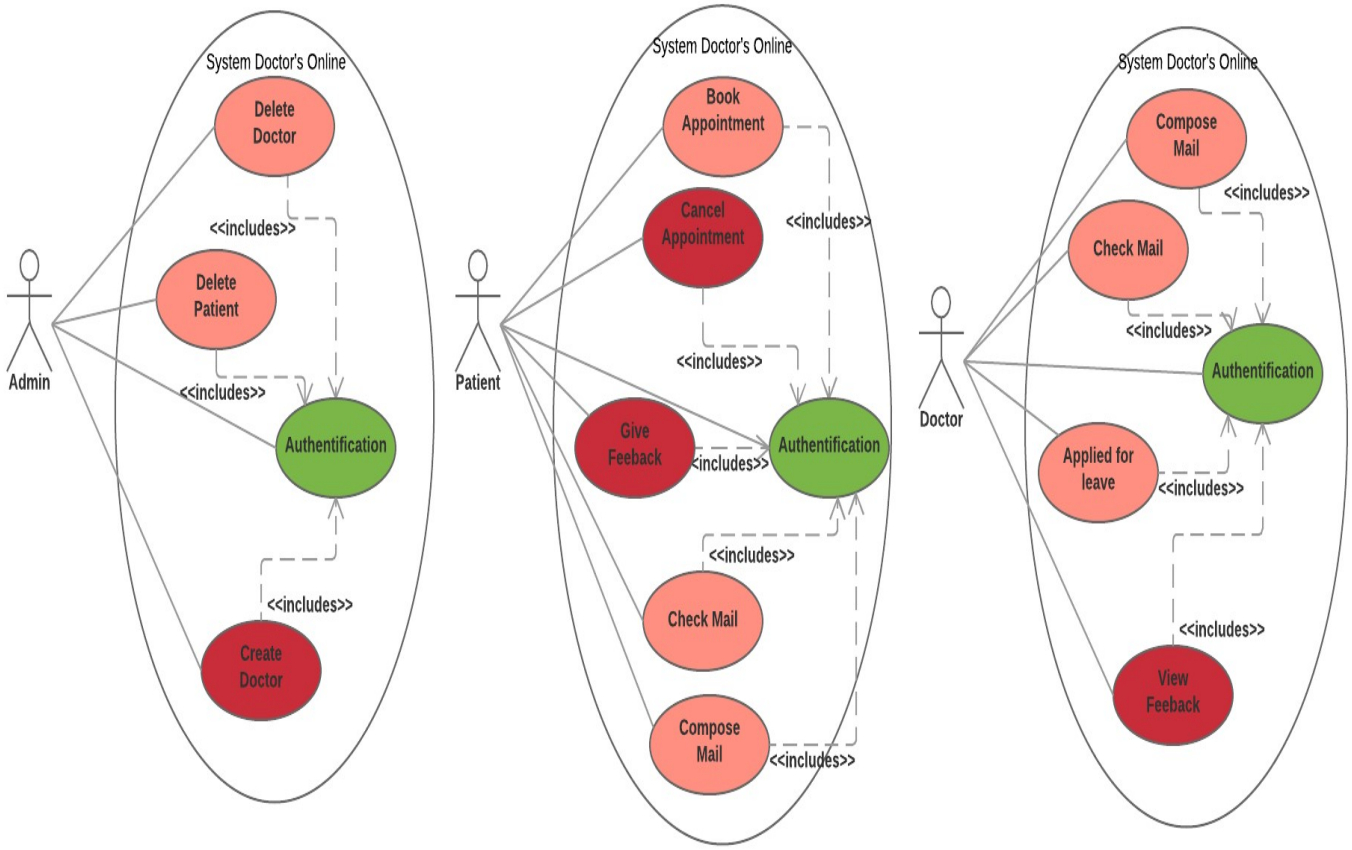


Figure 1.1

❖ **L'application Football Tactic Creator**

Langage de développement : C#

Contexte : C'est une application qui permet de créer les tactiques de football. Un utilisateur peut créer un compte, se connecter et créer sa propre tactique de jeux. Implémente 6 cas d'utilisations tous fonctionnels.

Qualité du code : Bon

Décision : Rejet on n'a pas assez de cas d'utilisation, pour pouvoir respecter les exigences de l'application recherchée.

Lien de l'application : https://github.com/stoian1929/Exam-Project

Cas d'utilisations	Description	Fonctionnement
Authentification	Permet à un utilisateur de s'authentifier.	Fonctionnel
Créer une tactique	Permet à un utilisateur de créer une tactique de jeux	Fonctionnel
Supprimer une tactique	Permet à un utilisateur de supprimer une tactique de jeux	Fonctionnel
Modifier une tactique	Permet à un utilisateur de modifier une tactique de jeux	Fonctionnel
Afficher une tactique	Permet à un utilisateur d'afficher une tactique de jeux	Fonctionnel

Tableau 1.2 Fonctionnalités de l'application 2

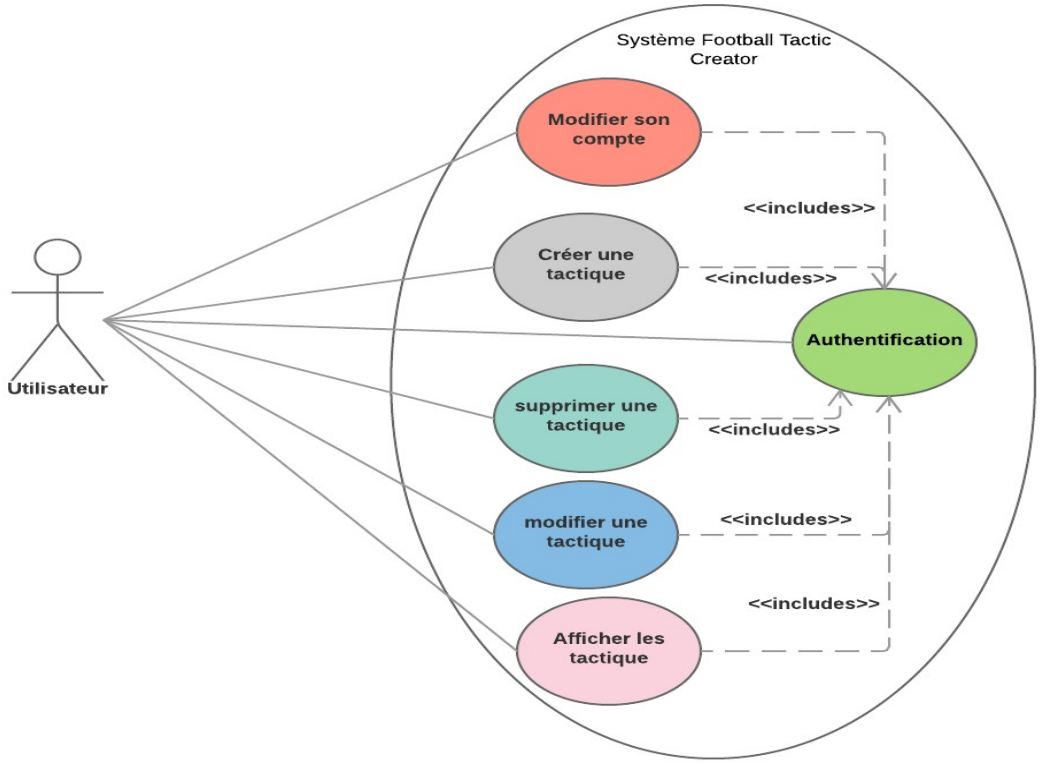


Figure 1.2

- **L'application easySchool system**

Langage de développement : PHP

Contexte : EasySchool system est une application qui permet de gérer une salle de classe. Un enseignant peut créer un compte et se connecter pour gérer les absences, mettre des cours en ligne, et envoyer un courriel aux étudiants. Un étudiant peut se connecter pour voir la liste de ces cours, et s'inscrire à un cours. La documentation de l'application est disponible en ligne. Elle implémente 12 cas d'utilisations tous fonctionnels.

