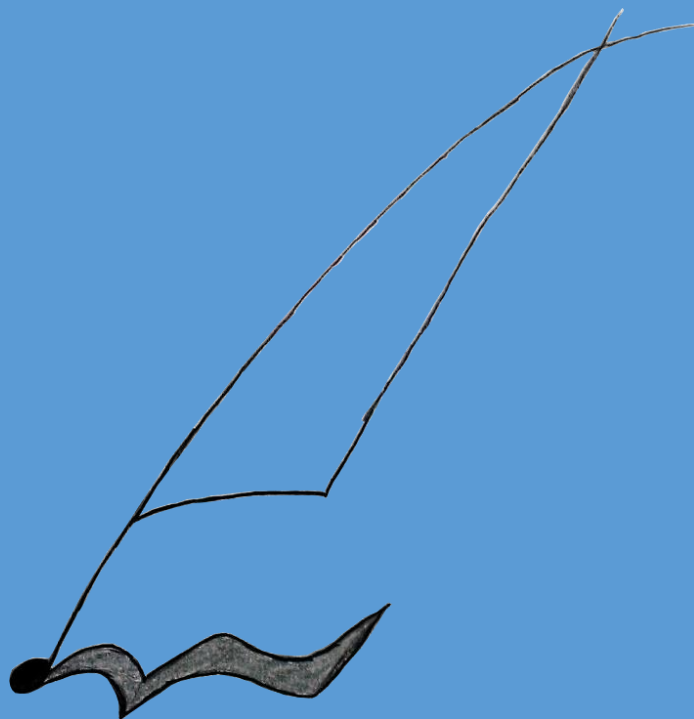




MUSIC SHEET WRITER

STRATÉGIE DE TEST



J. Racaud; A. Simon; J. Harraut; J. Blondeel; S. Daguene; F. Corradin

MUSIC SHEET WRITER

 Music Sheet Writer	Date de publication	17/11/2015	 Epitech Innovative Project
	Nom du projet	Music Sheet Writer	
	Objet du document	Stratégie de Test	
	Nom du chapitre	<i>Objectifs du document</i>	

Objectifs du document

Résumé

Le projet Music Sheet Writer est axé autour d'un logiciel d'édition de partition, qui permet la génération des dites partitions directement depuis un piano ou une guitare branché à l'ordinateur de l'utilisateur. Ce logiciel est accompagné d'un site internet pour la promotion du logiciel et qui fournit des fonctionnalités communautaires ; d'applications mobiles pour Android, iPhone et Windows Phone qui donneront aussi accès aux fonctions communautaires présentes sur le site internet.



Ce projet est donc vaste et complexe, d'où la nécessité d'avoir une stratégie de test solide et à maintenir pour la bonne réalisation de ce dernier. Ce document a pour but d'expliquer la stratégie de test mise en place.

Le projet étant constitué de plusieurs livrables à destination de différentes plateformes, les types de tests effectués ou encore la date de sortie de ces livrables peut varier d'un livrable à un autre.

Le logiciel sortira en version 1.0 en début Mars accompagné du site vitrine et permettra la gestion d'un projet, la lecture d'une partition et l'édition de cette dernière. La partie communautaire du site internet et les applications mobiles sortiront quant à eux en début Avril, aussi en version 1.0. Bien entendu ces dates-là pourront être modifiées en fonction de l'avancée de chacun des livrables. Différentes sorties sont prévues sur toute la durée du développement du projet.

Les tests communs à tous les livrables sont les tests unitaires. Ces derniers permettront de s'assurer du bon fonctionnement du code produit à chacune des étapes du développement du projet. La description des cas de tests et les résultats de ces derniers seront fournis dans des documents spécifiques créés depuis les outils de tests utilisés.

Les outils utilisés pour la génération des tests et leurs lancements sont spécifiques aux langages de programmation utilisés pour la réalisation des différents livrables.

 Music Sheet Writer	Date de publication	17/11/2015	 Epitech Innovative Project
	Nom du projet	Music Sheet Writer	
	Objet du document	Stratégie de Test	
	Nom du chapitre	Objectifs du document	

Glossaire

– A –

Activité (Android) : Composant applicatif fournissant un écran avec lequel les utilisateurs peuvent interagir afin d'effectuer une action. Une application est composée d'une ou plusieurs activités.

API (Application Programming Interface) : Une API est un ensemble d'artéfacts servant de façade pour permettre à un logiciel d'offrir des services à d'autres logiciels. C'est une porte d'accès à des fonctionnalités en faisant abstraction de leur fonctionnement.

– B –

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

Bug Tracker : Voir « Logiciel de suivi de problème ».

– F –

Fragment (Android) : Composant d'une activité représentant un comportement ou une portion de l'interface utilisateur. Une activité peut être composée d'une ou plusieurs fragments.

Framework : Un framework est un ensemble de composants qui permettent de structurer et de mettre en place les fondations d'un tout ou d'une partie d'un logiciel.

– H –

HTTP (Hypertext Transfer Protocol) : HTTP est un protocole de communication client-serveur. Il peut fonctionner sur n'importe quelle connexion fiable. Les clients HTTP les plus connus sont les navigateurs web permettant aux utilisateurs d'accéder à un serveur contenant des données.

– L –

Logiciel de suivi de problème (« Bug Tracker ») : Un logiciel de suivi de problèmes permet aux développeurs de garder la trace des problèmes connus au sein du système en développement, notamment de le but de les résoudre.

– M –

Mock : Les Mocks sont des objets simulés qui reproduisent le comportement d'objets réels de manière contrôlée. On utilise un mock dans le but de tester le comportement d'autres objets.

– 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).



REST (Representation State Transfer) : REST est un style d'architecture logiciel qui, la plupart du temps est utilisé pour implémenter un web service. La communication se fait donc entre un client et un serveur en utilisant le protocole HTTP.

– S –

Symfony : Symfony est un framework PHP pour créer des sites web et des applications web.

Script : Programme qui exécute plusieurs tâches à la suite.

– U –



 Music Sheet Writer	Date de publication	17/11/2015	 Epitech Innovative Project
	Nom du projet	Music Sheet Writer	
	Objet du document	Stratégie de Test	
	Nom du chapitre	<i>Objectifs du document</i>	

UI (User Interface) : Dispositif 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.

– **X** –

Xcode : Xcode est un environnement de développement pour Mac OS X ainsi que pour iOS.

Xcpretty : c'est un framework capable de rendre plus lisible et clair la sortie des tests pour pouvoir en créer des livrables.



 Music Sheet Writer	Date de publication	17/11/2015	 Epitech Innovative Project
	Nom du projet	Music Sheet Writer	
	Objet du document	Stratégie de Test	
	Nom du chapitre	<i>Description du document</i>	

Description du document

Titre	Music Sheet Writer : Stratégie de Test
Date de création	22/10/2015
Date de publication	17/11/2015
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 Test
Version du modèle	1.0



Tableau des révisions

Date	Auteur	Section(s)	Commentaire
22/10/2015	Jeremy Harrault	Toutes	Création du document
29/10/2015	Antoine Simon	Toutes	Rédaction partie iPhone
04/11/2015	Jeremy Harrault	2. Version/release 3. Types de tests et périmètre	Ajout de la partie Android
04/11/2015	Simon Daguenet	3 Type de test et périmètre 4. Types de tests 7. Outils	Modification des sections Windows phone
06/11/2015	Jeremy Harrault	4. Livrables 5. Types de tests et périmètre 7. Outils	Ajout de la partie Android
08/11/2015	Julien Blondeel	Toutes	Rédaction de la partie logicielle
09/11/2015	Simon Daguenet	Toutes	Ajout de la partie Windows Phone
10/11/2015	Julien Blondeel	Toutes	Modification suite à la réunion
10/11/2015	Antoine Simon	Toutes	Correction partie iPhone en fonction des remarques
14/11/2015	Jonathan Racaud	Résumé 1. Introduction	Rédaction
14/11/2015	Jeremy Harrault & Jonathan Racaud	Toutes	Première relecture
15/11/2015	Jeremy Harrault	Toutes	Relecture finale

 Music Sheet Writer	Date de publication	17/11/2015	 Epitech Innovative Project
	Nom du projet	Music Sheet Writer	
	Objet du document	Stratégie de Test	
	Nom du chapitre	Sommaire	

