

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

Music Sheet Writer

Music Sheet Writer

Bilan Architecture

Objectifs du document

Résumé

Glossaire

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 |
| 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 |

Sommaire

[1. Introduction 1](#_Toc438422828)

[1.1. Rappel de l’EIP 1](#_Toc438422829)

[1.2. Contexte et périmètre du projet 1](#_Toc438422830)

[2. Représentation de l’architecture globale 2](#_Toc438422831)

[3. Vue globale du projet 3](#_Toc438422832)

[3.1. Cas d’utilisation principaux 3](#_Toc438422833)

[3.2. Cas d’utilisation détaillés 3](#_Toc438422834)

[4. Vue logique de l’application 4](#_Toc438422835)

[4.1. Le logiciel 4](#_Toc438422836)

[4.2. Le site internet 4](#_Toc438422837)

[4.3. Les applications mobiles 4](#_Toc438422838)

[5. Vue processus 5](#_Toc438422839)

[5.1. Le logiciel 5](#_Toc438422840)

[5.2. L’API 5](#_Toc438422841)

[5.3. Les applications mobiles 5](#_Toc438422842)

[6. Vue déploiement 6](#_Toc438422843)

[6.1. Le logiciel 6](#_Toc438422844)

[6.2. Le site internet & les applications mobiles 6](#_Toc438422845)

[7. Implémentation 7](#_Toc438422846)

[7.1. Le logiciel 7](#_Toc438422847)

[7.2. L’API 7](#_Toc438422848)

[7.3. Les applications mobiles – iPhone 7](#_Toc438422849)

[7.4. Les applications mobiles – Android 7](#_Toc438422850)

[7.5. Les applications mobiles – Windows Phone 7](#_Toc438422851)

[8. Vue données 8](#_Toc438422852)

[8.1. Modélisation 8](#_Toc438422853)

[8.2. Description 8](#_Toc438422854)

[9. Taille et performance 9](#_Toc438422855)

[10. Annexes 10](#_Toc438422856)

[10.1. Le document du CDC 10](#_Toc438422857)

Liste des Tableaux

**No table of figures entries found.**

Liste des Figures

[Figure 1: Vue globale de l'implémentation de l'application mobile - Android 7](#_Toc439137323)

[Figure 2 : Vue applicative du package "Activité" 8](#_Toc439137324)

[Figure 3 : Vue applicative du package "Authenticateur" 8](#_Toc439137325)

[Figure 4 : Vue applicative du package "Fragment" 9](#_Toc439137326)

[Figure 5 : Vue applicative du package "Fragment" et "List Adapter" (Scores) 9](#_Toc439137327)

[Figure 6 : Vue applicative du package "Fragment" et "List Adapter" (Subscribers) 10](#_Toc439137328)

[Figure 7 : Vue applicative du package "Gestionnaire HTTP" 10](#_Toc439137329)

[Figure 8 : Vue applicative du package "Utilisateur" 11](#_Toc439137330)

[Figure 9 : Vue applicative du package "Partition" 11](#_Toc439137331)