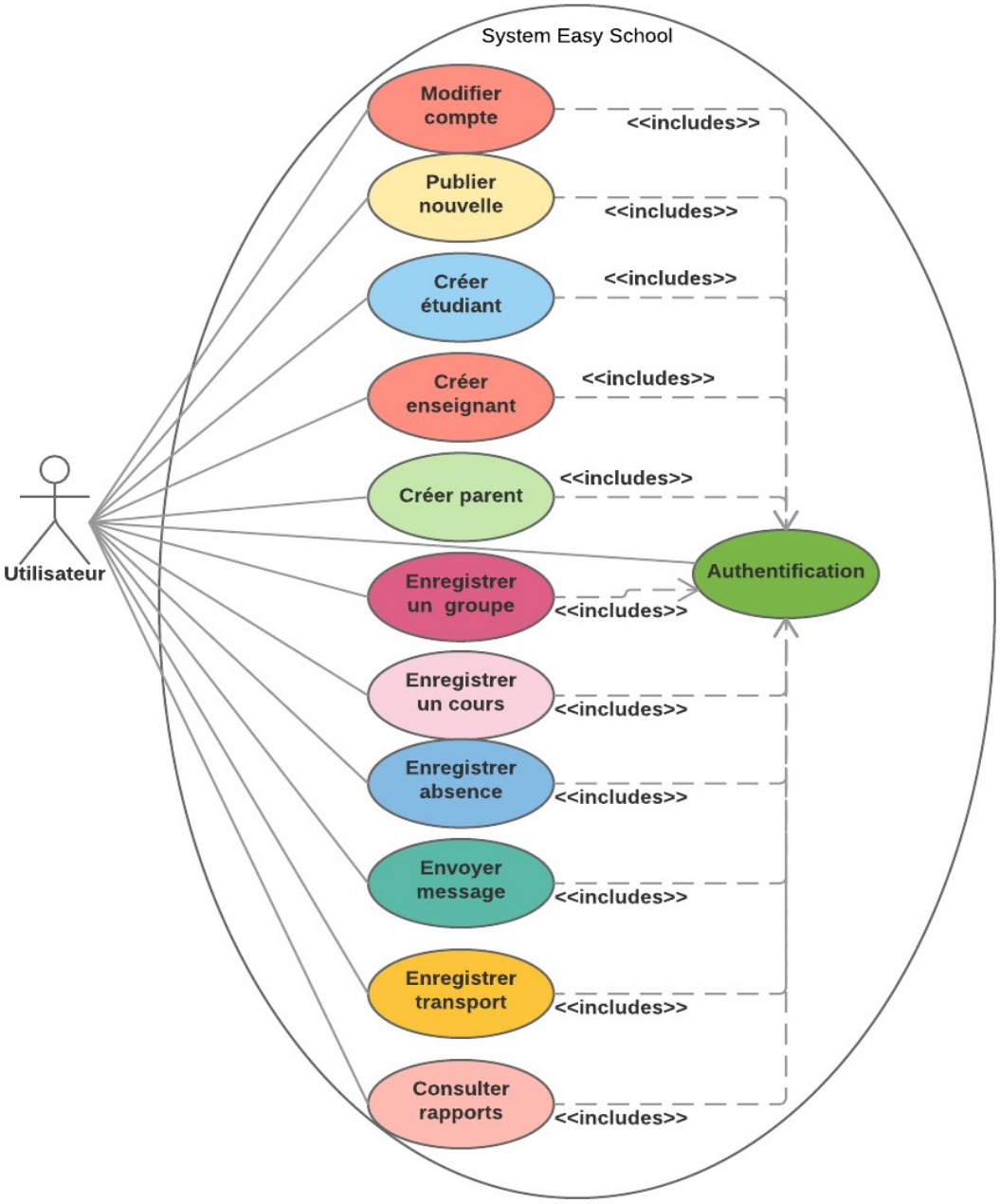
Qualité du code : Bon,

Décision : Rejet, car l'application est faite en PHP qui n'est pas un langage utilisé à l'ÉTS dans le cadre des laboratoires.

Liens:http://codes-sources.commentcamarche.net/source/101791-easyschool-systeme-gestion-des-etablissments-scolaires

Cas d'utilisations	Description	Fonctionnement
Authentification	Permet à l'utilisateur de s'authentifier	Fonctionnel
Modifier Compte	Permet à l'utilisateur de modifier son compte	Fonctionnel
Publier nouvelle	Permet à l'utilisateur de publier une nouvelle dans le système	Fonctionnel
Créer étudiant	Permet à l'utilisateur de créer un étudiant dans le système	Fonctionnel
Créer enseignant	Permet à l'utilisateur de créer un enseignant dans le système	Fonctionnel
Créer parent	Permet à l'utilisateur de créer un parent dans le système	Fonctionnel
Enregistrer groupe	Permet à l'utilisateur d'enregistrer un groupe de cours dans le système	Fonctionnel
Enregistrer cours	Permet à l'utilisateur d'enregistrer un cours dans le système	Fonctionnel
Enregistrer absence	Permet à l'utilisateur d'enregistrer les absences dans le système	Fonctionnel
Enregistrer transport	Permet à l'utilisateur d'enregistrer les informations de transport dans le système	Fonctionnel
Consulter les rapports	Permet à l'utilisateur de consulter les différents rapports.	Fonctionnel
Envoyer message	Permet à l'utilisateur d'enregistrer les absences dans le système	Fonctionnel

Tableau 1.3 Fonctionnalités de l'application 3



Cas d'utilisation: Application easySchool system

Figure 1.3

- **Lapplication Online Banking System**

Application Java (Spring Boot, Hibernate),

Langage de développement : Java (Spring Boot, Hibernate)

Contexte : OnlineBankingSystem est une application bancaire qui offre aux utilisateurs les fonctionnalités de création de compte, de dépôt et de retrait d'argent. Elle implémente 13 cas d'utilisation, toute fonctionnelle. La logique d'affaires est très intéressante et le langage est approprié pour créer un environnement d'apprentissage dans les laboratoires du cours LOG240

Qualité du code : Bon, très bonne implémentation de Spring MVC

Décision : Retenue en raison de sa logique d'affaires et de la qualité du code. Cette application satisfait les exigences du client. Il implémente des cas d'utilisations simples (13 cas d'utilisations) et le cadre technologique (Java, Springs, Boot, Hybernate, Maven) est adapté au laboratoire à l'ÉTS. De plus l'application contient des défauts, et plusieurs cas d'utilisation biens que fonctionnels nécessite d'être amélioré, ce qui est idéal pour les laboratoires de maintenance précédente ceux de test.

Lien de l'application: https://github.com/bijayregmi/OnlineBankingSystem

Cas d'utilisations	Description	Fonctionnement
Authentification	Permet à un client de créer un compte	Fonctionnelle
Login	Permet à un client ou à un admin de se connecter au syst ème	Fonctionnelle
Demander compte chèque	Permet aux clients de faire une demande de compte chèque	Fonctionnelle
Demander compte d'épargne	Permet aux clients de faire une demande de compte d'épargne	Fonctionnelle
Faire un depot dans un compte	Permet aux clients de faire un depot dans son compte	Fonctionnelle
Faire un retrait dans un compte	Permet aux clients de faire un retrait de son compte	Fonctionnelle
Afficher son profil	Permet aux clients d'afficher son profil	Fonctionnelle
Approuver la création d'un compte	Permet à l'administrateur d'approuver la création d'un compe	Fonctionnelle
Désapprouver la création d'un compte	Permet à l'administrateur de desapprouver la création d'un compe	Fonctionnelle
Afficher la liste des clients	Permet à l'administrateur d'afficher la liste des clients	Fonctionnelle

Modifier le profil d'un client	Permet à l'administrateur de modifier le profil d'un client	Fonctionnelle
Afficher l'historique de ses activités	Permet à l'administrateur d'fficher l'historique de ses activités	Fonctionnelle
Afficher les requêtes d'un client	Permet à l'administrateur d'afficher les requêtes d'un client	Fonctionnelle

Tableau 1.4 Fonctionnalités de l'application 4

Figure 1.4

CHAPITRE 2

ADAPTATION DU LABORATOIRE 1 LOG240

2.1 Énoncé du laboratoire

Une adaptation complète de l'énoncé du laboratoire a été faite. Le contexte du laboratoire a été redéfini suivant les exigences du client mentionné ci-haut. L'énoncé initial parlait de l'application FinanceJ, une application Java pas en très bon état et pas assez conviviale. Les étudiants devaient dans un premier temps faire la revue du code, après avoir déployé l'environnement de travail. Dans le cadre de notre projet, nous avons adapté l'énoncé du laboratoire à la nouvelle application OnlineBankingSystem, qui sera désormais proposé aux étudiants du cours LOG240. L'énoncé au complet se trouve en Annexe I du document.

2.2 Guide d'utilisation de l'application

L'application Open source OnlineBankingSystem est une application Maven, fait dans le langage Java utilisant plusieurs Frameworks tels que Spring et Hibernate. Ces différents Frameworks sont très populaires dans le domaine du domaine du développement et pourront très bien être utilisés dans un environnement d'apprentissage.

