

Introduction

Le XXème siècle a été marqué par la révolution de l'information, et le développement des systèmes informatiques, en effet la concurrence dans le domaine de la technologie de l'information devient plus forte et plus difficile.

Les entreprises et les startups alors doivent ~~assurés~~ assurer la qualité des produits délivrés.

Construire et maintenir un logiciel de qualité est une tâche difficile, ~~les clients changent souvent leurs exigences et leurs projets.~~ (expl: les client deviennent de plus en plus exigeants d'où l'importance de livrer un produit bien qualifié et testé..) . Par conséquent, le test des logiciels est une phase obligatoire pour les projets en cours pour garantir la haute qualité des produits finis.

le métier de testeur est de confirmer que l'application répond bien au cahier de charge et la description du besoin prononcé par le Business Owner et ceci en repérant les bugs et anomalies tout au long le cycle de vie du produit à livrer . L'activité de test est pour découvrir les erreurs et les défauts de l'application le plutôt possible pour diminuer les coûts de maintenance et améliorer la qualité.

Pour ~~ce se~~ faire, l'équipe QA (Quality Assurance) ~~d'assurance qualité~~ doit étudier et choisir parmi plusieurs types de tests sur le terrain (unitaire, fonctionnel , manuel, automatisés ...)

l'équipe QA (Quality Assurance) ~~d'assurance qualité~~ doit effectuer des tests manuels ou automatisés appropriés.

Dans ce rapport, j'examinerai en détail le fonctionnement des tests automatisés, ainsi que l'intégration des rapports sur les outils d'intégration Continue.

Le but de ce projet est d'utiliser Robot Framework pour développer des scripts d'automatisation en utilisant Selenium2Library pour tester l'interface utilisateur graphique. J'expliquerai comment les utilisateurs peuvent personnaliser leurs scripts automatisés et comment Robot Framework exécute les tests automatisés.

En deuxième phase, j'exposerai le Workflow opté pour l'intégration des tests automatiques et les étapes faites pour la configuration des plugins sur Jenkins, Octane et RobotFramework.

Cadre général du projet

Introduction

Le présent chapitre a pour objectif de mettre le projet dans son cadre général à savoir l'entreprise accueillante. Par la suite, nous enchaînerons avec la description du projet en question ainsi que la problématique et la solution proposée et nous finissons par la méthodologie de travail opté.

Présentation de l'organisme d'accueil :

Sofrecom est une entreprise de conseil et d'ingénierie spécialisée dans le domaine des télécommunications.

Sofrecom développe depuis plus 50 ans un savoir-faire unique dans les métiers du numérique, ce qui en fait un leader mondial du conseil et de l'ingénierie télécom.

Reconnue pour sa capacité à analyser et anticiper les tendances numériques, Sofrecom accompagne le développement et les transformations des opérateurs télécoms, des gouvernements et des régulateurs en leur apportant conseil et solutions opérationnelles.

Sofrecom est une filiale du Groupe Orange. Son expérience des marchés matures et des économies émergentes, conjuguée à sa solide connaissance des évolutions structurantes du marché des télécommunications en font un partenaire incontournable pour les opérateurs, gouvernements et investisseurs internationaux. Plus de 200 acteurs majeurs, dans plus de 100 pays, font confiance à Sofrecom dans la conduite de leurs projets stratégiques : transformation et optimisation, modernisation technologique, innovation et développement. Sofrecom est une entreprise riche de sa diversité : 1 400 consultants et experts, 30 nationalités, 11 implantations à travers le monde. Sofrecom est un puissant réseau de savoir-faire qui relie ses équipes aux experts du Groupe Orange, à ses partenaires industriels, ses partenaires locaux et à ses clients.

Présentation de Projet MVA :

~~MVA (Media Vocal Automated) est un outil visant à améliorer la satisfaction client. Cette application représente un service existant permettant de lancer des campagnes d'enquêtes téléphoniques de satisfaction automatisées.~~

~~Les clients ayant appelé un agent d'un call center reçoivent un message vocal leur demandant de noter leur niveau de satisfaction. Tous les clients non satisfaits sont ensuite rappelés par des agents.~~

Pour répondre aux enjeux majeurs d'amélioration de l'écoute clients, MVA (Message Vocal Automatisé) permet de sonder automatiquement le niveau de satisfaction d'un client ayant appelé un téléconseiller Orange. Des mesures spécifiques peuvent être ainsi prises auprès des non-satisfaits (exemple : rappel ciblé).

MVA permet un appel sortant automatisé pour contacter les clients Orange quelques heures après leur appel pour connaître leur niveau de satisfaction.

Les clients très insatisfait sont rappelés par un conseiller téléphonique qui catégorise l'insatisfaction en utilisant le service ClickToCall.

MVA offre plusieurs options de configuration comme la possibilité de :

- Programmer la plage horaire pour les appels MVA vers les clients
- Exclure des clients pour les appels MVA (VIP, faux numéro, etc ..)
- Filtrer de façon précise des clients pour les appels MVA

- Gérer les utilisateurs de MVA

Présentation des intervenant de projet :

Product Owner	Chef de projet / Scrum Master	Équipe de développement	Équipe test et validation	Intégrateur
Laurent ANDRE	Wided LIMAME	Omran SOUIBGUI Khoulood ENNAOUI	Aymen MHADHEB Ikram SASSI	Aymen DRIRA Youssef BOURBIA

Problématique :

Le test est une phase primordiale qui doit exister dans le processus de développement des logiciels pour gagner en terme de coût et de ressources. Il est donc crucial de s'assurer que le produit est au norme et répond au besoin , pour ceci on peut citer plusieurs type de tests (Fonctionnels, Opérationnels, Régression...) , et Deux méthodes pour les appliquer .

Au sein de l'équipe on a opté à automatiser les tests de Non régression (TNR) et concentrer les tests manuels sur les tests d'évolutions.

Dans ce contexte, Ma principale tâche est de rédiger des scénarios de tests et les les automatiser et ensuite partager les rapports de tests tout en étant en phase avec le reste de l'équipe .

les tests fonctionnels sont mais lorsque cette phase se fait manuellement, va engendrer une perte. on a opté décidé dans ce stage d'automatiser le processus de test pour assurer la production d'une application de qualité et minimiser le risque d'erreurs.

