

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

***– B –***

Bug (ou « bogue ») : Défaut de conception ou d’implémentation d’un programme informatique étant à l’origine d’un disfonctionnement.

***– 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).

***– U –***

UI (User Interface) : Dispositif de de dialogue entre un programme informatique et un utilisateur. Il permet à ce dernier d’effectuer des manipulations au sein du programme informatique et de présenter le résultat de ces manipulations.

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 |
| 04/11/2015 | Simon Daguenet | Toutes | Ajout de toutes les parties |
| 04/11/2015 | Simon Daguenet | Toutes | Ajout de la partie Windows phone |
| 09/11/2015 | Simon Daguenet | 3 : Type de test et périmètre  5 : Types de tests  7 : Outils | Modification des sections |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

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

### Windows phone

#### Plan de release

Voici la liste des releases majeurs de l’application Windows phone. Leurs numéros de version seront notés sous la forme X.0 où X est le numéro de la release majeur. Ce nombre sera incrémenté de 1 à chaque release.

D’autres releases mineures pourront être faites si :

* Une fonctionnalité rapide à implémenter est fortement demandée par la communauté et/ou devient urgent à implémenter pour satisfaire l’expérience MusicSheetWriter.
* Des « bugs » sont trouvés sur l’application, soit par l’équipe de développement, soit pas la communauté. Ces bugs pourront être remontés aux développeurs via la fonctionnalité de report de bug intégré de base au système Windows phone.

Une release mineure sera noté sous la forme X.Y où X est le numéro de la release majeure en cours et Y est le numéro de la release mineure.

#### Plan des release majeures

|  |  |  |  |
| --- | --- | --- | --- |
| Numéro de version | Date de sortie prévue | Description | Fonctionnalités ajoutées |
| 1.0 | 4/08/2016 | Cette version sera la première version de l’application Windows phone. Elle sortira en même temps que la première version du logiciel. Aucune version n’est prévus avant car la gestion des partitions ne sera pas utilisé sans le logiciel et l’aspect communautaire ne sera pas assez poussez pour se suffire à lui seul. | * Création et connexion à un compte utilisateur * Gestion des informations personnelles * Consultation des partitions de l’utilisateur * Gestion des partitions favorites de l’utilisateur * Gestion des abonnements de l’utilisateur * Recherche d’utilisateur * Recherche de partition * Lecture sonore d’une partition |
| 2.0 | 30/03/2017 | Cette version sera tournée vers les fonctionnalités communautaires et commencera son approche vers l’implémentation des fonctionnalités d’édition de partitions. | * Correction de bugs * Commenter les partitions * Envoie de message à d’autre utilisateur * Lecture d’une partition * Téléchargement d’une partition depuis le serveur sous format PDF |
| 3.0 | 30/09/2017 | Cette version rapprochera les fonctionnalités de lecture et d’édition de partition du logiciel et de l’application Windows phone. | * Correction de bugs * Création d’un projet MusicSheetWriter * Edition « note-à-note » d’une partition * Téléchargement d’une partition vers et depuis le serveur sous forme de projet MusicSheetWriter |

# Types de tests et périmètre

## Windows phone

### Test unitaire

Ces tests vérifieront le bon fonctionnement de classes ou méthode particulière. Ils seront effectués hors d’un contexte spécifique à l’application Windows phone MusicSheetWriter. En effet, ils ne nécessitent pas d’élément visuel particulier. Ce type de test sera notamment pour les classes et méthodes utilitaires et susceptible d’être réutilisé dans d’autres projets comme les classes d’appel HTTP, les classes de validation de formulaire ou encore d’authentification. On pourra ainsi valider la qualité du code et les performances d'un module. Les tests unitaires auront une couverture complète de l’application.

### Test de non-régression

Les tests de non-régression permettent de vérifier que des modifications n'ont pas altérées le fonctionnent de l'application. Les tests de non-régression auront une couverture complète de l’application. Ces tests seront établis à partir des tests unitaires.

### Test UI

Les tests UI vérifieront le bon fonctionnement des interactions entre l’application et l’utilisateur (l’appui d’un bouton ou le défilement d’une page, etc.) et de leurs conséquences (affichage d’un texte, déclenchement d’une animation, etc.). Ils vérifieront également la gestion des périphériques comme l’appareil photo. Chaque activité et fragment ayant un contenu visuel sera soumis à des tests UI.

### Test fonctionnel

Ces tests ont pour but de vérifier la conformité de l'application développée avec le cahier des charges initial. Ils sont donc basés sur les spécifications fonctionnelles et techniques. Les tests fonctionnels auront une couverture complète de l’application.