Sommaire

1. Introduction	1
1.1. Objectif de l'EIP	1
1.2. Principe de base du système	1
2. Version/release.....	2
2.1. Numérotation des versions	2
2.2. Logiciel.....	3
2.3. Site Web.....	4
2.4. Applications mobiles.....	5
3. Types de tests et périmètre	8
3.1. Commun à tous les livrables.....	8
3.2. Spécifiques au site web.....	8
3.3. Spécifiques aux applications mobiles.....	9
4. Livrables	10
4.1. Cas de tests	10
4.2. Rapport d'exécution de tests	12
4.3. Rapport sur le suivi global de la qualité	17
5. Types de tests	18
5.1. Logiciel.....	18
5.2. Site Web.....	20
5.3. Android.....	22
5.4. iPhone.....	25
5.5. Windows Phone	27
6. Outils.....	29
6.1. Dépôt de tests	29
6.2. Gestion de rejets/erreurs.....	29
6.3. Autres outils.....	29



 Music Sheet Writer	Date de publication	17/11/2015	 Epitech Innovative Project
	Nom du projet	Music Sheet Writer	
	Objet du document	Stratégie de Test	
	Nom du chapitre	<i>Liste des Tableaux</i>	

Liste des Tableaux

Tableau 1 : Plan de release du logiciel	3
Tableau 2 : Plan de release du site web	4
Tableau 3 : Plan de release de l'application Android	5
Tableau 4 : Plan de release de l'application iPhone.....	6
Tableau 5 : Plan de release de l'application Windows Phone.....	7
Tableau 7 : Composition d'un test unitaire	10
Tableau 6 : Composition d'un test d'acceptation	11

Liste des Figures

Figure 1 : Rapport d'exécution des test unitaires du logiciel [succès]	12
Figure 2 : Rapport d'exécution des tests unitaires du logiciel [échec].....	12
Figure 3 : Rapport d'exécution des tests unitaires du site web [succès]	13
Figure 4 : Rapport d'exécution des test unitaires du site web [échec]	13
Figure 5: Rapport d'exécution des tests de performance	14
Figure 5 : Rapport d'exécution des tests de l'application Android	14
Figure 6 : Rapport d'erreur d'un test de l'application Android	15
Figure 8 : Rapport d'exécution des tests de l'application iPhone	16
Figure 9 : Fenêtre de test sous Visual Studio	17
Figure 10 : Exemple de test unitaire avec QTestLib	18
Figure 11 : Ajout d'un test dans la configuration d'exécution avec QT	18
Figure 12 : Compilation des tests avec QT	18
Figure 13 : Exemple de test unitaire avec QT [succès].....	19
Figure 14 : Définition d'une configuration de lancement de tests sur Android Studio	22

 Music Sheet Writer	Date de publication	17/11/2015	 Epitech Innovative Project
	Nom du projet	Music Sheet Writer	
	Objet du document	Stratégie de Test	
	Nom du chapitre	<i>Introduction</i>	

1. Introduction

1.1. Objectif de l'EIP

EPITECH est l'école de l'expertise informatique, transformant une passion en véritable expertise. L'apprentissage à EPITECH est fondé sur une pédagogie par projets, individuels ou en groupe, validant un certain nombre de connaissances et de notions à assimiler. Tout au long de leur cursus, les étudiants se familiarisent avec le milieu professionnel, notamment grâce aux stages en première, troisième et cinquième année d'une période de quatre à six mois. L'école forme les étudiants à s'adapter à des situations inhabituelles avec la mise en place de rush (projets à réaliser sur un week-end, sur des sujets et notions dont les élèves n'ont aucune connaissance) ou le départ à l'international pendant leur quatrième année ; année durant laquelle l'étudiant va devoir faire preuve d'autonomie et de capacité d'adaptation.



Les Epitech Innovative Projects sont des projets à réaliser sur le cycle master du cursus Epitech. Ils sont conçus à la manière d'un véritable projet entrepreneurial, dans toutes ses composantes : business, techno, design & communication. Un EIP est appelé à devenir une start-up viable. Le but de l'EIP est donc de faire découvrir aux étudiants le monde de l'entrepreneuriat en leur demandant de mettre un peu un projet et de le réaliser en faisant face à des difficultés qu'ils n'avaient jusqu'alors pas rencontrées. Le principal obstacle est la gestion de groupe composé de membres dispersés dans des pays différents, faisant face alors aux problèmes de gestion du temps et des zones horaires pour leur quatrième année. Les problématiques de communication et de vente du produit sont aussi abordées.

1.2. Principe de base du système

Music Sheet Writer est un logiciel d'édition de partition destiné aux musiciens néophytes et amateurs qui n'ont pas forcément les connaissances théoriques du solfège pour écrire leurs compositions. Il se présente donc comme tout logiciel d'édition de partition existant, mais apporte une fonctionnalité majeure : la génération d'une partition depuis un piano ou une guitare branchés à l'aide d'un câble JACK ou d'une interface audio USB.

Le mot d'ordre de Music Sheet Writer est d'être simple d'utilisation. En effet, en ajoutant cette fonctionnalité, nous simplifions la phase d'écriture lors de la composition d'une musique. Laisant l'utilisateur se concentrer sur la musique avant son écriture. Bien entendu, les musiciens aguerris ne seront pas en reste puisque Music Sheet Writer incorporera les outils qui leur permettront d'écrire leurs musiques de manière très précise.

Ce projet est composé de plusieurs projets. Le principal est le logiciel d'édition de partition à proprement parler. Il sera disponible à la fois sur les plateformes Windows et Mac OS. Le second projet est le site internet qui permettra de vendre le logiciel. Ce site internet aura aussi des fonctionnalités communautaires basiques (partage de partition, suivi d'utilisateurs, mise en favoris de partitions...). Enfin, trois applications mobiles seront aussi développées et elles permettront d'accéder à l'aspect communautaire développé autour de Music Sheet Writer.

 Music Sheet Writer	Date de publication	17/11/2015	 Epitech Innovative Project
	Nom du projet	Music Sheet Writer	
	Objet du document	Stratégie de Test	
	Nom du chapitre	<i>Version/release</i>	

2. Version/release

2.1. Numérotation des versions



Malgré le fait que le projet soit constitué de différents livrables, les différentes versions de chacun des livrable suivra la même numérotation.

Les release de versions majeures seront de la forme vX.0 où X correspond au numéro de version majeure, cette valeur s'incrémente de 1 à chaque release d'un livrable.

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 Music Sheet Writer.
- Des bugs sont trouvés sur l'application, soit par l'équipe de développement, soit pas la communauté.

La notation d'une release sera donc 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.0



 Music Sheet Writer	Date de publication	17/11/2015	 Epitech Innovative Project
	Nom du projet	Music Sheet Writer	
	Objet du document	Stratégie de Test	
	Nom du chapitre	Version/release	

2.2. Logiciel

2.2.1. Plan de release

Numéro de version	Date de sortie prévue	Description	Fonctionnalités
1.0	18/03/16	Cette version sera la première version du logiciel. Elle implémentera les fonctionnalités de bases, à savoir la gestion de projet, la lecture et l'édition de partition.	Gérer un projet : <ul style="list-style-type: none"> - Créer un projet - Ouvrir / fermer un projet - Sauvegarder un projet - Importer un projet - Exporter un projet Lire une partition : <ul style="list-style-type: none"> - Démarrer la lecture - Mettre en pause la lecture - Placer et déplacer le curseur de lecture Editer une partition : <ul style="list-style-type: none"> - Modifier le type de mesure - Placer / supprimer un élément musical - Modifier le tempo - Placer les curseurs d'édition - Ajouter les notes à la partition
2.0	13/05/16	Dans cette version, l'utilisateur peut générer une partition directement en jouant de son instrument branché par port MIDI.	Générer une partition par port MIDI
3.0	30/09/16	Cette version sera la version finale du logiciel. Dans cette version, l'utilisateur peut générer une partition directement en jouant de son instrument branché par port JACK	Générer une partition par port JACK

Tableau 1 : Plan de release du logiciel



 Music Sheet Writer	Date de publication	17/11/2015	 Epitech Innovative Project
	Nom du projet	Music Sheet Writer	
	Objet du document	Stratégie de Test	
	Nom du chapitre	Version/release	

