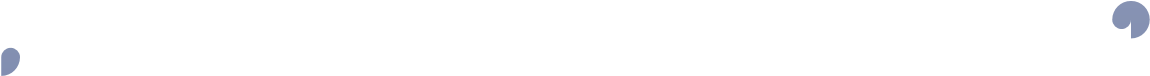
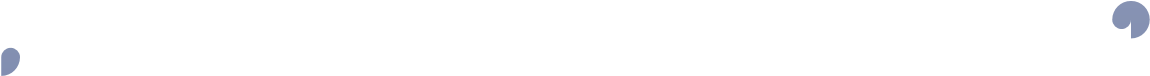


Dédicace

A

MES

PARENTS



Remerciements

Je tiens à remercier tout d’abord Allah le tout puissant, pour l’inteligence et la force qu’il m’a donné pour atteindre les objectifs de ce mémoire, je tiens également à exprimer ma gratitude et ma reconnaissance aux personnes suivantes :

* **M. GUIMEZAP Paul**, le Président Fondateur de l’IUC pour l’opportunité qu’il offre aux jeunes par le biais de son établissement de recevoir une formation et d’assurer un avenir certain ;
* **Dr. MEUKALEUNI Cyrille**, mon superviseur académique, pour sa disponibilité, ses conseils et son assistance ;
* **M. FOTSO KUATE Victor**, mon encadreur académique, pour son assistance exceptionnelle, sa disponibilité, ses nombreux encouragements et surtout ses judicieux conseils qui ont contribué à alimenter ma réflexion ;
* **M. LAGOUE Rodrigue**, mon encadreur professionnel, qui m’a beaucoup appris sur les défis à relever dans le monde des affaires ; il a partagé ses connaissances et expériences dans ce milieu, tout en m’accordant sa confiance, ses conseils et la patience;
* **M. TSOPBENG David**, le Directeur de l’Institut Supérieur des Technologies et du Design Industriel (ISTDI), pour tous les efforts qu’il a fournis en faveur à notre formation ;
* **M. NWOKAM Verlaine,** le Chef de département des masters génie informatique à l’IUC, pour tous ces efforts pendant notre formation ;
* **Dr FOUOKENG**, pour toutes les possibilités qu’il a su mettre à ma disposition pour le compte de ma formation, ses conseils, sa disponibilité, ses encouragements ;
* **M.** **NYAM Aquila**, pour ses conseils, sa disponibilité, ses encouragements ;
* **Ma famille,** pour leurs soutient moral, et financier, pour les conseils qu’ils me donnent au quotidien, ainsi que mes frères et sœurs pour leur amour inconditionné ;
* J’adresse mes sincères remerciements à tous les **professeurs**, **intervenants** et toutes les **personnes** qui par leurs paroles, leurs écrits, leurs conseils et leurs critiques ont guidé mes réflexions et ont accepté de me rencontrer et de répondre à mes questions durant mes recherches ;
* **Mes camarades classes**, amis et collègues, pour leurs aides et leurs disponibilités ;
* Tous ceux qui ont participé de près ou de loin à la rédaction de ce mémoire.



Sommaire

# Dédicace................................................................................................................................. i

Remerciements ...................................................................................................................... ii

Sommaire ............................................................................................................................. iii

Abréviations ......................................................................................................................... iv

Liste des figures ................................................................................................................... vi

Liste des tableaux ................................................................................................................ vii

Résumé .............................................................................................................................. viii

Abstract ................................................................................................................................ ix

Introduction générale ..............................................................................................................x

Partie I:...................................................................................................................................1

Etat de l’art .............................................................................................................................1

Chapitre I: Revue de la littérature ....................................................................................2

Section 1. Concepts et définitions ...............................................................................2

Section 2. L’utilité des tests et validation logiciel........................................................6

Section 3. Vue panoramique des solutions de tests logiciel ....................................... 11

Chapitre II: Cadrage du projet ..................................................................................... 13

Section 1. Contexte du projet et étude de l’existant ................................................... 13

Section 2. Cahier de charge....................................................................................... 18

Section 3. Méthodologie et outils envisagés .............................................................. 24

Partie II: Analyse, Conception et implémentation .............................................................. 47

Chapitre III: Analyse et conception .............................................................................. 48

Section 1. Analyse approfondie ................................................................................ 48

Section 2. Modélisation conceptuelle ........................................................................ 59

Section 3. Architecture générale de l’application ...................................................... 68

Chapitre IV: Résultats obtenus et discussion ................................................................. 73

Section 1: Outils et méthodes d’implémentation ....................................................... 73

Section 2: Présentation de la solution réalisée ........................................................... 76

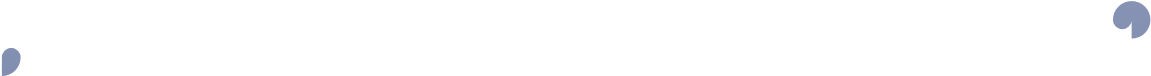
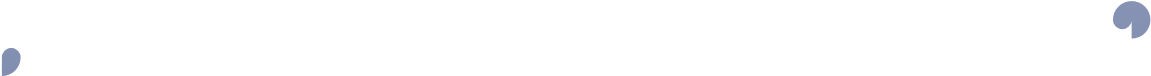
Section 3: Evaluation des résultats et impact sur l’entreprise ..................................... 81

Conclusion générale ............................................................................................................. 84

Annexes ............................................................................................................................... 85

Tables de matières ................................................................................................................ 89

Références bibliographiques ................................................................................................. 92



Abréviations

**AJAX** : Asynchronious Javascript And Xml

**API :** Application Programming Interface **ASP :** Active Server Pages

**CFTL** : Comité Français de Test Logiciel

**COM :** Component Object Model

**DOM :** Document Object Model

**HTML** : HyperText Markup Language **HTTP :** HyperText Transfert Protocol

**IBM** : International Business Machines

**IDE** : Integrated Development Environment

**ISTQB** : International Software Testing Qualification Board

**JVM :** Java Virtuel Machine

**MVC :** Modèle Vue Contrôleur

**OLE :** Object Linking and Embeding

**OS :** Operating system

**PC :** Personal Computer

**PDCA** : Plan Do Check Act

**PHP :** Hypertexte PreProcessor

**RAM :** Random Access Memory

**SC Group :** Smart Code Group

**SWT :** Standard Widget Toolkit

**UFT :** Unified Functional Testing

**UI** : User Interface

**UX** : User eXperience

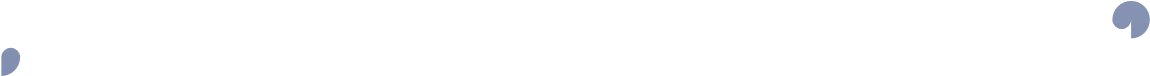
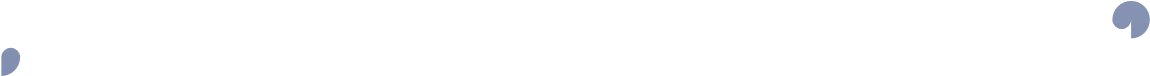
**VB :** Visual Basic

**VS Code** : Visual Studio Code

**VVT** : Validation Vérification et Test Logiciel

**WATIR :** Web Application Testing In Rugy

**BPMN:** Business Process Management and Notation



Liste des

figures

Figure 1. 1. Patch devant réduire des temps de réponse .....................................................7

Figure 2. 1: Processus de test manuel .............................................................................. 16

Figure 2. 2: Succession de tâches avec Gantt .................................................................. 23

Figure 2. 3: Présentation du fonctionnement de Scrum .................................................... 24

Figure 2. 4: Eclipse ......................................................................................................... 25

Figure 2. 5: Logo de Visual Studio Code ........................................................................ 26

Figure 2. 6: Fonctionnement de Protractor ...................................................................... 35

Figure 3.1 : Roue de Deming .......................................................................................... 61

Figure 3.2 : Organigramme de fonctionnement du test après optimisation ....................... 62

Figure 3.3 : Cas d’utilisation du logiciel de test ............................................................... 65

Figure 4.1: Logo de Creately ............................................................................................. 66

Figure 3.4 : Cycle de vie d’un logiciel, cycle en V .......................................................... 67

Figure 3.5 : Architecture de Selenium WebDriver ........................................................... 69

Figure 3.6 : Compatibilité entre navigateurs .................................................................... 71

Figure 4.2: Enregistrement du projet ................................................................................. 76

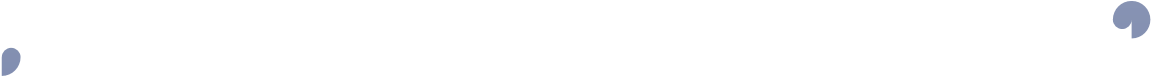
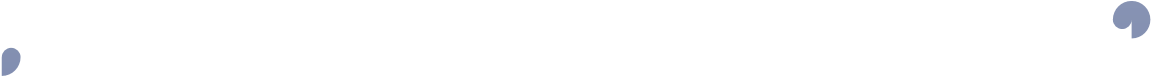
Figure 4.3: Démarrage de test du site smart code group ..................................................... 76

Figure 4.4: Cas de test effectué avec succès ....................................................................... 77

Figure 4.5: Exportation du test dans un IDE ...................................................................... 78

Figure 4.6: Ouverture du test sur Visual Studio Code ........................................................ 79

Figure 4.7: Test effectué avec erreur .................................................................................. 80

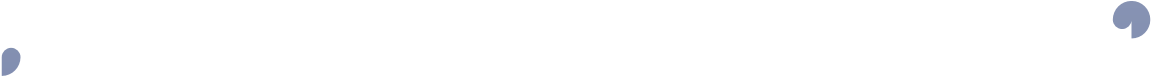
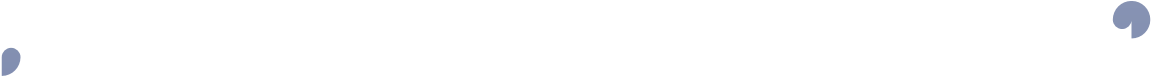


Liste des tableaux

Tableau 1.1 : Classification des anomalies par niveau de gravité .........................................5

Tableau 1.1 : Caractéristiques principales des solutions de tests existantes ........................ 12

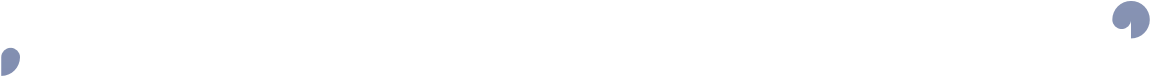
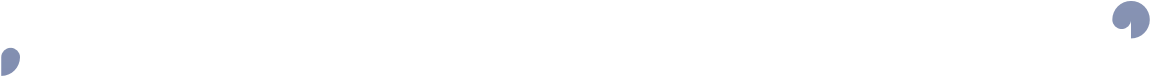
Tableau 2.1: Tableau de répartition des tâches.................................................................. 22



Résumé

L’environnement actuelle est dominé par l’économie numérique, l’on assiste à un développement exponentiel des équipements et logiciels informatiques, dans ce contexte, les logicielles doivent répondre aux besoins des utilisateurs en garantissant la continuité des activités sans interruptions. Par ailleurs, la principale préoccupation traitée dans ce mémoire de fin d’étude du cycle Master est celle de pouvoir trouver une solution qui donnerait la possibilité de sécuriser les applications SaaS pour qu’elles tendent vers la « sécurité optimale ». Le but de ce travail est de mettre sur pied un processus d’audit à la fois manuel et automatisé permettant d’identifier et réduire au maximum les possibilités d’attaque des logiciels SaaS qui peuvent limiter ou empêcher leur bon fonctionnement. En effet l’on souhaiterait avoir un produit fini exempt de faille ou tout au moins, qui dispose le moins possible, afin de garantir la fiabilité dudit produit et ainsi fidéliser la clientèle. Pour cela, nous avons commencé par étudier le processus d’audit de sécurité des SaaS préalablement utilisé à LAO SARL, ensuite nous avons étudié les d’audits de sécurité existants afin de faire une analyse comparative et d’en ressortir le meilleur, après cette étude, nous avons amélioré la pratique de l’audit de sécurité, grâce à méthode d’audit qui répond mieux au besoin interne de l’entreprise à savoir éviter un accroitre les possibilités de sécurisation de leur SaaS, la génération des rapports de meilleur rapport d’audit, l’ajout de l’outils de capture d’écran de graphe pour une meilleur prise de décision. Ainsi, en s’appuyant sur la méthode « pentesting », nous avons réalisé une un audit qui nous a permis d’aboutir à un niveau optimal de sécurité.

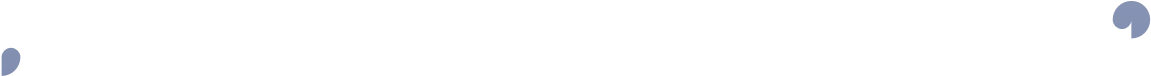
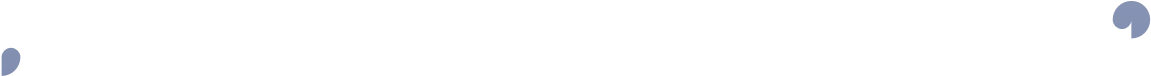
Mots clés : Audit, Pentesting, SaaS, Sécurité



Abstract

The current environment is determined by the digital economy, assisting in an exponential development of computer equipment and software, in this context, software must meet the needs of users by ensuring business continuity without interruptions. In addition, the main concern dealt with in this master's degree thesis is that of being able to find a solution that would make it possible to secure SaaS applications so that they tend towards “optimal security”. The goal of this work is to set up an audit process that is both manual and automated to identify and minimize the possibilities of using SaaS software that can limit or prevent their proper functioning. In fact, we would like to have a finished product that is free from flaws or at least that has as little as possible, in order to guarantee the reliability of said product and thus build customer loyalty. For this, we started to study the SaaS security audit process used at LAO SARL, then we studied the existing security audits in order to make a comparative analysis and to come out the best, after this study, we have improved the practice of security auditing, thanks to the audit method that better meets the internal need of the company to know to avoid an increase in the possibilities of securing their SaaS, the generation of better reporting reports. audit, adding graph screenshot tool for better decision price. Thus, by relying on the "pentesting" method, we carried out an audit which enabled us to achieve an optimal level of safety.

Keywords: Audit, Pentesting, SaaS, Security



Introduction générale

L’absorption de l’environnement national et international par l’économie numérique bat son plein avec la crise sanitaire du au covid-19. Les logiciels offrant les possibilités de travail étendu (télétravail), tant au niveau national et qu’international, sont très convoitées. A cet égard, les organisations sont en permanence à la recherche de logiciels pouvant au mieux, satisfaire leurs besoins aussi bien pour les petites que les grandes. Le logiciel intervient alors en réponse aux besoins de ces organisations notamment en leur apportant au moins un maintien, sinon une amélioration de leur productivité et du rendement de leurs activités. Et la garantie de ceci relève d’un alignement stratégique avec les organismes, mais aussi de la sécurité de celles-ci. D’où la nécessité des audits de sécurité afin de fiabiliser les produits logiciels en générale et particulièrement de applications SaaS qui se trouve dans jungle de l’internet.

La question principale de ce travail est de savoir comment procéder à des audits de sécurisation des logiciels SaaS. Le pentesting est une méthode d’audit de sécurité, qui va bien au-delà de l’audit conventionnel, en surpassant la prise en compte des dispositions sécuritaires qu’elle propose pour limiter des attaques, pour simuler de véritables attaques afin de prévenir d’éventuelles actions malveillantes. Ceci va offrir une de nombreux avantages et un meilleur rendu sécurité. En effet, les tests d’intrusions renvoient à l’utilisation de scénario programmé ou non pour tester un ou plusieurs aspects de la sécurité d’une application quelle qu’elle soit. Aujourd’hui, les applications SaaS sont fréquemment mises à jour fournissant de nouvelles caractéristiques et fonctions avec une expérience utilisateur améliorée due à une ergonomie de qualité. Cependant, chacun de ces changements, peut faire apparaitre des failles inattendues, c’est pour cela que chaque mise à jour devrait déclencher le test d’intrusion. Dans le cas contraire on prend le risque de lancer une mise à jour qui aboutirait à une ou plusieurs vulnérabilité(s), pouvant offrir des possibilités d’insécurité plus élevées, et donc si exploitées par un tiers, une mauvaise expérience utilisateur et des conséquences catastrophique pour l’entreprise.

En outre, les actions d’un test d’intrusion peuvent se répéter, dans ce cas de figure, un gain d’efficacité est assuré par une automatisation, avec un gain de temps, et une exécution plus rapide, ainsi que des rapports éditer tout aussi automatiquement. Ainsi, à la question principale évoquée ci-dessus, s’ajoute une question secondaire, à savoir comment automatiser le processus de tests d’intrusion logicielle en automatisant les tâches répétitives.

L’objectif principal est de réaliser un audit de sécurité grâce la méthode de penstesting d’une application SaaS avec les outils prévus à cet effet afin de lui assurer une sécurité optimale. Pour atteindre nos objectifs, il sera intéressant de faire un état de l’art et une étude sur la mise en place de la méthode d’audit, rédiger un cahier de charge pour la description du travail à effectuer, l’analyse, l’implémentation et la mise en œuvre du projet et l’évaluation des résultats obtenus.

Partie I:

# Etat dE l’art

## Chapitre I :Revue de la littérature

Dans ce chapitre, nous abordons de manière détaillée, les concepts au centre de notre thème, ainsi que l’utilité des audits de sécurité en illustrant les failles et leurs conséquences, et enfin une vue des méthodes d’audit existantes.

### Section 1. Concepts et définitions

D’entre la pléthore de problème que peut rencontrer un logiciel SaaS, un problème fondamental est « le problème de la sécurité », en plus de nombreux défauts et anomalies liés aux éléments qui le constituent. Aussi, dans cette partie, nous porterons notre attention sur les concepts (SaaS, audit…) et les failles avec les niveaux de la gravité associes.

#### 1.Le SAAS

Le cloud computing (*Informatique dans le nuage*) est une technique permettant de gérer des ressources (serveurs, système d’exploitation, des applications…) et d’adapter très rapidement une infrastructure à des variations de charges de manière totalement transparente pour l’administrateur et les utilisateurs. Il a la possibilité de se diversifier de manière typée ainsi qu’il suit : l’IaaS, le PaaS, le SaaS….

Le SaaS est un mode d’utilisation d’une solution logicielle qui se fait en utilisant l’application à distance, elle-même, étant hébergée par l’éditeur. Il se rencontre couramment pour des applications logicielles relatives au CRM ou Web Marketing.

Le Software As A Service (SaaS), ou Logiciel en tant que Service en Français, est **un modèle de distribution de logiciel au sein duquel un fournisseur tiers héberge les applications et les rend disponibles pour ses clients par l’intermédiaire d’internet**. Parmi les principaux fournisseurs d’un logiciel SaaS, on retrouve Salesforce, Oracle, IBM ou encore Microsoft.

La solution logicielle étant utilisée le plus souvent à partir d’un simple navigateur internet, elle permet à l’entreprise d’être dégagée de toutes contraintes d’installation, de mise à jour ou de toutes autres maintenances techniques. Elle permet également d’être utilisée par des collaborateur en situation de mobilité. La mise à disposition d’une solution SaaS peut être facturée par abonnement ou proportionnellement à l’usage. Il peut parfois comporter des frais de personnalisation et de mise à disposition du service. Dans le domaine du web marketing, les plateformes de gestion des campagnes e-mails, les outils d web analytique, et les serveurs publicitaires sont généralement proposés en mode SaaS.

1. **Avantages**

* Grâce à un logiciel SaaS, les entreprises n’ont plus besoin d’installer et de lancer des applications sur leurs propres ordinateurs ou sur leurs Data Centers. Le **coût d’acquisition de matériel est ainsi éliminé, au même titre que les coûts d’approvisionnement et de maintenance, de licence de logiciel, d’installation et de support**. On compte également plusieurs autres avantages ;
* Au lieu d’investir dans un logiciel à installer, et dans un équipement permettant de le prendre en charge, les utilisateurs souscrivent à une offre SaaS. En général, **l’offre se présente sous la forme d’un abonnement mensuel dont le tarif est proportionnel à l’utilisation**. Grâce à cette flexibilité, les entreprises peuvent organiser leur budget avec plus de précision et de facilité. De plus, il est possible de résilier l’abonnement à tout moment pour couper court aux dépenses ;
* Un autre avantage est la haute scalabilité. En fonction de ses besoins, **l’utilisateur peut accéder à plus ou moins de services et à des fonctionnalités à la demande**. Le Logiciel en tant que Service est donc adapté aux besoins propres à chaque business.
* De même, **plutôt que de devoir acheter régulièrement de nouveaux logiciels, les utilisateurs peuvent compter sur le fournisseur SaaS pour effectuer des mises à jour automatiquement et gérer l’ajout de patchs correctifs**. L’entreprise a donc moins besoin d’une équipe d’informaticiens internes.

Enfin, étant donné que les applications SaaS sont délivrées via internet, **les utilisateurs peuvent y accéder depuis n’importe quel appareil connecté et n’importe quelle position géographique**. L’accessibilité est l’un des grands points forts de ce modèle.

Par ailleurs, une**application SaaS peut être utilisée par des milliers, voire des millions d’utilisateurs finaux simultanément puisqu’elle est stockée sur le Cloud**. Par conséquent :

* Plus d’installation ;
* Plus de mise à jour (continuent chez le fournisseur) ;
* Plus de migration de données ;
* Payement à l’usage ;
* Test de nouveaux logiciels avec facilité ;

Même s’il rencontre tout de même quelques limites.

1. **Inconvénients**

* Limitation par définition aux logiciels proposées ;
* Pas de contrôle sur le stockage et la sécurisation des données associées au logiciel ;
* Réactivité des applications web pas toujours idéale ;

En bref, la mise à disposition de telles solutions est sans aucun doute indiquée dans la situation actuelle. Il demeure tout de même important de nous s’intéresser à l’audit sécurité des applications qui y sont hébergées.

1. **Sécurisation d’une application SaaS**

De manière globale, il s’agit de l’étude des systèmes de commande et de communication entre êtres vivants, machines et autres.

Et la sécurité est un état d’esprit confiant et tranquille d’une personne qui se croit ou se sent à l’abri du danger ; ce qui signifie qu’elle dépend de la personne qui ressent cela, du contexte dans lequel elle se trouve, du type de menaces qu’il peut avoir, et bien d’autre encore. Autrement dit, la sécurité dépend de tout un écosystème et donc il n’y a pas véritablement de sécurité, encore moins de cybersécurité, ce qui complique assez le problème.

Puis nous avons les produits, que les entreprises réalisent pour commercialiser, il peut s’agir de :

* Produit embarque,
* Sous-système,
* Solution,
* Logiciel ou suite de logiciel.

L’idée étant de les fournir à l’utilisateur, pour une utilisation dans un contexte propre à lui. Donc l’entreprise réalise un produit, pensant comprendre de quelle manière le client va s’en servir, d’autant plus que c’est son domaine. Ce qui est vrai dans un contexte fonctionnel, dans la mesure où le produit répond à un besoin du client. Aussi dans le domaine de la cybersécurité, c’est le contexte d’utilisation qui dimensionne le type de risque et d’exposition dont le client peut être victime. Faisant ainsi de la sécurisation des produit un domaine à prendre en compte.

Cependant, nous devons définir ce qu’il y a à protéger. Et en matière de cybersécurité, trois éléments clés sont à retenir :

* La confidentialité : les données ne doivent pas être divulguées ;
* L’intégrité : les données ne doivent pas être falsifiables ou fausses ;
* La disponibilité : l’accès aux données doit être contrôlée.

A ce niveau, la sécurisation se restreint à la fonctionnalité d’un produit. A ceux-là, nous pouvons ajouter les notions de :

* Traçabilité : pour répertorier les transactions d’un système
* Non-répudiation : pour éviter de nier tout participation à une transaction

Et même avec ceux-ci, comme dans le contexte actuel, les choses évoluent de manière exponentielle, dans le sens de rendre rapidement des moyens de sécurité mis en place obsolètes, la validité d’un produit en matière de sécurité, ne tient que le temps de se faire dépasser, ce qui arrive assez rapidement dans bien des cas.

Autrement un produit quel qu’il soit, finira bien par être compromis, s’il n’est pas amélioré de manière permanente. Et donc un jour tout produit SaaS finit par être compromis. Cependant il ne faut pas s’arrêter là, mais pouvoir retrouver ce qui s’est passé, de remonter à la cause, de circonscrire les effets et évidemment d’analyser et de remonter toutes les informations nécessaire la compréhension de ce qui s’est passé. Ceci afin d’améliorer les processus afin que la compromission ne se reproduise pas. Mais aussi qu’en cas d’attaque, de compromission et de dégâts occasionnes, que le produit SaaS permette de remonter les information (accès, nature des dégâts, éléments modifiées) y afférant (grâce aux logs par exemple). Ce qui permettra aux assurances de louer, aux clients de porter plainte au niveau de la justice, et en gros d’avoir tout un écosystème d’éléments permettant de minimiser la portée des effets de la compromission d’un produit SaaS. Aussi, nous avons également le cas des logs, qui dans leur fonctionnement normal, permettent de détecter les comportements qui sont également anormaux ou suspects. Et avec la non-répudiation ci-haut citée, pour se rassurer que pour un élément reçu, son émetteur est bien identifiable et celui-ci ne peut pas nier avoir réalisé cette action.

Nous avons par exemple les ransomwares, qui dans leurs actions touches à peu près tous les piliers principaux ci-dessus citées, à l’instar de l’intégrité car ils vont chiffrer toutes les données, la disponibilité car les données sont désormais rendues indisponibles, et la confidentialité car aujourd’hui les organismes attaqués par ceux-ci se retrouvent menacées de divulgations de leurs données si des conditions ne sont pas respectées. Tout ceci faisant paraître le fait qu’une attaque ne ciblera pas toujours qu’un mais la totalité des principaux piliers de la sécurité.

Pour la sécurisation, la plupart des organisations opterait pour la mise en place d’un ensemble de solutions techniques. Ce qui n’est pas une panacée pour nos problèmes de sécurité applicative, aussi nous opterons pour un processus échelonné comme suit :

* Identification des actifs: Il n’est pas juste question de ce que l’organisme veut protéger bien qu’important, mais aussi et surtout de ce que le client veut protéger ;
* Identifications des cadres réglementaires et normatifs : Il est question de limiter l’impact qui leur est associée, ce qui ajoute des contraintes ou empêche de faire un certain nombre de choses ;
* Identifier toutes les parties prenantes : car même si l’organisme est constitué de plusieurs équipes participant à la réalisation d’un produit SaaS, au même titre que la plateforme d’hébergement, les clients qui ont, eux, probablement des clients, l’état et autres utilisateurs qui sont tous à prendre en compte pour trouver la meilleure approche sécuritaire ;
* L’analyse de risque de cybersécurité (bonne pratique) : elle peut être légère, très lourdes, mais elle permet d’avoir une démarche structurée pour maîtriser le niveau et les actions de sécurisation. Par ailleurs, il n’y en a encore, des sociétés qui investissent énormément dans les risques, les analyses de risque, et les déclinaisons techniques, ceci tant, comme elles sont relativement ignorantes du contexte, elles ne savent pas comment s’arrêter, les exigences ne sont pas forcément suffisantes, le contexte est extrêmement important. Aussi nous avons :
  + La définition des architectures : pour la définition d’une architecture sécurisée, il faut savoir ce qu’on fait, c’est à dire : qu’est-ce qu’on fait? Comment est que le produit SaaS est segmenté? Dans quoi il s’inscrit? Comment faire pour qu’à partir de la manière dont les choses sont organisées, que la sécurité soit plus facile à assurer? Exemple, dans la mesure ou des niveaux d’accès sont définis dans l’utilisation d’un produit SaaS et que pour chacun d’eux le même mot de passe est utilisé, est-ce que celui-ci sera sauvegardée?
  + Les solutions techniques : Ceux-ci doivent être déterminer sur la base de tout ce qui précède.

Nous pouvons ainsi en déduire les grands thèmes d’une sécurisation réussie a l’instar de :

* + Analyse des risques et définition des exigences de sécurité ;
  + Définition des conditions d’utilisation du produit SaaS ;
  + Définition de l’architecture interne ;
  + Implémentation des solutions techniques / défense en profondeur ;
  + Protection de l’environnement (de développement, de production, de maintenance) ;
  + Gestion de fin de vie.

1. **Audit sécurité d’une application SaaS**

L'audit de sécurité d'un système d'information (SI) est une vue à un instant T de tout ou partie du SI, permettant de comparer l'état du SI à un référentiel. Il répertorie les points forts, et surtout les points faibles (vulnérabilités) de tout ou partie du système. Aussi, un auditeur devra dresser également une série de recommandations pour supprimer les vulnérabilités découvertes.

Une application SaaS se veut protégée. Cependant tous les organismes n'ont pas besoin du même niveau de sécurité. Un contrat doit donc mentionner l’étendu des tests. Il se peut, par exemple, que l'entreprise ne soit pas intéressée par un audit utilisant le Social engineering. De même, certaines phases comme la collecte d'informations dépendent directement du type d'audit réalisé, ce qui nous amené à la procédure d’un audit de sécurité.

**b. Procédure de l’audit de sécurité**

Cette partie peut se découpe en plusieurs étapes, à l’instar de :

* La prise de connaissance de manière extrêmement fine les attentes du client. Il convient de bien comprendre ses besoins et de les reformuler. Cette première étape est particulièrement importante dans la mesure où elle plante le contexte précis dans lequel l’audit va être mené : autant d’informations qui seront incluses dans le rapport d’audit afin d’en faciliter l’interprétation, même plusieurs années après sa réalisation.
* La lettre de mission, pour la procédure à venir avec deux objectifs principaux :
* Elle est le contrat qui lie l’entreprise et l’auditeur ;
* Elle permet d’informer les différentes personnes impliquées de l’arrivée d’un audit dans l’entreprise, et auprès des salariés avec une légitimation de cet audit par la direction.
* Le recueil de toutes les informations nécessaires pour préparer la mission. Il s’agit de récolter les éléments relatifs à la culture de l’entreprise, au contexte général toujours en corrélation avec le système d’information. Des rendez-nous sont donc organisés avec les personnes concernées.
* La réalisation de la mission, l'audition du système d'information de l'entreprise peu commencé.
* Une réunion de synthèse organisée entre l’auditeur et les personnes intéressées. Il s’agit de s’assurer ensemble que :
* Les questions de l’auditeur ont été bien comprises ;
* Les réponses ont été bien interprétées.

Le rapport est ensuite rédigé, de plusieurs manières (concis et plus complet), car il s’adresse en général à plusieurs types de publics. Celui-ci détaillé, expliquera les attentes de départ, le contexte, les limites, les faiblesses constatées, leur importance relative et les solutions. Il doit être clair et didactique. En aucun cas il ne doit être technique. Cependant il existe plusieurs techniques d’audit comme nous le verrons dans ce qui suit.

