

FACULTÉ DES SCIENCES ET TECHNIQUE
MOHAMMEDIA

RAPPORT DE STAGE

CONCEPTION ET DÉVELOPPEMENT DE
TESTS AUTOMATIQUES (NON
RÉGRESSION, SÉCURITÉ ET
PERFORMANCE) ET SON INTÉGRATION
DANS LA CHAÎNE CI/CD DE GitLab

FI : Génie Mathématiques et informatique Année universitaire : 2019/2020

Réalisé par :

BENAMI Anas [élève ingénieur]

Encadré par :

M. KHACHAFI Elmahdi

M. Khadir Omar

Dédicace

Au nom du dieu, celui qui fait miséricorde le
miséricordieux

Je dédie ce travail à ma famille qui ne cesse pas de m'encourager et me donner le soutien moral et matériel, à mes amis et à tous ceux qui ont croisé mon chemin et laisser de belles traces sur ma vie.

Remerciement

D'un esprit vivement reconnaissant, je tiens à remercier chaleureusement, mon encadrant M. **KHACHAFI Elmahdi**, pour le temps qu'il a consacré à m'apporter les outils méthodologiques indispensables à la conduite de ce modeste travail . Son exigence m'a grandement stimulé. mes vifs remerciements vont également à M. **Khadir Omar** pour sa patience, ses conseils pleins de sens et pour le suivi et l'intérêt qu'il a porté à mon travail. L'ensemble des collègues de classe pour leurs intérêts et leurs encouragements.

Je remercie également les membres du jury pour l'intérêt qu'ils ont porté à mon projet en acceptant d'examiner mon travail et de l'enrichir par leurs propositions. Aussi mes remerciements au corps professoral et administratif de la faculté des sciences et techniques qui déploient de grands efforts pour nous assurer une très bonne formation.

Table des matières

| | | |
|----------|---|-----------|
| 1 | Cadre Général | 9 |
| 1.1 | Introduction : | 9 |
| 1.2 | Cadre du travail : | 9 |
| 1.3 | Présentation de l'organisme d'accueil | 9 |
| 1.4 | Étude préalable | 11 |
| 1.4.1 | Présentation de la problématique et proposition de solution | 11 |
| 1.4.2 | Objectif de l'étude | 12 |
| 1.5 | Plan de travail | 12 |
| 1.5.1 | Organisation du rapport | 12 |
| 1.5.2 | Diagramme de Gantt | 13 |
| 1.6 | Conclusion | 13 |
| 2 | Conception et développement des tests | 14 |
| 2.1 | Introduction | 14 |
| 2.2 | Les tests de non régression | 14 |
| 2.2.1 | Définition : | 14 |
| 2.2.2 | Les technologies : | 15 |

| | | |
|----------|---|-----------|
| 2.2.2.1 | Cucumber | 15 |
| 2.2.2.2 | Selenium | 15 |
| 2.2.2.3 | Langage de programmation | 15 |
| 2.2.2.4 | Environnement de développement | 15 |
| 2.3 | Conception | 16 |
| 2.3.1 | Méthodologie de conception | 16 |
| 2.3.2 | Architecture | 16 |
| 2.3.3 | Génération des rapports : | 18 |
| 2.4 | Les tests de sécurité | 20 |
| 2.4.1 | Les outils du test : | 20 |
| 2.4.1.1 | Utilisation : | 20 |
| 2.4.1.2 | Nikto : | 20 |
| 2.4.1.3 | Wapiti : | 21 |
| 2.4.2 | Application : | 21 |
| 2.4.2.1 | Premier cas : un site vulnérable | 21 |
| 2.4.2.2 | Deuxième cas : La plateforme Rokhas | 23 |
| 2.5 | Les tests de performance | 24 |
| 2.6 | Conclusion | 24 |
| 3 | L'intégration dans la chaine CI/CD de GitLab | 25 |
| 3.1 | Introduction | 25 |
| 3.2 | Technologies | 25 |
| 3.3 | Définitions | 26 |
| 3.3.1 | L'intégration continue [CI] : | 26 |
| 3.3.2 | La livraison "Delivery"/Déploiement continue [CD] : | 27 |

| | | |
|---------|---|----|
| 3.3.2.1 | La différence entre déploiement continu et livraison continue . . | 27 |
| 3.4 | Principe de fonctionnement | 28 |
| 3.4.1 | Le fichier de configuration : | 28 |

Table des figures

| | | |
|-----|---|----|
| 2.1 | Diagramme de Classe [Architecture] | 16 |
| 2.2 | Scenario Utilisateur entre une date de naissance invalide [Plateforme Rokhas] | 17 |
| 2.3 | Rapport du scénario [Attachements des documents PH] | 18 |
| 2.4 | Rapport des scénarios de [Localisation du projet] | 19 |
| 2.5 | Rapport d'un scénario qui a échouer [Capture d'écran] | 19 |
| 2.6 | Output de la commande de test de vulnérabilités [Wapiti] | 22 |
| 2.7 | Rapport généré par Wapiti | 22 |
| 2.8 | Rapport généré par Nikto | 23 |
| 2.9 | Rapports Plateforme Rokhas [0 vulnérabilités] | 23 |
| 3.1 | Intégration/Livraison/Déploiement Continue | 27 |
| 3.2 | Exemple d'une pipline | 29 |

Introduction Générale

De nos jours le développement des applications et logiciels sont en constante évolution. Les entreprises pensent alors à gagner en temps et en performances afin d'automatiser la procédure de tests logiciels , car la phase de test est une étape incontournable de tout développement informatique,elle a pour objectif de vérifier que le livrable répond bien aux besoins exprimés par l'utilisateur.

Un test est un ensemble de cas à tester pour effectuer une vérification partielle d'un système. Il vise à trouver des bugs. Le but principal est d'identifier un nombre maximum d'anomalies du logiciel dans une courte durée avant livraison des produits aux clients. La qualité sera donc augmentée lorsque les problèmes seront corrigés.

Les tests représentent 30 à 50% du coût de développement, et pour les faire chaque fois d'une manière manuelle c'est lourd premièrement ,et deuxièmement il prend beaucoup du temps et risquer car il se fait par un être humain,ceci est un grand problème. 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.C'est à dire gain en temps et en productivité.

