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 enchainerons avec la description du projet en question ainsi que la problématique et la solution proposée.

Présentation de l'organisme d'accueil :

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

Sofrecom, filiale du groupe Orange, 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. Sofrecom, the Know-How Network

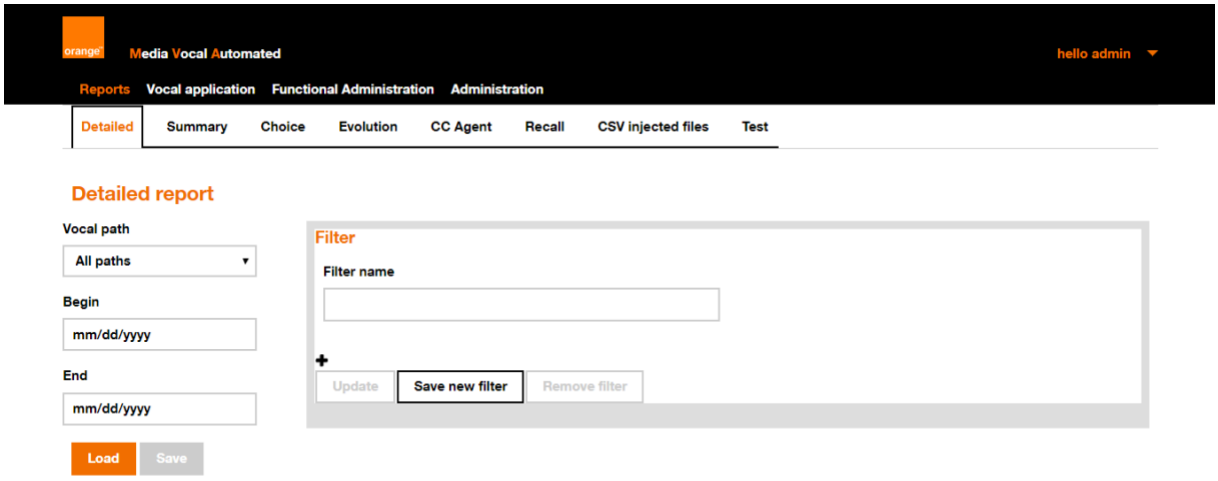
//presentation de l'equipe et role

Présentation de Projet MVA :

MVA (Media Vocal Automatisé) est un outil visant à améliorer la satisfaction client. 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.

MVA est un service existant permettant de lancer des campagnes d'enquêtes téléphoniques de satisfaction automatisées.

MVA est une application utilisant Campagne d'Appel pour effectuer des appels vocaux via le serveur OMS (Orange Media Server) serveur média.



Présentation des intervenant de projet :

Scrum Master	Product Owner	Équipe de développement	Équipe test et validation
-Wided LIMAME	-Laurent ANDRE	-Omran SOUIBGUI -Khouloud ENNAOUI	-Aymen MHADHEB -Ikram SASSI

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.

(à développer)

Conclusion:

Dans ce chapitre, nous avons décrit l'entreprise d'accueil et nous avons présenté le cadre du projet en précisant sa problématique.

Etat de l’art

Introduction:

Dans ce chapitre je vais faire …….

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.

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 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 tester 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

Dans ce chapitre nous avons présenté le contexte industriel avionique, les aspects théoriques nécessaires à la compréhension et à l'élaboration de ce projet en décrivant le processus de travail en premier lieu, puis nous avons analysé ce processus pour dégager les différents problèmes présents dans cette activité pour terminer par la proposition des solutions qui ramènent à corriger ces problèmes