**c. Les techniques d'audit de sécurité**

Dans l’implémentation d’un audit de sécurité, plusieurs modes se distinguent. Aussi nous avons les audits dit en mode « boite blanche », « boite grise » et « boite noire ».

**c.1 Audit en « boite blanche »**

Ce type de test nécessite des connaissances importantes sur la structure interne et l’implantation des systèmes à tester. En plus des spécifications fonctionnelles et d’une description à haut niveau du système, on doit également connaître l’architecture complète, les divers schémas, le code source et toute information nécessaire à la compréhension du système. Le test complet permet de déterminer, avec certitude, si les mécanismes de sécurité en place sont fonctionnels ainsi que de s'assurer qu’aucune erreur évidente n’est présente, que la mesure de sécurité s’exécute comme il se doit, sur une base continue et cohérente, et qu’il existe un soutien pour l’amélioration continue de l’efficacité de la mesure.

**c.2 Audit en « boite noire »**

Ce type d’audit ne nécessite aucune connaissance sur la structure interne et l’implantation des systèmes à tester. Une description du système à haut niveau ainsi qu’une connaissance de base des spécifications fonctionnelles sont suffisantes pour ce type de test. Le test de base permet de déterminer si les mécanismes de sécurité en place sont fonctionnels et de s'assurer qu’aucune erreur évidente n’est présente.

**c.3 Audit en « boite grise »**

L’accès de base représente un accès à un système, avec une identité, mais avec peu de privilèges. Exemple, la rigueur des sessions ou la confidentialité de l'information d’un membre connecté pourraient être testées avec ce niveau d’accès.

Et sur la base de ces différents modes, nous pouvons présenter de manière comparative les différents types d’audit dans le tableau ci-après.

*Tableau : Tableau comparatif des types d’audit*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Audit site Web**  **Sécurité site Internet** | **Test d'Intrusion Simulation d'attaque pirate** | **Audit de code applicatif et analyse d'applications** | **Audit Sécurité d'Infrastructures et Systèmes** |
| **Objectif principal** | Recherche, identification et analyse et correction des failles et vulnérabilités latentes sur les sites Web Internet. | Détection ex exploitation de vulnérabilités. Tentative d'intrusion dans le SI. Déterminer les axes d'attaques existants sur l'organisation. | Tester la sécurité d'une application. Recherche de bug / failles compromettant la sécurité. | Identifier les failles et vulnérabilités dans le SI au sein de l'organisation. Validation technique (paramétrage) des éléments du SI. |
| **Résultats attendus** | Liste priorisée des vulnérabilités identifiées. Application de correctifs et contre-mesures. Augmentation du niveau de sécurité Web. | Identifier les points de fragilité du SI. Impact réel d'une attaque sur le SI de l'organisation. | Validation de l'étanchéité d'une application à des attaques possibles. | Optimisation du niveau de sécurité interne à l'entreprise (sécurise des postes, serveurs, annuaires d'entreprise, configuration réseaux, PABX, etc.) |
| **Déroulement de l’intervention** | Expertise effectuée à distance | A distance : BlackBox (test d'intrusion en aveugle)  Sur site : Greybox (périmètre), Whitebox (cible précise dans le SI) | A distance (code source, utilisation de VMs) ou sur site in vivo. | Audit technique réalisé sur site |
| **Exemple de resultat** | Audit sécurité de Site Web d'entreprise. Sécurisation d'Extranet ou site eCommerce marchand. Vérification sécurité avant mise en production d'un nouveau site. | Exemple de résultats  Identifier les failles du SI de l'organisation. Test d'intrusion sur la sécurité physique des réseaux WiFi. Valider les comportements via l'ingénierie sociale. | Application métier sensible, application tierce en interaction avec le SI, validation application avant mise en production | Audit d'un annuaire d'entreprise. Identifier le vol d'information possible par un identifiant utilisateur. Corriger les failles du VoIP. Sécurisation du réseau interne à l'entreprise. |

A ce niveau nous pouvons sans conteste admettre que pour la méthode d’audit qui épouse le mieux notre projet est le test d’intrusion ou pentesting, que nous présenterons en détail dans la prochaine section.

**Section 2.** **Le pentesting**

Le pestesting, ou test d'intrusion, ou test de pénétration, ou simplement pentest, est une méthode qui consiste à analyser une cible (une adresse IP, une application, un serveur web, un réseau complet…) en se mettant dans la peau d'un attaquant (connu aussi sous le nom d’hacker malveillant ou pirate, mais nous y reviendrons) [2]. C’est donc une représentation ponctuelle d’une cible., et ne doit donc pas être confondu avec le scan de vulnérabilités.

En effet, le scan de vulnérabilité est une composante du test d’intrusion, c’est-à-dire une sous-partie. C’est plus précisément un scan (comme son nom l’indique) de la cible qui permet d’énumérer les vulnérabilités, sans tenter de les qualifier ou de vérifier si elles sont exploitables.

**1.** **Utilite du pentesting**

L’étendu des cyberattaques ne cesse de croitre depuis plusieurs années, et les motivations sont d’ordres diverses (économique, politique, ou le challenge). Même si la grande majorité du temps, elle est économique. Plusieurs groupes de hackers existent, et parmi eux des plus ou moins malveillants de par le monde.

La sécurisation des systèmes a souvent été cher et les chances de se faire pirater étaient minces. Cependant la donne a changé depuis car les possibilités piratage sont élevées, surtout dans un contexte ou la maitrise de la sécurité peine à décoller. Il faut donc absolument que chaque entreprise mette en place une politique de sécurisation adaptée, surtout dans le cas de la production d’applications comme les SaaS.

**1.** **OBJECTIFS DU PENTESTING**

A ce stade, nous pouvons aisément présenter les objectifs d’un test d’intrusion. Aussi nous avons :

* Identifier les vulnérabilités de son SI ou de son application ;
* Évaluer le degré de risque de chaque faille identifiée ;
* Proposer des correctifs de manière priorisée ;

Ainsi grâce au test d'intrusion, nous pouvons qualifier la sévérité de la vulnérabilité, la complexité de la correction, et l'ordre de priorité qu’il faut donner aux corrections. Aussi, le but n 'est pas malsain, mais il question est de s'assurer que ces vulnérabilités sont bien réelles (éviter de faux positifs). Un exemple probant, est le cas de ces entreprises ou nous se trouvent encore des vulnérabilités dans leurs serveurs, qui permettraient aux rançongiciels (un rançongiciel (ransonmware en anglais), est un logiciel malveillant qui prend en « otage » vos données personnelles et qui les restitue uniquement contre le versement d'une « rançon ») de se propager [2].

Un test d’intrusion permet d’en prendre conscience et de prioriser les corrections. Sinon la vulnérabilité risque de rester dans le système jusqu'à son exploitation malveillante. Une récente étude a établi que dans le monde, une entreprise se fait attaquer, en moyenne, toutes les 40 secondes par un rançongiciel [2].

En outre, Il est à noter qu’afin de sécuriser une application, les tests d’intrusion peuvent être faits à différents moments :

* Lors de la conception du projet, afin d’anticiper les éventuelles attaques ;
* Pendant la phase d’utilisation, à intervalle régulier
* Suite à une cyberattaque pour ne pas que ça se reproduise

Néanmoins, sa conduite demeure est à l’initiative de l’entreprise qui veut se tester. Et bien entendu comme nous l’avons précisé plut-haut (*cf Section 2- : 2- : c- Tableau*), il peut se faire aussi bien de l'extérieur (boite noire) avec n'importe quelle connexion Internet, de l'intérieur de l’infrastructure (boite blanche), sur le LAN (réseau interne de l'entreprise), qu’avec un peu des deux premiers (boite grise). Ceci par un pentesteur, qui est en charge de réaliser le test d’intrusion.

En bref, maintenant que nous avons défini un test d’intrusion et son utilité, nous verrons dans la prochaine section les différents types de vulnérabilité d’un SaaS. Nous pouvons cependant, admettre qu’un test d'intrusion qui intervient dans la vérification et l’identification des types de vulnérabilités ou d’attaques d'un SI et le cas échéant d’un SaaS (infrastructure, application…). Comme audit de sécurité, il est rapide et efficace. Il donne également des résultats plus pertinents.

**1.** **CHOIX DU TEST**

Pour déterminer le test à effectuer, le choix sera fonction de ce que l’entreprise veut tester. Ce dernier permet de définir la stratégie dont dépendent les résultats. La cible devra dont être minutieusement circonscrite et le type d’attaques dont l’entreprise souhaite se prémunir. Aussi, il faut déterminer le meilleur type de scan (*cf* ) et les meilleures conditions de connaissance du pentesteur avant de débuter le un test.

En somme, nous avons dénombré trois types de tests d'intrusion (interne, externe et un peu des deux), D’où se déduisent trois contextes plus Il y a trois conditions de test d'intrusion : boîte blanche, boîte grise, et boîte noire. Ainsi, en fonction de la situation de départ, un pentesteur peut avoir un niveau d'information qui lui permet d’approfondir ou non un test. Il ne nous reste plus qu’à effectuer un tour d’horizon sur les outils de pentest.

**Section 3.** **Vue panoramique des OUTILS PENTESTING**

Comme nous venons de voir, quel que soit le type de test d’intrusion envisagé, les outils abondent. Dans cette section, nous découvrirons les systèmes d'exploitation et les outils nécessaire à la réalisation d’un test d’intrusion. Il y’en a pour chacun des trois principaux systèmes que sont : Linux, Windows et Mobile avec Android.

* 1. Kali linux

C’est la distribution linux, qui regroupent essentiellement, un ensemble d’outils nécessaires aux tests de sécurité d'un système d'informatique quel qu’il soit. Cette distribution a pris la succession d’une autre : Backtrack. Il est disponible en plusieurs versions :

* 32 ou 64 bits
* Live CD
* Images Vbox et VMWare
* Versions pour processeurs ARM (utilisables sur matériel type Raspberry Pi)

Toutes ces versions sont disponibles en téléchargement gratuit sur le site officiel et rassemble un peu plus de 600 programmes pré installés a l’instar de :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| logiciel | role | Payant ou non |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

RANK LOGICIEL NOTE TÉLÉCHARGEMENT

1 SNACKNET 10/10 http://www.snacknet.ma

2 PRORAT 9/10

3 SUB7 8/10 -

4 BEAST 8/10 -

5 OPTIX 8/10 -

6 PROAGENT 8/10 -

7 POISON 8/10 -

8 BIFROST 7/10 -

9 C.I.A 7/10 -

10 C-ONE 7/10 -

Nmap : Le plus célèbre scanner de ports libre distribué par Insecure.org. Il est conçu pour détecter les ports ouverts, identifier les services hébergés et obtenir des informations sur le système d'exploitation d'un ordinateur distant.

Wireshark : Analyseur de paquets libre et gratuit. Il est utilisé dans le dépannage et l’analyse de réseaux informatiques

Metasploit : programme open source qui fournit des informations sur les vulnérabilités de systèmes informatiques et les exploite.

Burp Suite : application utilisé pour la sécurisation ou les tests d’intrusion des applications web.

C'est vraiment la boîte à outils principale d'un pentesteur. Nous allons d'ailleurs nous attarder plus longuement sur ce système d'exploitation un peu plus loin dans le cours.

Windows

Le système d'exploitation de Microsoft Windows 7, 8 ou 10 peut également être utilisé. Dans la grande majorité des cas, une version équivalente des outils de Kali Linux est disponible en version Windows. C'est évidemment le cas pour les plus connus d'entre eux :

NMAP

PuTTY

Metasploit Framework

Burp Suite

OWASP Zed Attack Proxy

Nessus

John the Ripper

Étant donné que des versions similaires et en open source gratuit existent sur Kali Linux, nous n'utilisons pas de système d'exploitation Microsoft dans ce cours.

Android

Les outils sur plateforme mobile sont parfois très pratiques quand on a besoin de tester de manière itinérante. Des outils très performants et très bien conçus sont disponibles pour l'utilisation sur Android :

zANTI

FaceNiff

AndroRAT

cSploit

Sachant que les outils installés sur Kali Linux couvrent les fonctionnalités de ses applications Android, nous ne les verrons pas dans ce cours.

Bon nombre sont gratuit , cependant les avantages sont bien plus accrues pour les solutions payante.

En résumé

La distribution la plus adaptée pour les tests d'intrusion est Kali Linux.

Les principaux outils utilisés par les pentesteurs sont souvent disponibles sur les 3 plateformes Windows, Linux, et Android.

« Une imperfection dans un composant ou un système qui peut conduire à ce qu’un composant ou un système n’exécute pas les fonctions requises, par exemple une instruction ou une définition de données incorrecte. Un défaut, si rencontré lors de l’exécution, peut causer la défaillance d’un composant ou d’un système ». [1, Glossaire CFTL/ISTQB, p 7]

1. **Erreur**

« Action humaine produisant un résultat incorrect ». [1, Glossaire CFTL/ISTQB, p 8]

##### c. Anomalie

« Toute condition qui dévie des attentes basées sur les exigences de spécifications, documents de conception, documents utilisateurs, standards etc., ou des perceptions ou expériences de quelqu’un. Les anomalies peuvent être trouvées pendant, mais pas uniquement, les revues, tests, analyses, compilations ou utilisation des produits logiciels ou de la documentation applicable. [1, Glossaire CFTL/ISTQB, p 2].

Une **anomalie** (ou **défaillance**) est un comportement observé différent du comportement attendu ou spécifié. Exemple. Le 4 juin 1996, on a constaté un défaut de fonctionnement. En effet, le 23 juillet, la commission d'enquête remet son rapport : La fusée a eu un comportement nominal jusqu'à la 36ème seconde de vol. Puis les systèmes de référence inertielle (SRI) ont été simultanément déclarés défaillants. Le SRI n'a pas transmis de données correctes parce qu'il était victime d'une erreur d'opérande trop élevée du "biais horizontal". Les raisons étaient multiples à savoir :

* Un bout de code d’Ariane IV (concernant le positionnement et la vitesse de la fusée) a été repris dans Ariane V,
* Il contenait une conversion d’un flottant sur 64 bits en un entier signé sur 16 bits,
* Pour Ariane V, la valeur du flottant dépassait la valeur maximale pouvant être convertie,
* Défaillance dans le système de positionnement,
* La fusée a “corrigé” sa trajectoire,
* Suite à une trop grande déviation, Ariane V s’est détruite.

Un défaut est différent d’une anomalie

Exemple : Une anomalie (telle une maladie) trouve toujours son explication dans un défaut (agent pathogène) et un défaut (un microbe latent) ne provoquera pas nécessairement une anomalie

En informatique, Un **test** désigne une procédure de vérification partielle d’un système ; son objectif principal est d’identifier un nombre maximum du comportements problématiques du logiciel. Il permet ainsi que les problèmes identifiés soient corrigés pour assurer la qualité.

Un test est un ensemble de cas à tester (état de l'objet à tester avant exécution du test, actions ou données en entrée, valeurs ou observations attendues, et état de l'objet après exécution), éventuellement accompagné d'une procédure d'exécution (séquence d'actions à exécuter). Il est lié à un objectif.

La réalisation d'un test amène donc à définir cet ensemble. Différents types de test permettent de détecter différents types de défaut. Des méthodes de spécification de test ont été élaborées pour permettre une plus grande rigueur dans cette activité de définition.

Un test vise à mettre en évidence des défauts de l'objet testé. Cependant, il n'a pas pour finalité de les corriger.

La définition d'un cas à tester précise les exigences s'appliquant à une spécification. Un objet ne peut être testé que si on peut déterminer précisément le comportement attendu en fonction des conditions auxquelles il est soumis. Si la spécification ne permet pas cette détermination, la propriété du logiciel qu'elle définit ne peut être testée.

Soumettre la spécification à cette contrainte de « testabilité » permet d'en améliorer la précision puisqu'elle oblige à expliciter les caractéristiques de l'objet. Ceci permet, en retour, de trouver plus tôt les erreurs de spécification. Cette contrainte est renforcée par certaines méthodes de développement comme le Test-Driven Development. L'ISTQB souligne le rapport de cette contrainte à la « maintenabilité » de l'objet.

L'activité de test d'un logiciel utilise différents types et techniques de tests pour vérifier que le logiciel est conforme à son cahier des charges ou ses spécifications (vérification du produit) et aux attentes du client (validation du produit). Elle est un des processus du développement de logiciels.

#### 2.lEs sourcEs du bug d’un logiciel

Les bugs informatiques proviennent des erreurs humaines telles qu’une incompréhension de besoin et des exigences de l’organisation commanditaire, le manque de savoir-faire, l’absence de compétences informatiques pour comprendre l’environnement de développement et de l’exécution, la complexité des plates-formes, et aussi une faible performance des systèmes informatisés.

La vérification du logiciel a pour but de démontrer que les produits logiciels issus d'une phase du cycle de développement sont conformes aux spécifications (incluant les exigences légales et réglementaires) établies lors des phases précédentes.

Elle a également pour but de détecter et rendre compte des fautes qui peuvent avoir été introduites au cours des phases précédentes la vérification.

La vérification du logiciel est composée des tests, partie prédominante pour les logiciels de faible taille, des activités de revues et d'analyse. Ces activités peuvent dans certains cas remplacer certains tests (exemple : un test qui ne pourrait pas être réalisé sans détérioration d'un composant matériel).

#### 3.Qualification de la gravite des anomalies

Il existe plusieurs niveaux d’anomalies illustrés sur ce tableau ci-dessous

*Tableau 1.1 : Classification des anomalies par niveau de gravité*

|  |  |
| --- | --- |
| **Niveau** | **Description** |
| **Mineure** | Une fonctionnalité non essentielle présente des dysfonctionnements sans bloquer l’exploitation de l’outil.  La solution peut attendre et les dysfonctionnements seront traités en dernier |
| **Majeure** | Une fonctionnalité indispensable est partiellement inopérante sans bloquer l’exploitation de l’outil. L’application peut continuer. Une solution de contournement est identifiée qu’il ne  faut pas négliger |
| **Bloquante** | L’anomalie provoque un arrêt du système ou une fonctionnalité indispensable est inexploitable pour des raisons fonctionnelles ou de performance, aucune solution de contournement.  Une anomalie bloquante doit faire l’objet d’un correctif rapide |

Sans nuire à la suite de ce rapport, nous nous permettrons de confondre, par abus de langage, **erreur** et **défaut** (tendance humaine à confondre cause et conséquence).

### Section 2. l’utilité des tests et validation logiciel

#### 1.Exemples de Bugs aux conséquences désastreuse

###### 1.1. Domaine aéronautique : Sonde « Mariner 1 »

En juillet 1962, la sonde Mariner1 (NASA) perd l’itinéraire quatre minutes après son lancement. La cause du problème est une erreur de transcription d’une équation ; une instruction du programme de guidage écrit en Fortran, contenait une virgule à la place d’un point.

###### 1.2. Banque de New York [21 novembre 1985] : pertes financières énormes

Le 21 novembre 1985, pour une variable entière passant malencontreusement de 32768 à 0 l’on a eu des conséquences très négatives à savoir :

* La crise des bons du Trésor : cette crise s’est manifestée par la suspension de la bourse de Nasdaq de New York, les activités ont été suspendus pendant plus de trois heures en raison d’un bugs informatiques. Un incident sans précédent qui met de nouveau en question la fiabilité des échanges électroniques. Le courtage des actions a cessé sur la plate-forme électronique d'environ 16H15 GMT à 19H25 GMT, paralysant l'indice composite Nasdaq.
* Un emprunt de 20 milliards de dollars de la banque au gouvernement pour financer la remise sur pied des échanges électroniques.
* Les intérêts de l’emprunt ont été un fardeau.

###### 1.3. Le Therac-25 [juillet 1985 -avril 1986] : 5 morts

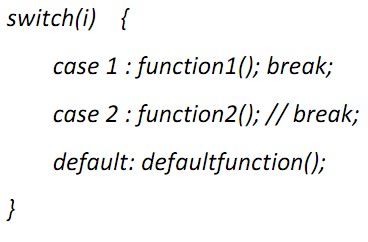
Le Therac-25 est un appareil médical doté d’une interface conviviale pour le traitement des cancers. Il émet un faisceau soit d'électrons (tissus superficiels), soit de photons (très hautes énergies pour les tissus internes). En 1976**,** lepremier prototype utilisé en milieu hospitalier, en 1982**,** on assite à la première vente. En Juin 1985,pour une tumeur à l'épaule, une patiente ressent une très forte chaleur. Elle reçoit entre 15000 à 20000 radiations au lieu de 200. Impossible de reproduire l'incident.

En Juillet 1985, l’on assiste à un phénomène identique pour une tumeur au bassin, le patient décède 5 mois plus tard. L'analyse conclue par une défaillance de micro-commutateurs, en décembre 1985 et mars 1986 : deux nouveaux incidents, 1 mort. L'analyse conclue par la présence de chocs électriques dus à un court-circuit. En Avril1986, un nouvel incident est produit et déclare un nouveau mort, mais le second pour un opérateur qui peut alors reproduire l'erreur. La société reconnaît l'erreur logicielle. L'erreur est corrigée. Mais en Janvier 1987,un dernier incident se produit et fait un dernier mort. De nouvelles erreurs logicielles sont découvertes ce qui conduit à l’arrêt d'exploitation du Therac-25. [Glossaire CFTL/ISTQB]

###### 1.4. Le crash d'AT&T

Lundi le 15 janvier 1990, près de la moitié des communications interurbaines qu’AT & T assure ont été en panne pendant neuf heures, 40 % des communications longue distance assurées par la compagnie aux Etats-Unis ont été coupées. AT & T assurant environ 70 % des communications intérieures américaines, cela signifie que 30 % environ des appels interurbains n'ont pu aboutir. Un bug a perturbé un logiciel, entrainant une congestion d'un central à NewYork s'est ensuite propagée par effet de dominos dans le reste du pays.

Le 15 janvier 1990, un patch devant réduire des temps de réponse.



*Figure 1. 1. Patch devant réduire des temps de réponse*

Source : bout de code issu du document Test et validation logiciel

La figure ci-dessus présente un patch devant réduire des temps de réponse mais dont le **break ;** a été oublier ainsi pour une utilisation pendant 9h on a connu :

* 6000 communications interrompues
* 70 000 000 de coups de fil qui n'ont pu être passés.
* La réputation d'AT&T entachée

En 1992 les ambulances de Londres sont dirigées par un logiciel fautif, ce qui entraine des pertes en vie humain ; en effet, en 1974, la Ligue arabe a commandé un système d'expédition assisté par ordinateur qui est resté inutilisé pendant 13 ans parce que les membres du syndicat ont refusé de le faire fonctionner. Un système de remplacement des tests d'acceptation échoué en 1990 et un autre système de remplacement a été conçu et ordonné. Le 26 Octobre 1992, le

League arabe a commencé à utiliser la nouvelle répartition assistée par ordinateur système

(CAD), connu sous le nom LASCAD. Mal conçu et mis en œuvre, son introduction a entraîné des retards importants dans l'attribution des ambulances, des rapports anecdotiques des temps d'attente de 11 heures. Une enquête ultérieure n'a trouvé aucune preuve pour étayer les allégations syndicales que jusqu'à 30 personnes pourraient avoir péri à la suite de l'accident. L'accident a coïncidé avec des centaines d'exceptions salle de contrôle des messages relatifs aux alertes que les équipes qui ont répondu à des situations d’urgence n'avaient pas signalé, et l’ambulance n’avait pas bougé de 50 mètres à moins de 3 minutes d'expédition. Alors le chef de la direction, John Wilby, a démissionné peu de temps après.

###### 1.5. Télévision : L’émission " TF1 Danse Avec les stars "

Après le bug du direct le 13 octobre 2014, TF1 va rembourser une énorme somme d’argent.

L’animateur de l’émission, "Vincent Cerrutti" annonce "un souci technique" : Un bug informatique qui empêche de comptabiliser le vote du public. TF1 remboursera les téléspectateurs. [**Écrit par Benjamin Guy]**.

A lasuite de ces différents cas de bugs énumérés et sachant que bien de cas graves n’ont pas été cités ici, l’on est en droit de se poser des questions : y’a-t-il un moyen de détecter ces anomalies ? Quels sont les différents outils et les méthodes pour réaliser des tests des applications et détecter les défauts ?

Effectivement il existe des moyens pour détecter des défauts et réaliser des tests, ces moyens, peuvent être automatiques ou manuels.

#### 2.Nécessite de la VVT (Validation, Vérification et Test logiciel)

Il est nécessaire d’assurer la fiabilité des logiciels dans les domaines critiques pour atteindre une très haute qualité imposée par les lois/normes/assurances et dans d’autres domaines pour atteindre le rapport qualité/prix jugé optimal (attentes du client) ; en effet, assurer la fiabilité du logiciel est une part cruciale dans plus de 50% du développement d’un logiciel critique et plus de 30% du développement d’un logiciel standard.

Plusieurs études ont été conduites et indiquent qu’entre 30 et 85 erreurs sont introduites par portion de 1000 lignes de code produites (Logiciel standard), ces chiffres peuvent fortement augmenter si les méthodes permettant d’assurer la fiabilité sont mal gérées au sein du projet ; voir la part importante de l’activité de tests dans les différentes méthodes de conception de logiciels.[1]

**Dans le cycle de vie du logiciel**, on assure la fiabilité / qualité du logiciel par l’activité de Vérification et de Validation. **La Vérification** est l’action de confirmer la satisfaction des exigences spécifiques via l’étude et la mise à disposition de preuves objectives et la **Validation** est un moyen de confirmer le respect d’exigences déterminées pour une utilisation spécifique prévue.

Pendant de développement d’un produit logiciel, plusieurs questions sont posées à savoir : Faisons-nous le travail attendu ? A-t-on décrit le « bon » système ? Est-ce que le logiciel fait ce que le client veut ? Les tests ont nécessairement un rôle prépondérant dans les méthodes agiles. Le test a pour objectifs de détecter d'éventuels écarts entre le comportement attendu et le comportement observé au cours des tests, ce qui élimine un grand nombre de fautes présentes dans le logiciel afin d'obtenir la confiance nécessaire avant l'utilisation opérationnelle. Il faut cependant noter que le nombre de fautes détectées ne peut pas être considéré comme un critère de réussite des tests. Le retour d'expérience montre en effet qu'à complexité technique et industrielle constante, un grand nombre d'erreurs détectées par rapport à d’autres projets « de référence » peut seulement être interprété comme l'indicateur d'un logiciel contenant un très grand nombre de fautes et non comme l'atteinte d'un bon taux de détection des fautes présentes. Il est donc très difficile d'avoir confiance en un logiciel ayant un grand nombre de fautes détectées par le test.

#### 3.Aperçu de l'automatisation des tests

L'automatisation des tests recouvre différents aspects. Nous limiterons notre vision de l'automatisation dans ce syllabus comme étant l'exécution automatique de tests fonctionnels, conçus au moins d'une certaine manière pour simuler un être humain exécutant des tests manuels. Il existe de nombreuses définitions différentes (voir le syllabus ISTQB de niveau avancé Automaticien de Tests) ; Cette vision de l’automatisation est celle qui est la plus adaptée à ce syllabus. Alors que l'exécution des tests est largement automatisée, l'analyse des tests, la conception des tests et leur implémentation sont généralement effectuées manuellement. La création et le déploiement des données utilisées lors des tests peuvent être partiellement automatisés mais se font souvent manuellement. L'évaluation de la réussite ou de l'échec d'un test peut faire partie de l’automatisation, via un comparateur intégré à l'automatisation, mais ce n’est pas toujours le cas. L'automatisation nécessite la conception, la création et la maintenance de différents niveaux de testware, y compris l'environnement dans lequel les tests seront exécutés, les outils utilisés, les bibliothèques de code qui fournissent les fonctionnalités, les scripts de test et les harnais de test, ainsi que les structures de log et de reporting pour évaluer les résultats des tests. Selon les outils utilisés, la surveillance et le contrôle de l'exécution des tests peuvent être une combinaison de processus manuels et automatisés. Il peut y avoir de différents objectifs poursuivis avec l'automatisation des tests fonctionnels, tels que :

* Améliorer l'efficacité des tests en réduisant le coût de chaque test. ;
* Tester plus et d'autres aspects que ceux que nous pourrions tester manuellement ;
* Réduire le temps nécessaire à l'exécution des tests ;
* Pouvoir pousser plus de tests plus tôt dans le cycle de vie du développement logiciel pour trouver et supprimer les défauts du code plus tôt (ce qui est appelé "shift left" pour le test) ;
* Augmenter la fréquence d'exécution des tests.

### Section 3. Vue panoramique des solutions de tests logiciel

Après avoir effectué une recherche et une analyse sur les outils de tests et les nouvelles pratiques, nous avons sélectionné les plus utilisés, ensuite nous avons effectué une analyse comparative : la plateforme de ces outils, les technologies, leurs licences dans la gestion des tests ; les différences, Ces détails sont illustrés sur le tableau ci-dessous :

|  |  |  |  |
| --- | --- | --- | --- |
| **Outils** | **Technologies/Langages** | **Avantages** | **Inconvénients** |
|  | Java, JavaScript, PHP, python, Open-Source | -Plusieurs navigateurs  -Réexécution des scripts.  -Pas d’interruption avec la sélection de systèmes  de build | -Scripts pas conviviaux  -Difficiles à modifier  -Pas de liaison de langage |
|  | Ruby, Open-Source | -Facile à utiliser, -Multi-navigateur | -Uniquement les tests Ruby  -Test mobiles pas améliorés  -Navigateurs de bureau identiques aux Web. |
|  | Node.js, Open-Source | -Extension des fonctionnalités  -Multi-navigateur.  -Avantages pour Angular | -Débogage difficile  -Faible performance  -Prise de temps |
|  | Python, JMV, Open-  Source | -Facile à installer  -Flexibilité des tests  -Gratuite | -Très sobre  -Mise en péril de la lisibilité  -Compliqué à prendre en main, |
|  | Groovy, Java,  Propriétaire, gratuite | -Facilement intégré aux applications tierces  -Gratuite  -Facile à manipuler | -Design pas convivial  -Interface mobile difficile à configurer  -IPhone non reconnue |
|  | Visual Basic,  Propriétaire | -Facile à prendre en main  -Simplifier l'écriture de scripts  -L’auto-complétions prévue | -Payant  -Gourmand en ressources  -SE Microsoft uniquement |
|  | Javascript, Python,  Propriétaire | -Interface utilisateur simple  -Le test de mots-clés facile  -Service client est de premier ordre | -Bogues difficiles à résoudre. -Manque de certains domaines --Pas facile à peaufiner. |
|  | C#, VB.Net,  Propriétaire | -Tests stables  -Faciles à maintenir  -Tout est automatisable | -Coûts élevés  -Complexe  -Difficile à maintenir |

*Tableau 1.1 : Caractéristiques principales des solutions de tests existantes*

Le tableau 1.1 présente les différents logiciels de tests qui existent avec pour chacun des caractéristiques bien définies, nous remarquons que la majorité des logiciels sont multiplateformes écris avec des langages différents et des licences propriétaires et open-sources. Les avantages et inconvénients de chaque logiciel seront illustrés plus tard dans le document.