Elle permet aux testeurs de se concentrer sur les tests des nouvelles fonctionnalités de l'application avec une meilleure fiabilité ainsi que de minimiser les risques d'erreurs.

L'automatisation des tests logiciels présente de nombreux avantages :

- Une vitesse incroyable : Les tests automatisés font des merveilles en vérifiant chaque millimètre. Dans de nombreux cas, il faudrait une éternité pour vérifier les mêmes choses manuellement (si ce n'est carrément impossible).

- Réutilisables : Une fois écrit par un ingénieur QA (Quality Assurance), les tests peuvent être utilisés encore et encore, à l'infini. Les mêmes modules peuvent être réutilisés pour d'autres tests sur le projet.
- Excellente couverture : Grâce à l'automatisation, On peut couvrir un grand nombre de variantes de cas de test. Cela inclut l'interaction avec plusieurs systèmes d'exploitation, navigateurs etc., ainsi que divers scénarios de comportement d'utilisateur, et bien plus encore.
- Rapports pratiques : La génération des journaux de tests prêts qui listent précisément tous les tests effectués et les bugs trouvés.
- Autosuffisance : Les tests automatisés peuvent être exécutés 24 heures sur 24, 7 jours sur 7, sans surveillance, puis vous présenter tous les résultats du test.

Cadre Général

1.1 Introduction :

Ce chapitre est consacré pour la présentation de l'organisme d'accueil, la précision du cadre du projet et la problématique puis l'énumération des étapes de travail à réaliser pour achever ce projet.

1.2 Cadre du travail :

Ce stage s'inscrit dans le cadre d'un projet de fin d'études pour l'obtention du diplôme d'ingénieur mathématiques informatique de la Faculté des Sciences et Techniques Mohammedia. Mon stage a été effectué au sein d'une Société de conseil opérationnel en systèmes d'information (Ribatis). Le sujet est intitulé Conception et développement de tests automatiques (non régression, sécurité et performance) et son intégration dans la chaîne CI/CD de GitLab.

1.3 Présentation de l'organisme d'accueil

RIBATIS est une entreprise de conseil opérationnel en systèmes d'information. Fondée en 2007, elle dispose d'un positionnement innovant alliant le recul des cabinets de conseil et le pragmatisme des sociétés d'ingénierie informatique. RIBATIS propose trois familles de services :

1.3. PRÉSENTATION DE L'ORGANISME ~~CA~~ **Cadre Général**

Conseil, Technologie et Formation.

RIBATIS a réussi le déploiement de plusieurs solutions technologiques à forte valeur ajoutée sur le marché, notamment E-DMAJ, un ERP dédié à la gestion des activités de l'entreprise au quotidien et ATHLETIS un système d'information spécialisé dans la gestion des fédérations, ligues et clubs sportifs.

Actuellement l'entreprise, basée à Casablanca, emploie une équipe d'une vingtaine d'ingénieurs et consultants fonctionnels haut niveau.

☐ Métier :

La mission au quotidien est de veiller au bonheur de l'utilisateur et du fonctionnaire en leur offrant une expérience digitale exceptionnelle. A travers les plateformes digitales qu'ils réalisent, ils visent une transformation radicalement positive de la relation citoyen/administration, en la convertissant en une relation de collaboration plutôt qu'une relation de contrainte.

☐ Approche :

l'approche est simple : Proposer les Services e-Gov sous forme d'une succession d'étapes faisant intervenir tour à tour le fonctionnaire et l'utilisateur.

Ainsi, au lieu d'avoir deux systèmes distincts : Le 1er servant à soumettre la demande usager (Interface externe) et le 2ème permettant à l'administration de la traiter (Interface interne), nos plateformes les fusionnent en un seul et même e-service. Ceci permet au fonctionnaire et à l'utilisateur de collaborer autour d'un même processus avec comme objectif commun de mener à bien la procédure, tout en respectant la réglementation en vigueur, les limites de responsabilité et la traçabilité des actions.

☐ Les plateformes :

— **CASAURBA** : Lancé en octobre 2014 et généralisée à la région Casablanca-Settat à fin 2015, Casaurba est une plateforme digitale collaborative assurant la gestion 100% dématérialisée des autorisations d'urbanisme de +36 communes

— **ROKHAS** : Lancée en Avril 2017, la plateforme Rokhas dématérialise la délivrance

des autorisations économiques, en assurant l'équilibre entre développement économique et respect de l'environnement et de la vie en société.

- **CasaOpenData** : Portail de données statistiques ouvertes, alimenté à partir des plateformes digitales Casaurba et Rokhas et publiant des indicateurs précis, actualisés au quotidien et conçus entant qu'outil d'aide à la décision et d'amélioration continue.
- **KAFC/Jibayat** : La Plateforme Karaz Administration Fiscale Communale (KAFC) est une plateforme de gestion globale et intégrée de l'ensemble des taxes, droits et redevances collectées au niveau des communes.
- **GO HUB** : La plateforme Geohub.ma permet aux institutions, aux entreprises et aux individus de créer facilement et gratuitement leurs propres géoportails. Ils peuvent ainsi, créer, publier, actualiser et valoriser leurs données géographiques.
- **PARAPHEURB** : Parapheur.ma est une plateforme cloud dédiée à la transformation digitale des circuits de signature au sein des administrations publiques et privées.

1.4 Étude préalable

1.4.1 Présentation de la problématique et proposition de solution

Souvent on commence par la conception, le développement, le test puis le déploiement d'une application sur un serveur, après un certain temps, on se rend compte qu'il faut ajouter de nouvelles fonctionnalités à notre application. En général ces nouvelles fonctionnalités ont une forte relation avec les fonctionnalités existantes, et par conséquent il y a une forte probabilité que ces dernières soient affectées de manière inattendue.

La solution est de tester l'intégralité de l'application après chaque modification du code,

Pour le faire de manière manuelle, c'est très difficile et prendra beaucoup de temps et risque d'oublier de ne pas tester quelque chose, donc la meilleure solution est l'automatisation des tests.

