



MUSIC SHEET WRITER DOCUMENTATION TECHNIQUE



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

MUSIC SHEET WRITER

 Music Sheet Writer	Date de publication	10/07/2016	 Epitech Innovative Project
	Nom du projet	Music Sheet Writer	
	Objet du document	Documentation Technique	
	Nom du chapitre	<i>Objectifs du document</i>	

Objectifs du document

Résumé

Ce document est un guide du développeur et se destine à toute personne intégrant le projet Music Sheet Writer.



Le projet consiste en un logiciel d'édition de partition permettant la génération de partition depuis une guitare ou un piano branché sur l'ordinateur de l'utilisateur. Il est accompagné d'un site internet et de trois applications mobiles Android, iOS et Windows Phone.

Le document se décompose en plusieurs section ayant chacune un but précis. La première section revient sur le contexte de l'EIP et du projet, elle permet donc de comprendre les tenant et aboutissant du projet. La section 2 décrit de manière général le projet et les différents livrables qui le compose.

La section 3 quant à elle, explique plus dans le détail chacun des livrables du projet (Logiciel, Site Web, Application iOS, Application Android et Application Windows Universal App) et décrit les normes de codage utilisées, l'architecture des livrable et leurs implémentations.

L'avant dernière et dernière section reviennent respectivement sur la stratégie de version/release des différent livrables et sur les outils collaboratifs utilisé au sein de l'équipe. Qu'ils soient d'ordre non technique (outils de communication) ou technique (gestionnaire des versions du code source).

Il n'est pas nécessaire de lire le document de manière linéaire, mais il est préférable de lire la section 1 et 2 en premier afin d'avoir une idée claire du projet et de son contexte. Ce qui aidera tout nouveau développeur d'intégrer au mieux l'équipe.

 Music Sheet Writer	Date de publication	10/07/2016	 Epitech Innovative Project
	Nom du projet	Music Sheet Writer	
	Objet du document	Documentation Technique	
	Nom du chapitre	<i>Objectifs du document</i>	

Glossaire

– B –

Bundle : Composant logique et indépendant dans un projet Symfony.

– E –

Echantillonnage (analyse de signal) : prélèvement des valeurs d'un signal à intervalle régulier.

– F –

Fenêtrage (analyse de signal) : procédé permettant d'obtenir qu'une partie limitée d'un signal.

FFT (Fast Fourier Transform) : Algorithme permettant d'obtenir le spectre fréquentielle d'un signal.

– G –

Gradle : Système de génération d'exécutable à partir d'un code source et d'un fichier de configuration de projet.

– H –

Hardcode : « Mettre en dur » en français, représente l'action d'insérer directement dans le code sources des valeurs de configuration ou provenant de sources extérieures.

– M –

Microsoft .NET : Microsoft .NET est le nom donné à un ensemble de produits et de technologies de l'entreprise Microsoft.



MVVM (Modèle-vue-vue modèle) : Le MVVM est une architecture et une méthode de conception utilisée dans le développement des applications basées sur les technologies Windows.

– 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).

– S –

Silverlight : Silverlight aide au développement d'application par l'apport d'outils de développement et des langages Microsoft .NET.



 Music Sheet Writer	Date de publication	10/07/2016	 Epitech Innovative Project
	Nom du projet	Music Sheet Writer	
	Objet du document	Documentation Technique	
	Nom du chapitre	<i>Description du document</i>	

Description du document

Titre	Music Sheet Writer : Documentation Technique
Date de création	26/06/2016
Date de publication	10/07/2016
Auteur	J. Racaud; A. Simon; J. Harrault; J. Blondeel; S. Daguenet; F. Corradin
Responsable	Jonathan Racaud
E-mail	musicsheetwriter_2017@labeip.epitech.eu
Sujet	Documentation Technique
Version du document	3.0
Version du modèle	1.1



Tableau des révisions

Date	Auteur	Section(s)	Commentaire
04/12/2017	Jonathan Racaud	Toutes	Création du document à partir de la version 2.0
09/12/2016	Simon Daguenet	3.5	Modification de la partie Windows phone 8.1 à Windows 10 UWP
09/12/2016	Jonathan Racaud	3.5	Correction orthographe et grammaire. Ajout des légendes des images. Mise à jour Sommaire et Liste des figures.
11/12/2017	Jeremy Harrault	3.1 3.3 4.2	Mise à jour de l'architecture du logiciel Ajout des normes de codage pour Android Mise à jour des release notes
17/12/2016	Jonathan Racaud	Toutes	Relecture et correction orthographique Ajout section 3.2.1 Modification résumé

 Music Sheet Writer	Date de publication	10/07/2016	 Epitech Innovative Project
	Nom du projet	Music Sheet Writer	
	Objet du document	Documentation Technique	
	Nom du chapitre	<i>Sommaire</i>	

Sommaire

1. Introduction	1
1.1. Objectif de l'EIP et Epitech	1
1.2. Principe de base du système	1
2. Le projet	2
3. Les livrables	3
3.1. Logiciel	3
3.2. Site Web	6
3.3. Application Android	12
3.4. Application iOS	18
3.1. Application Windows Phone	22
4. Versions/Releases	25
4.1. Numérotation des versions	25
4.2. Release note	25
5. Outils de collaborations	27
5.1. Outils de développement collaboratif	27
5.2. Outils de communication	27
6. Annexes	28



 Music Sheet Writer	Date de publication	10/07/2016	 Epitech Innovative Project
	Nom du projet	Music Sheet Writer	
	Objet du document	Documentation Technique	
	Nom du chapitre	<i>Liste des Tableaux</i>	

Liste des Tableaux

Tableau 1. Présentation des composants - Android	14
Tableau 2 : Plan de release du logiciel	25
Tableau 3 : Plan de release du site web	26
Tableau 4 : Plan de release des applications mobiles	26

Liste des Figures

Figure 1. Architecture de Music Sheet Writer.....	2
Figure 2. Architecture du logiciel	3
Figure 3. Architecture Site Web	6
Figure 4. Arborescence projet Symphony	6
Figure 5. Architecture AppBundle	8
Figure 6. MSW Bundle – organisation	9
Figure 7. Implémentation User - MSWBundle	10
Figure 8. Réponse API Rest - User	10
Figure 9. Arborescence dossier Resources – MSWBundle	11
Figure 10. Exemples noms de classes – Android	12
Figure 11. Arborescence des ressources - Android	13
Figure 12. Hiérarchisation des fichiers gradle - Android	13
Figure 13. Structure des composants - Android	14
Figure 14. Déclaration d'une configuration de lancement dans Android Studio	15
Figure 15. Liste des bibliothèques à inclure dans le fichier gradle pour l'application Android	16
Figure 16. Arborescence des classes de test - Android	16
Figure 17. Déclaration d'une configuration de lancement de tests dans Android Studio	16
Figure 18. Dossier Xcasset - iOS.....	18
Figure 19. Lancement application - iOS.....	18
Figure 20. Création vue - iOS	20
Figure 21. Création embed - iOS	20
Figure 22. Mode assistant	21
Figure 23. Schématisation MVVM	22
Figure 24. Organisation code source projet Music Sheet Writer	22
Figure 25 Gestion des images via le dossier Assets.....	23
Figure 26 Choix de l'émulateur ou du téléphone.....	24
Figure 27 Choix de la configuration et de la plateforme	24

 Music Sheet Writer	Date de publication	10/07/2016	 Epitech Innovative Project
	Nom du projet	Music Sheet Writer	
	Objet du document	Documentation Technique	
	Nom du chapitre	<i>Introduction</i>	

1. Introduction

Ce projet est un projet étudiant réalisé dans le cadre des études à Epitech en tant que projet de fin d'étude.

1.1. Objectif de l'EIP et Epitech

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 place 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 composant de la musique. Il se présente comme tout logiciel d'édition de partition existant, mais apporte une fonctionnalité majeure : la génération d'une partition depuis un piano ou une guitare branché 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. Laissant l'utilisateur se concentrer sur la musique avant son écriture.

Music Sheet Writer s'accompagne aussi d'applications mobiles disponibles sur Android, iOS et Windows Phone, ainsi que d'un site internet.

 Music Sheet Writer	Date de publication	10/07/2016	 Epitech Innovative Project
	Nom du projet	Music Sheet Writer	
	Objet du document	Documentation Technique	
	Nom du chapitre	<i>Le projet</i>	

2. Le projet

Le projet Music Sheet Writer est composé de plusieurs sous-projets ou livrables ayant tous un lien entre eux.

Le diagramme suivant présente les différents livrables et leurs relations.

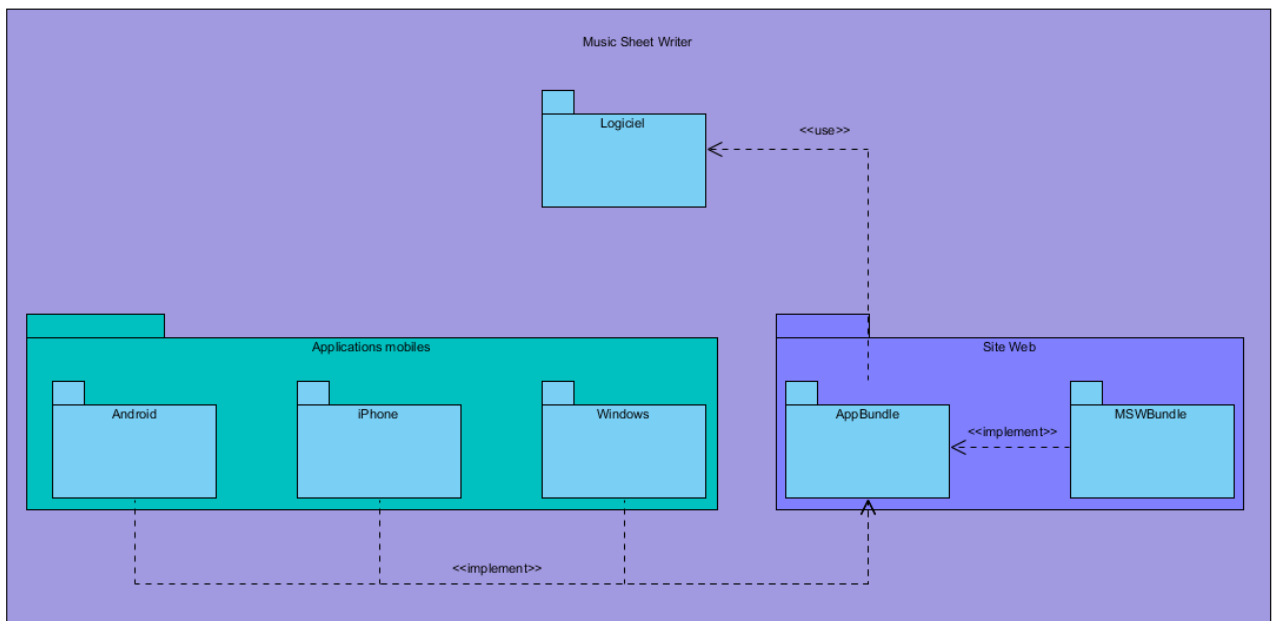




Figure 1. Architecture de Music Sheet Writer

Chacune des applications mobiles implémente les fonctionnalités données par le composant **API Rest** du serveur. Il en est de même pour le composant **Site Web** du serveur. De ce fait, toutes ces parties du projet sont dépendantes du composant **API Rest** qui est un des points les plus critiques du projet.