Nous avons proposé un guide complet du laboratoire 1, incluant les différentes étapes à suivre pour déployer l'application dans Intellij. L'énoncé et le guide d'installation du laboratoire 1 sont joints à ce rapport. Une copie du guide d'installation proposé se trouve également en annexe II

CHAPITRE 3

AUTOMATISATION DES TESTS

3.1 Généralités

[1] Un test automatisé est un test dont l'exécution ne nécessite pas l'intervention d'un humain. L'exécution de tests automatisés requiert donc l'utilisation de solutions informatiques dont le but est d'exécuter des actions, soit spécifiquement dans un navigateur web, soit plus généralement au niveau du système d'exploitation. Dans un environnement de développement web, il est nécessaire d'automatiser les tests des applications web, afin d'être capable d'assurer la qualité logiciel au cours du temps. [2] Les tests permettent de vérifier que le système répond aux spécifications développer par les analystes d'affaires. Les phases de test dans le cycle de développement d'un produit logiciel permettent d'assurer un niveau de qualité défini.

Dans le cours LOG240, les étudiants devront compléter l'implémentation de l'application proposée et ensuite faire, dans cet ordre, des tests boite noire et boite blanche. Pour les tests boite blanche, le code source de l'application sera fourni aux étudiants au afin de créer des tests avec JUnit. Pour les tests boite noire, l'application sera hébergée sur un serveur distant et les étudiants n'auront pas accès au code source. Lors des tests boite noire, ils devront développer des scripts devant servir à automatiser les tests.

[3] Plusieurs outils permettent d'automatiser les tests, parmi lesquels on peut citer :

— Selenium (Cadriciel open source de tests automatisés développé en Java)
— PhantomJS (utilisé pour automatiser des interactions avec des pages web)
— Jenkins (outil open source d'automatisation des tests écrits en java)
— Silk test (un outil d'automatisation de tests basé sur des rôles)

Selenium est l'outil qui a été proposé dans le cadre de la restructuration du laboratoire du cours LOG240 en raison de sa popularité et de ses possibilités d'utilisation, car permet d'écrire des tests d'automation pour toutes les applications web, quelque soit le langage. De plus il est l'un des outils les plus populaires utilisés en industrie pour l'automatisation des tests pour les applications Web.

3.2 Selenium et automatisation des tests

[4] Selenium est un de ces outils d'automatisation, pour les tests d'interface des applications Web. C'est un projet distribué sous la licence Apache 2.0. Il se compose de deux parties :

— **Selenium IDE** : c'est une extension de Firefox, qui permet d'enregistrer une suite d'actions, qu'il sera possible de rejouer à volonté. Il est directement intégré dans Firefox, et pas besoin d'avoir des connaissances en programmation pour l'utiliser.

— **Selenium WebDriver** : C'est un API, disponible pour plusieurs langages (Java Python...), permettant de programmer des actions sur l'interface, et à vérifier les réponses. Les actions à réaliser peuvent être exportées depuis Selenium IDE.

Dans le cas de notre projet, nous avons utilisé Selenium WebDriver et JUnit dans Eclipse, pour automatiser et exécuter les différents cas de test.

CHAPITRE 4

CONCEPTION ET IMPLÉMENTION DES SCRIPTS DE TEST

4.1 Selennium Web Driver, JUnit et Eclipse

JUnit

[5] JUnit est un framework open source d'automatiser les tests en Java. Il peut être utilisé sur plusieurs environnements (Eclipse, Intellij, Netbeans...). Le principal intérêt est de s'assurer que le code répond toujours au besoin même après d'éventuelles modifications. Les tests sont exprimés dans des classes sous la forme de cas de tests avec leurs résultats attendus. JUnit exécute ces tests et les compare avec ces résultats. Avec Junit, l'unité de tests est une classe dédiée qui regroupe des cas de tests. Ces cas de tests exécutent les tâches suivantes :

— création d'une instance de la classe et de tout autre objet nécessaire aux tests
— appel de la méthode à tester avec les paramètres du cas de test
— comparaison du résultat obtenu avec le résultat attendu : en cas d'échec, une exception est levée

Pour pouvoir utiliser JUnit dans un projet, il faut ajouter le fichier junit.jar dans le classpath du projet.

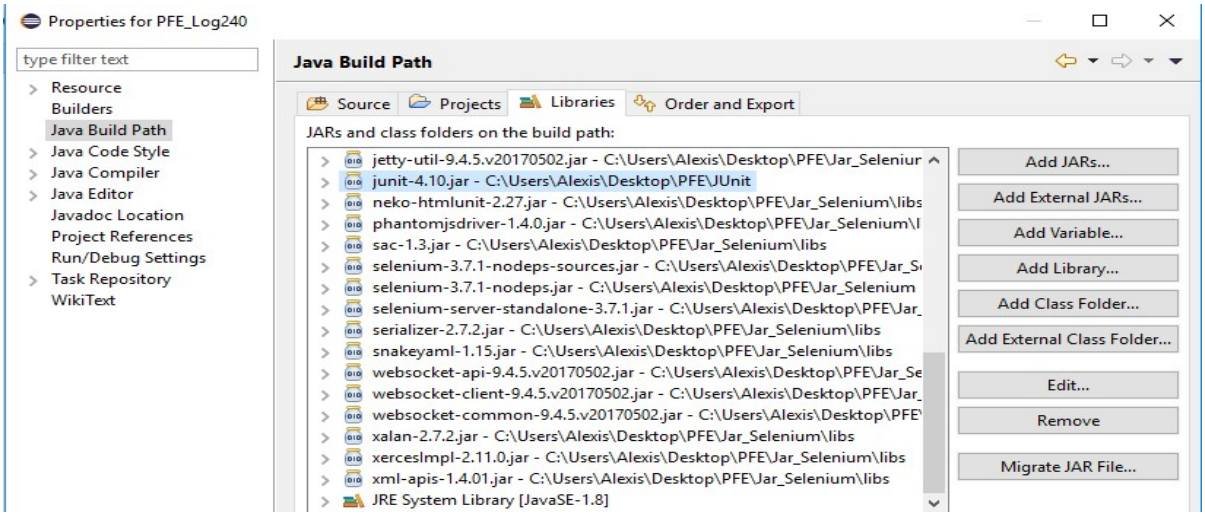


Figure 1.5

Une fois la librairie JUnit importée dans le projet, on peut exécuter les différentes méthodes de la librairie. Dans notre projet, nous allons utiliser les méthodes ci-dessous pour automatiser les scripts Selenium:

```
import org.junit.After;
import org.junit.Before;
import org.junit.Test;
import junit.framework.Assert;
```

Junit.framework.assert nous permet de définir les conditions de réussite de test. Les détails sur l'utilisation de méthodes JUnit avec Selenium seront développés plus bas dans la section Automatisation et Exécution des cas de tests

Selennium Web Driver

[6] WebDriver est un framework de tests fonctionnels issu du projet Selenium, célèbre outil d'automatisation de tests pour navigateurs. Il permet d'exécuter des tests dans des navigateurs du marché (Firefox, Chrome, Internet Explorer), distants ou en local. Il offre la possibilité d'exécuter des tests sur des navigateurs avec des évènements "natifs" tels qu'un utilisateur final le ferait.

Avec WebDriver, on peut exécuter des tests sur des navigateurs distants en parallèle sur de multiples plateformes et versions dans plusieurs langages de programmation (Java, C#, Python, Perl, Ruby, PHP), grâce à une architecture client/serveur.

[7] Le WebDriver est basé sur un modèle client-serveur : un client de test envoie des "commandes" via des requêtes HTTP à un serveur WebDriver après initialisation d'une session. Ce dernier distribue les commandes auprès des drivers des navigateurs concernés. Ces drivers exécutent les commandes sur les navigateurs en question via des mécanismes d'automatisation interne, de l'OS, tel qu'illustré sur la figure ci-dessous. Par défaut, les navigateurs et leurs drivers doivent être disponibles sur la même machine que le serveur WebDriver. Il faut donc indiquer dans le programme qui effectuera les tests le Webdriver à utiliser.