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

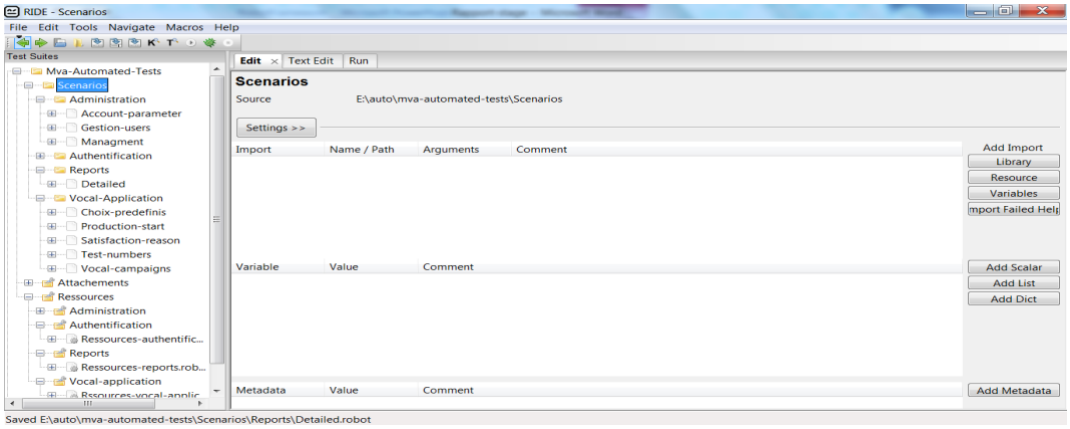
Robot Framework :

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.

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

2.1.Les bibliothèques utilisé

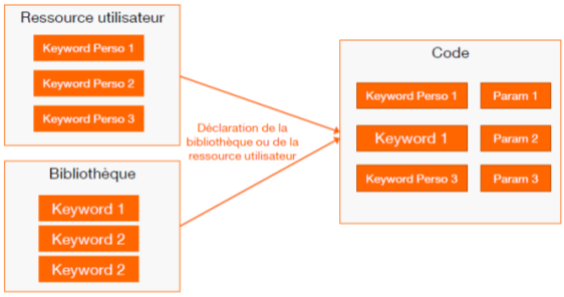
Les bibliothèques de test fournissent les fonctionnalités de test réelles qui se base sur des mots-clés. Il existe plusieurs bibliothèques standards intégrées au framework, et de nombreuses bibliothèques externes développées séparément peuvent être installées en fonction des besoins.

- **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 une couche qui englobe les :

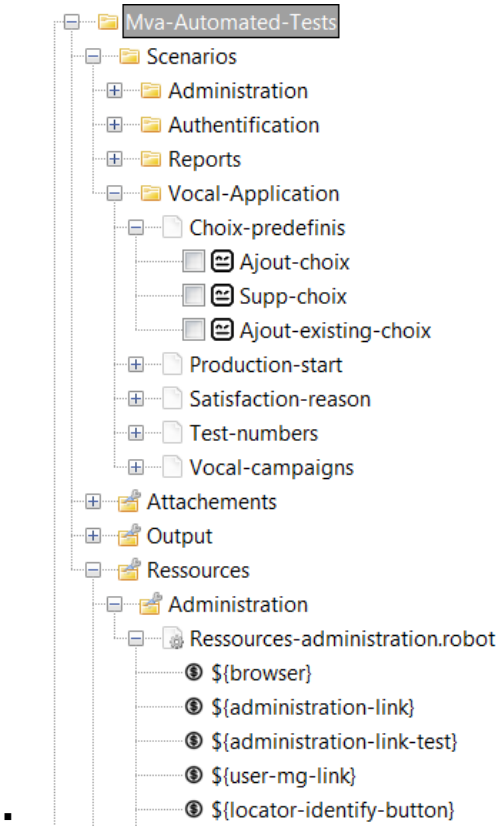
- Test Suite : ensemble de cas de tests qui regroupe le même volet fonctionnel par exemple.
- Test case : les étapes à dérouler pour vérifier un résultat attendu.