2.3. Site Web

2.3.1. Plan de release

Numéro de version	Date de sortie prévue	Description	Fonctionnalités ajoutées
0.1	18/03/16	Cette version sera la première version du logiciel. Elle permettra d'avoir accès à la première version du Site Web. Il inclura une présentation de celui-ci ainsi les fonctionnalités de gestion de compte.	Vitrine : <ul style="list-style-type: none"> - Présentation de Music Sheet Writer - Plateforme d'achat Gestion de compte utilisateur : <ul style="list-style-type: none"> - Création de compte - Connexion/Déconnexion - Modification des informations de compte - Gestion de ses partitions
1.0	08/04/16	Cette version inclura les fonctionnalités de partage de partitions ainsi que de recherche du contenu de la communauté.	Gestion de la communauté : <ul style="list-style-type: none"> - Rechercher un utilisateur - Recherche une partition - Consulter un profil utilisateur - Ajouter une partition en favoris - S'abonner à un utilisateur

Tableau 2 : Plan de release du site web

 Music Sheet Writer	Date de publication	17/11/2015	 Epitech Innovative Project
	Nom du projet	Music Sheet Writer	
	Objet du document	Stratégie de Test	
	Nom du chapitre	Version/release	

2.4. Applications mobiles



Les applications mobiles respectent le même plan de releases. La séparation en trois tableaux différents facilitera les modifications potentielles à effectuer sur l'un d'entre eux.

2.4.1. Android

2.4.1.1. Plan de release

Numéro de version	Date de sortie prévue	Description	Fonctionnalités ajoutées
1.0	4/08/2016	Cette version sera la première version de l'application Android. Elle sortira en même temps que la première version du logiciel. Aucune version n'est prévue avant car la gestion des partitions ne sera pas utilisé sans le logiciel et l'aspect communautaire ne sera pas assez poussé pour se suffire à lui seul.	<ul style="list-style-type: none"> – 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.	<ul style="list-style-type: none"> – Correction de bugs – Commenter les partitions – Envoi 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 Android.	<ul style="list-style-type: none"> – Correction de bugs – Création d'un projet Music Sheet Writer – Edition « note-à-note » d'une partition – Téléchargement d'une partition vers et depuis le serveur sous forme de projet Music Sheet Writer

Tableau 3 : Plan de release de l'application Android



 Music Sheet Writer	Date de publication	17/11/2015	 Epitech Innovative Project
	Nom du projet	Music Sheet Writer	
	Objet du document	Stratégie de Test	
	Nom du chapitre	Version/release	

2.4.2. iPhone

2.4.2.1. Plan de release

Numéro de version	Date de sortie prévue	Description	Fonctionnalités ajoutées
1.0	4/08/2016	Cette version sera la première version de l'application IOS. Elle sortira en même temps que la première version du logiciel. Aucune version n'est prévue avant car la gestion des partitions ne sera pas utilisée sans le logiciel et l'aspect communautaire ne sera pas assez poussé pour se suffire à lui seul.	<ul style="list-style-type: none"> – 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.	<ul style="list-style-type: none"> – Correction de bugs – Commenter les partitions – Envoi 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 IOS.	<ul style="list-style-type: none"> – Correction de bugs – Création d'un projet Music Sheet Writer – Edition « note-à-note » d'une partition – Téléchargement d'une partition vers et depuis le serveur sous forme de projet Music Sheet Writer

Tableau 4 : Plan de release de l'application iPhone



 Music Sheet Writer	Date de publication	17/11/2015	 Epitech Innovative Project
	Nom du projet	Music Sheet Writer	
	Objet du document	Stratégie de Test	
	Nom du chapitre	Version/release	

2.4.3. Windows Phone

2.4.3.1. Plan de release

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 prévue avant car la gestion des partitions ne sera pas utilisée sans le logiciel et l'aspect communautaire ne sera pas assez poussé pour se suffire à lui seul.	<ul style="list-style-type: none"> – 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.	<ul style="list-style-type: none"> – Correction de bugs – Commenter les partitions – Envoi 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.	<ul style="list-style-type: none"> – Correction de bugs – Création d'un projet Music Sheet Writer – Edition « note-à-note » d'une partition – Téléchargement d'une partition vers et depuis le serveur sous forme de projet Music Sheet Writer

Tableau 5 : Plan de release de l'application Windows Phone

 Music Sheet Writer	Date de publication	17/11/2015	 Epitech Innovative Project
	Nom du projet	Music Sheet Writer	
	Objet du document	Stratégie de Test	
	Nom du chapitre	<i>Types de tests et périmètre</i>	

3. Types de tests et périmètre

3.1. Commun à tous les livrables

3.1.1. Tests unitaires

Ces tests ont pour but de confirmer qu'une portion de code fournit bien une sortie correcte en fonction de paramètres d'entrées donnés, ceci afin de s'assurer qu'elle répond aux spécifications fonctionnelles. Les tests unitaires seront testé de manière unitaire et donc, sans contexte particulier, ce qui assurera leur fonctionnement en toutes circonstances (classe envoie de requête HTTP, de lecture d'une partition, etc.)

3.1.1. Tests d'intégration UI

Ces tests ont pour but de simuler les interactions entre l'utilisateur et le logiciel. Pour chaque élément ajouté dans l'interface, tel qu'un bouton ou un menu, le développeur doit tester ce dernier, afin de s'assurer que l'action entrepris à la bonne conséquence (ouverture d'une nouvelle fenêtre, lancement de la lecture, affichage de message d'erreur, etc.).

3.1.1. Tests d'acceptation

Ces tests ont pour but de valider un dispositif complet et non pas une simple conversion d'entrées-sorties. En effet, ces tests ne valident pas le code mais vérifient la conformité du produit avec le cahier de charges (création, ouverture et fermeture d'un projet, recherche et consultation d'un profil d'un utilisateur, etc.)

3.1.2. Tests de non-régressions

Ces tests ont pour but de vérifier que l'intégration ou la modification d'un bout de code n'a pas modifié le comportement du logiciel antérieurement validé. Ces tests portent sur l'exécution de tests unitaires déjà joués afin de s'assurer que le système répond toujours aux exigences spécifiées.

3.2. Spécifiques au site web

3.2.1. Tests de performance



Ces tests ont pour but de vérifier la fiabilité du serveur physique et de déterminer s'il est conforme aux besoins en termes d'usage de celui-ci et le cas échéant, la marge de manœuvre disponible avant de devoir revoir l'infrastructure. Afin que ces tests soient pertinents, ils doivent être lancés sur un serveur aux caractéristiques identiques et dans les mêmes conditions que le serveur de production.

3.2.1.1. Tests de stress

Ces tests ont pour but de déterminer les limites du serveur en termes d'usage. Au fur et à mesure du test, le nombre de requête simultanée augmente jusqu'à ce que le serveur ne réponde plus. C'est le nombre de requête simultanée qui ont contribué au plantage du serveur qui constitue le résultat du test de stress.

3.2.1.2. Tests de charge



Ces tests ont pour but de déterminer la durée pendant laquelle le serveur répond en condition de charge limite. Pour garantir une meilleure fiabilité, cette charge est déterminée à partir des résultats des tests de stress. Ainsi, les tests de charge devront être lancés après les tests de stress.

 Music Sheet Writer	Date de publication	17/11/2015	 Epitech Innovative Project
	Nom du projet	Music Sheet Writer	
	Objet du document	Stratégie de Test	
	Nom du chapitre	<i>Types de tests et périmètre</i>	

3.3. Spécifiques aux applications mobiles

3.3.1. Tests d'intégration des appels API

Ces tests ont pour but de vérifier le bon fonctionnement des actions produisant un appel à l'API et les conséquences sur l'interface utilisateurs des différentes réponses renvoyées par le serveur. Il sera ainsi possible de déterminer si l'application mobile réagit correctement en fonction des réponses renvoyées par l'API (ouverture de la page principale de l'application après une connexion réussie, affichage des messages d'erreurs appropriés si l'API renvoie une erreur, etc.). Ainsi, au sein des applications mobiles seules les pages étant liées à l'envoi d'une requête seront concernées par ces tests.

 Music Sheet Writer	Date de publication	17/11/2015	 Epitech Innovative Project
	Nom du projet	Music Sheet Writer	
	Objet du document	Stratégie de Test	
	Nom du chapitre	<i>Livrables</i>	

4. Livrables

4.1. Cas de tests

4.1.1. Arborescence

L'ensemble des tests pour chacun des livrables se trouveront dans le fichier /Tests/cas_de_test.docx.