1.4.2 Objectif de l'étude

- Réduire le délai de livraison des produits (déploiement des applications dans les serveurs de production).
- Réduire la charge de travail et le temps passé à exécuter des tests fonctionnelles.
- Couvrir un grand nombre de variantes de cas de test.
- Repérer rapidement un bug pendant les cycles de développement.
- Éviter au maximum les erreurs humaine.

1.5 Plan de travail

1.5.1 Organisation du rapport

Pour un bon travail il faut un rapport bien structuré qui peut être exploité après la mise en place de ces tests, pour cela le présent rapport vas être organiser notre de la manière suivante :

Dans le premier chapitre, on a met notre projet dans son cadre général en définissant la société d'accueil et en présentant le sujet avec une étude préalable.

Dans le deuxième chapitre intitulé Conception et développement des tests , nous allons présenter en premier lieu les tests de (non régression, sécurité et performance) et en deuxième lieu les outils et le principe de fonctionnement chaque test .

Enfin et au niveau du troisième et dernier chapitre intitulé L'intégration dans la chaine CI/CD de GitLab ,on vas présenter les différentes composantes de cette chaine et son rôle dans

l'automatisation.

1.5.2 Diagramme de Gantt

Le diagramme de Gantt est un outil de planification des tâches nécessaires pour la réalisation d'un projet quelque soit le secteur d'activité. Il permet de visualiser l'avancement des tâches d'un projet de manière simple et concise, de planifier et suivre les besoins en ressources humaines et matérielles et donc de pouvoir suivre l'avancement du projet.

Le diagramme suivant va représenter les taches principales à réaliser dans le projet.

Pas Encore

1.6 Conclusion

Dans ce chapitre, nous avons présenté l'organisme d'accueil encerclant le cadre d'étude, la problématique ainsi que les objectifs. Nous verrons comment implémenter ces différents types de test dans le chapitre suivant.

Conception et développement des tests

2.1 Introduction

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

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

En informatique aussi, *un battement d'ailes d'un papillon peut provoquer une tornade* **La théorie de chaos**, il faut donc tester afin de vérifier que les modifications n'ont pas apporté d'instabilités ou d'anomalies, aux conséquences parfois imprévisibles.

2.2 Les tests de non régression

2.2.1 Définition :

Le **test de non régression** est défini comme un type de test de logiciel pour s'assurer que les nouvelles modifications de code ne devraient pas avoir d'effets secondaires sur les fonc-

2.2. LES TESTS DE NON RÉGRESSION

2.2.1. Conception et développement des tests

tionnalités existantes. Il garantit que l'ancien code fonctionne toujours une fois les dernières modifications de code effectuées.

2.2.2 Les technologies :

2.2.2.1 Cucumber



est un outil qui prend en charge le développement basé sur le comportement (BDD). Il offre un moyen d'écrire des tests que tout le monde peut comprendre, quelles que soient leurs connaissances techniques.

2.2.2.2 Selenium



est un framework de test informatique développé en Java. Il permet d'interagir avec différents navigateurs web de même que le ferait un utilisateur de l'application. Il entre ainsi dans la catégorie des outils de test dynamique facilitant le test fonctionnel .

2.2.2.3 Langage de programmation



est un langage de programmation orienté objet créé par James Gosling et Patrick Naughton, employés de Sun Microsystems.

La société Sun a été ensuite rachetée en 2009 par la société Oracle.

2.2.2.4 Environnement de développement



Est un environnement de développement intégré de technologie Java destiné au développement de logiciels informatiques. Il est développé par JetBrains.

2.3 Conception

2.3.1 Méthodologie de conception

Pour faciliter la tâche on vas utiliser le langage de modélisation unifié (UML : Unified Modelling Language) c'est une notation qui permet de modéliser un problème de façon standard. Ce langage est né de la fusion de plusieurs méthodes existantes auparavant, et il est devenu un référence en terme de modélisation objet, à un tel point que sa connaissance devienne indispensable pour un développeur.

2.3.2 Architecture

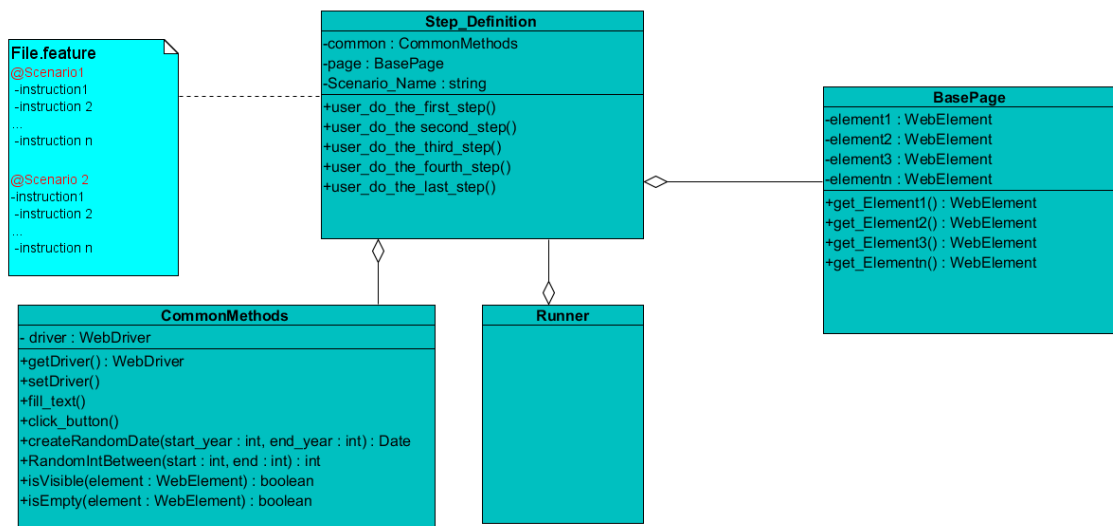


FIGURE 2.1 – Diagramme de Classe [Architecture]

D'abord on commence par la création d'un fichier avec l'extension **.feature** de Cucumber, dans lequel on écrit nos scénarios avec n'importe quelle langue (L'anglais, Le francais...), Chaque scénario est une liste d'étapes à suivre pour Cucumber. Pour que Cucumber comprenne les scénarios, ils doivent suivre certaines règles de syntaxe de base, appelées Gherkin.