-**Attachment** est une couche dont on trouve les fichier utilisé par l'application

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

-Ressources c'est une 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)



2.3.Enchaînement de travail:

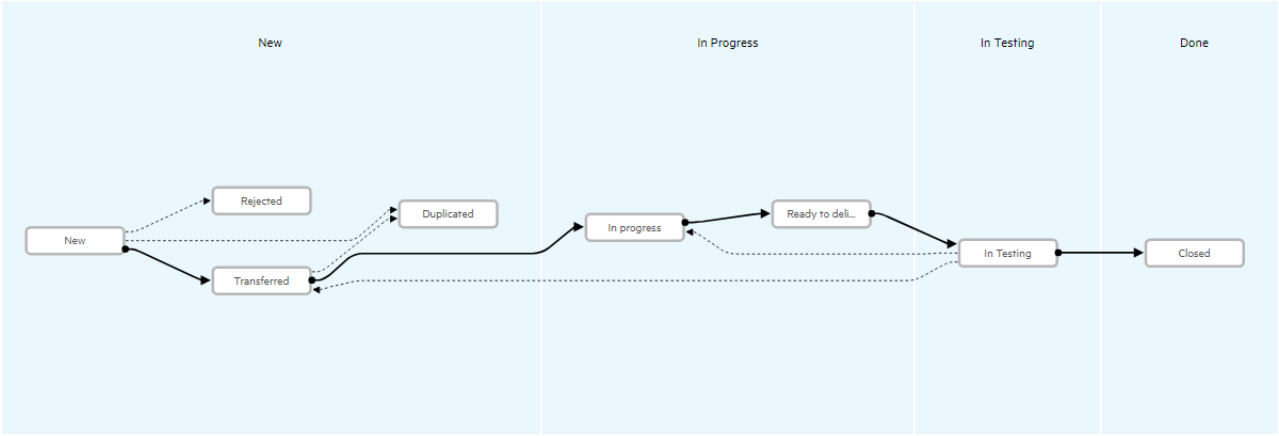
ce stage m'a permis d'automatiser une quarantaine de cas de test et pour se faire il faut tout d'abord préparer un plan de test dans lequel on décrit la portée et les activités de test du logiciel après il fallait identifier les bugs et vérifier s'ils sont automatisables ou pas et si ce le cas nous allons adopter le workflow dans la figure ci-dessous"n7ot noumrouha" dans lequel on trouve les différents états de bug ,depuis sa naissance "new" jusqu à sa correction "closed", tel que :

-new :

-deffered:

……..

et par la suite on doit passer à l'automatisation.



Exemple de scénario de test :

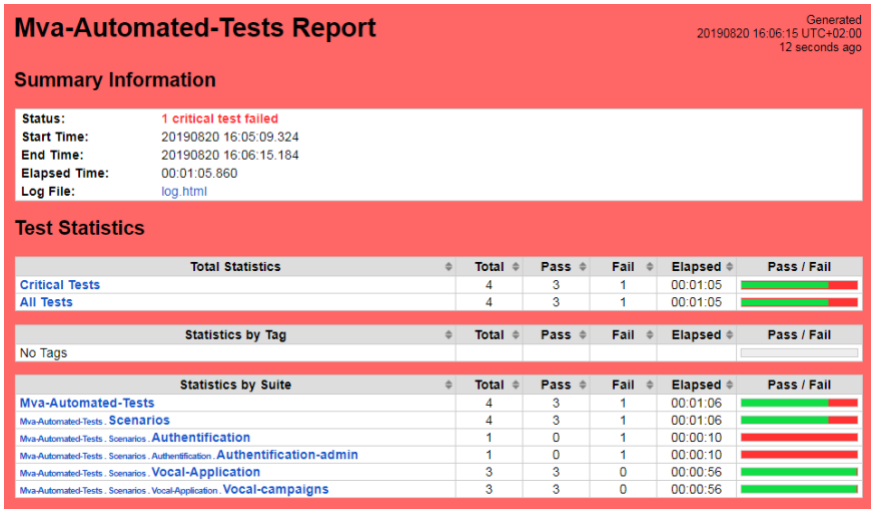
- Keywords

```
*** Keywords ***
Ajout
    [Arguments]    ${vocal-path}    ${new-locator}    ${button}
    Input-text    ${vocal-path}    ${new-locator}
    click button    ${button}

Modif-priority
    [Arguments]    ${action-locator}    ${new-locator}
    click link    ${action-locator}
    Element Should Contain    ${new-locator}    test

Input-text
    [Arguments]    ${vocal-path}    ${new-locator}
    wait until element is visible    ${new-locator}
    input text    ${new-locator}    ${vocal-path}

Action
    [Arguments]    ${action-button}
    click button    ${action-button}

Supp-path
    Choose Ok On Next Confirmation
    click link    ${del-path-link}
    Confirm Action
    Page Should not Contain    ${path-to-delete}
```