Dans ce chapitre, nous avons expliqué la revue de la littérature en présentant les concepts et définitions des tests logiciels, ensuite nous avons montré la nécessité de faire les vérifications et validations logiciels et enfin nous avons présenté un tableau dans lequel nous avons recenser les différents types de logiciels existants avec des caractéristiques spécifiques.

## Chapitre II: Cadrage du projet

Ce chapitre est réservé au cadrage du projet qui consiste à définir le contexte du projet et étudier le processus de test existant au sein de l’entreprise ensuite décrire le cahier de charge dans lequel l’on expliquera les besoins, les délais et les coûts du projet, et enfin présenter la méthodologie et les outils que nous envisageons d’utiliser.

### Section 1. Contexte du projet Et étudE dE l’Existant

#### 1.Contexte du projet

L’entreprise Smart Code Group est en réalité une société de services et d’ingénierie logicielle. Elle a été le cadre qui a favorisé la naissance de cette idée de projet parce qu’elle permanemment confrontée au problème général de la qualité du logiciel produit mis à la disposition des clients. En effet, malgré les tests manuels effectués d’une façon ou d’une autres, il arrive fréquemment que le retour des utilisateurs signale encore des bugs.

Comme tout entreprise digne de ce nom, **Smart Code Group** utilise en son sein des outils de collaborations et de planifications professionnelles pour mener à bien ses objectifs. Ces outils sont **Asana** et **Slack**

**Asana** est un gestionnaire de communication en ligne qui prend en charge plusieurs fonctionnalités notamment les espaces de travail, des projets, des tâches, des étiquettes, des notes, des commentaires, etc…. Il permet à l’équipe de gérer leur projet et leurs tâches en collaboration.

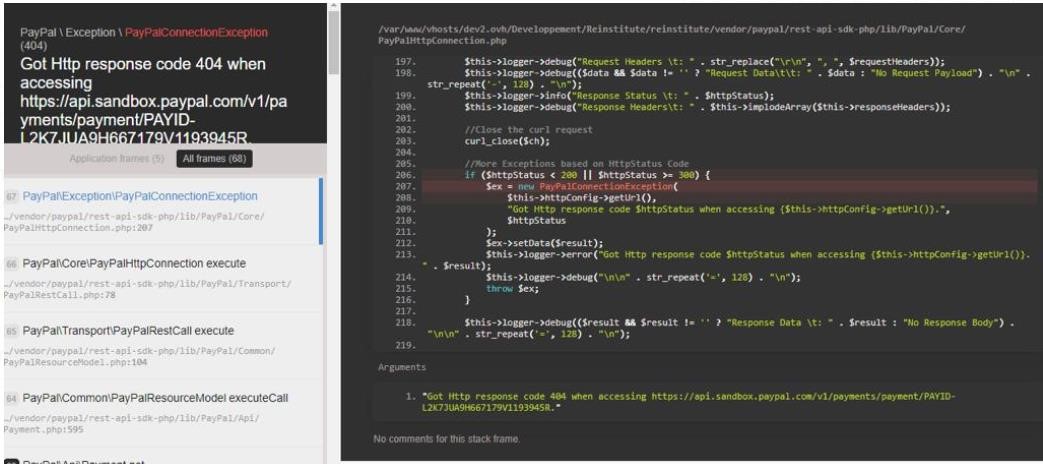
**Slack** quant à lui est une plateforme de communication collaborative en ligne permettant de faciliter les échanges entre les membres de l’équipe.

#### 2.Processus de test existant

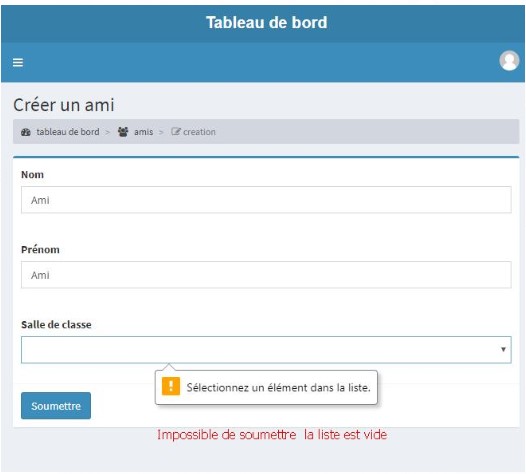
Les tests logiciels sont faits à **Smart Code Group** de façon manuels c’est-à-dire qu’une personne est spécialement recrutée pour effectuer des tests logiciels ; en effet, le processus de test se faisait au moment de la description des besoins c’est-à-dire qu’à partir du cahier de charges, l’on compare la fonctionnalité obtenue à celle attendue en entrant des jeux de données, si la fonctionnalité répond à la description, l’on valide et passe à la prochaine fonctionnalité ; par contre si elle affiche un bug, l’on créé une tache sur Asana dans laquelle on décrit la nature du bug.

Prenons un exemple d’une application développée par Smart code Group : [www.risingexcellenceinstitute.com;](http://www.risingexcellenceinstitute.com/) application web bilingue (Français et anglais) permettant de gérer les services offerts par un institut et gérer ses clients (parents, étudiants/élèves, professeurs), capable de s’afficher sur tous les types d’appareils (ordinateurs, tablettes, téléphone mobile).

Les erreurs que l’on rencontrait après les tests sont les suivantes :

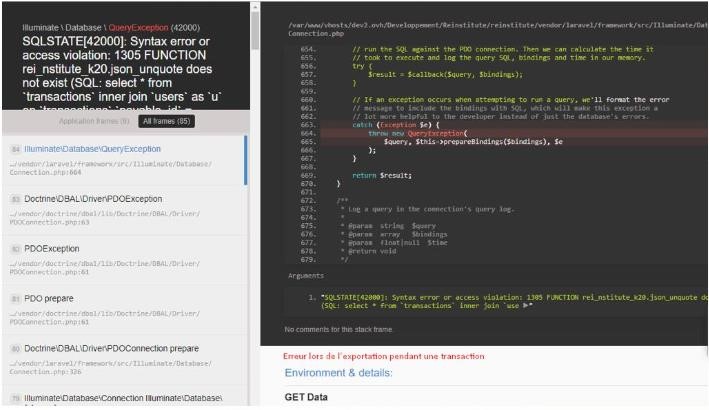


Le payement avec PayPal renvoie une page d’erreur symphonie.

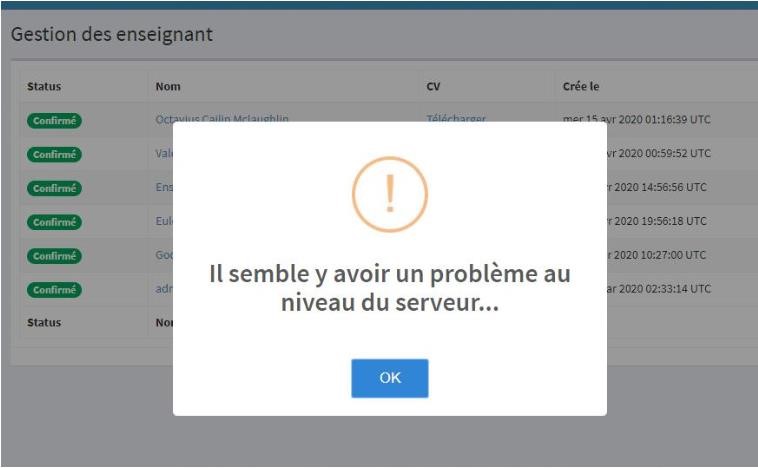


Après la création du compte étudiant il n’est pas redirigé vers son tableau de bord comme décrit dans le cahier de charge, il renvoi une page vide ; impossible d’ajouter un ami car la liste déroulante est « Salle de classe » est vide.





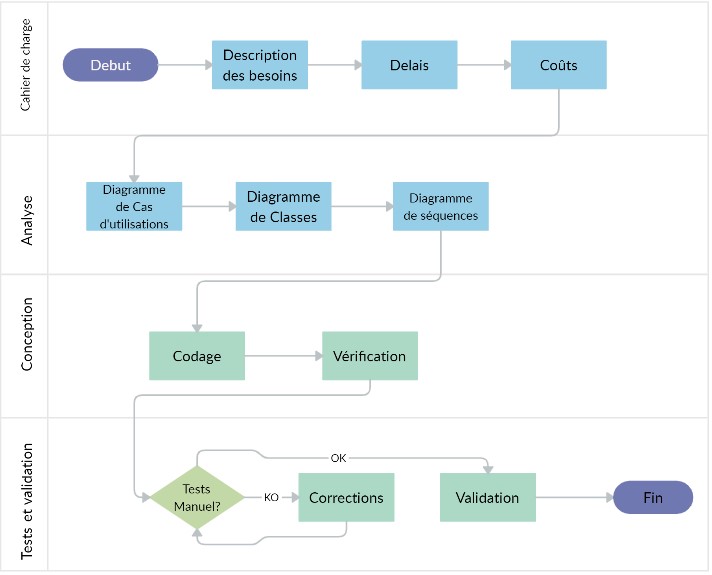
Lorsqu’on clique sur exporter pour filtrer la liste des transactions sans préalablement sélectionner les dates l’on est redirigé vers une page d'erreur.



Impossible de supprimer un enseignant, la suppression renvoie un Pop-up d’erreur.

Les tests manuels peuvent se dérouler de deux façons différentes : via des tests exploratoires ou des tests avec scénarios en fonction de la pertinence du logiciel qui est développé :

* **Tests avec Scénario** : dans cette approche, l’on définit des parcours au préalable à l’aide du cahier de charge et les suit pour contrôler la qualité de l’application sur des problématiques bien précises.
* **Test exploratoire** : pour ce test l’on navigue librement dans l’application en remplissant les champs des formulaires par exemple, pour y déceler le maximum de bugs et de problèmes.



*Figure 2. 1: Processus de test manuel*

La figure 2.1 présente le processus de test effectué à Smart Code group avant optimisation conçue avec la norme BPMN 2.0

#### 3.Constat

Le test manuel présente des avantages dont le principal est qu’il permet de tester l’UI

(User interface) et l’UX (User eXperience) du logiciel qui sont l’affichage correct du texte, les liens, les images, etc... Cela permet de déceler des bugs qui serraient visibles par des utilisateurs mais n’auraient pas pu être repéré par un robot.

Pendant l’opération de test, lorsque l’ingénieur test découvre un bug, il crée une tâche dans la plateforme **Asana** et l’attribut à la personne responsable de la fonctionnalité, cette dernière reçoit automatiquement un mail professionnel la notifiant de la tâche créée. Ladite tâche contient une description du bug avec une ou plusieurs images à l’appui.

Si le codeur ne comprend toujours pas la nature du bug, il le notifie alors sur Slack et l’ingénieur testeur pourra lui donner plus amples informations.

Les tests manuels deviennent lourds et fastidieux au bout d’un certains moments. Imaginons que nous avons une dizaine de logiciels à développer avec un ou deux testeurs : ils laisseront passer beaucoup de bugs car plusieurs tâches seront répétitives alors qu’ils peuvent les répliquer aussi facilement.

A partir de ce constat, l’on peut penser à la possibilité d’envisager une automatisation fitelle partielle de ce processus de tests et validation logicielle.

### Section 2. Cahier de charge

#### 1. Justification dE l’étudE

Dans un contexte ainsi dominé par l’informatisation dans les différents secteurs d’activités, l’un des soucis majeurs des éditeurs ou encore des sociétés de services et d’ingenierie informatique est de mettre au point des solutions logicielles, exempt d’erreurs, qui produisent des résultats attendus dans les délais prévus. Mais dans beaucoup d’entreprise éditrice de logiciels comme **Smart Code Group** c’est au moment de la mise en œuvreou de l’utilisation d’un produit logiciel que l’on se rend compte que certains **« Bugs »** existent encore et empêchent le bon fonctionnement du logiciel pendant son utilisation. Cette situation pose des désagréments graves à l’endroit des clients c’est-à-dire des organisations et entreprises qui ont bien voulus se doter de ces logiciels ceci entraine des préjudices graves autant pour les clients que pour les fournisseurs de services tel que **Smart Code Group.**

L’automatisation des tests est une solution qui permet de réduire la charge de travail et le temps passé à exécuter des tests de logiciels. Il est par exemple utile d'automatiser les tests fonctionnels et de non-régression. Le test logiciel automatisé est bien souvent complémentaire aux tests manuels et exploratoires, etc… En plus de faire gagner du temps à l’équipe de développement et ainsi aider à réduire le délai de livraison, l'automatisation des tests fonctionnels présente également le grand avantage de considérablement réduire la marge d'erreur des tests, une fois qu’on a fait le choix d'automatiser les tests, la mise en œuvre peut s'avérer être un vrai défi. En effet, ce qui fonctionne aujourd’hui peut ne plus fonctionner demain, et le problème s’aggrave en présence d’une architecture fragile ou d’un manque de compétences adaptées.

**Problématique**

Au regard de ces situations, nous nous sommes posé la question centrale suivante :

Comment procéder à des tests pouvant garantir la fiabilité d’un produit logiciel avant son déploiement chez un client ? Cette question centrale nous amène à nous poser deux autres questions auxquelles les réponses contribueraient certainement à résoudre le problème central posé à savoir : quelles sont les tests qui doivent impérativement se faire lors de l’ingénierie logicielle et comment peut-on automatiser les tests logiciels afin de prévenir et éradiquer tout anomalie de fonctionnement du produit logiciel final.

En outre, l’automatisation des tests ne vient pas empêcher les bugs mais plutôt permettre de les élaguer facilement. Les tests à Smart Code Group sont faits uniquement de manière humaine mais cette façon de faire n’est pas toujours optimale d’où la nécessité d’envisager l’automatisation des tests.

**Intérêt et objectifs**

L’intérêt de notre travail se perçoit clairement en ce qu’un bon service rendu, un produit logiciel qui rencontre la satisfaction du client, une application informatique qui assure une bonne continuité des activités en garantissant de bons résultats dans les meilleurs délais et avec sécurité maximale, sont autant de facteurs qui contribuent à la fidélisation des clients, à la notoriété de la société éditrice ou du fournisseur et donc du gain de chacune des parties

L’objectif principal de notre travail est de développer des scripts logiciels paramétrable capable de réaliser différents tests selon des processus qui seront définis dans le but de corriger toutes les éventuelles erreurs et exceptions d’un produit logiciel pour sa mise au point. Cet objectif principal se décline dans les objectifs spécifiques suivantes :

* Inventorier tous les tests nécessaires en assurance qualité d’un logiciel
* Définir le processus et les différentes étapes des tests logiciels
* Modéliser la solution informatique envisagée,
* Réaliser l’outil logiciel permettant d’automatiser une application types de smart Code Group
* Envisager une standardisation de cet outil logiciel pour qu’il puisse tester automatiquement d’autres logiciels

#### 2.Spécifications Du Projet

**Spécifications fonctionnelles**

Les besoins fonctionnels peuvent d’exprimer en quelques grands points :

* **Authentification à la solution**

Cette fonctionnalité est une procédure, par laquelle un système informatique certifie l’identité d’une personne ou d’un ordinateur. Le but de cette procédure étant **d’autoriser** la personne à accéder à certaines ressources sécurisées. Il va comparer les informations des utilisateurs autorisés stockées dans une base de données (en local ou sur un serveur d’authentification) à celles fournies. L’accès sera autorisé seulement si les informations sont identiques. C’est l’administrateur du système d’information qui octroie les droits et paramètre l’accès.

L’utilisateur possédant un compte d’accès (identifiant + mot de passe) n’aura accès qu’aux ressources dont il est autorisé à voir

* **Création des scripts de tests**

Un script de test dans le test logiciel est un ensemble d'instructions qui seront effectuées sur le système en cours de test pour vérifier que le système fonctionne comme prévu. C’est un programme court écrit dans un langage de programmation utilisé pour tester une partie de la fonctionnalité d'un système logiciel. Les scripts de test écrits en tant que programme court peuvent soit être écrites à l'aide d'un outil de test de l’interface graphique fonctionnelle spéciale automatisée ou dans un endroit bien connu (langage de programmation). Comme indiqué dans la norme IEEE, ISO et la CEI.

* **Connection à la plateforme Slack**

Slack est l’outil principal de collaboration et de communication utilisé à Smart Code Group, la liaison de notre logiciel à Slack permettra de réduire le besoin de passer d’un outil à un autre, réduire les obstacles à l’information, terminer les tâches urgentes au plus vite, assurer que les projets continuent à avancer, optimiser la prise de décisions

* **Connection à la plateforme Asana**

Asana quant à lui est un outil de planification de tâches permettant de créer les tâches et attribuer les deadlines pour mieux suivre l’avancement des projets, connaitre votre emploi du temps en affichant vos activités sur un calendrier. Les périodes creuses et les conflits y étant aisément repérables, l’on peut rapidement procéder à des ajustements, Consulter la progression de tous les projets en temps réel grâce aux portefeuilles, permettant ainsi d’adopter une approche proactive quant à la gestion des risques et de tenir chaque partie prenante informée.

* **Générer des rapports de tests**

Le rapport de test est un document contenant des informations sur les actions effectuées (analyses de cas, bogues détectés, temps passé, etc.) et les résultats de cette performance (tests ayant échoué / réussi, nombre de bugs et de plantages, etc.). Le rapport de test bien fait permet d’évaluer l’état actuel du projet et la qualité du produit. Il est possible de prendre des mesures correctives si nécessaire, le rapport de test peut être le document final qui détermine si le produit est prêt à être publié ou utilisé. En effet, la qualité et la transparence sont les conditions préalables indispensables à la création d’un rapport de test, parce que c’est la clé visible pour la métrique client de notre évaluation de travail.

* **Paralléliser les tests**

Le parallélisme consiste à mettre en œuvre des architectures permettant de traiter des informations de manière simultanée, ces techniques ont pour but de réaliser le plus grand nombre d'opérations en un temps le plus petit possible

* **Uploader et télécharger les rapports de test**

Le terme "**upload**" désigne l'action de transférer un ou plusieurs rapports depuis son ordinateur vers une autre machine (qui peut être un serveur). L'action inverse, qui consiste à transférer des rapports depuis le serveur ou la machine distante vers son ordinateur de travail, est nommée "**download**" ou « **téléchargement** ».

* **Intégrer l’outils de capture d’écran.**

L'outil Capture permet de faire une capture d'écran d'une fenêtre, d'une zone rectangulaire, d'une zone de forme quelconque ou de l'écran entier. Les captures peuvent être par la suite annotées grâce à un feutre interactif et sauvegardées au format .png, .gif, .jpg, ou encore être envoyées par mail.

**Les spécifications techniques**

**Environnement logiciel**

* Un système d’exploitation Windows ou Linux ❖ Des IDE : VS Code, Eclipse, etc...
* Des outils de travail collaboratif : Slack, Asana, Trello
* Un SGBD MySQL
* Framework Selenium

**Environnement Matériels**

* Un ordinateurs PC Core I5 avec 6 giga-octets de mémoire vive et 500Go de capacité ;
* Un modem pour connexion internet ;
* Un bureau de travail ;
* De l’électricité et une rallonge électrique.

#### 3.Planification

Compte tenu des contraintes de temps liés à ce travail, l’étude, l’analyse, la conception, réalisation et la mise en œuvre de ce projet s’étaleront sur quatre (04) mois. La date de début est fixée au 10 Mars 2020 et la date de fin de mise en œuvre de la première version est fixé au 15 Juillet 2020.

Le tableau suivant montre la disposition des différentes activités dans le cycle de vie du projet.

Tableau 2.1: *Tableau de répartition des tâches*

|  |  |  |
| --- | --- | --- |
| **Taches** | **Libellé** | **Antécédant** |
| A | Installation de l’environnement de travail | --- |
| B | Découverte du fonctionnement de test logiciel | --- |
| C | Etude relative à la mise en place du projet | A, B |
| D | Etude de l’existant | B |
| E | Conception de la solution | C, D |
| F | Architecture du système | D, E |
| G | Création des scripts | E, F |
| H | Intégration d’outils de capture d’écran | G |
| I | Connection à Slack | H |
| J | Connection à Asana | H |
| K | Génération des rapports de test | G, H |
| L | Uploade et téléchargement des rapports de tests | K |

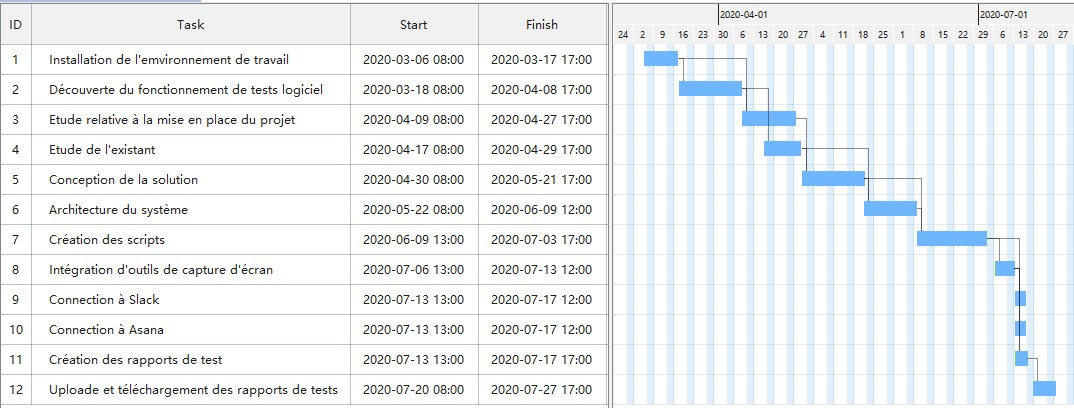
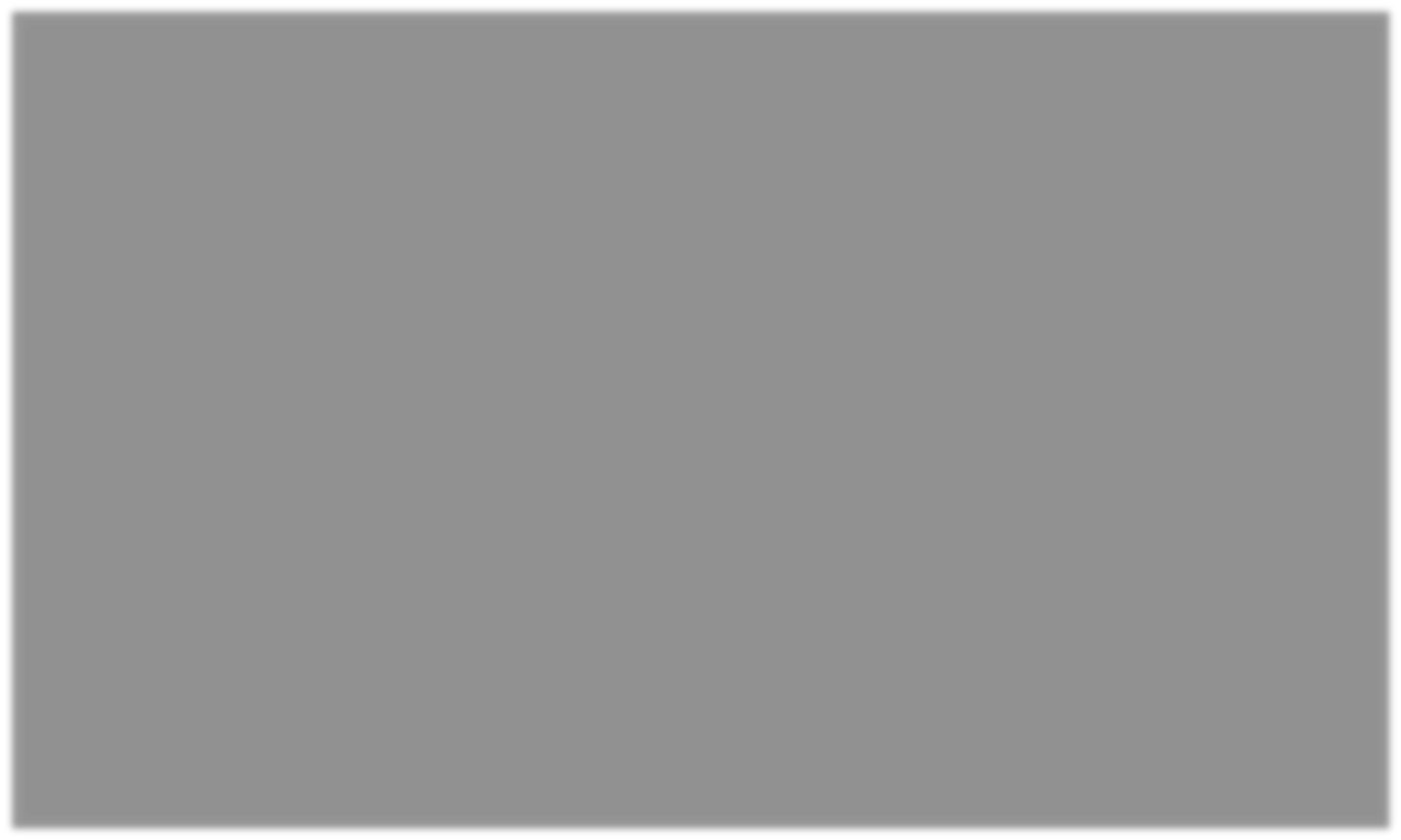
**Diagramme de Gantt**

**Le diagramme de GANTT** est un outil permettant de modéliser la planification des tâches nécessaires à la réalisation d’un projet, étant donné la relative facilitée de lecture des diagrammes GANTT, cet outil est utilisé par la quasi-totalité des chefs de projet dans tous les secteurs. Le diagramme de GANTT représente un outil pour le chef de projet, permettant de représenter graphiquement l’avancement du projet, mais c’est également un bon moyen de communication entre les différents acteurs d’un projet.

*Figure 2. 2:*

*Succession de t*

*âches avec Gantt*



Sources : Par nos soins

**Evaluation des coûts**

Le **CFTL** (Comité Français de Test Logiciel) et **L’ISTBQ** (International Software Testing Qualification Board) évaluent les coûts des logiciels en fonction de la criticité de celui-ci :

Le coût d’un **logiciel critique** est estimé à plus de cinquante pour cent (50%) du coût du développement du dit logiciel.

Le coût d’un **logiciel standard** est évalué à environ trente pour cent (30%) du coût total de développement. [Glossaire CFTL/ISTQB]

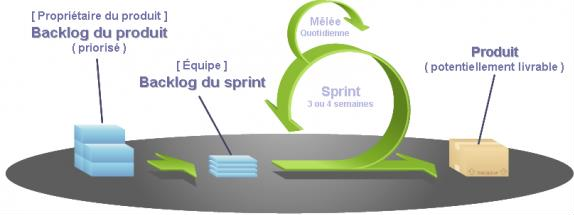
### Section 3. Méthodologie et outils envisagés

Les méthodes utilisées au sein de Smart Code Group sont les méthodes agiles, ce sont des groupes de pratique, de pilotage et de réalisation de projet ; ces méthodes se veulent plus pragmatiques que les méthodes traditionnelles et surtout elles impliquent au maximum le client et permettent une grande réactivé à ses demandes.

#### 1.Méthodologie de travail : scrum

**Scrum** est la méthode utilisée à Smart Code Group ; l'approche Scrum suit les principes de la méthodologie Agile, c'est-à-dire l'implication et la participation active du client tout au long du projet, Considéré comme un cadre (framework en anglais) de gestion de projet, Scrum se compose de plusieurs éléments fondamentaux : des rôles, des événements, des artefacts, des règles. Il s'agit d'une approche empirique (c'est-à-dire qui se base sur l'expérience), dynamique et participative de la conduite du projet. Au rugby, la mêlée est une phase indispensable car elle permet au jeu de repartir sur d'autres bases. Même chose pour Scrum : l'équipe se réunit quotidiennement lors d'une réunion de synchronisation, appelée mêlée quotidienne, afin de suivre l'avancement du projet

L'équipe Scrum est auto-organisée et pluridisciplinaire, c'est-à-dire qu'elle choisit la meilleure façon d’accomplir son travail et qu'elle possède toutes les compétences nécessaires à l'accomplissement du projet. La flexibilité, la créativité et la productivité de l'équipe sont ainsi optimisées. L'équipe Scrum se compose d’un Scrum Master, un Product Owner (ou propriétaire du produit en français), une équipe de développement.



*Figure 2. 3: Présentation du fonctionnement de Scrum*

Source : Wikipédia / Scrum(développement)

* **Product Backlog :** liste des fonctionnalités du produit. (Il ne doit pas nécessairement contenir toutes les fonctionnalités attendues dès le début du projet, il va évoluer durant le projet en parallèle des besoins du client.)
* **Sprint Backlog :** planification des éléments du Product Backlog à mettre en œuvre lors du Sprint pour livrer l'incrément de produit doté des fonctionnalités requises pour cette étape. Le Sprint Backlog n'est pas figé, mais est amené à évoluer durant le Sprint.
* Le Sprint, le cœur de Scrum

Cette approche repose sur des itérations de 2 à 4 semaines. Ce sont **les fameux "Sprints"**. Il s'agit des sous-parties d'un projet. Chaque sprint a un objectif et une liste de fonctionnalités à réaliser.

Quant aux outils envisagés, il s’agit du framework Selenium développer en java ; il permettra d’écrire les tests automatisés en Java, PHP, Python, Ruby, .NET, Perl ; les éditeurs de code utilisés sont Eclipse et Visual Studio Code.

**Eclipse**



*Figure 2. 4: Eclipse*

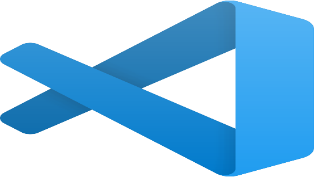
Source : Eclipse Foundation

**Eclipse IDE** est un environnement de développement intégré libre (le terme Eclipse désigne également le projet correspondant, lancé par IBM) extensible, universel et polyvalent, permettant potentiellement de créer des projets de développement mettant en œuvre n'importe quel langage de programmation. Eclipse IDE est principalement écrit en Java (à l'aide de la bibliothèque graphique SWT, d'IBM), et ce langage, grâce à des bibliothèques spécifiques, est également utilisé pour écrire des extensions.

La spécificité d'Eclipse IDE vient du fait de son architecture totalement développée autour de la notion de plug-in (en conformité avec la norme OSGi) : toutes les fonctionnalités de cet atelier logiciel sont développées en tant que plug-in.