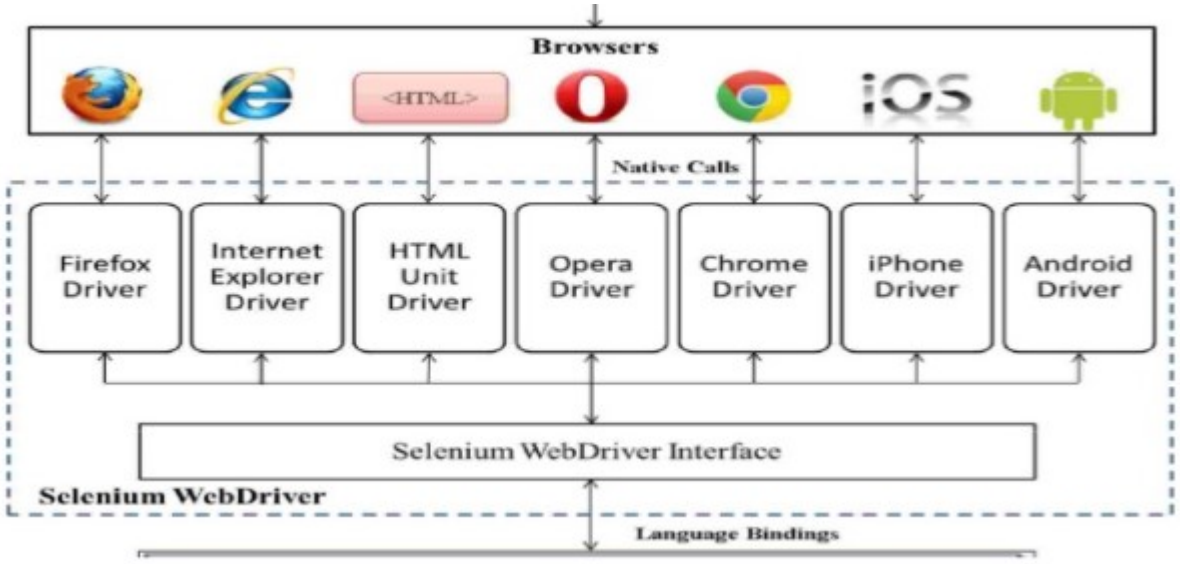


Figure 1.6 [7]

Pour pouvoir utiliser Webdriver dans un projet Eclipse, il faut ajouter les fichiers .jar de la librairie Selenium dans le classpath du projet. Il faut pour cela :

- Ouvrir le lien https://selenium-release.storage.googleapis.com/index.html et sélectionner le dossier le plus récent contenant les différents outils de Selenium.
- Télécharger selenium-server-standalone-3.8.1.jar et selenium-server-3.8.1.zip puis dézipper le second fichier.
- À partir du projet crée précédemment, sélectionner les propriétés du projet, puis cliquer sur ''Java Build Path'' (voir figure ci-dessous)
- Dans l'onglet librairies, cliquer sur le bouton « Add External Jar »
- Sélectionner successivement tous les fichiers .jar téléchargé (selenium-server-standalone et selenium-server) à partir de leur répertoire sur le disque et les ajoutés à la librairie. La figure ci-dessous montre les différentes librairies importées dans le projet Eclipse

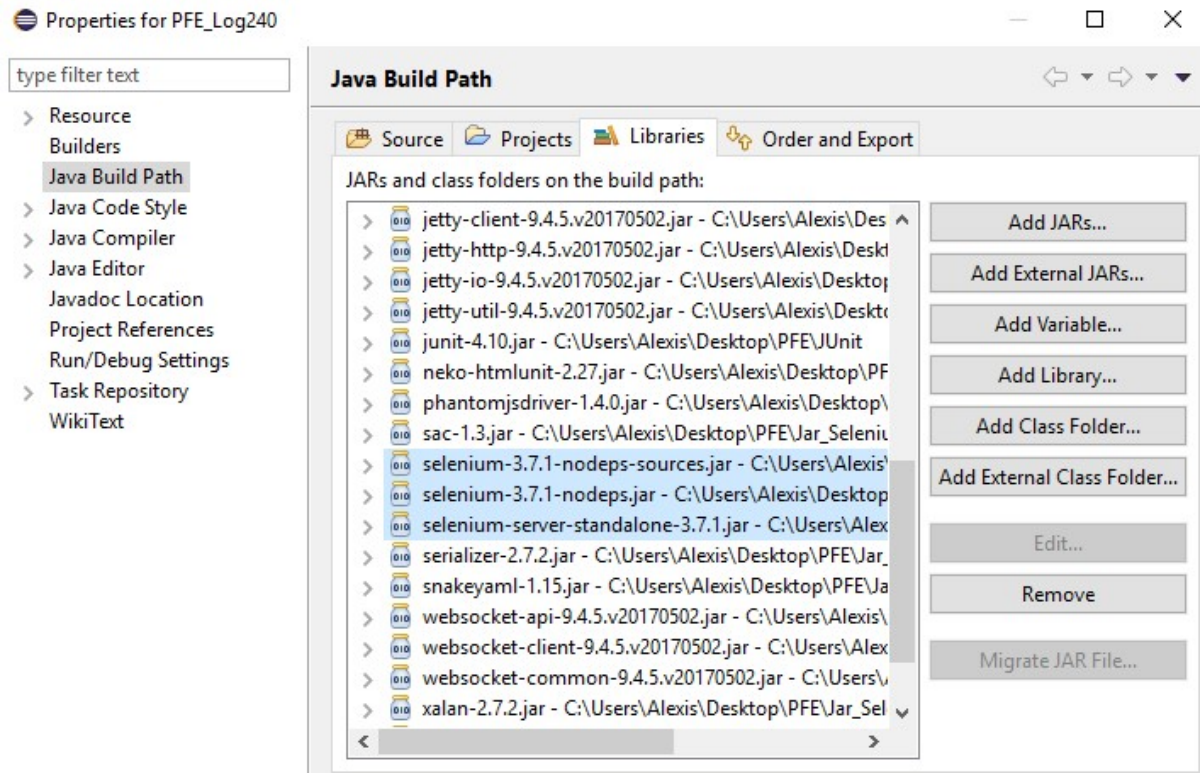


Figure 1.7

4.2 Automatisation et Exécution des cas de tests

Une fois les librairies JUnit et Selenium importées, nous pouvons désormais écrire nos différents cas de test et les automatiser. Nous avons utilisé la structure JUnit et importé les librairies ci-dessous :

```
import java.util.concurrent.TimeUnit;
import org.junit.After;
import org.junit.Before;
import org.junit.Test;
import junit.framework.Assert;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

	public class SeleniumJUnit {

	@Before
	public void beforMethod(){
	}
	@After
	public void afterMethod(){
```

```
                driver.close();
        }
        @Test
//Insersion de tous les cas de test dans cette séquence.
        }
```

[8] La méthode @Before est tout d'abord exécutée avant les méthodes de test, et la méthode @After est appelée à la fin, généralement pour arrêter le test. Les tests même sont appelés dans des méthodes @Test, elles font des traitements et vérifie le bon comportement des classes testées par des méthodes assert de JUnit. Toute assertion non vérifiée est signalée comme défaillante. Dans notre cas, nous avons utilisé la méthode Assert.assertTrue() pour permettre à JUnit de conclure si un test terminé par un succès ou pas (les détails d'implémentation se trouvent en annexe II). Dans notre (TestCase), nous avons enchainé plusieurs sections @Test. Ainsi, si une section @Test échoue, le TestCase ne s'arrête pas, mais continue sur les sections @Test suivantes (s'il y en a).

Pour l'application OnlineBankingSystem, nous avons développé neuf cas de tests dont les différents scripts se trouvent dans l'annexe II. Nous avons importé la librairie **ChromeDriver** de Selenium, pour pouvoir rouler nos différents tests sur le navigateur Google Chrome. L'appel du navigateur s'est fait de la manière suivante :

```
public class SeleniumJUnit {
              WebDriver driver;
       @Before
       public void beforMethod(){
              try {
                     System.setProperty("webdriver.chrome.driver", "C:\\Chemin du
repertoire\\chromedriver.exe");
                     driver = new ChromeDriver();
                     driver.manage().deleteAllCookies();
                     driver.manage().window().maximize();
                     driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);
                     driver.manage().timeouts().pageLoadTimeout(30,
TimeUnit.SECONDS);
                     driver.get("http://localhost:8009/");

              } catch (Exception e) {
                     e.printStackTrace();
              }}
```

Pour localiser les éléments sur la page web, nous avons utilisé les méthodes de recherche de la librairie WebDriver de Selenium. Nous avons le plus utilisé les recherches par ‘id’, par ‘name’, par ‘linkText’, et par ‘xpath’, tel que présenté sur la figure ci-dessous.