Bien que globalement le logiciel d'édition de partition soit indépendant des autres livrables, il existe une dépendance entre l'**API Rest** et ce dernier et concerne les partitions générées par le **Logiciel**. Cette dépendance est expliquée plus en détail dans la partie [3.2.2 AppBundle](#).

Si vous voulez des informations sur l'organisation du code de chacun des livrables et de leurs designs en détail, se référer au document AA2.

De même, pour savoir exactement les fonctionnalités qui seront présentes dans le projet final, se référer au cahier des charges.

 Music Sheet Writer	Date de publication	10/07/2016	 Epitech Innovative Project
	Nom du projet	Music Sheet Writer	
	Objet du document	Documentation Technique	
	Nom du chapitre	Les livrables	

3. Les livrables

3.1. Logiciel

3.1.1. Implémentation

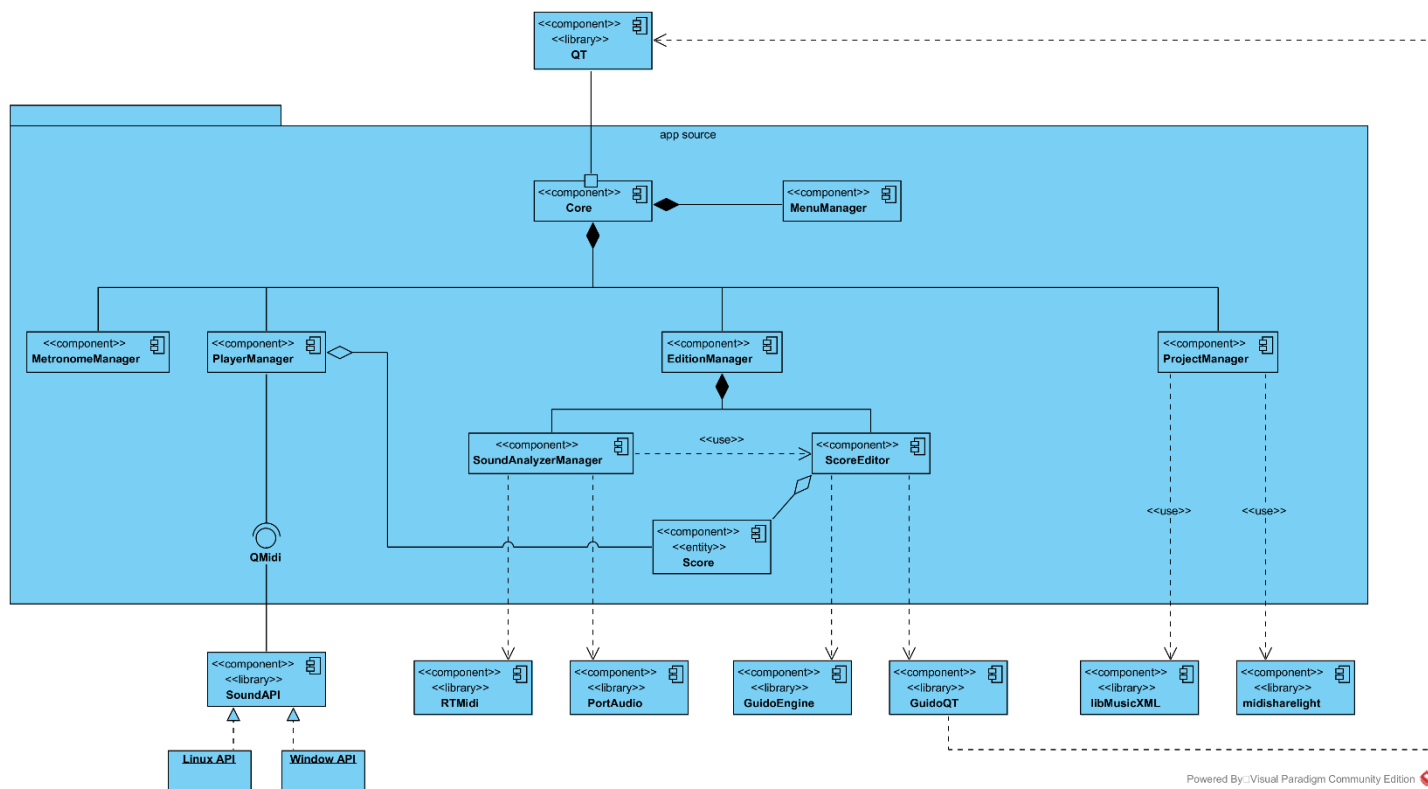


Figure 2. Architecture du logiciel

3.1.1.1. Gestion de projet

La gestion de projet englobe l'ensemble des actions qui concerne le projet à savoir créer, ouvrir, enregistrer, importer, exporter et fermer un projet. Cette gestion de projet se fait au travers du composant **ProjectManager** qui fait appel notamment aux librairies libMusicXML pour l'import et midisharelight pour l'export en Midi.



Un projet est représenté par la classe **Project** qui contient les informations du projet comme son nom ou son emplacement mais également un objet **Score** qui est la représentation de la partition.

L'import est géré par la classe **Parse**, elle permet de créer un fichier au format GUIDO depuis un fichier au format MusicXML, pour ensuite le parser manuellement. Il est plus simple par rapport au code existant de parser manuellement du GUIDO que directement du MusicXML, sachant que la lib actuellement utilisé est la LibGuido.

3.1.1.2. Édition de partition

L'édition de partition se fait au travers du composant **ScoreEditor**. Son but est d'éditer l'objet **Score** que contient le projet ouvert dans l'éditeur. Cela requiert deux composants pour fonctionner :

- La librairie **GuidoEngine** qui permet la gestion de la Guido Music Notation (gmn) qui est un langage pour la représentation de la musique au niveau d'une partition.

 Music Sheet Writer	Date de publication	10/07/2016	 Epitech Innovative Project
	Nom du projet	Music Sheet Writer	
	Objet du document	Documentation Technique	
	Nom du chapitre	<i>Les livrables</i>	

- La librairie **GuidoQt** qui gère l’affichage de la partition généré par la librairie GuidoEngine.

L’édition de partition se découpe ainsi :

- Le composant **EditionToolBar** contient l’ensemble des actions rattachées aux éléments contenu dans l’éditeur de partition.
- Lorsqu’une des actions est appelée, l’action va modifier le contenu de notre objet Score présent dans ScoreEditor et un signal est émis.
- Ce signal est reçu par le composant **EditionManager**, qui se charge de faire le lien entre l’éditeur (composant EditionToolBar) et l’affichage de la partition qui est géré par le composant **ScoreViewerWidget**.

3.1.1.3. Lecture de partition

La lecture de partition requière plusieurs composants pour fonctionner. Suivant les systèmes d’exploitation utilisé ces composants ne sont pas les même. C’est le cas pour les librairies qui vont jouer le son.

Le lecteur de partition utilise différentes interfaces pour accéder aux différentes bibliothèques suivant le système d’exploitation. La classe **QMidi** contrôle l’interface audio. Cette dernière va envoyer les évènements MIDI devant être joués au synthétiseur à l’API de son Windows ou Linux.

Le lecteur de partition est géré par la classe **PlayerManager** qui contient toutes les actions disponibles à savoir jouer, pause, arrêt, reprise.

3.1.1.4. L’analyse audio

L’analyse du son ce fait au travers du composant **SoundAnalyzerManager**. Il s’occupe de l’analyse audio par entrées Jack et MIDI



L’analyse par port Jack s’effectue en plusieurs étapes :

- Récupération du son : Le son est récupéré par l’entrée Jack de l’ordinateur grâce à la librairie **PortAudio**. L’échantillonnage est configurable
- Application du fenêtrage : Le fenêtrage de Hanning permet de clarifier le signal reçu.
- Application de la FFT : La FFT permet de récupérer le spectre fréquentiel du signal reçu.
- Reconnaissance de la note : la fréquence fondamentale et la durée de la note permettent d’identifier la note
- Envoie de la note à l’éditeur : la note est envoyée à l’éditeur pour affichage

L’analyse MIDI s’effectue par le biais de la récupération de donnée MIDI d’un instrument par l’ordinateur. Pour effectuer cette opération, nous avons besoin d’un instrument avec une sortie MIDI et d’un câble USB qui transfèrent les données MIDI vers votre ordinateur via un port USB.

Cette analyse MIDI s’effectue par l’utilisation de la librairie **RTMidi**. Elle nous permet la visualisation des données reçues depuis le piano.

Cette analyse nous permet la création de fichier midi depuis la classe **MIDIFile** en récupérant les données reçues.



 Music Sheet Writer	Date de publication	10/07/2016	 Epitech Innovative Project
	Nom du projet	Music Sheet Writer	
	Objet du document	Documentation Technique	
	Nom du chapitre	<i>Les livrables</i>	

3.1.2. Norme de codage

La norme de codage utilisée pour le logiciel suit intégralement les règles énoncées sur <http://geosoft.no/development/cppstyle.html>.

3.1.3. Problèmes existant

Pour la génération de partition par analyse audio, l'algorithme de la FFT ne permet pas de détecter avec précision des notes simultanées.

 Music Sheet Writer	Date de publication	10/07/2016	 Epitech Innovative Project
	Nom du projet	Music Sheet Writer	
	Objet du document	Documentation Technique	
	Nom du chapitre	<i>Les livrables</i>	

3.2. Site Web

3.2.1. Norme de codage

Nous suivons les normes de codage et bonnes pratiques officielle énoncé sur le site de symfony. Elles sont disponible à l'adresse suivante : https://symfony.com/doc/2.7/best_practices/index.html.

3.2.2. Architecture globale

La figure 1 de l'introduction représentait une version simplifiée de l'architecture actuellement en place au niveau du site internet. Vous trouverez ci-dessous l'architecture actuelle du site web :

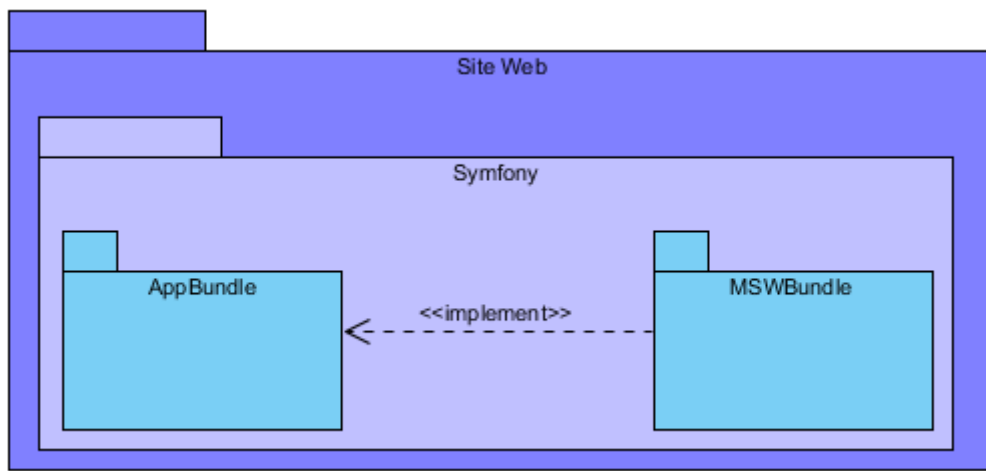


Figure 3. Architecture Site Web

Le développement de la partie site web du projet Music Sheet Writer est donc fait à l'aide du Framework PHP Symfony dans sa version 2.7. Toutes les procédures d'installation des différents outils de développement dont vous aurez besoin se trouvent dans le document d'installation (ID1).