Plusieurs logiciels commerciaux sont basés sur ce logiciel libre, comme par exemple IBM Lotus Notes 8, IBM Symphony ou Websphere Studio Application Developer.

**Visual Studio Code**



*Figure 2. 5: Logo de Visual Studio Code*

Source: wikipedia.org/Visual\_Studio\_Code

**Visual Studio Code** est un éditeur de code extensible développé par Microsoft pour Windows, Linux et macOS2. Les fonctionnalités incluent la prise en charge du débogage, la mise en évidence de la syntaxe, la complétion intelligente du code, la refactorisation du code et Git intégrer. Les utilisateurs peuvent modifier le thème, les raccourcis clavier, les préférences et installer des extensions qui ajoutent des fonctionnalités supplémentaires. Le code source de Visual Studio Code provient du projet logiciel libre et open source VS Code de Microsoft publié sous la licence MIT permissive, mais les binaires compilés sont des logiciels gratuits pour toute utilisation.

Notre approche méthodologique consistera tout d’abord à dresser un état de l’art sur ce sujet pour étudier les fonctionnalités et les limites des éventuelles solutions existantes ensuite d’élaborer un cahier de charges, de modéliser la solution envisagée, d’utiliser des méthodes et outils pour réaliser cette solution afin d’atteindre les objectifs fixés.

Les résultats escomptés sont principalement de 02 natures

❖ Les documents de conception, de réalisation et d’utilisation de la solution informatique ❖ L’outil informatique de test proprement dit

Notre mémoire sera organisé autour deux grandes parties : Etat de l’art qui est composé du la revue de la littérature et du l’étude de l’existant puis la présentation des outils de test et leurs rôles, la Conception de la solution qui comprend la modélisation de la solution et discussion des résultats obtenus.

#### 2.Langage de modélisation

L’analyse de ce projet sera faite avec le langage UML 2.0. "UML est un langage de modélisation standard de l'industrie avec des symboles graphiques riches et un ensemble complet de diagrammes et d'éléments. UML (Unified Modeling Language) est un langage de modélisation unifié (créé en 1994), né de la fusion des trois méthodes de modélisation d'objet :

* OMT : Object Modeling Technique (créé par Jim Rumbaugh).
* BOOCH : Nom tiré de son inventeur (Grady Booch).
* OOSE: Object Oriented Software Engineering.

UML est un langage pour visualiser, spécifier, construire et documenter les abstractions d'un système logiciel. C’est un langage formel et normalisé : il permet un gain de précision et de stabilité, c’est un support de communication performant : il permet grâce à sa représentation graphique, d'exprimer visuellement une solution objet, de faciliter la comparaison et l'évolution de solution. Son caractère polyvalent et sa souplesse en font un langage universel.

Un diagramme UML est une représentation graphique, qui permet de modéliser un aspect bien précis du système, chaque type de diagramme UML possède une structure et des concepts prédéfinis. Un diagramme donne à l'utilisateur un moyen de visualiser et de manipuler des éléments de modélisation. Au total UML définit treize types de diagrammes. Mais, nous avons eu besoin d'utiliser seulement deux diagrammes parmi les treize proposés par ce langage.

Le choix de l’UML par rapport à l’autre moyen d’analyse se justifie par le fait que UML facilitera l’implémentation du code source, il nous a permis de bien comprendre le fonctionnement des diagrammes et de maitriser leur usage au sein de notre projet.

Nous allons citer l’utilité de quelques diagrammes :

* Diagramme de cas d'utilisation représente les fonctions du système du point de vue des utilisateurs ;
* Diagramme de classes montre une collection d'éléments statiques (classes), leur contenu (attributs, opérations, types) et les relations entre eux (associations). Permet de décrire la structure statique d'un système.
* Les diagrammes de composants permettent de décrire l'architecture physique et statique d'une application en termes de modules : fichiers sources, librairies, exécutables, etc….
* Les diagrammes de séquence sont une représentation temporelle des objets et de leurs interactions. Ils permettent de bien comprendre le fonctionnement du système ; modéliser la vie des objets dans le temps et leur chronologie. Ils représentent les interactions, les communications entre objets.
* Les diagrammes d'états-transitions représentent le comportement d'un classificateur ou d'une méthode en termes d’états. Le diagramme d'états-transitions sert à représenter des automates d'états finis, sous forme de graphes d'états, reliés par des arcs orientés qui décrivent les transitions. Les diagrammes d'état-transitions permettent de décrire les changements d'états d'un objet ou d'un composant, en réponse aux interactions avec d'autres objets/composants ou avec des acteurs. Une transition représente le passage instantané d'un état vers un autre. Une transition est déclenchée par un événement. En d'autres termes : c'est l'arrivée d'un événement qui conditionne la transition.

Nous allons nous arrêter à ces quelques diagrammes, car la liste est longue.

#### 3.Etude critique des outils les plus utilises

L’objectif ici est d’étudier en profondeur les logiciels de tests qui existent avec chacun sa particularité et ses manquements afin d’effectuer une étude comparative de chacun de ces logiciels.

#### 3. 1.Selenium

Licence : Open-source

**Selenium** est l’un des Framework de test open-source les plus populaires pour diverses applications et plates-formes Web telles que Windows, Mac et Linux. En tant que cadre d’automatisation de choix des testeurs d’automatisation Web, en particulier ceux avec des compétences avancées de programmation et de script, ses scripts peuvent être écrits dans différents langages de programmation tels que Java, PHP, Python, C #, Groovy, Ruby et Perl. Avec une grande flexibilité et différents niveaux de complexité pour les testeurs, Selenium est sans aucun doute la base de la plupart des autres outils de test en général. L’objectif de cet outil est d’enregistrer des scénarios de tests et de les rejouer automatiquement après afin de voir s’il y a des régressions.

**Selenium** est composé d’un ensemble de logiciels différents :

* **Selenium IDE** est un plugin Firefox qui permet d’enregistrer et de lire des interactions avec le navigateur Web. Utilisez cette fonction soit pour dérouler des scripts simples, soit pour accélérer la création de scripts Selenium RC ou Selenium Webdriver.
* **Selenium Server** (anciennement Selenium RC Server) : il est utilisé pour exécuter soit des scripts de style Selenium soit des scripts Remote Selenium Webdriver. La version

2.x remplace l’ancien serveur Selenium RC et est conçue pour être compatible avec votre infrastructure existante.

* **Selenium Client Drivers** : afin de créer des scripts qui interagissent avec le serveur Selenium (Selenium RC, Selenium distance Webdriver) ou afin de créer localement des scripts Selenium WebDriver, vous devez faire usage de pilotes spécifiques aux drivers client. Sauf indication contraire, les drivers intègrent à la fois les pilotes des version 1 et 2
* **Selenium Grid** est un outil qui accélère considérablement les tests fonctionnels des applications Web en tirant parti de votre infrastructure informatique existante. Il vous permet de facilement exécuter plusieurs tests en parallèle, sur plusieurs machines, dans un environnement hétérogène.

Chaque logiciel ci-dessus fonctionne avec une approche différente pour la mise en automatisation des tests. Même si la plupart des ingénieurs QA utilisant Selenium se concentre sur un ou deux outils (car ils répondent globalement aux besoins de leur projet), l’utilisation de tous les outils vous donnera beaucoup d’options différentes pour aborder différents problèmes d’automatisation de tests. La suite complète des outils est composée d’un ensemble de fonctions de tests spécifiquement adaptés aux besoins de tests des applications web de tous types. Ces opérations sont très flexibles, ce qui permet de nombreuses options permettant de localiser les éléments de type UI et aussi de comparer les résultats des tests attendus avec le comportement des applications actuelles. Une des principales caractéristiques de Selenium est qu’il est supporté par de multiples plates-formes des navigateurs Web

**Avantages :**

Selenium peut fonctionner avec une liaison de langue pour aider le développeur en travaillant dans la langue avec laquelle il peut être à l'aise. Il peut s'agir de Java, C #, python, PHP, etc…, Le test avec Selenium peut pratiquement s'exécuter sous plusieurs navigateurs différents, ses scripts sont principalement créés par la manière d'enregistrer les actions à l'aide de l'application Web en cours de test, dans un navigateur. Les scripts de sélénium peuvent être enregistrés et également réexécutés à tout moment. Ces tests peuvent également être créés manuellement à l'aide des outils de développement Web, il n'interrompt ni n'interfère avec la sélection de systèmes de build, d'outils de reporting. On dit que le sélénium s'intègre bien avec tous les outils populaires.

**Inconvénients :**

Selenium ne prend pas en charge les tests exécutés en parallèle, sur un ordinateur, un support que Selenium ne fournit pas est le partage de tests et de résultats autrement que par des moyens manuels, Les scripts de test de Selenium ne sont pas très conviviaux quand on parle de lisibilité, et aussi difficiles à modifier, Selenium ne dispose pas d'installations de gestion des tests. Les scripts de test sont enregistrés en tant que fichiers sans attributs. Par conséquent, pour organiser ces scripts individuels, que ce soit de quelque manière que ce soit via une interface utilisateur supplémentaire, il faut une application personnalisée ou un outil du tiers. Nous ne pouvons pas dire que Selenium est complet en termes d'automatisation car il nécessite également des liaisons de langage, des cadres tiers et tout cela pour être réellement efficace.

C’est à cause de ces nombreux avantages que Test de sélénium est utilisé pour l'interface utilisateur, la régression et les tests d'acceptation. De plus, comme le sélénium permet un développement rapide des tests, il devient un choix populaire. Il s'agit d'un choix populaire parmi les informaticiens qui l'utilisent pour automatiser les tâches répétitives et basées sur le Web de l'administration.

La procédure d’installation de Selenium Webdriver se trouve en annexe 1.

#### 3. 2.Watir

Licence : Open-source

**Watir** est une bibliothèque Ruby open-source et un outil de test d’application Web développé en C # pour les tests d’automatisation Web. Watir prend en charge les tests de navigateur croisés, y compris Firefox, Opera, navigateur sans tête et IE. Il prend en charge les tests de site Web HTML et AJAX, le modèle de page et de contrôle. Watir prend également en charge les tests pilotés par les données et s’intègre aux outils BBD tels que RSpec, Cucumber et Test / Unit

Watir (Web Application Testing In Ruby), est une famille open-source de Ruby bibliothèques pour l’automatisation des navigateurs web. Il pilote Internet Explorer, Firefox, Chrome, Opera et Safari, et est disponible sous forme de gemme RubyGems. Watir a été principalement développé par Bret Pettichord et Paul Rogers.

**Fonctionnalités**

Le projet Watir se compose de plusieurs petits projets. Les plus importants sont watir classique, watir-webdriver et watirspec.

**Watir classique**

**Watir classique** utilise le fait que Ruby a intégré des capacités OLE (Object Linking and Embedding). En tant que tel, il est possible de piloter Internet Explorer par programme. Watir classique fonctionne différemment des outils de test basés sur HTTP, qui fonctionnent en simulant un navigateur. Au lieu de cela, Watir classique pilote directement le navigateur via le protocole OLE, qui est implémenté sur l’architecture COM (Component Object Model).

Le COM permet la communication interprocessus (comme entre Ruby et Internet Explorer) et la création et la manipulation d'objets dynamiques (ce que fait le programme Ruby à Internet Explorer). Microsoft appelle cette automatisation OLE et appelle le programme de manipulation un contrôleur d'automatisation. Techniquement, le processus Internet Explorer est le serveur et sert les objets d'automatisation, exposant leurs méthodes ; tandis que le programme Ruby devient alors le client qui manipule les objets d'automatisation.

**Watir-webdriver**

**Watir-webdriver** est une version moderne de l'API Watir basée sur Selenium.

Selenium 2.0 (selenium-webdriver) se veut l'implémentation de référence de la spécification WebDriver. Dans Ruby, Jari Bakken a implémenté l'API Watir comme un wrapper autour de l'API Selenium 2.0. Non seulement Watir-webdriver est dérivé de Selenium 2.0, mais il est également construit à partir de la spécification HTML, donc Watir-webdriver doit toujours être compatible avec les spécifications W3C existantes.

**Watirspec**

Watirspec est la spécification exécutable de l'API Watir, comme RubySpec l’est pour Ruby

**Avantages :**

Watir offre les avantages suivants : C’est un outil open source et très facile à utiliser, il est développé en Ruby et toute application Web qui fonctionne dans un navigateur peut être facilement automatisé à l'aide de watir, tous les derniers navigateurs sont pris en charge dans Watir, ce qui facilite les tests

**Inconvénients :**

Comme tout autre logiciel, Watir a également ses limites : Watir est pris en charge uniquement pour le framework de test Ruby et il ne peut être utilisé avec aucun d'autres cadres de test, les tests mobiles à l'aide de Watir ne sont pas améliorés et les navigateurs de bureau sont imités pour qu’ils se comportent comme des navigateurs mobiles au lieu d'agir comme des appareils en temps réel.

#### 3. 3.Protractor

Licence : Open-source

**Protractor** est un Framework d’automatisation fonctionnelle open source (également connu sous le nom de framework de test End to End) spécialement conçu pour vérifier la santé des applications web AngularJS. C’est un programme Node.js qui supporte les frameworks de test Jasmine, Mocha et Cucumber. Il utilise Selenium WebDriver pour piloter les navigateurs et simuler l’interaction de l’utilisateur avec une application AngularJS exécutée dans un navigateur. L’attente automatique de Protractor peut automatiquement exécuter l’étape suivante de votre test au moment où la page Web termine les tâches en attente…

Le Framework a été spécialement conçu pour Angular, cela veut dire qu’il intègre des APIs et des mécanismes uniques permettant de tester des éléments spécifiques à Angular sans effort particulier. De plus, nous savons qu’Angular effectue souvent des opérations asynchrones (appel API, détection de changement …) et Protractor est capable d’attendre n’importe quelle tâche avant de continuer l’exécution du test. Protractor n'a besoin que de deux fichiers pour exécuter le premier test, le fichier de spécifications (code de test) et le fichier de configuration. Le fichier de spécification contient le code de test et l'autre contient les détails de configuration tels que le chemin du fichier de spécification, les détails du navigateur, l'URL de test, les paramètres du framework, etc.… timeout, le framework sera ramassé aux valeurs par défaut.

Une particularité de Protractor qui est souvent la raison pour laquelle il est choisi, est sa

faculté à proposer des tests de reconnaissance d’image à l’aide du plugin « checkAppByImage ». Ce plugin est très complet et permet de :

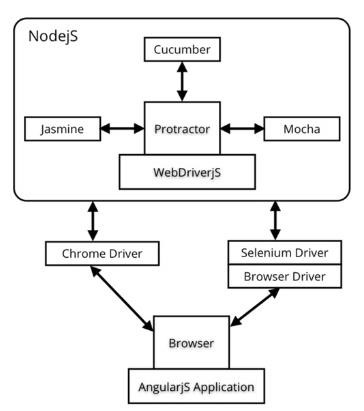
* Comparer une image en entier
* Accepter plus ou moins de différence entre les images (% de pixels différents, la différence de chaque pixel se faisant par rapport à leur couleur)
* Ne pas comparer certaines zones des images (exemple : ne pas comparer des zones de publicité)

**Avantages :**

L’on peut choisir entre les bibliothèques pour écrire les tests. Certains choix populaires sont le jasmin, le moka, le concombre. En dehors de cela, il existe des tonnes de plugins et de bibliothèques pour étendre les fonctionnalités de Protractor, il prend en charge les tests multinavigateurs. Encore plus l’on peut exécuter plusieurs instances de navigateurs simultanément, Protractor offre des avantages supplémentaires pour tester les applications Angular, mais votre application ne doit pas nécessairement l'utiliser.

**Inconvénients :**

Le débogage de Protractor est difficile à réaliser, cela prendrait beaucoup de temps pour maitriser toutes les APIs et techniques. La possibilité de multiplication des images (espace de stockage et de classement). Une faible performance. Beaucoup de « Promesses » laissant sousentendre un processus asynchrone alors que ce n’est pas le cas. (Les promesses sont un élément du JavaScript permettant de l’asynchrone, ça « promet » que l’information arrivera mais ne dis pas quand exactement). Il ne permet que l’automatisation Web uniquement, Il faut néanmoins faire attention à la gestion, le classement et la réutilisation des images lorsque ses tests reposent essentiellement sur ces dernières.



*Figure 2. 6: Fonctionnement de Protractor*

#### 3. 4. Robot Framework

Licence : Open-source

**Robot** **Framework** est un outil d’automatisation de test, développé au départ par Nokia Networks puis devenu open source, qui dispose d’une communauté active et réactive. C’est un framework d’automatisation open-source qui implémente l’approche par mot clé pour les tests d’acceptation et le développement piloté par les tests d’acceptation (ATDD). Robot Framework est indépendant du système d’exploitation et des applications. Le framework de base est implémenté en utilisant Python et fonctionne également sur Jython (JVM) et IronPython (.NET). Ses capacités de test peuvent être étendues par des bibliothèques de tests implémentées avec Python ou Java, et les utilisateurs peuvent créer de nouveaux mots-clés de niveau supérieur à partir de ceux existants en utilisant la même syntaxe que celle utilisée pour créer des cas de test. Il possède un éditeur par défaut (Ride.py) mais il est possible d’utiliser n’importe quel éditeur pour l’écriture des scripts Le lancement des tests peut se faire de 2 manières : soit par ligne de commande ou directement avec l’IHM. Les rapports générés sont complets et facile à lire.

Robot Framework est ouvert et extensible et peut être intégré à pratiquement n'importe quel autre outil pour créer des solutions d'automatisation puissantes et flexibles. Être open source signifie également que Robot Framework est gratuit à utiliser sans frais de licence. Robot Framework a une syntaxe simple, utilisant des mots-clés lisibles par l'homme. Ses capacités peuvent être étendues par des bibliothèques implémentées avec Python ou Java. Le cadre est entouré d'un riche écosystème, composé de **bibliothèques** et d’outils développés en tant que projets distincts. Le projet Robot Framework est hébergé sur **GitHub** où vous pouvez trouver davantage de documentation, de code source et de suivi des problèmes. Les téléchargements sont hébergés chez **PyPI**. Robot Framework a une architecture modulaire qui peut être étendue avec des **bibliothèques** intégrées et personnalisées.

Les données sont définies dans des fichiers en utilisant la syntaxe indiquée dans des **exemples**. Un fichier contenant des tests ou des tâches crée une suite et le placement de ces fichiers dans des répertoires crée une structure imbriquée de suites. Lorsque l'exécution est lancée, le framework analyse d'abord les données. Il utilise ensuite des mots-clés fournis par les bibliothèques pour interagir avec le système cible. Les bibliothèques peuvent communiquer avec le système soit directement, soit en utilisant d'autres outils comme pilotes.

L'exécution du test ou de la tâche est lancée à partir de la ligne de commande. En conséquence, vous obtenez un rapport et un journal au format HTML ainsi qu'une sortie XML. Ceux-ci fournissent un aperçu détaillé de ce que fait votre système.

**Avantages**

Robot Framework est entièrement gratuit, facile à installer et à plugger avec des outils d’intégration continu (Jenkins, Git…), de véritables IDE sont disponibles et sont un réel bonheur à utiliser, la flexibilité des tests (IHM, API, bout en bout…) et la possibilité à être multi-navigateur, Les tests sont simples à écrire par des développeurs et des testeurs avec l’utilisation de mots-clés. De Nombreux outils associés, matures et complets **:** Robot Framework ce n’est pas seulement un langage. Il y a un *runner* de tests bien entendu mais également d’autres outils pour générer des rapports, ou encore un *runner* lançant les tests en parallèle. Tous ces outils proposent énormément d’options et sont très bien documentées. On sent que ces outils ont été bien pensés ; vous arriverez forcément à vos fins lorsque vous mettrez en place votre intégration continue.

**Inconvénients**

Robot Framework est peu avenant de prime abord : très sobre, on se demande si l’outil a réellement des utilisateurs, les logs d’exécution sont supers car très complets facilitant les investigations et le debug, mais du coup mise en péril de la lisibilité des spécifications ellesmêmes, le langage très bien conçu, puissant et complet, mais compliqué à prendre en main, mais mécaniquement ce sont des outils autrement plus complexes à prendre en main qu’un simple éditeur de texte, en outre il est pas adapté à des fonctionnels non-techniques, il fera par contre la joie des testeurs qui ont généralement l’habitude de manipuler des outils complexes quand il ne s’agit pas de coder, ces mêmes testeurs qui peuvent se sentir frustrés par d’autres outils comme Cucumber qui imposent des contraintes et des limitations, seront comblés par l’utilisation de Robot Framework, une communauté restreinte mais très active.Pas d’autocomplétions possible avec d’autres éditeurs hormis celui proposé par défaut. La syntaxe est particulière et peut générer de la réticence chez certains développeurs. Pour certaines utilisations, le testeur “fonctionnel” (sans connaissance technique) aura des difficultés à écrire des tests.

#### 3. 5.Katalon Studio

Licence : Gratuit

**Katalon** Studio est une puissante solution d’automatisation de test intégrée pour les applications Web, mobiles et Web. Il peut également prendre en charge différents navigateurs / appareils tels que Chrome, Firefox, IE, Android et iOS. Contrairement à Selenium, il prend en charge différents niveaux de compétences de test, même les non-programmeurs peuvent être faciles à démarrer un projet de test d’automatisation. Katalon Studio peut également être intégré dans les professions CI / CD et fonctionne bien avec d’autres outils populaires tels que qTest, Jira, Git et Jenkins. Sa fonctionnalité intéressante Katalon Analytics fournit aux utilisateurs des rapports d’exécution de tests, notamment des métriques, des diagrammes et des graphiques.

Quelques fonctionnalités de Katalon :

* Prise en charge des différents navigateurs internet
* Intégration du BDD avec Cucumber
* Génération de rapport html, PDF, JUnit ou sur le cloud (version entreprise)
* Permet de créer des tests pilotés par les données (csv, Excel, base de données)
* Intégration GIT
* Outil Spy pour permettre l’enregistrement facile des composants (boutons, zone de texte...)
* Dépôt de plugins gratuits et payants

**Avantages :**

L'outil peut être utilisé pour tester les éléments tels que les APIs, les applications Web, des applications mobiles, nécessite des connaissances minimales en programmation pour tirer parti de l'outil de test des applications. Facilement intégré aux applications tierces telles que Slack. Plugins prêts, pratiques et GRATUITS disponibles pour rendre les tests plus efficaces. Les scripts de test sont interchangeables avec Selenium Webdriver, La courbe d'apprentissage est relativement rapide et facile à démarrer avec vos premiers tests de base. Il y a aussi beaucoup de documentation si l’on est coincé sur quelque chose. La reconnaissance fiable des objets, les tests sont flexibles

**Inconvénients :**

Étant donné que cet outil est quelque peu nouveau sur le marché, il est donc parfois difficile de trouver des solutions si un utilisateur est coincé dans un problème. Cependant, les développeurs KS font de leur mieux pour trouver la solution qui vous convient le plus tôt possible, le design n’est pas très convivial, on a l'impression d'interagir avec une application créée au début des années 2000, L'automatisation de l'interface utilisateur mobile est assez difficile à configurer. Automatisation Android, une fois que nous avons créé un script à l'aide d'un mobile particulier à l'aide de l'enregistreur, nous n'avons pas pu l'utiliser sur un autre modèle de téléphone. Lors de l'enregistrement, les identifiants créés par le Katalon Studio ont changé pour le téléphone suivant et, par conséquent, il ne reconnaît pas les autres téléphones. Sur l'automatisation de l'interface utilisateur IPhone, la plupart des IPhone ne sont pas reconnus.

Nous avons passé deux mois à faire des recherches et c'est assez incohérent.

#### 3. 6.Unified Functional Testing (UFT)

Licence : Commercial

**Unified Functional Testing (UFT)** est un outil de test commercial populaire pour les tests fonctionnels et de régression. Il offre un ensemble complet de fonctionnalités pour les API, les services Web et les tests GUI des applications Web, de bureau et mobiles sur toutes les plates-formes. Il a avancé la fonction de reconnaissance d’objet basée sur l’image, le mécanisme de gestion des erreurs, la documentation automatisée et les composants de test réutilisables. Visual Basic Scripting Edition est utilisé par UFT pour enregistrer le processus de test et le contrôle d’objet. Il est intégré avec Mercury Business Process Testing et Mercury Quality Center. UFT se présente sous la forme d'un environnement complet de développement comportant nativement l'ensemble des outils nécessaires à la conception, à l’exécution et au reporting des tests automatisés. Les scripts de test sont écrits dans le langage Visual Basic, facilitant considérablement l'utilisation de jeux de données au format Excel, et la gestion des objets\* se fait de façon très simple grâce à l'outil "Object repository".

UFT peut agir sur n'importe quel logiciel, un navigateur internet, un client mail, l'invite de commande, le dernier guide Larousse de cuisine sur CD-ROM, bref tout ce qu'un utilisateur peut effectuer comme action sur une application de type client lourd, UFT peut le faire. D'autre part, depuis que QTP répond au doux nom d'UFT, il s'est doté des fonctionnalités d'un autre logiciel proposé par HP, "HP Service Test" permettant aux utilisateurs de concevoir et d'exécuter des tests de d'API et de webservices. UFT offre également la possibilité de tester ses applications mobiles IOS et Android (avec toutefois certaines contraintes dont, la nécessité d'utiliser une solution tierce "M-eux" ainsi que la version des appareils mobiles à tester).

Grâce à l’Object Spy, aucune connaissance du code source de l’application n'est requise pour identifier un objet (champs texte, bouton, menu déroulant...). Il suffit de lancer l'application à tester, de démarrer l’*objet Spy* puis de cliquer sur l'objet sur lequel on souhaite pouvoir effectuer des actions automatiques dans son scénario, l’objet de test est identifié et peut être enregistré dans l’*Object repository.*

L’*Object repository, est* donc la bibliothèque d'objets de vos scénarios (N.B. : C'est également lui qui contient les différents points de contrôle de fichier que vous aurez définis).

Ainsi, lors de la rédaction d'un scénario dans UFT votre liste d'objets est prédéfinie et il ne vous reste plus qu'à l'appeler dans votre scénario.

**Avantages :**

UFT n'est pas le seul à permettre l'automatisation de test d'applications de type client lourd, mais UFT est simple à prendre en main et dispose de nombreux outils permettant de simplifier l'écriture de scripts. Aucun besoin d'être un développeur confirmé pour écrire un scénario en VB. Quelques notions simples en informatique, les boucles, les conditions, etc... Couplées à vos talents de testeurs suffiront amplement à rédiger vos tests automatiques,

L’auto-complétions prévue par l'IDE est très confortable et vous indiquera la syntaxe des boucles, des conditions si vous ne les connaissez pas en VB.

**Inconvénients :**

UFT est commercialisé sous License renouvelable par trimestre et par utilisateur, il ne peut être installé que sous Microsoft Windows, or dans un contexte ou la tendance est à la réduction des coûts et où les organisations tentent l’expérience open source avec des OS gratuits comme Linux, il peut s'agir là d'un obstacle à l'utilisation de cette solution d'automatisation. Sur le plan technique, UFT étant assez gourmand en ressources système, il nécessitera une machine puissante pour s’exécuter et l’exécution d'une campagne de test sera donc plus lente, à ressources égales, qu'avec un outil comme Selenium. D'autre part, UFT ne peut exécuter qu'un test sur une application à la fois quand Selenium offre la possibilité de paralléliser l'exécution des test grâce à son plugin Selenium grid.

#### 3. 7.TestComplete

Licence : Commercial

**TestComplete** est un outil de test commercial puissant pour les tests Web, mobiles et de bureau. TestComplete prend en charge plusieurs langages de script, notamment JavaScript,

Python, VBScript, JScript, Delphi Script, le script C ++ et le script C #. Son test d’interface utilisateur automatisé utilise la fonctionnalité d’enregistrement et de relecture sans script pour créer facilement des tests d’interface utilisateur automatisés. TestComplete peut permettre aux testeurs d’effectuer des tests pilotés par des mots clés et des données. Il s’intègre également avec Jenkins dans un processus CI.

Selon le fournisseur, les principaux avantages commerciaux comprennent :

* **Facile à utiliser pour tous les niveaux d’expérience :** utilisez la puissante fonction d'enregistrement et de lecture pour exécuter des tests ou choisissez parmi plusieurs langages de programmation tels que Python, VB.NET ou JavaScript pour écrire le vôtre.
* **Accès instantané à des environnements réels dans le cloud :** accédez en temps réel aux derniers appareils, résolutions, navigateurs et systèmes d'exploitation de notre laboratoire cloud. Exécutez des tests en parallèle sur plus de 1500 environnements distants pour améliorer la vitesse et la couverture des tests.
* **Reconnaissance d'objets robuste pour les tests d'interface utilisateur :** automatisez tout ce que vous avez vu sur l'interface utilisateur. Avec le moteur robuste d'identification d'objet de TestComplete, vous pouvez choisir parmi plus de 50 000 propriétés d'objet dans plus de 500 types de contrôle, y compris tous les principaux frameworks et contrôles tiers.
* **Conçu pour les tests continus :** intégrez facilement TestComplete avec des outils de cadre d'automatisation de test, des systèmes CI tels que GIT ou Mercurial, ou des platesformes de gestion de test comme Jira ou QAComplete.

Les fonctionnalités de TestComplete sont les suivantes :

* Test sans script ;
* Tests basés sur les données ;
* Test multi-navigateurs ;
* Test fonctionnels ;
* Tests de régression
* Test parallèle

**Avantages**

TestComplete a une interface utilisateur simple, vous n'avez donc pas besoin d'être un codeur pour l'utiliser, il a beaucoup de fonctionnalités à intégrer avec différents types de logiciels - basés sur le Web, basés sur le bureau, etc… Le test de mots-clés est vraiment facile à utiliser. Tout le monde peut le comprendre en quelques heures, Le soutien qu'ils fournissent est bien. Leur service client est de premier ordre et ils disposent de nombreuses ressources pour les utilisateurs débutants à plus avancés. Des fonctionnalités plus avancées sont fournies et sont utiles. Il peut facilement s'intégrer à votre flux de travail de développement.

**Inconvénients**

TestComplete peut nécessiter des étapes manuelles pour mettre à jour les cas de test, il a quelques bogues difficiles à résoudre. Un test bêta pour les versions majeures serait bien. En d'autres termes, ce serait bien de voir ce que les tests se feront ou ne feront pas pour chaque version. Une fonction d'apprentissage intégrée au produit, quelque chose comme ce que Postman a fait récemment pour sa dernière version. Cela aiderait les débutants ainsi que les utilisateurs plus expérimentés. Comparé à d'autres outils, TestComplete manque de certains domaines tels que le chargement de la bibliothèque de scripts pendant l'exécution, dans TestComplete, vous devez avoir plus d'expérience en l'utilisant pour répondre à vos besoins. En somme il n’est pas facile à peaufiner.