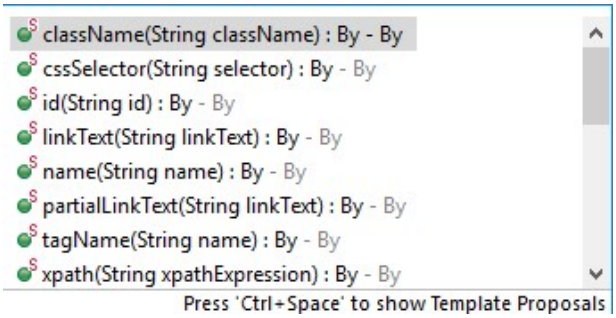


Figure 1.8

4.3 Resultat des cas de tests

Les scripts de test sont tous dans le document en Annexe II. Parmi les 9 cas de tests implémentés, on a 1 échec et 8 succès (voir figure 1.9). Le cas de test qui à échouer nous permet de constater une erreur dans l’application. En effet la validation des informations des différents champs nécessaire pour la création d’un compte utilisateur ne respecte pas les contraintes demandés.

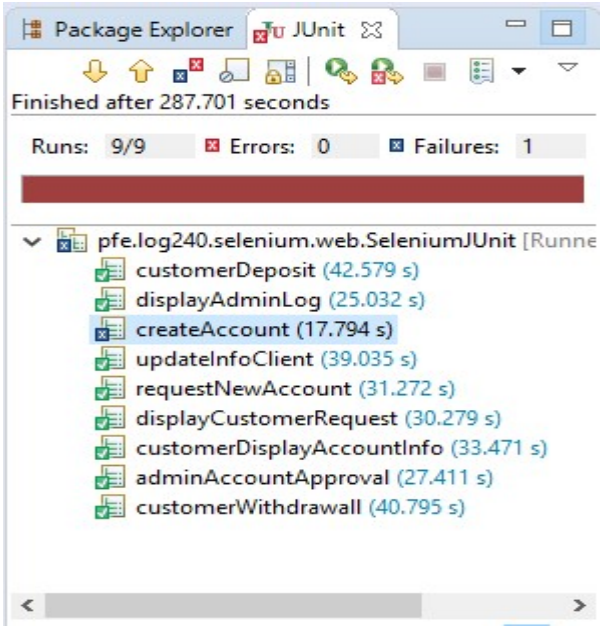


Figure 1.9

CONCLUSION

Les objectifs du projet ont été atteints malgré quelques difficultés rencontrées. Nous avons eu beaucoup de peine à trouver une application open-source répondant aux exigences du projet. Nous avons déployé une vingtaine d'applications, avant de trouver l'application OnlineBankingSystem, dont le langage et les cadriciel correspondent parfaitement aux exigences. La seconde difficulté a été de déployer l'application et de comprendre son fonctionnement, afin de créer un guide d'installation. Le guide d'installation décrit comment exécuter l'application à partir d'IntelIJ et comment créer un test automatique avec Selenium et JUnit. Plusieurs améliorations sont à faire dans l'application OnlineBankingSystem, pour compléter sa logique d'affaires. Ceci pourra être un bon exercice à ajouter dans les laboratoires du cours LOG240. Pour ce qui est de l'automatisation des tests, l'utilisation des librairies de Selenium et de JUnit, est réussie, malgré les difficultés techniques rencontrées.

LISTE DE RÉFÉRENCES

[1] http://www.test-recette.fr/recette/tests-fonctionnels-automatises/

[2] https://www.supinfo.com/articles/single/2585-tests-automatises

[3] https://www.supinfo.com/articles/single/2585-tests-automatises

[4] http://atatorus.developpez.com/tutoriels/java/test-application-web-avec-selenium/

[5] https://www.jmdoudoux.fr/java/dejae/chap011.htm

[6] https://makina-corpus.com/blog/metier/2014/webdriver-executer-vos-tests-sur-des-navigateurs-distants-en-parallele

[7] http://www.supinfo.com/articles/single/1692-selenium-ide

[8] https://fr.wikipedia.org/wiki/JUnit

[9] https://fr.wikipedia.org/wiki/Assurance_qualit%C3%A9_logicielle

ANNEXE I

ENONCE DU LABORATOIRE

– Énoncé Laboratoire 1 –

Familiarisation et prise en charge

Énoncé

OnlineBankingSystem est une application bancaire qui offre aux utilisateurs les fonctionnalités de création de compte, de dépôt et de retrait d'argent.

Remi, développeur Web, à conçu cette application pour répondre au besoin de l'entreprise Xbank ou il travaillait comme analyste programmeur.

Ayant trouvé un autre emploi, Remi vient de quitter son poste à Xbank qui se presse alors de recueillir les services de votre équipe pour assurer désormais la maintenance de **OnlineBankingSystem**.

L'état du logiciel **OnlineBankingSystem** n'est pas reluisant. Ses utilisateurs trouvent qu'il n'est pas assez convivial, qu'il exécute des opérations qui peuvent briser l'intégrité de la base de données, et que certaines de ses fonctionnalités ne s'exécutent même pas. Votre équipe a donc la mission de le prendre en charge le logiciel dans un environnement de maintenance réel pour pouvoir l'améliorer, conformément aux normes qui régissent le domaine.

Travail à faire –

Avant de penser à la révision du code d'**OnlineBankingSystem**, votre première tâche consiste à mettre en place votre environnement de travail et prendre connaissance de ce logiciel à travers cet environnement. De ce fait, vous devez effectuer les différentes étapes suivantes :

1. Assurez-vous que la MACHINE VIRTUELLE (VMware) qui vous a été fournie fonctionne. Pour ce faire, démarrez le logiciel émulateur de terminal « PUTTY » et utilisez l'adresse IP ou Host Name, le compte et le mot de passe fournis par les chargés de laboratoire.
2. Créez et configurez un compte GITLAB pour chaque membre de l'équipe, ainsi que le projet sur lequel seront régulièrement déposés les codes sources modifiés du logiciel **OnlineBankingSystem**.

3. Configurez l'environnement de développement du logiciel IntelliJ pour créer votre projet MAVEN et ajouter les fichiers du logiciel **OnlineBankingSystem** qui sont postés sur le Site web du cours. Puis connectez IntelliJ et Gitlab pour pouvoir déposer les codes ajoutés dans la branche correspondante.
4. Créez et configurez un compte sur le logiciel de gestion de projets TRAC pour chaque membre de l'équipe et le client. Puis configurez votre environnement comme indiqué en prenant en compte les taches, les versions du laboratoire et les dix étapes de la maintenance.

Note : Un guide technique qui contient toutes les explications nécessaires pour réaliser les tâches énumérées précédemment.

Travail à remettre –

En plus du travail pratique indiqué précédemment, qui doit être corrigé en présence de tous les membres de chaque équipe (correction interactive).

De plus, vous devez rédiger un rapport sur votre travail, utilisez le gabarit de rapport pour structurer le contenu. Une version électronique du rapport doit également être déposée via Moodle et le code source de projet Maven doit être déposé sur GitLab et être identifié par un « tag ».

ANNEXE II

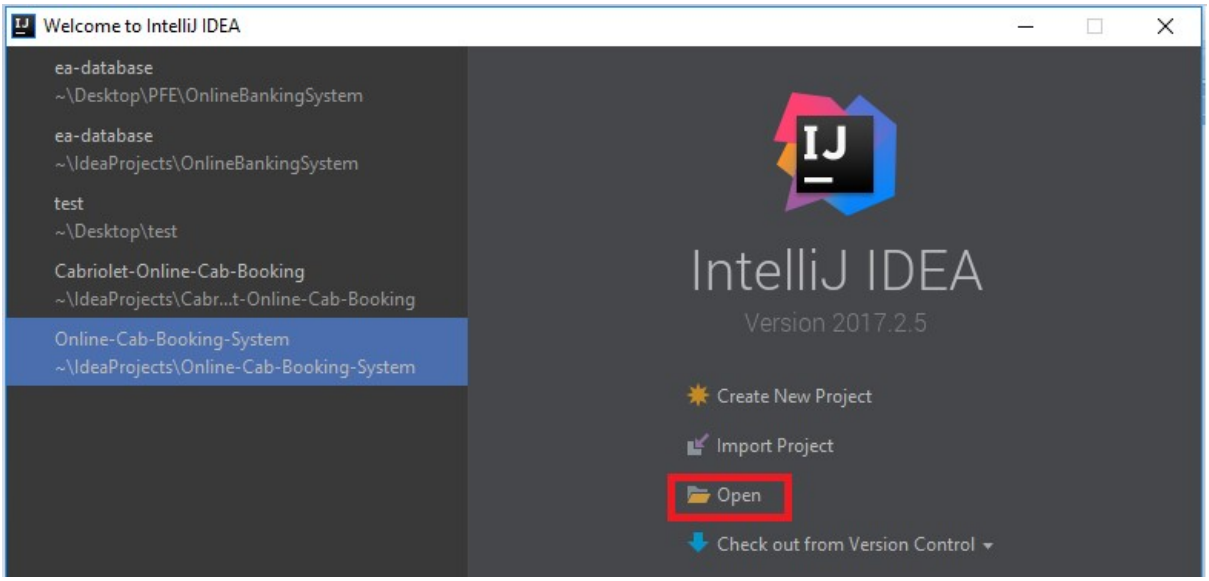
GUIDE D'INSTALLATION

IntelliJ est un environnement de développement intégré (IDE). Il contient un espace de travail de base et un système de plug-in extensible pour la personnalisation de l'environnement. IntelliJ est écrit en Java et son utilisation principale est de développer des applications Java, mais il peut également être utilisé pour développer des applications dans d'autres langages de programmation grâce à l'utilisation de plugins, y compris: Ada, ABAP, C, C ++, COBOL, Fortran, Haskell , JavaScript, Lasso, Lua, naturel, Perl, PHP, Prolog, Python, R, Ruby (y compris Ruby on Rails framework), Scala, Clojure, Groovy, Scheme, et Erlang.

