

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

Music Sheet Writer

Music Sheet Writer

Bilan Architecture

Objectifs du document

Résumé

Music Sheet Writer est un large projet contenant différents livrables : un logiciel d’édition de partition (le projet principal), un site internet pour la présentation du projet comprenant aussi un espace communautaire et des applications mobiles pour iOS, Android et Windows Phone permettant l’accès à cet espace communautaire.

Ce document a pour but de représenter l’architecture du projet Music Sheet Writer. Nous y décrivons les différents cas d’utilisations des différents projets et détaillons ensuite ces cas d’utilisation en décrivant leurs implémentations et les différents processus leur étant associés.

Pour des raisons de confort et de facilité de lecture, nous avons volontairement omis la description de certains processus jugés simples et dont la représentation à l’aide de diagramme n’aurait pas été pertinent (déconnexion de son compte utilisateur, changement de tempo sur une partition…).

Nous décrivons aussi l’architecture globale de Music Sheet Writer en présentant les différents projets et les interactions qu’il y a entre les applications mobiles le site internet.

L’interaction entre ces différents projets se fait à l’aide au travers l’utilisation d’une API Rest développée spécifiquement pour le projet. Cette API est utilisée à la fois par les applications mobiles, mais aussi par l’espace communautaire du site internet (le rôle de cette API est la gestion de cet espace communautaire). Nous utilisons d’ailleurs les termes API Rest et site internet de manière interchangeable au sein de ce document pour parler de l’API Rest.

Vous trouverez aussi en annexe les liens vers le cahier des charges du projet ainsi que le document de description de l’API Rest.

Glossaire

***– A –***

**API (Application Programming Interface)** : une API est une interface pour langages de programmation, matérialisées par des primitives, permettant à une application d'accéder à des programmes système pour, par exemple, communiquer ou extraire des données.

***– B –***

**Bundle** : Un bundle est un dossier composé de fichiers (PHP, CSS, images, etc) qui implémente une fonctionnalité ou application web spécifique (blog, forum, etc).

***– 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 qu’elle connexion fiable. Les clients HTTP les plus connus sont les navigateurs web permettant aux utilisateurs d’accéder un serveur contenant des données.

***– J –***

**Jack** : Jack est une interface permettant de faire transiter un signal audio entre deux appareils comme un téléphone et des écouteurs.

**JSON** : JSON (JavaScript Object Notation) est un format de données textuelles dérivé de la notation des objets du langage Javascript. Il permet de représenter de l’information structurée. Un document JSON a pour fonction de représenter de l'information accompagnée d'étiquettes permettant d'en interpréter les divers éléments, sans aucune restriction sur le nombre de celles-ci.

***– M –***

**MVC** : Le Modèle-Vue-Controller est un patron d’architecture logicielle permettant de séparer les différents composants d’un logiciel. Ces composants étant l’interface utilisateur (Vue), les données manipulées (Modèle) et la logique du logiciel (Contrôleur).

**MVVM** : Pour Modèle-Vue-VueModèle est un patron d’architecture logicielle permettant de séparer les différents composants d’un logiciel à la manière d’un MVC.

***– R –***

**Route** : Une route est une URL.

***– S –***

**Symfony** : Symfony est un framework PHP.

***– T –***

**Twig** : Twig est un moteur de template PHP qui est directement intégré dans le framework Symfony.

***– U –***

**URL** : URL (de l’anglais Uniform Resource Locator, littéralement « localisateur uniforme de ressource »), auquel se substitue informellement le terme adresse web, désigne une chaîne de caractères utilisée pour adresser les ressources web: document HTML, image, son ou une page web.

***– X –***

**Xcode** : Xcode est un environnement de dévelopement disponible sur Mac OS pour le développement d’applications iOS.

Description du document

|  |  |
| --- | --- |
| Titre | Music Sheet Writer : Bilan Architecture |
| Date de création | 21/12/2015 |
| Date de publication | 10/01/2016 |
| Auteur | J. Racaud;A. Simon;J. Harrault;J. Blondeel;S. Daguenet;F. Corradin |
| Responsable | Jonathan Racaud |
| E-mail | musicsheetwriter\_2017@labeip.epitech.eu |
| Sujet | Bilan Architecture |
| Version du modèle | 2.0 |

Tableau des révisions

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Auteur | Section(s) | Commentaire |
| 21/12/2015 | Jeremy HARRAULT | Toutes | Création du document à partir de la version 1.0 |
| 22/12/2015 | Julien BLONDEEL | Vue global du projet | Réalisation des diagrammes |
| 29/12/2015 | Jeremy HARRAULT | 7.4 Les applications mobiles – Android | Ajout des diagrammes |
| 31/12/2015 | Jeremy HARRAULT | 7.4 Les applications mobiles – Android | Ajout des explications textuelles |
| 31/12/2015 | Antoine Simon | 7.3 Les applications mobiles – iPhone | Ajout des diagrammes et de leurs explications |
| 02/01/2016 | Florian CORRADIN | 4.1 & 4.2 | Rédaction |
| 03/01/2016 | Simon DAGUENET | 5 Vue processus | Diagrammes de séquences API |
| 04/01/2016 | Jonathan RACAUD | Introduction | Rédaction |
| 04/01/2016 | Florian CORRADIN | 7.1 Logiciel | Rédaction de la partie |
| 05/01/2016 | Simon DAGUENET | 7.4 Les applications mobiles – Windows Phone | Ajout des diagrammes avec explications |
| 06/01/2016 | Julien BLONDEEL | 5 Vue processus | Diagrammes de séquence du logiciel |
| 06/01/2016 | Jonathan RACAUD | 7.2 Le site internet et l’API | Rédaction de la partie |
| 07/01/2016 | Jonathan RACAUD | 8 Vue Modèle | Rédaction de la partie |
| 08/01/2016 | Florian CORRADIN | 4.1 & 4.2 | Modification des diagrammes |
| 09/01/2016 | Simon DAGUENET | Glossaire | Rédaction |
| 09/01/2016 – 10/01/2016 | Jonathan Racaud | Toutes | Correction orthographique |

Sommaire

[1. Introduction 1](#_Toc440206912)

[1.1. Rappel de l’EIP 1](#_Toc440206913)

[1.2. Contexte et périmètre du projet 1](#_Toc440206914)

[2. Représentation de l’architecture globale 2](#_Toc440206915)

[3. Vue globale du projet 3](#_Toc440206916)

[3.1. Cas d’utilisation principaux 3](#_Toc440206917)

[3.2. Cas d’utilisation détaillés 4](#_Toc440206918)

[4. Vue logique de l’application 8](#_Toc440206919)

[4.1. Le logiciel 8](#_Toc440206920)

[4.2. Le site internet 9](#_Toc440206921)

[4.3. Les applications mobiles 11](#_Toc440206922)

[5. Vue processus 12](#_Toc440206923)

[5.1. Le logiciel 12](#_Toc440206924)

[5.2. L’API 15](#_Toc440206925)

[6. Vue déploiement 19](#_Toc440206926)

[7. Implémentation 20](#_Toc440206927)

[7.1. Le logiciel 20](#_Toc440206928)

[7.2. Le site internet et l’API 24](#_Toc440206929)

[7.3. Les applications mobiles – iPhone 31](#_Toc440206930)

[7.4. Les applications mobiles – Android 33](#_Toc440206931)

[7.5. Les applications mobiles – Windows Phone 40](#_Toc440206932)

[8. Vue données 42](#_Toc440206933)

[9. Annexes 43](#_Toc440206934)

Liste des Figures

[Figure 1. Architecture globale de Music Sheet Writer 2](#_Toc440206935)

[Figure 2. Cas d'utilisation global du logiciel 3](#_Toc440206936)

[Figure 3. Cas d'utilisation global du site internet 3](#_Toc440206937)

[Figure 4. Cas d'utilisation global des applications mobiles 4](#_Toc440206938)

[Figure 5. Cas d'utilisation logiciel : éditer une partition 4](#_Toc440206939)

[Figure 6. Cas d'utilisation logiciel : lire une partition 5](#_Toc440206940)

[Figure 7. Cas d'utilisation logiciel : gérer un projet 5](#_Toc440206941)

[Figure 8. Cas d'utilisation site internet & applications mobiles : accéder à la communauté 6](#_Toc440206942)

[Figure 9. Cas d'utilisation site internet & applications mobiles : gérer son compte utilisateur 7](file:///C:\Users\jonat\Documents\Epitech\EIP\Documentation\AA2\2017_AA2_musicsheetwriter.docx#_Toc440206943)

[Figure 10. Cas d'utilisation site internet : accéder à la vitrine 7](#_Toc440206944)

[Figure 11. Vue logique du logiciel 8](#_Toc440206945)

[Figure 12. Diagramme de déploiement du serveur 9](#_Toc440206946)

[Figure 13. Vue logique du site internet 10](#_Toc440206947)

[Figure 14. Vue logique des applications mobiles 11](#_Toc440206948)

[Figure 15. Diagramme de séquence logiciel : créer un projet 12](#_Toc440206949)

[Figure 16. Diagramme de séauence logiciel : fermer un projet 12](#_Toc440206950)

[Figure 17. Diagramme de séquence logiciel : sauvegarder un projet 13](#_Toc440206951)

[Figure 18. Diagramme de séquence logiciel : placer un élément musical 13](#_Toc440206952)

[Figure 19. Diagramme de séquence logiciel : supprimer un élément musical 14](#_Toc440206953)

[Figure 20. Diagramme de séquence logiciel : générer une partition depuis un instrument 14](#_Toc440206954)

[Figure 21. Diagramme de séquence logicel : démarrer la lecture 14](#_Toc440206955)

[Figure 22. Diagramme de séquence logiciel : arrêter la lecture 15](#_Toc440206956)

[Figure 23. Diagramme de séquence API : création de compte 15](#_Toc440206957)

[Figure 24. Diagramme de séquence API : connexion 16](#_Toc440206958)

[Figure 25. Diagramme de séquence API : changement d'addresse email 17](#_Toc440206959)

[Figure 26. Diagramme de séquence API : réinitialisation du mot de passe 18](#_Toc440206960)

[Figure 27. Diagramme de déploiement général 19](#_Toc440206961)

[Figure 28. Diagramme de package global logiciel 20](#_Toc440206962)

[Figure 29. Diagramme de classe logiciel : couche présentation 20](#_Toc440206963)

[Figure 30. Diagramme de classe logiciel : lecture de partition 21](#_Toc440206964)

[Figure 31. Diagramme de classe logiciel : gestion de projet 22](#_Toc440206965)

[Figure 32. Diagramme de classe logiciel : gestion de partitions 22](#_Toc440206966)

[Figure 33. Diagramme de classe logiciel : modèle 23](#_Toc440206967)

[Figure 34. Diagramme de package global site internet 24](#_Toc440206968)

[Figure 35. Diagramme de classe site internet : component APIBundle 25](file:///C:\Users\jonat\Documents\Epitech\EIP\Documentation\AA2\2017_AA2_musicsheetwriter.docx#_Toc440206969)

[Figure 36. Diagramme de classe site internet : controller APIBundle 26](#_Toc440206970)

[Figure 37. Diagramme de classe site internet : exceptions APIBundle 26](#_Toc440206971)

[Figure 38. Diagramme de classe site internet : services APIBundle 27](#_Toc440206972)

[Figure 39. Diagramme de classe site internet : entités APIBundle 28](#_Toc440206973)

[Figure 40. Diagramme de classe site internet : ressources MSWBundle 29](#_Toc440206974)

[Figure 41. Diagramme de classe site internet : controllers MSWBundle 30](#_Toc440206975)

[Figure 42. Vue globale iPhone 31](#_Toc440206976)

[Figure 43. Diagramme de classe iPhone 32](#_Toc440206977)

[Figure 44. Diagramme de package global Android 33](#_Toc440206978)

[Figure 45. Vue applicative du package "Activité" 34](#_Toc440206979)

[Figure 46. Vue applicative du package "Authenticateur" 35](#_Toc440206980)

[Figure 47. Vue applicative du package "Fragment" 36](#_Toc440206981)

[Figure 48. Vue applicative du package "Fragment" et "List Adapter" (Scores) 37](#_Toc440206982)

[Figure 49. Vue applicative du package "Fragment" et "List Adapter" (Subscribers) 37](#_Toc440206983)

[Figure 50. Vue applicative du package "Gestionnaire HTTP" 38](#_Toc440206984)

[Figure 51. Vue applicative du package "Partition" 39](#_Toc440206985)

[Figure 52. Vue applicative du package "Utilisateur" 39](#_Toc440206986)

[Figure 53. Diagramme de package global Windows Phone 40](#_Toc440206987)

[Figure 54. Vue applicative Windows Phone 41](#_Toc440206988)

[Figure 55. Modèle de la base de données 42](file:///C:\Users\jonat\Documents\Epitech\EIP\Documentation\AA2\2017_AA2_musicsheetwriter.docx#_Toc440206989)

# Introduction

## Rappel 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 weekend, 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’entreprenariat 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.

## Contexte et périmètre du projet

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. Laissant 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.

# Représentation de l’architecture globale

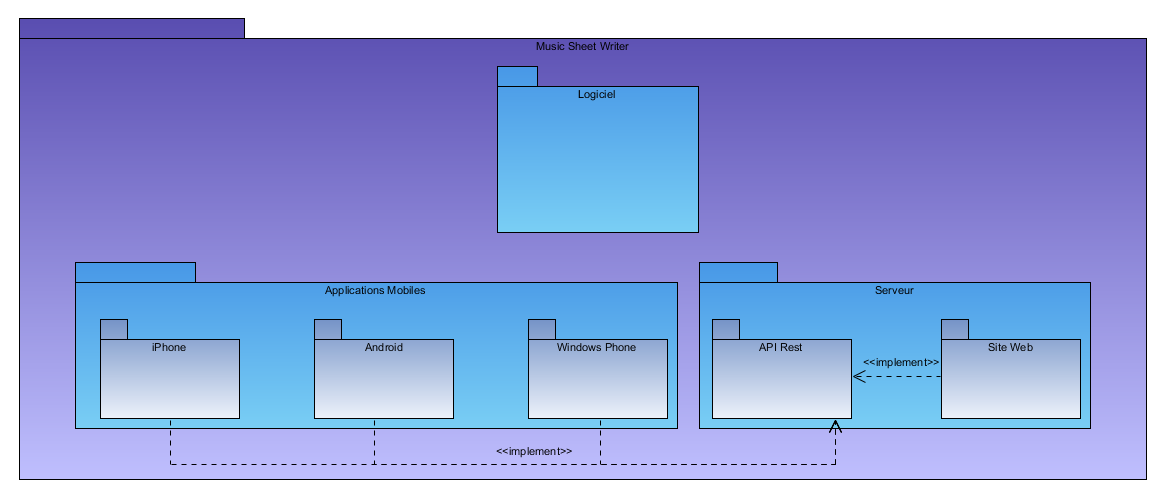


Figure 1. Architecture globale de Music Sheet Writer

Le logiciel est le principal livrable du projet Music Sheet Writer. Il est aussi le seul qui ne dépend pas des autres projets pour sa réalisation.