# Introduction

## Rappel de l’EIP

## Contexte et périmètre du projet

# Représentation de l’architecture globale

# Vue globale du projet

## Cas d’utilisation principaux

### Le logiciel

### Le site internet

### Les applications mobiles

## Cas d’utilisation détaillés

### Le logiciel

#### Editer une partition

#### Lire une partition

#### Gérer un projet

### Le site internet & application mobile

#### Accéder à la communauté

#### Gérer son compte utilisateur

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

# 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

## Le site internet

## Les applications mobiles

# Vue processus

## Le logiciel

## L’API

### Création de compte

#### Description

#### Vue dynamique

### Connexion

#### Description

#### Vue dynamique

### Changement d’adresse email

#### Description

#### Vue dynamique

### Réinitialisation du mot de passe

## Les applications mobiles

# Vue déploiement

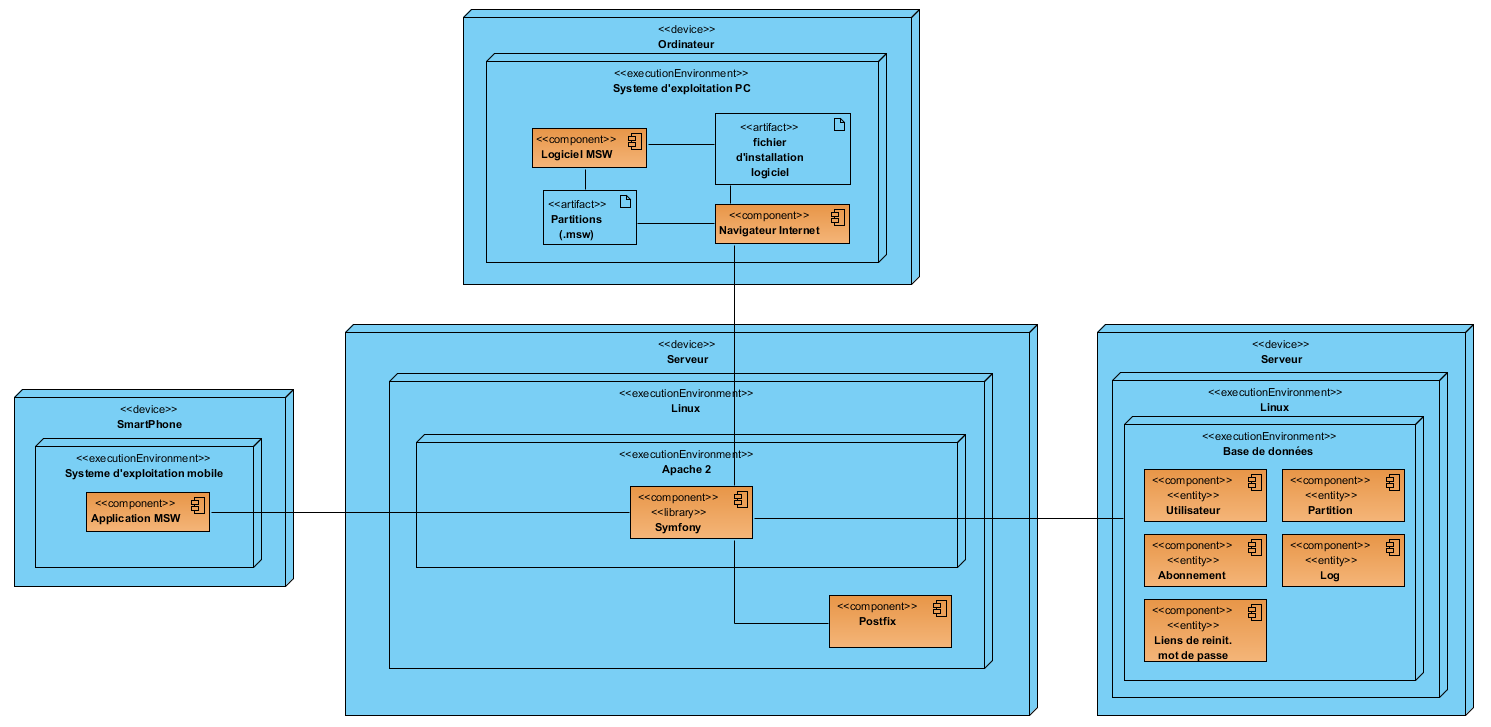


Figure : Déploiement des composants de Music Sheet Writer

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

### Couches applicatives

## L’API

### Vue globale

### Couches applicatives

## Les applications mobiles – iPhone

### Vue globale

### Couches applicatives

## Les applications mobiles – Android

### Vue globale

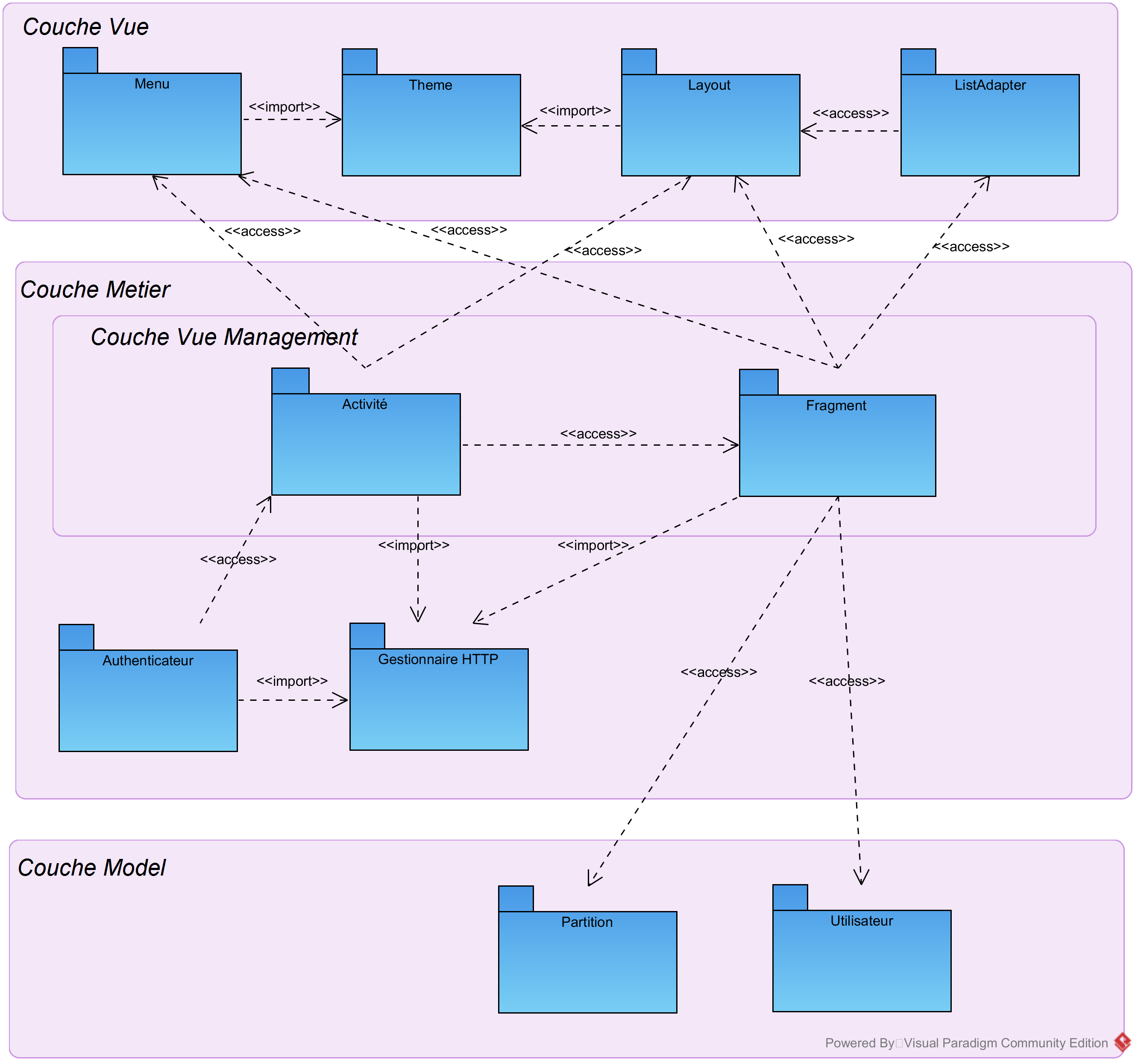


Figure 2: Vue globale de l'implémentation de l'application mobile – Android

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

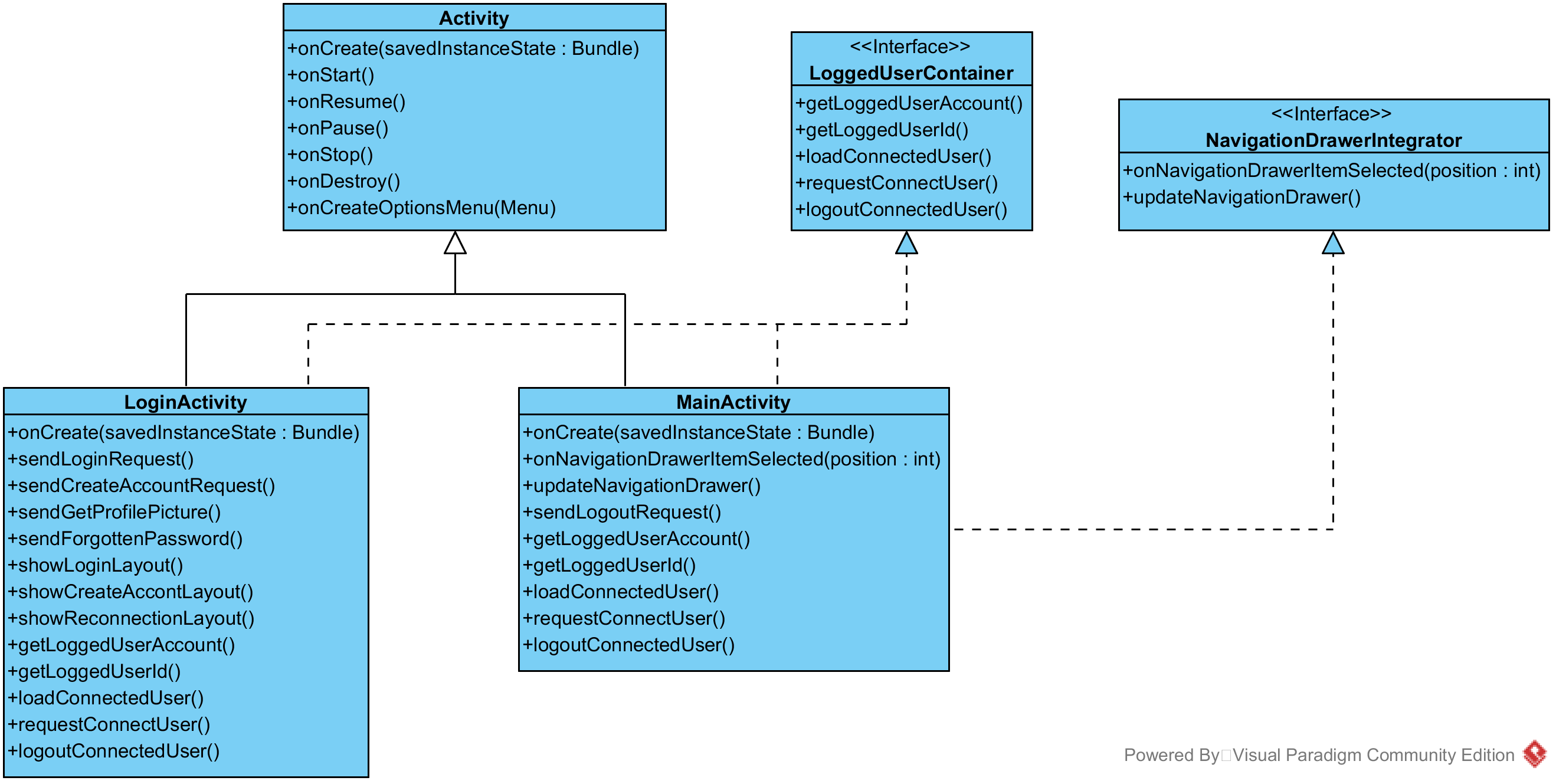


Figure 3 : Vue applicative du package "Activité"