2.3. CONCEPTION Chapitre 2. Conception et développement des tests

Exemple :

MAÎTRE D'OUVRAGE :

☒ PERSONNE PHYSIQUE
 ☐ PERSONNE MORALE
 ☐ ADMINISTRATION PUBLIQUE

Maître d'ouvrage existant : Qualité du maître d'ouvrage :

En choisissant un client de votre portfolio, ses données seront automatiquement affichées. La qualité du maître d'ouvrage choisie influence la liste des documents exigés.

| | | |
|---------------------------------|------------------------------------|----------------------------------|
| Civilité * | Nom * | Prénom * |
| Nom du père * | Nom de la mère * | |
| CIN/Passeport/Titre de séjour * | Date fin validité pièce identité * | Lieu délivrance pièce identité * |
| Nationalité Maroc | Date de naissance * | Lieu de naissance * |
| Profession | Téléphone * | Email |
| Adresse * | Ville * | Pays * Maroc |

```

@InvalidDateOfBirth
Scenario: Invalid date of birth
  And User enter his Quality
  And User Enter his Civility
  And User enter his last_name "Mahjoubi"
  And User enter his first_name "Fahd"
  And User enter his father's name "Mohammed"
  And User enter his mother's name "Fatima"
  And User enter his CIN "0890123"
  And User enter his expiration date of CIN
  And User enter his city of delivery of CIN "Casablanca"
  And User enterhis date of birth day
  And User entre his place of birth "Casablanca"
  And User enter his phone number "0666871394"
  And User enter his email "fahd78@gmail.com"
  And User enter his address "hay el ouahda casablanca 1987"
  And User enter his city's name "Casablanca"
  And User enter his country
  Then User Clicks the submit button!
  
```

FIGURE 2.2 – Scenario Utilisateur entre une date de naissance invalide [Plateforme Rokhas]

La deuxième chose, on crée la classe **BasePage** (C'est une classe Java qui contient comme attributs les éléments web [Bouton,inputText,checkBox... etc] de la page web qu'on veut tester)

2.3. CONCEPTION

Chapitre 2. Conception et développement des tests

on récupèrent les éléments web à l'aide du framework Selenium.

Nb : Dans cet exemple les éléments sont les attributs du formulaire.

Après on crée La classe **Step Definition** qui est le coeur de notre travail. Dans cette classe on crée, pour chaque étape (instruction) du scénario, une méthode qui traduit les mots en code java en utilisant le framework Selenium .

En fin on crée la classe **Runner** qui est responsable de lancement des scénarios a exécuter, En différencier entre les scénarios par l'annotation **@ Entête** de chaque scénario, dans notre cas le scénario s'appel **InvalidDateOfBirth**.

Nb : On peut lancé le nombre de scénarios qu'on veut.

La classe **Commond Methodes** juste pour garder la notion du clean code.

2.3.3 Génération des rapports :

Cucumber nous aide beaucoup, car il nous donne la possibilité de générer des rapport avec différentes formats (HTML,json,xml,...) après chaque exécution du test.

Exemple : Un seul scénario

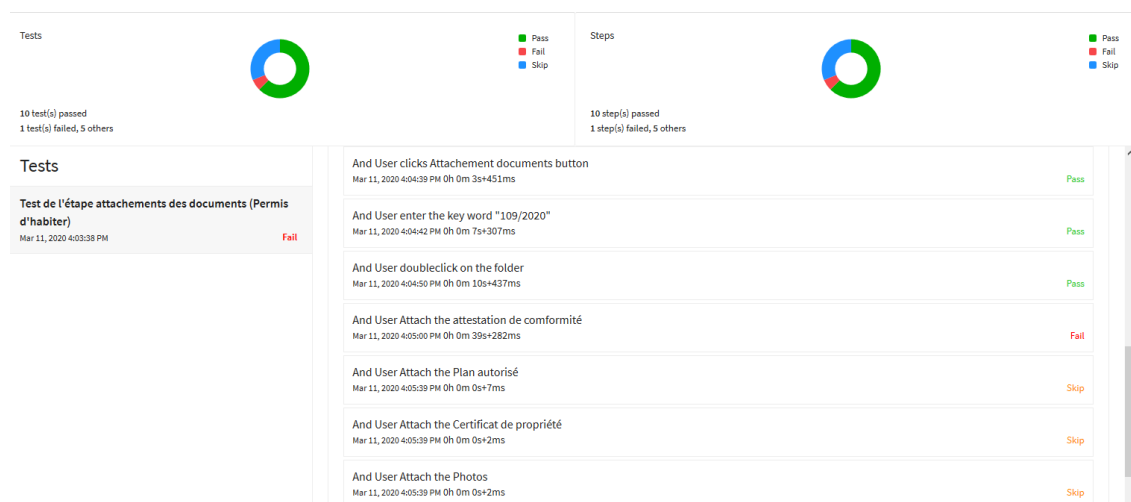


FIGURE 2.3 – Rapport du scénario [Attachements des documents PH]

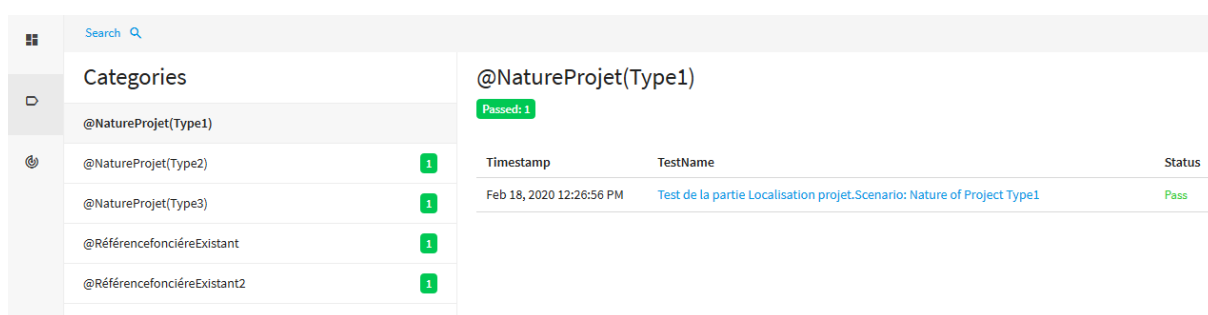
2.3. CONCEPTION Chapitre 2. Conception et développement des tests

Comme vous voyez, on a les graphiques qui nous indique par pourcentage, le taux des étapes réussies [en vert], échouées [en rouge] et les étapes ignorées [en bleu].