Méthodologie du travail

La méthodologie de travail est parmi les nécessités vitales pour garantir les résultats attendus et minimiser les risques d'un résultat indésirable, c'est pour cela l'équipe de projet MVA chez Sofrecom Tunisie suit une méthodologie agile de modélisation bien connue dans le domaine de développement des logiciels informatiques, C'est Scrum.

Dans ce cadre j'ai assisté au mêlée quotidienne dont l'objectif est d'évaluer l'avancement du travail, identifier les obstacles nuisant à la progression, garder l'équipe concentrée sur l'objectif du sprint, améliorer l'esprit d'équipe, et permettre une communication objective sur l'avancement

D'un autre côté Scrum donne un cadre pour l'organisation de ces réunions d'une durée limitée à 15 minutes, tout le monde doit être debout et 3 questions posées à chaque membre de l'équipe :

- Qu'est-ce que j'ai fait hier ?
- Qu'est-ce que je vais faire aujourd'hui ?
- Qu'est-ce que j'ai comme problèmes ?

Définition de Test :

Le test est l'exécution ou l'évaluation d'un système ou d'un composant par des moyens automatiques ou manuels, pour vérifier qu'il répond à ses spécifications ou identifier les différences entre les résultats attendus et les résultats obtenus, d'en augmenter la qualité.

Un test ressemble à une expérience scientifique. Il examine une hypothèse exprimée en fonction de trois éléments : les données en entrée, l'objet à tester et les observations attendues. Cet examen est effectué sous conditions contrôlées pour pouvoir tirer des conclusions et, dans l'idéal, être reproduit.

IEEE (Standard Glossary of Software Engineering Terminology)

Activités de test dans Scrum

Les testeurs suivent les activités au cours des différentes étapes de Scrum :

- **Planification des sprints** : Dans la planification des sprints, un testeur doit sélectionner une histoire d'utilisateur dans le carnet de produit à tester. En tant que testeur, il doit décider combien d'heures (estimation de l'effort) il faut pour terminer les tests pour chacune des user stories sélectionnées. Il doit aussi savoir quels sont les objectifs du sprint et il contribue au processus de priorisation
- **Sprint** : Soutenir et accompagner les développeurs lors des tests unitaires et tester l'histoire de l'utilisateur lorsque les développeurs ont terminé.
L'exécution du test est effectuée dans un laboratoire où le testeur et le développeur travaillent main dans la main.
Les défauts sont consignés dans l'outil de gestion des défauts, qui font l'objet d'un suivi quotidien.
Les défauts peuvent être conférés et analysés lors de la réunion Scrum. Ils doivent être testés à nouveau dès que le problème est résolu et déployé.

Le testeur assiste aussi à toutes les réunions quotidiennes du stand-up pour parler et il peut apporter n'importe quel élément de backlog qui ne peut pas être complété dans le sprint actuel et le mettre au prochain sprint.

Le testeur est responsable du développement des scripts d'automatisation. Il planifie les tests d'automatisation avec le système d'intégration continue (CI).

L'automatisation reçoit toute l'importance en raison des délais de livraison courts.

L'automatisation des tests peut être réalisée en utilisant divers outils open source ou payants disponibles sur le marché. Cela s'avère efficace pour garantir que tout ce qui doit être testé est couvert.

Une couverture de test suffisante peut être obtenue par une communication étroite avec l'équipe.

Examiner les résultats de l'automatisation des CI et envoyer des rapports aux parties prenantes.

Exécution de tests non fonctionnels pour les user stories approuvées.

Coordonner avec le client et le propriétaire du produit pour définir les critères d'acceptation des tests d'acceptation.

À la fin du sprint, le testeur effectue également des tests d'acceptation (UAT) dans certains cas et confirme que les tests sont complets pour le sprint actuel.

- Rétrospective Sprint : Le testeur déterminera ce qui ne va pas et ce qui s'est bien passé dans le sprint actuel, il identifie les leçons apprises et les best practice.
- Rapport de test : Les rapports de métrique Scrum Test offrent aux parties prenantes une transparence et une visibilité sur le projet. Les métriques rapportées permettent à une équipe d'analyser leurs progrès et de planifier leur future stratégie pour améliorer le produit.

Conclusion

I. Introduction

La réalisation de ce projet nécessite une étude des besoins, Pour ce faire nous entamerons en premier lieu les besoins de l'utilisateur en termes d'exigences fonctionnelles et non fonctionnelles du système, tout en empruntant du langage de modélisation UML le diagramme de cas d'utilisation ainsi que le diagramme de séquence.

II. Spécifications des besoins

L'analyse des besoins nous permet de comprendre les besoins de l'utilisateur pour pouvoir identifier les différentes fonctions de l'application. En premier lieu nous allons identifier les acteurs, Puis nous décrivons ces besoins qui se divisent en besoin fonctionnels et non fonctionnels.

2.1 Acteurs

pour l'exécution des tests sur effectué sur la solution MVA , Nous distinguons deux types rôles : Membre de l'équipe Qualif, Contributeur sur la solution (développeur , intégrateur ..)

- ~~● Intégrateur MVA : C'est l'acteur principal du système qui est chargé de l'administration de la plateforme et de la configuration des scénarios de tests.~~
- ~~● Membre d'équipe MVA: C'est l'acteur secondaire du système, il permet uniquement d'exécuter les jobs et de consulter les rapports générés.~~
- Membre de l'équipe Qualif: C'est l'acteur principal qui réalise toutes les fonctionnalités de notre solution.
- Contributeur sur la solution: C'est l'acteur secondaire du système, il permet uniquement d'exécuter les jobs et de consulter les rapports générés.

