# Rapport de Test

Projet Jamais404

Vallat Quentin, Friche Colin, Jurasz Loïc

INF3DLM-A – HE-ARC



Table des matières

[Rapport de Test 1](#_Toc38547111)

[1. Introduction 3](#_Toc38547112)

[2. Concept de test 3](#_Toc38547113)

[2.1 Attentes du projet EN TERMES DE qualité de produit 3](#_Toc38547114)

[2.2 Objectifs de tests 3](#_Toc38547115)

[2.3 Périmètre de test 4](#_Toc38547116)

[2.4 Contour de la stratégie et l’approche des tests 4](#_Toc38547117)

[2.5 Procédure de test 4](#_Toc38547118)

[3. Plan de test 5](#_Toc38547119)

[3.1 Périmètre de test 5](#_Toc38547120)

[3.2 Stratégie détaillées de test 6](#_Toc38547121)

[3.3 Infrastructure et logistique de test 6](#_Toc38547122)

[3.4 Gestion des risques 7](#_Toc38547123)

[4 Rapport de test 7](#_Toc38547124)

[4.1 Test Unitaire 7](#_Toc38547125)

[4.2 Test d’intégration 7](#_Toc38547126)

[4.3 Analyse SonarCloud 8](#_Toc38547127)

[5 Conclusion 8](#_Toc38547128)

## 1. Introduction

Dans le cadre du cours de Qualité du logiciel à la He-arc en troisième année de Bachelor en ingénierie informatique, il y a un projet de groupe à réaliser pour les cours de Java Entreprise Edition et Qualité du logiciel. Ce rapport présente la partie du travail réalisé pour le cours de qualité du logiciel. Dans ce cours, l’objectif est de mettre en place l’intégration continue du projet réalisé en JEE. Cette intégration continue comprend toutes les parties de tests vue en cours.

Le projet jamais404 est un projet qui permet de trouver des pages sur le site n’ayant pas encore été trouvée. On peut ensuite mettre un commentaire sur une page déjà trouvée et également voir les pages que nous avons trouvées.

## 2. Concept de test

### 2.1 Attentes du projet EN TERMES DE qualité de produit

* Site conforme aux normes standard du Web
* Affichage d’erreurs claires si un formulaire n’est pas valide
* Pas de problèmes signalés par SonarCloud
* Uniquement un mail formé correctement est accepté
* Site pas trop lourd à charger
* Accès au site uniquement par les personnes connectées
* Interface responsive
* Pouvoir rechercher des pages

### 2.2 Objectifs de tests

Le but de nos tests est qu’il couvre au moins 30% du code. Il faut également que le temps de réponse ne soit pas trop long. Le but est de vérifier le bon fonctionnement de notre site web.

Un objectif parallèle qui ne fait pas partie des tests est d’avoir un code propre. Si le code est cohérent, il peut plus aisément être maintenu.

### 2.3 Périmètre de test

Nous avons 3méthodes de tests différentes ainsi qu’une méthode d’analyse de qualité du code :

* Test d’intégration avec Katalon
* Tests unitaires avec JUnit
* Test de charge avec OctoPerf
* Qualité de code avec SonarCloud

### 2.4 Contour de la stratégie et l’approche des tests

Nous allons faire au minimum un test par attente de qualité du projet.

Nous lancerons au préalable les tests unitaires puis on les intègre au déploiement continu. Nous effectuerons les tests d’intégration sur Katalon que nous pourrons au préalable exécuter manuellement. Nous pourrons ensuite également les intégrer au déploiement continu.

Les tests sur OctoPerf ne sont malheureusement pas faisables actuellement.

Le contrôle de qualité du code par SonarCloud est intégré directement.

### 2.5 Procédure de test

Les sous-chapitres suivants décrivent la procédure de test suivie par notre équipe.