Le site web est découpé en deux projets. Le premier étant le site vitrine avec un espace communautaire. Le second est le développement d’une API Rest pour cet espace communautaire.

Cette API est utilisée par les applications mobiles, qui donnent les mêmes fonctionnalités que l’espace communautaire du site internet.

# Vue globale du projet

## Cas d’utilisation principaux

### Le logiciel

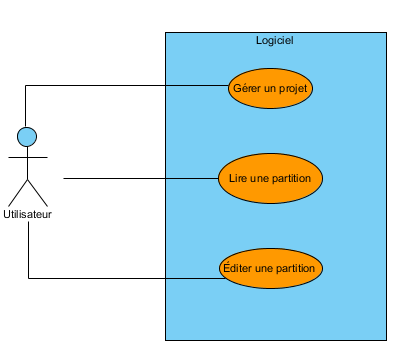


Figure . Cas d'utilisation global du logiciel

### Le site internet

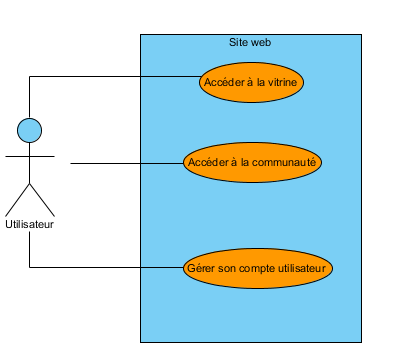


Figure . Cas d'utilisation global du site internet

### Les applications mobiles

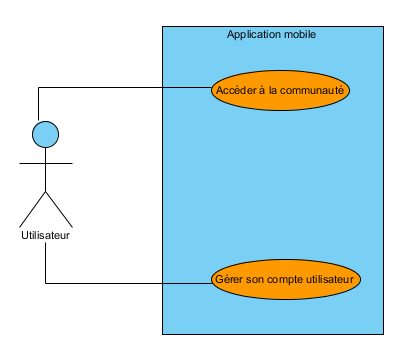


Figure . Cas d'utilisation global des applications mobiles

## Cas d’utilisation détaillés

### Le logiciel

#### Editer une partition

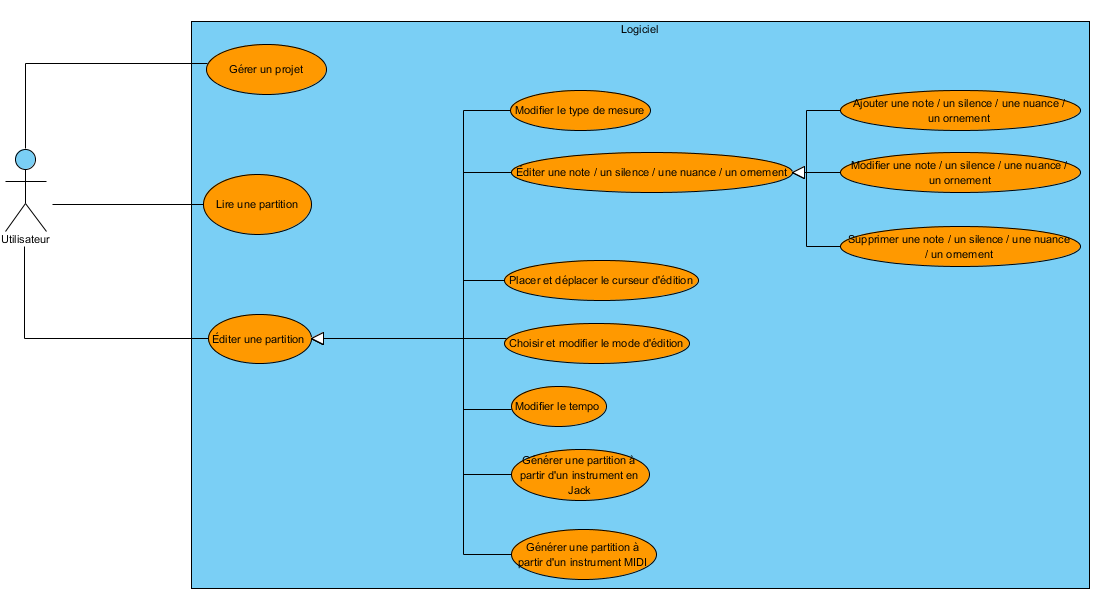


Figure . Cas d'utilisation logiciel : éditer une partition

#### Lire une partition

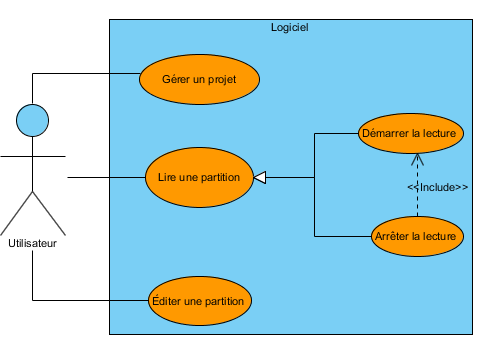


Figure . Cas d'utilisation logiciel : lire une partition

#### Gérer un projet

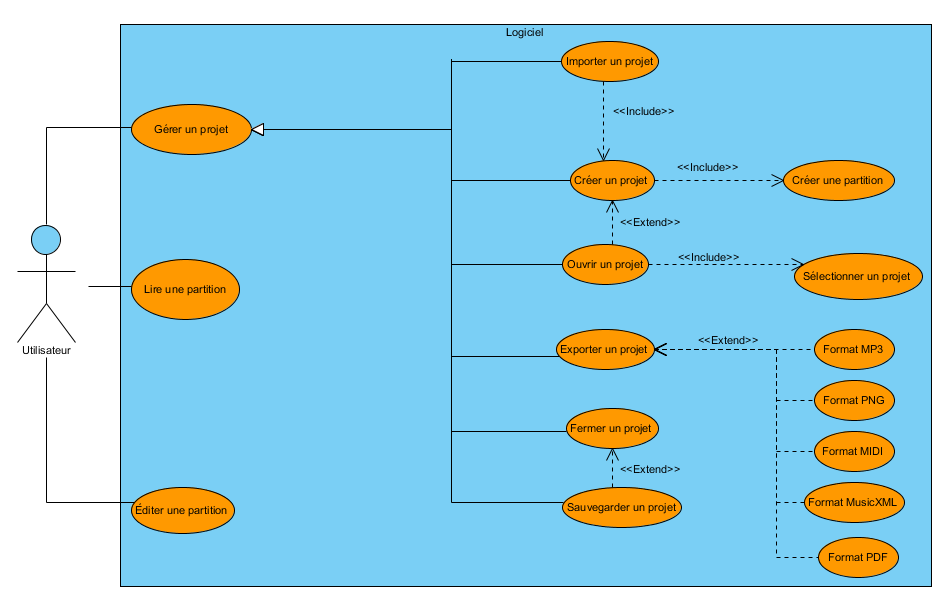


Figure . Cas d'utilisation logiciel : gérer un projet

### Le site internet & application mobile

#### Accéder à la communauté

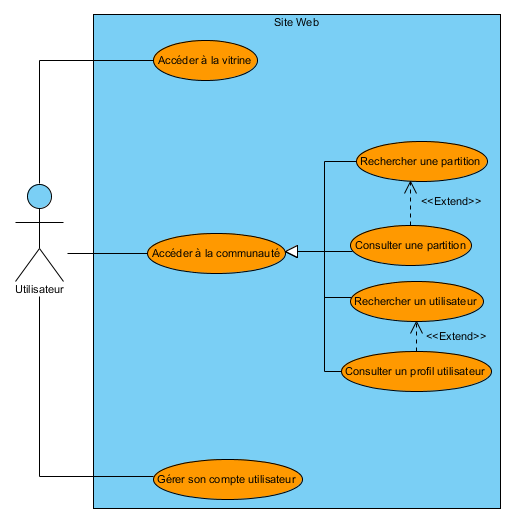


Figure . Cas d'utilisation site internet & applications mobiles : accéder à la communauté

#### Gérer son compte utilisateur

Figure 9. Cas d'utilisation site internet & applications mobiles : gérer son compte utilisateur

#### Accéder à la vitrine (site internet uniquement)

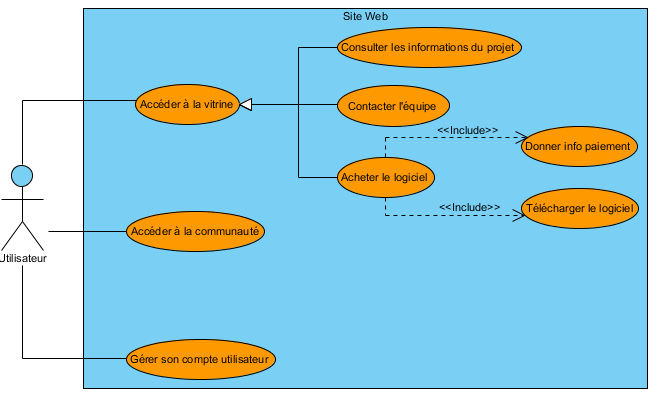


Figure . Cas d'utilisation site internet : accéder à la vitrine

# Vue logique de l’application

Pour chacun des livrables, la vue logique est présentée grâce à un diagramme de composant ainsi qu’une description textuelle mettant en avant les composants essentiels du diagramme.

## Le logiciel

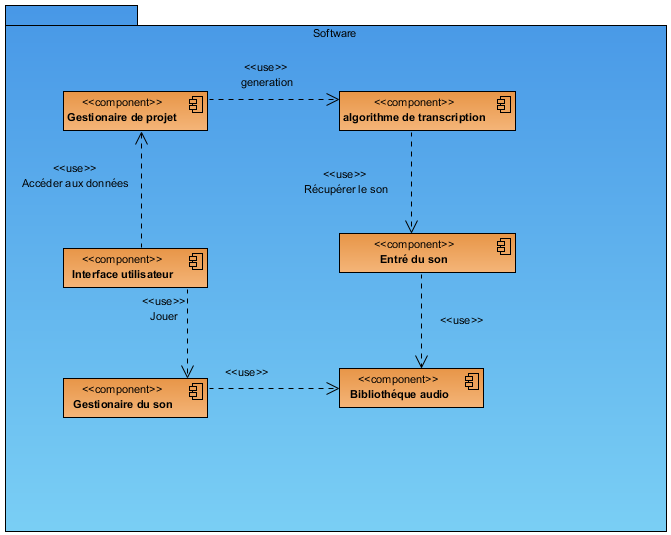


Figure . Vue logique du logiciel

Le projet regroupe toutes les données nécessaires pour l’édition de partition, l’export de partition, la lecture, l’affichage et la suppression de la partition.

L’interface utilisateur permet à l’utilisateur du logiciel d’avoir un aperçu de la partition sur laquelle il travail, de la modifier et de gérer l’ensemble de ses projets.

Le composant « Gestionnaire du son » va gérer toute les sorties sonore du logiciel notamment lors de la lecture d’une partition.

Le composant « Algorithme de transcription » s’occupe lui de transcrire les données numériques reçu depuis l’instrument dans un format de notation musicale.

Le composant « Entrée du son » quand à lui permet de récupérer les signaux sonore ou donnée MIDI envoyé depuis un instrument de musique branché à l’ordinateur.

## Le site internet

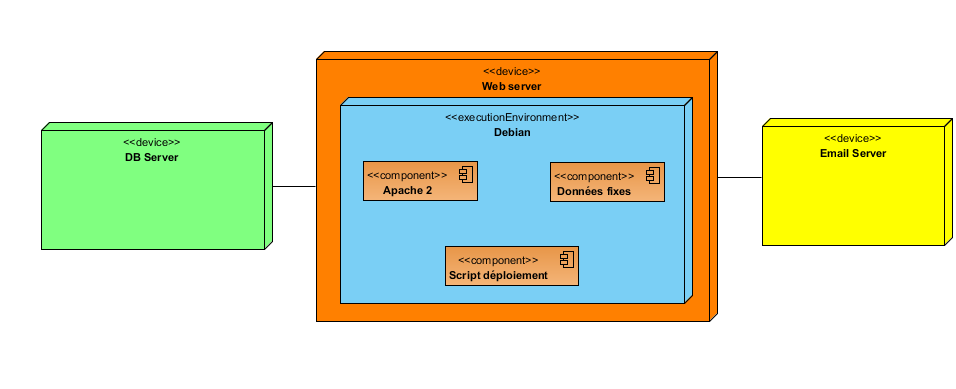


Figure . Diagramme de déploiement du serveur

Le site web est hébergé sur un serveur linux et fonctionne avec le serveur web Apache 2.

La structure logicielle Symfony est utilisée pour regrouper toutes les interactions entre le système de mail, la base de données et l’application.

Des données fixes sont aussi utilisées pour charger les comptes administrateurs en base lors du déploiement de l’application en environnement de production.

La base de données regroupe toutes les données utilisateur, comprend les partitions, informations des utilisateurs et données de l’espace communautaire. La base de données peut être du type MySQL ou PostgreSQL.

Le serveur d’application héberge le site internet vitrine et l’API Rest. Un serveur apache 2 est utilisé pour permettre le fonctionnement du Framework Symfony.

Le serveur mail est utilisé pour envoyer des mails depuis le site internet vers les utilisateurs ou pour permettre à ces derniers de contacter l’équipe de développement.

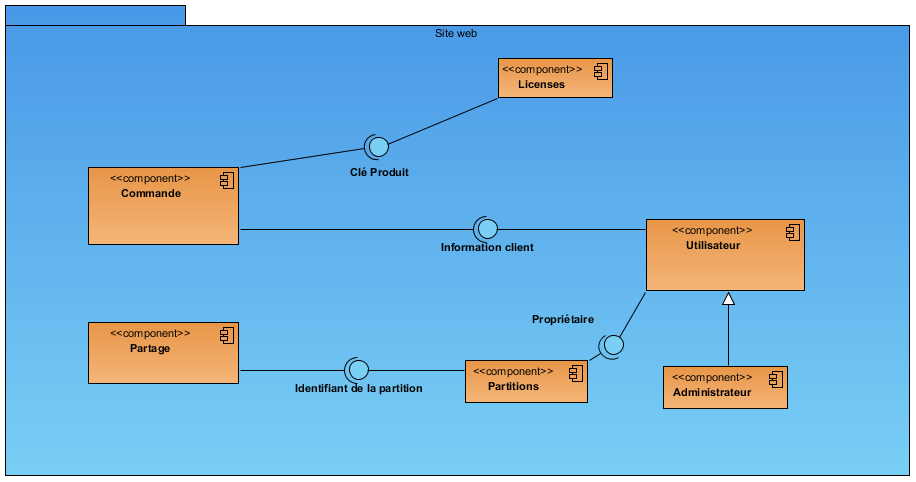


Figure . Vue logique du site internet

Les clients du site internet sont représentés par le composant utilisateur qui est utilisé par héritage pour les administrateurs qui sont des utilisateurs avec des privilèges.

Le partage de partition (composant partage) permet l’accès aux partitions pour un client connecté. Le composant partition regroupe l’ensemble des actions qui peuvent être effectués par l’utilisateur sur la gestion de ces partitions.