2.2 Besoins Fonctionnels

Ayant identifié les acteurs de notre solution, nous passons **par** dégager les fonctionnalités que l'application devrait satisfaire. Les besoins fonctionnels se résument dans les points suivant :

- L'automatisation des *tests de non régression ou tests fonctionnels*
- La génération et la consultation des rapports de tests

2.3 Besoins non Fonctionnels

Outre les besoins fonctionnels, le système doit assurer les besoins non fonctionnels qui constituent l'ensemble des contraintes d'amélioration de la qualité de ce dernier. Il doit donc répondre aux critères suivants :

- Evolutivité : le système peut intégrer d'autres modules pour étendre ses fonctionnalités.
- Confidentialité : le système doit être capable de garantir un accès sécurisé à l'application tout en respectant les droits et les permissions de chaque utilisateur.
- **Intégration continue** : le système doit assurer l'intégration continue afin de partager les rapport avec le reste d'équipe.

III. Diagramme de cas d'utilisation

Le diagramme de cas d'utilisation permet de représenter visuellement une séquence d'actions réalisées par un système, représenté par une boîte rectangulaire, produisant un résultat sur un acteur, appelé acteur principal, et ceci indépendamment de son fonctionnement interne.

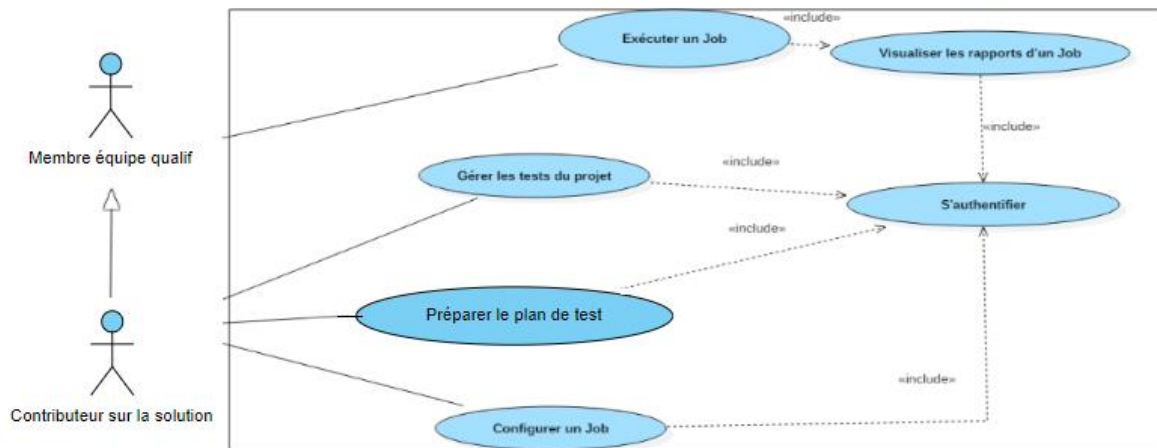


Figure 3. 1 : Diagramme de cas d'utilisation global

Le système offre à l'acteur cinq possibilités :

- **Préparer le plan de test:** cette tâche permet la rédaction de cahier de test ainsi que la compréhension du besoin décrit sur ALM Octane.
- *Gérer les tests du projet. Cette tâche comprend l'ajout, la modification, la suppression et la consultation d'une liste de tests.*
- ~~Gérer les comptes des utilisateurs. Ce cas permet d'identifier qui peut avoir un accès au système et de définir leurs rôles.~~
- *Configurer un Job : Cette tâche permet de spécifier le mode de configuration d'un Job.*
- *Exécuter un Job. Ce cas permet à l'acteur d'exécuter les cas de test sur le job*
- *Visualiser les rapports d'un job. L'acteur consulte les rapports générés suite à une exécution d'un Job*

IV. Diagramme de séquence système

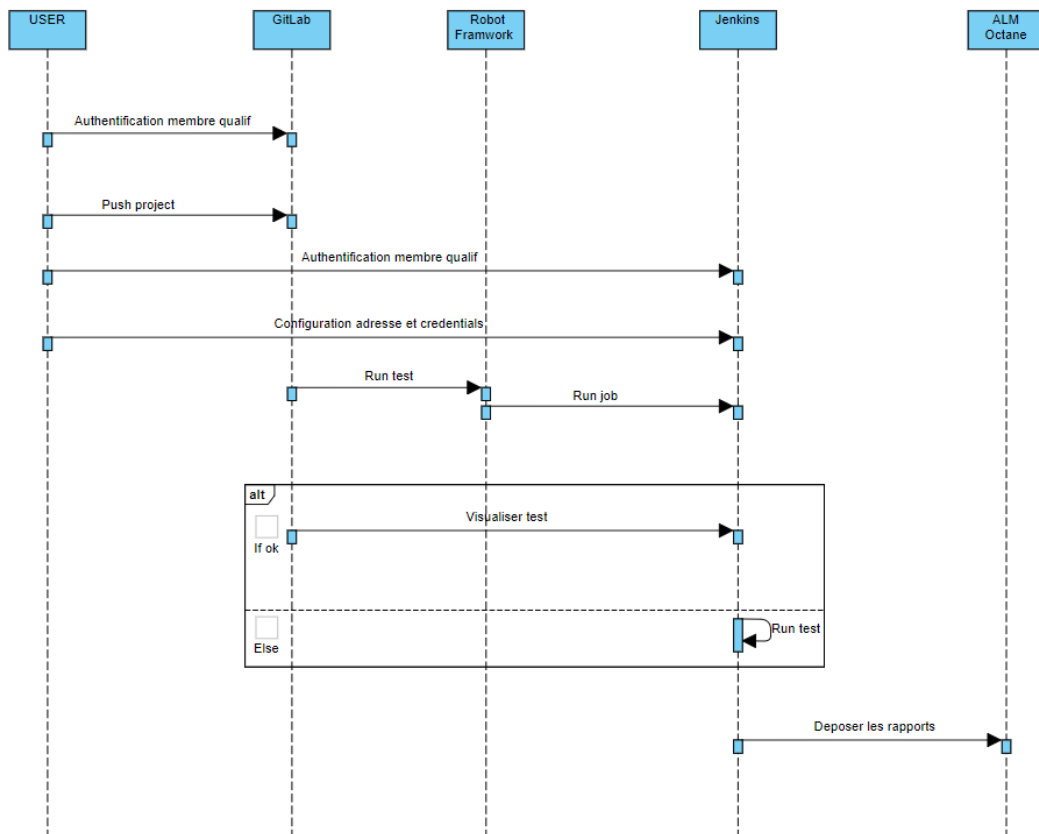


Figure 3. 6 : Diagramme de séquence Système

Le diagramme de séquence permet de représenter le déroulement de système dans le temps. En effet à travers ce diagramme nous pouvons déterminer les interactions entre les objets du système pour mieux comprendre les fonctionnalités internes du système.