#### 3. 8.Ranorex

Licence : Commercial

**Ranorex** est une société de développement de logiciels de premier plan qui propose un logiciel novateur d’automatisation des tests. Ranorex facilite les tests, fait gagner du temps dans le processus de test et permet aux clients d’assurer la plus haute qualité de leurs produits. Ses outils flexibles et son retour sur investissement rapide en font le choix idéal pour les entreprises de toutes les tailles.

l’outil tout-en-un d’automatisation des tests. Pour le test d’applications de bureau, Web ou mobiles, avec des outils d’automatisation simples sans code, un IDE complet, une reconnaissance d’objet robuste, des rapports flexibles et Selenium WebDriver intégré.

Ranorex est une infrastructure d’automatisation de test à interface graphique permettant de tester des applications de bureau, Web et mobiles. C’est un outil d’automatisation commerciale assez complet pour les tests Web, mobiles et de bureau. L’outil propose des fonctionnalités avancées pour la reconnaissance de GUI, des scripts de test réutilisables, la détection de bugs et l’enregistrement / lecture. La création de tests sans code est également une fonctionnalité très utile qui permet aux nouveaux testeurs d’automatisation d’apprendre et d’appliquer l’automatisation des tests à leurs projets. L’outil prend en charge l’intégration de Selenium pour les tests d’applications Web. Les testeurs peuvent distribuer l’exécution de leurs tests sur les plates-formes et les navigateurs en utilisant la grille Selenium.

La fonctionnalité et l'outil le plus important du Ranorex est Ranorex Spy, ce qui le rend unique par rapport aux autres outils d'automatisation des tests. Ranorex comprend toutes les fonctions et fonctionnalités qui analysent l'application de bureau, mobile et Web dans le test et il comprend également leurs contrôles et éléments d'interface utilisateur.

Fonctionnement de Ranorex :

* Suivez les éléments de l'interface utilisateur
* Le Mode d'édition fonctionne avec RanoreXPath
* Contient un éditeur de chemin
* Possibilité de créer des fichiers d'instantanés Ranorex

**Avantages :**

L'outil est professionnel, Il est très utile pour l'automatisation des tests mobiles pour nos applications Android et iOS. Les tests sont stables et faciles à entretenir, Un non-programmeur peut facilement convertir une étape enregistrée en méthode de code utilisateur où il / elle peut utiliser n'importe quelle instruction conditionnelle et personnaliser le résultat par défaut avec des informations supplémentaires, Ranorex peut être utilisé par ceux qui n'ont aucune connaissance du code via des modules d'enregistrement, et 99% du temps, c'est tout ce qui est nécessaire. En outre, il permet aux modules de code d'être écrits et exécutés par des personnes expérimentées en C #, ce qui signifie que presque tout est possible à automatiser.

**Inconvénients :**

Malheureusement les coûts sont élevés, ces outils ne sont pas bon marché. En outre, Ranorex en raison de sa complexité, il n'est pas facile à première vue. Vous avez besoin de formation et de temps pour apprendre à préparer les tests, les niveaux de reporting et de reporing peuvent être améliorés en ajoutant des formats de tableau et des formats plus pris en charge comme Excel, le logiciel est capricieux et difficile à utiliser et à comprendre ; cela peut parfois être frustrant. L’on recommande de l'utiliser sur un PC puissant avec beaucoup de RAM car le logiciel Ranorex Studio et Spy peuvent parfois se planter sur des PC plus faibles.

#### 3. 9.Telerik TestStudio

Licence : Commercial

**Telerik** TestStudio propose une solution pour automatiser les tests des applications bureautiques, Web et mobiles, y compris les tests d’interface utilisateur, de charge et de performance. Il prend en charge différents langages de programmation, y compris JavaScript, Silverlight, WPF, MVC, AJAX, HTML, MVC et ASP.NET. Telerik TestStudio propose des fonctionnalités avancées pour les tests croisés sur navigateur, les tests manuels, les outils de suivi des bogues et l’enregistrement / lecture. La version d’essai gratuite est disponible sur demande directe.

L’enregistrement et lecture de test sans script, l’exécution de tests inter-navigateurs : Internet Explorer, Firefox, Chrome et Safari (navigateur Web), abstraction et réutilisation d’éléments, Intégration à Visual Studio 2010/2012, MS Team Foundation Server, intégration de NUnit, MbUnit, XUnit, validation de l’interface utilisateur basée sur les phrases, intégration continue avec Microsoft Build Server, CruiseControl et TeamCity, Débogueur visuel, Testez la personnalisation en C # et VB.NET, Essais exploratoires, Test manuel, Intégration à HP Quality Center, Cadre de test intégré, DOM Explorer, Intégration du débogueur Web Fiddler, Tests automatisés basés sur les données, Intégration d’outils de suivi de bogues.

Test Studio possède plusieurs fonctionnalités à savoir :

* Web et de bureau automatisés
* Les Plug-in Visual Studio
* Les tests manuels
* Les essais exploratoires
* Les tests fonctionnels de l'interface utilisateur
* La planification et exécution des tests
* L’exécution de tests parallèles sur plusieurs machines
* Le support d'intégration continue
* L’interface utilisateur Web de test de charge
* Les tests mobiles
* Les Tests API

**Avantages**

L’interface très conviviale, le référentiel d'éléments est une fonctionnalité intéressante et aide vraiment à gagner du temps lorsque l'interface est modifiée. C # ou VB.net peuvent être utilisés pour éditer du code, ce qui le rend encore plus facile pour ceux qui ont des connaissances préalables sur ces langues, C'est très bien pour les testeurs qui ont peu d'expérience en codage mais qui sont prêts à apprendre. Il est très utile pour les tests de régression et aide à identifier les problèmes qui auraient pu nuire aux utilisateurs finaux. Il est également intéressant de suivre les performances des pages dans le temps. La possibilité de relier plusieurs tests ensemble et de réutiliser le test est une fonctionnalité intéressante.

**Inconvénients**

L'IDE de codage est un peu lente par rapport à Visual Studio, il est assez cher, mais il y a plusieurs fonctionnalités qui compensent le prix. Le temps de démarrage du client est un peu lent. L'application est un peu lente, quand il y a beaucoup de tests ouverts. L'environnement de planification automatisé est un peu difficile à maintenir. La documentation utilisateur et le matériel de formation semblent un peu dépassés. L'intégration avec les systèmes de contrôle de version plus récents tels que VSTS est boguée.

**RECAPITULATIF GENERAL DE LA RECHERCHE SUR LA DOCUMENTATION DES TESTS**

Le bon choix d’outil d’automatisation, de processus de test et d’équipe est un acteur important pour le succès de l’automatisation. Les méthodes manuelles et automatisées vont de pair pour des tests réussis.

Après études des solutions de tests logiciels existants, il en ressort que chacune des solutions citées plus haut présente des avantages et des inconvénients. Le logiciel qui a le plus de chance d’être retenue est celui qui présente des caractéristiques suivantes : la simplicité et la puissance.

Permettant d’exécuter des cas de tests ouverts dans la fenêtre courante dans une autre fenêtre, la fiabilité de faire des tests stables et faciles à entretenir, personnalisant les résultats, facilement intégré aux applications tierces telles que Slack, Asana et générant des rapports de test avec outils de capture intégré qui permettra d’intégrer les images dans les rapports de tests en cas de besoin, ajouter au besoin plusieurs autres compétences fortes sur les tests en élargissant le spectre de recherche, avec précision, complétude, traçabilité des documents, une architecture simple et modulaire, une politique de traitements des erreurs clairement définie en temps réel, interface utilisateur convivial, pouvant fonctionner avec une liaison de langue pour aider le développeur en travaillant dans la langue avec laquelle il peut être à l'aise. Les tests s'exécutant sous plusieurs navigateurs différents. Les scripts pouvant être enregistrés et également réexécutés à tout moment.

Au regard de l’analyse faite il en ressort que nous pouvons orienter notre travail selon l’un des choix suivants :

* Choisir une solution d’automatisation de test existants qui se rapproche à solution de notre problème et l’améliorer avec des nouvelles fonctionnalités.
* Choisir une solution payante et la paramétrer pour l’adapter
* Utiliser une solution de bout en bout

La première approche semble intéressante et c’est celle que nous allons adopter, parce qu’elle plus réaliste et plus efficace.

L’évaluation des outils existants montre que les éléments qu’il faudrait prendre en compte pour l’optimisation sont :

* Une gestion de l’upload et du download sans utiliser de gros hacks ;
* Une gestion simple des données de test séparées des tests eux-mêmes. C’est-àdire pourvoir jouer plusieurs fois un test avec des données différentes ;
* Une connexion aux outils de collaboration et planification telles que Slack et Asana ;
* La génération des rapports de tests pour son utilisation ;
* Une meilleure performance. Au-delà d’un certain nombre de tests, les temps de retour doivent être courts et brefs, pour tous les passer depuis un poste de développeur et d’avoir un retour en quelques minutes. D’où la mise en place de l’intégration continue. Sur le point des performances ;
* Il faudrait ajouter une fonctionnalité permettant de paralléliser les tests ; une intégration à Internet Explorer pour utiliser Selenium IDE avec ce navigateur (si c’est le navigateur choisi comme cible par le client par exemple).

Dans ce chapitre, nous avons défini le cadre du projet en présentant le contexte et l’étude de l’existant, puis nous avons fait une étude critique des logiciels de test existant en précisant les avantages et les inconvénients de chaque logiciel.

# Partie II: Analyse, Conception et implémentation

## Chapitre III:Analyse et conception

Ce chapitre parlera de l’analyse et la conception du logiciel en faisant tout d’abord une analyse approfondie des tests ensuite une modélisation complète de la solution et enfin une architecture générale.

### Section 1. Analyse approfondie

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.

Les phases de test dans le cycle de développement d'un produit logiciel permettent d'assurer un niveau de qualité défini en accord avec le client. Une procédure de test peut donc être plus ou moins fine, et par conséquent l'effort de test plus ou moins important et coûteux selon le niveau de qualité requis. Aujourd'hui, les métiers du test se développent considérablement. C'est en grande partie grâce à une prise de conscience de la complexité ou de la criticité des produits. Il est alors important que ces différentes phases soient bien intégrées dans le cycle de développement sur la base de bonnes pratiques et de la rationalisation du processus.

Il existe différentes façons de classer les tests informatiques. Nous proposons ici une classification selon trois perspectives : la nature de l'objet à tester (perspective étroitement liée au cycle de développement), l'accessibilité de la structure de l'objet et la propriété de l'objet (performance par exemple). Ces trois perspectives ne permettent cependant pas de classer le test de non-régression .

#### 1.classification sElon la naturE dE l’objEt

Un niveau de tests comme un groupe d'activités de tests qui sont organisées et gérées ensemble.

Un niveau de tests est lié aux responsabilités dans un projet.

Le Comité français du test logiciel (CFTL) identifie quatre niveaux de test :

* Test unitaire (ou test de composants).
* Test d'intégration (anciennement test technique ou test d'intégration technique).
* Test système (anciennement test fonctionnel ou test d'intégration fonctionnel ou homologation).
* Test d'acceptation (anciennement test usine ou recette).

##### Test unitaires ou de composants

Un test unitaire est une procédure permettant de vérifier le bon fonctionnement d'une partie précise d'un logiciel ou d'une portion d'un programme (appelée « unité » ou « module »).

On écrit un test pour confronter une réalisation à sa spécification. Le test définit un critère d'arrêt (état ou sorties à l'issue de l'exécution) et permet de statuer sur le succès ou sur l'échec d'une vérification. Grâce à la spécification, on est en mesure de faire correspondre un état d'entrée donné à un résultat ou à une sortie. Le test permet de vérifier que la relation d'entrée / sortie donnée par la spécification est bel et bien réalisée.

L’utilité d’un test unitaire est de trouver rapidement les erreurs, sécuriser la maintenance et documenter le code, Lors d'une modification d'un programme, les tests unitaires signalent les éventuelles régressions. En effet, certains tests peuvent échouer à la suite d'une modification, il faut donc soit réécrire le test pour le faire correspondre aux nouvelles attentes, soit corriger l'erreur se situant dans le code.

##### Fonctionnement

On définit généralement 4 phases dans l'exécution d'un test unitaire :

* **Initialisation** (fonction setUp) : définition d'un environnement de test complètement reproductible (une fixture).
* **Exercice** : le module à tester est exécuté.
* **Vérification** (utilisation de fonctions assert) : comparaison des résultats obtenus avec un vecteur de résultat défini. Ces tests définissent le résultat du test : SUCCÈS (SUCCESS) ou ÉCHEC (FAILURE). On peut également définir d'autres résultats comme EVITE (SKIPPED).
* **Désactivation** (fonction tearDown) : désinstallation des fixtures pour retrouver l'état initial du système, dans le but de ne pas polluer les tests suivants. Tous les tests doivent être indépendants et reproductibles unitairement (quand exécutés seuls) **Utilisation**

Il s'agit pour le programmeur de tester un module, indépendamment du reste du programme, ceci afin de s'assurer qu'il répond aux spécifications fonctionnelles et qu'il fonctionne correctement en toutes circonstances. Cette vérification est considérée comme essentielle, en particulier dans les applications critiques. Elle s'accompagne couramment d'une vérification de la couverture de code (évaluation de la couverture structurelle), qui consiste à s'assurer que l'ensemble des tests conduit à exécuter l'ensemble (ou une fraction déterminée) des instructions présentes dans le code.

##### Test d'intégration

Le **test d'intégration** est une phase dans les tests, qui est précédée des tests unitaires et est généralement suivie par les tests de validation. Dans le test unitaire, on vérifie le bon fonctionnement d'une partie précise d'un logiciel ou d'une portion d'un programme (appelée « unité » ou « module ») ; dans le test d’intégration, chacun des modules indépendants du logiciel est assemblé et testé dans l’ensemble

L'objectif de chaque phase de test est de détecter les erreurs qui n'ont pas pu être détectées lors de la précédente phase. Pour cela, le test d’intégration a pour cible de détecter les erreurs non détectables par le test unitaire. Le test d’intégration permet également de vérifier l'aspect fonctionnel, les performances et la fiabilité du logiciel. L'intégration fait appel en général à un système de gestion de versions, et éventuellement à des programmes d'installation. Cela permet d'établir une nouvelle version, fondée soit sur une version de maintenance, soit sur une version de développement.

##### L'organisation d'un test d’intégration

Le but de l'organisation d'un test d’intégration est de définir la stratégie de l'activité de test d’intégration en termes d'ordre d’intégration, de test à réaliser, du matériel sur lequel seront lancés les tests et les outils /procédure employée.

Nous avons employé la procédure suivante pour l'organisation d'un test d’intégration.

* Introduction et organisation
  + Présentation du document
  + Décrire l'organisation en termes de procédure à suivre et les outils et matériels disponibles pour l'équipe d’intégration
* Stratégie
  + Identifier le produit fini du test d’intégration
  + Identifier les spécifications du test d'intégration qui doivent être produites
  + Mettre en évidence l’intérêt de chaque test
  + Définir l'ordre dans lequel les tests doivent être effectués
* Contenu des spécifications du test d’intégration
  + Pour chaque spécification mentionnée dans la stratégie, définir l’assemblage inhérent aux tests et les attentes du design qui sont à vérifier
  + Identifier la configuration matérielle requise pour le test
  + Lister les outils et logiciels de test
  + Identifier les assemblages précédemment testés nécessaires au test.

Cette procédure est adaptée en fonction des besoins. Pour un système comprenant plusieurs sous-systèmes, la dernière procédure (contenu des spécifications des tests d'intégration) sera usuellement effectuée pour chacun des sous-systèmes, par exemple.

Il existe plusieurs méthodes pour les tests d’intégration dont les plus courantes sont : **Topdown**, **Bottom**-**up**, **Sandwich** et **Big**-**bang,** et chaque étape présente ses avantages et inconvénients, pour notre cas nous utiliserons la méthodes **Bottom-up** car il permet Localisation facile des erreurs, aucun besoin de stub, les modules réutilisables sont testés correctement et les tests peuvent se faire en parallèle avec le développement.

Pour les applications utilisant les nouvelles technologies et donc des ateliers de génie logiciel (Eclipse - Visual Studio - JBuilder - JDeveloper…), les tests d’intégration ont évolué vers de l’intégration continue. L’intégration continue est la fusion des tests unitaires et des tests d’intégration, car le programmeur détient toute l’application sur son poste et peut donc faire de l’intégration tout au long de son développement.

##### Test Système

Les **tests système** de logiciel ou de matériel réfèrent à un processus de test d'un système intégré afin d'évaluer sa conformité aux exigences spécifiées. Les tests système appartiennent à la classe des tests de type boîte noire, et en tant que tels, ne devraient exiger aucune connaissance de la conception interne du code ou de la logique1.

En règle générale, les tests système prennent comme entrée tous les composants logiciels « intégrés » (ayant réussi les tests d'intégration) mais aussi le système logiciel lui-même intégré à n'importe quel système(s) matériel(s) compatible(s). Le but des tests d'intégration est de détecter d'éventuelles incohérences entre les unités logicielles intégrées ensemble (appelés assemblages) et entre les assemblages et le matériel. Les tests système sont un type plus restreint de test, ils cherchent à détecter des défauts à la fois dans les « inter-assemblages » mais également au sein du système dans son ensemble.

Les tests système sont réalisées sur la totalité du système dans le contexte de Spécifications Fonctionnelles des Besoins et/ou de Spécifications Techniques des Besoins. Les tests système vérifient non seulement la conception, mais également le comportement et les attentes présumées du client. Ils sont également destinés à réaliser des tests aux limites définies dans les spécifications du logiciel/matériel mais également par-delà ces limites.

##### Test d’acceptation

Le **test d'acceptation** est une phase de développement des projets, visant à assurer formellement que le produit est conforme aux spécifications (réponse donnée à un instant « t » aux attentes formulées). Elle s'inscrit dans les activités plus générales de qualification. Cette étape implique, en la présence effective des différents acteurs du projet, maîtrise d’œuvre et maîtrise d'ouvrage, le déroulement rigoureux de procédures de tests préalablement décrits, et l'identification de tout écart fonctionnel ou technique

La procédure de recette se déroule en deux étapes principales : les tests système et les tests d'acceptation utilisateur.

Si la première étape a lieu chez le fournisseur, la deuxième se déroule en revanche généralement dans les locaux et avec les infrastructures du client

#### 2.Classification selon l'accessibilité de la structure de l'objet

* Technique de conception de test par boîte blanche (white box) : technique de conception de test, en général fonctionnel, fondée sur l'analyse de la structure interne du composant ou du système.
* Technique de conception de test par boîte noire (black box) : technique de conception de test, fonctionnel ou non, qui n'est pas fondée sur l'analyse de la structure interne du composant ou du système mais sur la définition du composant ou du système.

Par extension, on appelle couramment les tests issus de ces types de techniques de conception, tests boîte blanche (ou tests structurels) et tests boîte noire.

Les tests boîte blanche vérifient la structure interne de l'objet, par exemple l'exécution des branches des instructions conditionnelles. Les tests unitaires et d'intégration sont souvent spécifiés à l'aide des techniques de conception de test par boîte blanche. Pour certains types de logiciel, des normes prescrivent les techniques de conception de tests par boîte blanche à utiliser (par exemple, la norme américaine RTCA/DO-178B pour les logiciels d'avionique).

Les tests boîte noire vérifient la définition de l'objet. Les tests système et d'acceptation sont souvent spécifiés à l'aide des techniques de conception de test par boîte noire, mais rien n'empêche d'utiliser ces techniques pour définir des tests unitaires ou d'intégration

##### Test boite blanche

Dans la théorie des systèmes, une boîte blanche (de l'anglais white box), ou boîte transparente, est un module d'un système dont on peut prévoir le fonctionnement interne car on connaît les caractéristiques de fonctionnement de l'ensemble des éléments qui le composent. Autrement dit une boîte blanche est un module qui comporte aussi peu de boîtes noires que possible. On qualifie les systèmes mixtes de « boîtes grises ». Les systèmes complexes s'articulent autour du paradigme boîte noire - boîte blanche, c'est-à-dire qu'ils forment un ensemble cohérent dont il convient de prévoir le fonctionnement ou les scénarios. Cette théorie est particulièrement bien adaptée au problème des tests logiciels en informatique.

Pour comprendre le fonctionnement d'une boîte blanche, afin de s'assurer qu'elle est effectivement une boîte blanche, il convient de procéder à des tests structurels connus sous le nom « test par boîte blanche », de l'anglais white-box Testing (on trouve aussi les expressions anglaises glass-box testing ou encore clear-box testing).

En informatique, c'est une méthode de test logiciel visant à analyser un programme informatique dont on connaît exactement le fonctionnement interne. On peut utiliser le code source du programme.

Dans le domaine particulier de la sécurité informatique, on parle de test d'intrusion ou de vulnérabilités. Cette méthode permet de trouver des erreurs subtiles qu'il est difficile de localiser avec des outils d'analyse automatique (tels que SPlint, Rats ou Flawfinder). On peut reprocher à cette méthode d'être coûteuse en temps car l'analyse manuelle est lente. De plus, dans le cas où l'on trouve une erreur, on ne sait que difficilement si elle sera exploitable en pratique

##### Test boite noire

La boîte noire est représentée de façon élémentaire en affichant les entrées et les sorties mais en masquant le fonctionnement interne. Tout peut être représenté sous forme d'une boîte noire : un transistor, un algorithme, un réseau comme internet, le fonctionnement d'une entreprise ou les relations humaines au sein d'un groupe.

Le contraire d'une boîte noire, dit boîte blanche, est un système dont les mécanismes sont visibles et permettent d'en comprendre le fonctionnement. Une bicyclette illustre bien ce type de système parce que, contrairement à ce qui se passe avec une boîte noire, les mécanismes de propulsion, de guidage, d'adhérence et de freinage sont visibles au premier coup d'œil.

Le test de la boîte noire, ou test de la boîte opaque, est utilisé en programmation informatique et en génie logiciel pour tester un programme en vérifiant que les sorties obtenues sont bien celles prévues pour des entrées données.

L'expression « boîte noire », ou « boîte opaque », vient du fait que les composants et les processus du dispositif de traitement ne sont pas visibles ou transparents et que le programme testé n'est pas étudié

Son utilisation se fait lors des tests après développement dans un logiciel, par exemple pour la table de décision et lors du test d'intrusion, pour tester la vulnérabilité du système de l'extérieur.

#### 3.Classification selon la propriété de l'objet

On ne peut pas être exhaustif, on se contentera de quelques exemples :

* Test fonctionnel ;
* Test de performance ;
* Test d'intrusion ;
* Test utilisateur.

Une liste de toutes les caractéristiques d'un logiciel a été établie par la norme ISO 9126.

En dehors du cas très particulier de systèmes extrêmement simples, il est impossible de tester exhaustivement un logiciel, car le nombre de configurations possibles croît de façon exponentielle selon le nombre de mises en situation différentes que le logiciel pourra être appelé à traiter ; le nombre de configurations accessibles, bien qu'inférieur, reste tout de même prohibitif. La réussite des tests ne permet donc pas de conclure au bon fonctionnement du logiciel. On essaye cependant, de faire en sorte que si un bug est présent, le test le mette en évidence, notamment en exigeant une bonne *couverture* des tests :

* Couverture en points de programme : chaque point de programme doit avoir été testé au moins une fois ;
* Couverture en chemins de programme : chaque séquence de points de programme possible dans une exécution doit avoir été testée au moins une fois (impossible en général) ;
* Couverture fonctionnelle : chaque fonctionnalité métier de l'application doit être vérifiée par au moins un cas de test.

Si l'on veut des assurances plus fortes de bon fonctionnement, on peut utiliser des méthodes formelles.

Les bibliothèques de tests telles que JUnit pour le langage Java permettent de faciliter l'écriture de tests unitaires par l'apport des méthodes assert permettant de vérifier le comportement du programme.

Selon la complexité du logiciel, des séquences de vérification globale peuvent s'avérer nécessaires. Celles-ci mettent en jeu la maîtrise d'ouvrage et toutes les composantes du projet, au-delà du logiciel lui-même (processus, organisation, formation, accompagnement du changement) : réception, qualification, certification, simulation, VABF (vérification d'aptitude au bon fonctionnement) … Les termes varient selon les contextes.

Un **test de** **performance** est un test dont l'objectif est de déterminer la performance d'un système informatique. L'acception la plus courante de ce terme est celle dans laquelle ces tests logiciels vont avoir pour objectif de mesurer les temps de réponse d'un système applicatif en fonction de sa sollicitation. Cette définition est donc très proche de celle de test de charge où l'on mesure le comportement d'un système en fonction de la charge d'utilisateurs simultanés. Seuls les tests de charge permettent de valider correctement une application ou un système avant déploiement, tant en qualité de service qu'en consommation de ressources

Ces types de texte peuvent être de plusieurs types, notamment :

* **Test de charge** : il s'agit d'un test au cours duquel on va simuler un nombre d'utilisateurs virtuels prédéfinis, afin de valider l'application pour une charge attendue d'utilisateurs. Ce type de test permet de mettre en évidence les points sensibles et critiques de l’architecture technique. Il permet en outre de mesurer le dimensionnement des serveurs, de la bande passante nécessaire sur le réseau, etc. ;
* **Test de performance** : proche du Test de charge, il s'agit d'un test au cours duquel on va mesurer les performances de l'application soumise à une charge d'utilisateurs. Les informations recueillies concernent les temps de réponse utilisateurs, les temps de réponse réseau et les temps de traitement d’une requête sur le(s) serveur(s). La nuance avec le type précédent réside dans le fait qu'on ne cherche pas ici à valider les performances pour la charge attendue en production, mais plutôt vérifier les performances intrinsèques à différents niveaux de charge d'utilisateurs ;
* **Test de dégradations des transactions** : il s'agit d'un test technique primordial au cours duquel on ne va simuler que l'activité transactionnelle d'un seul scénario fonctionnel parmi tous les scénarios du périmètre des tests, de manière à déterminer quelle charge le système est capable de supporter pour chaque scénario fonctionnel et d'isoler éventuellement les transactions qui dégradent le plus l'ensemble du système. Ce test peut tenir compte ou non de la cadence des itérations, la représentativité en termes d'utilisateurs simultanés et sessions simultanées n'étant pas ici un objectif obligatoire, s'agissant ici plutôt de déterminer les points de contention générés par chaque scénario fonctionnel. La démarche utilisée est d'effectuer une montée en charge linéaire jusqu'au premier point de blocage ou d'inflexion. Pour dépasser celui-ci, il faut paramétrer méthodiquement les composants systèmes ou applicatifs afin d'identifier les paramètres pertinents, ce jusqu'à obtention des résultats satisfaisants. Méthodologiquement, ce test est effectué avant les autres types de tests tels que performance, robustesse, etc. où tous les scénarios fonctionnels sont impliqués ;
* **Test de stress** : il s'agit d'un test au cours duquel on va simuler l'activité maximale attendue tous scénarios fonctionnels confondus en heure de pointe de l'application, pour voir comment le système réagit au maximum de l'activité attendue des utilisateurs. La durée du palier en pleine charge, en général de deux heures, doit tenir compte du remplissage des différents caches applicatifs et clients, ainsi que de la stabilisation de la plateforme de test suite à l'éventuel effet de pic-rebond consécutif à la montée en charge. Dans le cadre de ces tests, il est possible de pousser le stress jusqu'à simuler des défaillances systèmes ou applicatives afin d'effectuer des tests de récupération sur incident (*Fail-over*) ou pour vérifier le niveau de service en cas de défaillance ;
* **Test de robustesse, d'endurance, de fiabilité** : il s'agit de tests au cours desquels on simulera une charge importante d'utilisateurs sur une durée relativement longue, pour voir si le système testé est capable de supporter une activité intense sur une longue période sans dégradations des performances et des ressources applicatives ou système. Le résultat est satisfaisant lorsque l'application a supporté une charge supérieure à la moitié de la capacité maximale du système, ou lorsque l'application a supporté l'activité d'une journée ou plusieurs jours/mois/années, pendant huit à dix heures, sans dégradation de performance (temps, erreurs), ni perte de ressources systèmes ;
* **Test de capacité, test de montée en charge** : il s'agit d'un test au cours duquel on va simuler un nombre d'utilisateurs sans cesse croissant de manière à déterminer quelle charge limite le système est capable de supporter. Éventuellement, des paramétrages peuvent être effectués, dans la même logique que lors des tests de dégradation, l'objectif du test étant néanmoins ici de déterminer la capacité maximale de l'ensemble systèmeapplicatif dans une optique prévisionnelle ;
* **Test aux limites** : il s'agit d'un test au cours duquel on va simuler en général une activité bien supérieure à l'activité normale, pour voir comment le système réagit aux limites du modèle d'usage de l'application. Proche du test de capacité, il ne recouvre pas seulement l'augmentation d'un nombre d'utilisateurs simultanés qui se limite ici à un pourcentage en principe prédéfini, mais aussi les possibilités d'augmentation du nombre de processus métier réalisés dans une plage de temps ou en simultané, en jouant sur les cadences d'exécutions, les temps d'attente, mais aussi les configurations de la plateforme de test dans le cadre d'architectures redondées (*Crash Tests*) ;
* **Test de résilience** : il s'agit d'un test au cours duquel on va simuler une activité réelle tout en simulant des pannes en environnement réel et de vérifier que le système informatique continue à fonctionner. Proche du test de robustesse, il permet de s'assurer de la résilience des applications et de l'infrastructure sous-jacente. Pour provoquer les pannes pendant le test, on peut utiliser des chaos monkey qui mettent aléatoirement hors service des instances dans l’environnement de test.
* Il existe d'autres types de tests, plus ciblés et fonction des objectifs à atteindre dans la campagne de tests : o **Test de Benchmark** (comparaisons de logiciel, matériels, architectures…), o **Tests de non-régression des performances,** o **Tests de Composants,** o **Tests de Volumétrie des données,** o **Tests de Fonctionnement, etc.**

Étant entendu qu'en principe un type de test correspond à un type d'objectif, et que dans une matrice de couverture des tests les résultats se recoupent obligatoirement, il est rare (et coûteux) de réaliser l’ensemble de ces tests pour une application donnée.

Si l'application est déjà en production, ou en phase pilote, on peut aussi, afin de connaître les performances du système, réaliser une métrologie, qualifiée de surveillance de la production, qui permettra d'observer dans le détail le fonctionnement du système sur la base d'actions réelles des utilisateurs. Les résultats d'une telle campagne de métrologie permettant de connaître les fonctionnalités réellement utilisées, et leur fréquence d'utilisation, ils peuvent ensuite servir de base pour orienter les tests à réaliser dans des simulations futures, ou servir de base à une solution de supervision de la production.