## Les applications mobiles

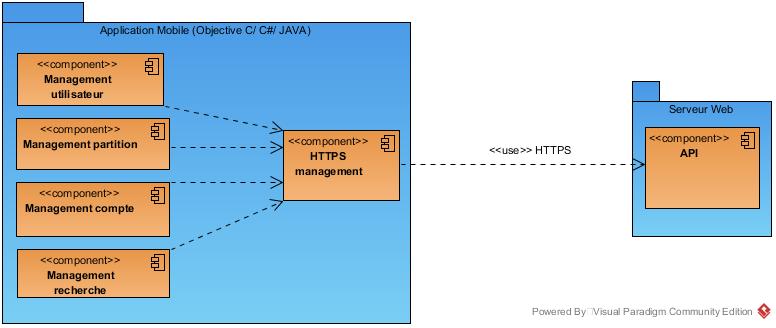


Figure . Vue logique des applications mobiles

L’architecture et la relation entre les différents composants des applications mobiles sont les mêmes sur les plateformes Android, iOS, Windows Phone.

Chacune de ces applications sera développées dans son langage natif, Java pour Android, Objective C pour IOS et C# pour Windows Phone.

Le rôle des applications mobiles étant de donner l’accès à l’aspect communautaire du projet, elles sont découpées en plusieurs composants gérant les différents aspects de ce projet. Ces composants sont :

* Management utilisateur
* Management partition
* Management compte
* Management recherche

Tous ces composants s’articulent autour du composant HTTPS Management. Ce dernier s’occupe de la communication avec le serveur et l’API Rest mise en place sur ce dernier.

# Vue processus

## Le logiciel

### Créer un projet

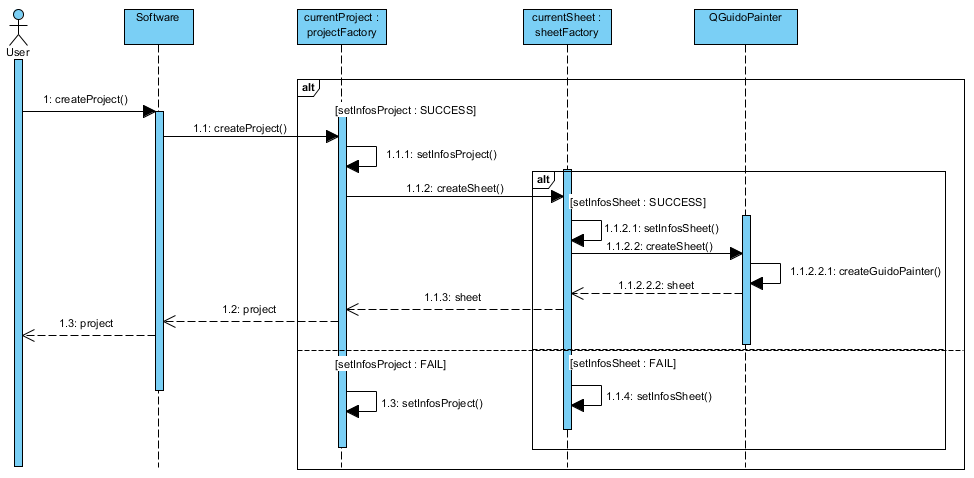


Figure . Diagramme de séquence logiciel : créer un projet

### Fermer un projet

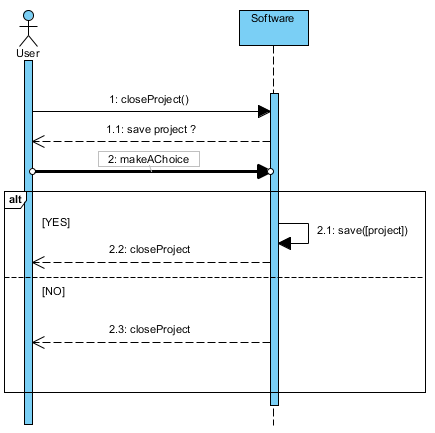


Figure . Diagramme de séauence logiciel : fermer un projet

### Sauvegarder un projet

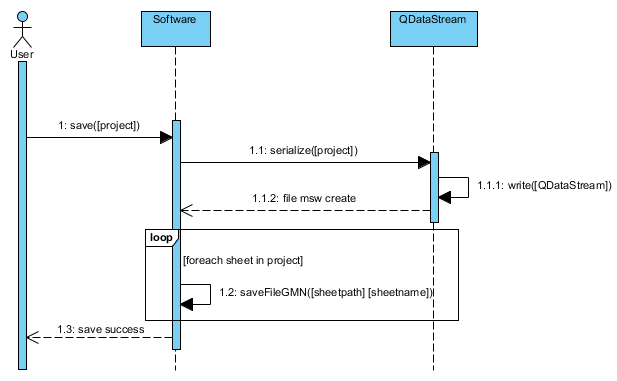


Figure . Diagramme de séquence logiciel : sauvegarder un projet

### Placer un élément musical

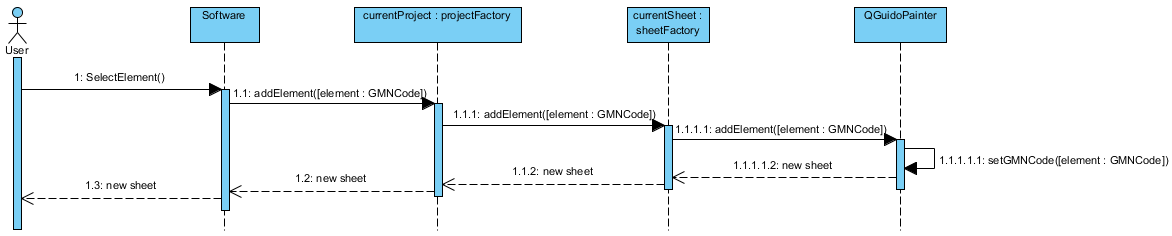


Figure . Diagramme de séquence logiciel : placer un élément musical

### Supprimer un élément musical

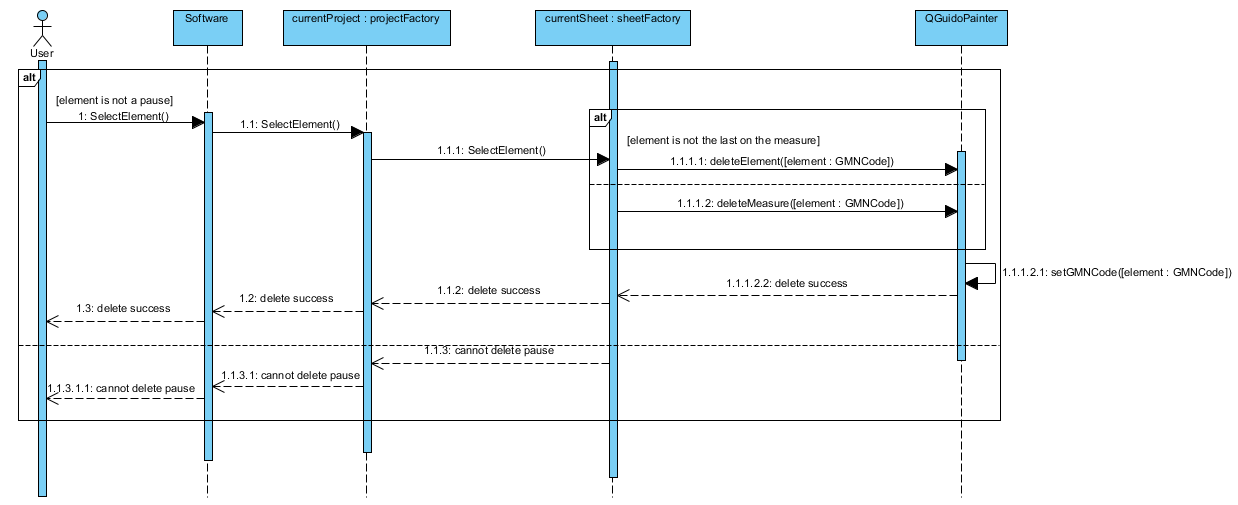


Figure . Diagramme de séquence logiciel : supprimer un élément musical

### Générer une partition depuis un instrument musique

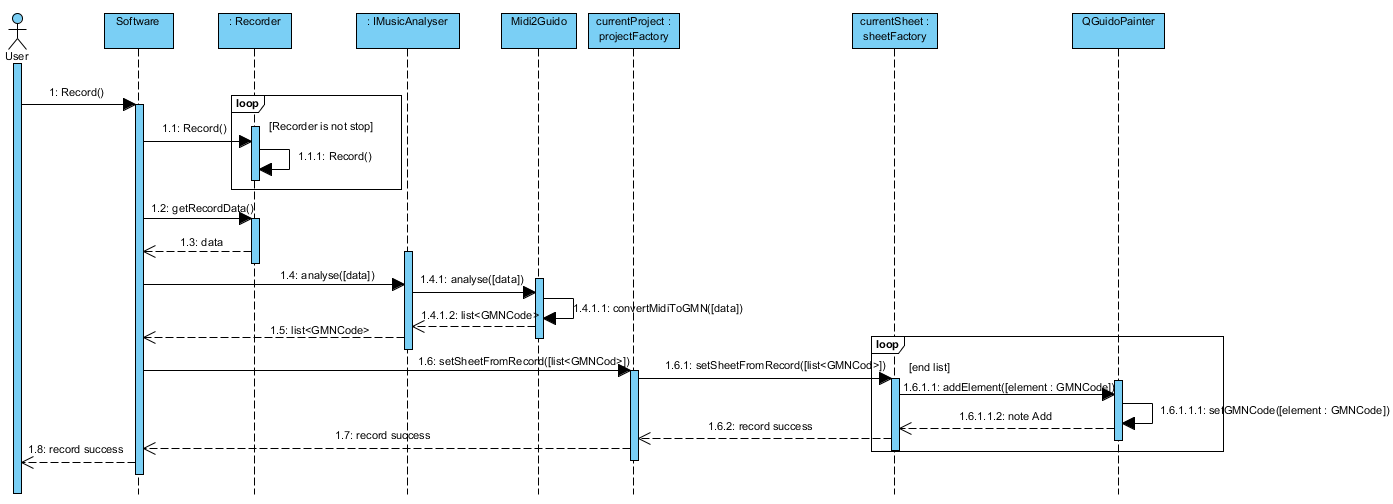


Figure . Diagramme de séquence logiciel : générer une partition depuis un instrument

### Démarrer la lecture

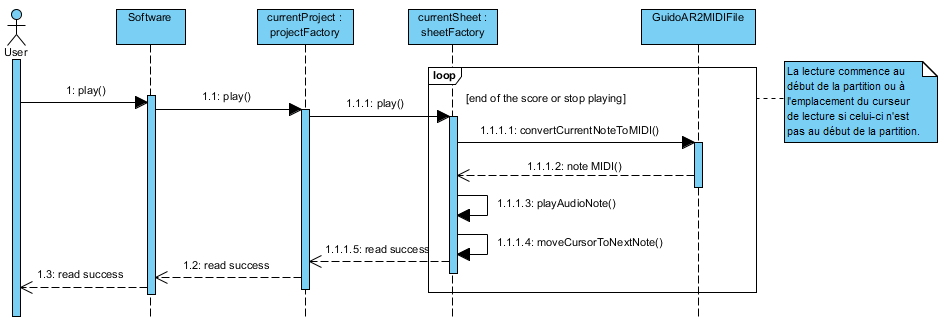


Figure . Diagramme de séquence logicel : démarrer la lecture

### Arrêter la lecture

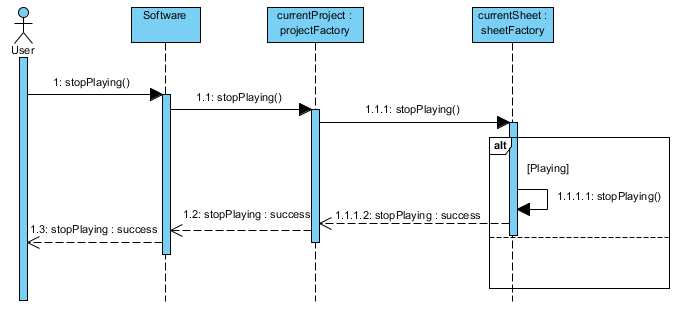


Figure . Diagramme de séquence logiciel : arrêter la lecture

## L’API

### Création de compte

#### Description

La création d’un compte est soumise à plusieurs contraintes d’unicité. En effet, le nom d’utilisateur ainsi que l’adresse email ne doivent appartenir à aucun autre utilisateur. Si tel est le cas, un nouvel utilisateur est créé et un email de confirmation d’adresse mail est envoyé.

#### Vue dynamique

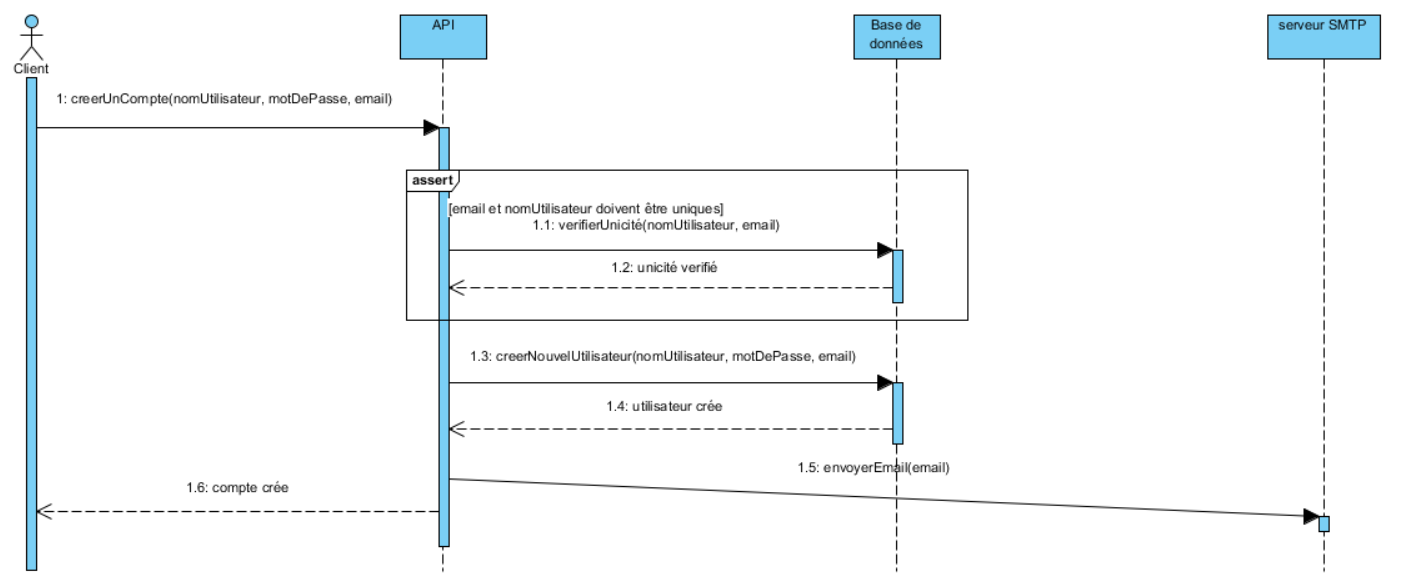


Figure . Diagramme de séquence API : création de compte

### Connexion

#### Description

La connexion d’utilisateur se fait grâce au nom d’utilisateur et au mot de passe. Le compte doit être actif – c’est-à-dire qu’il ne doit pas avoir été fermé précédemment – pour pouvoir se connecter avec.