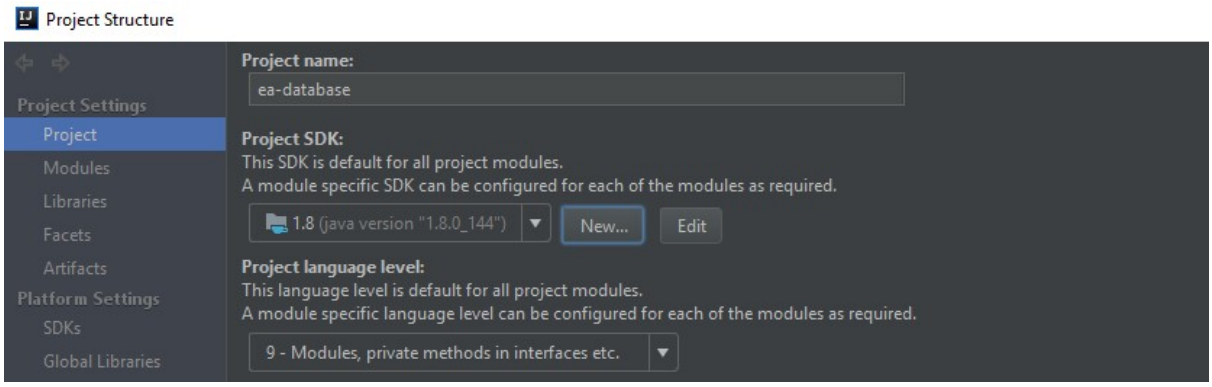
Afin que chaque membre de votre équipe puisse accéder au code de FinanceJ pour pouvoir modifier et l'enregistrer sur GitLab à chaque fois qu'un bout de code parait intéressant, il faut créer un projet Maven sur IntelliJ et le connecter avec le projet créé précédemment sur GitLab.

3.1- Création du projet Maven

a. Télécharger et dézipper le projet dans un dossier de votre choix
b. Démarrer IntelIJ IDEA
c. Cliquer sur **Open** et sélectionner le dossier du projet et cliquer sur **OK**



d. Après le chargement de l'application, Installer le bon SDK : Cliquer sur File puis sur project **structure** et installer le bon SDK (Java version 1.8). Java SDK est nécessaire pour déployer les applications Java. S'il est déjà installé, plus besoin de le faire. Si ce n'est pas installé, suivre le lien suivant pour l'installer :
http://www.oracle.com/technetwork/java/javase/downloads/index.html

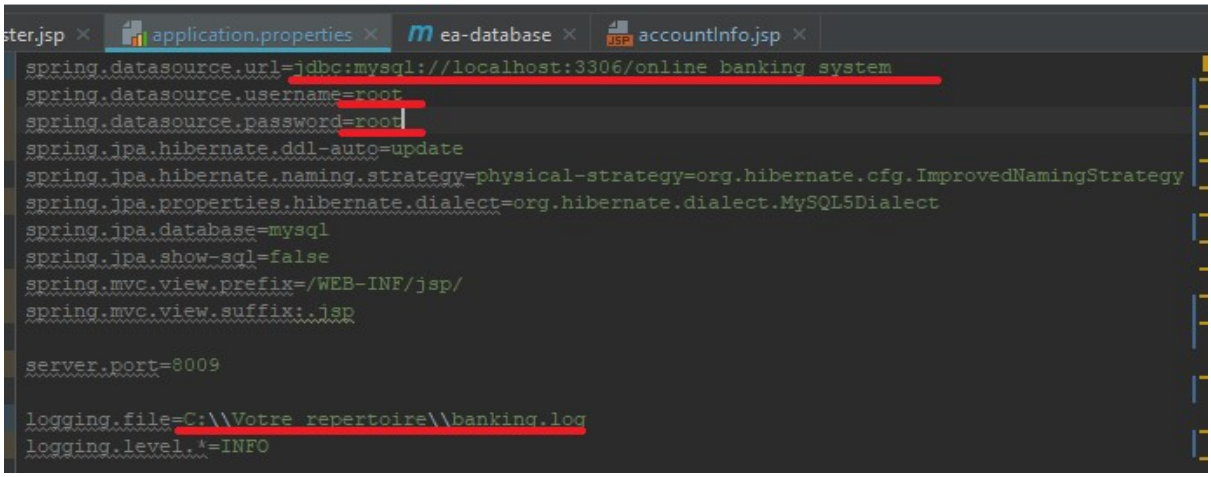


e. Cliquez sur le bouton « new » pour configurer le cadriciel du projet (Project SDK).

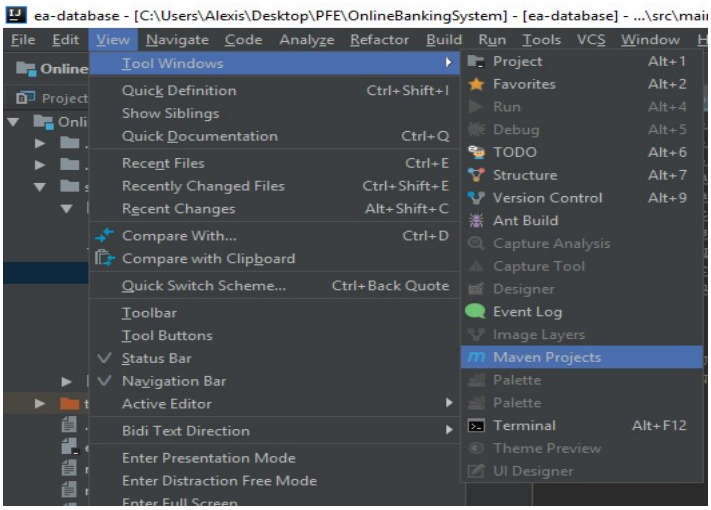
f. Sélectionnez le dossier : « C:/oracle/ jdk8_* » (ou « C:\Program Files\Java\jdk1.8.* » sur votre ordinateur personnel). Attention : « JavaSE-1.8 (jre1.8.*) » existe et n'est pas compatible, car il ne contient pas certaines commandes comme « javadoc ».
    a. Si jamais, le dossier est inexistant, vous pouvez télécharger et installer le JDK8 disponible sur le site d'Oracle.

g. Cliquer sur « OK »

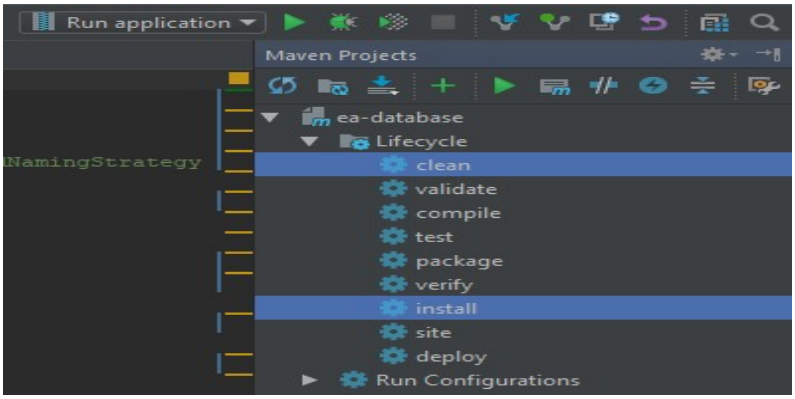
h. Charger le logging file : Répandre **src**, puis **Resources,** et double-cliquer pour ouvrir **application.properties.** changer le **logging.file** pour un repertoire valide changer les informations de connexion à la BD (url, Password, Username).
L'**url** permet spécifier l'emplavement de la BD. Le **password** et le **username** sont des information des informations d'authentification de la BD.
Le **logging.file**, permet de configurer les fichiers de log (enregistrer les différents évènements relatifs au fonctionnement de l'application).