### Section 2. Modélisation conceptuelle

La modélisation consiste à créer une représentation virtuelle de la réalité de façon à faire ressortir les points auxquels on s’intéresse. L’analyse consiste à définir les besoins de l’utilisateur et la conception d’un système consiste à produire une architecture du produit logiciel.

#### 1.Démarche qualité

Les projets liés à l’évolution ou à l’implémentation d’un SI sont souvent sujets à des dérives en termes de coûts et de durée, et recouvrent rarement le périmètre initialement défini. La démarche Qualité est nécessaire pour fiabiliser la gestion de projet, au moyen de différents outils de normalisation des méthodes de travail et de vérification/validation. L’adoption de cette démarche doit amener à une meilleure maîtrise des coûts et de la durée du projet.

**La démarche Qualité** consiste à trouver l’adéquation entre la réponse aux besoins du projet, l’expression correcte de ces besoins par des spécifications adéquates qui passent par une écoute attentive du client, et une réalisation répondant à l’expression des besoins. Si les spécifications sont conformes aux besoins mais que la réalisation ne répond pas aux spécifications et donc aux besoins, on parle de défauts dans la réalisation. C’est de **la nonqualité**. Si la réalisation est conforme aux besoins alors que les spécifications n’étaient pas bonnes, on a eu de la chance, on parle de qualité aléatoire. Enfin, si la réalisation est conforme aux spécifications mais que ces dernières ont surévalué les besoins, on parle **de sur-qualité**.

La démarche qualité doit s’intégrer à toutes les phases du projet :

##### • Phases d’études préalables et de définition fonctionnelle du besoin

Lors des phases d’étude préalable et de définition fonctionnelle des besoins, la qualité se caractérise par la capacité des études produites à adresser les véritables objectifs du projet et de complètement satisfaire les attentes associées. Lors de ces phases, les contrôles qualité reposent d’une part, sur la vérification du respect des principes de la méthodologie mise en œuvre et d’autre part, sur des revues visant à garantir l’adéquation et la cohérence des solutions proposées avec les attentes entre les différentes phases (schéma directeur, puis analyse préalable puis définition fonctionnelle du besoin). Le processus qualité s’attachera également à s’assurer du bon fonctionnement du cycle de validation pour vérifier que tous les principes et solutions proposés ont fait l’objet d’une validation.

##### • Phases de réalisation

Le contrôle de la qualité porte d’une part, sur la vérification de la méthodologie mise en œuvre et d’autre part, sur des revues de code permettant de garantir les performances et la maintenabilité des applications développées.

##### • Phases de test

Les phases de test permettent de vérifier d’une part, le bon fonctionnement intrinsèque des applications livrées et d’autre part, l’adéquation entre les fonctions réalisées par ces applicatifs et les fonctions spécifiées dans les dossiers de définition du besoin. En conséquence, lors de ces phases, le contrôle de qualité va consister à vérifier le bon fonctionnement de ce processus.

##### • Contrôle

Contrôle de l’élaboration des fiches de tests (cycle de test, cas de test, description des jeux de test, description des résultats attendus),

##### • Reproduction de test par sondage

Lors des phases de test, il est important de refaire quelques tests aléatoirement par une autre personne pour limiter les risques d'erreur.

#### 2.La roue de Deming

La roue de Deming (de l’anglais Deming Wheel) dite PDCA (Plan-Do-Check-Act) est un modèle d’amélioration continue utiliser en management de la qualité. Son nom vient du statisticien William Edwards Deming. La roue de Deming est un moyen mnémotechnique permettant de repérer avec simplicité les étapes à suivre pour améliorer la qualité dans une organisation.

##### Principe

La méthode comporte quatre étapes, chacune entraînant l’autre, et vise à établir un cercle vertueux en capitalisant sur les connaissances acquises. Sa mise en place doit permettre d’améliorer sans cesse la qualité d’un produit, d’un service, d’une organisation.

**P**LAN : Préparer, planifier (ce que l’on va réaliser) : définir les priorités, les objectifs, le plan d’action, les indicateurs de mesure du succès, mettre à disposition les ressources nécessaires, etc…

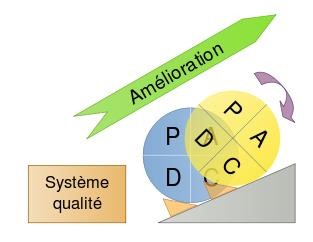
**D**O : Développer, réaliser, mettre en œuvre le plan d’action défini en P

**C**HECK : Contrôler, vérifier que les actions ont été réalisées et qu’elles ont permis d’obtenir le résultat attendu. Evaluer les écarts (comparer P et D)

**A**CT : Agir, ajuster, réagir

L’automatisation de test se situe nécessairement sur les deux derniers principes à savoir **CHECK** et **ACT** qui sont la vérification et l’ajustement ; par conséquent notre travail d’optimisation des processus de tests consiste à produire une démarche qualité assurant la fiabilité et la sécurité des logiciels, Appliquer avec rigueur le cycle **PDCA** est une des conditions nécessaires du succès. Il empêche de s’arrêter après la phase D. Les phases C et A sont l’essence même de l’amélioration continue (vérifier que l’on a obtenu les résultats attendus, corriger d’éventuels écarts, capitaliser sur ce que l’on a appris pour continuer à améliorer)

La figure suivante illustre bien le principe du PDCA.

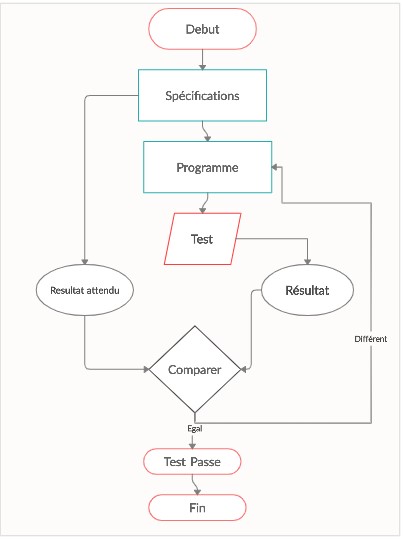
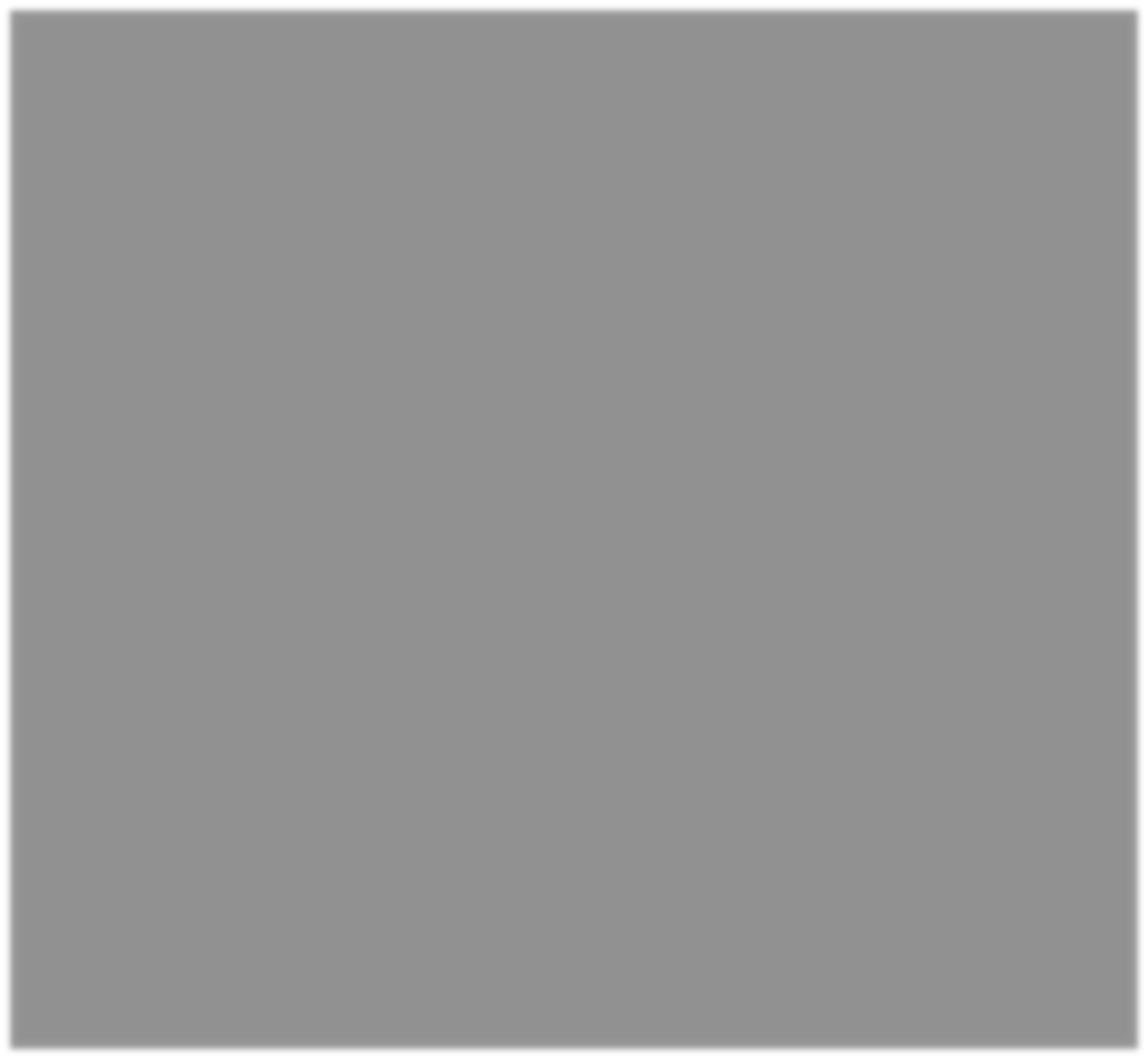


*Figure 3.1 : Roue de Deming*

Source : Image tiré de « La boite à outils du responsable qualité »

À chaque étape, la roue avance d’un quart de tour. Cette avancée représente l’action de progresser. Une cale est représentée sous la roue de Deming pour symboliser que les étapes PDCA s’enchaînent toujours dans cet ordre. Si chaque étape est bien respectée et conduite, ce déroulement empêche la roue de redescendre grâce au retour d’expériences, à un système d’audits réguliers ou à un système documentaire qui capitalise les bonnes pratiques et les décisions, que l’organisation soit lourde ou non, le **PDCA** est toujours applicable. La roue tourne simplement plus ou moins vite.

##### ORGANIGRAMME DU FONCTIONNEMENT DES TESTS



*Figure 3.2 : Organigramme de fonctionnement du test après optimisation*

Source : Par nos soins

D’une manière générale, tout commence par la spécification des besoins et le développement du logiciel, la procédure de test suit avec un résultat attendu, après le dit test, le résultat obtenu est comparé au résultat attendu ; si les résultats sont identiques alors la fonctionnalité est mise en attente pour les tests d’intégrations ; sinon l’on trouve l’on réitère le processus.

#### 3.Généralités sur le langage uml

UML (Uniform Model Language) est un langage de représentation destiné en particulier à la modélisation objet. UML est devenu une norme OMG en 1997. Ce langage de modélisation propose un formalisme qui impose de "penser objet" et permet de rester indépendant d'un langage de programmation donné. Pour ce faire, UML normalise les concepts de l'objet (énumération et définition exhaustive des concepts) ainsi que leur notation graphique. Il peut donc être utilisé comme un moyen de communication entre les étapes de spécification conceptuelle et les étapes de spécifications techniques.

##### DIAGRAMME DE CAS D’UTILISATION

Les **diagrammes de cas d'utilisation** (DCU) sont des diagrammes UML utilisés pour donner une vision globale du comportement fonctionnel d'un système logiciel. Ils sont utiles pour des présentations auprès de la direction ou des acteurs d'un projet, mais pour le développement, les cas d'utilisation sont plus appropriés. Un cas d'utilisation représente une unité discrète d'interaction entre un utilisateur (humain ou machine) et un système. Il est une unité significative de travail. Dans un diagramme de cas d'utilisation, les utilisateurs sont appelés acteurs (actors), ils interagissent avec les cas d'utilisation (use cases). Les diagrammes de cas d'utilisation sont souvent confondus avec les cas d'utilisation. Bien que ces deux concepts soient reliés, les cas d'utilisation sont bien plus détaillés que les diagrammes de cas d'utilisation. Cela permet donc de comprendre qui est l'acteur et ce que le système doit réaliser.

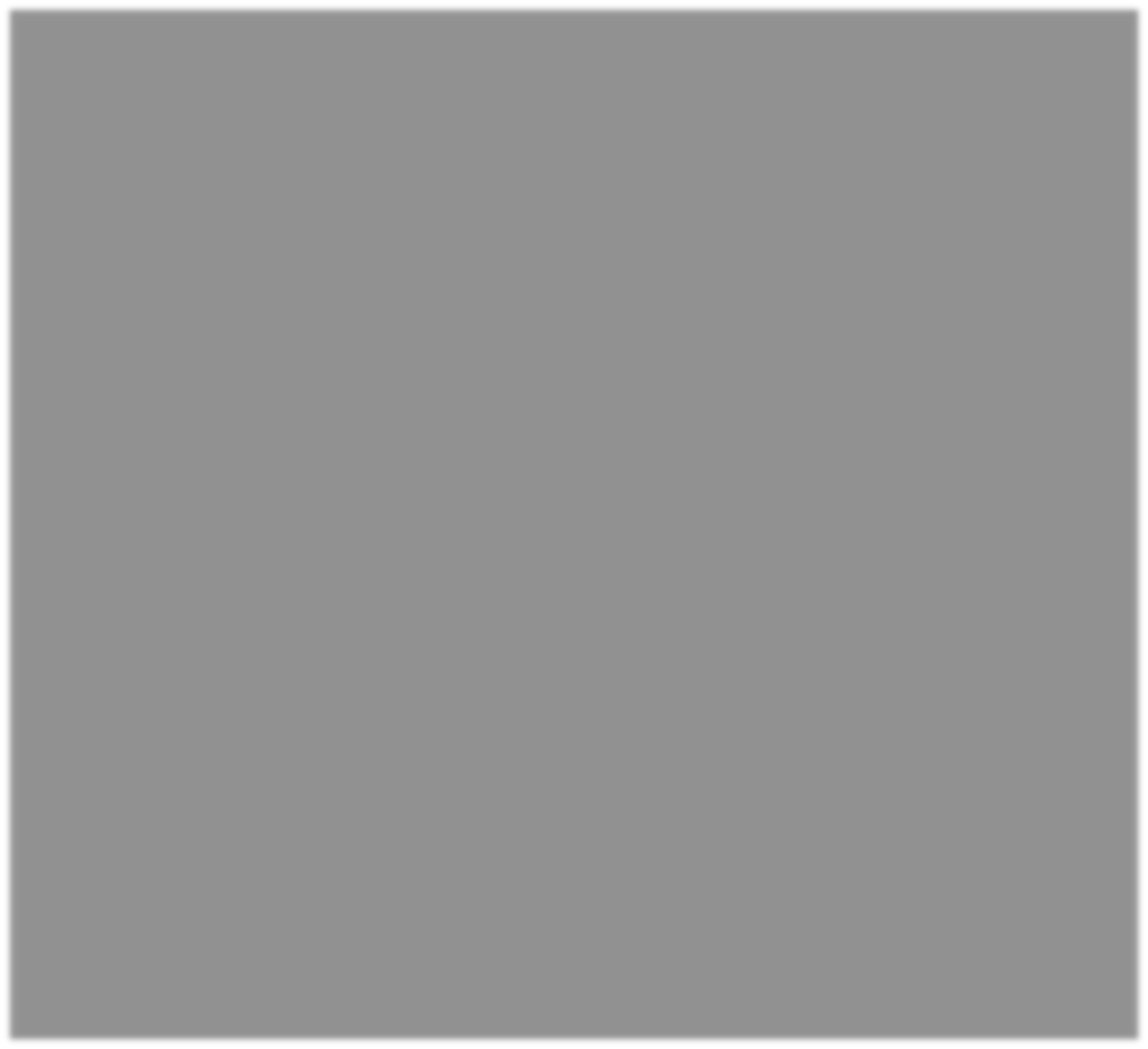
Les diagrammes de cas d’utilisation permettent de recueillir, d’analyser, d’organiser les besoins, et de recenser les grandes fonctionnalités d’un système. Il s’agit donc de la première étape UML d’analyse d’un système. Il offre un bon moyen de communication entre la maitrise d’ouvrage et le maitre d’œuvre. Les éléments constitutifs sont les acteurs, les cas d’utilisation, ainsi que les relations dans les cas d’utilisation.

**Acteur** : Un acteur est l’idéalisation d’un rôle joué par une personne externe, un processus ou une chose qui interagit avec un système. Le terme acteur ne désigne pas seulement les utilisateurs humains mais également les autres systèmes. Pour trouver les acteurs d’un système, il faut identifier quels sont les différents rôles que vont devoir jouer ses utilisateurs (ex :testeur, responsable d’agence, utilisateur final, . . .) mais aussi s’intéresser aux autres systèmes avec lesquels le système va devoir communiquer comme : Les périphériques manipulés par le système (imprimantes, distributeur de billet, . . .) , des logiciels déjà disponibles à intégrer dans le projet, des systèmes informatiques externes au système mais qui interagissent avec lui, etc.

Il convient de vérifier que les acteurs communiquent bien directement avec le système par émission ou réception de messages.

**Cas utilisation** : Pour identifier les cas d’utilisation, il faut se placer du point de vue de chaque acteur et déterminer comment et surtout pourquoi il se sert du système. Il faut éviter les redondances et limiter le nombre de cas en se situant (un bon niveau d’abstraction). Nommez les cas d’utilisation avec un verbe (l’infinitif suivi d’un complément) en vous plaçant du point de vue de l’acteur et non pas de celui du système.

La figure suivante présente le diagramme de cas d’utilisation de notre système



*Figure 3.3 : Cas d’utilisation du logiciel de test*

Source : Par nos soins

##### Outils de modélisation

Il existe beaucoup d’outils pour effectuer une analyse avec le langage UML. Certains de ces outils couvrent la totalité des digrammes proposés par le langage.

Dans le cadre de ce projet, nous utiliserons **Creately**, qui est Creately est un logiciel de diagramme et d'organigramme facile à utiliser conçu pour la collaboration en équipe. Prend en charge plus de 40 types de diagrammes et contient des milliers de modèles dessinés par des professionnels ; l’outil de modélisation crée tous les types de diagrammes, comme les flowcharts, diagrammes de Venn et bien sûr les diagrammes UML. L’outil en ligne a été écrit en HTML5 et convainc par un temps de réponse rapide. Avant de pouvoir l’utiliser, connectezvous avec **votre adresse email** ou un compte Google ou Facebook. Après la période d’essai de

14 jours, **la version gratuite** ne peut être utilisée qu’avec des fonctionnalités réduites. Toutefois, vous pouvez le mettre à niveau vers un compte Premium avec un abonnement mensuel. **L’outil Thème** permet de déterminer la famille de couleurs du diagramme, mais il est limité à des éléments de flèche dans la version test. En cliquant sur les classes, les composants, etc., vous pouvez toutefois toujours régler rapidement la couleur et la police des symboles. Vous pouvez également créer **différents niveaux** en un rien de temps en cliquant sur l’élément de menu correspondant (à l’extrême droite de la ligne d’outils). En outre, il n’est pas nécessaire de planifier la construction du diagramme étape par étape à l’avance. Vous pouvez facilement annuler les erreurs avec la fonction de suppression. Si vous voulez interposer un élément, il détecte le changement et propose des options d’insertion appropriées.



*Figure 4.1: Logo de Creately*

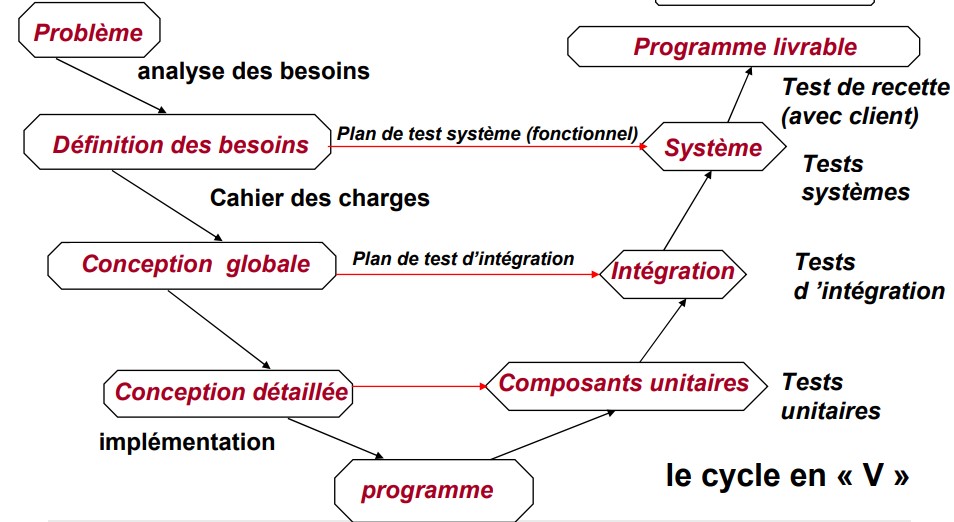
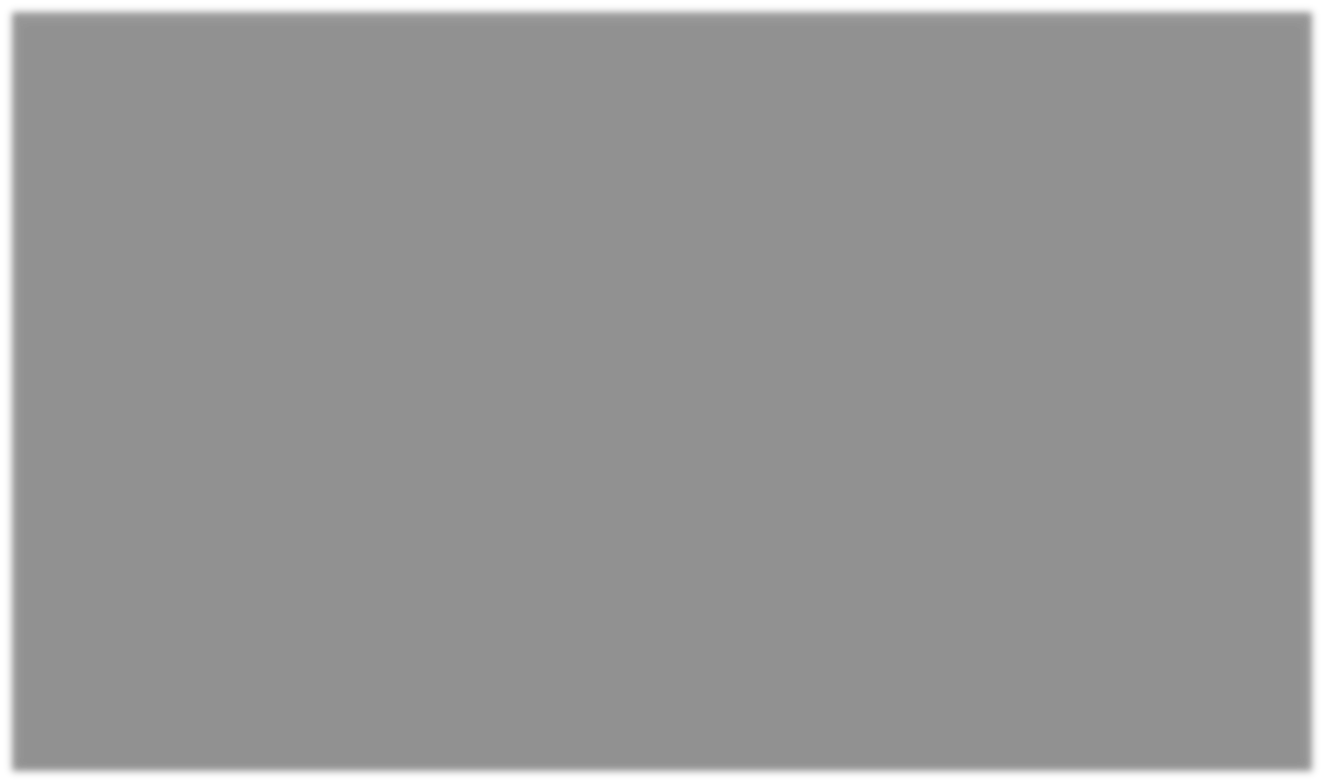
Source : creately.com

##### HIERARCHISATION DES TESTS

Cette démarche se déroule au moment de définir le cycle de vie d’un logiciel : prenons un exemple de **cycle en V** dont les éléments sont :

* Expression des besoins
* Conception globale
* Conception détaillée
* Implémentation
* Test unitaires
* Tests d’intégration
* Tests système
* Produit fini

Dans cette logique, le plan de test commence à la toute première phase qui est l’expression des besoins



*Figure 3.4 : Cycle de vie d’un logiciel, cycle en V*

Source : Yves Le Traon (France Telecom) et Benoit Baudry (IRISA)

### Section 3. ArchitEcturE généralE dE l’application

Dans cette section, nous verrons en détail l'architecture Selenium WebDriver. L'architecture de Selenium WebDriver concerne la manière dont Selenium fonctionne en interne. Nous savons que Selenium est un outil d'automatisation de navigateur qui interagit avec le navigateur et automatise les tests de bout en bout d'une application Web.

#### 1.Composition de Selenium

Selenium est une suite d'outils. Il se compose de **Selenium IDE**, **Selenium RC**, **Selenium Webdriver** et **Selenium Grid**.

**Selenium IDE :**

Selenium IDE (Integrated Development Environment) est un plugin Firefox. C'est le framework le plus simple de la suite Selenium. Cela nous permet d'enregistrer et de lire les scripts. Même si nous pouvons créer des scripts à l'aide de Selenium IDE, nous devons utiliser Selenium RC ou Selenium WebDriver pour écrire des cas de test plus avancés et plus robustes. **Selenium RC :**

Selenium RC était le principal projet Selenium pendant longtemps avant que la fusion WebDriver n'apporte Selenium 2. Selenium 1 est toujours activement pris en charge (en mode maintenance). Il s'appuie sur JavaScript pour l'automatisation. Il prend en charge Java, Javascript, Ruby, PHP, Python, Perl et C #. Il prend en charge presque tous les navigateurs.

**Remarque** : Selenium RC est officiellement obsolète.

**Selenium WebDriver :**

Selenium WebDriver est une infrastructure d'automatisation de navigateur qui accepte les commandes et les envoie à un navigateur. Il est implémenté via un pilote spécifique au navigateur. Il contrôle le navigateur en communiquant directement avec lui. Selenium WebDriver prend en charge Java, C #, PHP, Python, Perl, Ruby.

**Prise en charge du système d’exploitation :** Windows, Mac OS, Linux, **prise en charge du navigateur** : Solaris, Mozilla Firefox, Internet Explorer, Google Chrome 12.0.712.0 et supérieur, Safari, Opera 11.5 et supérieur, Android, iOS, HtmlUnit 2.9 et supérieur

**Selenium Grid :**

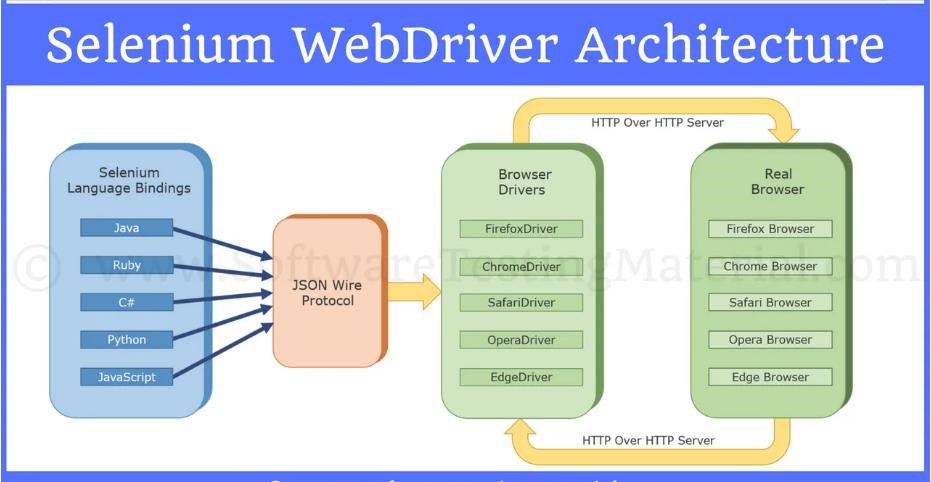
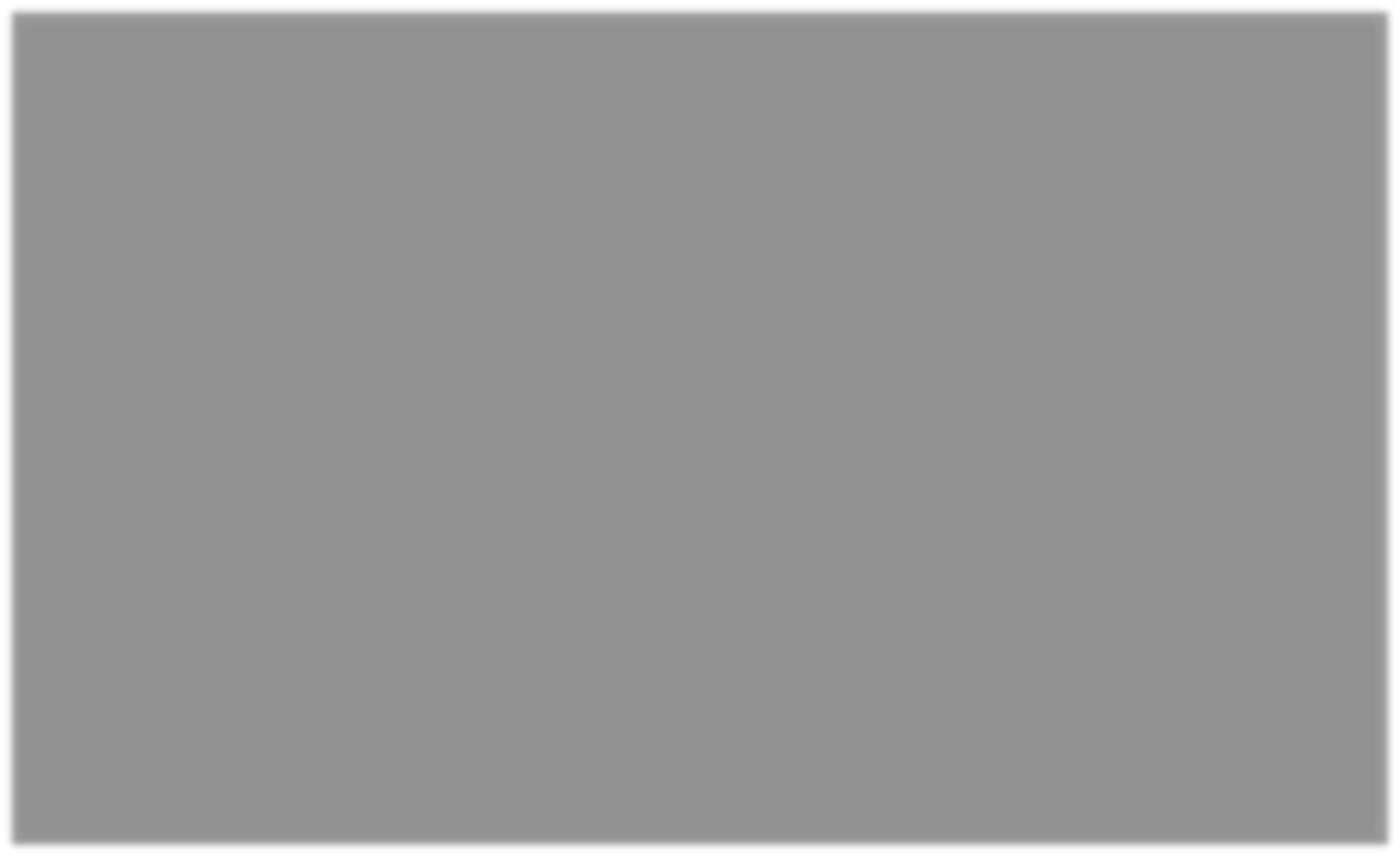
Selenium Grid est un outil utilisé avec Selenium RC pour exécuter des tests sur différentes machines contre différents navigateurs en parallèle. Autrement dit, exécuter plusieurs tests en même temps sur différentes machines exécutant différents navigateurs et systèmes d'exploitation.