#### Vue dynamique

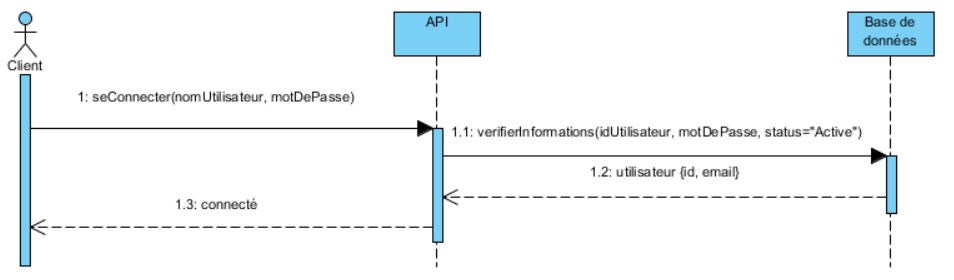


Figure . Diagramme de séquence API : connexion

### Changement d’adresse email

#### Description

Lors d’un changement d’adresse email, la nouvelle adresse ne doit pas déjà être attribué à un compte existant. SI elle est unique, alors un mail de vérification est envoyé à l’ancienne adresse email et un mail de confirmation à la nouvelle. L’utilisateur devra donc validé sa nouvelle adresse email.

#### Vue dynamique

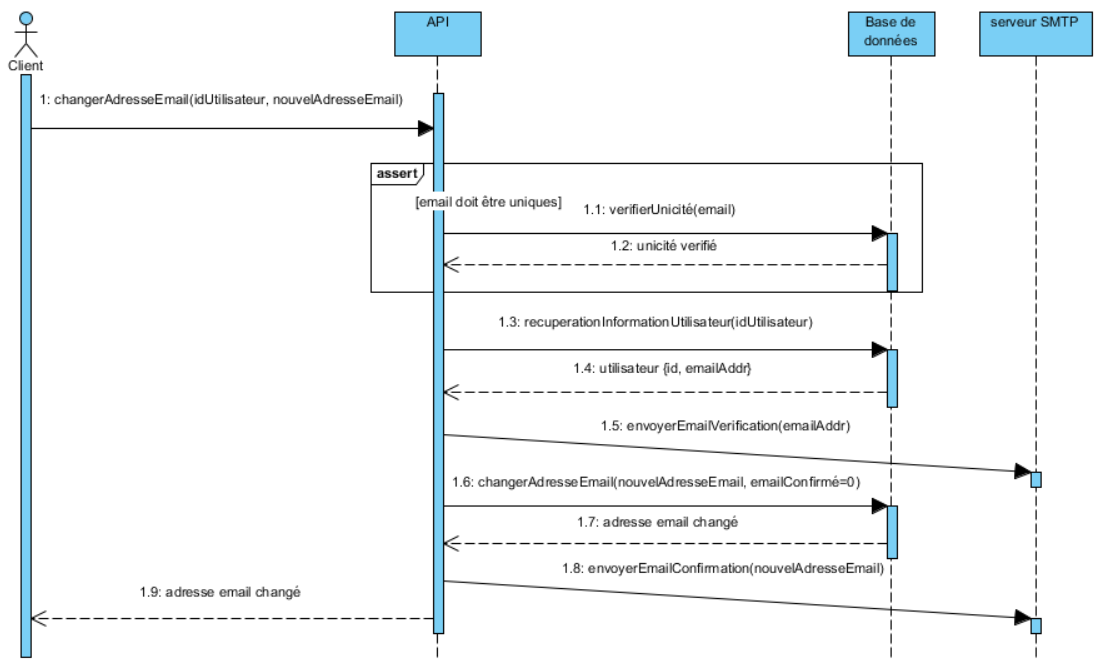


Figure . Diagramme de séquence API : changement d'addresse email

### Réinitialisation du mot de passe

#### Description

La réinitialisation du mot de passe est un processus appliqué lorsqu’un utilisateur ne se souvient plus de son mot de passe. Ce processus se fait en deux temps. Tout d’abord, l’utilisateur fait une demande de réinitialisation. Un lien lui permettant de réinitialiser son mot de passe lui sera alors envoyé par mail. Une fois arrivé sur le lien, l’utilisateur pourra soumettre un nouveau mot de passe.

#### Vue dynamique

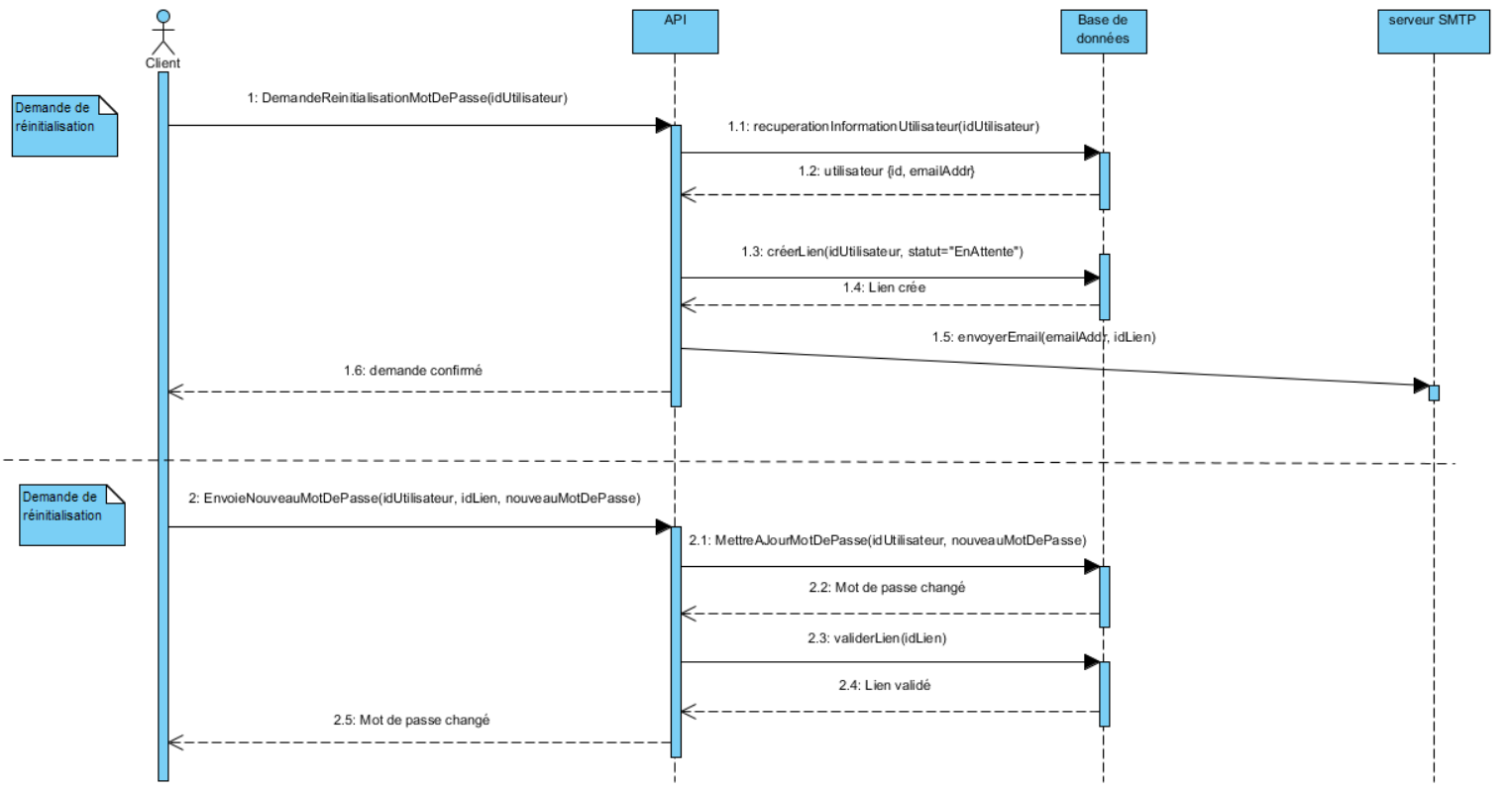


Figure . Diagramme de séquence API : réinitialisation du mot de passe

# Vue déploiement

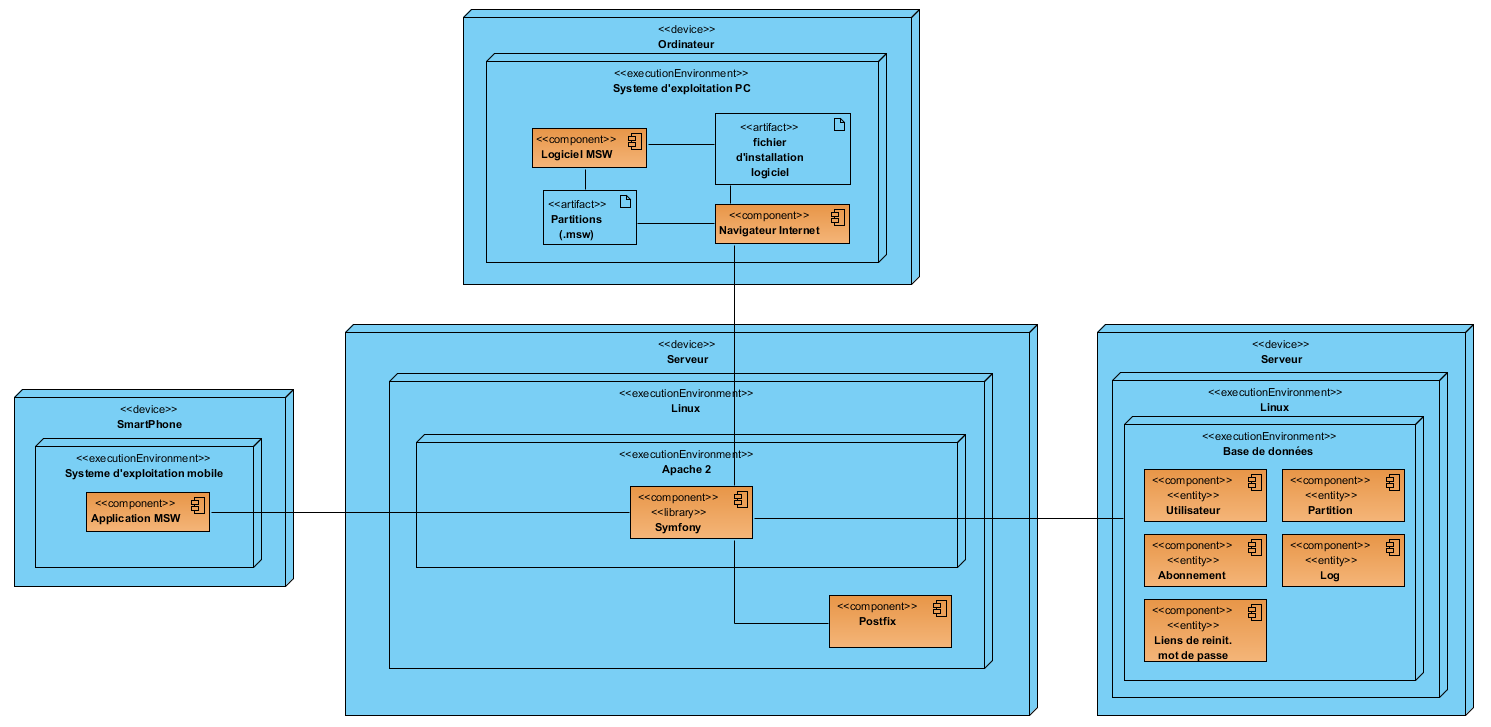


Figure . Diagramme de déploiement général

Sur le diagramme ci-dessus :

* Le « système d’exploitation PC » peut être Windows ou MAC
* Le « système d’exploitation mobile » peut être Android, iPhone ou Windows Phone
* La « base de données » peut être MySQL ou PostgreSQL.

# Implémentation

## Le logiciel

### Vue globale

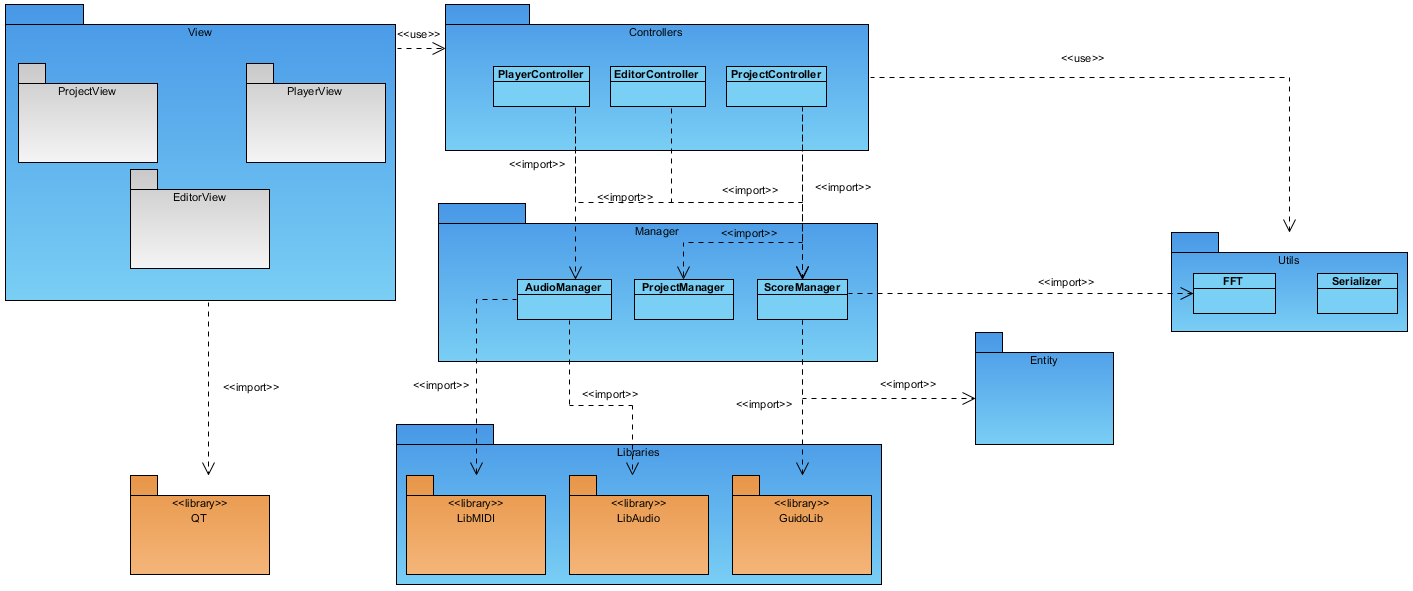


Figure . Diagramme de package global logiciel

Le logiciel suit une architecture MVC où la vue est représentée par les composants « ProjectView », « PlayerView**»**  et « EditorView ». Les contrôleurs vont permettre de contrôler l’ensemble des actions sur les vue correspondantes. Ainsi chaque vue a son contrôleur associé.