D'après le schéma présenté par la figure 3.6. Diagramme de séquence système, l'intégrateur développe les scénarios de tests en utilisant Robot Framework, une fois que cette action effectuée, l'intégrateur s'authentifie au repository local de type Git, sur Gitlab, ajoute le projet contenant tous les scénarios, puis il fait la sauvegarde.

Par la suite, l'intégrateur s'authentifie sur Jenkins et configure l'accès au repository Git en ajoutant l'adresse et les credentials.

Jenkins va permettre le lancement des tests automatisés qui vont être exécutés par le plugin Robot Framework.

L'utilisateur lance les tests selon le mode d'exécution : soit en mode campagne unitaire soit en mode scénario spécifié.

Le résultat des tests sera visualisé sur Jenkins au niveau des rapports générés.

Dans le cas où le résultat du test est non concluant, l'utilisateur devra dérouler à nouveau les tests jusqu'à atteindre le pourcentage toléré.

v. Conclusion

Dans ce chapitre, nous avons explicité les différents besoins de notre étude et mise en place, nous avons présenté les diagrammes de cas d'utilisation afin d'exposer les fonctionnalités dont la solution est capable de fournir.

De même, nous avons élaboré un diagramme de séquence système pour expliquer les interactions entre les différents systèmes intervenants dans la solution.

Nous sommes prêts à entamer dans le chapitre suivant, la partie réalisation de notre projet.

Chapitre réalisation:

Introduction:

Dans ce chapitre nous allons parler de partie technique de stage dont on trouve la partie automatisation des tests dont on vas expliqué les différentes étapes suivies pour arriver à

automatiser les tests et de l'autre côté nous allons parler de l'intégration continue qui assure la partie devops de ce projet.

1. Environnement logiciel

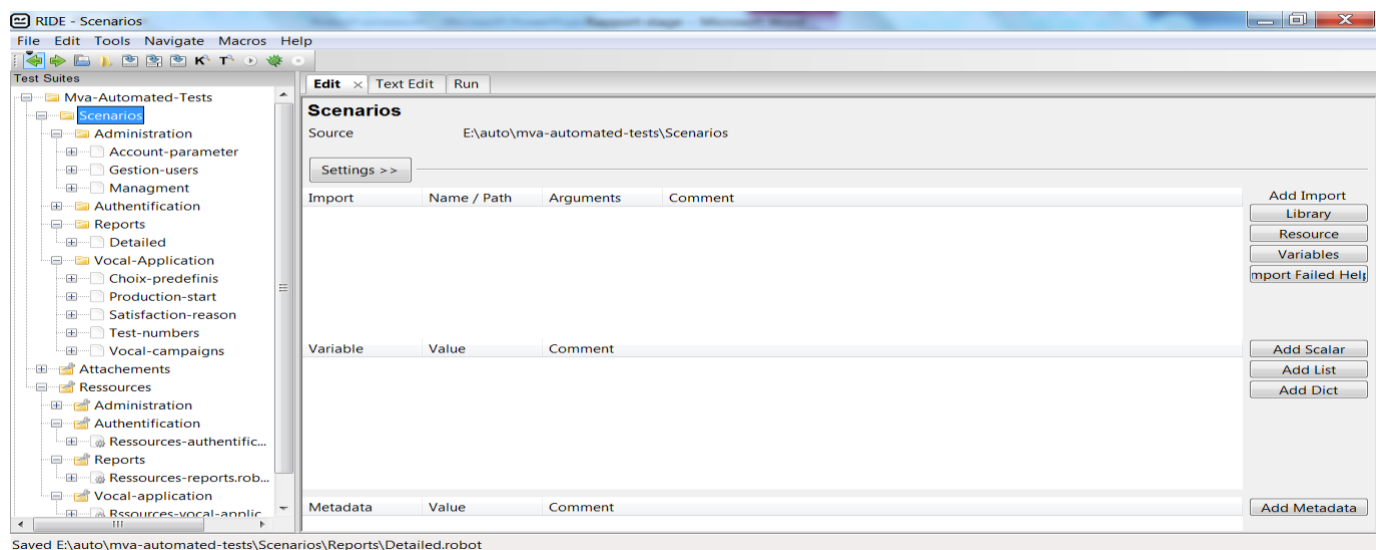
Robot Framework :

Robot Framework est une infrastructure d'automatisation de test open source basé sur python pour les tests d'acceptation et le développement piloté par les tests d'acceptation. Il suit différents styles de scénario de test - axé sur les mots clés, basé sur le comportement et basé sur les données pour l'écriture de scénarios de test. Cette fonctionnalité le rend très facile à comprendre. Les scénarios de test sont écrits en utilisant le style de mot clé dans un format tabulaire.

Robot Framework fournit un bon support pour les bibliothèques externes, des outils open source pouvant être utilisés pour l'automatisation. La bibliothèque la plus populaire utilisée avec Robot Framework est la bibliothèque Selenium utilisée pour le développement Web et les tests d'interface utilisateur. Robot Framework peut être lié avec Jenkins via un pipeline pour assurer la continuité.

Ride

Ride est l'IDE officiel de Robot Framework, l'interface graphique de RIDE est implémentée à l'aide wxPython (wxPython est une implémentation libre en Python de l'interface de programmation wxWidgets). Il permet de lancer directement les tests sans passer par la ligne de commande.



Jenkins

Jenkins est un serveur d'automatisation Java open source qui offre un moyen simple de configurer un pipeline d'intégration et de distribution continues (CI / CD).

Il est aussi un ordonnanceur qui va lancer des tâches (jobs) qui peuvent être déclenchées automatiquement. Les tâches conservent les résultats d'exécution et peuvent émettre un reporting, un mail, ...