- Cas de test

```
1  *** Settings ***
2  Library          Selenium2Library
3  Resource         ../../Ressources/Authentification/Ressources-authentification.robot
4  Library          OperatingSystem
5
6  *** Test Cases ***
7  authentification-admin-success
8      Authentification-mva    ${url}    ${browser}    ${password-admin}    ${login-admin}
9      Profile-admin
10     close browser
11
12 authentification-admin-wrong-pwd
13     Authentification-mva    ${url}    ${browser}    ${password-incorrect}    ${login-admin}
14     Page should contain    ${message-erreur-lg-mp-incorrect}
15     close browser
16
17 authentification-admin-wrong-login
18     Authentification-mva    ${url}    ${browser}    ${password-admin}    ${login-incorrect}
19     Page should contain    ${message-erreur-lg-mp-incorrect}
20     close browser
21
22 authentification-admin-wrong-login-mp
23     Authentification-mva    ${url}    ${browser}    ${password-incorrect}    ${login-incorrect}
24     Page should contain    ${message-erreur-lg-mp-incorrect}
25     close browser
```

Exemple de rapport de test :



Mva-Automated-Tests Report

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

Summary Information

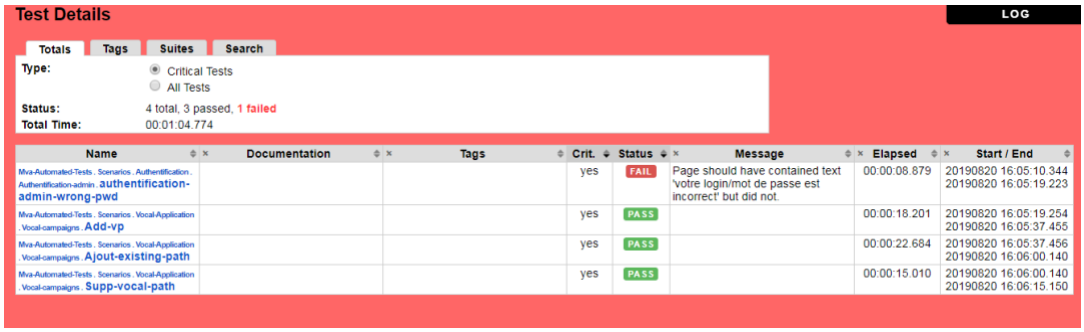
Status:	1 critical test failed
Start Time:	20190820 16:05:09.324
End Time:	20190820 16:06:15.184
Elapsed Time:	00:01:05.860
Log File:	log.html