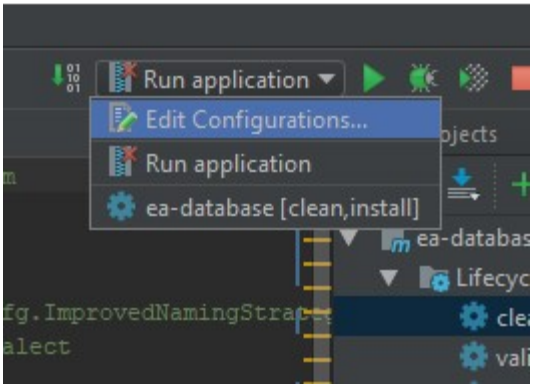
i. Créer un Maven Project : Cliquer sur **View** – **Tool Windows** – **Maven Projects.** OnlineBankingSystem est une application Maven

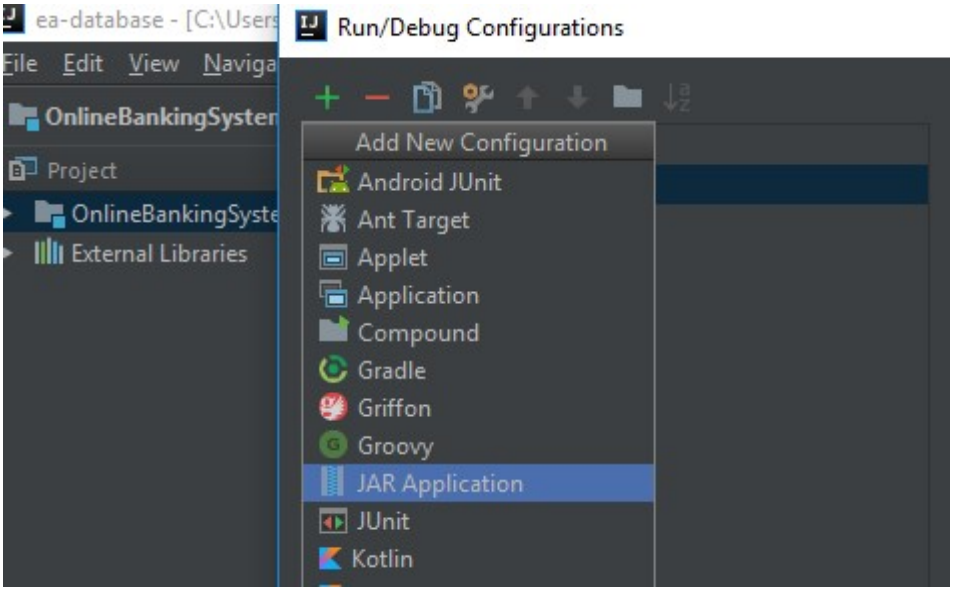


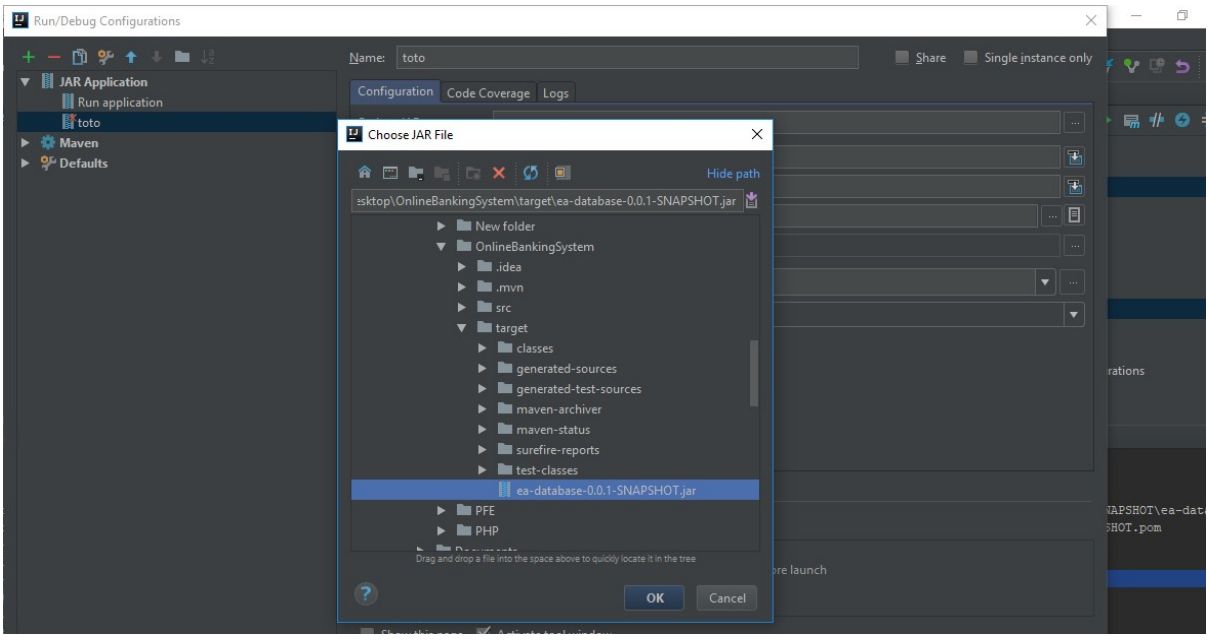
j. Dérouler **Lifecycle,** sélectionner simultanément **clean** et **install** puis cliquer sur **Run Maven Build.** Vous devez obtenir le message **Build Success** à la fin du build. Après cette étape, on peut rouler l'application à partir d'IntelIJ. Après chaque modification du code sourcre, il faut à nouveau lancer **Run Maven Build** pour obtenir un nouveau build et exécuter l'application pour tester les changements.



k. Pour exécuter l'application, Cliquer sur **Edit Configurations** puis cliquer sur la **croix verte** et sur **JAR application,** choisir l'emplacement du .jar de l'application, qui se trouve en principe dans le dossier de l'application, telle qu'indiqué sur les captures ci-dessous


Run application
Edit Configurations...
Run application
ea-database [clean,install]


ea-database - [C:\Users
File Edit View Naviga
OnlineBankingSyster
Project
OnlineBankingSyste
External Libraries
Run/Debug Configurations
Add New Configuration
Android JUnit
Ant Target
Applet
Application
Compound
Gradle
Griffon
Groovy
JAR Application
JUnit
Kotlin


Run/Debug Configurations
Name: toto
Share
Single instance only
JAR Application
Run application
toto
Maven
Defaults
Configuration
Code Coverage
Logs
Choose JAR File
Hide path
sktop\OnlineBankingSystem\target\ea-database-0.0.1-SNAPSHOT.jar
New folder
OnlineBankingSystem
.idea
.mvn
src
target
classes
generated-sources
generated-test-sources
maven-archiver
maven-status
surefire-reports
test-classes
ea-database-0.0.1-SNAPSHOT.jar
PFE
PHP
Drag and drop a file into the space above to quickly locate it in the tree
OK
Cancel

l. Lancer l'application. Après le chargement Saisir **localhost:8009** dans le navigateur.

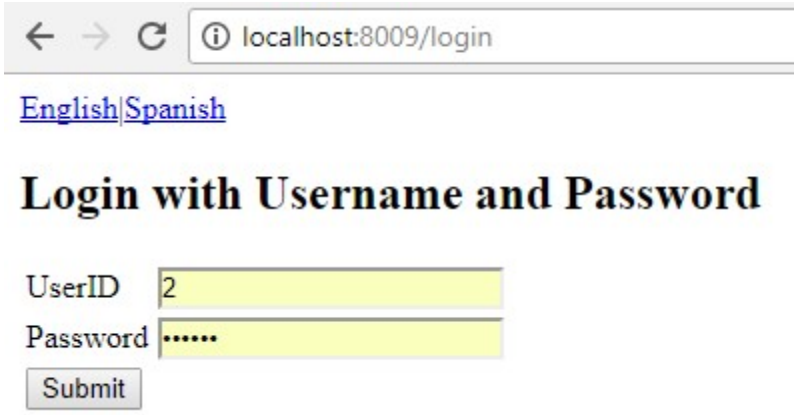
m. Créer un compte utilisateur en cliquant sur **Register Here**

n. Créer un autre compte utilisateur et lui accorder le privilège **administrateur** en exécutant le script sql suivant. **1** représente l'id de l'utilisateur créé

```
10    /*Move user from customer to staff */
11    delete from customer where username = 1;
12    insert into staff(username) values(1);
13    update user_roles set role = 'ROLE_ADMIN' where username = 1;
```

<u>Note</u> : Vous devez améliorer cette fonctionnalité.

o. Pour se connecter, entrer l'id de l'utilisateur (Par exemple 1) et le mot de passe. L'id s'incrémente automatiquement.