Le package « Manager » représente la couche métier de l’application.

### Couches applicatives

#### Affichage d’un projet

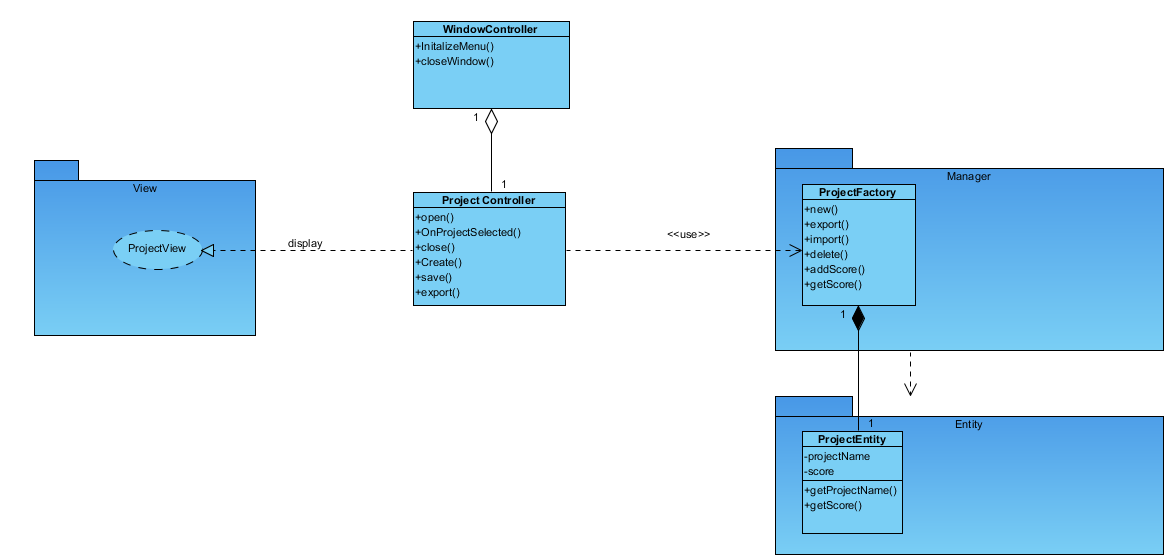


Figure . Diagramme de classe logiciel : couche présentation

La classe « WindowController » contrôle l’ensemble des actions et des évènements qui peuvent être déclenchés dans la fenêtre principale de l’application.

**«**ProjectController » contrôle juste la vue concernant la gestion de projet, cette classe est donc contenue dans le « WindowController »puisqu’elle fait partie de la fenêtre principale.

Les actions du contrôleur sont appliquées à la vue du projet « ProjectView ».

#### Lecture d’une partition

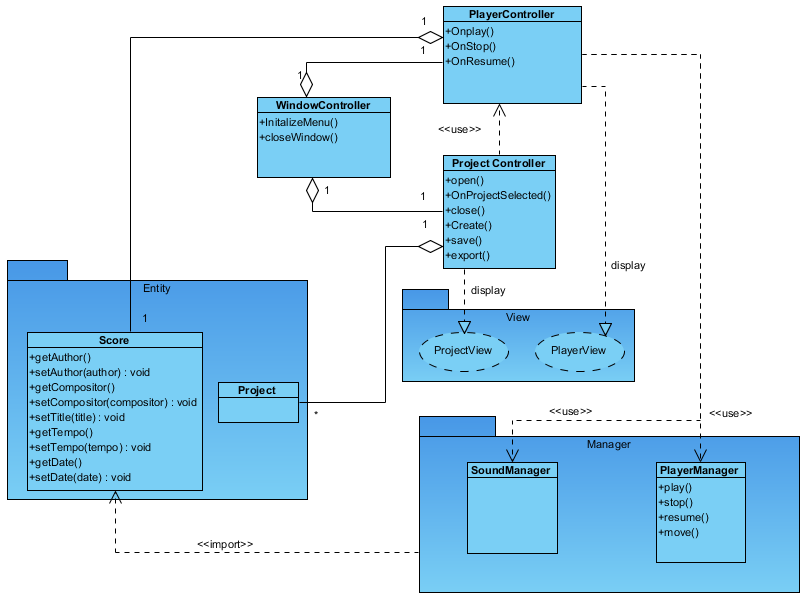


Figure . Diagramme de classe logiciel : lecture de partition

La Classe « PlayerControlleur » contrôle la vue « PlayerView » et est contenu dans la classe « WindowController ».

La logique du code implémentant les actions de la lecture de partitions est dans le package Manager. C’est la classe « PlayerManager » qui va exécuter les tâches nécessaires à la lecture de partition et utilise la classe « SoundManager » pour tout ce qui a trait à l’audio.

#### Gestion d’un projet

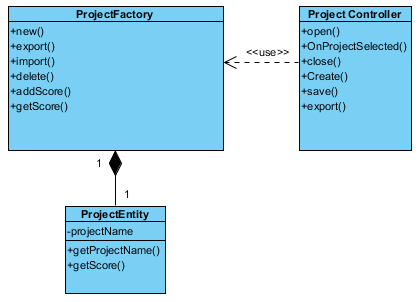


Figure . Diagramme de classe logiciel : gestion de projet

Toute la gestion du projet est gérée par la classe « ProjectFactory » qui contient toutes les méthodes nécessaires pour la création, suppression modification d’un projet.

#### Gestion de partitions

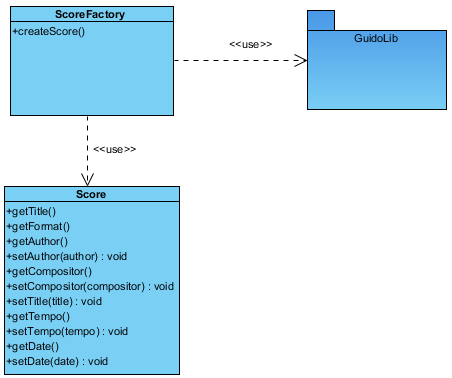


Figure . Diagramme de classe logiciel : gestion de partitions

#### Les Entités

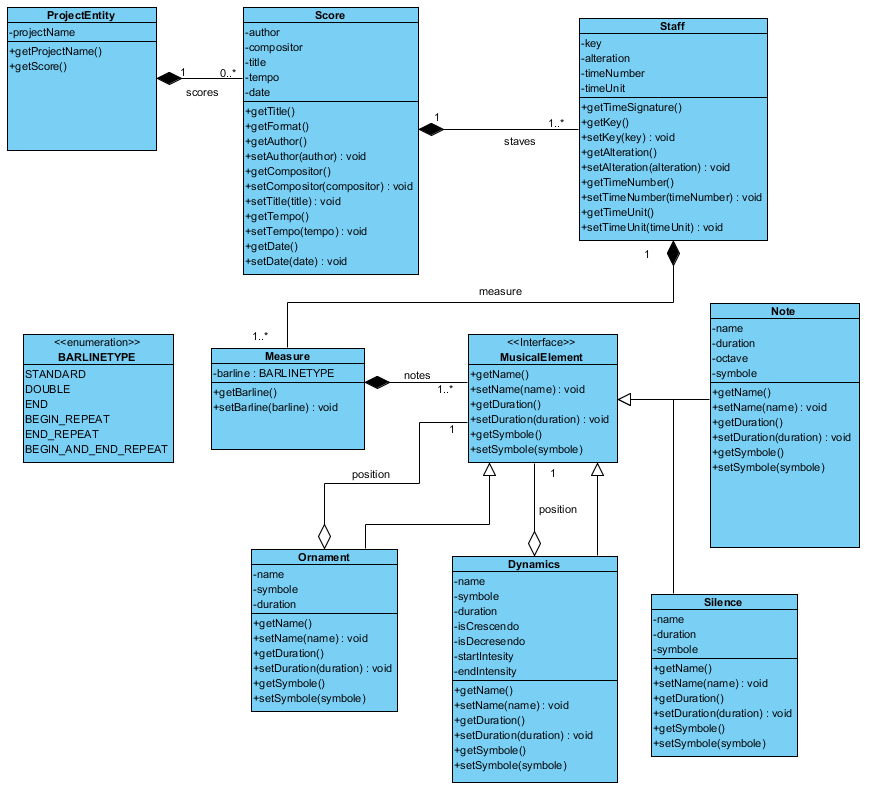


Figure . Diagramme de classe logiciel : modèle

Un projet est un ensemble de partition. Une partition est composée de plusieurs portées « Staff » qui elles même sont composées de plusieurs mesures « Measure ». Ce sont ces dernières qui contiennent les notes, silences et autres éléments musicaux.

Les entités « Note », « Silence », « Ornament », « Dynamics » représentent respectivement les notes, silence, ornements et nuances. Soit les différents éléments musicaux que l’on peut retrouver sur une partition de musique.

## Le site internet et l’API

### Vue globale

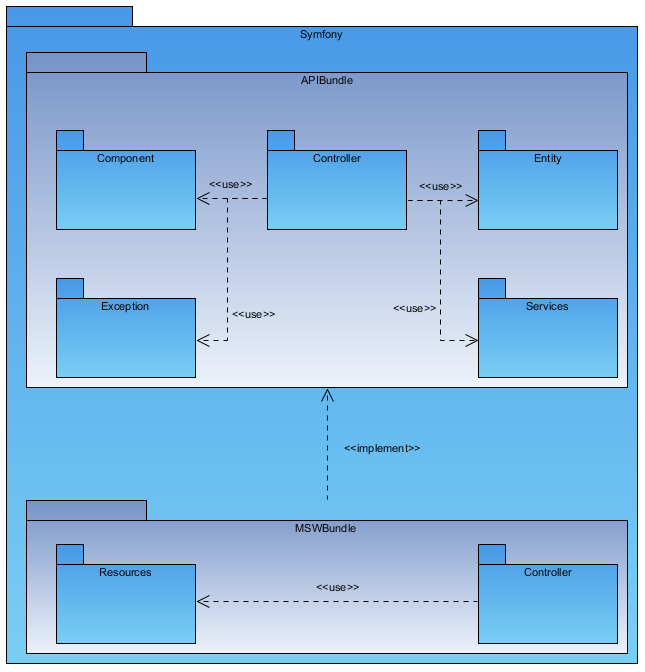


Figure . Diagramme de package global site internet

### Couches applicatives

#### API Bundle

Le bundle « APIBundle » est le bundle qui implémente l’API Rest mise à disposition des applications mobiles et de l’espace communautaire du site internet.

##### Component

Figure 35. Diagramme de classe site internet : component APIBundle

Les components sont des classes utiles pouvant être réutilisées par n’importe qu’elle autre classe présente dans « l’APIBundle ».

La classe « MSWResponse » hérite de la classe « Response » donnée par le framework Symfony. Cette classe est utilisée par tous les « Controller » (un Controller doit forcément retourner une réponse).

La classe « MSWSerializer » nous permet de sérialiser toutes les données au format JSON (format utilisé dans la définition de l’API) ou de retrouver les différents objets envoyés au format JSON par les différents clients de l’API.

La classe « MSWUtilities » regroupe les différentes méthodes ou attributs qui sont utiles au développement de l’API.

##### Controller

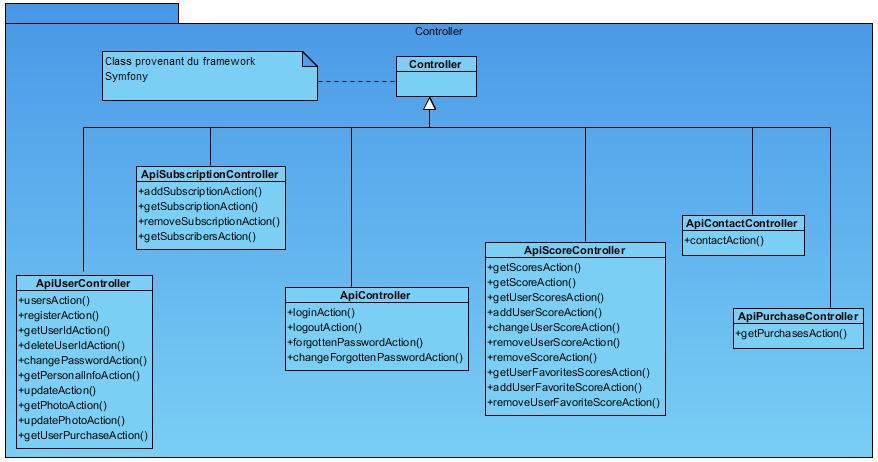


Figure . Diagramme de classe site internet : controller APIBundle

Les contrôleurs héritent tous de la classe Controller donnée par le framework Symfony. Ce sont eux qui implémentent les différentes routes de l’API Rest.

Les routes ont toutes été regroupées par domaine : les routes concernant les utilisateurs se retrouvent dans le Controller « APIUserController », celles concernant les partitions dans « APIScoreController », etc.

##### Exception

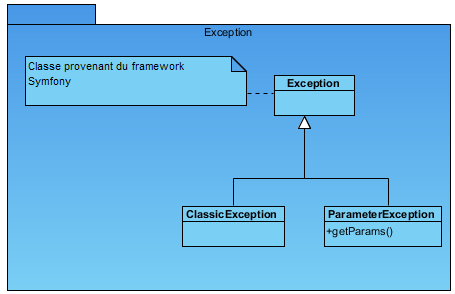


Figure . Diagramme de classe site internet : exceptions APIBundle

Nous avons définis nos propres exceptions, basées sur la classe « Exception » de Symfony. Elles sont principalement utilisées par les différents « Services » que nous avons définis pour implémenter le comportement de l’API Rest.

##### Services

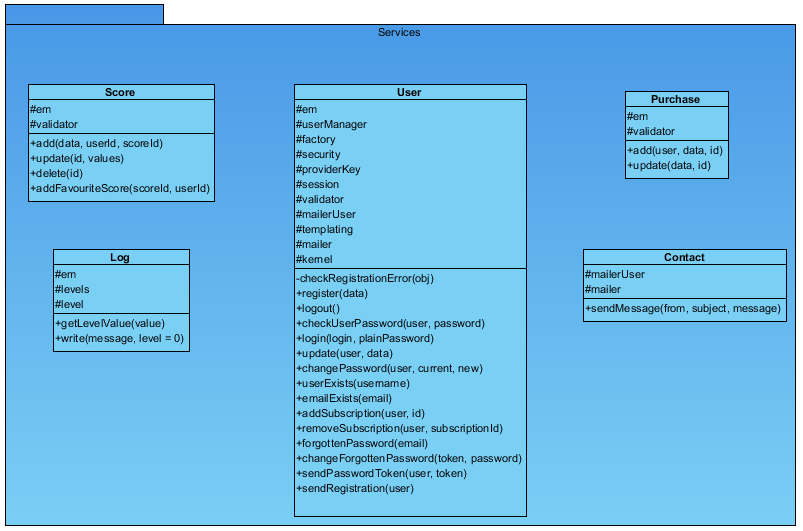


Figure . Diagramme de classe site internet : services APIBundle

Les services sont des objets permettant la réalisation d’opérations spécifiques sur un ou plusieurs objets et qui peuvent être réutilisés n’importe où dans le code. Par exemple, le service « User » permet de persister une entité « User » en base de donnée. Cette action est redondante et peux s’effectuer à différents endroits dans le code.