Selenium WebDriver est une API orientée objet bien conçue qui prend en charge de nombreux langages tels que Java, C #, Python, etc.

L'interface de programmation d'application (API) fonctionne comme une interface entre divers composants logiciels. L'API Selenium Webdriver facilite la communication entre les langues et les navigateurs. Selenium prend en charge de nombreux langages de programmation tels que Java, C #, Python, etc., et prend également en charge plusieurs navigateurs tels que Google Chrome, Firefox, Internet Explorer, etc. Chaque navigateur a une logique différente pour effectuer des actions telles que le chargement d'une page, la fermeture du navigateur, etc...

*Figure 3.5 :*

*Architecture de Selenium WebDriver*



Source : softwaretestingmaterial.com

#### 2.dEtails dE l’architEcturE

Il existe quatre composants de l'architecture Selenium :

1. Bibliothèque cliente Selenium
2. Protocole JSON Wire sur HTTP
3. Pilotes de navigateur
4. Navigateurs

**Bibliothèques clientes Selenium / liaisons de langues :**

Selenium prend en charge plusieurs bibliothèques telles que Java, Ruby, Python, etc., les développeurs de Selenium ont développé des liaisons de langage pour permettre à Selenium de prendre en charge plusieurs langues

**Protocole JSON Wire sur le client HTTP :**

JSON signifie JavaScript Object Notation. Il est utilisé pour transférer des données entre un serveur et un client sur le Web. JSON Wire Protocol est une API REST qui transfère les informations entre le serveur HTTP. Chaque BrowserDriver (tel que FirefoxDriver, ChromeDriver, etc.) possède son propre serveur HTTP.

**Pilotes de navigateur :**

Chaque navigateur contient un pilote de navigateur distinct. Les pilotes de navigateur communiquent avec le navigateur respectif sans révéler la logique interne des fonctionnalités du navigateur. Lorsqu'un pilote de navigateur reçoit une commande, cette commande sera exécutée sur le navigateur respectif et la réponse reviendra sous la forme d'une réponse HTTP. **Navigateurs :**

Selenium prend en charge les navigateurs multi-tuyaux tels que Firefox, Chrome, IE, Safari etc…Voyons comment Selenium WebDriver fonctionne en interne :

En temps réel, vous écrivez un code dans votre interface utilisateur (par exemple Eclipse IDE) en utilisant l'une des bibliothèques clientes Selenium prises en charge (par exemple Java).



*Figure 3.6 : Compatibilité entre navigateurs*

Sources : Par nos soins

#### 3.Exemple

WebDriver driver = new FirefoxDriver () ; driver.get(https://www.softwaretestingmaterial.com)

Une fois que vous êtes prêt avec votre script, vous cliquerez sur Exécuter pour exécuter le programme. Sur la base des déclarations ci-dessus, le navigateur Firefox sera lancé et il naviguera vers le site Web de *softwartestingmaterial*. Nous voyons ici ce qui se passera en interne après avoir cliqué sur Exécuter jusqu'au lancement du navigateur Firefox.

Une fois que vous avez cliqué sur Exécuter, chaque instruction de votre script sera convertie en URL à l'aide du protocole JSON Wire sur HTTP. Les URL seront transmises aux pilotes du navigateur. (Dans le code ci-dessus, nous avons pris FirefoxDriver). Ici, dans notre cas, la bibliothèque cliente (java) convertira les instructions du script au format JSON et communiquera avec FirefoxDriver. L'URL ressemble à celle ci-dessous.

http://localhost:8080/ {"url » :"https://www.softwaretestingmaterial.com"}

Chaque pilote de navigateur utilise un serveur HTTP pour recevoir les requêtes HTTP. Une fois que l'URL atteint le pilote du navigateur, le pilote du navigateur transmettra cette demande au navigateur réel via HTTP. Ensuite, les commandes de votre script sélénium seront exécutées sur le navigateur.

* Si la demande est une demande *POST,* il y aura une action sur le navigateur
* Si la demande est une demande *GET,* la réponse correspondante sera générée à l'extrémité du navigateur et elle sera envoyée via HTTP au pilote de navigateur et au pilote de navigateur via JSON Wire Protocol et l'envoie à l'interface utilisateur (Eclipse IDE).

Ce chapitre présente en détails l’analyse approfondie des différents tests ensuite une généralité sur le langage UML et enfin l’architecture générale de Selenium

## Chapitre IV: Résultats obtenus et discussion

Ce chapitre présente le résultat de notre implémentation en précisant les outils et méthodes utilisés, puis le langage utilisé et enfin l’approche de développement

### Section 1: outils Et méthodEs d’implémentation

#### 1.Selenium IDE

IDE (Integrated Development Environment) est l'outil que vous utilisez pour développer vos cas de test Selenium. C'est une extension Chrome et Firefox facile à utiliser et c'est généralement le moyen le plus efficace de développer des cas de test. Il enregistre les actions des utilisateurs dans le navigateur, à l'aide des commandes Selenium existantes, avec des paramètres définis par le contexte de cet élément, IDE Selenium est conçu pour enregistrer les interactions avec les sites Web afin de générer et à maintenir l'automatisation du site, les tests et vous évite d'avoir à parcourir manuellement les prises répétitives. Les fonctionnalités incluent l’enregistrement et lecture des tests sur Firefox et Chrome, l’organisation des tests en suites pour une gestion facile, la sauvegarde et chargement de scripts, pour une lecture ultérieure, la prise en charge de Selenium 3.

#### 2.Langages utilisés

Le langage utilisé ici est le PHP, c’est un "langage de programmation" libre, principalement utilisé pour produire des pages Web dynamiques via un serveur HTTP, mais pouvant également fonctionner comme n'importe quel langage interprété de façon locale. PHP est un langage impératif orienté objet. PHP a permis de créer un grand nombre de sites web célèbres, comme Facebook et Wikipédia Il est considéré comme une des bases de la création de sites web dits dynamiques mais également des applications web.

PHP est un langage de script utilisé le plus souvent côté serveur : dans cette architecture, le serveur interprète le code PHP des pages web demandées et génère du code (HTML, XHTML, CSS par exemple) et des données (JPEG, GIF, PNG par exemple) pouvant être interprétés et rendus par un Navigateur web. PHP peut également générer d'autres formats comme le WML, le SVG et le PDF.

Il a été conçu pour permettre la création d'applications dynamiques, le plus souvent développées pour le Web. PHP est le plus souvent couplé à un serveur Apache bien qu'il puisse être installé sur la plupart des serveurs HTTP tels que IIS ou Nginx. Ce couplage permet de récupérer des informations issues d'une base de données, d'un système de fichiers (contenu de fichiers et de l'arborescence) ou plus simplement des données envoyées par le navigateur afin d'être interprétées ou stockées pour une utilisation ultérieure. C'est un langage peu typé et souple et donc facile à apprendre par un débutant mais, de ce fait, des failles de sécurité peuvent rapidement apparaître dans les applications. Pragmatique, PHP ne s'encombre pas de théorie et a tendance à choisir le chemin le plus direct. Néanmoins, le nom des fonctions (ainsi que le passage des arguments) ne respecte pas toujours une logique uniforme, ce qui peut être préjudiciable à l'apprentissage. Son utilisation commence avec le traitement des formulaires puis par l'accès aux bases de données. L'accès aux bases de données est aisé une fois l'installation des modules correspondants effectuée sur le serveur. La force la plus évidente de ce langage est qu'il a permis au fil du temps la résolution aisée de problèmes autrefois compliqués et est devenu par conséquent un composant incontournable des offres d'hébergements. Il est multi-plateforme : autant sur Linux qu'avec Windows il permet aisément de reconduire le même code sur un environnement à peu près semblable (prendre en compte les règles d'arborescences de répertoires qui peuvent changer). Libre, gratuit, simple d'utilisation et d'installation, ce langage nécessite comme tout langage de programmation une bonne compréhension des principales fonctions usuelles ainsi qu'une connaissance aiguë des problèmes de sécurité liés à ce langage. La version 5.3 a introduit de nombreuses fonctions nouvelles : les espaces de noms (Namespace), un élément fondamental de l'élaboration d'extensions, de bibliothèques et de frameworks structurés, les fonctions anonymes, les fermetures, etc…

## 3.Approche de développement

L’approche de développement prônée ici est l’approche agile, c’est une nouvelle approche centrée sur la satisfaction du besoin du client et non sur les termes contractuels. Le projet est découpé en petit bloc puis les blocs sont hiérarchisés en fonction du besoin du client. Cela permet d’éviter le superflu et de se concentrer au début de chaque cycle sur ce qui est de la valeur pour l’utilisateur final. Le feedback permanent devient une règle et des validations du client à chaque étape. Le changement est autorisé et doit être encouragé même tardivement car c’est un avantage décisif qui permet de ne pas se priver des idées en cours de route et surtout d’éliminer les mauvaises idées lancées en début du projet. Elle favorise la co-construction en intégrant l’annonceur lui-même dans le travail quotidien. Au niveau de l’agile, la documentation n’est pas nécessaire. Ce qui permet de réaliser le projet rapidement en acceptant tous les changements pour répondre au besoin du client. La méthode agile garantie la satisfaction du client et non la conformité au terme d’un contrat de développement.

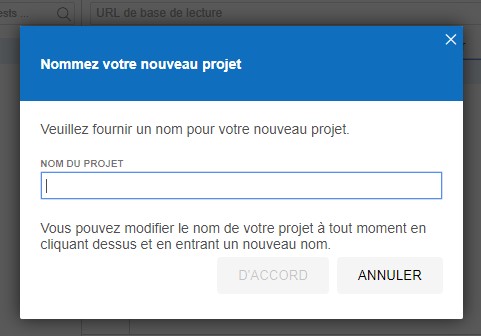
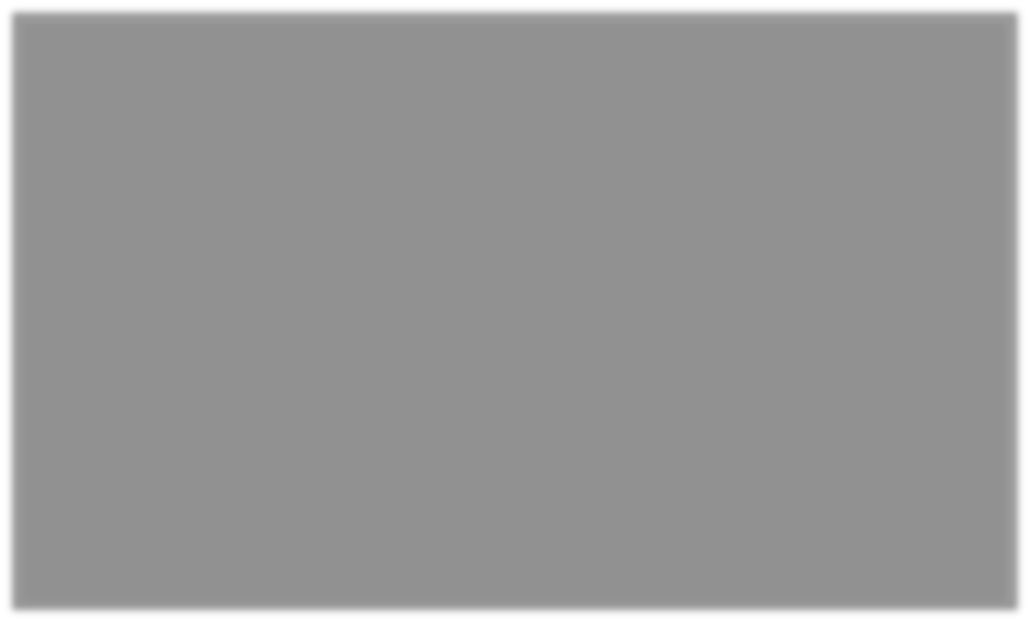
Malgré les avantages de l’approche **agile**, elle a aussi ses limites et inconvénients.

L’exigence de cette méthode sur les réunions quotidiennes en physique et la collaboration de façon rapprochée constituent une limite pour les grands projets qui peuvent nécessiter des intervenants et développeurs extérieures qui sont dans un espace géographique très large. Ses exigences sur le développement progressif et adaptatif ne favorisent pas les clients qui veulent un planning de développement et un calendrier fixé. Aussi le manque de documentation peut constituer un frein pour la maintenance et l’évolution du projet.

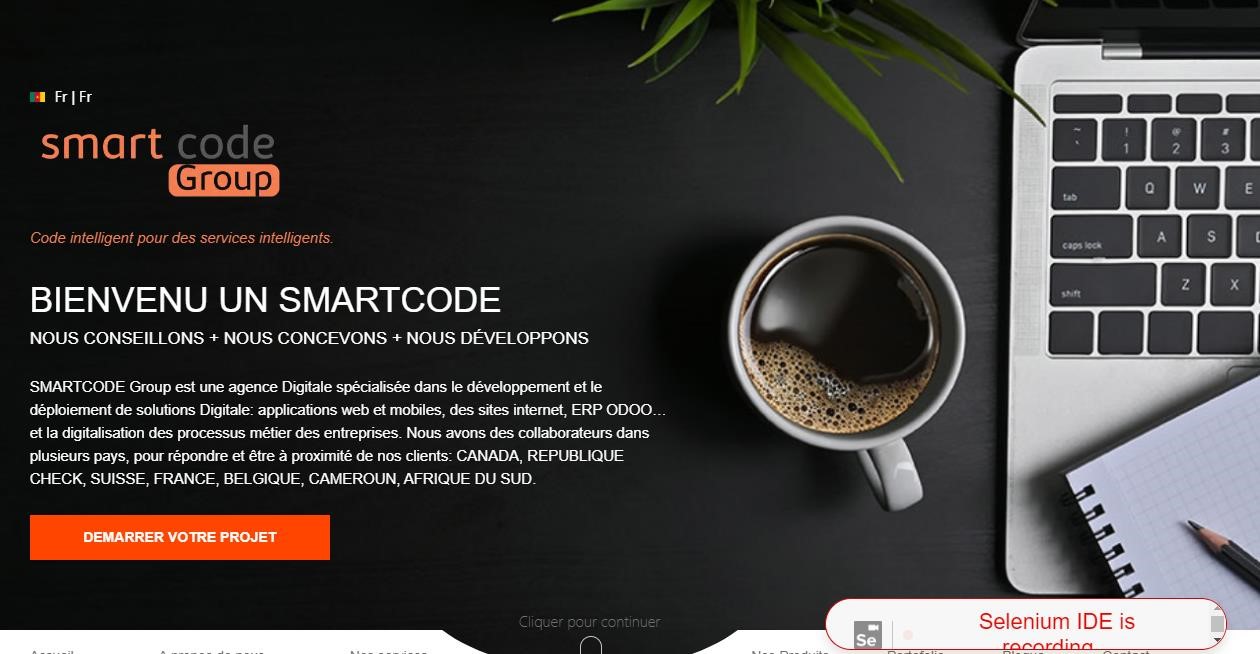
# Section 2: Présentation de la solution réalisée

Démarrage du projet

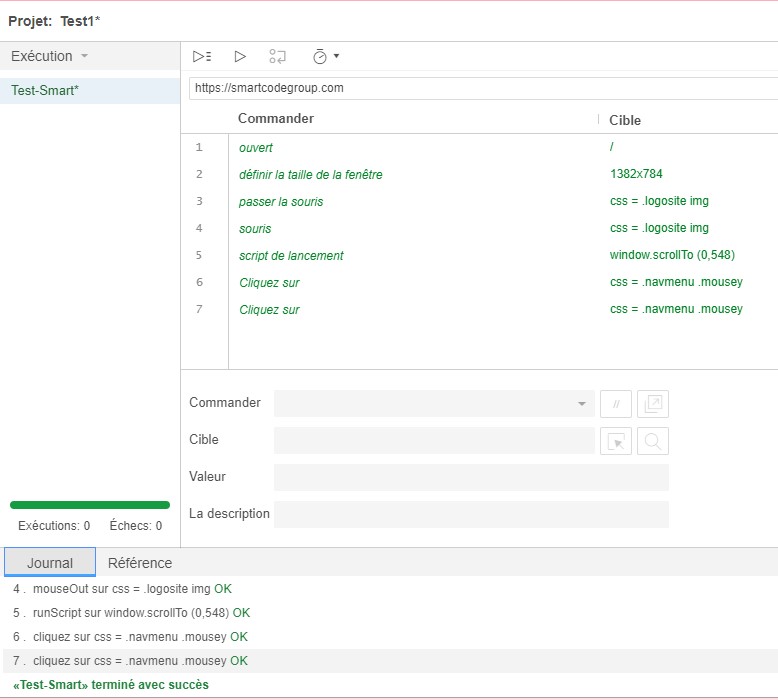
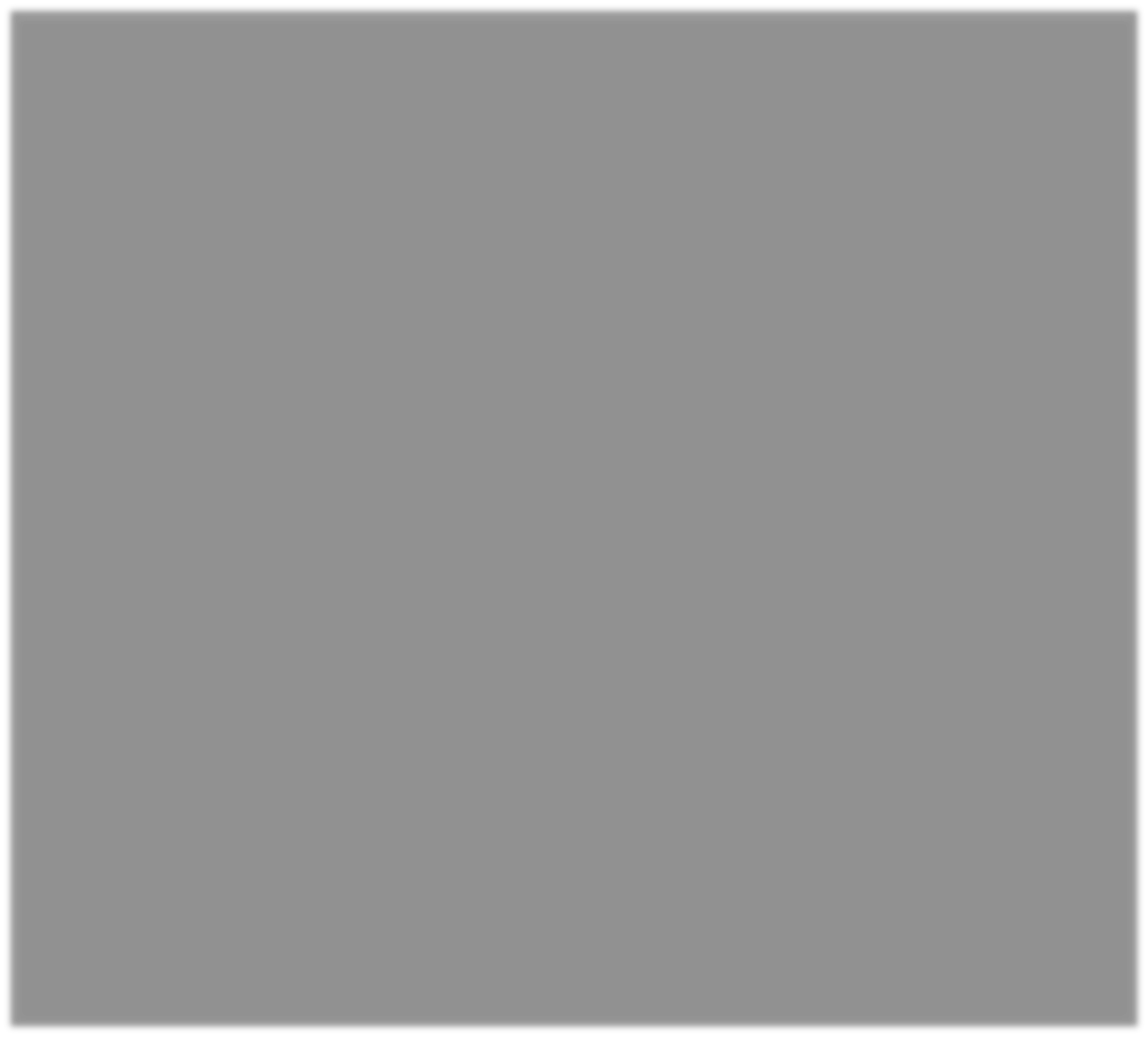
Commencer par entrer le nom du projet dont vous voulez réaliser le test



*Figure 4.2: Enregistrement du projet* Ensuite entrer l’URL du site web à tester et lancer l’enregistrement



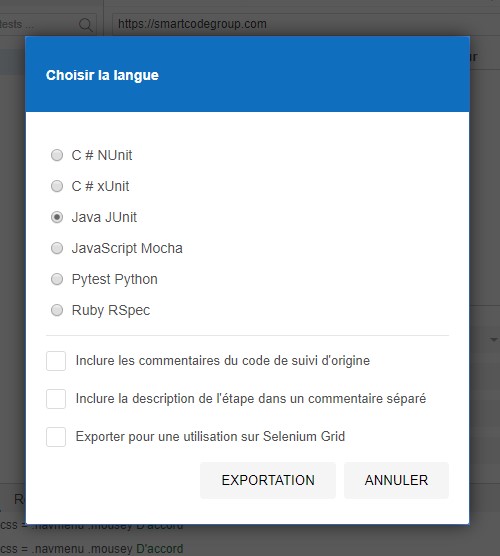
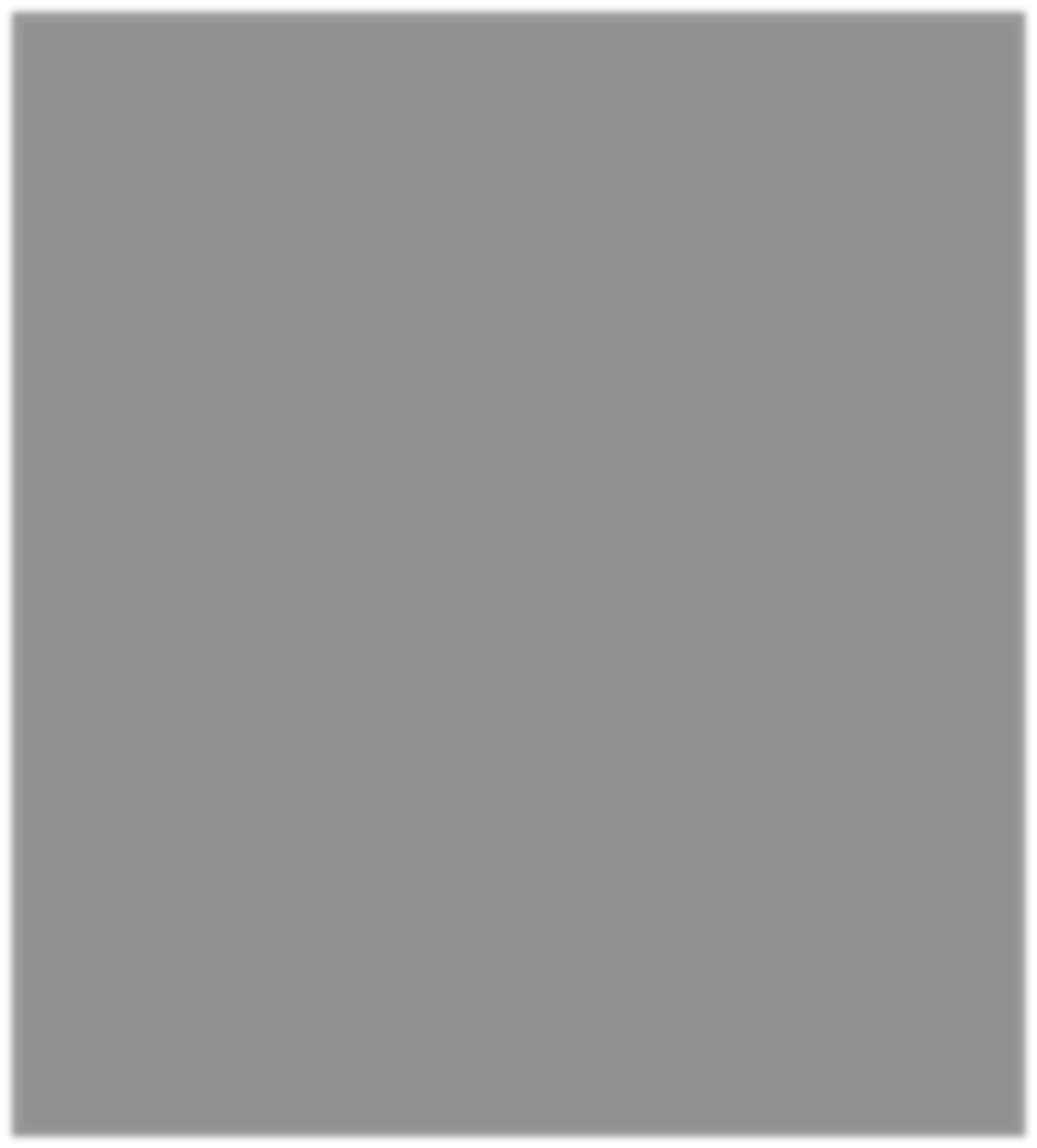
*Figure 4.3: Démarrage de test du site smart code group*



*Figure 4.4: Cas de test effectué avec succès*

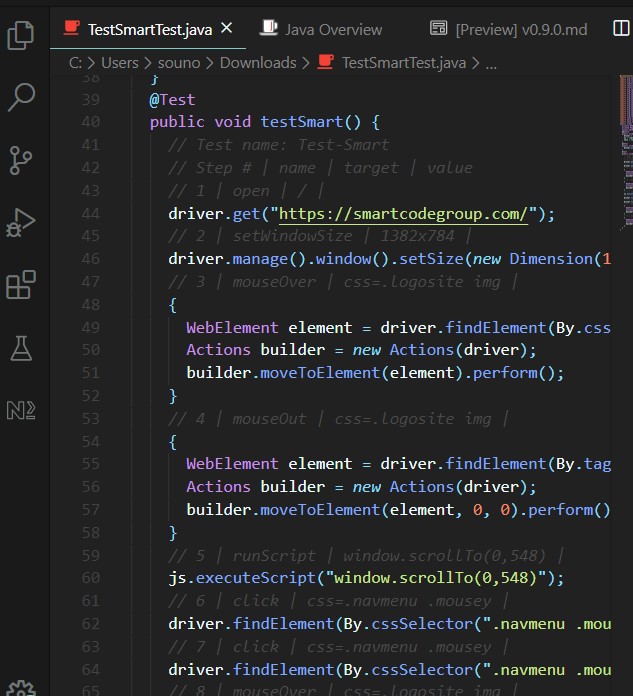
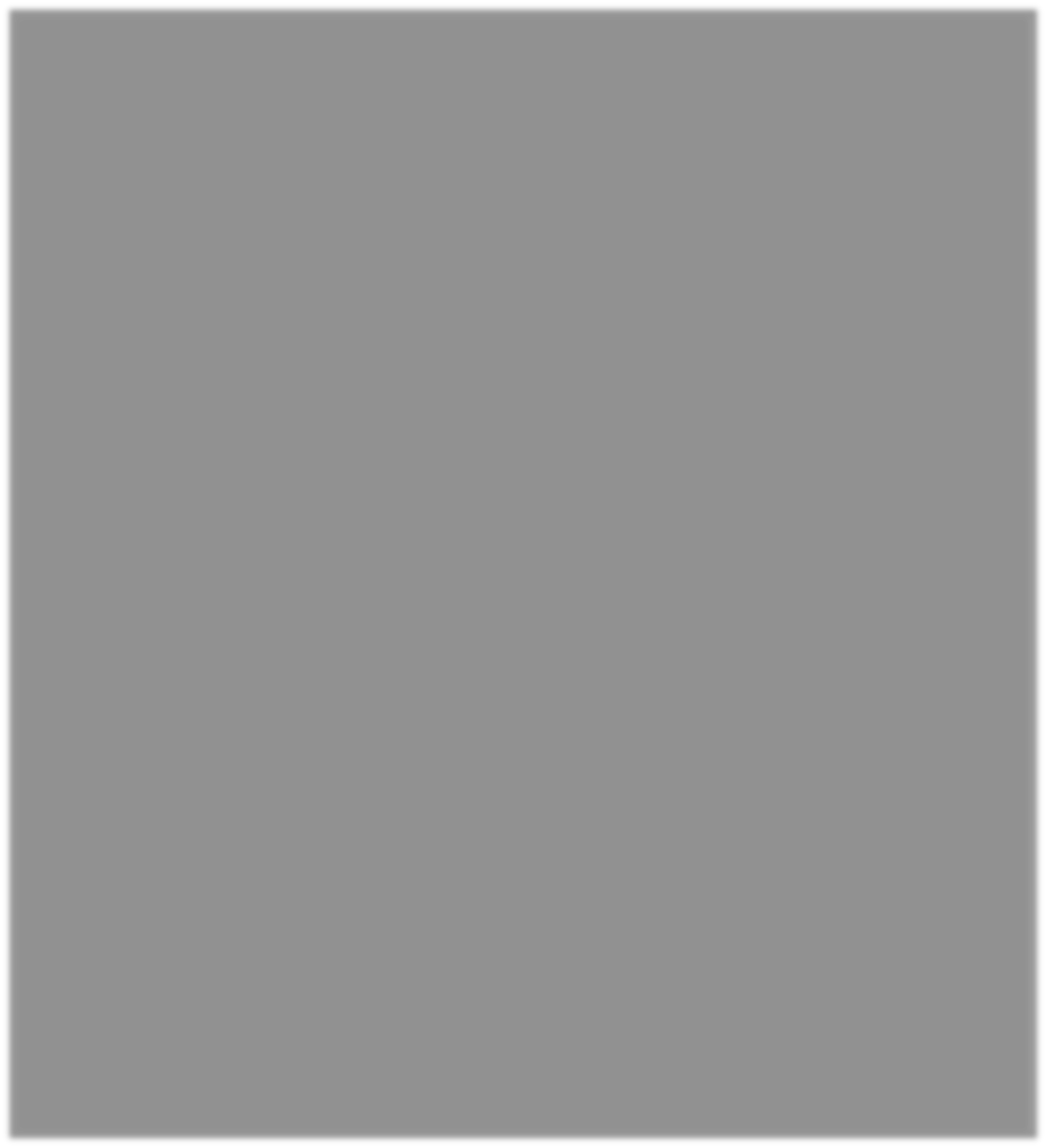
La couleur verte indique que, le test a été effectué avec succès, l’on remarque que tout est **Ok**, et le message renvoyé est « **Test-Smart terminé avec Succès** »

