

Jonathan Racaud

Music Sheet Writer

Music Sheet Writer

Documentation Technique

Objectifs du document

Résumé

Glossaire

***– G –***

Gradle : Gradle est un 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.

***– R –***

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

Description du document

|  |  |
| --- | --- |
| Titre | Music Sheet Writer : Documentation Technique |
| Date de création | 08/04/2016 |
| Date de publication | 08/04/2016 |
| Auteur | Jonathan Racaud |
| Responsable | Jonathan Racaud |
| E-mail | musicsheetwriter\_2017@labeip.epitech.eu |
| Sujet | Documentation Technique |
| Version du modèle | 1.0 |

Tableau des révisions

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Auteur | Section(s) | Commentaire |
| 04/08/2016 | Jonathan Racaud | Toutes | Création du document |
| 21/04/2016 | Jeremy Harrault | 3. Application Android | Ajout partie Android |
| 23/04/2016 | Jeremy Harrault | 3. Application Android | Ajout installation « espresso » + problème de consistance des noms mentionnés. |
| 29/04/2016 | Jeremy Harrault | 3. Application Android | Ajout des problèmes connus et axes d’amélioration |
| 30/04/2016 | Jeremy Harrault | 3. Application Android | Ajout stratégie de release et structure et presentation des composants |

Sommaire

[1. Logiciel 1](#_Toc447872840)

[2. Site Web 1](#_Toc447872841)

[2.1. API Rest 1](#_Toc447872842)

[2.2. Site vitrine et espace communautaire 1](#_Toc447872843)

[3. Application Android 1](#_Toc447872844)

[4. Application iOS 1](#_Toc447872845)

[5. Application Windows Phone 1](#_Toc447872846)

Liste des Tableaux

**Aucune entrée de table d'illustration n'a été trouvée.**

Liste des Figures

**Aucune entrée de table d'illustration n'a été trouvée.**

# Introduction

# Le projet

# Les livrables

## Logiciel

### Norme de codage

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

### Implémentation

### Tests

### Historique des modifications

### Problèmes existant

## Site Web

### API Rest

### Site vitrine et espace communautaire

## Application Android

### 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, icones ou encore fichiers de traduction. Le 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 *activities* 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.

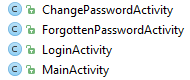
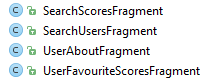
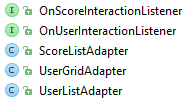


Figure 3.1. Exemples de noms de classes pour l'application Android

L’organisation des ressources est imposée par la manière dont Android gère les ressources d’une application. A savoir, 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 accroit la portabilité et la compatibilité de l’interface utilisateur de l’application sur d’autres téléphone.

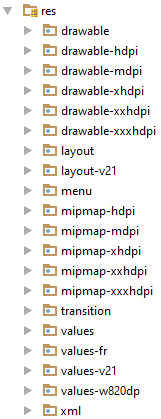


Figure 3.2. Arborescence des ressources pour l'application Android

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*.

C:\Users\Jeremy\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Untitled Diagram (2).png

Figure 3.3. Hierarchisation des fichiers gradle pour l'application Android

### Structure et présentation des composants

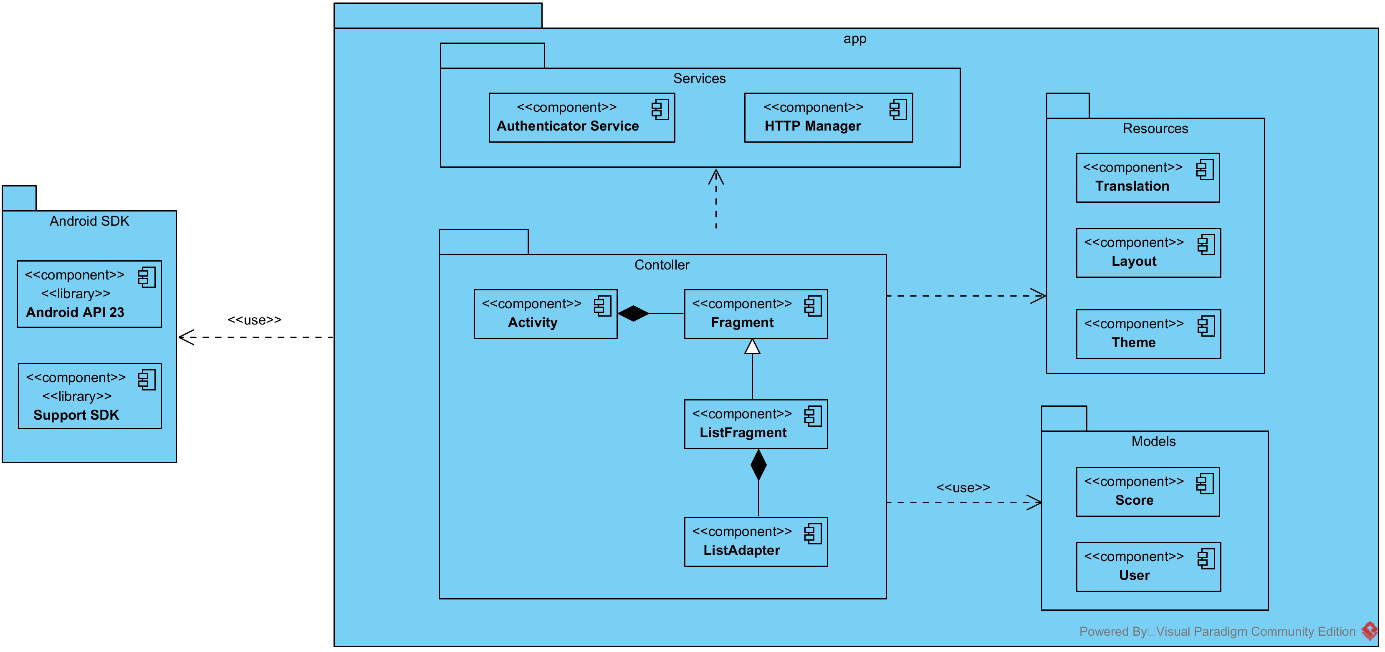


Figure 3.4. Structure des composants de l'application Android

|  |  |
| --- | --- |
| Nom des composants | Description est 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-activity ». 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 comprends, 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’élément 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 facilité leur travail. |
| Model (package) | Il s’agit de la définitions des entités que les « controllers » seront amenés à manipulées. |
| 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 3.1 Présentation des composants de l'application Android.

