

J. Racaud; A. Simon; J. Harrault; J. Blondeel; S. Daguenet; F. Corradin

Music Sheet Writer

Music Sheet Writer

Stratégie de Tests

Objectifs du document

Résumé

Glossaire

***– R –***

Release : Mise à disposition d’une version de l’application. Elle peut être privée, semi publique ou publique. En général une release fait état de la version commerciale (ou plus détaillée pour des releases fermées).

Description du document

|  |  |
| --- | --- |
| Titre | Music Sheet Writer : Stratégie de Tests |
| Date de création | 22/10/2015 |
| Date de publication | 15/11/2015 |
| Auteur | J. Racaud; A. Simon; J. Harrault; J. Blondeel; S. Daguenet; F. Corradin |
| Responsable | Jonathan Racaud |
| E-mail | musicsheetwriter\_2017@labeip.epitech.eu |
| Sujet | Stratégie de Tests |
| Version du modèle | 1.0 |

Tableau des révisions

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Auteur** | **Section(s)** | **Commentaire** |
| 22/10/2015 | Jeremy HARRAULT | Toutes | Création du document |

Sommaire

[1. Introduction 1](#_Toc433295550)

[1.1. Objectif de l’EIP 1](#_Toc433295551)

[1.2. Principe de base du système 1](#_Toc433295552)

[1.3. Documents de référence 1](#_Toc433295553)

[2. Version/release 2](#_Toc433295554)

[3. Types de tests et périmètre 3](#_Toc433295555)

[4. Livrables 4](#_Toc433295556)

[4.1. Cas de tests 4](#_Toc433295557)

[4.2. Rapport d’exécution de tests 4](#_Toc433295558)

[4.3. Rapport sur le suivi global de la qualité 4](#_Toc433295559)

[5. Types de tests 5](#_Toc433295560)

[5.1. Tests XXX-1 5](#_Toc433295561)

[5.2. Tests XXX-2 5](#_Toc433295562)

[6. Estimation Globales de la charge de travail 6](#_Toc433295563)

[7. Outils 7](#_Toc433295564)

[7.1. Dépôt de tests 7](#_Toc433295565)

[7.2. Gestion de rejets/erreurs 7](#_Toc433295566)

[7.3. Autres outils 7](#_Toc433295567)

# Introduction

## Objectif de l’EIP

## Principe de base du système

## Documents de référence

# Version/release

# Types de tests et périmètre

# Livrables

## Cas de tests

## Rapport d’exécution de tests

## Rapport sur le suivi global de la qualité

# Types de tests

## Tests XXX-1

### Objectif

### Environnement et conditions de réalisation

### Configurations particulières

### Planning et charge

### Critère de démarrage des tests

### Critères de passage/échec

## Tests XXX-2

### Objectif

### Environnement et conditions de réalisation

### Configurations particulières

### Planning et charge

### Critère de démarrage des tests

### Critères de passage/échec

# Estimation Globales de la charge de travail

# Outils

## Dépôt de tests

Pour réaliser ce projet, nous utilisons un dépôt SVN sur lequel il y a un répertoire par livrable. Chaque répertoire de livrable contient un dossier Tests dans lequel sont mis tous les tests relatifs au livrable.

## Gestion de rejets/erreurs

Un bugtracker va être mis en place afin de faciliter la gestion des rejets et erreurs. Cela implique que lorsque le développeur réalise un test et que celui-ci retourne une erreur, il faut le remonter sur le bugtracker.

## Autres outils

### Outils d’automatisation des tests

#### 7.3.1.1 Logiciel

L’automatisation des tests sous QT se fait à l’aide du framework QTest (<http://doc.qt.io/qt-5/qtest-overview.html>) qui permet notamment le lancement automatique des tests mais également fournit une liste de fonctions permettant de simuler les actions utilisateurs.

#### 7.3.1.2 Application mobile : Android

Pour l’automatisation de test, Android Studio intègre nativement le framework JUnit (<http://junit.org/>). Il gère le lancement automatique des tests.

#### 7.3.1.3 Application mobile : IPhone

Pour l’automatisation des tests de l’interface, il sera nécessaire d’utiliser le *framework* *KIF* installé à l’aide *cocoaPod*. Pour le reste *Xcode* se charge de l’automatisation des tests de performance, de méthode, de classe, etc.

#### 7.3.1.4 Site internet

Pour effectuer les tests sur le site web aussi bien sur la partir interface utilisateur que les tests en arrière-plan on utilise PHPunit qui lance tous les tests regroupés dans les classes des bundles du projet. Cet outil permet d’effectuer plusieurs tests unitaires possibles sur des valeurs de retour de fonction. Les tests seront automatiquement lancés par le script de mise en production.

### Outils pour les tests de performance

### Outils pour les tests d’UI (Application mobile : Android)

Le framework Espresso (<https://google.github.io/android-testing-support-library/docs/espresso/index.html>) est utilisé. Il fournit une bibliothèque de fonctions permettant de simuler les actions utilisateur mais également de vérifier l’état des éléments graphiques.