GitLab

GitLab est une plate-forme de développement collaborative open source. Il offre une solution de stockage en ligne et de développement collaboratif de projets logiciels de grande envergure. Le référentiel comprend un contrôle de version pour activer et prendre en charge différentes chaînes et versions de développement, permettant aux utilisateurs d'inspecter le code précédent et de le restaurer en cas de problèmes imprévus.

ALM Octane

ALM Octane est une plate-forme Web de gestion du cycle de vie des applications qui permet aux équipes de collaborer facilement, de gérer le pipeline de distribution de produits et de visualiser l'impact des modifications.

2. Automatisation des tests

L'automatisation de test permet de jouer à volonté des tests de non-régression à la suite de la livraison d'une nouvelle version de produit. https://fr.wikipedia.org/wiki/Automatisation_de_test

Pour l'automatisation des applications web on utilise principalement la bibliothèque selenium2Library qui exécute les tests sur une véritable instance de navigateur.

2.3. Enchaînement de travail:

ce stage m'a permis d'automatiser une quarantaine de cas de test, pour se faire il faut suivre les étapes suivante :

préparer le plan de test selon les spécifications.

éviter de créer des tests exhaustifs

tester manuellement les US en variant les browsers (ie , chrome , ff)

écrire les anomalies identifiées sur octane

identifier les tests manuels et automatisable

automatiser les cas de tests en ajoutant des tags pour chaque cas de tests

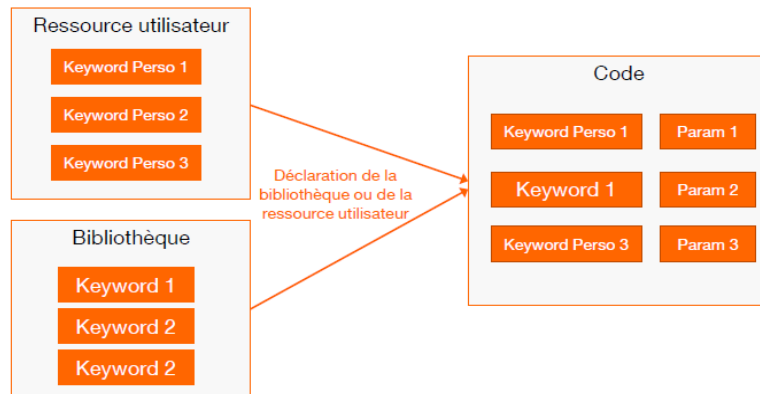
2.1. Les bibliothèques utilisé

Les bibliothèques de test fournissent les fonctionnalités de test réelles qui se base sur des mots-clés. Il existe plusieurs bibliothèques standards intégrées au framework, et de nombreuses bibliothèques externes développées séparément peuvent être installées en fonction des besoins.

- **BuiltIn** : la bibliothèque standard de Robot Framework qui fournit souvent un ensemble de mots-clés génériques. Elle est importée automatiquement et donc toujours disponible. Les mots clés fournis peuvent être utilisés pour des vérifications (par exemple *Should Be Equal*, *Should Contain*)
- **Selenium2Library** : Une bibliothèque externe de tests web qui exécute des tests dans une véritable instance de navigateur et qui fonctionne dans la plupart des navigateurs.

Robot Framework s'appuie sur l'instanciation de keywords comme la figure ... le montre dont on a deux types :

- Keywords natifs : Ils sont des fonctions prédéfinis dans les bibliothèques utilisés par Robot Framework.
- Keywords créés par l'utilisateur : Ils sont des fonctions personnalisés par le testeur construit en utilisant les keywords natifs.



2.2.Architecture logique de scripts de test:

La figure ... représente l'architecture opté pour l'automatisation des tests de l'application MVA qui décompose en:

-**Scenarios** est la couche qui englobe les :

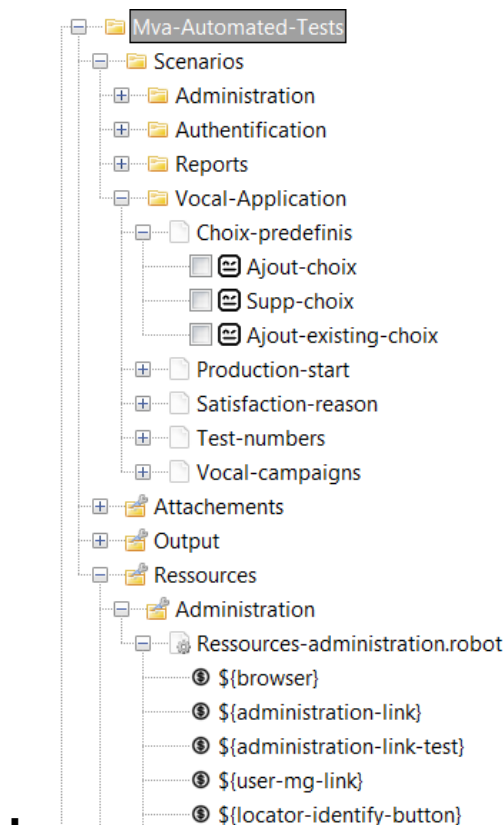
- Test Suite : ensemble de cas de tests qui regroupe le même volet fonctionnel par exemple.
- Test case : les étapes à dérouler pour vérifier un résultat attendu.

-**Attachment** est la couche dont on trouve les fichiers et/ou les images utilisé par l'application.

-**Output** est la couche dont on enregistre les rapport de test généré par Robot Framework


-**Ressources** c'est la couche dont on trouve des fichiers de type ROBOT qui englobe:

- Variables: ce sont les variables utilisé par les tests cases et les keywords .
- Keywords : keyword développé par l'utilisateur, et qui utilise des keywords de haut niveau (les keywords de bibliothèque natifs)



Exemple de plan de test :

la figure .. représente un exemple de plan de test du cas “Authentification-admin” dont on définit la description de cas de test, les étapes à suivre et le résultat attendu