La différence avec les classes du package « Component » est que les classes du package « Component » n’ont pas vocation à toucher à la base de donnée ou aux différentes entités.

##### Entity

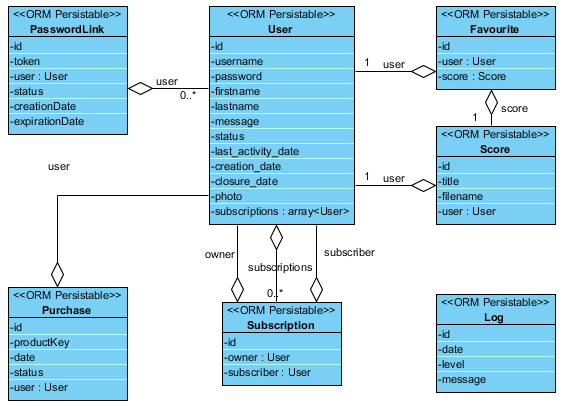


Figure . Diagramme de classe site internet : entités APIBundle

Voici les différentes entités utilisé par l’API Rest. Ces entités sont les différentes données envoyées aux clients de l’API Rest. Elles sont aussi le schéma de définition de la base de données utilisée par l’API (cf. : [8.1 Vue données](#_Vue_données)).

#### MSW Bundle

Le bundle « MSWBundle » est le bundle réalisant le site internet du projet Music Sheet Writer. Ce dernier étant client de l’API Rest précédemment définie, il ne contient aucune entité spécifique.

##### Resources

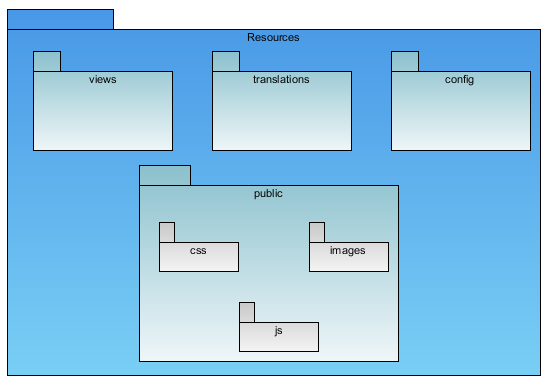


Figure . Diagramme de classe site internet : ressources MSWBundle

Le package « Resources » regroupe les différentes ressources utilisées par le bundle « MSWBundle » pour la réalisation du site internet. Il s’agit des différentes vues (template twig), des fichiers de traduction (translations), de configuration (config) ou encore tous les fichiers nécessaires à un site internet (package « public »).

##### Controller

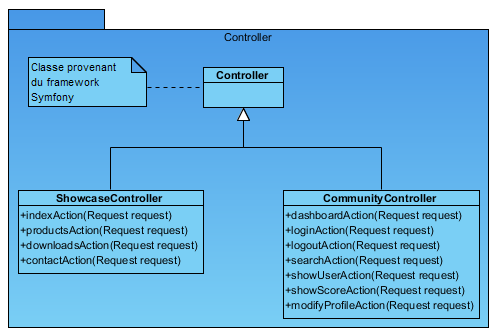


Figure . Diagramme de classe site internet : controllers MSWBundle

Tout comme les contrôleurs du bundle « APIBundle » qui implémente le comportement de l’API Rest (les routes), les contrôleurs du « MSWBundle » permettent d’associer les différentes vues aux différentes routes du site internet.

## Les applications mobiles – iPhone

### Vue globale

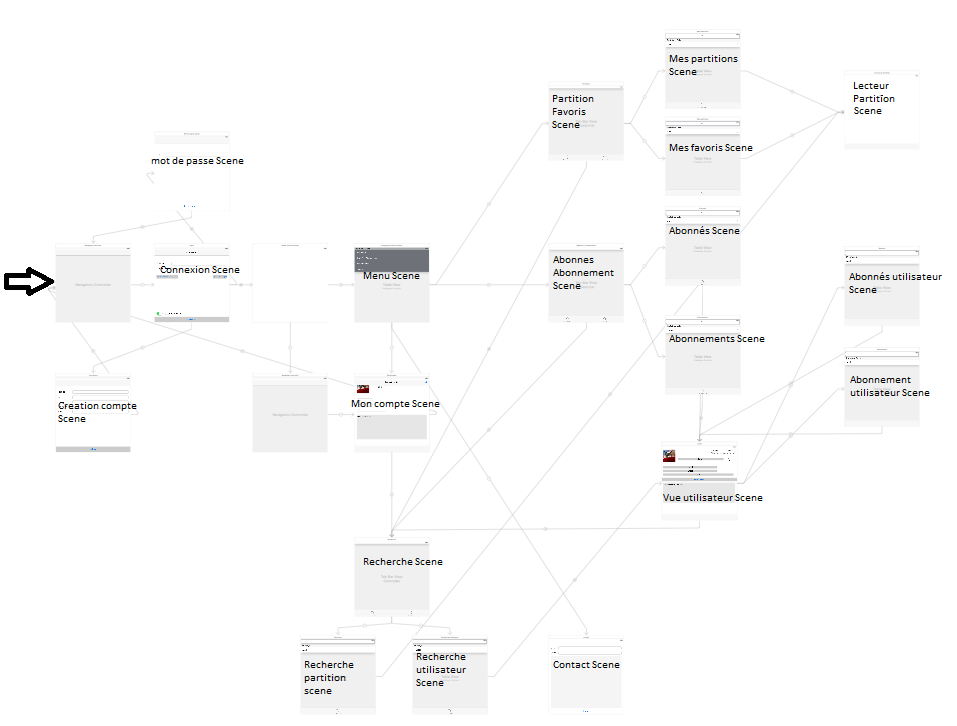


Figure . Vue globale iPhone

Le développement sur iOS s’effectue à l’aide d’un storyboard. Le storyboard permet d’agencer l’ordre des différentes vues. Par exemple, la flèche en noir décrit le point d’entrée de l’application, soit la vue de l’écran de démarrage. Le prochain écran est la page de connexion « Connexion Scene ». Depuis cette dernière nous pouvons soit allé sur la page de mot de passe oublié « mot de passe Scene », soit la page de création de compte « Creation compte Scene » ou encore la page de profile utilisateur « Mon compte Scene ».

### Couches applicatives

Chacune des pages du storyboard doit être associée à une classe spécifique. Les différents contrôles de la vue seront alors associés aux attributs et méthodes de la classe implémentant la logique de la page.

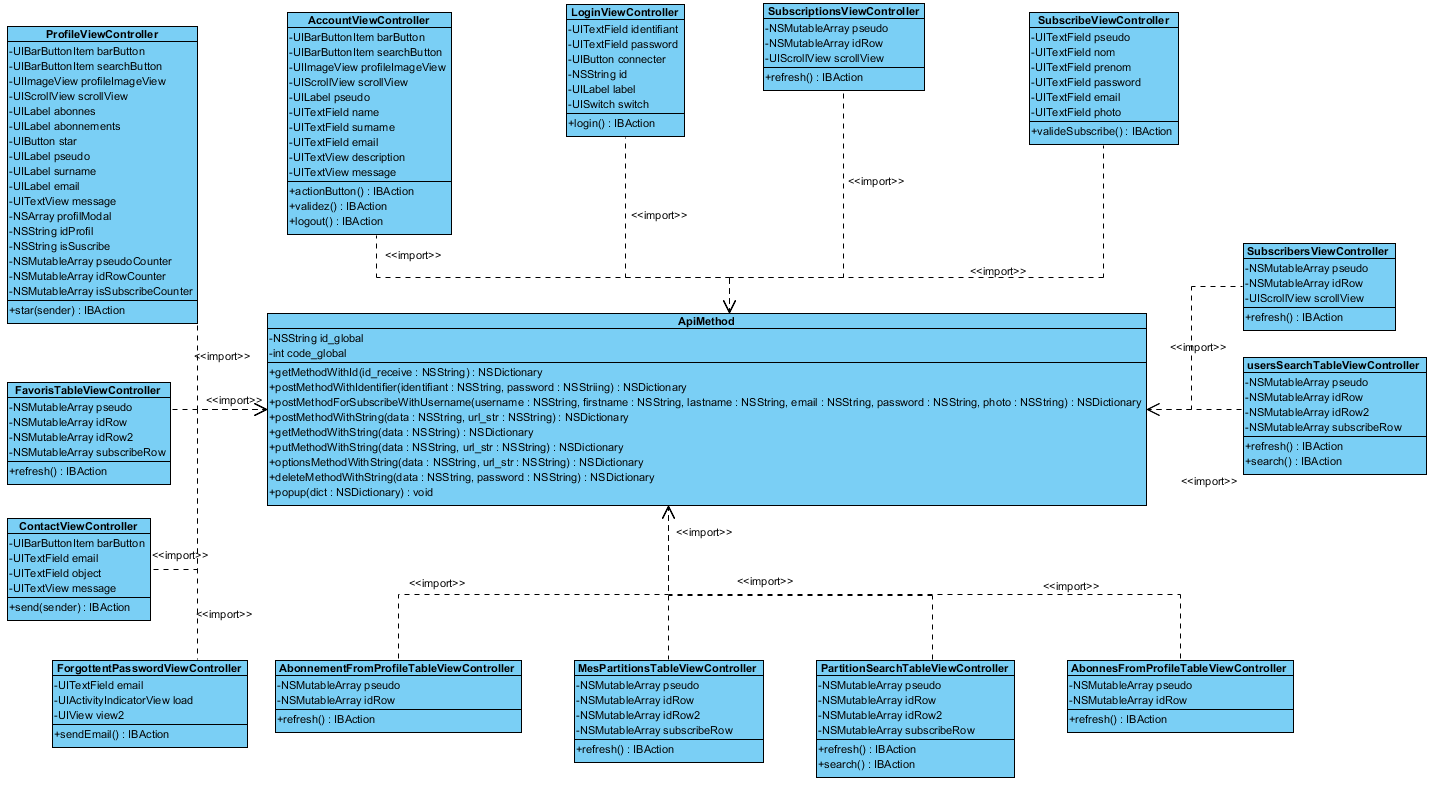


Figure . Diagramme de classe iPhone

Le diagramme ci-dessus représente les différentes classes associées aux différentes vues de l’application.

Elles importent toutes une classe nommée ApiMethode qui s’occupe de faire le lien avec l’API Rest.

## Les applications mobiles – Android

### Vue globale

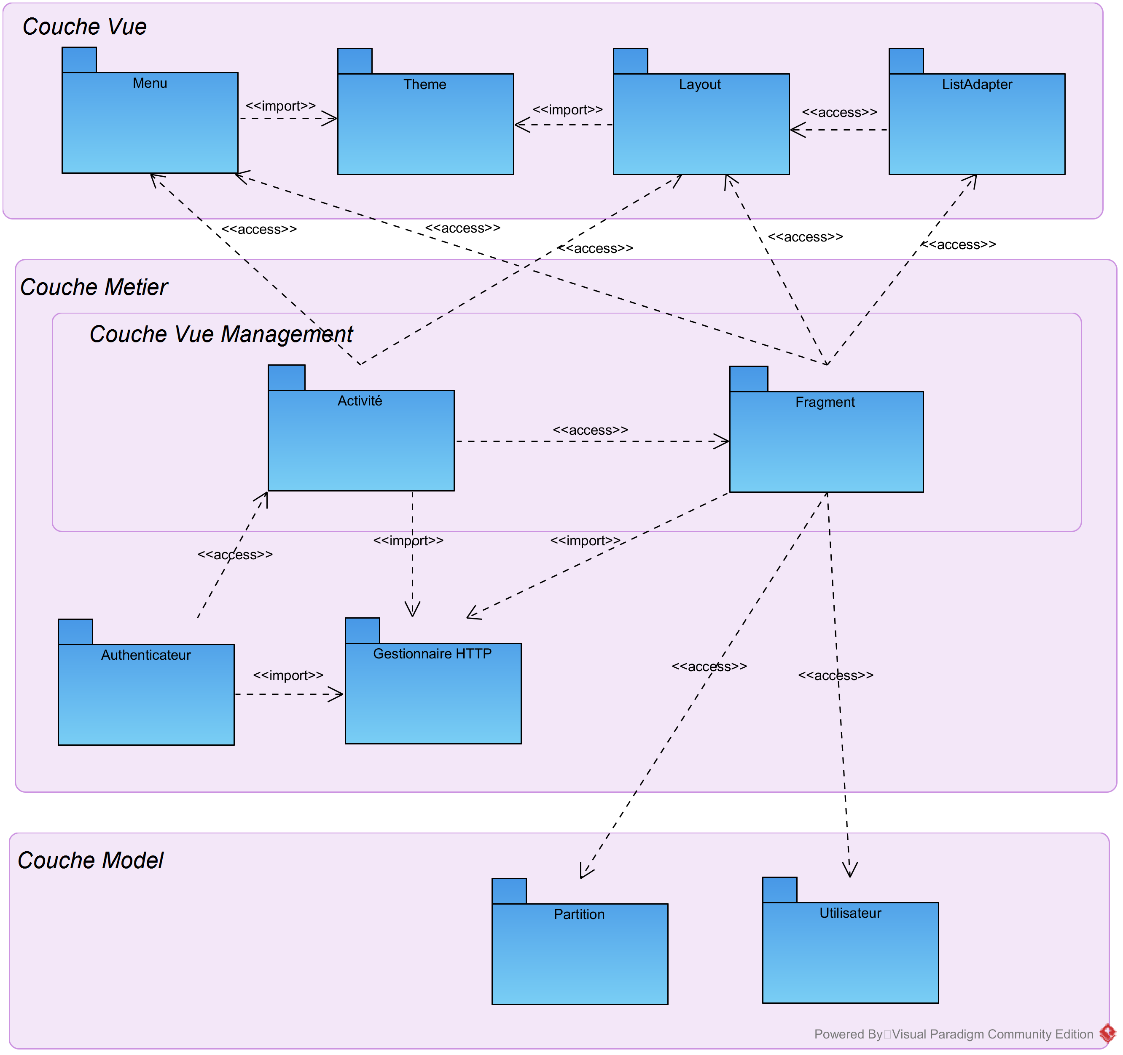


Figure . Diagramme de package global Android

L’architecture globale de l’application Android se décompose en 3 couches : la couche vue, la couche métier et la couche model. A la différence d’une architecture MVC, les vues n’interagissent jamais avec la couche model. En effet, c’est la sous-couche vue-management qui récupère les informations des modèles, les traite et les envoie à la vue. Cette architecture se rapproche donc plus du schéma MVVM.