Il nous liste aussi tous les étapes du scénario et indique devant chaque étape [Pass, Fail ou bien Skip].

NB : une fois Cucumber rencontre une étape échouée, il ignore toutes les étapes qui la suivent.

Exemple : Plusieurs scénarios



| Categories | @NatureProjet(Type1) |
|-----------------------------|----------------------|
| @NatureProjet(Type1) | Passed: 1 |
| @NatureProjet(Type2) | 1 |
| @NatureProjet(Type3) | 1 |
| @RéférencefoncièreExistant | 1 |
| @RéférencefoncièreExistant2 | 1 |

| Timestamp | TestName | Status |
|--------------------------|---|--------|
| Feb 18, 2020 12:26:56 PM | Test de la partie Localisation projet.Scenario: Nature of Project Type1 | Pass |

FIGURE 2.4 – Rapport des scénarios de [Localisation du projet]

En cas d'échec d'un test il fait une capture d'écran pour qu'on puisse savoir exactement qu'est ce qui est passé.

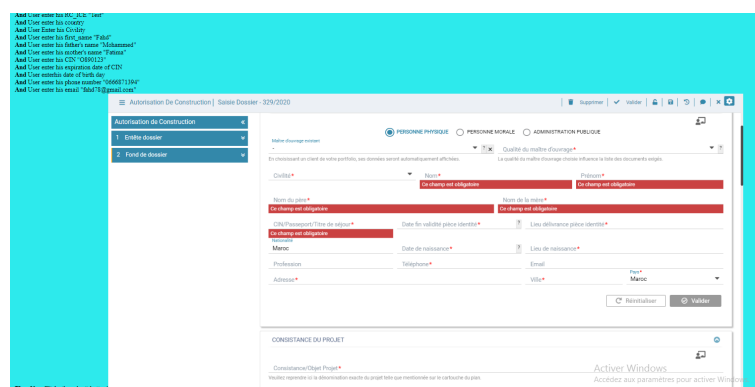


FIGURE 2.5 – Rapport d'un scénario qui a échouer [Capture d'écran]

2.4 Les tests de sécurité

Il est clair que les conséquences des violations de la sécurité sont dévastatrices, aujourd'hui, les tests de sécurité jouent un rôle vital dans toutes les applications Web, car les pirates continuent d'inventer chaque jour de nouvelles techniques qui peuvent être pour l'argent, la reconnaissance et même pour le plaisir. Les personnes ayant moins de compétences en piratage peuvent détruire une application Web si l'application est mal sécurisée.

L'objectif principal des tests de sécurité est de déterminer les vulnérabilités d'un système et de déterminer si ses données et ses ressources sont protégées contre les intrus potentiels.

2.4.1 Les outils du test :

2.4.1.1 Utilisation :

Les scanners de vulnérabilités peuvent être utilisés dans des objectifs licites ou illicites :

- **Objectifs licites :** les experts en sécurité informatique ou les entreprises utilisent les scanners de vulnérabilités pour trouver les failles de sécurité des systèmes informatiques et des systèmes de communications de leurs entreprises dans le but de les corriger avant que les pirates informatiques ne les exploitent.
- **Objectifs illicites :** les pirates informatiques utilisent les mêmes outils pour trouver les failles dans les systèmes des entreprises pour les exploiter à leur avantage.

2.4.1.2 Nikto :



est un scanner de vulnérabilité de ligne de commande de logiciel gratuit qui analyse les serveurs web pour les fichiers / CGI dangereux. Il effectue des vérifications génériques et spécifiques au type de serveur. Il capture et imprime également tous les cookies reçus.

2.4. LES TESTS DE SÉCURITÉ

Chapitre 2.4.1 Conception et développement des tests

Le code Nikto lui-même est un logiciel gratuit, mais les fichiers de données qu'il utilise pour piloter le programme ne le sont pas.

2.4.1.3 Wapiti :



est un outil open source qui analyse les applications Web pour détecter de multiples vulnérabilités, notamment les injections de base de données, cross-site scripting, XXE injection, des injections SQL, XPath, PHP, ASP et JSP.

Il effectue également des tâches de test de pénétration supplémentaires, telles que la recherche de fichiers potentiellement dangereux sur les serveurs, Les résultats collectés sont automatiquement stockés dans un fichier HTML.

2.4.2 Application :

2.4.2.1 Premier cas : un site vulnérable

On a cherché sur Internet sur des sites vulnérables pour s'assurer que nos outils peuvent détecter des vulnérabilités s'elles existent.

http ://www.dvwa.co.uk/ Ses principaux objectifs sont d'aider les professionnels de la sécurité à tester leurs compétences et leurs outils dans un environnement juridique, à aider les développeurs Web à mieux comprendre les processus de sécurisation des applications Web et à aider les enseignants / étudiants à enseigner / apprendre la sécurité des applications Web dans un environnement de salle de classe.

• Output de wapiti :

Parmi les avantages de Wapiti et Nikto qu'ils sont facile à utiliser, il suffit d'écrire quelques commandes pour lancer un scan.