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

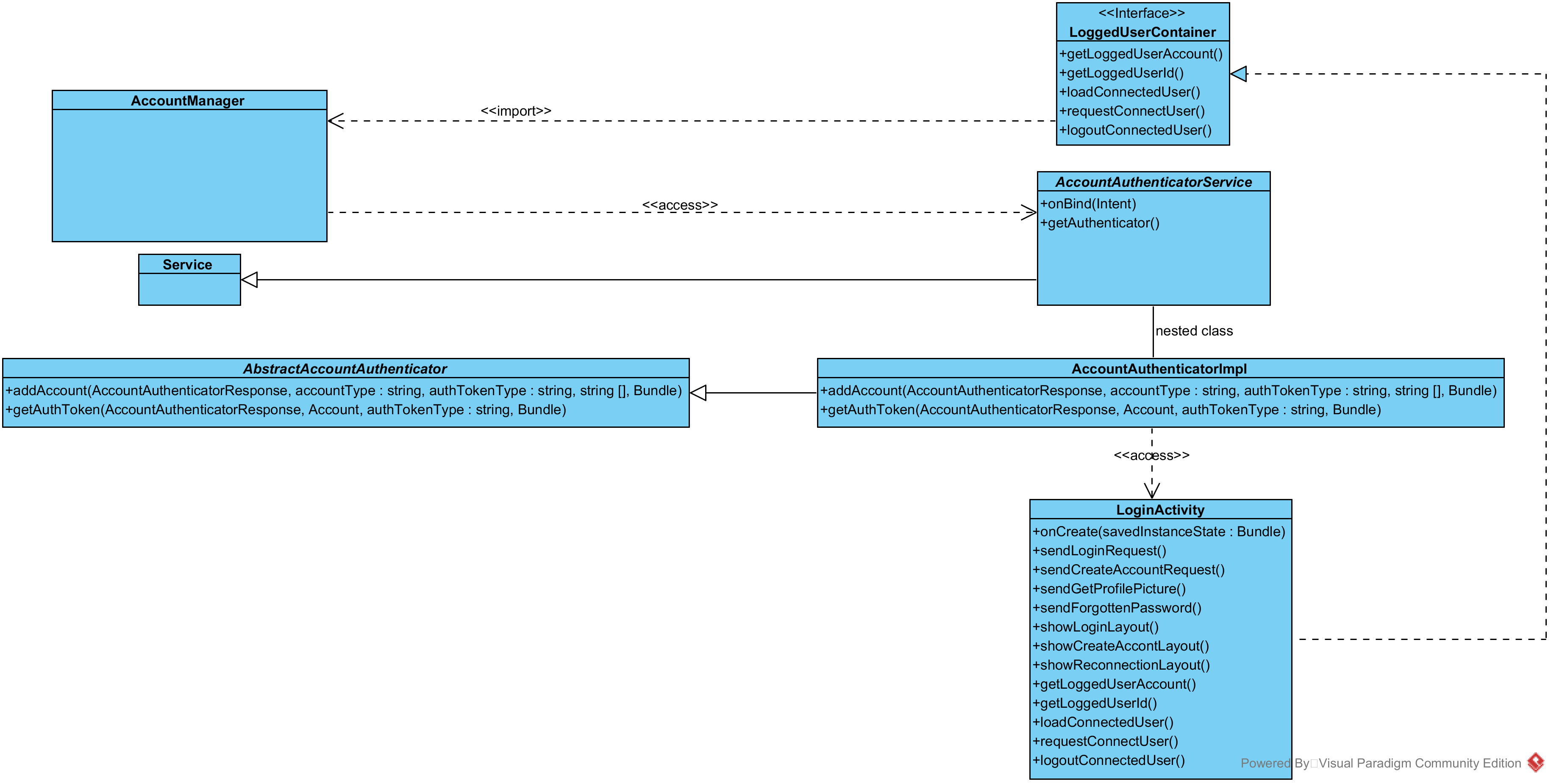


Figure 4 : Vue applicative du package "Authenticateur"