Un projet Symfony, dans sa version 2.7, est organisé avec l'arborescence suivante :













	app	2/29/2016 5:43 PM	File folder	
	bin	2/29/2016 5:45 PM	File folder	
	files	2/29/2016 5:43 PM	File folder	
	src	2/29/2016 5:45 PM	File folder	
	web	2/29/2016 5:43 PM	File folder	
	.gitignore	2/29/2016 5:43 PM	Text Document	1 KB
	composer.json	2/29/2016 5:43 PM	JSON File	3 KB
	composer.lock	2/29/2016 5:43 PM	LOCK File	73 KB
	phpunit.phar	2/29/2016 5:43 PM	PHAR File	3,006 KB
	README.md	2/29/2016 5:43 PM	MD File	1 KB

Figure 4. Arborescence projet Symfony

 Music Sheet Writer	Date de publication	10/07/2016	 Epitech Innovative Project
	Nom du projet	Music Sheet Writer	
	Objet du document	Documentation Technique	
	Nom du chapitre	<i>Les livrables</i>	



- Le dossier **app** contient tous les fichiers en relation avec la configuration du projet et les logs.
- Le dossier **src** contient les sources du projet. C'est au sein de ce dernier que se trouvent les différents **bundles** du projet.
- Le dossier **web** est l'espace public du projet. C'est au sein de celui-ci que se trouvent les ressources nécessaires à l'utilisateur pour l'utilisation du site internet, comme les images ou scripts JavaScripts.

Nous ne rentrerons pas plus en détail dans la description d'un projet Symfony. Nous vous invitons à regarder la documentation officielle et tutoriels officiels de Symfony pour cela.

Une des notions principales de Symfony est le système de bundle. Un bundle est un composant logique et indépendant dans un projet Symfony et ne devrait pas avoir de dépendance envers d'autres bundles qui n'ont pas vocation à être distribué au public. Dans notre projet, deux bundles ont été développés : le premier est le bundle **AppBundle** et est l'implémentation de notre API Rest définie dans le document 2017_API_musicsheetwriter.pdf ; le second est le bundle **MSWBundle**. Ce dernier contient toute la logique du site vitrine et de l'espace communautaire du projet et fait appel aux fonctionnalités de l'API Rest.

L'implémentation des bundles suit les bonnes pratiques définies dans la documentation officielle de Symfony :

- Documentation officielle : <https://symfony.com/doc/2.7/index.html>
- Bonnes pratiques : https://symfony.com/doc/2.7/best_practices/index.html

 Music Sheet Writer	Date de publication	10/07/2016	 Epitech Innovative Project
	Nom du projet	Music Sheet Writer	
	Objet du document	Documentation Technique	
	Nom du chapitre	<i>Les livrables</i>	

3.2.3. AppBundle

Comme expliqué en introduction de cette partie, le bundle AppBundle est l'implémentation directe de l'API Rest définie dans le document 2017_API_musicsheetwriter.pdf. Cette dernière définit l'ensemble des routes pouvant être utilisé par l'utilisateur tout en définissant les différents rôles que doit avoir ce dernier pour pouvoir accéder à certaines routes et fonctions.

L'ensemble des routes et donc fonctionnalités définies dans le document de l'API se font à partir des routes commençant par **/api/**. Ceci est un choix dans l'implémentation de l'API Rest et n'est donc pas défini dans le document de définition.

Vous trouverez ci-dessous l'organisation de ce bundle telle qu'il est implémenté actuellement :

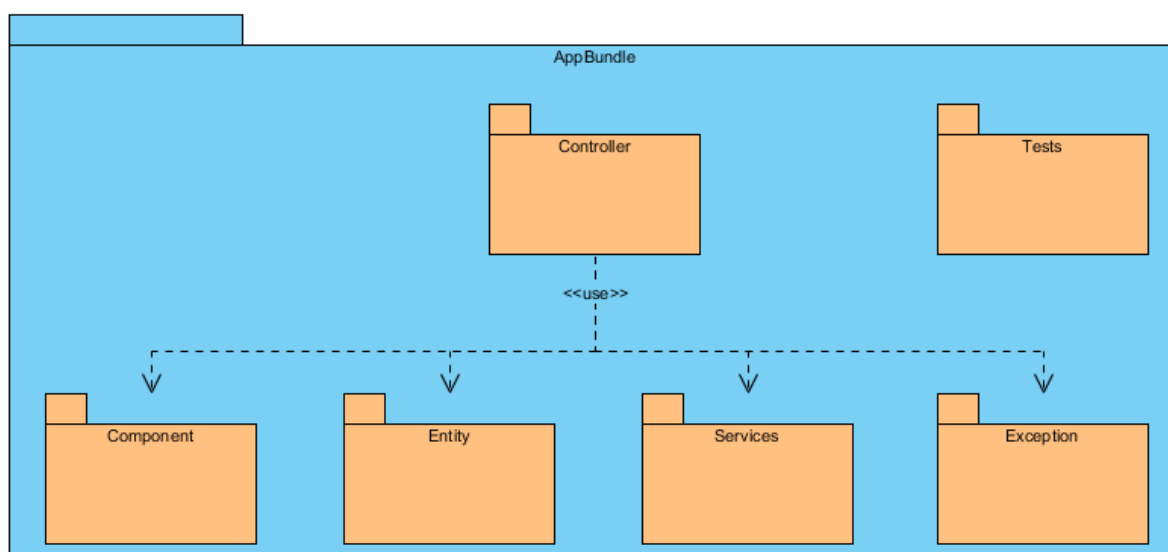




Figure 5. Architecture AppBundle

Le bundle contient plus de dossiers que ceux représentés dans la figure ci-dessus. Nous n'en parlerons pas ici car il s'agit de dossier contenant des fichiers de configuration générés lors de la création du bundle et qui n'ont pas besoin d'être modifiés pour le projet.

Le bundle est donc composé de différents dossiers :

- **Controller** : Contient les fichiers des classes PHP implémentant les différentes routes définies par l'API Rest. Ces classes sont souvent appelées par le terme anglais *Controller*.
- **Component** : Contient les classes PHP permettant le formatage des réponses renvoyées par les différents Controllers au format spécifié par l'API Rest ainsi que d'autres classes pouvant être utiles à l'implémentation de l'API Rest. Une des règles pour pouvoir ajouter une classe ou fonction aux classes déjà présentes est que cet ajout est indépendant de tous les autres composants du bundle. Cependant les autres composants peuvent être dépendants des informations contenues dans ces classes-là.
- **Entity** : Contient les classes PHP représentant les modèles gérés par l'API Rest (Users, Scores, etc). Aucune logique ne doit être implémentée dans ces classes-là.

 Music Sheet Writer	Date de publication	10/07/2016	 Epitech Innovative Project
	Nom du projet	Music Sheet Writer	
	Objet du document	Documentation Technique	
	Nom du chapitre	<i>Les livrables</i>	

- **Services** : Contient les classes PHP implémentant la logique de l'API Rest. C'est au sein de ces dernières que s'effectue le traitement des entités. Par exemple, le service **User**, s'occupe de l'implémentation des fonctionnalités liés aux utilisateurs et à la manipulation de l'entité **User**.
- **Exception** : Contient les classes PHP d'exception qui sont utilisées en cas d'erreur dans la manipulation des données de la part de l'utilisateur suite à une mauvaise utilisation de l'API ou dans le cas de problèmes liés au serveur en lui-même.
- **Tests** : Contient la définition des tests unitaires pour le bundle.

Nous avons parlé dans la partie [2. Le projet](#) d'une dépendance entre le logiciel et l'API Rest. Cette dépendance se situe au niveau des partitions de musique. L'API Rest ne peut upload sur le serveur que les partitions écrites à notre format (**msw**).

3.2.4. MSWBundle

Le bundle MSWBundle est le second bundle lié à la partie serveur du projet Music Sheet Writer. Il est responsable du site internet vitrine et de l'espace communautaire qui lui fait appels aux fonctionnalités de l'API Rest.

Voici l'organisation interne du bundle :

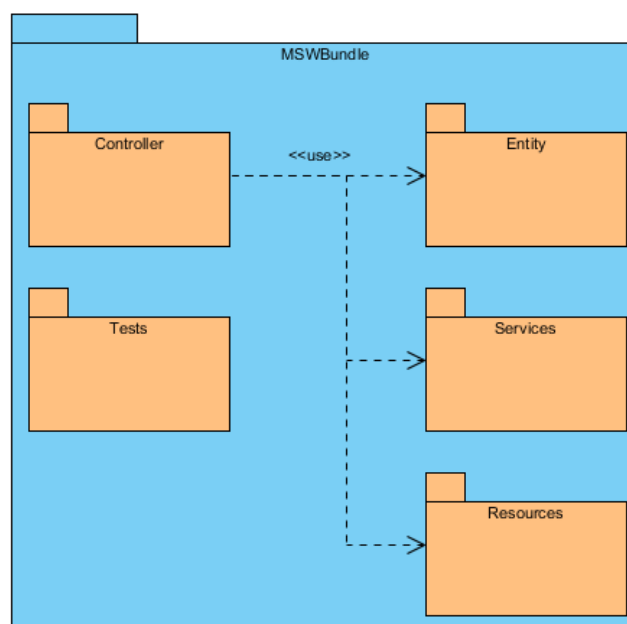




Figure 6. MSW Bundle – organisation

Comme pour le bundle AppBundle, nous ne décrivons ici que les dossiers réellement utilisés dans le développement de ce dernier. Les autres dossiers contiennent les fichiers de configuration liés au bundle.

Le dossier est composé de différents dossiers :

- **Controller** : Contient les classes PHP s'occupant des routes pouvant être atteintes par l'utilisateur. Deux classes sont pour le moment implémentées : **ShowcaseController** pour

 Music Sheet Writer	Date de publication	10/07/2016	 Epitech Innovative Project
	Nom du projet	Music Sheet Writer	
	Objet du document	Documentation Technique	
	Nom du chapitre	Les livrables	

toutes les routes liées au site vitrine et **CommunityController** pour les routes liées à l'espace communautaire.