ANNEXE III

SCRIPTS DE TESTS

```
package pfe.log240.selenium.web;

import java.util.concurrent.TimeUnit;

import org.junit.After;
import org.junit.Before;
import org.junit.Test;
import junit.framework.Assert;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;



	public class SeleniumJUnit {

		WebDriver driver;

	@Before
	public void beforMethod(){

		try {
			System.setProperty("webdriver.chrome.driver",
"C:\\Users\\Alexis\\Desktop\\PFE\\chromedriver.exe");
			driver = new ChromeDriver();
			driver.manage().deleteAllCookies();
			driver.manage().window().maximize();
			driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);
			driver.manage().timeouts().pageLoadTimeout(30, TimeUnit.SECONDS);

			driver.get("http://localhost:8009/");

		} catch (Exception e) {
			e.printStackTrace();
		}
	}

	@After
	public void afterMethod(){

		driver.close();

	}
```

```
	@Test
	public void createAccount(){
		try {
			driver.findElement(By.linkText("Register Here")).click();
			Thread.sleep(3000);
			driver.findElement(By.id("firstName")).sendKeys("Selenium");
			driver.findElement(By.id("lastName")).sendKeys("Log240");
			driver.findElement(By.id("dateOfBirth")).sendKeys("20001212");
			driver.findElement(By.id("phone")).sendKeys("5142338597");
			driver.findElement(By.id("email")).sendKeys("log240@gmail.com");
			driver.findElement(By.id("street")).sendKeys("23 asert");
			driver.findElement(By.id("city")).sendKeys("montreal");
			driver.findElement(By.id("state")).sendKeys("QC");
			driver.findElement(By.id("zipCode")).sendKeys("h0r 2e8");
			driver.findElement(By.id("country")).sendKeys("canada");
			driver.findElement(By.id("password")).sendKeys("abc123");
			driver.findElement(By.id("btnAdd")).click();
			Assert.assertTrue(driver.getTitle().equals("Login"));
		} catch (InterruptedException e) {
			e.printStackTrace();}}
	@Test
	public void adminAccountApproval(){
		try {
			driver.findElement(By.linkText("Login Here")).click();
			Thread.sleep(3000);
			driver.findElement(By.name("username")).sendKeys("2");
			driver.findElement(By.name("password")).sendKeys("abc123");
			driver.findElement(By.name("submit")).click();
			Thread.sleep(3000);
			driver.findElement(By.linkText("Go TO Staff Page")).click();
			Thread.sleep(3000);
			driver.findElement(By.linkText("Approve")).click();
			Assert.assertTrue(driver.getTitle().equals("Home"));
		} catch (InterruptedException e) {
			e.printStackTrace();}}
	@Test
	public void updateInfoClient(){
		try {
			driver.findElement(By.linkText("Login Here")).click();
			Thread.sleep(3000);
			driver.findElement(By.name("username")).sendKeys("2");
			driver.findElement(By.name("password")).sendKeys("abc123");
			driver.findElement(By.name("submit")).click();
			Thread.sleep(3000);
			driver.findElement(By.linkText("Go TO Staff Page")).click();
			Thread.sleep(3000);
			driver.findElement(By.linkText("List Customers")).click();
			Thread.sleep(3000);
			driver.findElement(By.linkText("Update")).click();
			Thread.sleep(3000);
			driver.findElement(By.id("firstName")).clear();
			driver.findElement(By.id("firstName")).sendKeys("Totooooo");
			driver.findElement(By.id("btnAdd")).click();
			Assert.assertTrue(driver.getTitle().equals("List Of Customers"))
		} catch (InterruptedException e) {
			e.printStackTrace();
		}	}
```

```
	@Test
	public void displayAdminLog(){
		try {
			driver.findElement(By.linkText("Login Here")).click();
			Thread.sleep(3000);
			driver.findElement(By.name("username")).sendKeys("2");
			driver.findElement(By.name("password")).sendKeys("abc123");
			driver.findElement(By.name("submit")).click();
			Thread.sleep(3000);
			driver.findElement(By.linkText("Go TO Staff Page")).click();
			Thread.sleep(3000);
			driver.findElement(By.linkText("Get My Activities")).click();
			Assert.assertTrue(driver.getTitle().equals("Staff History"));
		} catch (InterruptedException e) {
			e.printStackTrace();}}

	@Test
	public void displayCustomerRequest(){
		try {
			driver.findElement(By.linkText("Login Here")).click();
			Thread.sleep(3000);
			driver.findElement(By.name("username")).sendKeys("2");
			driver.findElement(By.name("password")).sendKeys("abc123");
			driver.findElement(By.name("submit")).click();
			Thread.sleep(3000);
			driver.findElement(By.linkText("Go TO Staff Page")).click();
			Thread.sleep(3000);
			driver.findElement(By.linkText("List Customers")).click();
			Thread.sleep(3000);
			driver.findElement(By.className("show")).click();
			Assert.assertTrue(driver.getTitle().equals("List Of Customers"));

		} catch (InterruptedException e) {

			e.printStackTrace();
		}}

	@Test
	public void requestNewAccount(){
		try {
			driver.findElement(By.linkText("Login Here")).click();
			Thread.sleep(3000);
			driver.findElement(By.name("username")).sendKeys("1");
			driver.findElement(By.name("password")).sendKeys("abc123");
			driver.findElement(By.name("submit")).click();
			Thread.sleep(3000);
			driver.findElement(By.linkText("Go TO Customer Page")).click();
			Thread.sleep(3000);
			driver.findElement(By.linkText("Choose an Account")).click();
			Thread.sleep(3000);
			driver.findElement(By.linkText("Saving Account")).click();
			Assert.assertTrue(driver.getTitle().equals("Customer Page"));

		} catch (InterruptedException e) {

			e.printStackTrace();
		}
	}
```

```
	@Test
	public void customerDisplayAccountInfo(){
		try {
			driver.findElement(By.linkText("Login Here")).click();
			Thread.sleep(3000);
			driver.findElement(By.name("username")).sendKeys("1");
			driver.findElement(By.name("password")).sendKeys("abc123");
			driver.findElement(By.name("submit")).click();
			Thread.sleep(6000);
			driver.findElement(By.linkText("Go TO Customer Page")).click();
			Thread.sleep(3000);
			driver.findElement(By.linkText("Saving Account : 1")).click();
			Thread.sleep(3000);
			driver.findElement(By.id("info")).click();
			Assert.assertTrue(driver.getTitle().equals("Insert title here"));
		} catch (InterruptedException e) {
			e.printStackTrace();}}
	@Test
	public void customerWithdrawall(){
		try {
			driver.findElement(By.linkText("Login Here")).click();
			Thread.sleep(3000);
			driver.findElement(By.name("username")).sendKeys("1");
			driver.findElement(By.name("password")).sendKeys("abc123");
			driver.findElement(By.name("submit")).click();
			Thread.sleep(6000);
			driver.findElement(By.linkText("Go TO Customer Page")).click();
			Thread.sleep(3000);
			driver.findElement(By.linkText("Saving Account : 1")).click();
			Thread.sleep(3000);
			driver.findElement(By.xpath("//*[@id='amount']")).sendKeys("1000");
			Thread.sleep(3000);
		driver.findElement(By.xpath("//*[@id='container']/form[1]/input[4]")).click();
			Assert.assertTrue(driver.getTitle().equals("Customer Page"));
		} catch (InterruptedException e) {
			e.printStackTrace();
		}}
	@Test
	public void customerDeposit(){
		try {
			driver.findElement(By.linkText("Login Here")).click();
			Thread.sleep(3000);
			driver.findElement(By.name("username")).sendKeys("1");
			driver.findElement(By.name("password")).sendKeys("abc123");
			driver.findElement(By.name("submit")).click();
			Thread.sleep(6000);
			driver.findElement(By.linkText("Go TO Customer Page")).click();
			Thread.sleep(3000);
			driver.findElement(By.linkText("Saving Account : 1")).click();
			Thread.sleep(3000);
			driver.findElement(By.xpath("//*[@id='amount']")).sendKeys("100");
			Thread.sleep(3000);
		driver.findElement(By.xpath("//*[@id='container']/form[1]/input[4]")).click();
			Assert.assertTrue(driver.getTitle().equals("Customer Page"));
		} catch (InterruptedException e) {
			e.printStackTrace();}}}
```

APPENDICES <s'il y a lieu>

<Texte>