Le système s’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

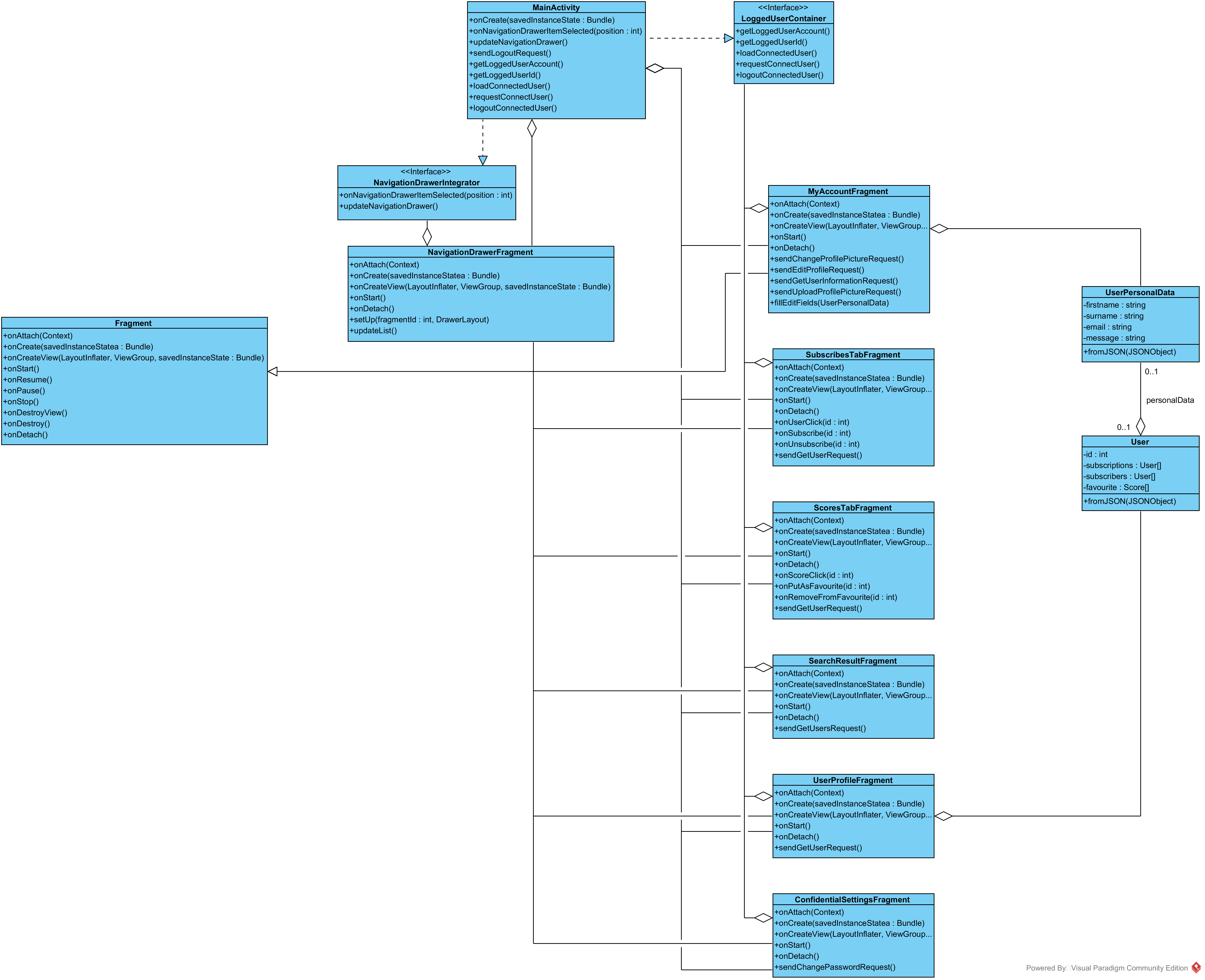


Figure 5 : Vue applicative du package "Fragment"