					
Path	Test Name	Description	Step	Step Description	Expected Results
	CT02. Authentification à l'application MVA	ACTEUR Un identifiant de type profil "Admin"	Step1	Entrer l'url de l'application web MVA	La page d'authentification s'affiche
		EVENEMENT DECLENCHEUR L'acteur veut s'authentifier à l'application MVA	Step2	Entrer un bon login/mot de passe et cliquer sur "connecter"	Identification OK
		DESCRIPTION CAS TEST Cette fonctionnalité permet de s'authentifier à l'application MVA	Step3	Entrer un faux login/mot de passe et cliquer sur "OK"	Identification KO
		PRE-CONDITIONS - L'acteur possède d'un compte MVA	Step4	Déconnecter de MVA	L'utilisateur est déconnecté de l'application MVA
			Step5	Quitter l'application MVA	L'application MVA est fermé
		RESULTATS ATTENDUS a. Identification OK à l'application MVA ==> Le champ "login" et le champ "mot de passe" sont rempli avec des valeurs existe dans la base MVA b. Identification KO à l'application MVA c. L'utilisateur n'a accès qu'au dossier en question dépendamment du profil de l'utilisateur connecté	Step6	Re-ouvrir l'application MVA (Entrer l'url de l'application web MVA)	L'application MVA est ouvert
			Step7	Vérifier que le champ "Identifiant" est rempli avec la valeur du dernier login connecté	Le champ "login" est rempli avec la valeur du dernier login connecté
			Step8	Vérifier que l'utilisateur n'a accès qu'au fonctionnalités adéquates	L'utilisateur n'a accès qu'au dossier en question
			Step9	Attendre sans rien faire pendant le temps d'inactivité (en mode Online ou hors ligne)	L'utilisateur est déconnecté de l'application MVA automatiquement (un message est affiché)

Exemple de lancement de test automatisé

Les tests peuvent être lancés soit via Ride directement soit via la commande « *pybot* » comme décrit ci-dessous

```
> pybot --variable Browser:GC --variable Bibliotheque:LibSelenium2Library  
"Authentication-admin.robot"
```

Il est possible de lancer les tests par « tag » à l'aide de « *-i --include tag ** » ou de spécifier le chemin vers les fichiers résultats on utilisant « *-o --output file* » « *-l --log file* » « *-r --report file* »

On peut aussi ré-exécuter les tests « failed » à partir d'un autre rapport d'exécution à l'aide de « *-R --rerunfailed output* »

Exemple de rapport de test de Robot Framework

suite à l'automatisation du cas de test "authentication-admin" et tout les cas de tests de "campagne vocal" nous avons illustré les bugs de non apparition de message d'erreurs lors d'authentification avec un faux login ou mot de passe comme montré dans la figure .. d'exemple de rapport généré

Mva-Automated-Tests Report

Generated
20190820 16:06:15 UTC+02:00
12 seconds ago

Summary Information

Status: **1 critical test failed**
Start Time: 20190820 16:05:09.324
End Time: 20190820 16:06:15.184
Elapsed Time: 00:01:05.860
Log File: [log.html](#)

Test Statistics

Total Statistics	Total	Pass	Fail	Elapsed	Pass / Fail
Critical Tests	4	3	1	00:01:05	<div><div></div></div>
All Tests	4	3	1	00:01:05	<div><div></div></div>

Statistics by Tag	Total	Pass	Fail	Elapsed	Pass / Fail
No Tags					<div><div></div></div>

Statistics by Suite	Total	Pass	Fail	Elapsed	Pass / Fail
Mva-Automated-Tests	4	3	1	00:01:06	<div><div></div></div>
Mva-Automated-Tests . Scenarios	4	3	1	00:01:06	<div><div></div></div>
Mva-Automated-Tests . Scenarios . Authentication	1	0	1	00:00:10	<div><div></div></div>
Mva-Automated-Tests . Scenarios . Authentication . Authentication-admin	1	0	1	00:00:10	<div><div></div></div>
Mva-Automated-Tests . Scenarios . Vocal-Application	3	3	0	00:00:56	<div><div></div></div>
Mva-Automated-Tests . Scenarios . Vocal-Application . Vocal-campaigns	3	3	0	00:00:56	<div><div></div></div>

Nous cliquons sur la ligne du test FAIL. Nous renvoie vers le fichier log.html, incluant un screenshot, ici cela donne un journal de test en échec montré par la figure qui détaille l'existence de bug

SUITE Authentication	00:00:00.000	REPORT
Full Name: Mva-Automated-Tests.Scenarios.Authentication		
Source: E:\auto\mva-automated-tests\Scenarios\Authentication		
Start / End / Elapsed: 20190820 16:05:09.379 / 20190820 16:05:19.223 / 00:00:09.844		
Status: 1 critical test, 0 passed, 1 failed 1 test total, 0 passed, 1 failed		
SUITE Authentication-admin	00:00:09.841	
Full Name: Mva-Automated-Tests.Scenarios.Authentication.Authentication-admin		
Source: E:\auto\mva-automated-tests\Scenarios\Authentication\Authentication-admin.txt		
Start / End / Elapsed: 20190820 16:05:09.382 / 20190820 16:05:19.223 / 00:00:09.841		
Status: 1 critical test, 0 passed, 1 failed 1 test total, 0 passed, 1 failed		
TEST authentication-admin-wrong-pwd	00:00:08.879	
Full Name: Mva-Automated-Tests.Scenarios.Authentication.Authentication-admin.authentication-admin-wrong-pwd		
Start / End / Elapsed: 20190820 16:05:10.344 / 20190820 16:05:19.223 / 00:00:08.879		
Status: FAIL (critical)		
Message: Page should have contained text 'votre login/mot de passe est incorrect' but did not.		
KEYWORD Resources-authentication .Authentication-mva \${url}, \${browser}, \${password-incorrect}, \${login-admin}	00:00:07.897	
KEYWORD Selenium2Library .Page Should Contain \${message-erreur-lg-mp-incorrect}	00:00:00.981	
Documentation: Verifies that current page contains text.		
Start / End / Elapsed: 20190820 16:05:18.242 / 20190820 16:05:19.223 / 00:00:00.981		
KEYWORD Selenium2Library .Capture Page Screenshot	00:00:00.249	
16:05:19.223 FAIL Page should have contained text 'votre login/mot de passe est incorrect' but did not.		

3.Intégration continue

L'intégration continue (CI) est une pratique de DevOps dans laquelle les membres de l'équipe commit régulièrement les modifications de code dans le repository de contrôle de version, après que les builds et les tests automatisés sont exécutés. La livraison continue (CD) est une série de pratiques dans lesquelles les modifications de code sont automatiquement construites, testées et déployées en production.