- **Entity** : Contient les classes PHP représentant les données manipulées par l'API Rest. Leurs définitions suivent celle de l'API Rest, mais est différent de celle du bundle AppBundle. Ici, les objets sont très proches de leur représentation dans les réponses renvoyées par l'API Rest.

```
{
  "is_subscription": true,
  "personal_data": {
    "id": 1,
    "username": "bob",
    "firstname": "bob",
    "lastname": "marley",
    "email": "bob.marley@jamaica.jm",
    "message": "I have only one love",
    "photo": "/images/default_avatar.png",
    "nb_subscriptions": 45,
    "nb_subscribers": 125,
    "nb_favourites": 30,
    "nb_scores": 30,
  },
  "subscription": [],
  "subscriber": [],
  "score": {
    "own": [],
    "favourite": []
  }
}
```



Figure 7. Réponse API Rest - User

```
class User
{
    private $isSubscription;
    private $personalData;
    private $subscriptions;
    private $subscribers;
    private $ownScores;
    private $favouriteScores;
    private $last_activity_date;

    ...
}
```

Figure 8. Implémentation User - MSWBundle

- **Service** : Les classes qui se trouvent dans ce dossier ont le même rôle que celles du dossier **Service** du bundle AppBundle. Manipuler les entités. Ici, une seule classe a été définie : **EntityFactory**. Cette classe s'occupe de passer les entités des réponses JSON de l'API Rest en objets PHP exploitable pour l'espace communautaire.

 Music Sheet Writer	Date de publication	10/07/2016	 Epitech Innovative Project
	Nom du projet	Music Sheet Writer	
	Objet du document	Documentation Technique	
	Nom du chapitre	<i>Les livrables</i>	

- **Resources** : Contient l'ensemble des ressources utilisées pour le front-end du site internet. Vous trouverez dedans plusieurs dossiers :

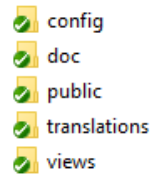




Figure 9. Arborescence dossier Resources – MSWBundlle

- **Config** : Fichiers de configuration
- **Doc** : Fichiers de documentation
- **Public** : Dossier contenant les ressources utilisées sur les pages web tel que les images, vidéos ou autres scripts JavaScripts.
- **Translations** : Fichiers de traduction
- **Views** : Les différentes vues du projet. Ces vues peuvent être des pages web complètes ou des blocks de code HTML réutilisables. Ces vues utilisent le moteur de template Twig. Vous trouverez plus d'information sur ce moteur à l'adresse : <http://twig.sensiolabs.org/>.
- **Test** : Contient les différentes classes se chargeant d'effectuer les tests unitaires.

 Music Sheet Writer	Date de publication	10/07/2016	 Epitech Innovative Project
	Nom du projet	Music Sheet Writer	
	Objet du document	Documentation Technique	
	Nom du chapitre	Les livrables	

3.3. Application Android

3.3.1. Organisation du projet

Le projet est composé d'un unique module **app** regroupant le code source de l'application et les ressources tels que les images, icônes ou encore fichiers de traduction. Les fichiers Java composant le code source de l'application sont dans le dossier **app/java/main**. Toutes les classes se trouvent dans le package principal **com.musicsheetwriter.musicsheetwriter**.

Par la suite, elles sont regroupées dans des sous-packages selon leurs fonctions et/ou leur type. Par exemple, les classes contenant le code sources des **fragments** utilisés dans des onglets se trouvent dans le package **fragmenttabs**.

Autre exemple, les **adapters** pour liste se trouvent dans le package **listadapter**. Seules les classes **activities** sont à la racine du package principal.

Le nom des classes doit être explicite et en rapport avec ce que la classe implémente. Il doit se terminer par la nature de la classe. En effet, pour les fragments, les noms de classes se termine par **Fragment**. De même, les noms des activités se terminent par **Activity** et celui des adapters pour liste par **ListAdapter**. Cela permet d'identifier rapidement le type de classe que l'on manipule. Ci-dessous se trouvent des exemples de noms de classes.

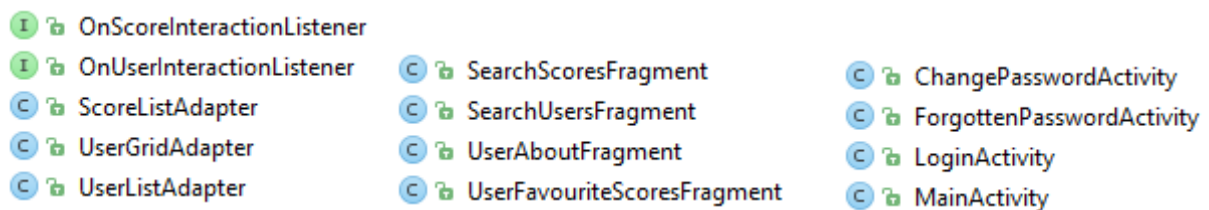




Figure 10. Exemples noms de classes – Android

L'organisation des ressources est imposée par la manière dont Android gère les ressources d'une application. A savoir qu'ils sont regroupés dans des dossiers différents en fonction de la nature de la ressource et des spécificités des téléphones pour lesquels ils s'appliquent. Par exemple, les images commune à tous les téléphones se trouvent dans le dossier **drawable** ; en revanche, le dossier **drawable-hdpi** contient les images qui ne seront utilisées que sur des téléphones ayant des écrans de « haute-densité ».

Vous pouvez trouver plus de détails sur la gestion des ressources sur Android en consultant la page <http://developer.android.com/guide/topics/resources/providing-resources.html>.

Il est important de tenir compte des différents type de téléphone lorsqu'une ressource est ajouté car cela accroît la portabilité et la compatibilité de l'interface utilisateur de l'application sur d'autres téléphone.

 Music Sheet Writer	Date de publication	10/07/2016	 Epitech Innovative Project
	Nom du projet	Music Sheet Writer	
	Objet du document	Documentation Technique	
	Nom du chapitre	<i>Les livrables</i>	

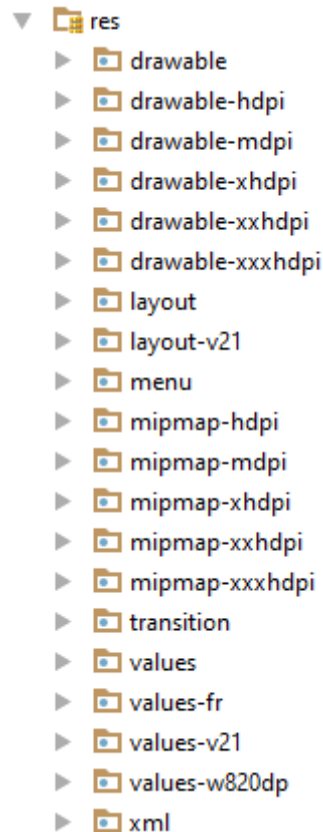


Figure 11. Arborescence des ressources - Android

Sous Android Studio, un projet Android est associé à un ou plusieurs fichiers **gradle**. Un unique fichier est associé au projet dans son ensemble tandis que chaque module au sein du projet a également son propre fichier gradle.

Le fichier gradle du projet contient les informations de configuration des autres fichiers gradle. Ces derniers en revanche définissent les configurations des modules auxquels ils sont associés. Ainsi, les dépendances de certaines bibliothèques, la version cible de l'application et d'autres informations peuvent être modifiés en éditant les fichiers gradle.

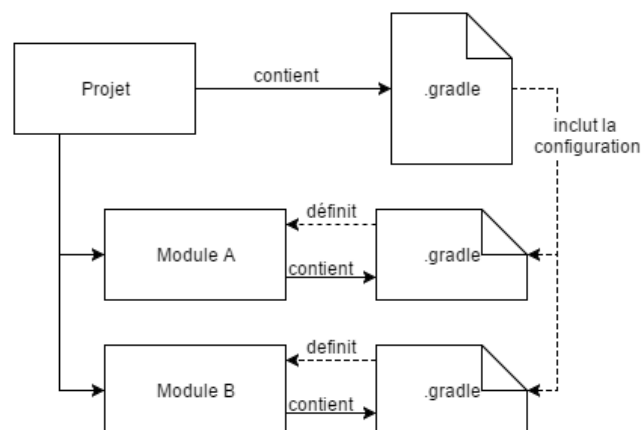




Figure 12. Hiérarchisation des fichiers gradle - Android

 Music Sheet Writer	Date de publication	10/07/2016	 Epitech Innovative Project
	Nom du projet	Music Sheet Writer	
	Objet du document	Documentation Technique	
	Nom du chapitre	Les livrables	