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

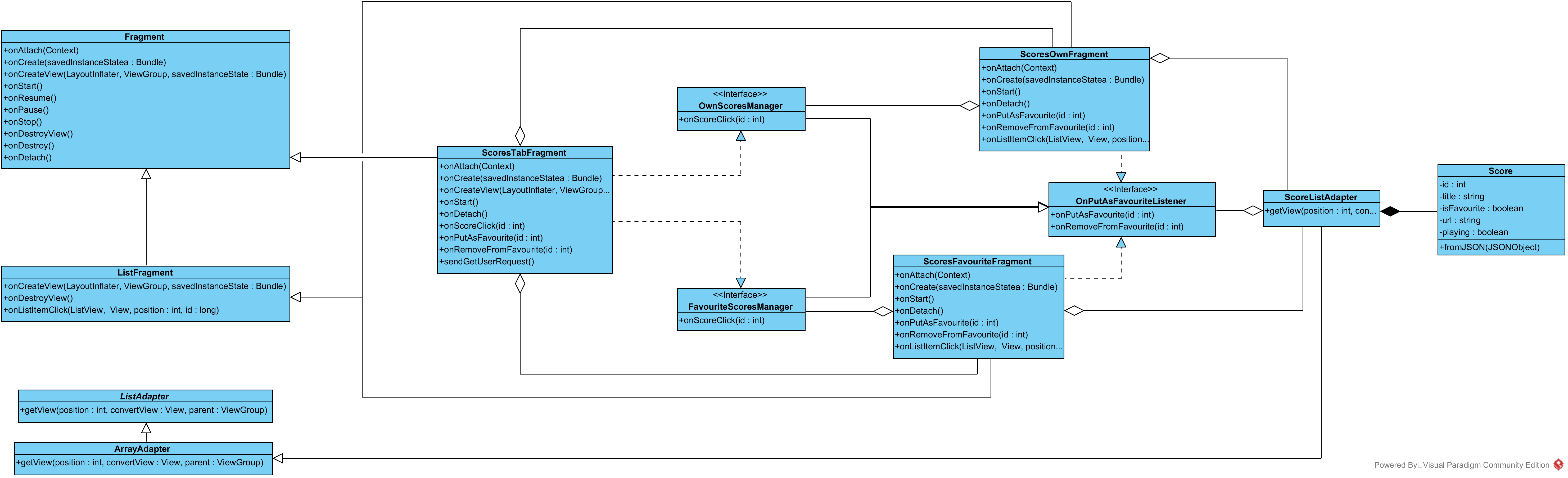


Figure 6 : Vue applicative du package "Fragment" et "List Adapter" (Scores)

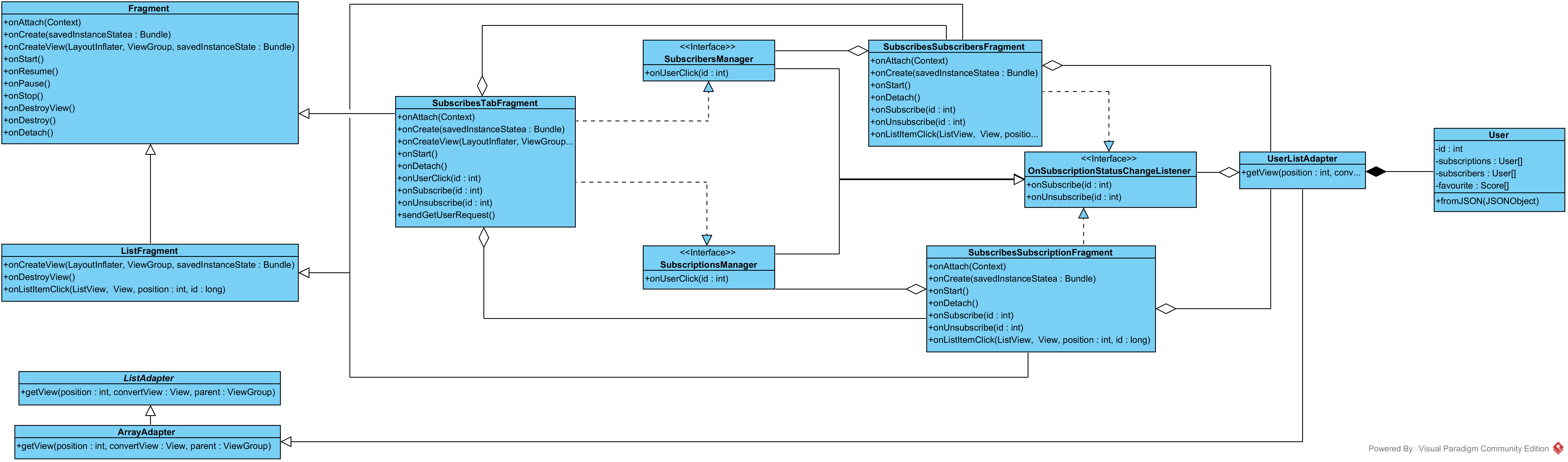


Figure 7 : Vue applicative du package "Fragment" et "List Adapter" (Subscribers)