Le document sera découpé par livrable et les cas de test seront écrit dans l'ordre suivant :

- Logiciel
 - o Tests unitaires
 - o Tests d'intégration UI
 - o Tests d'acceptation
 - o Tests de non-régression
- Site Web
 - o Tests unitaires
 - o Tests d'intégration UI (front-end uniquement)
 - o Tests d'acceptation
 - o Tests de non-régression
 - o Tests de stress
 - o Tests de charge
- Android
 - o Tests unitaires
 - o Tests d'intégration UI
 - o Tests d'intégration appel à l'API
 - o Tests d'acceptation
 - o Tests de non-régression
- iPhone
 - o Tests unitaires
 - o Tests d'intégration UI
 - o Tests d'intégration appel à l'API
 - o Tests d'acceptation
 - o Tests de non-régression
- Windows Phone
 - o Tests unitaires
 - o Tests d'intégration UI
 - o Tests d'intégration appel à l'API
 - o Tests d'acceptation
 - o Tests de non-régression

4.1.2. Tests unitaires

<i>Chaque test comportera...</i>	<i>Afin de...</i>
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.

Tableau 6 : Composition d'un test unitaire

 Music Sheet Writer	Date de publication	17/11/2015	 Epitech Innovative Project
	Nom du projet	Music Sheet Writer	
	Objet du document	Stratégie de Test	
	Nom du chapitre	Livrables	

4.1.3. Tests d'acceptation et d'intégration

Chaque test comportera...	Afin de...
La fonctionnalité testée (<i>test d'acceptation uniquement</i>)	Vérifier la conformité entre le cahier des charges et le système.
Une courte description présentant le test	Connaitre la fonctionnalité testée et comprendre le contexte dans lequel se trouve le user s'il devait faire le test manuellement
Les données utilisées en tant qu'entrée pour le test	Savoir quelles valeurs sont utilisées pour le test
Les conditions à succès	Connaitre quelle et comment est la sortie du test
Le type de scénario qu'il simule (voir ci-dessous)	Connaitre la nature du test
La durée approximative du test	Connaitre le temps nécessaire à son exécution normale.

Tableau 7 : Composition d'un test d'acceptation

Le type de scénario peut être :

- Basic : c'est le flow d'exécution tel qu'il est censé se produire en condition d'utilisation normale
- Alternative : c'est le flow d'exécution tel qu'il est censé se produire en condition d'utilisateur particulière
- Exception : c'est le flow d'exécution tel qu'il est censé se produire en condition d'erreur

Le livrable des cas de tests devra être mis à jour aussitôt qu'une campagne de test est terminée d'être codé, même si tous les tests ne sont pas en succès.



4.1.4. Tests de performance

4.1.4.1. Tests de stress

Chaque test comportera...	Afin de...
Une courte description présentant le test	Connaitre l'élément testé
La cadence d'accroissement du nombre de requêtes simultanément envoyées au serveur au cours du temps.	Connaitre la vélocité du test de stress.
Le nombre de requête simultanément envoyées au début et à la fin du test	Connaitre la couverture du test.
Le temps limite du test	Connaitre le temps maximal à son exécution.

4.1.4.2. Tests de charge

Chaque test comportera...	Afin de...
Une courte description présentant le test	Connaitre l'élément testé
La durée du test	Connaitre durant laquelle le serveur sera soumis aux conditions de charge limite
Le nombre nominal de requête simultanément envoyées au cours du test	Connaitre la charge limites à laquelle est soumis le serveur
Le temps qu'il faut pour que le nombre de requêtes simultanées ait atteint le nombre nominal	Connaitre le temps nécessaire avant le début effectif du démarrage du test.

 Music Sheet Writer	Date de publication	17/11/2015	 Epitech Innovative Project
	Nom du projet	Music Sheet Writer	
	Objet du document	Stratégie de Test	
	Nom du chapitre	Livrables	

4.1.5. Tests des appels API

Chaque test comportera...	Afin de...
Une courte description présentant le test	Connaitre la fonctionnalité testée et comprendre le contexte dans lequel se trouve le user s'il devait faire le test manuellement
La route et la méthode de la requête testée	Se repérer dans le document de l'API en cas de problème
Les paramètres de la requête envoyés	Savoir quelles valeurs sont utilisées pour le test
Les conditions à succès	Connaitre les conséquences attendues sur l'application, notamment sur l'interface utilisateur.

4.2. Rapport d'exécution de tests

4.2.1. Logiciel

Les rapports d'exécution de tests apparaissent dans une console lorsqu'on lance le test. Néanmoins, la librairie QTestLib fournit la possibilité d'exporter ces résultats au format XML.

```

***** Start testing of QtQuickSampleApplicationTest *****
Config: Using QTest library 5.2.1, Qt 5.2.1
PASS : QtQuickSampleApplicationTest::initTestCase()
PASS : QtQuickSampleApplicationTest::myCalculatorViewModelUserInputDefaultValuesTest()
PASS : QtQuickSampleApplicationTest::cleanupTestCase()
Totals: 3 passed, 0 failed, 0 skipped
***** Finished testing of QtQuickSampleApplicationTest *****

```

Figure 1 : Rapport d'exécution des test unitaires du logiciel [succès]



En cas d'erreur, la fonction ayant échoué est mentionnée par un « **FAIL !** » et un message correspondant à l'erreur est indiqué ainsi que le fichier et la ligne à laquelle l'erreur s'est produite.

```

***** Start testing of QtQuickSampleApplicationTest *****
Config: Using QTest library 5.2.1, Qt 5.2.1
PASS : QtQuickSampleApplicationTest::initTestCase()
FAIL ! : QtQuickSampleApplicationTest::myCalculatorViewModelUserInputDefaultValuesTest() 'model.getUserInput() == 1' returned FALSE. (Expect the user input to be zero by default)
...\\QtQuickSampleApp\\QtQuickSampleTest\\QtQuickSampleApplicationTest.cpp(19) : failure location
PASS : QtQuickSampleApplicationTest::cleanupTestCase()
Totals: 2 passed, 1 failed, 0 skipped
***** Finished testing of QtQuickSampleApplicationTest *****

```

Figure 2 : Rapport d'exécution des tests unitaires du logiciel [échec]

 Music Sheet Writer	Date de publication	17/11/2015	 Epitech Innovative Project
	Nom du projet	Music Sheet Writer	
	Objet du document	Stratégie de Test	
	Nom du chapitre	Livrables	

4.2.2. Site Web

Tous les rapports d'exécution de tests seront fournis dans des fichiers au format texte. En cas de succès, le message « OK » apparaît.

```
$ phpunit -c app
PHPUnit 4.3.5 by Sebastian Bergmann.

Configuration read from /Users/javier/Desktop/symfony_demo_2/app/phpunit.xml.dist
Conf: .....
Time: 3.44 seconds, Memory: 49.75Mb

Time OK (17 tests, 21 assertions)
OK (17 tests, 21 assertions)
```

Figure 3 : Rapport d'exécution des tests unitaires du site web [succès]

Si un test échoue, le message « FAILURES ! » apparaît et pour chaque erreur remontée, la classe et la ligne sont affichées.

```
Time: 0 seconds, Memory: 4.25Mb

There was 1 failure:



1) Blogger\BlogBundle\Tests\Entity\BlogTest::testSlugify
Failed asserting that two strings are equal.
--- Expected
+++ Actual
@@ @@
-a day with symfony2
+a-day-with-symfony2

/var/www/html/symblog/symblog/src/Blogger/Bundle/Tests/Entity/BlogTest.php:15

FAILURES!
Tests: 1, Assertions: 2, Failures: 1.
```

Figure 4 : Rapport d'exécution des test unitaires du site web [échec]

Les résultats des tests de performances seront sous la forme de graphes comme celui-ci :

 Music Sheet Writer	Date de publication	17/11/2015	 Epitech Innovative Project
	Nom du projet	Music Sheet Writer	
	Objet du document	Stratégie de Test	
	Nom du chapitre	Livrables	

