Pisano Vincent Vasseur William Morel Emilien

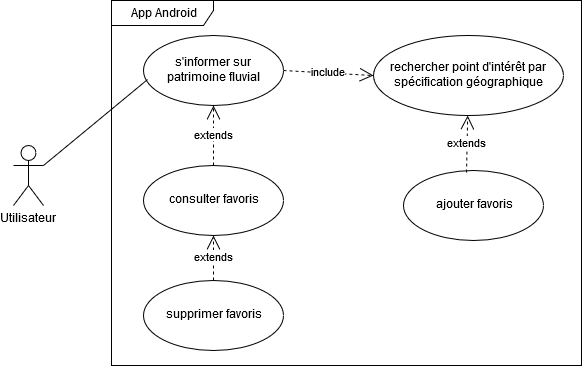
**Rapport :**

**Introduction :**

Le projet Au fil de l’Ô consiste à créer une application Android communiquant avec une base de données via une API de type RESTful. Le but du projet est de mettre en valeur le patrimoine touristique du département en général et de son patrimoine fluvial remarquable des bords de Seine et de Marne en particulier. L’API est réalisée en PHP et s’appuie sur une base de données alimentée par un fichier JSON et retourne ses informations elle aussi en JSON. L’application Android est réalisée en Java.

Le groupe est composé de 3 membres dont William Vasseur, Emilien Morel et Vincent Pisano. Emilien s’est occupé de réaliser l’API ainsi que la connexion a la base de données. De plus Vincent et Emilien ont réalisée l’extraction des données JSON ainsi que l’insertion dans la base de données des données. D’un autre coté William a débuté l’application en réalisant l’écran permettant la sélection sur critère d’un point d’intérêt. Suite a cela Vincent et William ont travailler ensemble pour faire communiquer l’API et l’application Android.

**Cas d’utilisation UML :**



**OpenData :**

Une donnée ouverte (en anglais open data) est une information publique brute, qui a vocation à être librement accessible et réutilisable. La philosophie pratique des données ouvertes préconise une libre disponibilité pour tous et chacun, sans restriction de copyright, brevets ou d'autres mécanismes de contrôle.

**Droits OpenData :**

Les Open Data, ou données ouvertes, sont des données auxquelles l'accès est totalement public et libre de droit, au même titre que l'exploitation et la réutilisation.

Dans notre projet les données OpenData sont proposées sur le site de la région Ile-de-France. Nous aurons ainsi les données de tous les ponts, passerelles et viaducs du patrimoine fluvial de notre région.

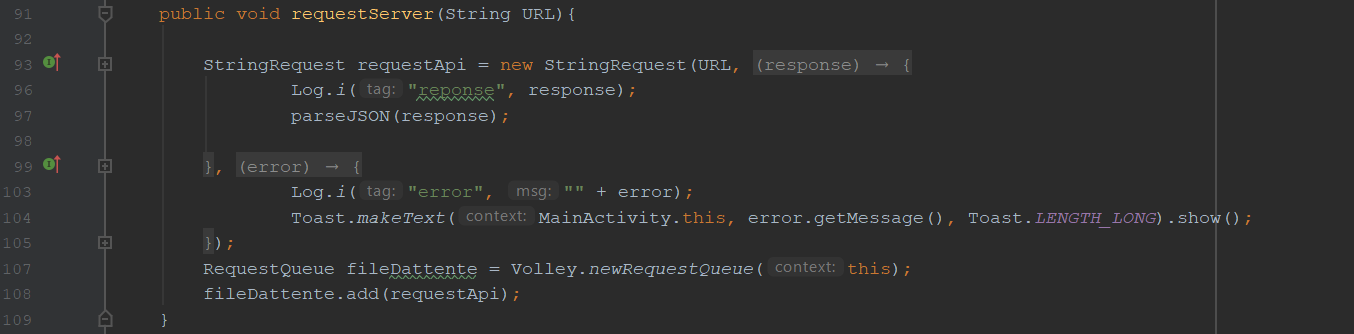
**Différences XML JSON :**

|  |  |
| --- | --- |
| **JSON** | **XML** |
| JSON est orienté données. | XML est généralement orienté document. |
| JSON prend en charge les tableaux. | XML ne supporte pas les tableaux. |
| JSON est moins sécurisé que XML. | XML est plus sécurisé grâce au typage des données. |
| JSON prend en charge uniquement les types de données text, number et boolean. | XML prend en charge de nombreux types de données tels que texte, nombre, images, etc. De plus, XML offre des options pour transférer le format ou la structure des données avec les données réelles. |