### Couches applicatives

#### Activité

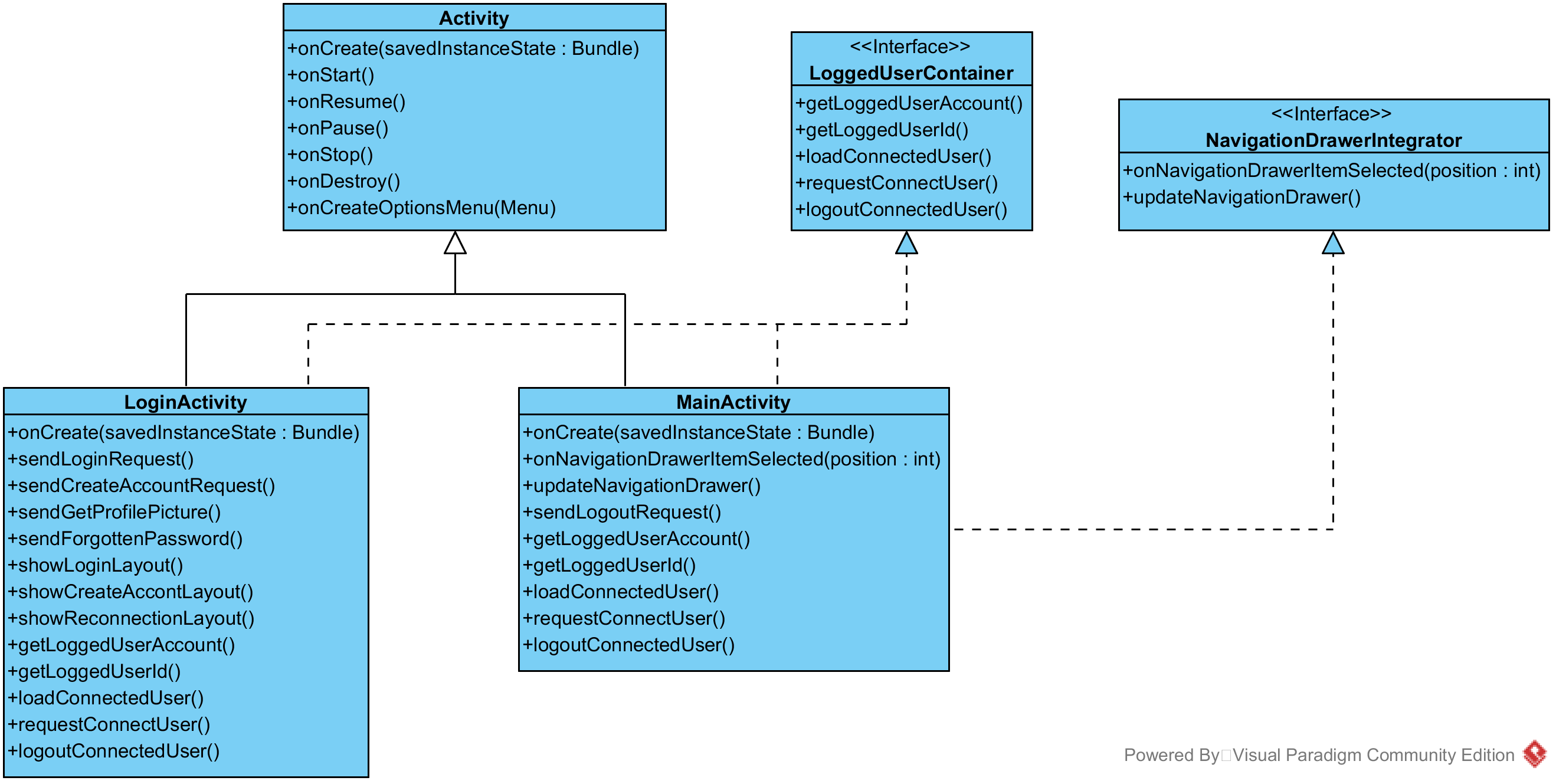


Figure . Vue applicative du package "Activité"

L’application présente deux activités :

* LoginActivity : Elle implémente les fonctionnalités d’authentifications, de création de compte ou de réinitialisation de mot de passe.
* MainActivity : Elle implémente les fonctionnalités de déconnexion et assure la gestion des fragments utilisés au sein de l’application. Elle est notamment composée d’un menu latéral afin de présenter les écrans principaux de l’application. L’interaction avec ce menu latéral se fait grâce à l’implémentation de l’interface « NavigationDrawerIntegrator ».

L’interface LoggedUserContainer est utilisée pour assurer l’implémentation de la connexion à un compte utilisateur, la déconnexion, la vérification et le renouvellement d’une telle connexion auprès du serveur distant.

#### Authenticateur

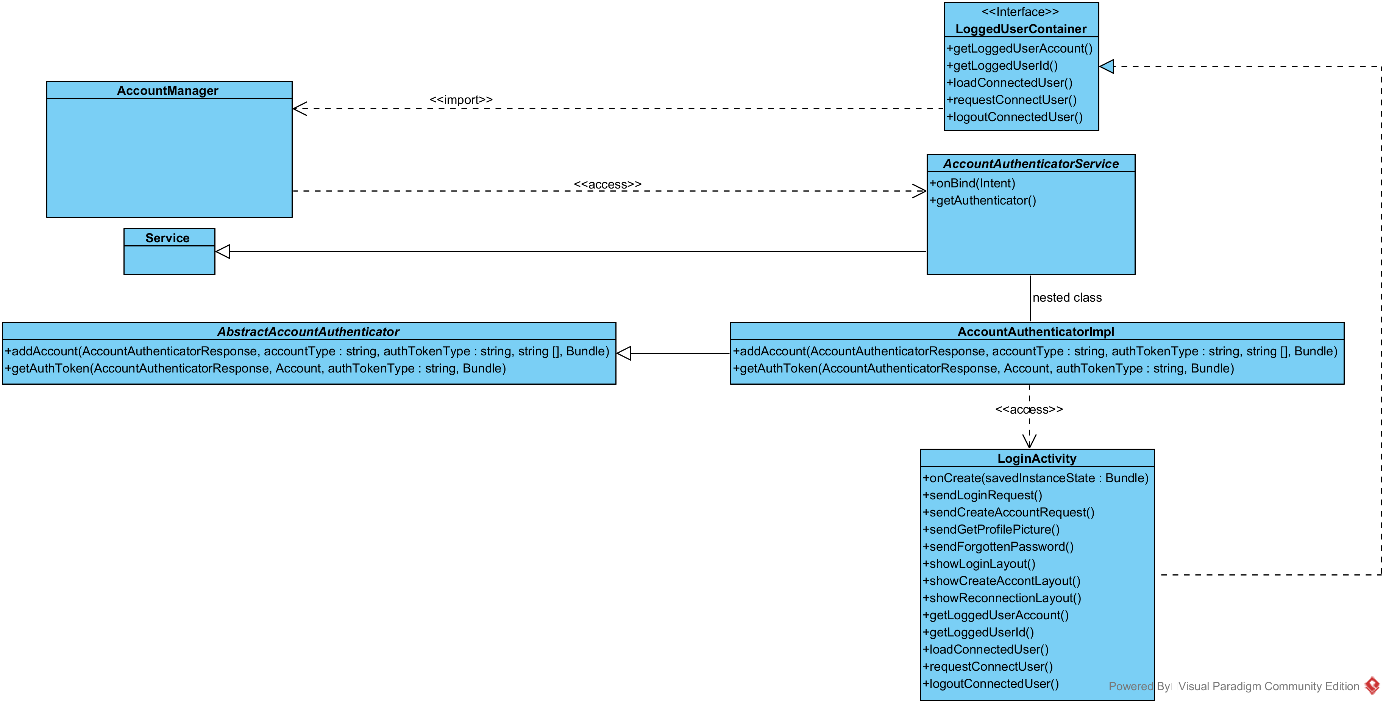


Figure . Vue applicative du package "Authenticateur"

Le système d’authentification assure la liaison entre le compte Music Sheet Writer de l’utilisateur à l’appareil Android. Les activités implémentant « LoggedUserContainer » utilisent les fonctionnalités de l’ « AccountManager » qui lui-même utilise un service fournissant les instructions à exécuter (« AccountAuthenticatorImpl » ici).

Ainsi, LoginActivity n’est jamais lancé par l’application Music Sheet Writer directement. En effet, le service d’authentification (« AccountAuthenticatorService ») fournit les informations de LoginActivity au système et c’est ce dernier qui lance l’activité.

Ce choix architectural assure une meilleure modularité en élevant le niveau de cohésion tout en gardant le même degré de dépendance entre l’application et le système Android.

#### Fragment/List Adapter

##### Global

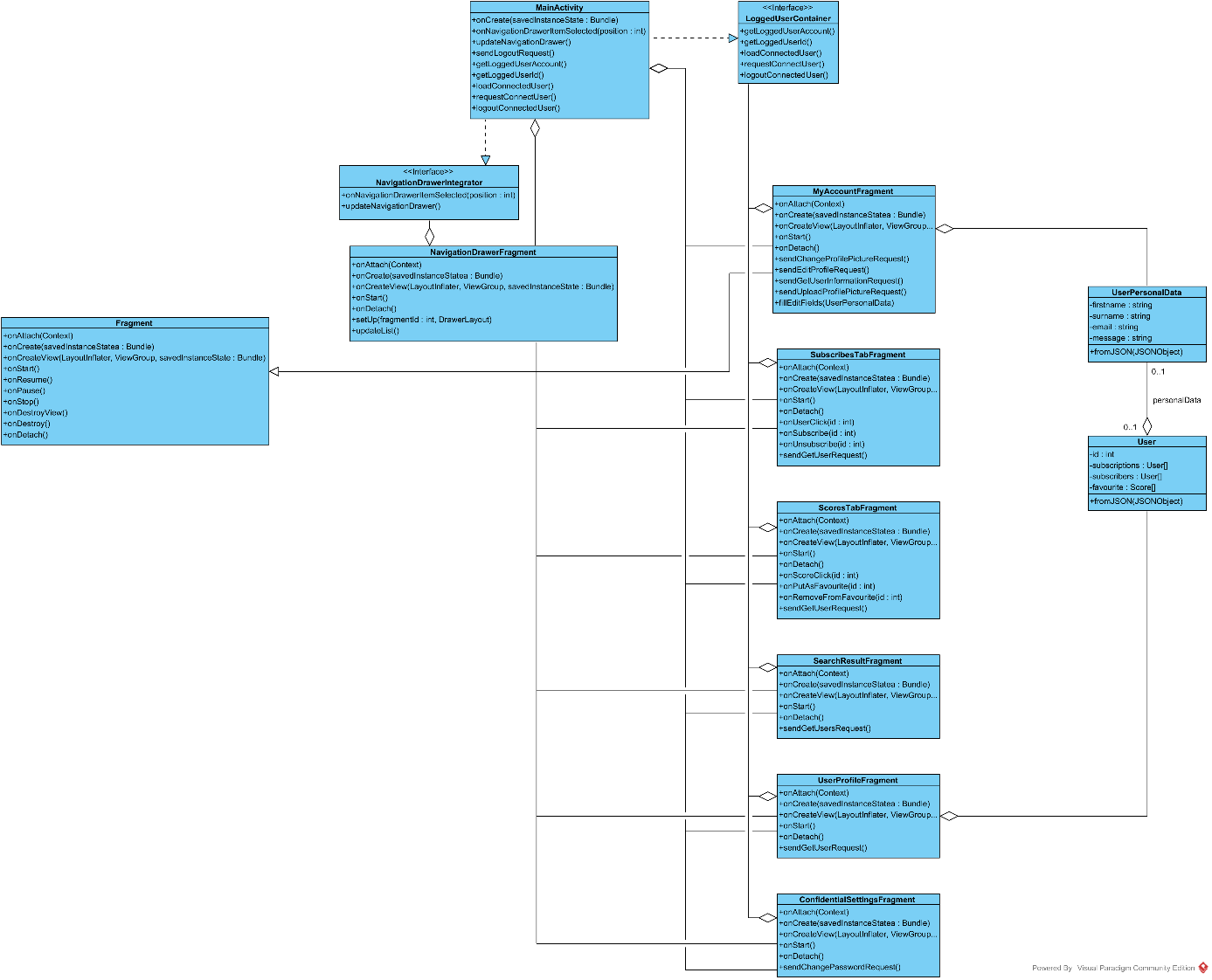


Figure . Vue applicative du package "Fragment"

L’architecture de l’application Android privilégie l’utilisation de fragment aux activités. En effet, la création de fragment nécessite moins de ressource système à la création et destruction. De plus, l’agencement visuel et dynamique de fragments est plus flexible qu’avec des activités.

Les fragments présents sur ce diagramme, c’est-à-dire les classes héritant de la classe « Fragment » sont considérés comment étant de niveau 1 et sont instanciés à partir de la MainActivity. Ils sont totalement indépendants les uns des autres puisque qu’aucun fragment de niveau 1 ne peut interagir avec un autre. En cas d’échange d’informations entre deux fragments, elles seront relayées par la MainActivity.

Chaque fragment possède une instance de classe implémentant « LoggedUserContainer » afin d’avoir accès au compte de l’utilisateur et aux fonctionnalités associés (connexion / déconnexion / vérification / renouvellement). Cette interface assure une dépendance nulle entre les fragments et MainActivity puisqu’ils pourraient être instanciés par n’importe quelle activité implémentant cette interface.