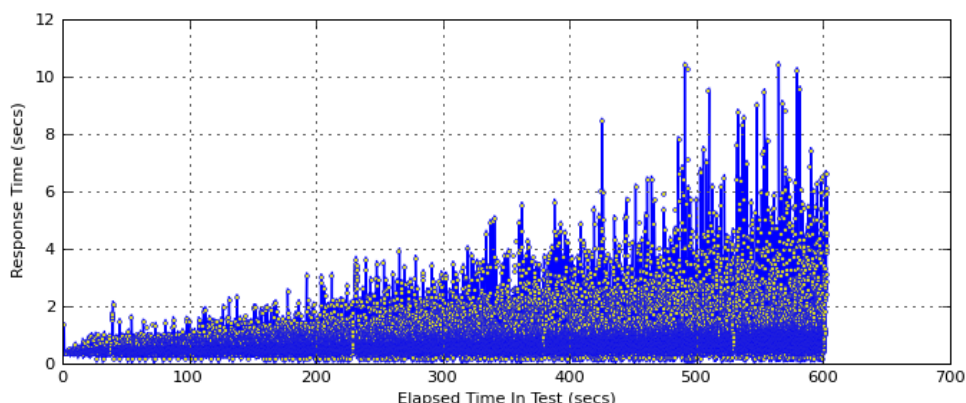


Figure 5: Rapport d'exécution des tests de performance

4.2.3. Android

Le livrable présentant le rapport d'exécution des tests pour Android sera sous la forme d'un fichier HTML. Il présentera tous les tests joués, regroupé par suite et campagne de tests. Les résultats de ces derniers seront donnés et, pour les tests ayant échoué, le fil d'exécution du test sera affiché afin de montrer exactement les raisons qui ont amené le test à échouer.

LoginActivityTest_Lo...: 1 m 24 s



8 total, 1 error, 7 passed

[Collapse](#) | [Expand](#)

msw.com.musicsheetwriter.test.login_activity.LoginActivityTest_Login

testFail_accountClosed	error	37.73 s
testFail_accountNotActivated	passed	10.45 s
testFail_badCredentials	passed	7.11 s
testFail_badPassword	passed	7.30 s
testFail_emptyField	passed	5.78 s
testFail_noNetwork	passed	6.84 s
testPreCondition	passed	801 ms
testSuccess	passed	7.80 s

Figure 6 : Rapport d'exécution des tests de l'application Android

 Music Sheet Writer	Date de publication	17/11/2015	 Epitech Innovative Project
	Nom du projet	Music Sheet Writer	
	Objet du document	Stratégie de Test	
	Nom du chapitre	Livrables	

testFail_accountClosed

error 37.73 s

```



android.support.test.espresso.NoActivityResumedException: No activities in stage RESUMED. Did you forget to launch the activity.
(test.getActivity() or similar)?
at dalvik.system.VMStack.getThreadStackTrace(Native Method)
at java.lang.Thread.getStackTrace(Thread.java:580)
at android.support.test.espresso.base.DefaultFailureHandler.getUserFriendlyError(DefaultFailureHandler.java:82)
at android.support.test.espresso.base.DefaultFailureHandler.handle(DefaultFailureHandler.java:53)
at android.support.test.espresso.ViewInteraction.runSynchronouslyOnUiThread(ViewInteraction.java:184)
at android.support.test.espresso.ViewInteraction.doPerform(ViewInteraction.java:115)
at android.support.test.espresso.ViewInteraction.perform(ViewInteraction.java:87)
at msw.com.musicsheetwriter.test.login_activity.LoginActivityTest_Login.testFail_accountClosed(LoginActivityTest_Login.java:133)
at java.lang.reflect.Method.invoke(Native Method)
at java.lang.reflect.Method.invoke(Method.java:372)
at android.test.InstrumentationTestCase.runMethod(InstrumentationTestCase.java:214)
at android.test.InstrumentationTestCase.runTest(InstrumentationTestCase.java:199)
at android.test.ActivityInstrumentationTestCase2.runTest(ActivityInstrumentationTestCase2.java:192)
at junit.framework.TestCase.runBare(TestCase.java:134)
at junit.framework.TestResult$1.protect(TestResult.java:115)
at android.support.test.internal.runner.junit3.AndroidTestResult.runProtected(AndroidTestResult.java:77)
at junit.framework.TestResult.run(TestResult.java:118)
at android.support.test.internal.runner.junit3.AndroidTestResult.run(AndroidTestResult.java:55)
at junit.framework.TestCase.run(TestCase.java:124)
at android.support.test.internal.runner.junit3.NonLeakyTestSuite$NonLeakyTest.run(NonLeakyTestSuite.java:63)
at junit.framework.TestSuite.runTest(TestSuite.java:243)
at junit.framework.TestSuite.run(TestSuite.java:238)
at android.support.test.internal.runner.junit3.DelegatingTestSuite.run(DelegatingTestSuite.java:103)
at android.support.test.internal.runner.junit3.AndroidTestSuite.run(AndroidTestSuite.java:69)
at android.support.test.internal.runner.junit3.JUnit3ClassRunner.run(JUnit3ClassRunner.java:90)
at org.junit.runners.Suite.runChild(Suite.java:128)
at org.junit.runners.Suite.runChild(Suite.java:27)
at org.junit.runners.ParentRunner$3.run(ParentRunner.java:290)
at org.junit.runners.ParentRunner$1.schedule(ParentRunner.java:71)
at org.junit.runners.ParentRunner.runChildren(ParentRunner.java:288)
at org.junit.runners.ParentRunner.access$000(ParentRunner.java:58)
at org.junit.runners.ParentRunner$2.evaluate(ParentRunner.java:268)
at org.junit.runners.ParentRunner.run(ParentRunner.java:363)
at org.junit.runner.JUnitCore.run(JUnitCore.java:137)
at org.junit.runner.JUnitCore.run(JUnitCore.java:115)
at android.support.test.internal.runner.TestExecutor.execute(TestExecutor.java:54)
at android.support.test.runner.AndroidJUnitRunner.onStart(AndroidJUnitRunner.java:240)
at android.app.Instrumentation$InstrumentationThread.run(Instrumentation.java:1873)

```

Figure 7 : Rapport d'erreur d'un test de l'application Android

4.2.4. iPhone

Tous les rapports d'exécution de tests seront fournis dans des fichiers au format texte.
Voici un exemple de rapport suite à la réussite de tous les tests :

 Music Sheet Writer	Date de publication	17/11/2015	 Epitech Innovative Project
	Nom du projet	Music Sheet Writer	
	Objet du document	Stratégie de Test	
	Nom du chapitre	Livrables	

```

▶ Building MSW/MSW2 [Debug]
▶ Check Dependencies
▶ Building Pods/KIF [Debug]
▶ Check Dependencies
▶ Building Pods/Pods-MSW2Tests [Debug]
▶ Check Dependencies
▶ Building MSW/MSW2Tests [Debug]
▶ Check Dependencies
▶ Running script 'Check Pods Manifest.lock'
▶ Running script 'Embed Pods Frameworks'
▶ Running script 'Copy Pods Resources'
All tests
Test Suite MSW2Tests.xctest started
LogTest
  ✓ test00LogFail (3.399 seconds)
  ✓ test01MotDePasseOublie (8.548 seconds)
  ✓ test02Inscription (7.422 seconds)
  ✓ test03Log (3.926 seconds)
  ✓ test04Deconnexion (5.903 seconds)
  ✓ test05ChangerDeMotPasse (14.272 seconds)
  ✓ test06ChangeFirstname (5.414 seconds)
  ✓ test07Menu (6.868 seconds)
  ✓ test08Swipe (5.708 seconds)
  ✓ test09SwipeMenu (2.007 seconds)
  ✓ test10Ajouter (5.572 seconds)
  ✓ test11 (0.000 seconds)
LoginTest
  ✓ testConnexion (2.451 seconds)
  ✓ testConnexionfausse (1.171 seconds)
  ✓ testGetMethodWithId (0.687 seconds)
  ✓ testGetMethodWithString (0.704 seconds)
  ✓ testOptionsMethodWithString (0.678 seconds)
  ✓ testPostMethodForSubscribeWithUsername (0.778 seconds)
  ✓ testPostMethodWithIdentifier (1.185 seconds)
  ✓ testPostMethodWithString (1.170 seconds)
  ✓ testPutMethodWithString (0.740 seconds)
testApiMethod
  ✓ testGetMethodWithId (0.697 seconds)
  ✓ testPostMethodForSubscribeWithUsername (0.707 seconds)
  ✓ testPostMethodWithIdentifier (1.173 seconds)
  ✓ testPostMethodWithString (1.170 seconds)|



```

Executed 25 tests, with 0 failures (0 unexpected) in 82.353 (82.365) seconds

Figure 8 : Rapport d'exécution des tests de l'application iPhone

4.2.5. Windows Phone

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 du 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 ou échoué.