3.3.2. Structure et présentation des composants

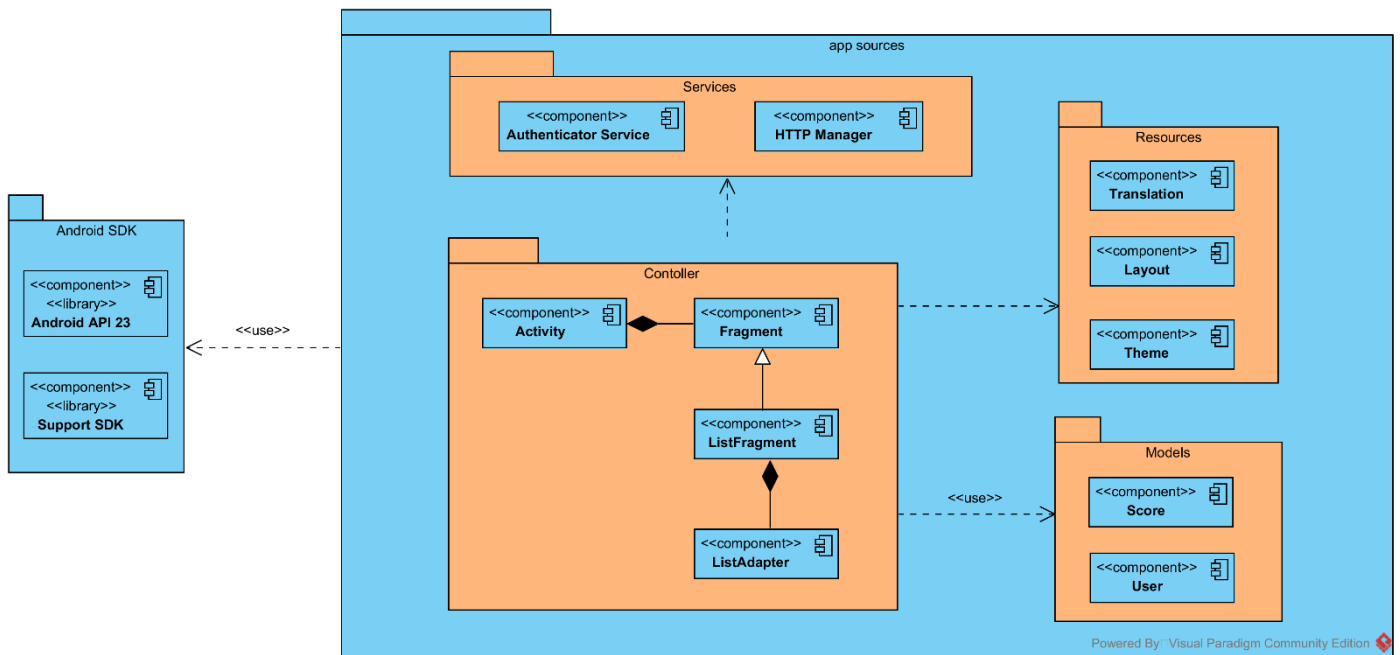




Figure 13. Structure des composants - Android

Nom des composants	Description et rôles
Activity	C'est le composant central d'une application Android. Au sein de l'application Music Sheet Writer. Une activité est une partie de l'application ayant un rôle distinct. Par exemple, « affichage d'une partition », « affichage d'un profil utilisateur », « connexion/création de compte » ou encore « réinitialisation du mot de passe ».
Fragment	Un fragment est considéré comme une « sous-activité ». Il peut gérer une partie de la couche métier et/ou graphique de l'activité à laquelle il appartient. Au sein de l'application Music Sheet Writer, certaines activités n'ont pas de fragment car le rôle de l'activité est suffisamment spécifique pour ne pas être subdivisé comme la « réinitialisation du mot de passe » par exemple. En revanche, certaines activités sont amenées à avoir plusieurs écrans et à gérer plusieurs tâches. C'est le cas par exemple de « la consultation d'un profil utilisateur ». En effet, cela comprend, la visualisation de ses partitions, de ses abonnements et de ses informations personnelles. Ainsi, trois fragments sont nécessaires pour un tel cas.
ListFragment	Il s'agit d'un type de fragment spécifique qui ne s'occupe d'une liste d'éléments comme une liste abonnements ou de partitions.
ListAdapter	Les ListAdapters gère les données et la mise en forme de ces données au sein d'une liste.
Service (package)	Il s'agit des différents services auxquels ont accès les fragments et les activités afin de faciliter leur travail.
Model (package)	Il s'agit de la définition des entités que les « contrôleurs » seront amenés à manipuler.
Resources (package)	Il s'agit des ressources graphiques, de fichiers de traduction ou contenant des valeurs afin d'éviter de les « mettre en dur ».

Tableau 1. Présentation des composants - Android

 Music Sheet Writer	Date de publication	10/07/2016	 Epitech Innovative Project
	Nom du projet	Music Sheet Writer	
	Objet du document	Documentation Technique	
	Nom du chapitre	Les livrables	

3.3.3. Norme de codage

La norme de codage utilisée pour l'application Android suit intégralement les règles énoncées sur :

- <https://source.android.com/source/code-style.html#use-javadoc-standard-comments>
- <https://google.github.io/styleguide/javaguide.html#s7.3.2-javadoc-exception-overrides>

3.3.4. Installation et lancement de l'application

La procédure de récupération du code source et d'installation de l'environnement est détaillée dans le document d'installation (ID1).

Pour lancer l'application, il suffit de créer une configuration en cliquant sur **Run → Edit Configuration...** Dans la fenêtre contextuelle, entrer les informations comme suit :

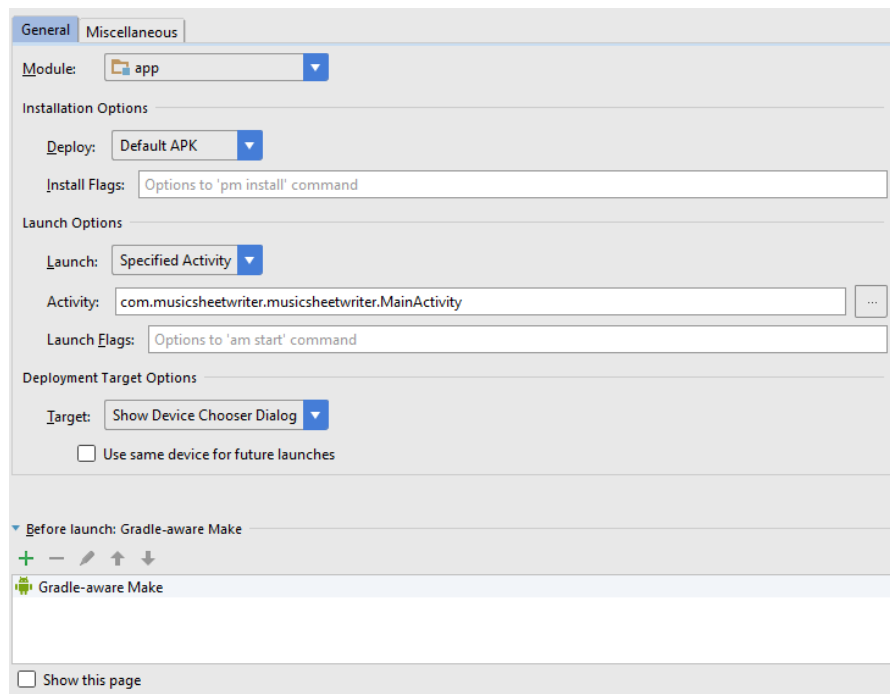





Figure 14. Déclaration d'une configuration de lancement dans Android Studio

Ensuite, il suffit de sélectionner **Run → Run 'app'** ou tout simplement cliquer sur  pour lancer l'application. Il suffira juste de choisir sur quel émulateur ou téléphone branché à l'ordinateur l'application doit se lancer.

3.3.5. Tests

Il existe différents types de tests sur Android. Ces types sont définis dans le document TS. Le code source des tests se trouve dans **app/androidTest/java**. Les tests utilisent le framework JUnit couplé au AndroidJUnitRunner pour l'automatisation des tests et la bibliothèque espresso pour les tests d'intégration et d'interface utilisateurs. Toutes ces bibliothèques sont déjà liées au projet grâce au fichier gradle se trouvant dans le module app.

 Music Sheet Writer	Date de publication	10/07/2016	 Epitech Innovative Project
	Nom du projet	Music Sheet Writer	
	Objet du document	Documentation Technique	
	Nom du chapitre	Les livrables	

```
// Android unit test
androidTestCompile 'com.android.support:support-annotations:23.1.1'
androidTestCompile 'com.android.support.test:runner:0.4.1'
androidTestCompile 'com.android.support.test:rules:0.4.1'
// Hamcrest library
androidTestCompile 'org.hamcrest:hamcrest-library:1.3'
// UI testing with Espresso
androidTestCompile 'com.android.support.test.espresso:espresso-core:2.2.1'
// UI testing with UI Automator
androidTestCompile 'com.android.support.test.uiautomator:uiautomator-v18:2.1.1'
```

Figure 15. Liste des bibliothèques à inclure dans le fichier gradle pour l'application Android

Toutes les classes regroupant les cas de test se trouvent dans le package principal **com.musicsheetwriter.musicsheetwriter.test**. Les tests de chaque élément se trouvent dans un package séparé. Par exemple, les suites de test relatives à la classe **LoginActivity** se trouvent dans le package **loginactivity**.



Figure 16. Arborescence des classes de test - Android

Pour lancer les tests, il suffit de créer une configuration de lancement en sélectionnant :

Run ➔ **Edit Configuration...** ➔ **+** ➔ **Android Tests** et de remplir les informations comme suit :

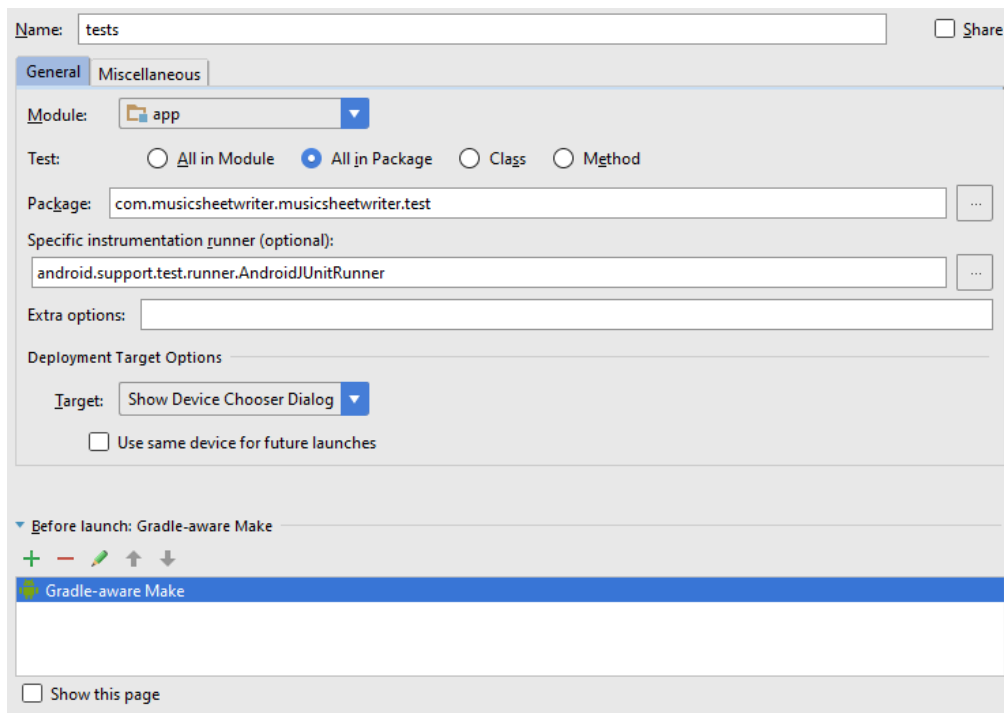






Figure 17. Déclaration d'une configuration de lancement de tests dans Android Studio

 Music Sheet Writer	Date de publication	10/07/2016	 Epitech Innovative Project
	Nom du projet	Music Sheet Writer	
	Objet du document	Documentation Technique	
	Nom du chapitre	<i>Les livrables</i>	

3.3.6. Problèmes connus et axes d'amélioration

Actuellement, sans accès à internet, l'application est inutile car aucune information ne peut être récupérée. Ainsi, il serait très bénéfique pour l'utilisateur que certaine des données de l'utilisateur soit gardé en cache en utilisant soit la base de données SQL Lite intégré à Android, soit en utilisant le cache mémoire d'Android. Les informations tels que la liste et les images des partitions de l'utilisateur, sa liste d'abonnements, et ces informations personnelles.

 Music Sheet Writer	Date de publication	10/07/2016	 Epitech Innovative Project
	Nom du projet	Music Sheet Writer	
	Objet du document	Documentation Technique	
	Nom du chapitre	Les livrables	

3.4. Application iOS

3.4.1. Organisation du projet

L'application iOS est gérée dans un système de **workspace**. Ce dernier peut contenir plusieurs projets. Dans notre cas nous en avons deux : l'application en elle-même et un projet pour les tests.

Le projet est organisé selon un système simple, tous les fichiers « .m » et « .h » communs sont regroupés dans un dossier portant leur nom ou fonction par rapport à l'application. Par exemple test.m et test.h seront dans le dossier test.