#### 2.5.1 Etapes principales

Les étapes 1 et 2 sont exécutées sur le poste de développement lors de leur création, puis testées sur l’environnement de déploiement. Si une des étapes révèle des problèmes qui peuvent être corrigés, ils le seront avant de passer à la suite.

1. Création et exécution des test unitaires avec JUnit
2. Création et exécution des tests d’intégration avec Katalon
3. Analyse de la qualité du code avec SonarCloud
4. Analyse des résultats dans un rapport

#### 2.5.2 Environnement et outils de test

Afin d’être le plus proche du rendu final attendu, l’environnement de test est le même que l’environnement de production.

#### 2.5.3 Plan de communication

L’équipe est composée de 3 développeurs : Colin Friche, Quentin Vallat, Loïc Jurasz. Etant donné que l’équipe est assez petite, la communication a été assez aisée. Le travail s’effectuait fréquemment en même temps en étant en appel vocal les uns avec les autres. Des issues ont également été créées dans GitHub afin de signaler certains problèmes.

## 3. Plan de test

### 3.1 Périmètre de test

#### 3.1.1 Périmètre fonctionnel

* Création d’un compte
* Authentification d’un compte
* Accès à la page de profile
* Logout
* Recherche d’une page
* Accès au profil de celui qui a découvert la page en premier
* Ajout de la page trouvée sur le profil
* Ajout de commentaire sur les pages

#### 3.1.2 Types de test

* Fonctionnalité
  + Pages et commentaires correctement formatés
  + Recherche fonctionnelle
  + Ajout de commentaire
* Fiabilité
  + Résistance aux tests
* Ergonomie
  + Que le site soit User Friendly
* Efficience
  + Temps de chargement du site acceptable
* Maintenabilité
  + Avoir un bon score avec SonarCloud
* Portabilité
  + Affichage responsive

#### 3.1.3 Niveaux de test

* Test unitaire avec JUnit
* Test d’intégration avec Katalon

### 3.2 Stratégie détaillées de test

* Création d’un compte
  + Tester la création d’un compte
  + Tester la validité des entrées du formulaire
* Recherche d’une page
  + Tester que la recherche fonctionne
  + Tester l’affichage de la réussite d’une page nouvelle trouvée
  + Tester l’affichage d’une page ayant déjà été trouvée
  + Tester l’affichage des commentaires sur la page
  + Tester le lien sur le profil de la personne ayant découvert la page
* Création d’un commentaire
  + Tester l’ajout d’un commentaire sur une page trouvée
* Page de profile
  + Tester l’accès à la page de profile
  + Tester l’affichage des pages trouvées par l’utilisateur

### 3.3 Infrastructure et logistique de test

Environnement

* Windows 10 pour les tests d’intégration (Katalon)
* Windows 10, Mac pour les tests unitaires (JUnit)
* Windows 10, Mac pour les tests d’ergonomie

### 3.4 Gestion des risques

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **N°** | **Description** | **Source** | **Impact** | **Proba-**  **bilité** | **Gravité** | **Mitigation** |
| 1 | Difficulté de développement de l’application | Interne | 0.8 | 0.2 | 0.16 | Documentation sur les outils de développement existants |
| 2 | Mauvaise communication dans l’équipe | Interne | 0.7 | 0.2 | 0.14 | Séance de discussion (rencontre ou vocal) et respect de l’organisation mise en place |
| 3 | Difficulté de l’utilisation des outils | Interne | 0.6 | 0.4 | 0.24 | Documentation et lecture du cours |
| 4 | Non-respect des délais | Interne | 0.8 | 0.2 | 0.16 | Travail ponctuel et planning |

## 4 Rapport de test

### 4.1 Test Unitaire

Notre objectif était de tester nos modèles et nos repositories afin de vérifier le bon fonctionnement de nos données.

Pour les tests unitaires, nous avons utilisé JUnit. Chaque model fait le sujet d’une classe de test.