 Music Sheet Writer	Date de publication	17/11/2015	 Epitech Innovative Project
	Nom du projet	Music Sheet Writer	
	Objet du document	Stratégie de Test	
	Nom du chapitre	Livrables	

Le livrable présentant le rapport d'exécution des tests pour l'application Windows phone sera sous la forme d'un fichier RTF.

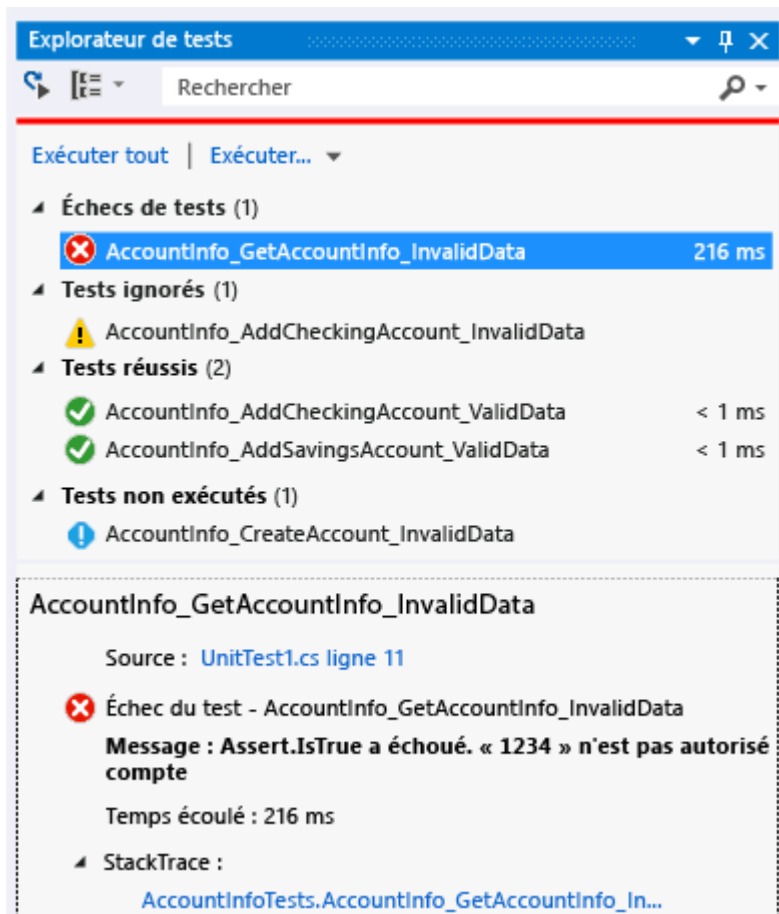




Figure 9 : Fenêtre de test sous Visual Studio

4.3. 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 connaître la qualité du code en visualisant les tests effectués. Le chef de groupe pourra également s'appuyer sur les tests d'acceptation qui lui permettront de vérifier la conformité de l'application développée avec le cahier des charges initial.

Ce document devra contenir :

- Les résultats de test de non-régression à une date donnée en gardant la trace des précédents tests effectués afin d'avoir une estimation de la qualité du code dans le temps.
- Les résultats des tests d'acceptation afin de valider les fonctionnalités implémentées et les fonctionnalités restantes.
- Les graphiques du logiciel de suivi de problèmes afin d'évaluer les efforts mis dans la correction des bugs.

 Music Sheet Writer	Date de publication	17/11/2015	 Epitech Innovative Project
	Nom du projet	Music Sheet Writer	
	Objet du document	Stratégie de Test	
	Nom du chapitre	Types de tests	

5. Types de tests

5.1. Logiciel

5.1.1. Tests Unitaire

5.1.1.1. Objectif

Il s'agit pour le programmeur de tester une fonction, indépendamment du reste du programme, ceci afin de s'assurer qu'elle répond aux spécifications fonctionnelles et qu'elle fonctionne correctement en toutes circonstances.

5.1.1.2. Environnement et conditions de réalisation

Les tests unitaires peuvent être joués à n'importe quel moment du développement et cela sur toute nouvelle fonction implémentée au sein du projet. Ces tests seront réalisés à l'aide de QT (sous version Windows et Mac OS) qui possède une librairie spécialisée pour les tests : QTestLib.

```
#include <QtTest/QtTest>

class TestQString: public QObject
{
    Q_OBJECT
private slots:
    void toUpper();
};

void TestQString::toUpper()
{
    QString str = "Hello";
    QCOMPARE(str.toUpper(), QString("HELLO"));
}
```

Figure 10 : Exemple de test unitaire avec QTestLib

5.1.1.3. Configurations particulières

Une fois notre classe créée et les fonctions qu'on souhaite tester implémentées, il suffit d'ajouter à la fin du fichier QTEST_MAIN(NomDeLaClasseTest). Cela va créer un exécutable lors de la compilation, qui une fois lancé va exécuter les tests dans l'ordre dans lequel ils sont définis.



```
class TestQString: public QObject { ... };
QTEST_MAIN(TestQString)
```

Figure 11 : Ajout d'un test dans la configuration d'exécution avec QT

Pour la compilation, il faut indiquer au compilateur qu'on utilise la librairie QTestLib.

```
/myTestDirectory$ qmake -project "CONFIG += qtestlib"
/myTestDirectory$ qmake
/myTestDirectory$ make
```

Figure 12 : Compilation des tests avec QT

 Music Sheet Writer	Date de publication	17/11/2015	 Epitech Innovative Project
	Nom du projet	Music Sheet Writer	
	Objet du document	Stratégie de Test	
	Nom du chapitre	Types de tests	

5.1.1.4. Planning et charge

Les tests unitaires peuvent être écrits à la suite de la spécification de la ou des classes qu'ils testent. La charge de travail dépend de ce qui est testé.

5.1.1.5. Critère de démarrage des tests

Le logiciel doit être compilé.

5.1.1.6. Critère de passage/échec

Les critères de passage/échec sont définis par la comparaison entre un résultat attendu et le résultat retourné par la fonction testée. Si une erreur est détectée, celle-ci est retournée et le test s'arrête.

```

***** Start testing of TestQString *****
Config: Using QTest library %VERSION%, Qt %VERSION%
PASS  : TestQString::initTestCase()
PASS  : TestQString::toUpper()
PASS  : TestQString::cleanupTestCase()
Totals: 3 passed, 0 failed, 0 skipped
***** Finished testing of TestQString *****

```

Figure 13 : Exemple de test unitaire avec QT [succès]

5.1.2. Tests d'intégration UI

5.1.2.1. Objectif

Ces tests ont pour but de vérifier le bon déroulement des interactions entre l'utilisateur et l'interface utilisateur du logiciel.

5.1.2.2. Environnement et conditions de réalisation

Ces tests interviennent dès qu'on ajoute un élément, tel qu'un bouton ou un champ texte, à l'interface utilisateur. Comme pour les tests unitaires, on se sert la librairie QTestLib qui permet notamment de simuler les actions utilisateurs.

5.1.2.3. Configurations particulières

[Voir partie 5.1.1.3.](#)

5.1.2.4. Planning et charge



Les tests unitaires peuvent être écrits à la suite de la spécification de la ou des classes qu'ils testent. La charge de travail dépend de ce qui est testé.

5.1.2.5. Critère de démarrage des tests

Le logiciel doit être compilé.

5.1.2.6. Critères de passage/échec

[Voir partie 5.1.1.6.](#)

 Music Sheet Writer	Date de publication	17/11/2015	 Epitech Innovative Project
	Nom du projet	Music Sheet Writer	
	Objet du document	Stratégie de Test	
	Nom du chapitre	<i>Types de tests</i>	

5.2. Site Web

5.2.1. Tests unitaires

5.2.1.1. Objectif

Le but de ces tests va être de comparer la valeur de retour d'une fonction avec la valeur attendue. Cela permet de retrouver facilement le moment où le programme ne fonctionne pas comme prévu et de trouver le problème rapidement.