Il peut y avoir certaines fois plusieurs fichiers dans le même dossier s'ils sont relativement semblable ou s'ils fonctionnent ensemble. Par exemple le dossier test connexion contient une classe pour l'affichage de l'erreur et une autre pour les tests de connexion.

Chaque méthode est formatée selon le prototype suivant : **VoiciUneMethodeDeClasse**. Cette règle de nomination est générale et utilisée par Apple.

Un autre point clef de ce projet est la gestion des **assets** ; chaque image, logo ou tout autre chose visuel extérieur venant à être intégrée au projet devront être placée dans le dossier **Xcassets** directement par glissé-déposé au travers de XCode. Ce dossier permet une classification claire des images et par la même occasion permet en cas de changement de ne le faire qu'à cette endroit-là et non à chaque fois que l'image est affiché. Il viendra aussi gérer les différentes tailles selon l'appareil.

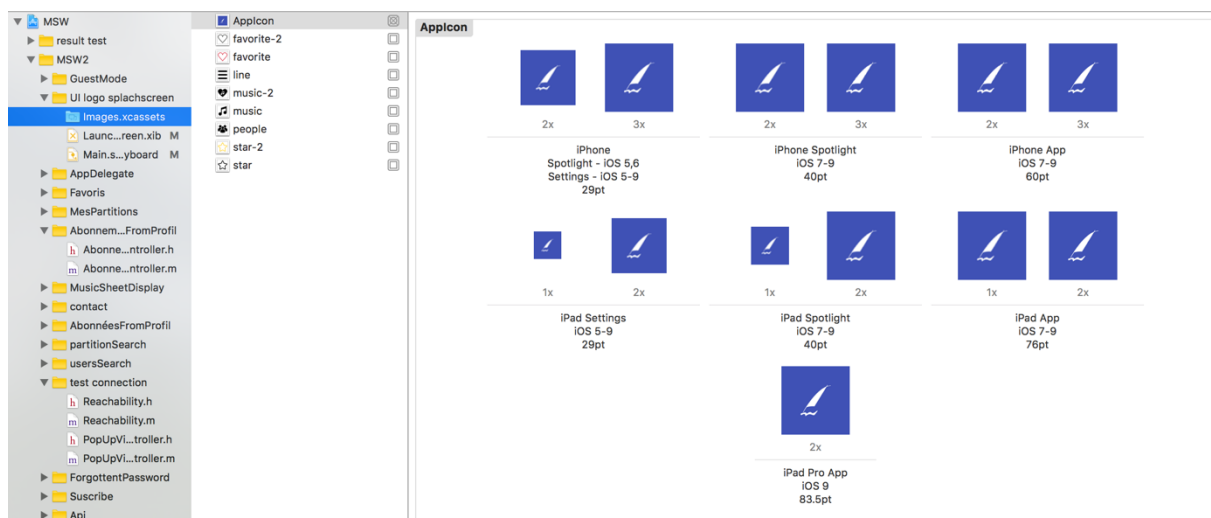


Figure 18. Dossier Xcasset - iOS

3.4.2. Installation et démarrage du projet

Le processus d'installation de l'application dans votre environnement de développement se trouve dans le document d'installation (ID1).

Pour lancer le projet, il suffit de sélectionner un émulateur et de cliquer sur **Run**.

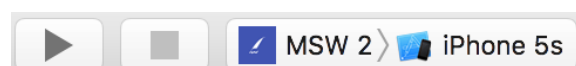




Figure 19. Lancement application - iOS

 Music Sheet Writer	Date de publication	10/07/2016	 Epitech Innovative Project
	Nom du projet	Music Sheet Writer	
	Objet du document	Documentation Technique	
	Nom du chapitre	<i>Les livrables</i>	

3.4.3. Démarrage et gestion des tests

Nous avons différents types de tests mis en place pour l'application :

- Les tests d'interface utilisateur gérés par la librairie KIF.
- Les tests unitaires gérés par XCode.

Tous les tests, peu importe leurs natures, se trouvent dans le dossier **MSW2Test**. Ce dossier contient deux sous dossiers :

- **testUI** : Pour les tests liés à l'interface utilisateur.
- **test logique** : Pour les tests unitaires.

Pour lancer les tests il suffit d'aller dans le **test navigator** et de cliquer sur le petit symbole **play** à côté du nom du fichier pour lancer tous les tests ou sinon de cliquer sur le symbole devant la fonction de test pour n'en lancer qu'un.

Certains tests ont des dépendances avec d'autres, surtout les tests d'interfaces. Il faut donc faire attention à ce détail lors du lancement des tests.

3.4.4. Compréhension générale des vues

Le cœur du projet se trouve dans le fichier **MainStoryboard** qui permet la gestion de toutes les vues mais aussi des liens entre elles. Grâce à ce fichier, on peut modifier toutes les informations de l'application.

Chaque vue est de base liée avec une classe qui s'occupera d'implémenter les fonctionnalités de cette dernière et de toute autres vues du même type. Si nous voulons avoir un comportement spécifique pour une vue, il faut alors la lier avec une classe propre au projet.

Pour pouvoir une classe personnelle à une vue il suffit de rentrer son nom dans **Custom class**. De plus, certains liens entre les vues permettent l'échange de données entre les classes, c'est pour cela que certains liens ont des identifiants.

Le projet a été développé pour pouvoir fonctionner sur plusieurs modèles d'iPhone. Il faut donc, dans la création de vue, placer des layouts et les tester pour être sûr du bon fonctionnement sur le reste des appareils.



3.4.5. Création de vue

XCode facilite énormément de tâches en ce qui concerne la création des vues et leurs comportements, c'est pourquoi il faut faire extrêmement attention lors de leurs manipulations.

Nous donnons dans le reste de cette section un exemple de création de vue afin de comprendre comment l'application a été développée.

Nous allons créer un **View Controller** (classe permettant la manipulation d'une vue) au sein d'un autre View Controller. Ceci se fait à l'aide d'un lien appelé **embed**.

Tout d'abord il faut créer une vue. Placer ensuite à l'intérieur un **Container View**.

 Music Sheet Writer	Date de publication	10/07/2016	 Epitech Innovative Project
	Nom du projet	Music Sheet Writer	
	Objet du document	Documentation Technique	
	Nom du chapitre	Les livrables	

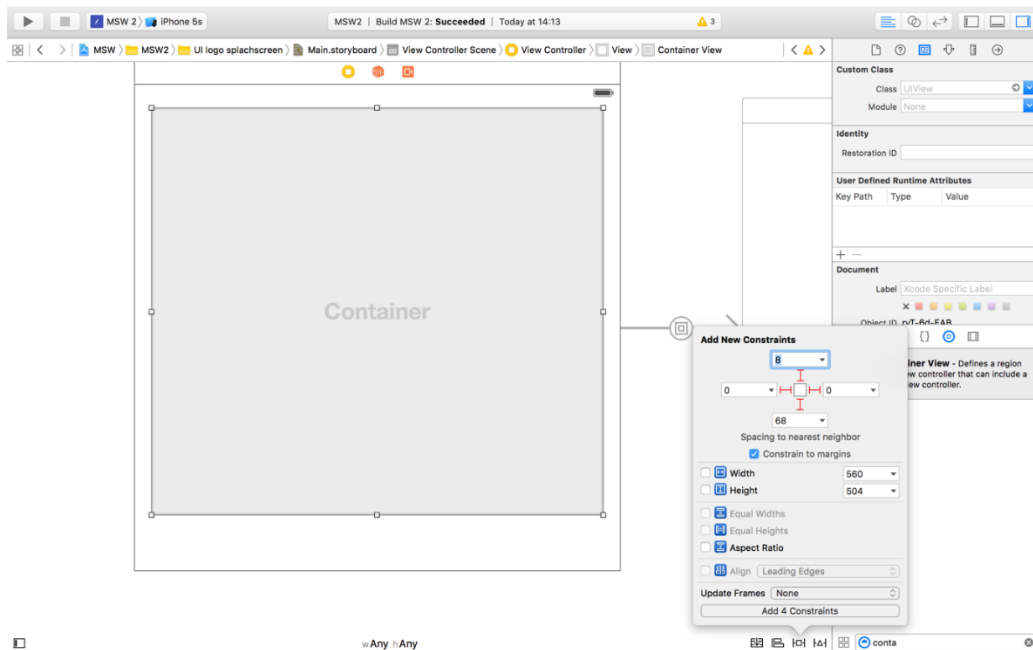


Figure 20. Création vue - iOS

Un View Controller a été créé par défaut. Nous ne l'utiliserons pas, il faut donc le supprimer.

Placer ensuite un **Table View Controller** et créer un lien (ctrl + clique gauche) entre le container et le Table View, il faut ensuite choisir **Embed**.

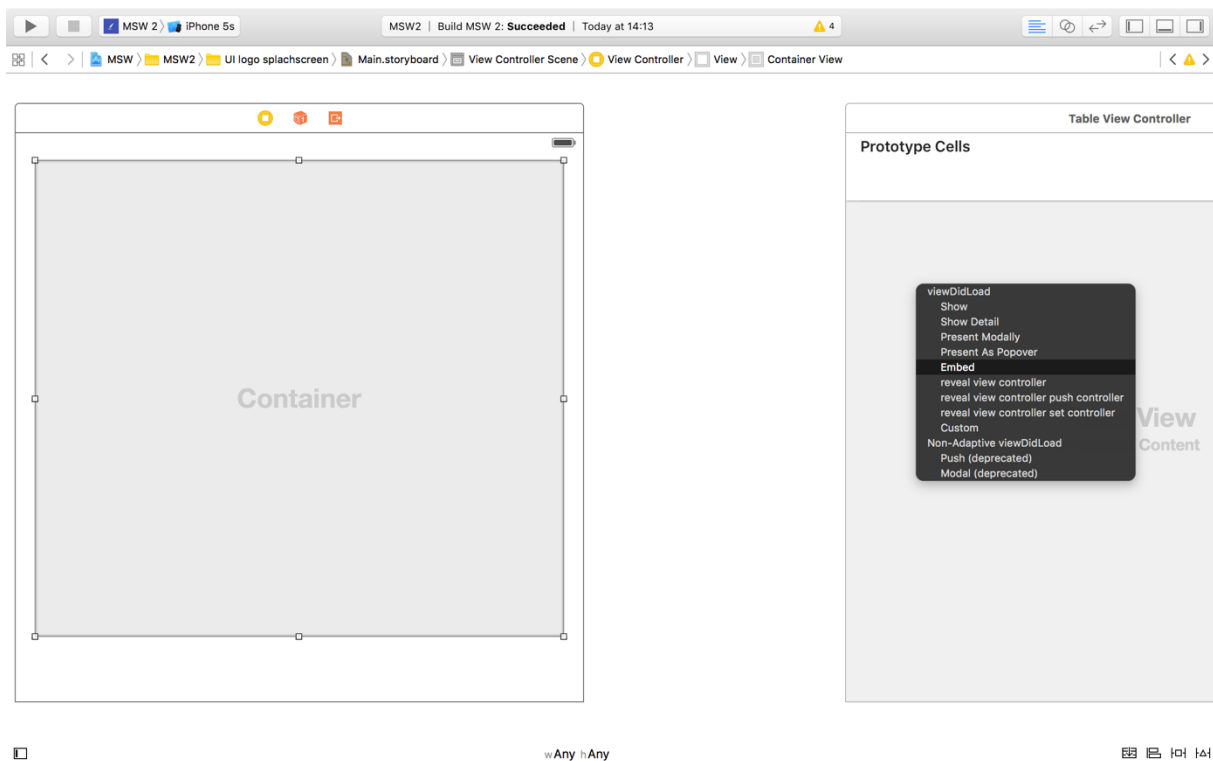




Figure 21. Création embed - iOS

 Music Sheet Writer	Date de publication	10/07/2016	 Epitech Innovative Project
	Nom du projet	Music Sheet Writer	
	Objet du document	Documentation Technique	
	Nom du chapitre	Les livrables	

À partir de là, une vue vient d'être créée avec à l'intérieur une Table View. L'intérêt de faire un embed est que nous pouvons quand même créer un Table View Controller alors que si l'on place un simple Table View dans la vue il pourrait y avoir des problèmes si la gestion devient complexe. Donc grâce à un lien embed on peut gérer un View Controller dans un autre View Controller. Le point important est donc d'avoir ces controller et non une simple vue.