# Livrables

## Cas de tests

### Windows phone

#### L’arborescence

Le livrable des cas de tests pour l’application Windows phone suivra la même arborescence que dans leur implémentation qui se trouvera dans le projet « test » de la solution Music Sheet Writer.

Les cas de tests pour l’application Windows phone seront chaque nouvelle implémentation de fonctionnalité (le login, la création de compte, ..). Ce système permet une fréquence de mise à jour des tests corrects.

#### Test Unitaire

|  |  |
| --- | --- |
| *Chaque test comportera…* | *Afin de…* |
| Une courte description présentant le test | Connaitre l’élément testé |
| Les données utilisées en temps qu’entré pour le test | Savoir quelle valeur sont testés et par substitution, lesquels ne le sont pas |
| Les conditions à succès | Connaitre quelle et comment est la sortie du test |
| La durée approximative du test | Connaitre le temps nécessaire à son exécution normale. |

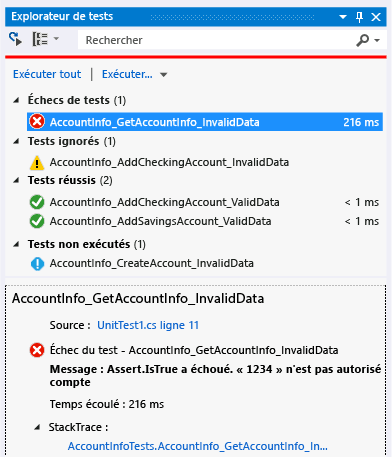
#### Test UI

|  |  |
| --- | --- |
| *Chaque test comportera…* | *Afin de…* |
| Une courte description présentant le test | Connaitre la fonctionnalité testé et comprendre le contexte dans lequel se trouve le user s’il devait faire le test manuellement |
| Les données utilisées en temps qu’entré pour le test | Savoir quelles valeurs sont utilisées pour le test |
| Les conditions à succès | Connaitre quelle et comment est la sortie du test |
| La durée approximative du test | Connaitre le temps nécessaire à son exécution normale. |

## Rapport d’exécution de tests

Le rapport d’exécution de tests s’effectuera via l’interface de Microsoft Visual Studio 2015 dans un projet « unit test » qui sera ajouté à la solution projet. Dans cette interface nous sommes en mesure d’observer le résultat des tests effectués auparavant. Ainsi, nous pouvons observer le résultat d’un test réussi et un échoué.

Le livrable présentant les cas de tests pour Windows phone sera sous la forme d’un fichier (.rtf). Il présentera tous les tests joués, regroupé par suite et campagne de tests.



## Rapport sur le suivi global de la qualité

Le rapport sur le suivi global de la qualité s’effectuera grâce au test de non-régression qui permettra au chef de groupe d’avoir une vision des tests effectués précédemment par le développeur. Ainsi il sera en mesure de connaitre la qualité du code en visualisant les tests effectué. Le chef de groupe pourra également s’appuyer sur les tests fonctionnels qui lui permettront de vérifier la conformité de l'application développée avec le cahier des charges initial.

# Types de tests

## Windows phone

### Tests unitaire

#### Objectif

Les tests unitaires sont utilisés de manière générale afin de permettre le bon déroulement du développement de l’application. Ils permettront de tester individuellement les composants de l’application. On pourra ainsi valider la qualité du code et les performances d'un module.

#### Environnement et conditions de réalisation

Ces tests pourront être joués à n’importe quel moment du développement. Le test sera réalisé aussitôt qu’une classe ou une fonctionnalité n’agissant pas sur l’interface graphique est spécifiée. Il sera sous la forme d’une ou plusieurs classes C# représentant les suites de test, chacune comportant des méthodes représentant les cas de test à exécuter.

Si la suite de test vérifie une simple classe (Validateur d’email, validateur de mot de passe, téléchargeur d’image) elle pourra être codée en une classe. En revanche, si la suite de test vérifie fonctionnalité comportant un ensemble de classes (expéditeur de requête HTTP), il est préférable de la diviser en plusieurs suites de test pertinentes pour les regrouper dans un même dossier.

#### Configurations particulières

Seul un périphérique capable de lancer l’application à tester est nécessaire.

#### Planning et charge

Les tests seront effectués régulièrement afin de valider l’avancée du projet et le code préalablement effectué. Cela permettra également de valider la qualité du code et les performances d'un module.

#### Critère de démarrage des tests

L’application doit être installée sur le périphérique cible.