5.2.1.2. Environnement et conditions de réalisation

Les tests unitaires pour le site web seront exécutés sur la partie interface utilisateur et sur l'API. Avant la mise en production les tests seront effectués premièrement en environnement de développement puis une fois déposés sur le SVN, ils seront testés en environnement de production pour vérifier que le comportement soit bien le même.

En environnement de développement, les serveurs de tests ne seront pas configurés dans des conditions optimales. Les serveurs n'ont pas besoin de configuration de sécurité ou de performance particulières. Le serveur de développement est moins performant que celui utilisé en production.

5.2.1.3. Planning et charge

Les tests unitaires peuvent être écrits à la suite de la spécification de la ou des classes qu'ils testent. La charge de travail dépend de ce qui est testé.

5.2.1.4. Critère de démarrage des tests

Pour démarrer les tests, les serveurs web, de base de données et de messagerie doivent être fonctionnels et bien configurés.

5.2.1.5. Critères de passage/échec

Les critères de passage/échec sont définis par la comparaison entre un résultat attendu et le résultat retourné par la fonction testée.

5.2.2. Tests de performance

5.2.2.1. Objectif

Les tests de performances ont pour but de vérifier les performances sur l'ensemble des services de l'application pour s'assurer d'un temps de réponse cohérent entre le serveur et l'utilisateur.

5.2.2.2. Environnement et conditions de réalisation

Ces tests sont réalisés uniquement en environnement de production. Dans des conditions d'utilisation extrême pour pouvoir connaître les limites de l'API et du site web sur les serveurs de productions.



5.2.2.3. Configurations particulières

Les configurations pour les tests de performances seront définies dans des fichiers XML utilisés par Pylot pour écrire les tests.

5.2.2.4. Planning et charge

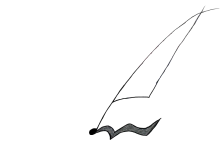

5.2.2.5. Critère de démarrage des tests

Pour démarrer les tests l'application doit être configurée pour être en mode de production.

 Music Sheet Writer	Date de publication	17/11/2015	 Epitech Innovative Project
	Nom du projet	Music Sheet Writer	
	Objet du document	Stratégie de Test	
	Nom du chapitre	<i>Types de tests</i>	

5.2.2.6. Critères de passage/échec

Les critères de passage à échec sont définis par plusieurs facteurs qui prennent en compte le temps de réponse, la complexité de la requête et la charge actuelle du serveur.

 Music Sheet Writer	Date de publication	17/11/2015	 Epitech Innovative Project
	Nom du projet	Music Sheet Writer	
	Objet du document	Stratégie de Test	
	Nom du chapitre	Types de tests	

5.3. Android

5.3.1. Tests unitaires

5.3.1.1. Objectif

L'objectif de ces tests est de vérifier le bon fonctionnement de certaines classes ou méthodes particulières appelées au sein de l'application Android. Ces tests concernent exclusivement les fonctions n'utilisant pas de contexte spécifique à l'application Android Music Sheet Writer pour fonctionner. C'est-à-dire qu'ils n'utilisent aucun élément visuel. Ces tests sont la plus basse couche de test à réaliser sur l'application Android et ne nécessitent pas de contexte Android pour fonctionner.

5.3.1.2. 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 java représentant les suites de tests, chacune comportant des méthodes représentant les cas de tests à exécuter.

Ces tests pourront couvrir de simples classes (validateur d'email, validateur de mot de passe, etc.) ou une fonctionnalité plus complexe comprenant un ensemble de classe (expéditeur de requête HTTP).

5.3.1.3. Configurations particulières

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

En outre, Android Studio facilite grandement la manipulation car il gère lui-même le lancement de l'application et des tests. Il suffit d'ajouter une nouvelle configuration de lancement de test. Cette dernière permet de sélectionner les tests à lancer et définir un périphérique cible.

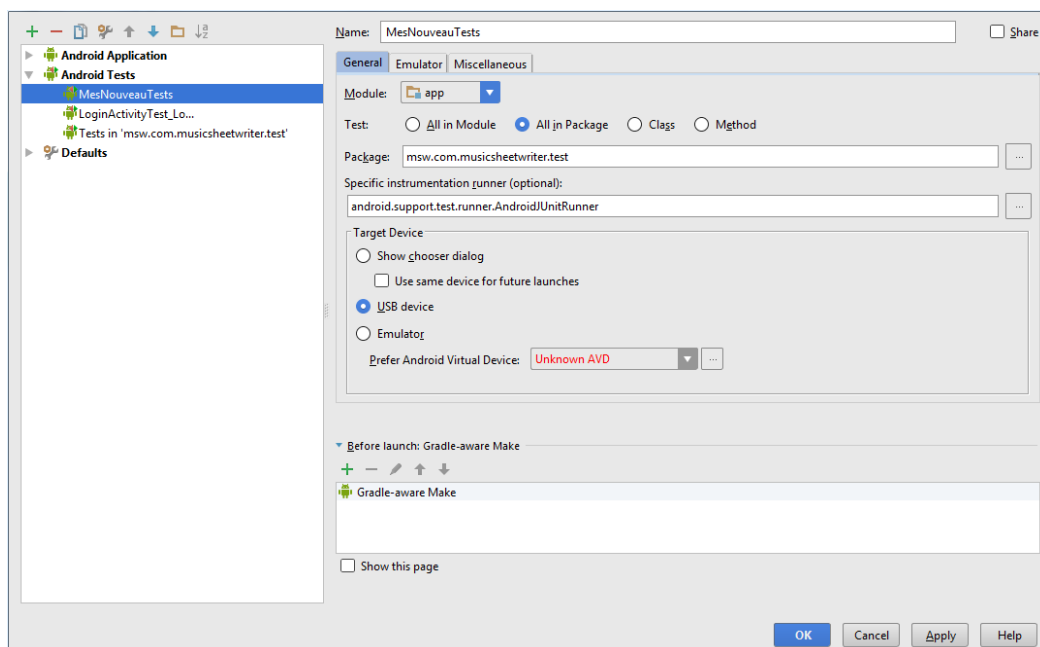




Figure 14 : Définition d'une configuration de lancement de tests sur Android Studio

5.3.1.4. Planning et charge

Les tests unitaires peuvent être écrits à la suite de la spécification de la ou des classes qu'ils testent. La charge de travail dépend de ce qui est testé.

 Music Sheet Writer	Date de publication	17/11/2015	 Epitech Innovative Project
	Nom du projet	Music Sheet Writer	
	Objet du document	Stratégie de Test	
	Nom du chapitre	<i>Types de tests</i>	

5.3.1.5. Critère de démarrage des tests

L'application doit être installée sur le périphérique cible.

5.3.1.6. Critères de passage/échec

Les critères de passage/échec sont définis par la comparaison entre un résultat attendu et le résultat retourné par la fonction testée.

5.3.2. Test d'intégration UI

5.3.2.1. Objectif

L'objectif de ces tests est de vérifier le bon fonctionnement de la capture des événements graphiques et les conséquences qu'elles engendrent sur l'interface utilisateur. 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 indépendantes les unes des autres. En effet, chaque activité ou fragment aura une ou plusieurs classes de test associées.

5.3.2.2. 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 ou d'un fragment sont spécifiés. Il sera sous la forme d'une ou plusieurs classes java représentant les suites de tests, chacune comportant des méthodes représentant les cas de tests à exécuter.

5.3.2.3. Configurations particulières

[Voir la partie 5.2.1.3.](#)

5.3.2.4. Planning et charge

Les tests d'intégration UI seront écrits à la suite de la spécification de la ou des classes qu'ils testent. La charge de travail dépend de ce qui est testé.

5.3.2.5. Critère de démarrage des tests

L'application doit être installée sur le périphérique cible.

5.3.2.6. Critères de passage/échec

Les critères de passage/échec sont définis par la comparaison entre un résultat attendu et le résultat retourné par la fonction testée.



5.3.3. Test d'intégration des appels API

5.3.3.1. Objectif

L'objectif de ces tests est de vérifier l'envoi des différentes requêtes à l'API suite aux actions utilisateur ainsi que l'impact des réponses sur l'interface graphique. Ces tests concernent les classes chargées de l'affichage graphique (c'est-à-dire les activités et les fragments) et envoyant des requêtes à l'API. De telles classes seront testées de manière indépendantes les unes des autres. En effet, chaque classe de test n'aura qu'une activité ou fragment associé.