Vous pouvez ensuite lier une classe à la vue, le moyen le plus simple est d'utiliser l'assistant XCode.

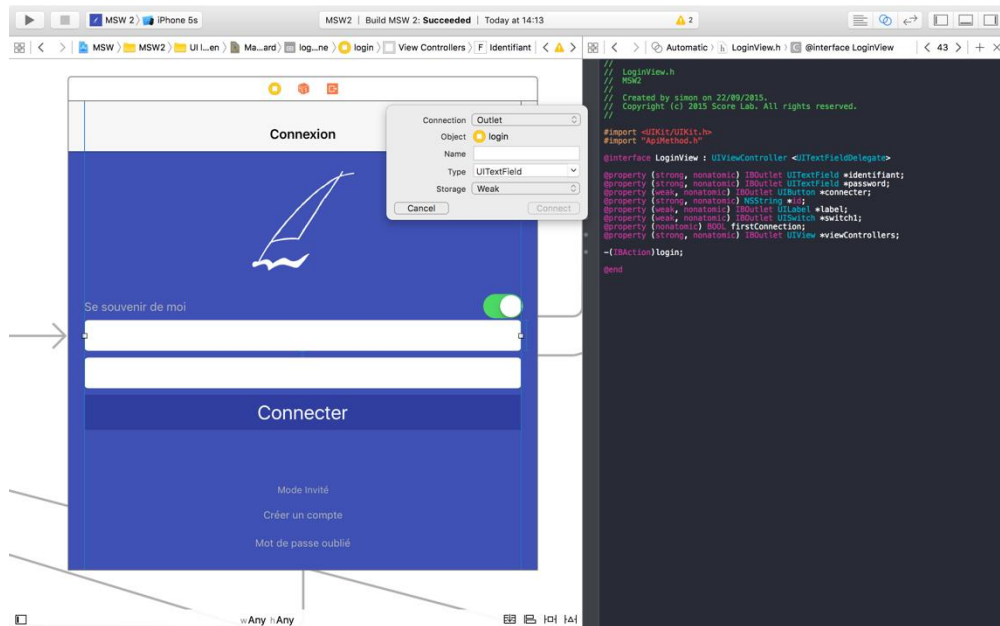




Figure 22. Mode assistant

3.4.6. Problèmes et axes d'amélioration

Un des problèmes pouvant être rencontré est le blocage lors d'une requête HTTPS. L'application utilise la librairie **AFNetworking** qui parfois sans raison apparente ne réagit plus à une requête et se bloque pendant plusieurs secondes. Un moyen de régler ce genre de problèmes serait de remplacer le code utilisant AFNetworking dans le fichier **ApiMethod.m** par les méthodes natives à iOS.

 Music Sheet Writer	Date de publication	10/07/2016	 Epitech Innovative Project
	Nom du projet	Music Sheet Writer	
	Objet du document	Documentation Technique	
	Nom du chapitre	<i>Les livrables</i>	

3.1. Application Windows Phone

3.1.1. Organisation du projet

Le projet est géré par un système de **solution**, cette dernière comprend un ensemble de projets.

Le projet Music Sheet Writer est composé de plusieurs dossiers et fichiers tels que les images, icônes ou encore fichiers de traduction. Les fichiers .cs composant le code source de l'application se trouvent à l'intérieur de dossiers spécifiques selon l'architecture MVVM (Modèle-vue-vue modèle). Ce dernier est implémenté à l'aide de Silverlight et l'outil MVVM Light dont les ressources officielles se trouvent à l'adresse suivante : <https://mvvmlight.codeplex.com/>.

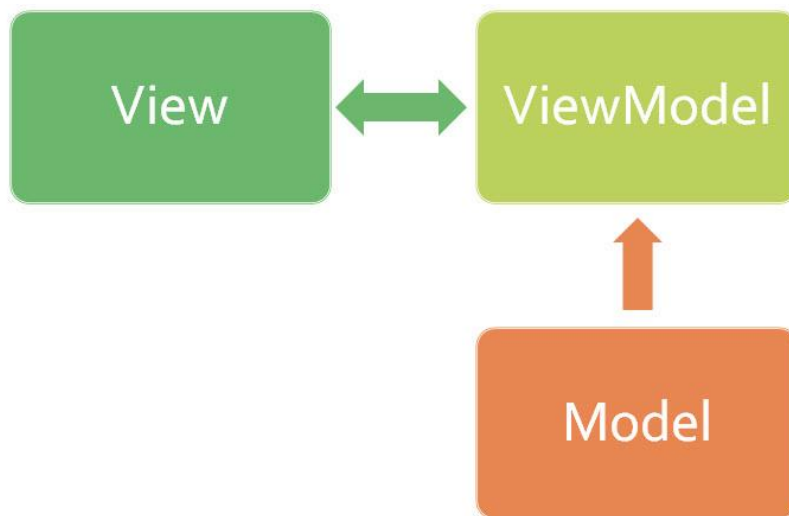


Figure 23. Schématisation MVVM

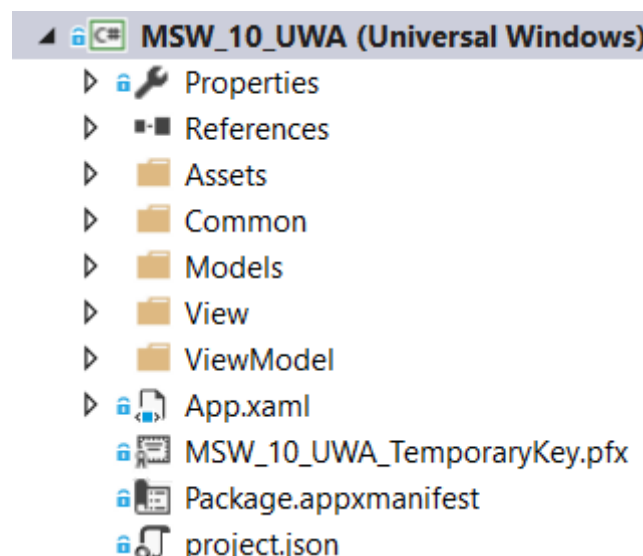




Figure 24. Organisation code source projet Music Sheet Writer

La gestion des images, logo ou tout autre ressources visuelles extérieur venant à être intégrée au projet devront être placée dans le dossier **Assets**. Ce dossier permet une classification claire des

 Music Sheet Writer	Date de publication	10/07/2016	 Epitech Innovative Project
	Nom du projet	Music Sheet Writer	
	Objet du document	Documentation Technique	
	Nom du chapitre	Les livrables	

images et par la même occasion permet en cas de changement de ne le faire qu'à cet endroit-là. Il viendra aussi gérer les différentes tailles d'images pour le projet.

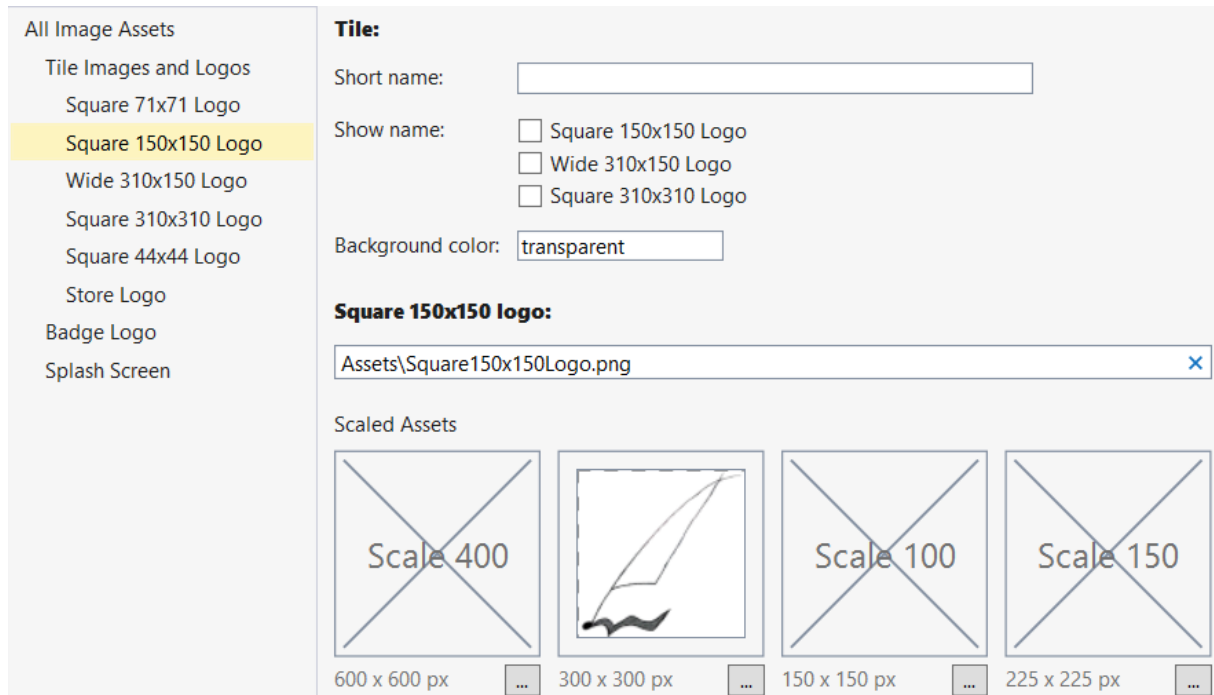


Figure 25 Gestion des images via le dossier Assets

Le dossier **Common** comprend l'ensemble des fichiers .cs qui sont utilisés dans plusieurs fichiers du projet afin d'éviter les redondances de code inutile dans le projet.

