

simon.daguenet@epitech.eu

Music Sheet Writer

Music Sheet Writer

[Sujet du document]

Objectifs du document

Résumé

Ce document est le Template des documents pour le projet MusicSheetWriter. Il comprend les éléments de base à rendre au Lab EIP : Résumé, Glossaire, Description du document, Sommaire ; et définie également les styles utilisé pour la présentation du document.

Glossaire

***– M –***

Mot : Définition

***– S –***

Signification de l’Acronyme (SDA) : Définition

Description du document

|  |  |
| --- | --- |
| Titre | Music Sheet Writer : [Sujet du document] |
| Date de création | [Date de création] |
| Date de publication | [Date de publication] |
| Auteur | simon.daguenet@epitech.eu |
| Responsable | Jonathan Racaud |
| E-mail | musicsheetwriter\_2017@labeip.epitech.eu |
| Sujet | [Sujet du document] |
| Version du modèle |  |

Tableau des révisions

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Auteur | Section(s) | Commentaire |
|  |  |  |  |

Sommaire

[1. Partie 1 2](#_Toc434592146)

[1.1. Sous-partie 1.1 2](#_Toc434592147)

[1.2. Sous-partie 1.2 2](#_Toc434592148)

[1.3. Sous-partie 1.3 2](#_Toc434592149)

[2. Partie 2 2](#_Toc434592150)

Liste des Tableaux

[Tableau 1: Tableau en Row-Heading 2](#_Toc434592579)

[Tableau 2: Tableau en Column-Heading 2](#_Toc434592580)

[Tableau 3: Tableau en Double-heading 2](#_Toc434592581)

Liste des Figures

[Figure 1: Nom de l'image 2](#_Toc434592447)

# 

# Types de tests et périmètre

## Android

### Test unitaire

Ces tests vérifieront le bon fonctionnement de classes ou méthode particulière. Ils seront effectués hors d’un contexte spécifique à l’application Android 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.

### Test d’intégration

Ces tests vérifient le bon fonctionnement des classes et méthode couplé avec l’Interface utilisateur de l’application. Les activités et fragments concernés par ces tests seront testés indépendamment les uns des autres. Ces tests seront de deux types différents :

#### Test composant 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 d’appel à API

Les tests d’appel à API vérifieront le bon fonctionnement des actions nécessitant ou incluant l’envoie de requête à l’API MusicSheetWriter suite à une action utilisateur particulière ainsi que les conséquences sur l’interface utilisateurs de chacune des réponses possible renvoyée par le serveur. Chaque activité et fragment contenant un appel à l’API sera soumis à des tests d’appel à l’API.

### Test de non-régression

Ces tests seront établis à partir des tests unitaires et d’intégration et permettront de vérifier l’intégration d’une nouvelle partie du code ou d’une modification du code existant en s’assurant que les fonctionnalités de l’application sont toujours valides.

### Test d’acceptation

Ces tests vérifieront le bon fonctionnement de certains scénario prédéfinit et couvrant l’ensemble des fonctionnalités de l’application. A la différence des tests des tests d’intégrations, ces tests vérifieront l’application dans son ensemble, et pas les activités et fragments de manière unitaire.

## IPhone

### Test unitaire

Ces tests vérifieront le bon fonctionnement de classes ou méthode particulière. Ils seront effectués hors d’un contexte spécifique à l’application IOS 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 API, les classes de validation de formulaire ou encore d’authentification.

Les tests d’appel à API vérifieront le bon fonctionnement des actions nécessitant ou incluant l’envoie de requête à l’API MusicSheetWriter suite à une action utilisateur particulière ainsi que les conséquences sur l’interface utilisateurs de chacune des réponses possible renvoyée par le serveur. Chaque activité et fragment contenant un appel à l’API sera soumis à des tests d’appel à l’API. C’est test ne vérifierons pas le contenu de la réponse de l’API mais seulement

### Test d’intégration

Ces tests vérifient le bon fonctionnement des classes et méthode couplé avec l’Interface utilisateur de l’application. Les activités et fragments concernés par ces tests seront testés indépendamment les uns des autres. Ces tests seront de deux types différents :

#### Test composant UI

Ils sont essentiels pour vérifier certaines fonctions de l’application qui ne peuvent pas être vérifiées autrement que par des actions de l’utilisateur.

Chaque test et construit pour tester chaque partie de l’application, comme les menus ou le changement de code.

Des *Mocks* sont utilisés pour les tests de l’interface. Ce sont des objets permettant de simuler les actions d’un utilisateur. Certaines erreurs ne venant pas du code mais du *Main.Storyboard* (c’est le fichier qui permet la création et le management de toute l’interface d’une application IPhone si l’on utilise Xcode) ne peuvent être visibles que par cette méthode. Chaque test est écris après la création de l’interface. Ces tests sont donc là pour vérifier la bonne conduite de l’interface malgré une maintenance.

### Test de non-régression

Ces tests seront établis à partir des tests unitaires et d’intégration et permettront de vérifier l’intégration d’une nouvelle partie du code ou d’une modification du code existant en s’assurant que les fonctionnalités de l’application sont toujours valides.

### Test d’acceptation

Ces tests vérifieront le bon fonctionnement de certains scénario prédéfinit et couvrant l’ensemble des fonctionnalités de l’application. A la différence des tests des tests d’intégrations, ces tests vérifieront l’application dans son ensemble, et pas les activités et fragments de manière unitaire.

## 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.

## Logiciel

### Test composant 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, etc.).

### 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](https://fr.wikipedia.org/wiki/Sp%C3%A9cification_(informatique)) et qu'elle fonctionne correctement en toutes circonstances. Ces tests seront effectués sur l’ensemble des fonctions qui sont utilisées dans le logiciel.

### Tests fonctionnels

Ces tests ont pour but de valider un dispositif complet et non pas une simple conversion d'entrées-sorties. C’est un ensemble de tests unitaires ayant pour but de valider le bon déroulement d’une fonctionnalité du logiciel, comme par exemple la création d’un projet ou l’édition d’une partition.

### Tests de non-régression

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.

## Site web

### Tests Unitaires

Les tests unitaires seront effectués sur l’ensemble des fonctions de chaque classe de service pour permettre de vérifier le fonctionnement de tous les services de l’application. C’est test vont permettre directement de vérifier que les fonctions de bas niveau fonctionnent parfaitement en évaluant chaque valeurs de retour pour l’ensemble des cas possibles sur une opération.

### Tests fonctionnels

Les tests fonctionnels sont un ensemble de tests unitaire qui vont permettre de vérifier le bon déroulement d’une fonctionnalité de l’application comme par exemple l’inscription d’une personne sur le site internet. Ces tests permettent entre autre la validation d’une version et d’organiser les tests de telles manière qu’il soit plus compréhensible par le développeur.