Tous les tests unitaires sont réussis, il n’y a donc pas de possibilité de corruptions ou d’erreurs de données dans notre application

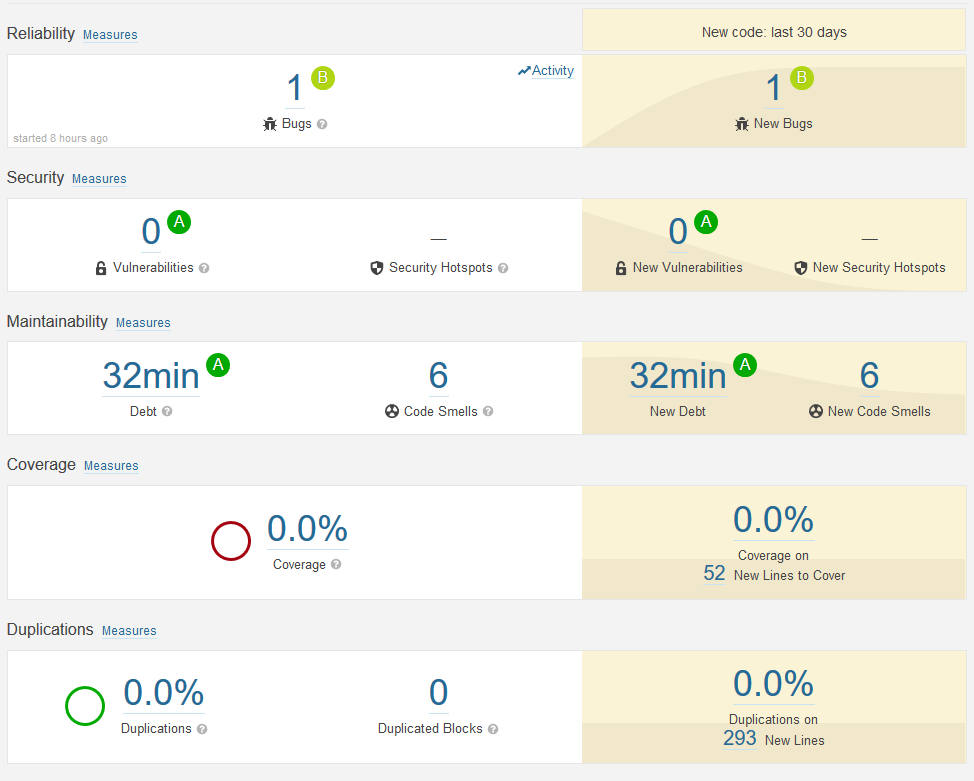
### 4.2 Test d’intégration

Notre objectif est de tester les fonctionnalités de base de notre site web soit utilisables. Afin de réaliser ces test, Katalon a été employé. Les tests effectués servent à vérifier que le login et le register fonctionnent. Nous vérifions également l’accès à la recherche, ainsi que le bon fonctionnement de la recherche de pages, c’est-à-dire une page pas encore trouvée et une page déjà trouvée. Nous vérifions également l’ajout de commentaires et de l’accès à la page de profil, ainsi que la déconnexion.

Les résultats des tests montrent que le login et toutes les fonctionnalités de notre application fonctionnent.

### 4.3 Analyse SonarCloud

Notre objectif pour l’analyse de SonarCloud était de ne pas avoir de bugs et le moins de vulnérabilités possibles.

Comme on le voit sur l’image, nous avons résolus tous les bugs sauf un, qui n’est pas un bug mais que SonarCloud considère comme un. Les unitsTest ne sont également pas prit en compte pour le pourcentage de coverage. Il n’y a également aucune vulnérabilité.

## 5 Conclusion

En implémentant ces différents tests, nous avons pu mettre en lumière certains mauvais fonctionnements de notre application et nous avons pu les corriger. SonarCloud s’est avérés particulièrement utile pour la résolution des vulnérabilités. Les différents tests mit en place nous ont permit d’assurer un bon fonctionnement de notre application.