On a opté à utiliser l'outil d'intégration continue [Jenkins](#) vu les [plugins](#) mise à disposition par sa communauté

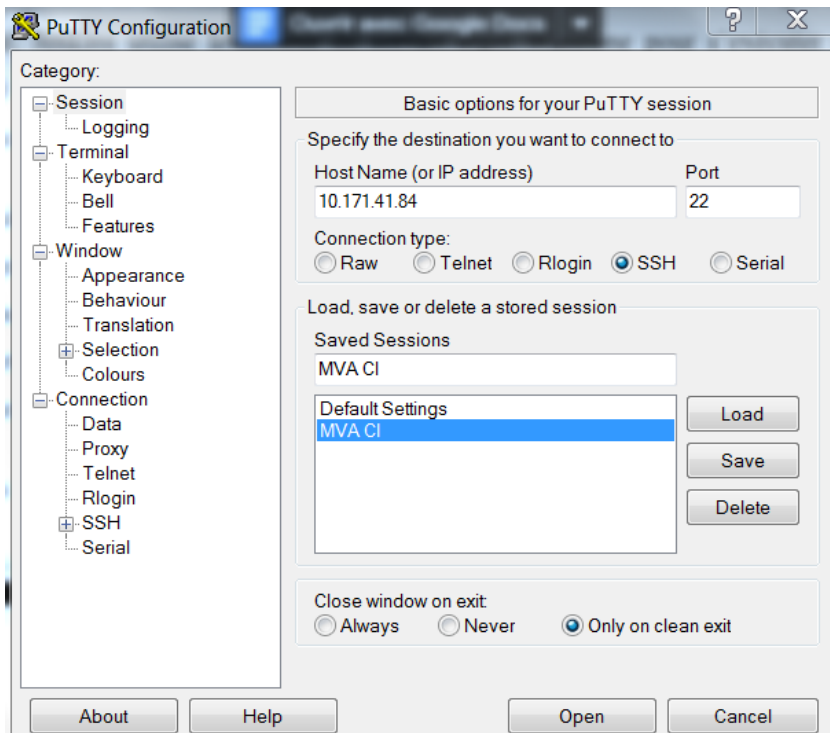
-Étape et procédure réalisée :

Ma mission était de mettre en place la 2eme partie de la chaîne de l'intégration continue, qui consistait à l'automatisation des tests et partage du rapport sur Jenkins et Octane .

1- Configuration et installation :

Pour la mise en place de l'IC :

installation Jenkins : nous avons installé Jenkins sur le serveur dédié qui est distant , d'où on a eu à utiliser Putty comme la figure ... le montre



et pour se faire nous avons besoin de la machine virtuelle centos dont on a exécuter les différentes étapes d'installation de jenkins

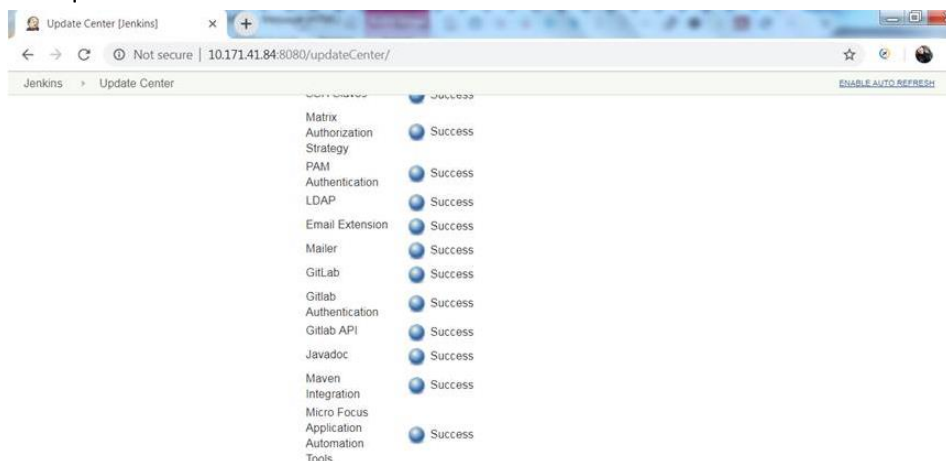


Installation Plugin: nous avons procédé par une recherche du plugin approprié pour chaque étape:

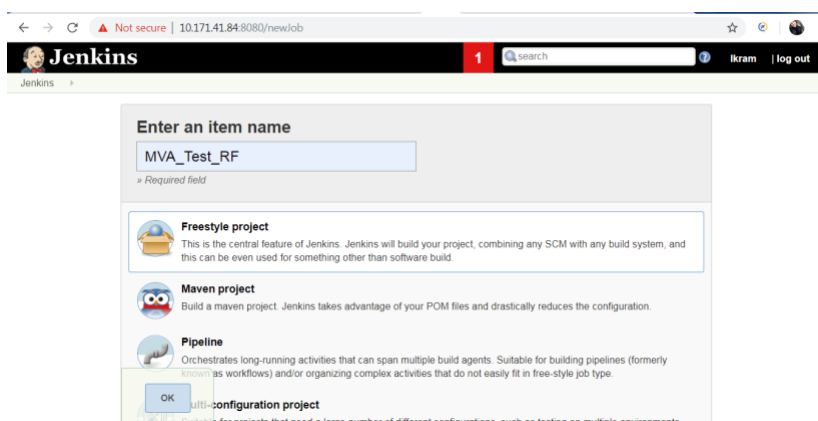
1- Robot Framework : Ce plugin collecte et publie les résultats des tests Robot Framework dans Jenkins.
2- Gitlab plugin: Ce plugin est un déclencheur de build qui permet à GitLab de déclencher des builds Jenkins lorsque du code est "pushed" ou qu'une demande de "merge" est créée.

3- Micro Focus Application Automation Tools : Ce plugin vous permet également de télécharger les résultats des tests vers ALM. En outre, les utilisateurs d'ALM Octane peuvent suivre et déclencher les pipelines Jenkins à partir de l'interface utilisateur.