##### Fragments avec List Adapter

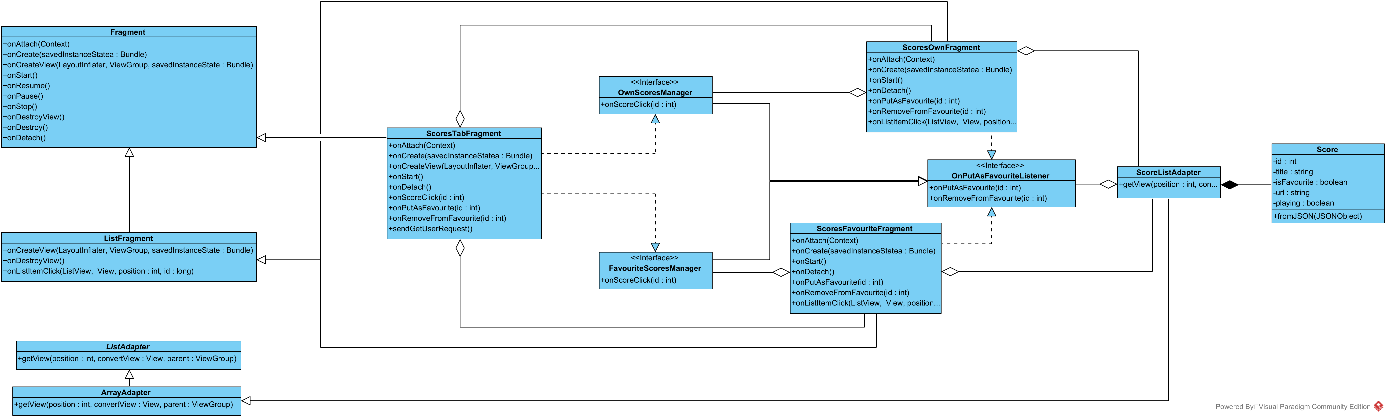


Figure . Vue applicative du package "Fragment" et "List Adapter" (Scores)

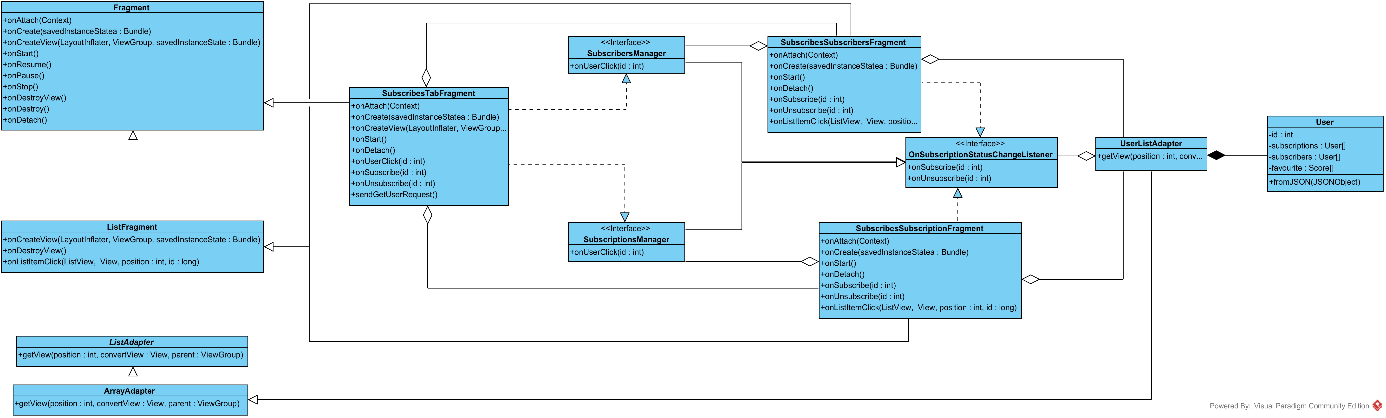


Figure . Vue applicative du package "Fragment" et "List Adapter" (Subscribers)

Certains fragments de niveau 1 nécessitent d’instancier d’autre fragment, étant considérés de niveau 2. Cette configuration concerne la gestion des abonnements/abonnées et a gestion des partitions. Les deux cas respectent le même schéma architectural :

* Le fragment de niveau 1 possède deux onglets chacun étant une instance d’un fragment.
* Ces deux fragments contiennent chacun une liste et héritent donc de « ListFragment »
* Une interface par onglet assure l’envoie d’information depuis les fragments enfants (niveau 2) au fragment parent (niveau 1).
* Les deux onglets doivent afficher le même type d’information et les listes contenues dans les fragments enfant utilisent le même type de ListAdapter.
* Une interface assure l’envoie d’informations depuis l’adapter au ListFragment.

#### Gestionnaire HTTP

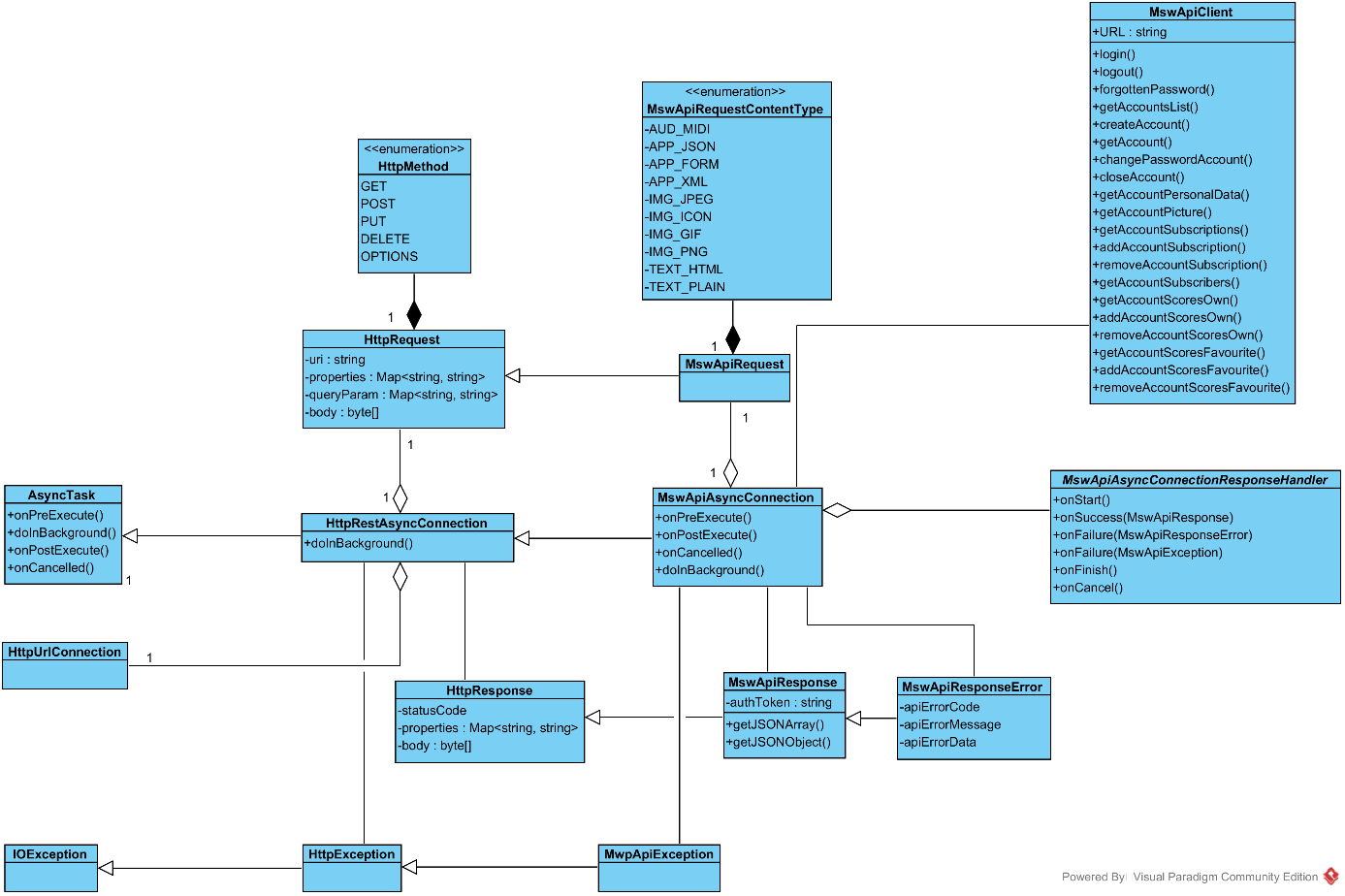


Figure . Vue applicative du package "Gestionnaire HTTP"

Les fragments et les activités interagissent avec le gestionnaire HTTP en appelant les fonctions membres de « MswApiClient ».

L’envoie de requête HTTP au sein de l’application utilise deux classes fournis par l’API Android :

* AsyncTask est utilisé pour exécuter une action de manière asynchrone sur un autre *thread* que le *thread* principale. En effet, le système Android n’autorise pas les opérations réseaux sur le *thread* principal.
* HttpUrlConnection implémente l’envoie d’une requête HTTP.

Ces deux classes sont agrégées au sein de « HttpRestAsyncConnection » qui encapsule également la notion de REST. Cette classe reste totalement générique et pourra être utilisé dans d’autre projet.

« MswApiAsyncConnection » surcharge les fonctionnalités de « HttpRestAsyncConnection » et permet l’ajout de Callback au cours de l’envoie de la requête, la réception et le traitement de la réponse HTTP. Les instructions à exécuter sont envoyés via une instance de « MswApiAsyncConnectionResponseHandler »

#### Model

Les modèles sont les classes représentant des entités en contenant leurs informations associées. Les entités possèdent une opération *static* « fromJSON » permettant d’instancier une entité à partir d’un retour JSON provenant de l’API.

##### Partition

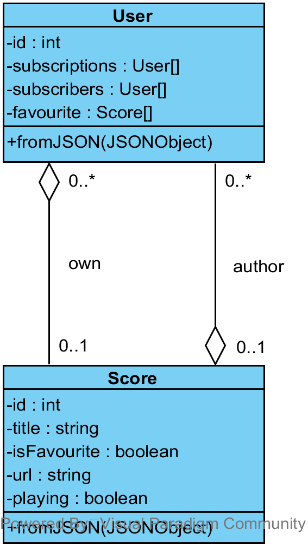


Figure . Vue applicative du package "Partition"

##### Utilisateur

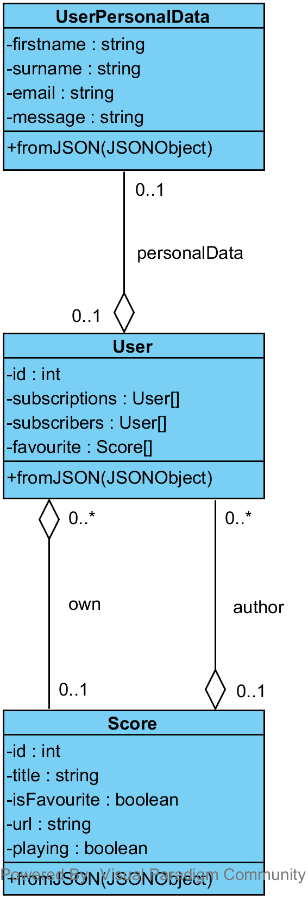


Figure . Vue applicative du package "Utilisateur"

## Les applications mobiles – Windows Phone

### Vue globale

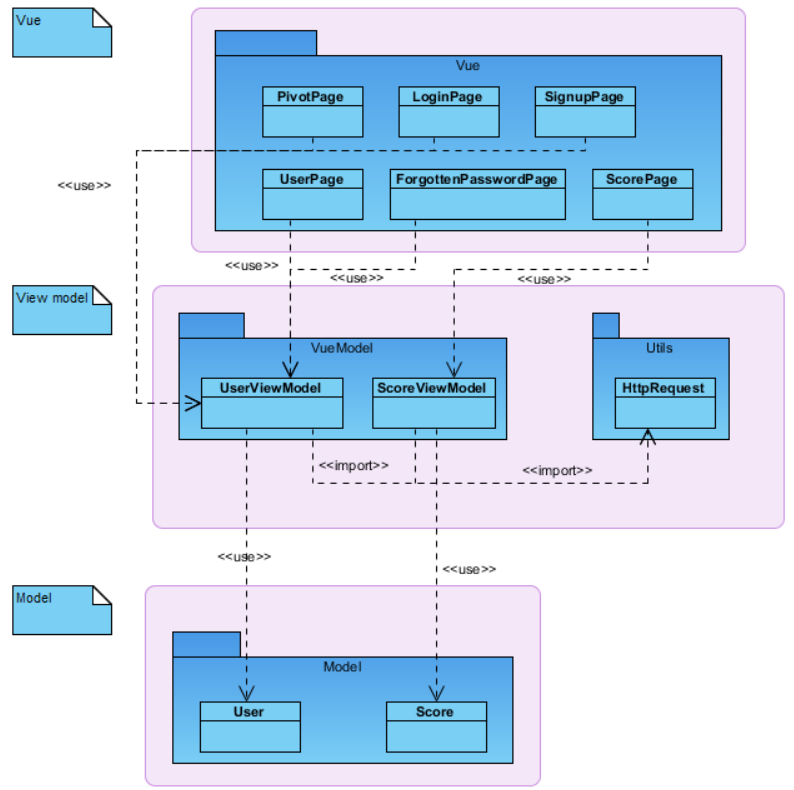


Figure . Diagramme de package global Windows Phone

L’architecture globale de l’application Windows Phone se décompose en trois couches: la couche vue, la couche vue-modèle et la couche modèle. L’architecture employée est celle du schéma MVVM.

### Couches applicatives

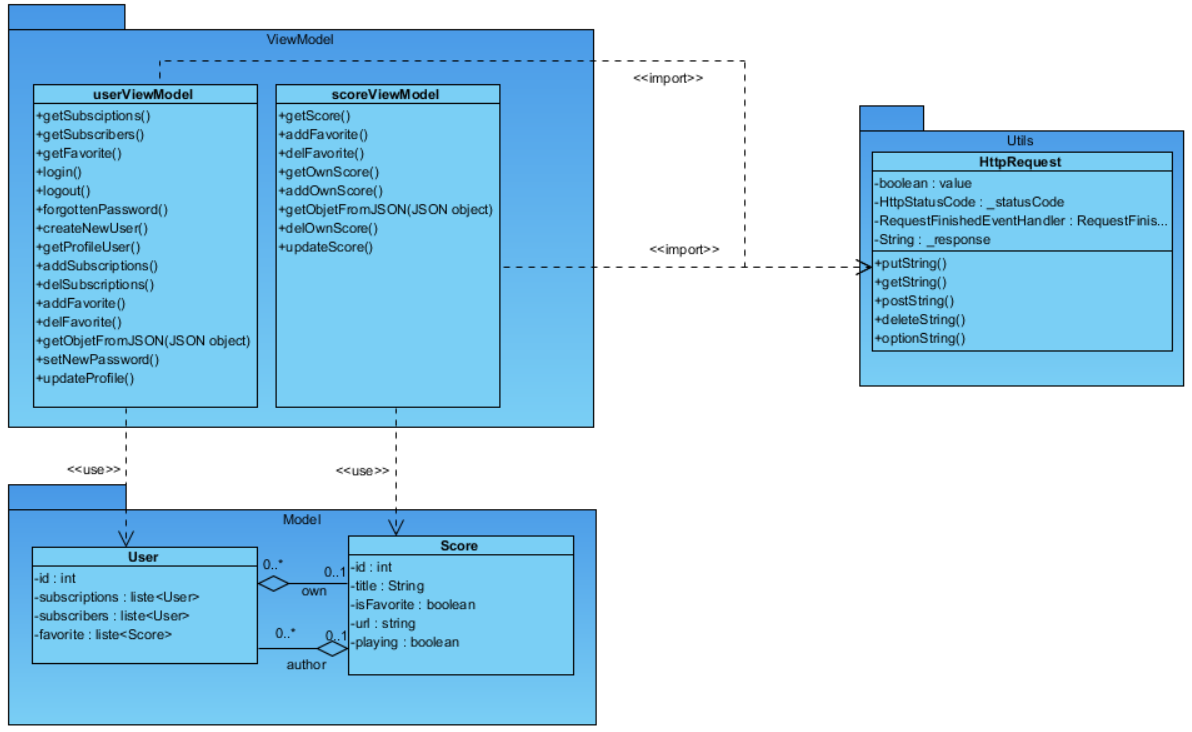


Figure . Vue applicative Windows Phone

L’application se compose de deux classes « vue-modèle »:

* « userViewModel » : Elle implémente toutes les interactions liées à l’utilisateur.
* « scoreViewModel » : Elle implémente les interactions liées aux partitions

Ces deux classes font un appel récurant à la classe « HttpRequest » afin d’établir une connexion avec l’API et de récupérer les données afin de les afficher.

# Vue données

Figure 55. Modèle de la base de données

Le modèle de base de données est défini depuis les entités du bundle Symfony APIBundle précédemment évoqué dans le chapitre [7.2 Le site internet et l'API](#_Le_site_internet).

# Annexes

CTRL clic permet de visualiser ce(s) document(s)

[](https://labeip.epitech.eu/svn/2017/musicsheetwriter/rendu/2017_API_musicsheetwriter.pdf)