Chapitre 2 Conception et développement des tests

```

17 # Starting Wapiti Scan...
18 $ wapiti -u http://www.dvwa.co.uk/ -v 2 --color
19
20      _   _   _   _   _   _   _   _   _   _   _   _   _   _   _   _
21     / \ / \ / \ / \ / \ / \ / \ / \ / \ / \ / \ / \ / \ / \ / \
22    /___/_/___/_/___/_/___/_/___/_/___/_/___/_/___/_/___/_/___/_/
23
24 Wapiti 3.0.1 (wapiti.sourceforge.net)
25 [*] You are lucky! Full moon tonight.
26 [*] GET http://www.dvwa.co.uk/ (0)
27 [*] GET http://www.dvwa.co.uk/js/jquery.nivo.slider.pack.js (1)
28 [*] GET http://www.dvwa.co.uk/index.php (1)
29 [*] GET http://www.dvwa.co.uk/js/jquery-1.6.2.min.js (1)
30 [*] GET http://www.dvwa.co.uk/google-analytics.com/ga.js (1)
31 [*] GET http://www.dvwa.co.uk/favicon.ico (1)
32 [*] GET http://www.dvwa.co.uk/css/all.css (1)
33 [*] GET http://www.dvwa.co.uk/nivo-slider/themes/default/default.css (1)
34 [*] GET http://www.dvwa.co.uk/css/nivo-slider.css (1)
35 [*] Saving scan state, please wait...
36 Note
37 =====
38 This scan has been saved in the file /var/lib/gitlab-runner/.wapiti/scans/www.dvwa.co.uk_folder_cdc889b.db
39 [*] Wapiti found 9 URLs and forms during the scan
40 [*] Loading modules:
41         mod_crlf, mod_exec, mod_file, mod_sql, mod_xss, mod_backup, mod_htaccess, mod_blindsqli, mod_permanentxss, mod_nikto,
42         o, mod_delay, mod_buster, mod_shellshock, mod_methods, mod_ssrf
43 [*] Launching module xss...
44 [*] GET http://www.dvwa.co.uk/ (0)
45 [*] GET http://www.dvwa.co.uk/google-analytics.com/ga.js (1)
46 [*] GET http://www.dvwa.co.uk/css/all.css (1)
47 [*] GET http://www.dvwa.co.uk/css/nivo-slider.css (1)
48 [*] GET http://www.dvwa.co.uk/favicon.ico (1)
49 [*] GET http://www.dvwa.co.uk/index.php (1)
50 [*] GET http://www.dvwa.co.uk/js/jquery-1.6.2.min.js (1)
51 [*] GET http://www.dvwa.co.uk/js/jquery.nivo.slider.pack.js (1)
52 [*] GET http://www.dvwa.co.uk/nivo-slider/themes/default/default.css (1)
53 [*] Launching module file

```

FIGURE 2.6 – Output de la commande de test de vulnérabilités [Wapiti]

Après chaque scan Wapiti génère un rapport décrit la catégorie et le nombre du vulnérabilités détecter.

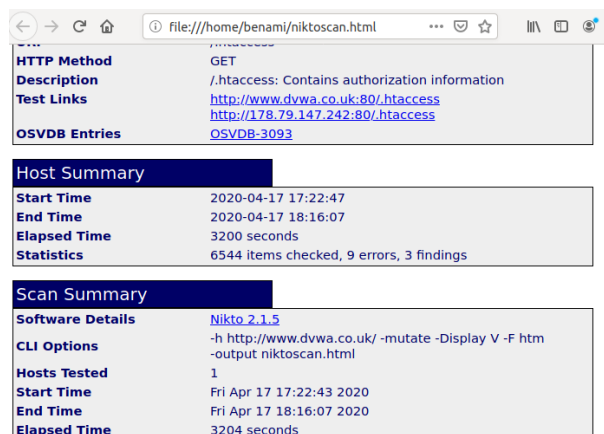
| Category | Number of vulnerabilities found |
|--------------------------------------|---------------------------------|
| Cross Site Scripting | 0 |
| Htaccess Bypass | 0 |
| Backup file | 0 |
| SQL Injection | 0 |
| Blind SQL Injection | 1 |
| File Handling | 0 |
| Potentially dangerous file | 0 |
| CRLF Injection | 0 |
| Commands execution | 0 |
| Resource consumption | 2 |
| Internal Server Error | 0 |

FIGURE 2.7 – Rapport généré par Wapiti

2.4. LES TESTS DE SÉCURITÉ

Chapitre 2.4.1 Conception et développement des tests

La même chose pour Nikto :



| | |
|----------------------|--|
| HTTP Method | GET |
| Description | /.htaccess: Contains authorization information |
| Test Links | http://www.dvwa.co.uk:80/.htaccess http://178.79.147.242:80/.htaccess |
| OSVDB Entries | OSVDB-3093 |

| | |
|---------------------|--|
| Host Summary | |
| Start Time | 2020-04-17 17:22:47 |
| End Time | 2020-04-17 18:16:07 |
| Elapsed Time | 3200 seconds |
| Statistics | 6544 items checked, 9 errors, 3 findings |

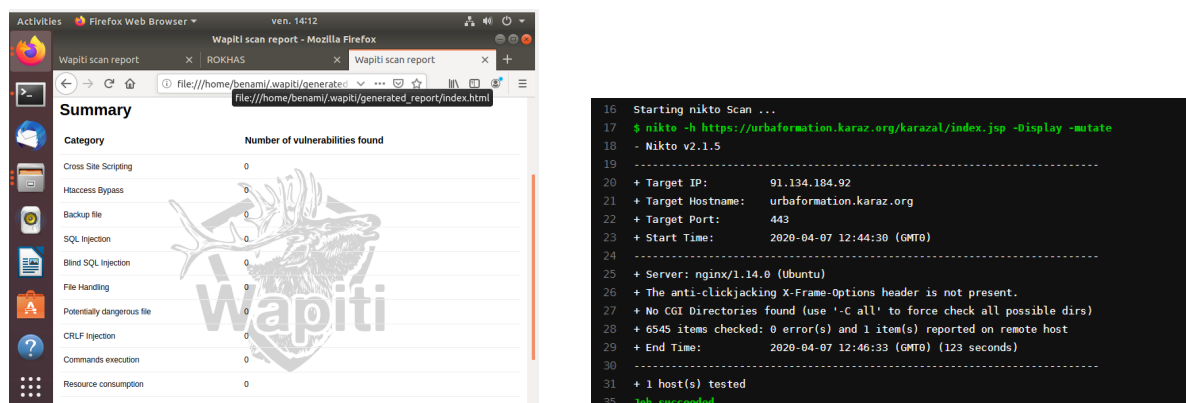
| | |
|-------------------------|---|
| Scan Summary | |
| Software Details | Nikto 2.1.5 |
| CLI Options | -h http://www.dvwa.co.uk/ -mutate -Display V -F htm -output niktoscan.html |
| Hosts Tested | 1 |
| Start Time | Fri Apr 17 17:22:43 2020 |
| End Time | Fri Apr 17 18:16:07 2020 |
| Elapsed Time | 3204 seconds |