#### Critères de passage/échec

Les critères de passage/échec seront définis via les tests effectués. En effet, si la réponse retour est la même que la réponse attendu le test sera passé sinon ce sera un échec.

### Tests de non-régression

#### Objectif

Les tests de non-régression sont utilisés de manière générale afin de permettre le bon déroulement du développement de l’application. Les tests de non-régression permettent de vérifier que des modifications n'ont pas altérées le fonctionnent de l'application. Les tests de non-régression auront une couverture complète de l’application.

#### Environnement et conditions de réalisation

Les tests de non-régression seront utilisés tout au long du projet et sur toutes les releases. Les tests seront effectués sous Microsoft Visual Studio via la création d’un projet de test unitaire.

#### Configurations particulières

Seul un périphérique capable de lancer l’application à tester est nécessaire.

#### Planning et charge

Les tests seront effectués régulièrement afin de valider l’avancée du projet et le code préalablement effectué. Cela permettra également de vérifier que des modifications n'ont pas altérées le fonctionnent de l'application.

#### Critère de démarrage des tests

L’application doit être installée sur le périphérique cible.

#### Critères de passage/échec

Les critères de passage/échec seront définis via les tests effectués. En effet, si la réponse retour est la même que la réponse attendu le test sera passé sinon ce sera un échec.

### Test UI

#### Objectif

L’objectif de ces tests et de vérifier le bon fonctionnement de la capture des événements graphiques et les conséquences qu’elles engendrent sur l’interface utilisateurs. Ces tests concernent les classes chargées de l’affichage graphique, c’est-à-dire les activités et les fragments. De telles classes seront testées de manière unitaire. En effet, chaque activité ou fragment aura une ou plusieurs classes de test associées.

#### Environnement et conditions de réalisation

Ces tests pourront être joués à n’importe quel moment du développement. Le test sera réalisé aussitôt que l’interface et les événements graphiques d’une activité. Il sera sous la forme d’une ou plusieurs classes C# représentant les suites de test, chacune comportant des méthodes représentant les cas de test à exécuter.

#### Configurations particulières

Seul un périphérique capable de lancer l’application à tester est nécessaire.

#### Planning et charge

Les tests seront effectués régulièrement afin de valider l’avancée du projet et le code préalablement effectué. Cela permettra également de valider la qualité du code et les performances d'un module.

#### Critère de démarrage des tests

L’application doit être installée sur le périphérique cible.

#### Critères de passage/échec

Les critères de passage/échec seront définis via les tests effectués. En effet, si la réponse retour est la même que la réponse attendu le test sera passé sinon ce sera un échec.

### Tests fonctionnels

#### Objectif

Les tests fonctionnels sont utilisés de manière générale afin de permettre le bon déroulement du développement de l’application. Ces tests ont pour but de vérifier la conformité de l'application développée avec le cahier des charges initial. Ils sont donc basés sur les spécifications fonctionnelles et techniques. Les tests fonctionnels auront une couverture complète de l’application.

#### Environnement et conditions de réalisation

Les tests fonctionnels seront utilisés tout au long du projet et sur toutes les releases. Les tests seront effectués sous Microsoft Visual Studio via la création d’un projet de test unitaire.

#### Configurations particulières

Seul un périphérique capable de lancer l’application à tester est nécessaire.

#### Planning et charge

Les tests seront effectués régulièrement afin de valider l’avancée du projet et le code préalablement effectué. Cela permettra également de vérifier la conformité de l'application développée avec le cahier des charges initial. Ils sont donc basés sur les spécifications fonctionnelles et techniques.

#### Critère de démarrage des tests

L’application doit être installée sur le périphérique cible.

#### Critères de passage/échec

Les critères de passage/échec seront définis via les tests effectués. En effet, si la réponse retour est la même que la réponse attendu le test sera passé sinon ce sera un échec.

# Estimation Globales de la charge de travail

La charge de travail est déjà prise en compte dans le « gant » que nous avons fourni. Nous aurons à effectuer les tests nécessaires lors de chaque réalisation de l’une des fonctionnalités.

# Outils

## Windows phone

### Dépôt de tests

Les tests seront sur un dépôt SVN. Ils seront placés dans les sources de la solution Windows phone, dans un projet nommé « test ».

### Gestion de rejets/erreurs

Aucun outil de gestion des erreurs n’est prévu.

### Autres outils

#### Outils d’automatisation des tests

Pour l’automatisation des tests de l’interface, il sera nécessaire d’utiliser l’interface de Microsoft Visual Studio 2015 dans un projet « unit test » qui sera ajouté à la solution projet.