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

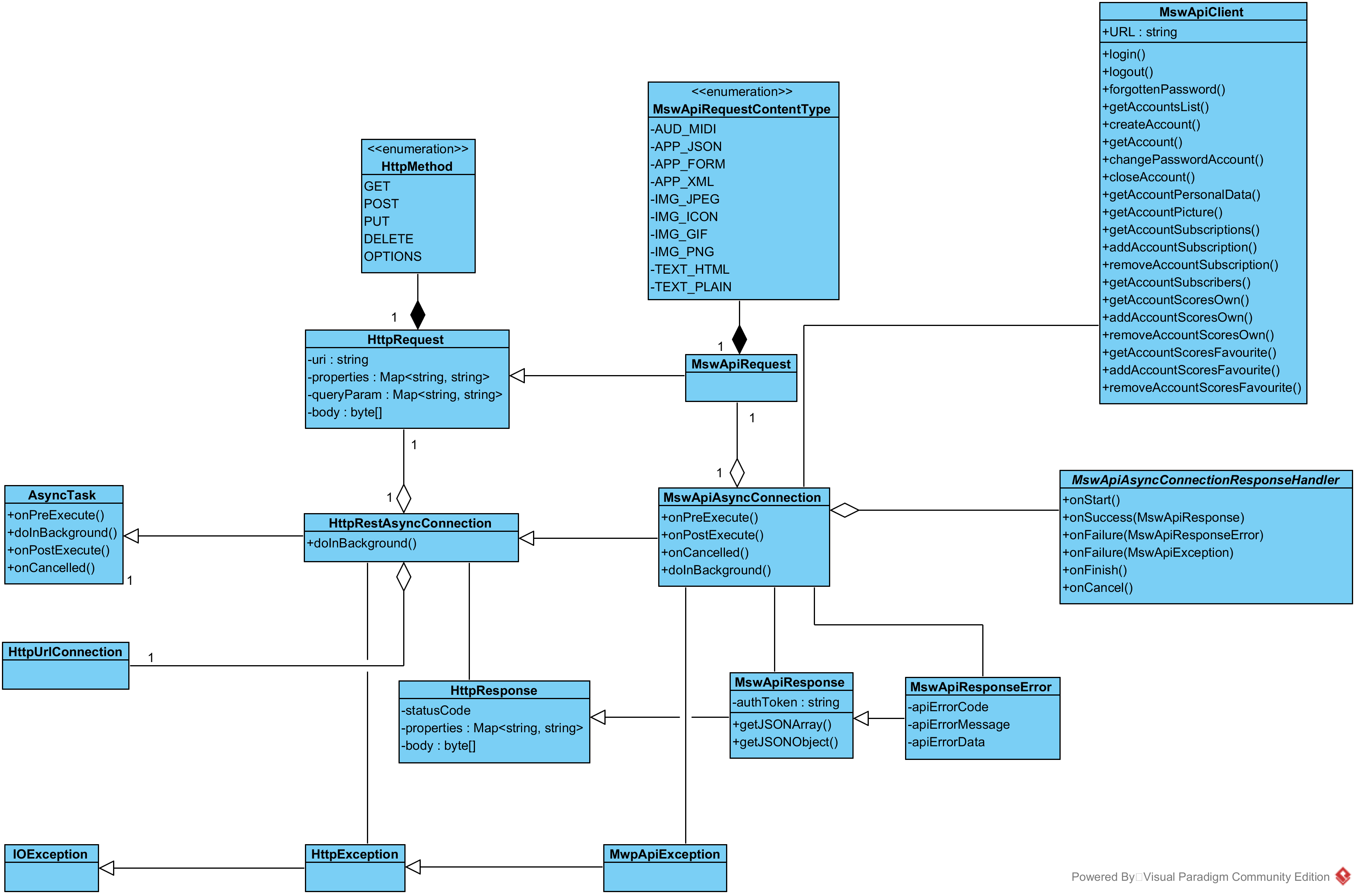


Figure 8 : Vue applicative du package "Gestionnaire HTTP"