5.3.3.2. Environnement et conditions de réalisation

Ces tests pourront être joués à n'importe quel moment du développement. Le test sera réalisé aussitôt que les requêtes à envoyer et que leur impact sur l'interface graphique sont spécifiés pour une activité ou un fragment. Il sera sous la forme d'une ou plusieurs classes java représentant les suites de tests,

 Music Sheet Writer	Date de publication	17/11/2015	 Epitech Innovative Project
	Nom du projet	Music Sheet Writer	
	Objet du document	Stratégie de Test	
	Nom du chapitre	<i>Types de tests</i>	

chacune comportant des méthodes représentant les cas de tests à exécuter. Pour de tels tests, chaque classe de test vérifiera une URL d'API pour une seule activité ou un seul fragment.

Pour chaque requête, les tests devront comprendre les cas où l'API renvoie une réponse valide et les cas où l'API renvoie une erreur.

5.3.3.3. Configurations particulières

[Voir la partie 5.3.1.3](#)

5.3.3.4. Planning et charge



Les tests d'appel à l'API peuvent être écrits à la suite de la spécification de la ou des classes qu'ils testent. La charge de travail dépend de ce qui est testé.

5.3.3.5. Critère de démarrage des tests

L'application doit être installée sur le périphérique cible et l'API doit être fonctionnelle.

5.3.3.6. Critères de passage/échec

Les critères de passage/échec sont définis par la comparaison entre un résultat attendu et le résultat retourné par la fonction testée.

 Music Sheet Writer	Date de publication	17/11/2015	 Epitech Innovative Project
	Nom du projet	Music Sheet Writer	
	Objet du document	Stratégie de Test	
	Nom du chapitre	<i>Types de tests</i>	

5.4. iPhone

5.4.1. Tests unitaires

5.4.1.1. 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.

5.4.1.2. Environnement et conditions de réalisation

Les tests unitaires peuvent être joués à n'importe quel moment du développement et cela sur toute nouvelle fonction implémentée au sein du projet.

5.4.1.3. Configurations particulières

Aucune configuration particulière n'est requise pour ce test.

5.4.1.4. Planning et charge

Les tests unitaires peuvent être écrits à la suite de la spécification de la ou des classes qu'ils testent. La charge de travail dépend de ce qui est testé.

5.4.1.5. Critères de passage/échec

Les critères de passage/échec sont définis par la comparaison entre un résultat attendu et le résultat retourné par la fonction testée.

5.4.2. Tests d'intégration UI

5.4.2.1. Objectif

L'objectif de ce test est de vérifier la partie interface de l'application. Certaines parties n'étant pas détectables via le code, la création de *mocks* est donc la meilleure solution. Ces tests serviront à mettre en évidence les problèmes liés à l'interface et par la même occasion de tester à nouveau une partie des fonctionnalités de l'application.

5.4.2.2. Environnement et conditions de réalisation

Ces tests interviennent dès qu'on ajoute un élément, tel qu'un bouton ou un champ texte, à l'interface utilisateur.

Tous les tests se feront via *Xcode* qui contient une interface claire de test et seront aussi lancés grâce à *xcodebuilder* pour permettre un export des tests.

5.4.2.3. Configurations particulières



Aucune configuration particulière n'est requise. Cependant, le framework de test fixe une contrainte de temps pour la réalisation des tests.

5.4.2.4. Planning et charge

Les tests d'intégration UI peuvent être écrits à la suite de la spécification de la ou des classes qu'ils testent. La charge de travail dépend de ce qui est testé.



5.4.2.5. Critère de démarrage des tests

Pour démarrer ce test l'application doit être fermée et compilée sur le téléphone ou simulateur.

 Music Sheet Writer	Date de publication	17/11/2015	 Epitech Innovative Project
	Nom du projet	Music Sheet Writer	
	Objet du document	Stratégie de Test	
	Nom du chapitre	<i>Types de tests</i>	

5.4.2.6. Critères de passage/échec

Les critères de passage/échec sont aussi définis par la comparaison entre un résultat attendu et le résultat retourné par la fonction testée. Ces critères prennent aussi en compte les contraintes fixées par le framework de test.

 Music Sheet Writer	Date de publication	17/11/2015	 Epitech Innovative Project
	Nom du projet	Music Sheet Writer	
	Objet du document	Stratégie de Test	
	Nom du chapitre	<i>Types de tests</i>	

5.5. Windows Phone

5.5.1. Tests unitaires

5.5.1.1. 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.

5.5.1.2. 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 tests, chacune comportant des méthodes représentant les cas de tests à exécuter.

5.5.1.3. Configurations particulières

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

5.5.1.4. Planning et charge

Les tests unitaires peuvent être écrits à la suite de la spécification de la ou des classes qu'ils testent. La charge de travail dépend de ce qui est testé.

5.5.1.5. Critère de démarrage des tests

L'application doit être installée sur le périphérique cible.

5.5.1.6. Critères de passage/échec

Les critères de passage/échec sont définis par la comparaison entre un résultat attendu et le résultat retourné par la fonction testée.

5.5.2. Tests d'intégration UI

5.5.2.1. Objectif

L'objectif de ces tests est de vérifier le bon fonctionnement de la capture des événements graphiques et les conséquences qu'elles engendrent sur l'interface utilisateur. Ces tests concernent les classes chargées de l'affichage graphique.

5.5.2.2. 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. Il sera sous la forme d'une ou plusieurs classes C# représentant les suites de tests, chacune comportant des méthodes représentant les cas de tests à exécuter.

5.5.2.3. Configurations particulières



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

5.5.2.4. Planning et charge

Les tests d'intégration UI seront écrits à la suite de la spécification de la ou des classes qu'ils testent. La charge de travail dépend de ce qui est testé.

5.5.2.5. Critère de démarrage des tests

L'application doit être installée sur le périphérique cible.

 Music Sheet Writer	Date de publication	17/11/2015	 Epitech Innovative Project
	Nom du projet	Music Sheet Writer	
	Objet du document	Stratégie de Test	
	Nom du chapitre	<i>Types de tests</i>	

5.5.2.6. Critères de passage/échec

Les critères de passage/échec sont définis par la comparaison entre un résultat attendu et le résultat retourné par la fonction testée.

5.5.3. Tests d'acceptation

5.5.3.1. Objectif

Les tests d'acceptation 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 d'acceptation auront une couverture complète de l'application.

5.5.3.2. Environnement et conditions de réalisation

Les tests d'acceptation 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.

5.5.3.3. Configurations particulières

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

5.5.3.4. 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.

5.5.3.5. Critère de démarrage des tests

L'application doit être installée sur le périphérique cible.

5.5.3.6. 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.

 Music Sheet Writer	Date de publication	17/11/2015	 Epitech Innovative Project
	Nom du projet	Music Sheet Writer	
	Objet du document	Stratégie de Test	
	Nom du chapitre	<i>Outils</i>	

6. Outils

6.1. 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 projet de test dans lequel sont mis tous les tests relatifs au livrable.

6.2. Gestion de rejets/erreurs

Nous utiliserons le logiciel de suivi de problèmes [LeanTesting](#) 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 « bug tracker ».

6.3. Autres outils

6.3.1. Outils d'automatisation des tests

6.3.1.1. Logiciel

L'automatisation des tests sous QT se fait à l'aide du framework [QTest](#) qui permet notamment le lancement automatique des tests mais également fournit une liste de fonctions permettant de simuler les actions d'un utilisateur.

6.3.1.2. Android

Pour l'automatisation de test, Android Studio intègre nativement le framework [JUnit](#). Il gère le lancement automatique des tests.

6.3.1.3. iPhone

L'IDE *Xcode* gère l'automatisation des tests.

6.3.1.4. Site internet

Pour effectuer les tests sur le site web aussi bien sur la partie 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.

6.3.2. Outils pour les tests de performances

6.3.2.1. Site Web

Pyload est un outil gratuit et open source pour tester les performances et l'évolutivité des services web.

6.3.3. Outils pour les tests d'intégration UI

6.3.3.1. Android

Le framework [Espresso](#) est utilisé. Il fournit une bibliothèque de fonctions permettant de simuler les actions utilisateur mais également de vérifier l'état des éléments graphiques.

6.3.3.2. iPhone

C'est le framework KIF qui sera utilisé pour l'automatisation des tests d'intégration UI.