Le dossier **Models** représente les modèles utilisés par l'architecture MVVM. On trouvera **Score.cs** et **User.cs** qui représentent respectivement une partition et un utilisateur. L'application utilise ces deux modèles dans toutes les actions effectuées.



Le dossier **View** représente les vues utilisées par l'architecture MVVM. Les vues correspondent aux éléments visuels auxquels l'utilisateur va pouvoir interagir. Ces actions seront ensuite traitées par les classes du dossier **ViewModel**.

Le dossier **ViewModel** représente les modèles de vue utilisées par l'architecture MVVM. Ces fichiers permettent l'interaction des utilisateurs avec l'application. Ils s'occupent d'implémenter la logique derrière les actions de l'utilisateur.

Le nom des classes doit être explicite et en rapport avec ce que la classe implémente. Ce nom est suivi de **View** ou **ViewModel** en fonction de son rôle et doit être présent dans le dossier correspondant. Une **View** sera toujours associée à une **ViewModel** et vice et versa.

3.1.2. Norme de codage

La norme de codage utilisée pour l'application Windows suit intégralement les règles énoncées sur <https://msdn.microsoft.com/fr-fr/library/ff926074.aspx>

 Music Sheet Writer	Date de publication	10/07/2016	 Epitech Innovative Project
	Nom du projet	Music Sheet Writer	
	Objet du document	Documentation Technique	
	Nom du chapitre	<i>Les livrables</i>	

3.1.3. Installation et démarrage du projet

La procédure de récupération du code source et d'installation de l'environnement est détaillée dans le document d'installation (ID3).

Il suffit de choisir sur quel émulateur ou téléphone branché à l'ordinateur l'application doit se lancer.

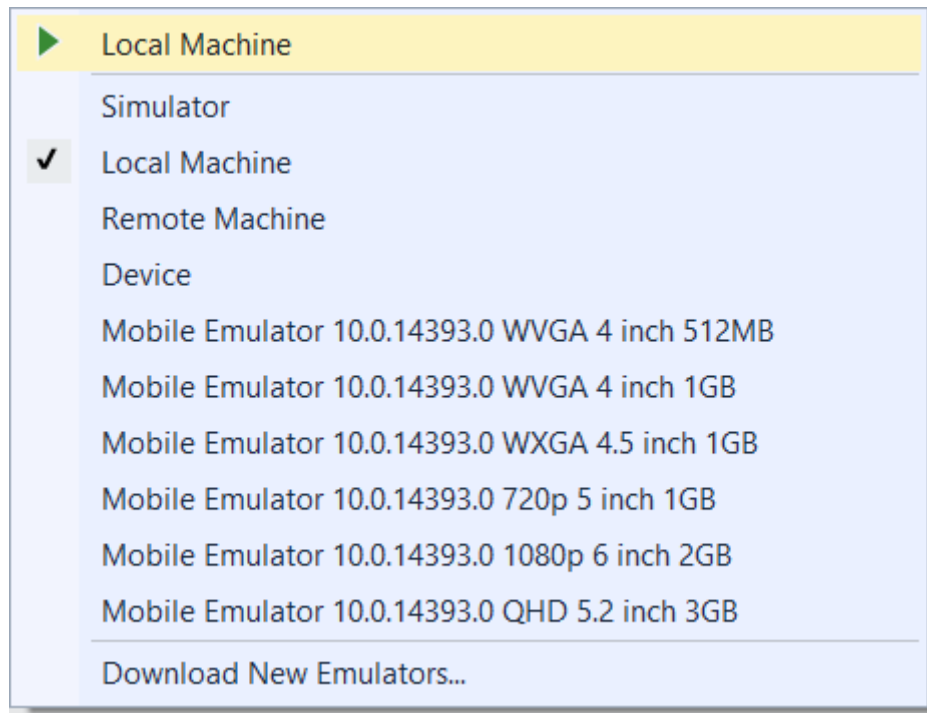


Figure 26 Choix de l'émulateur ou du téléphone

On pourra également choisir la configuration et la plateforme.

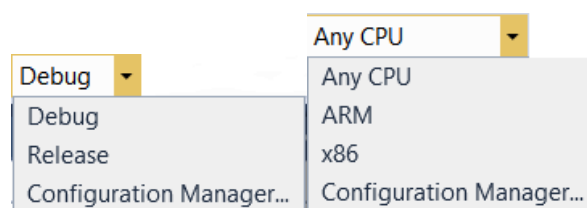




Figure 27 Choix de la configuration et de la plateforme

 Music Sheet Writer	Date de publication	10/07/2016	 Epitech Innovative Project
	Nom du projet	Music Sheet Writer	
	Objet du document	Documentation Technique	
	Nom du chapitre	Versions/Releases	

4. Versions/Releases

4.1. Numérotation des versions

Le projet étant constitué de différents livrable, chacun d'entre eux aura leur propre version. Or, le système appliqué pour la numérotation des versions

La numérotation des versions suit le schéma X.Y.Z, où :



- X est incrémenté pour les releases majeurs incluant de nouvelles fonctionnalités indépendantes de celle déjà existante. Y et Z sont remis à 0.
- Y est incrémenté pour les releases contenant des fonctionnalités additionnelles et complémentaires à celles déjà présentes. Z est remis à 0
- Z est incrémenté pour les releases corrigeant les bugs.

4.2. Release note

4.2.1. Logiciel

Numéro de version	Date de sortie prévue	Description	Fonctionnalités ajoutées
0.1 (Beta)	08/07/16 (sortie)	Cette version est présentée au lab EIP et fait office de version beta du logiciel. Dépendamment des résultats à l'issue de la beta, cette version pourra devenir public courant juillet 2016.	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 Générer une partition par port MIDI
1.0	20/10/16 (sortie)	Cette version implémente toutes les fonctionnalités demandées dans le CDC. Elle corrige les bugs de la version beta et ajoute la génération de partition par port Jack. Elle est présentée à l'occasion du Challenge EIP.	Générer une partition par port Jack : <ul style="list-style-type: none"> - Notes asynchrones
1.0.1	25/11/16 (sortie)	Corrige les bugs liés à la génération de partition par port Jack. Elle est présentée à l'occasion du Trophée Epitech Experience.	
1.1	20/01/16	Améliore la génération de partition par port Jack en intégrant la détection de touches simultanées.	Générer une partition par port Jack : <ul style="list-style-type: none"> - Notes simultanées

Tableau 2 : Plan de release du logiciel

 Music Sheet Writer	Date de publication	10/07/2016	 Epitech Innovative Project
	Nom du projet	Music Sheet Writer	
	Objet du document	Documentation Technique	
	Nom du chapitre	Versions/Releases	

4.2.2. Site Web

Numéro de version	Date de sortie prévue	Description	Fonctionnalités ajoutées
0.1	01/06/16 (sortie)	Cette version du site web présente Music Sheet Writer et inclue un formulaire de contact.	Vitrine : <ul style="list-style-type: none"> - Présentation de Music Sheet Writer - Formulaire de contact
1.0	20/10/16	Cette version donnera accès à l'espace communautaire du site web et permettra le téléchargement du logiciel. Elle est présentée à l'occasion du Challenge EIP.	Plateforme d'achat du logiciel Gestion de compte utilisateur : <ul style="list-style-type: none"> - Création de compte - Connexion/Déconnexion - Modification des informations de compte - Gestion de ses partitions 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 3 : Plan de release du site web

4.2.3. Applications mobiles

Numéro de version	Date de sortie prévue	Description	Fonctionnalités ajoutées
1.0	20/10/2016	Cette version inclura directement toutes les fonctionnalités demandées dans le CDC. Elle est présentée à l'occasion du Challenge EIP.	Gestion de compte utilisateur : <ul style="list-style-type: none"> - Création de compte - Connexion/Déconnexion - Modification des informations de compte - Gestion de ses partitions 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
2.0 (Windows Phone uniquement)	25/11/2016	L'application Windows Phone est disponible en application universelle pour Windows 10.	

Tableau 4 : Plan de release des applications mobiles

Pour le moment, les applications mobiles suivent le même plan de release. Il est possible (voir probable) que nous les séparions si nécessaire. Ainsi, il est possible qu'un version 1.1 sorte sur Android avant que ces mêmes fonctionnalités ne se retrouvent sur iPhone.

 Music Sheet Writer	Date de publication	10/07/2016	 Epitech Innovative Project
	Nom du projet	Music Sheet Writer	
	Objet du document	Documentation Technique	
	Nom du chapitre	<i>Outils de collaborations</i>	

5. Outils de collaborations

5.1. Outils de développement collaboratif

5.1.1. SVN

Nous utilisons un seul serveur SVN pour la gestion de plusieurs livrables. Ces livrables sont les applications mobiles et le site internet.

Chacun des livrable a son propre dossier contenant la même structure :

- Branch
- Tag
- Trunk : Seul le code stable doit être dans ce dossier
- Release

La récupération de ces dépôts est décrite pour chacun des livrables dans le document d'installation.

5.1.2. Gitlab

Nous utilisons avant le même serveur SVN que celui utilisé pour les applications mobiles et le site internet pour la gestion des sources du logiciel.

Nous avons cependant choisi de passer sur un serveur Gitlab et donc un dépôt Git pour la gestion des sources du logiciel. Ce choix a été fait pour plusieurs raisons :

- La gestion des branches de manière plus facile.
- Une intégration d'un bugtracker simple d'utilisation au sein de Gitlab.

Ce choix c'est fait à un moment où tous les membres de l'équipe sont passé sur le développement du logiciel et il a permis de faciliter le développement sur ce dernier.

5.1.3. Bugtracker

Nous utilisons donc le bugtracker intégré de Gitlab pour remonter les différents problèmes du logiciel. Ce dernier nous permet de créer des branches liées aux problèmes remontés ou bien de fermer des tickets de bug en même temps qu'une réparation est effectuée.

5.2. Outils de communication

5.2.1. Slack



Slack est notre principal moyen de communication en interne. Il permet la création de channel que tout un chacun peut joindre ou non. Il permet aussi d'avoir des notifications d'autres applications collaboratives que nous utilisons tels que Google Calendar ou Trello.

5.2.2. Google Calendar

Nous avons mis en place un Google Calendar commun à chacun des membres du projet Music Sheet Writer pour la gestion des réunions faites en interne.

5.2.3. Trello

Trello est un outil de TODO liste collaboratif que nous utilisons pour fixer les objectifs à court et moyen terme à réaliser. Il permet aussi de voir l'avancée du développement de chacun des livrables et de savoir qui travaille sur quoi.

 Music Sheet Writer	Date de publication	10/07/2016	 Epitech Innovative Project
	Nom du projet	Music Sheet Writer	
	Objet du document	Documentation Technique	
	Nom du chapitre	<i>Annexes</i>	

6. Annexes

- Document d'installation : [2017_ID2_musicsheetwriter.pdf](#)
- Document de pré-requis : [2017_PR2_musicsheetwriter.txt](#)
- Architectural Assessments 2 : [2017_AA2_musicsheetwriter.pdf](#)
- Test Strategies : [2017_TS_musicsheetwriter.pdf](#)