FIGURE 2.8 – Rapport généré par Nikto

2.4.2.2 Deuxième cas : La plateforme Rokhas

- Output de wapiti et de Nikto :

On a attaqué la plateforme Rokhas par les deux outils, et les deux confirment que la plateforme est bien sécurisé, pas des failles de sécurité, pas des vulnérabilités :



Wapiti scan report - Mozilla Firefox

Summary

| Category | Number of vulnerabilities found |
|----------------------------|---------------------------------|
| Cross Site Scripting | 0 |
| Httaccess Bypass | 0 |
| Backup file | 0 |
| SQL Injection | 0 |
| Blind SQL Injection | 0 |
| File Handling | 0 |
| Potentially dangerous file | 0 |
| CRLF Injection | 0 |
| Commands execution | 0 |
| Resource consumption | 0 |

```
16 Starting nikto Scan ...
17 $ nikto -h https://urbaformation.karaz.org/karazal/index.jsp -Display -mutate
18 - Nikto v2.1.5
19
20 + Target IP: 91.134.184.92
21 + Target Hostname: urbaformation.karaz.org
22 + Target Port: 443
23 + Start Time: 2020-04-07 12:44:30 (GMT0)
24
25 + Server: nginx/1.14.0 (Ubuntu)
26 + The anti-clickjacking X-Frame-Options header is not present.
27 + No CGI Directories found (use '-C all' to force check all possible dirs)
28 + 6545 items checked: 0 error(s) and 1 item(s) reported on remote host
29 + End Time: 2020-04-07 12:46:33 (GMT0) (123 seconds)
30
31 + 1 host(s) tested
35 Job succeeded
```

FIGURE 2.9 – Rapports Plateforme Rokhas [0 vulnérabilités]

2.5 Les tests de performance

2.6 Conclusion

L'intégration dans la chaîne CI/CD de GitLab

3.1 Introduction

L'approche CI/CD permet d'augmenter la fréquence de distribution des applications grâce à l'introduction de l'automatisation au niveau des étapes de développement des applications. Les principaux concepts liés à l'approche CI/CD sont l'intégration continue, la distribution continue et le déploiement continu. L'approche CI/CD représente une solution aux problèmes posés par l'intégration de nouveaux segments de code pour les équipes de développement et d'exploitation.

3.2 Technologies



Git est un logiciel de gestion de versions décentralisé. C'est un logiciel libre créé par Linus Torvalds, auteur du noyau Linux, et distribué selon les termes de la licence publique générale GNU version 2. En 2016, il s'agit du logiciel de gestion de versions le plus populaire qui est utilisé par plus de douze millions de personnes.



GitLab est un logiciel libre de forge basé sur git proposant les fonctionnalités de wiki, un système de suivi des bugs, l'intégration continue et la livraison continue. Développé par GitLab Inc et créé par Dmitriy Zaporozhets et par Valery Sizov, le logiciel est utilisé par plusieurs grandes entreprises informatiques incluant IBM, Sony, la NASA, Alibaba, Oracle, Leibniz Rechenzentrum, Boeing, Autodata et SpaceX...

3.3 Définitions

3.3.1 L'intégration continue [CI] :

L'intégration continue nous fournit une bonne solution lorsque l'entreprise travaille sur un vaste projet ou un client souhaite avoir un logiciel à la fois complet et complexe. Différentes équipes travaillent à la conception de pans de l'application et les développeurs se chargent de programmer les différentes fonctionnalités. Après un travail de plusieurs mois voire de plusieurs années, l'intégralité du travail doit être regroupée et c'est alors que les problèmes surviennent. Dans un tel cas, la détection et la correction des erreurs, le regroupement de tous les fragments de code peut prendre plusieurs mois pour finalement se rapprocher de la phase de test finale et du déploiement.

Dans le cadre de la continuous integration, l'intégration du nouveau code est effectuée de façon bien plus précoce et pas uniquement lorsque toutes les parties prenantes ont terminé leur sous-domaine. Au lieu de cela, les développeurs intègrent leur code terminé une ou plusieurs fois par jour dans la mainline, le code source qui est accessible par tous les programmeurs. tant donné qu'il s'agit toujours dans ce cas de sections de code relativement courtes, l'intégration est elle aussi plutôt rapide. Seules quelques minutes sont nécessaires à un développeur pour mettre le résultat de son travail à disposition du reste de l'équipe. Si l'on découvre alors une erreur, elle peut être immédiatement localisée et, dans le meilleur des cas, corrigée rapidement.

3.3.2 La livraison "Delivery"/Déploiement continue [CD] :

La **livraison continue** consiste à automatiser toutes les phases de développement, depuis la modification des lignes de code, en incluant la compilation, l'exécution des tests unitaires, des tests d'intégration et des tests fonctionnels, jusqu'à la livraison d'un package que l'équipe de Production peut déployer à la demande. En déploiement continu, on y ajoute une dernière phase automatisée : le déploiement du package en production sans intervention humaine.

Le **déploiement continu (CD)** est donc la suite de l'intégration continue. Une fois que les tests sont validés sur l'environnement de dev, il faut mettre en production. Le déploiement continu consiste donc à automatiser les actions de déploiements qui étaient auparavant réalisées manuellement. C'est pour cette raison que l'on parle souvent de CI/CD ensemble, l'un va difficilement sans l'autre.

3.3.2.1 La différence entre déploiement continu et livraison continue

Le déploiement continu c'est un idéal que peu d'entreprises ont réellement mis en place. La plupart du temps les équipes IT préfèrent avoir la main sur la dernière étape du déploiement. Dans ce cas on parle donc de livraison continue, toutes les étapes du déploiement sont automatisées sauf la dernière : la mise en production.