4- API access côté Octane : Pour que Jenkins accède à ALM Octane, vous devez lui attribuer une clé d'accès enregistrée. Jenkins utilise la clé d'accès pour l'authentification lors de la communication en tant que client avec ALM Octane.



une fois les plugins installés et configurés , nous avons testé la chaine par la création d'un nouveau job comme montré sur la figure ci après: ..



2- Configuration du lien avec Gitlab :

Après on a configuré la récupération du code source déjà placé sur GitLab

Source Code Management

☐ None
☒ Git

Repositories

Repository URL:

Credentials:

Branches to build

Branch Specifier (blank for 'any'):

Repository browser:

Additional Behaviours:

La figure représente la configuration du build qui exécute la commande batch suivante qui elle génère le résultat, exécuté par Robot framework, affiché dans la figure ainsi que le rapport généré

Build

Execute Windows batch command

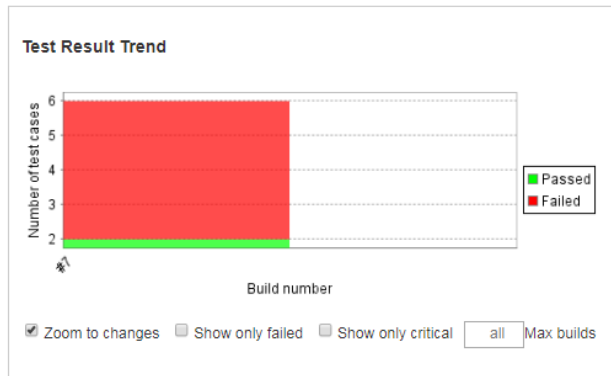
Command:

See [the list of available environment variables](#)

La figure représente un graphique de la tendance des résultats de test généré par Jenkins dont on trouve le nombre de cas de tests passé on mentionnant leurs status “passed” ou “failed”

Robot Framework Test Results

Executed: 20190807 16:09:43.718
Duration: 0:00:49.743 (+0:00:49.743)
Status: 6 critical test, 2 passed, 4 failed
6 test total (±0), 2 passed, 4 failed
Results: [report.html](#)
[log.html](#)
[Original result files](#)



2- Configuration du lien avec Octane :

et vers la fin Jenkins va déposer ces résultats et rapports dans ALM Octane à l'aide d'un pipeline déjà crée.

1- config coté jenkins: nous avons installées le plugin "Micro Focus Application Automation Tools" comme nous avons décrit précédemment

2- config coté Octane : API Access qui n'était pas disponible sur la version utilisé par l'équipe. une demande vers l'équipe adéquate a été faite pour la permission d'installer ce plugin.

Mais à cause des vacances du responsable et sur la fin du stage, nous avons pas pu terminer cette tâche .

Conclusion :

Dans ce chapitre j'ai présenté l'architecture de notre application, l'environnement logiciel de la réalisation ainsi que quelques interfaces utilisateurs tout en décrivant les outils utilisés pour cette réalisation. Dans une deuxième partie j'ai expliqué l'intégration continue assurée par jenkins.

Conclusion générale

Le présent rapport a été réalisé dans le cadre de stage d’immersion en entreprise effectué au sein de la société Sofrecom Tunisie qui s’inscrit dans le cadre de la formation d’ingénieur en informatique à l’Ecole Nationale des sciences de l’informatique.

Le but de notre projet était l’automatisation des tests de l’application MVA ainsi que la mise en place d’une solution d’intégration continue à l’aide d’un outil devop “Jenkins” pour assurer la continuité.

De nos jours, l’automatisation des tests jouent un rôle important dans le processus de développement logiciel en raison de la complexité croissante des applications. Il améliore la qualité du logiciel et réduit les coûts du projet. Bien que l’automatisation des tests présentent de nombreux avantages, ils ne peuvent pas totalement remplacer les tests manuels. Les tests manuels et automatisés doivent être effectués en parallèle. Un défi du projet et des tests automatisés est qu’ils sont vulnérables au changement du logiciel testé. Une mise à jour de l’application peut provoquer des résultats de test inattendus si la structure des tests n’est pas bien organisée. Ainsi, la maintenance deviendra un problème lorsque le nombre de cas de test augmentera énormément.

L’automatisation des tests continueront à se développer dans le futur. De nombreux outils d’automatisation ont été créés et de nouvelles bibliothèques améliorées pour Robot Framework seront mises en vente afin de répondre aux besoins de la communauté des tests de logiciels d’automatisation.

Annexe

Installation de Robot Framework (à mettre dans l’annexe)

1. Download and install Python 2.7 (version **2.7.11**):

(<https://www.python.org/ftp/python/2.7.11/python-2.7.11.msi>)



!! Il faut ajouter le python.exe aux variables d'environnement

2. > `python -m pip install -U pip`

3. > `pip install robotframework`

4. > `pip install robotframework-selenium2library`

5. > `pip install -U selenium`

6. Télécharger et installer « **wxPython2.8-win32-unicode-2.8.12.1-py27.exe** »

(<https://sourceforge.net/projects/wxpython/files/wxPython/2.8.12.1/wxPython2.8-win32-unicode-2.8.12.1-py27.exe/download>)

7. > `pip install robotframework-ride`

8. > `ride.py`

9. > `pip install robotframework-appiumlibrary`

10. Télécharger et installer « **autoit-v3-setup.exe** » (<https://www.autoitscript.com/cgi-bin/getfile.pl?autoit3/autoit-v3-setup.exe>)

11. Télécharger et installer « **ActivePython-2.7.10.12-win32-x86.msi** »

12. Télécharger et installer « **python-2.7.11.amd64.msi** »

13. Télécharger et installer « **pywin32-217.win32-py3.3.exe** »

14. Télécharger et installer « **PIL-1.1.7.win32-py2.7.exe** »

15. Télécharger « **robotframework-autoitlibrary-master.zip** »

(<https://github.com/qitaos/robotframework-autoitlibrary/archive/master.zip>), décompresser le zip puis à partir de l'invite de commande en **mode administrateur**, accéder au répertoire

« **robotframework-autoitlibrary-master** » et lancer la commande :

> `python setup.py install`

Octane :