Pour l’exportation dans l’IDE, enregistrer le test effectuer puis cliquer sur **Exportation**, sélectionner la langue et cliquez sur le bouton **EXPORTATION.**

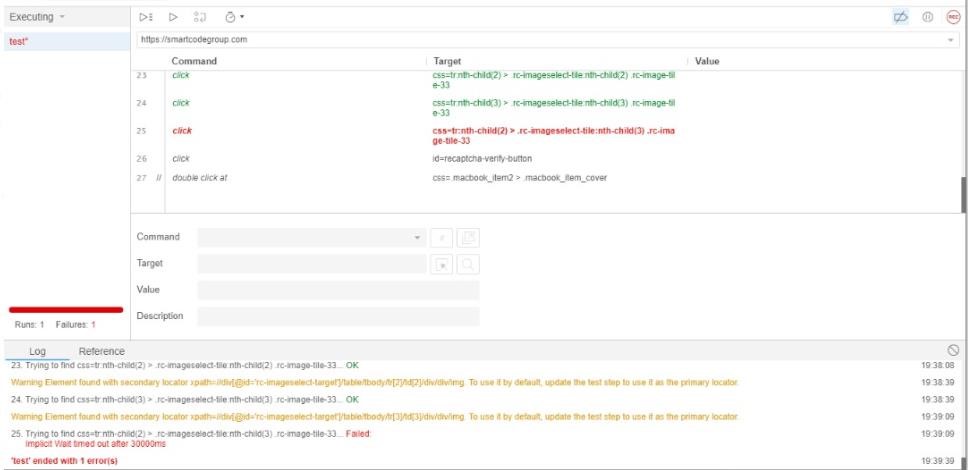
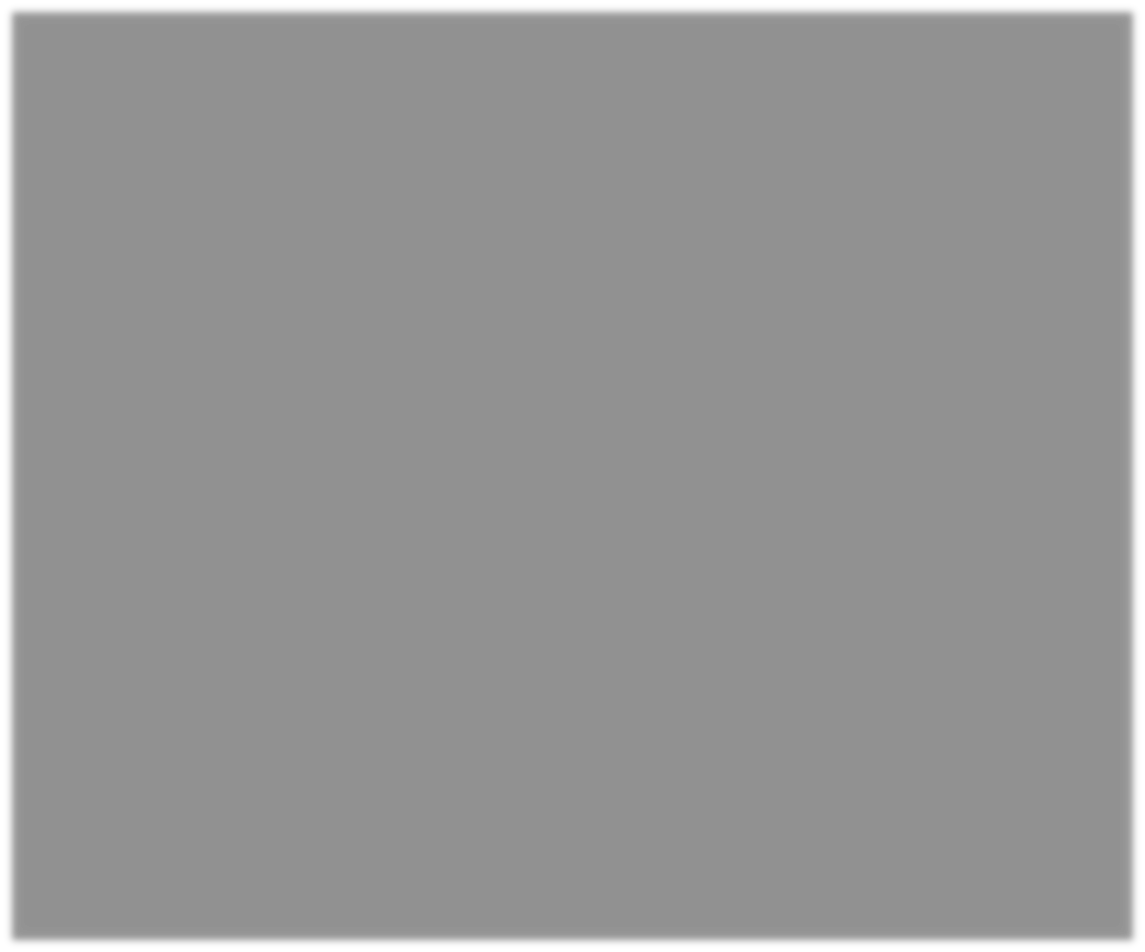


*Figure 4.5: Exportation du test dans un IDE*

Après exportation, nous pouvons commencer à écrire les scripts de tests dans l’environnement Visual Studio Code.



*Figure 4.6: Ouverture du test sur Visual Studio Code*



*Figure 4.7: Test effectué avec erreur*

La couleur **rouge** indique qu’il y’a une erreur dans le code.

# Section 3: Evaluation des résultats et impact sur l’EntrEprisE

1.INTERETS DES TESTS AUTOMATISES

Les systèmes d’information sont amenés à évoluer régulièrement de par la nécessité d’intégrer de nouvelles applications, de les mettre à jour ou de mettre en œuvre les évolutions à la demande des différentes entités métiers de l’entreprise. Les stratégies digitales d’omni canal accélèrent encore un peu plus les besoins de transformation des systèmes d’information qui doivent intégrer, de plus en plus, d’applications hétérogènes dans des délais de plus en plus bref.

Comment dans ces conditions assurer la fiabilité des systèmes d’information ?

Il faut le tester afin de vérifier que les modifications n’ont pas apporté d’instabilités ou d’anomalies, aux conséquences parfois imprévisibles. En informatique aussi, un battement d’ailes d’un papillon peut provoquer une tornade. Exemples récents, près d’un million de clients d’un producteur d’électricité se sont vus prélever deux fois leur facture…, un organisme hospitalier perd 80 millions d’euros de factures impayés suite à un changement de système informatique…

Ces deux exemples illustrent à quel point les tests, et plus particulièrement les tests de non régression, sont cruciaux pour les entreprises. Pourtant, ils sont quelques fois négligés car chronophages et fastidieux pour les équipes. De plus, la phase de test de non régression est en bout de chaîne de tous projets menés en mode classique ou agile, donc soumise aux contraintes fortes des plannings et du budget. Les tests de non régressions automatisées prennent, alors, tout leur sens.

Une fois opérationnel, l’automatisation des tests a apporté de nombreux avantages à Smart Code Group à savoir :

* Le gain de temps dû à un allègement de la charge de travail ; en effet des fonctionnalités qui se traitaient en 2 jours se sont réduite en une demi-journée.
* L’augmentation de la productivité,
* La fiabilité et la qualité des tests effectués,
* L’accroissement du niveau de complétude des tests réalisés,
* La démobilisation des équipes sur les tâches répétitives au profit de tâches à plus grande valeur ajoutée (Analyse des résultats, définition des cas de tests, planification…).

2.FONCTIONNEMENT DES TESTS FONCTIONNEL

Aux premiers abords, le fonctionnement d’un automate de tests semble très simple. De ce fait, les avantages paraissent évidents pour les propriétaires d’un tel outil.

Beaucoup voient l’automate de tests comme un robot qui effectue des actions réalisées par un utilisateur via un outil de recording (enregistrement du mouvement de la souris, de clics de souris, etc.) et les répète à volonté. Les scénarios de tests peuvent ainsi être réalisés à l’infini sans surveillance (donc le soir ou pendant les jours non ouvrés). Une fois l’ensemble des cas de tests joués par le robot, celui-ci produit un rapport d’exécution qui permet d’analyser les tests effectués. Toutefois, dans la réalité, avant de pouvoir jouer des cas de tests « tout seul », le robot de tests nécessite d’être paramétré et les scénarios de tests codés. Ces tâches requièrent bien souvent de solides connaissances en développement.

D’un point de vue technique, les automates de tests fonctionnent par reconnaissance d’objets des interfaces homme-machine (boutons, listes déroulantes, etc.) qu’ils stockent dans un répertoire. En mode « enregistrement » (Via un outil de recording), ils capturent les interactions réalisées sur des objets lors du déroulement du scénario de test. Un script (un programme) est ainsi généré automatiquement dans le langage de référence de l’outil utilisé.

Cependant, afin d’obtenir des résultats de tests structurants, il est nécessaire d’introduire des points de vérification/contrôle, des points de synchronisation/d’attente et autres résultats attendus directement à l’intérieur du programme et, parfois, de corriger des parties de programmes écrits automatiquement par l’outil de recording. Bien souvent, ces actions doivent être programmées manuellement (en codant des scripts) car on atteint vite les limites du mode

« enregistrement » du robot de tests. Une fois l’ensemble des cas de tests traduits sous forme de script, on peut enfin activer le mode exécution, le robot joue alors les différents scénarios de tests à partir de ce script avec les données choisies par le testeur.

Par conséquent, il faut bien garder à l’esprit que le robot de tests reste un outil complexe à utiliser. Le niveau de complexité de l’outil varie fortement en fonction du robot adopté. En effet, le langage supporté par l’outil et les fonctionnalités qu’il propose sont des facteurs structurants à ne pas négliger lors du choix définitif. Il faut aussi noter que l'automate de tests fonctionnels diffère d'un outil de test de charge et de performance. En effet, il ne permet pas de mesurer les temps de réponse d’une application ni sa montée en charge. Il a pour objectif de réaliser des tests de non régression ou des tests fonctionnels.

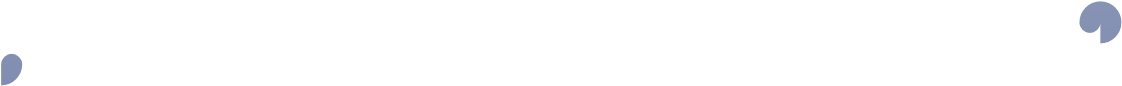
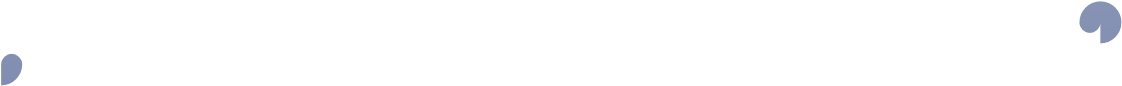
3.Perspectives

À la suite de notre projet, nous envisageons des perspectives à savoir la création des tests de sécurité et la Business Inteligence. Un Test de sécurité permet de mesurer l’exposition d’un site web ou des applications aux menaces informatiques. Appelé aussi test d’intrusion ou **test de sécurité web** il a pour but d’identifier des actions permettant de réduire les risques de compromissions et de perte de données.

Lorsqu’un site web est mis en ligne, les entreprises sous-estiment l’envergure de la surface exposée, et l’étendue des vulnérabilités qui peuvent s’y trouver. De plus, il n’est pas toujours évident de surveiller les incidents de sécurité de toutes les technologies utilisées sur un site… Un **test d’intrusion applicatif** permet de mettre en évidence les points sensibles de vos plateformes, afin de vous permettre de mettre en place les bonnes pratiques et actions correctives qui amélioreront leur sécurité

La business Inteligence ou l’informatique décisionnelle est l'informatique à l'usage des décideurs et des dirigeants d'entreprises. Elle désigne les moyens, les outils et les méthodes qui permettent de collecter, consolider, modéliser et restituer les données, matérielles ou immatérielles, d'une entreprise en vue d'offrir une aide à la décision et de permettre à un décideur d’avoir une vue d’ensemble de l’activité traitée.

Ce type d’application repose sur une architecture commune dont les bases théoriques viennent principalement de Ralph Kimball, Bill Inmon et DanLinstedt.



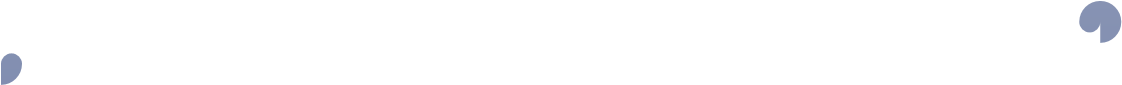
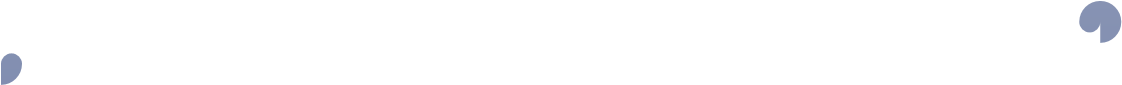
Conclusion

générale

En définitif, ce projet a été effectué dans le but d’améliorer le processus de test et validation logiciel, et la réalisation de ce projet concernant l’optimisation du processus de test et validation logiciel a été d’un grand apport dans la mesure où il nous a permis de proposer une solution en développant une application répondant aux besoins de l’entreprise, malgré les contraintes rencontrées dans cette démarche.

Tout au long de ce projet, nous sommes investis à analyser le fonctionnement de test et validation logiciel de Smart Code Group, laquelle analyse nous a permis de desceller des dysfonctionnements que nous avons jugés susceptibles d’influencer le rendement et la productivité de l’entreprise à court et à long terme. C’est fort de cette remarque que nous avons retenu le thème : **« Optimisation du processus de test et validation logiciel ».**

Toutefois, la réalisation de notre projet qui a été faite à moins 100%, nous restons convaincus qu’une appréciation de la part de nos enseignants et toute autre personne du domaine ou pas sera la bienvenue pour l’amélioration des travaux. Rendus au terme de notre investigation, notre modeste contribution à ce sujet constitue pour nous un pas vers l’univers professionnel. Ainsi, conscient des limites d’un tel exploit, nous restons ouverts à toutes critiques et remarques qui nous inciterait à persévérer sur le chemin de la perfection.



Annexes

**Annexe 1** : Procédure d’installation de Selenium

##### Installation ou configuration

Pour commencer à utiliser WebDriver, vous devez obtenir le pilote approprié sur le site :

Téléchargements Selenium HQ. À partir de là, vous devez télécharger le pilote correspondant au (x) navigateur (s) et / ou à la (aux) plate-forme (s) sur lesquels vous essayez d'exécuter WebDriver. Par exemple, si vous testiez Chrome, le site Selenium vous indiquera :

Pour télécharger.

Enfin, avant de pouvoir utiliser WebDriver, vous devez télécharger les liaisons de langue appropriées, par exemple si vous utilisez C #, vous pouvez accéder au téléchargement depuis la page Selenium HQ Downloads pour obtenir les fichiers .dll requis ou les télécharger en tant que packages dans Visual Studio. Via le gestionnaire de paquets Nugget.

Les fichiers requis doivent maintenant être téléchargés. Pour plus d'informations sur l'utilisation de WebDriver, reportez-vous à la documentation de selenium-webdriver.

##### Pour Visual Studio

La méthode la plus simple pour installer Selenium WebDriver consiste à utiliser un gestionnaire de paquets NuGet.

Dans votre projet, cliquez avec le bouton droit sur "Références" et cliquez sur "Gérer les packages NuGet" comme indiqué :

Ensuite, tapez dans la zone de recherche " *webdriver* ". Vous devriez alors voir quelque chose comme ceci :

Installez " **Selenium.WebDriver** ", et " **Selenium.Support** " (le package de support inclut des ressources supplémentaires, telles que Wait) en cliquant sur le bouton Installer sur le côté droit.

Ensuite, vous pouvez installer vos WebDrivers que vous souhaitez utiliser, comme l'un de ceux-ci :

* Selenium.WebDriver. ChromeDriver (Google Chrome)
* PhantomJS.

**Selenium** est un ensemble d'outils conçus pour automatiser les navigateurs. Il est couramment utilisé pour les tests d'applications Web sur plusieurs plates-formes. Quelques outils sont disponibles sous le parapluie Selenium, tels que Selenium WebDriver (ex-Selenium RC), Selenium IDE et Selenium Grid.

**WebDriver** est une *interface de* contrôle à distance qui vous permet de manipuler des éléments DOM dans des pages Web et de contrôler le comportement des agents utilisateurs. Cette interface fournit un langage qui a été implémenté pour diverses plates-formes telles que :

* (Mozilla Firefox)
* (Google Chrome)
* (Apple Safari)
* InternetExplorerDriver (MS InternetExplorer)
* MicrosoftWebDriver ou Edge Driver (MS Edge)
* (navigateur Opera)

**Selenium WebDriver** est l'un des outils de Selenium qui fournit des API orientées objet dans divers langages pour permettre un meilleur contrôle et l'application de pratiques de développement de logiciels standard. Pour simuler avec précision la manière dont un utilisateur interagira avec une application Web, il utilise les « événements de niveau du système d'exploitation natif », contrairement aux « événements JavaScript synthétisés ».

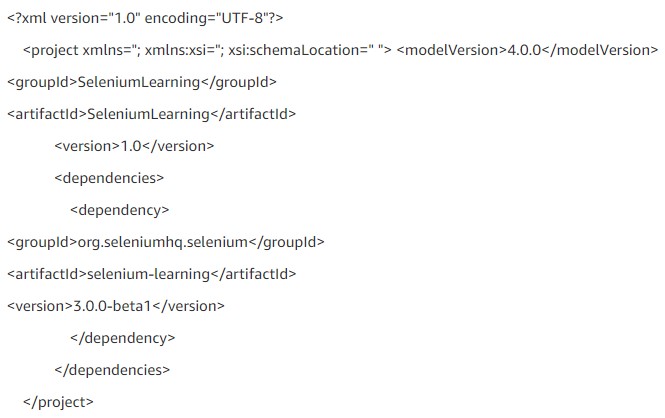
##### Installation ou configuration pour Java

Pour écrire des tests en utilisant Selenium WebDriver et Java comme langage de programmation, vous devez télécharger les fichiers JAR de Selenium WebDriver sur le site Web de Selenium.

Il existe plusieurs façons de configurer un projet Java pour le WebDriver Selenium, l'un des plus faciles à utiliser est d'utiliser Maven. Maven télécharge les liaisons Java requises pour Selenium WebDriver, y compris toutes les dépendances. L'autre méthode consiste à télécharger les fichiers JAR et à les importer dans votre projet.

**Étapes pour configurer le projet Selenium WebDriver à l'aide de Maven :**

1. Installez Maven sur la fenêtre Windows suivant ce document :
2. Créer un dossier avec le nom selenium-learing
3. Créez un fichier dans le dossier ci-dessus en utilisant n'importe quel éditeur de texte 4. Copiez le contenu de la figure ci-dessous



*Figure : Code d’initialisation* Source : cours-framework-java/ebook-pour-apprendre-le-framework-selenium-webdriver

**Remarque** : assurez-vous que la version que vous avez spécifiée ci-dessus est la plus récente. Vous pouvez vérifier la dernière version à partir d’ici :

1. À l'aide de la ligne de commande, exécutez la commande ci-dessous dans le répertoire du projet. mvn clean install

La commande ci-dessus téléchargera toutes les dépendances requises et les ajoutera ensuite au projet.

1. Écrivez ci-dessous la commande pour générer un projet Eclipse que vous pouvez importer dans l'IDE Eclipse. mvn eclipse: eclipse
2. Pour importer le projet dans Eclipse ide, vous pouvez suivre les étapes ci-dessous

Ouvrez Eclipse -> Fichier -> Importer -> Général -> Projet existant dans l’espace de travail -> Suivant -> Parcourir -> Localisez le dossier contenant -> Ok -> Terminer

Installez le plug-in m2eclipse en cliquant avec le bouton droit sur votre projet et sélectionnez Maven -> Activer la gestion des dépendances.

##### Étapes pour configurer le projet Selenium Webdriver à l'aide de fichiers Jar

1. Créez un nouveau projet dans Eclipse en suivant les étapes ci-dessous.

Ouvrir Eclipse -> Fichier -> Nouveau -> Projet Java -> Fournir un nom (apprentissage du sélénium) -> Terminer

1. Téléchargez les fichiers jar depuis. Vous devez télécharger à la fois **Selenium Standalone Server** et **Selenium Client & WebDriver Language Bindings**. Puisque ce document parle de Java, vous devez télécharger uniquement jar depuis la section Java.

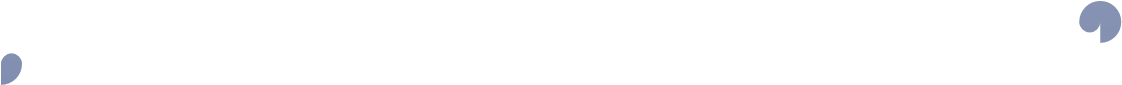
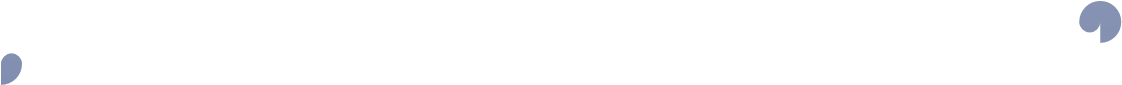
Remarque : Selenium Standalone Server est requis uniquement si vous souhaitez utiliser le serveur distant pour exécuter les tests. Comme ce document est tout au-dessus de la mise en place du projet, il est donc préférable d'avoir tout en place.

1. Les pots seront téléchargés dans un fichier zip, décompressez-les. Vous devriez pouvoir voir directement .jar
2. Dans éclipse, cliquez avec le bouton droit sur le projet que vous avez créé à l'étape 1 et suivez les étapes ci-dessous.

Propriétés -> Java Build Path -> Sélectionnez l'onglet Bibliothèques -> Cliquez sur Ajouter des fichiers Jars externes -> Localisez le dossier JAR décompressé que vous avez téléchargé ci-dessus -> Sélectionnez tous les fichiers lib dossier lib -> Cliquez sur OK > Localisez le même dossier décompressé -> Sélectionnez le fichier jar qui se trouve en dehors du dossier lib () -> Ok

De même, ajoutez le Selenium Standalone Server après l'étape ci-dessus.

1. Vous pouvez maintenant commencer à écrire du code de sélénium dans votre projet.



Tables de mat

ières

Dédicace................................................................................................................................. i

Remerciements ...................................................................................................................... ii

Sommaire ............................................................................................................................. iii

Abréviations ......................................................................................................................... iv

Liste des figures ................................................................................................................... vi

Liste des tableaux ................................................................................................................ vii

Résumé .............................................................................................................................. viii

Abstract ................................................................................................................................ ix

Introduction générale ..............................................................................................................x

[Partie I: 1](#_Toc147754)

[Etat de l’art 1](#_Toc147755)

[Chapitre I: Revue de la littérature 2](#_Toc147756)

[Section 1. Concepts et définitions 2](#_Toc147757)

[1. Définitions 2](#_Toc147758)

[2. Les sources du bug d’un logiciel 4](#_Toc147759)

[3. Qualification de la gravite des anomalies 5](#_Toc147760)

[Section 2. L’utilité des tests et validation logiciel 6](#_Toc147761)

[1. Exemples de Bugs aux conséquences désastreuse 6](#_Toc147762)

[2. Nécessite de la VVT (Validation, Vérification et Test logiciel) 9](#_Toc147763)

[3. Aperçu de l'automatisation des tests 10](#_Toc147764)

[Section 3. Vue panoramique des solutions de tests logiciel 11](#_Toc147765)

[Chapitre II: Cadrage du projet 13](#_Toc147766)

[Section 1. Contexte du projet et étude de l’existant 13](#_Toc147767)

[1. Contexte du projet 13](#_Toc147768)

[2. Processus de test existant 13](#_Toc147769)

[3. Constat 17](#_Toc147770)

[Section 2. Cahier de charge 18](#_Toc147771)

[1. Justification de l’étude 18](#_Toc147772)

[2. Spécifications Du Projet 19](#_Toc147773)

[3. Planification 22](#_Toc147774)

[Section 3. Méthodologie et outils envisagés 24](#_Toc147775)

[1. Méthodologie de travail : scrum 24](#_Toc147776)

[2. Langage de modélisation 27](#_Toc147777)

[3. Etude critique des outils les plus utilises 29](#_Toc147778)

[3. 1. Selenium 29](#_Toc147779)

[3. 2. Watir 31](#_Toc147780)

[3. 3. Protractor 33](#_Toc147781)

[3. 4. Robot Framework 35](#_Toc147782)

[3. 5. Katalon Studio 37](#_Toc147783)

[3. 6. Unified Functional Testing (UFT) 39](#_Toc147784)

[3. 7. TestComplete 40](#_Toc147785)

[3. 8. Ranorex 42](#_Toc147786)

[3. 9. Telerik TestStudio 44](#_Toc147787)

[Partie II: Analyse, Conception et implémentation 47](#_Toc147788)

[Chapitre III: Analyse et conception 48](#_Toc147789)

[Section 1. Analyse approfondie 48](#_Toc147790)

[1. Classification selon la nature de l’objet 48](#_Toc147791)

[2. Classification selon l'accessibilité de la structure de l'objet 52](#_Toc147792)

[3. Classification selon la propriété de l'objet 54](#_Toc147793)

[Section 2. Modélisation conceptuelle 59](#_Toc147794)

[1. Démarche qualité 59](#_Toc147795)

[2. La roue de Deming 60](#_Toc147796)

[3. Généralités sur le langage uml 63](#_Toc147797)

[Section 3. Architecture générale de l’application 68](#_Toc147798)

[1. Composition de Selenium 68](#_Toc147799)

[2. Details de l’architecture 70](#_Toc147800)

[3. Exemple 71](#_Toc147801)

[Chapitre IV: Résultats obtenus et discussion 73](#_Toc147802)

[Section 1: Outils et méthodes d’implémentation 73](#_Toc147803)

[1. Selenium IDE 73](#_Toc147804)

[2. Langages utilisés 73](#_Toc147805)

[3. Approche de développement 75](#_Toc147806)

[Section 2: Présentation de la solution réalisée 76](#_Toc147807)

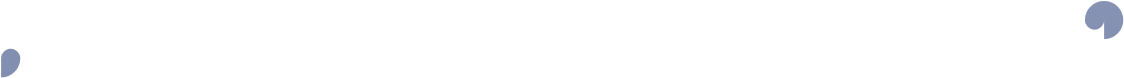
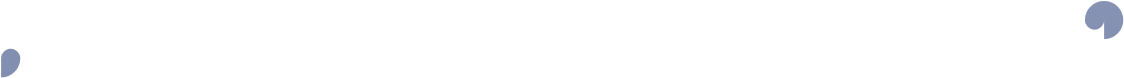
[Section 3: Evaluation des résultats et impact sur l’entreprise 81](#_Toc147808)

Conclusion générale ............................................................................................................. 84

Annexes ............................................................................................................................... 85

Tables de matières ................................................................................................................ 89

Références bibliographiques ................................................................................................. 92



Référ

ences bibliograph

i

ques

##### Livres et ouvrages

* [1] Ioannis Parissis (2009), *Méthodes et outils pour le test logiciel*, Génie logiciel, Université Joseph – François ouvrier - Grenoble I, 75 p.
* [2] Florence Gillet Goinard, Bernard Seno (2016), *La boîte à outils du responsable qualité*,Dunod 3ème édition, 192p.
* [3] Lavallée, François (2020), *Coût de la qualité : À la découverte des coûts insoupçonnés de vos processus de gestion de la qualité* (French Edition), Aliter Concept, 256 p.
* [4] Pascal Roques (2018), *Uml 2.5 par la pratique*, 8eme édition : Eyrolles, 408p.
* [5] Alistair Cockburn, (2001). *Rédiger des cas d'utilisation efficaces*, Technologies objet : Eyrolles, 320p.
* [6] Fabrice Ambert, Fabrice Bouquet (2016), *Introduction au test logiciel*. Science et technologie, 163p.

##### Documents consultés

* ELOMA Albert, Enseignant à l’IUC 2019-2020, Tests et validation logiciel, non Publié.
* Marc Hage Chahine (2018). *L’ECRITURE DES CAS DE TEST*, La taverne du testeur.
* Standard glossary of terms used in Software Testing (2007), ISTQB, version 2.0, December.
* Glossaire *CFTL/ISTQB des termes utilisés en tests* de logiciels.
* Comité Français de Test Logiciel, CFTL
* TCHUMMOGNI Emmanuel Perry, Enseignant à l’IUC 2019-2020, Conduite de

Projet, non publié.

* NDOUMA, Enseignant à l’IUC 2019-2020, Entrepreneuriat et création d’entreprise, Non publié.
* AZEBAZE Anatole Guy Blaise, Enseignant à l’IUC 2019-2020, Initiation à la vie Professionnelle.