Test Statistics

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

Statistics by Tag	Total	Pass	Fail	Elapsed	Pass / Fail
No Tags					

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



Test Details

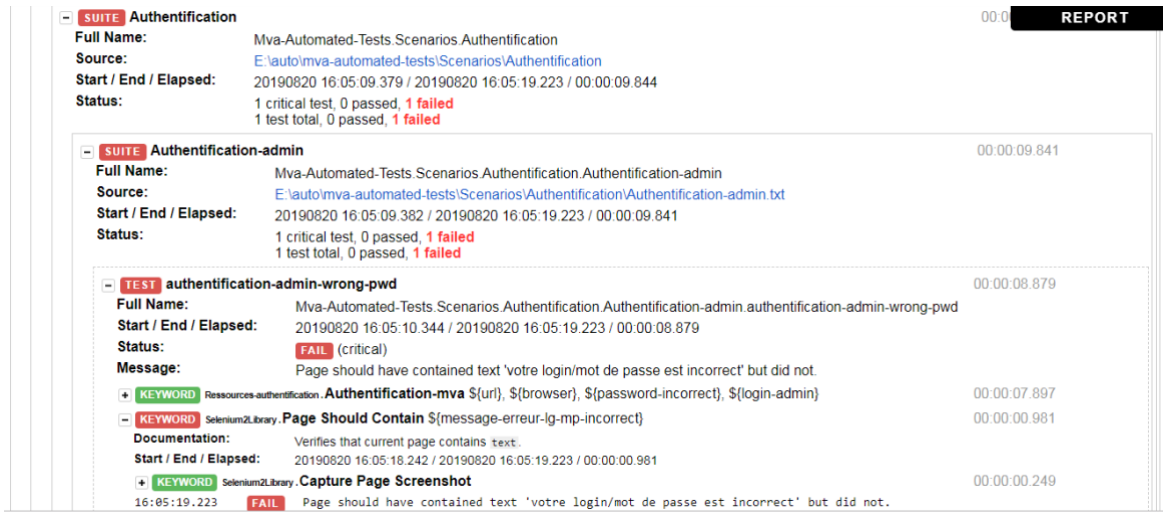
LOG

Totals | Tags | Suites | Search

Type: Critical Tests / All Tests

Status: 4 total, 3 passed, 1 failed
Total Time: 00:01:04.774

Name	Documentation	Tags	Crit.	Status	Message	Elapsed	Start / End
Mva-Automated-Tests . Scenarios . Authentification . Authentification-admin . authentification-admin-wrong-pwd			yes	FAIL	Page should have contained text 'votre login/mot de passe est incorrect' but did not.	00:00:08.879	20190820 16:05:10.344 20190820 16:05:19.223
Mva-Automated-Tests . Scenarios . Vocal-Application . Vocal-campaigns . Add-vp			yes	PASS		00:00:18.201	20190820 16:05:19.254 20190820 16:05:37.455
Mva-Automated-Tests . Scenarios . Vocal-Application . Vocal-campaigns . Ajout-existing-path			yes	PASS		00:00:22.684	20190820 16:05:37.456 20190820 16:06:00.140
Mva-Automated-Tests . Scenarios . Vocal-Application . Vocal-campaigns . Supp-vocal-path			yes	PASS		00:00:15.010	20190820 16:06:00.140 20190820 16:06:15.150



SUITE Authentification — 00:0 REPORT

Full Name: Mva-Automated-Tests.Scenarios.Authentification
Source: E:\auto\mva-automated-tests\Scenarios\Authentification
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 Authentification-admin — 00:00:09.841

Full Name: Mva-Automated-Tests.Scenarios.Authentification.Authentification-admin
Source: E:\auto\mva-automated-tests\Scenarios\Authentification\Authentification-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 authentification-admin-wrong-pwd — 00:00:08.879

Full Name: Mva-Automated-Tests.Scenarios.Authentification.Authentification-admin.authentification-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 Ressources-authentification . Authentification-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.

////Realisation kpi test executer manuellement et ttest auto 9dech mn wa7da

lazemni nthabet ml faisabilité d'auto test

cas de test utilsé 1 seule fois et prend bcp de temp sert a rien de l'automatiser

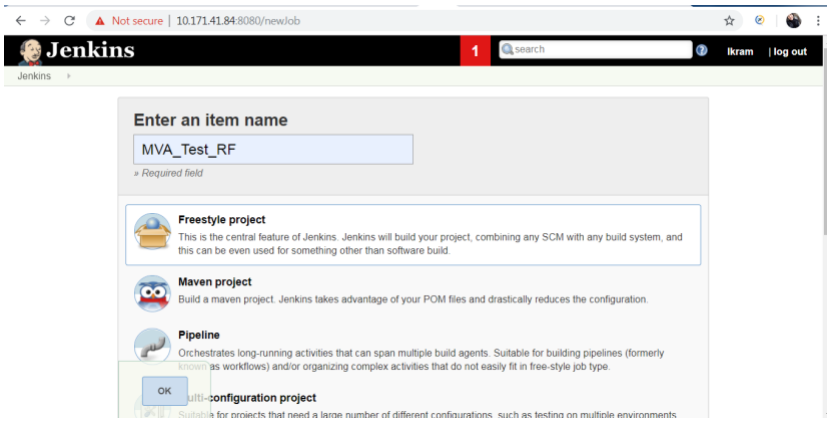
les tests automatisé enrichi soit sanity test or regression test à l'aide de tag ""