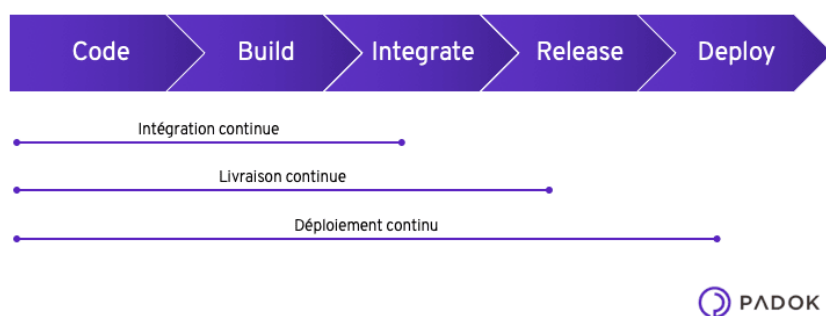


FIGURE 3.1 – Intégration/Livraison/Déploiement Continue

Le déploiement continu inclut donc la livraison continue. Le déploiement continu va plus

loin que la livraison continue en orchestrant automatiquement le déploiement des nouvelles fonctionnalités.

3.4 Principe de fonctionnement

3.4.1 Le fichier de configuration :

En substance, le système GitLab CI / CD comprend trois ingrédients principaux :

- Runners
- Pipelines
- Stages [Étapes]
- Jobs [Tâches]

Expliquons chacun d'eux, du bas de la liste :

- **Jobs** : La tâche est la plus petite unité à exécuter dans GitLab CI / CD. Il est souvent appelé étape de construction. Il peut s'agir d'une tâche de génération ou de compilation ; il peut exécuter des tests unitaires ...
Une seule tâche peut contenir plusieurs commandes (scripts) à exécuter.
- **Stages** : Chaque tâche appartient à un seul étape. Une étape peut contenir zéro, un ou plusieurs tâches à exécuter. Tous les tâches d'une étape s'exécutent en parallèle.
L'étape suivante n'est exécutée que si tous les travaux de l'étape précédente se terminent avec succès - ou s'ils sont marqués comme autorisés à échouer.
- **Pipelines** : Un pipeline est un parapluie pour les tâches et les étapes. Pipeline les orchestre et les rassemble. Le pipeline s'exécute lorsque vous envoyez une nouvelle validation ou une nouvelle balise, exécutant tous les travaux à leurs étapes dans le bon ordre. La configuration entière du pipeline est stockée dans le fichier de configuration **.gitlab-ci.yml**

3.4. PRINCIPES DE L'INTÉGRATION DANS LA CHAÎNE CI/CD DE GitLab

- **Runners** : GitLab Runner est le projet open source utilisé pour exécuter les tâches et renvoyer les résultats à GitLab. Il est utilisé conjointement avec GitLab CI / CD .

Exemple :

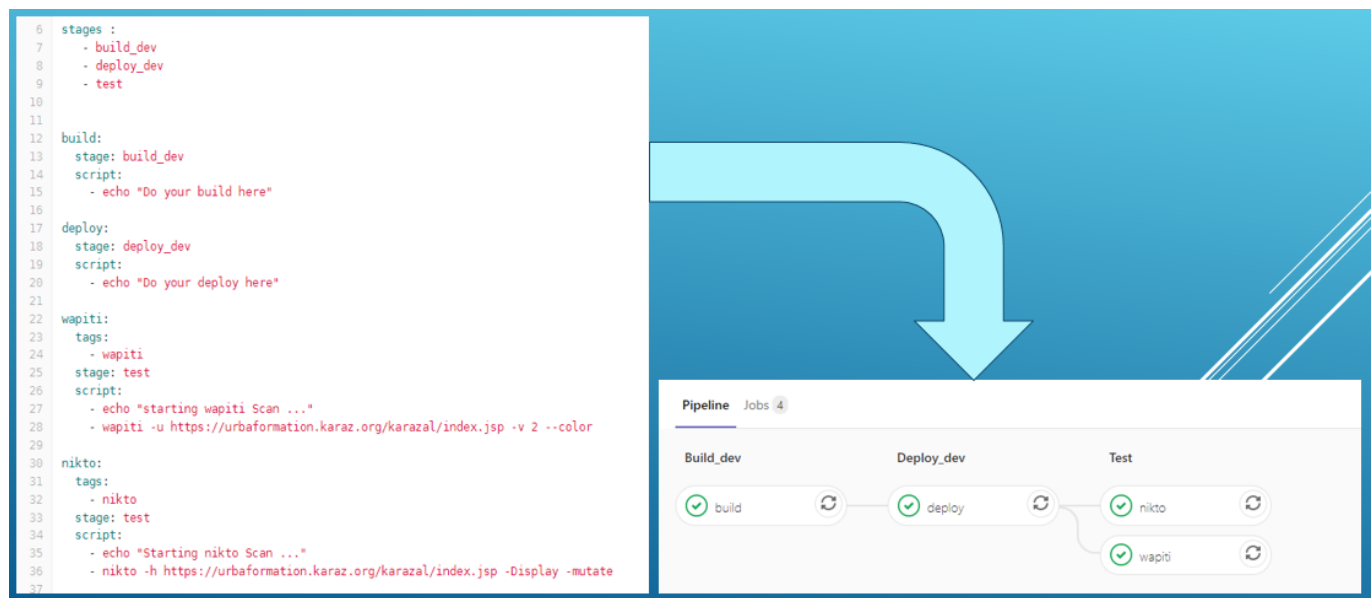


FIGURE 3.2 – Exemple d’une pipeline

Note : le mot clé tags a pour but de spécifier le runner qui va exécuter le script du stage. Par défaut GitLab contient des runners qui peuvent exécuter les commandes (Bash, C++, Maven, Nodejs...), Mais dans notre cas, on a ajouté nos 2 runners un pour exécuter les commandes wapiti et l’autre pour exécuter les commandes de Nikto.