**Choix du JSON :**

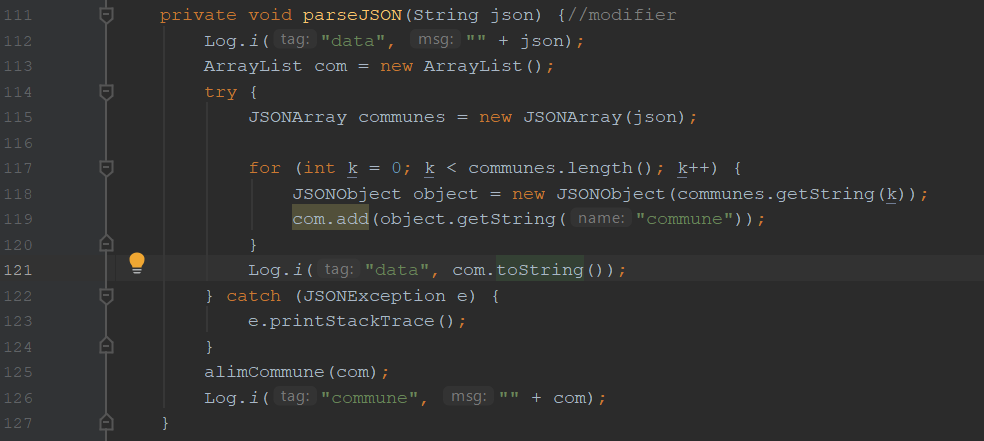
Il est simple à lire et à écrire, est orientée pour les données ce qui est le cas dans notre projet et aussi par le fait qu’ils sont plus lisibles que les fichiers XML. C’est pour cela que l’on choisit le format JSON. Il sera aussi plus facile de récupérer et de traiter les données d’un JSON que d’un XML, notamment au travers de notre API.

**Présentation du code :**



Cette capture d’écran nous présente la fonction « requestServer » qui permet de réaliser une requête vers l’API. Cette méthode prend en paramètre un string qui est l’URL.

A la ligne 93 on créer une variable « requestApi » de type « StringRequest » qui permet de créer une nouvelle requête GET à partir d’un string qui est l’URL. Cette méthode retourne soit une « error » soit une « response ». Dans la « response », on redirige vers la méthode « parseJSON » avec comme paramètre la réponse retourner par l’API. A la ligne 107 on créer une file d’attente ordonnée qui permet de stocker dans l’ordre les requêtes qu’on lui ajoutera par la suite. Ensuite a la ligne 108 on ajoute à la file d’attente créer à la ligne précédente la « requestApi » créer précédemment à l’aide de la méthode « add ».



Cette image nous montre la méthode « parseJSON » qui sert à interpréter la réponse en JSON retourné par l’API en une « ArrayList ». Cette méthode prend comme paramètre un string « json » qui sont les données retourner par l’API.

Au début de la méthode a la ligne 113, on créer l’ArrayList qui contiendra les communes présentent dans la variable « json » c’est pour cela que l’on a appelé cette ArrayList « com ».

Dans un try nous commençons par créer une « JSONArray » du nom de « communes » avec comme argument le « json » qui est donné en paramètre de la méthode « parseJson ». Ensuite on fait une boucle qui a pour but d’alimenter l’ArrayList « com » avec les communes présentent dans la réponse de l’API qui est dans notre cas le « json ».

Pour finir à la ligne 125, on appelle la méthode « alimCommune » qui sert à alimenter le spinner des communes avec les communes retournées par l’API en fonction du département choisi.

**Conclusion :**

Le cahier des charges qui nous a été demandé est respecté. Notre application possède toutes les fonctionnalités demandées tels que la recherche de point d’intérêt en fonction du département, de la commune ainsi que du type de point d’intérêt. De plus le système de favoris demandé est opérationnel et utilise les techniques souhaitées.

Ce projet fut novateur et inspirant car il utilise des langages nouveaux pour nous tel que Android. De plus la communication entre une application Android et une API RESTful fut un aspect très intéressant du projet.