3.Intégration continue

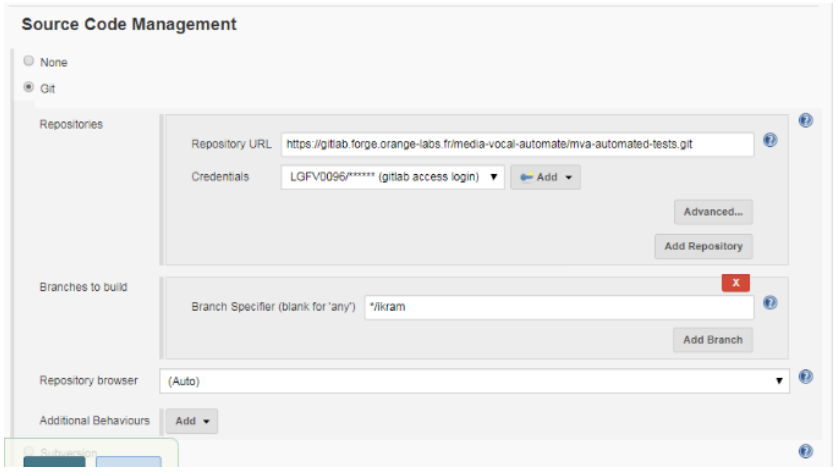
L'intégration continue (CI) est une pratique de DevOps dans laquelle les membres de l'équipe commit régulièrement les modifications de code dans le repository de contrôle de version, après que les builds et les tests automatisés sont exécutés. La livraison continue (CD) est une série de pratiques dans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

On a choisi d'intégrer l'outil d'intégration continue Jenkins, qui est l'un des meilleurs outils actuellement.

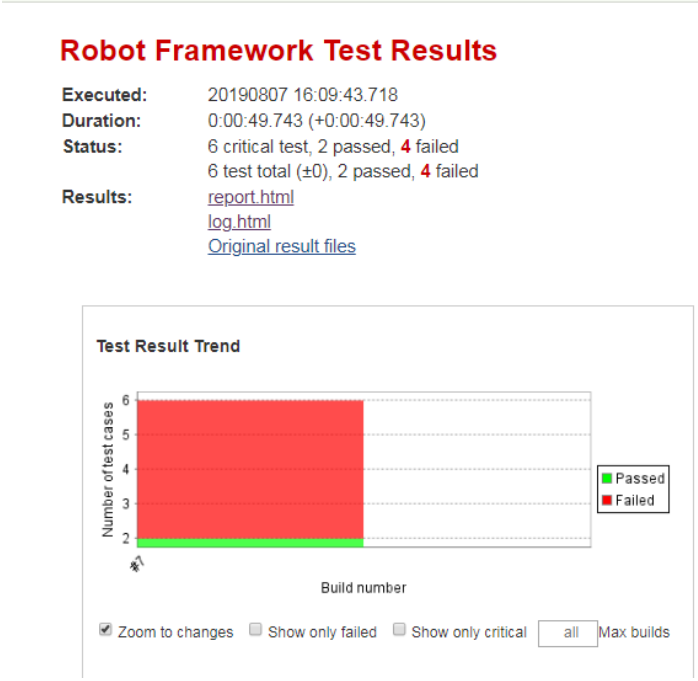
Pour l'organisation l'IC on a créer le job "MVA_Test_RF" avec les tests comme la figure … le montre



Après on a configurer la récupération du code source pour le prendre à partir de gitLab



La figure …. représente la configuration de build pour qu'elle exécute la commande batch suivante dont elle génère le résultat, exécuté par Robot framework, affiché dans la figure …. ainsi que le rapport généré



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

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/cgibin/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