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

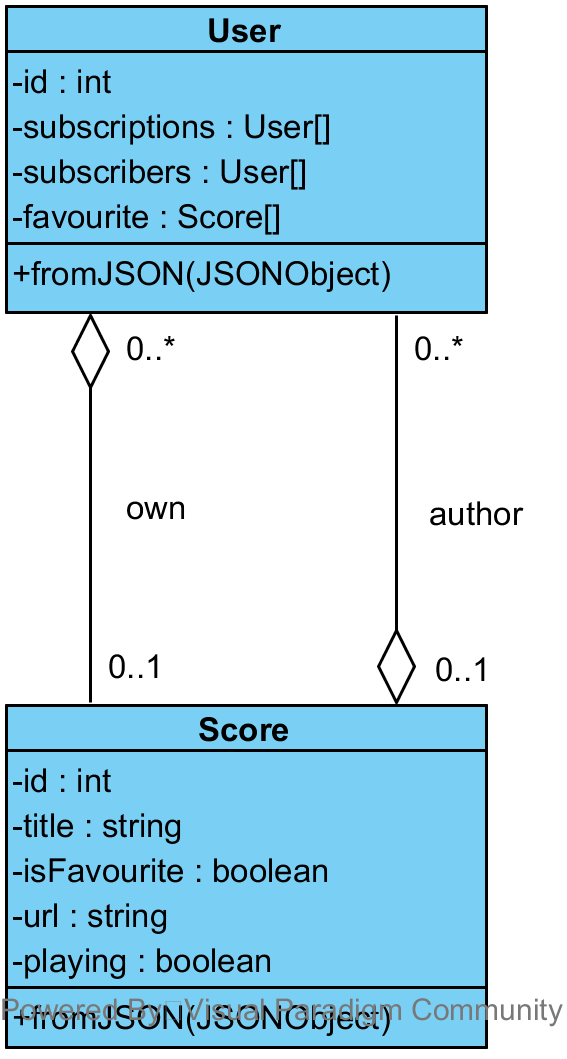


Figure 9 : Vue applicative du package "Utilisateur"

##### Utilisateur

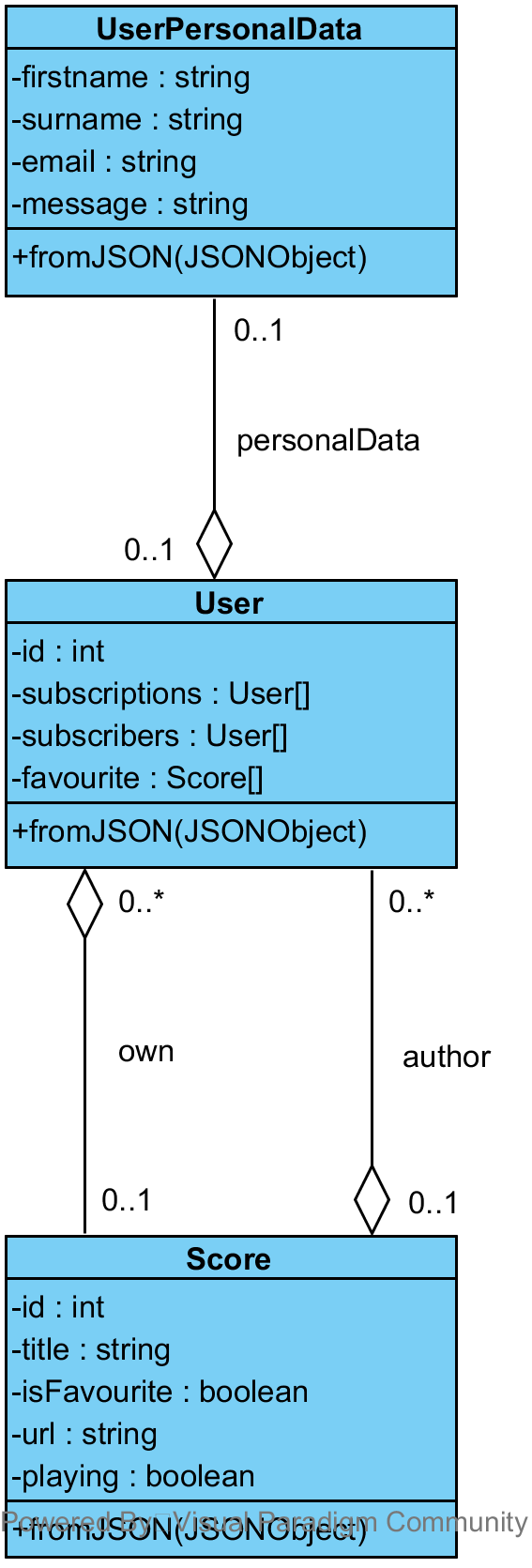


Figure 10 : Vue applicative du package "Partition"

## Les applications mobiles – Windows Phone

### Vue globale

### Couches applicatives

# Vue données

## Modélisation

## Description

# Taille et performance

# Annexes

CTRL clic permet de visualiser ces documents

## Le document du CDC