### Installation et lancement de l’application

La procédure de récupération du code source et d’installation de l’environnement est détaillé dans le ID1.

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

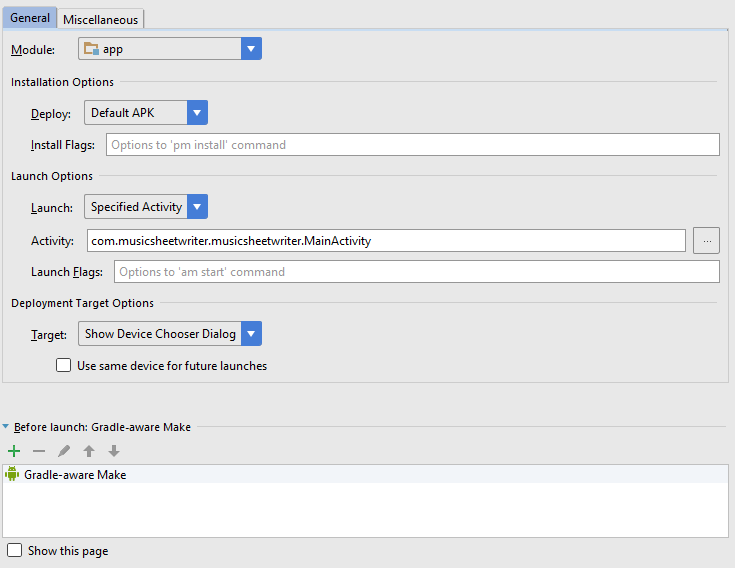


Figure 3.5. Déclaration d'une configuration de lancement dans Android Studio

Ensuite, il suffit de sélectionner **Run 🡺 Run ‘app’** ou tout simplement cliquer sur C:\Users\Jeremy\AppData\Local\Microsoft\Windows\INetCache\Content.Word\2016-04-23_01h41_34.png pour lancer l’application. Il suffira juste de choisir sur quel émulateur ou téléphone branché à l’ordinateur l’application doit se lancer.

### Tests

Il existe différents types de tests sur Android. Ces types sont définit 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*.

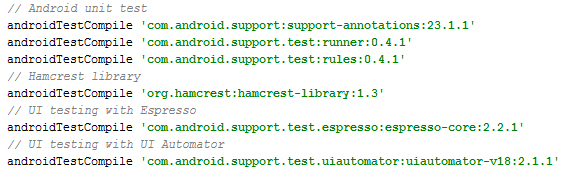


Figure 3.6. Liste des bibliothèques à inclure dans le fichier gradle pour l'application Android

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*.

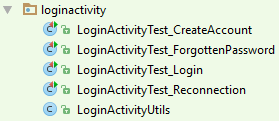


Figure 3.7. Arborescence des classes de test

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 :



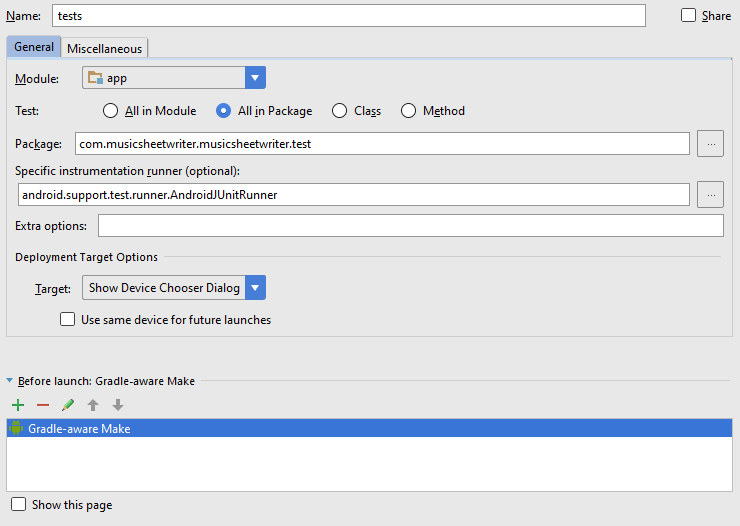


Figure 3.8. Déclaration d'une configuration de lancement de tests dans Android Studio

### ¨Problèmes connus et axes d’amélioration

Actuellement, sans accès à internet, l’application est inutile car aucune information ne peut être récupérer. 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.

### Gestion des releases

La stratégie des releases est la même que celle définie dans le Test Stratégie document (cf. Annexe).

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

* X est incrémenté pour les releases majeurs. Y est Z sont remis à 0.
* Y est incrémenté pour les releases contenant des fonctionnalités additionnelles complémentaires à celles déjà présentes. Z est remis à 0
* Z est incrémenté pour les releases corrigeant les bugs.

## Application iOS

## Application Windows Phone

# Gestion des sources et bug

## SVN

## Git

## Bugtracker

# Annexes

* API
* AA2
* TS
* Autres ressources