ANDROID - TP3

Source et pdf du cours et de ce TP :
https://bitbucket.org/olevitt/technologies-mobiles/src

L'objectif de ce TP est de réaliser une application utilisant les données de la star (vélostar) pour informer l'utilisateur des disponibilités de vélos en station.
Ce TP fera appel aux notions suivantes :

- Requêtes HTTP (webservice)
- Parsing JSON / XML
- Stockage de données SQLite
- Multithreading
- ListView

Il est fortement conseillé d'avoir le cours à portée de main et de ne pas hésiter à lire la documentation officielle http ://d.android.com ainsi que la multitude de tutoriaux disponibles.

1 Présentation de l'API star

La star a fait de gros efforts d'ouverture des données (open data) ce qui permet à tous les développeurs d'utiliser les données dans leurs applications (qu'elles soient libres ou fermées, gratuites ou payantes, avec ou sans pub ...).
Les données disponibles sont nombreuses, avec entre autres les horaires des bus, la géolocalisation des arrêts, les infos trafic, le nombre de vélos disponibles dans chaque station de vélostar ...
Les données se présentent sous deux formes : des téléchargements pour les données statiques (.txt facilement parsable) et un webservice pour les données temps réel.
La liste des données disponibles est consultable à cette adresse : http ://data.keolis-rennes.com/fr/les-donnees/donnees-et-api.html
Dans ce TP, nous n'allons travailler qu'avec le webservice, n'hésitez pas à jeter un oeil aux données téléchargeables (ainsi qu'aux données publiques du site data.gouv.fr) pour ...un projet par exemple ?

2 Utilisation de l'API star

L'API star est un webservice classique : l'accès aux données se fait via des requêtes HTTP. La réponse est donnée au choix en XML ou en JSON (dans ce TP on n'utilisera que la partie JSON, plus simple)
Toutes les requêtes se présentent sous la même forme :
http ://data.keolis-rennes.com/json/ ?version=2.0&key=1RJLZ38TUFZSWTW&cmd=getbikestations&station=all

- http ://data.keolis-rennes.com : adresse de base de l'API
- /json : demande un retour en JSON (à remplacer par xml si besoin)
- / ?version=2.0 : défini la version de l'API à utiliser (1.0/2.0/2.1, cf doc)
- &key=1RJLZ38TUFZSWTW : chaque application est identifiée à partir de cette clé (permet d'avoir des statistiques et d'appliquer des quotas). Vous pouvez obtenir une clé gratuitement en vous inscrivant sur le site.
- &cmd=getbikestations : commande à éxecuter (cf doc)
- &station=all : un ou plusieurs paramètres pour préciser la demande (optionnel, cf doc)

Pour récupérer les données il faut donc :
- Construire l'URL cible en fonction des données à récupérer
- Exécuter une requête HTTP vers cette URL
- Parser (= analyser) le contenu JSON / XML de la réponse

3 Hello API !

- Créer un nouveau projet avec une activity
- Ajouter la permission INTERNET au manifest
- Dans un premier temps nous n'allons pas nous occuper de l'UI et simplement afficher le résultat d'une requête dans la log
- Ouvrir le lien [Lien trop long] dans un navigateur
- (Des sites de visualisation de JSON / XML comme http ://jsonviewer.stack.hu/ peuvent être utiles)
- Utiliser HttpURLConnection pour faire une requête HTTP vers cette adresse

Pour lire le contenu d'un InputStream (flux entrant), on peut utiliser le code suivant :

```
public String readStream(InputStream inputStream) {
    BufferedReader reader = new BufferedReader(new InputStreamReader(inputStream);
    String ligne = null;
    String contenu;
    while ((ligne = reader.readline()) != null) {
        contenu += ligne;
    }
}
```

- Afficher dans la log le contenu de la réponse.

4 Parsing des données

- Créer un objet JSONObject à partir des données récupérées précedemment
- En utilisant plusieurs fois la méthode getJSONObject et la méthode getInt, naviguer dans l'arbre JSON et récupérer la valeur du code retour fourni par le webservice.
- Afficher un Toast en fonction de la valeur de ce code (0 = OK, pour les autres codes voir la doc)
- En utilisant getJSONArray au bon moment, récupérer l'ensemble des stations de vélostar

1. Combien y a t'il de stations vélostar ?
2. Quel est le nom de la station ayant le numéro 55 ?
3. Combien a t'elle de vélos disponibles ? A t'elle des places libres pour rendre un vélo ?

- Créer une classe JAVA "Station" représentant une station vélostar
- Donner à cette classe les mêmes attributs que les données disponibles dans la réponse JSON
- Instancier, à partir du JSONObject, une liste d'objet JAVA "Station"

5 Stockage des données

L'objectif est de stocker les données chargées pour éviter de devoir toutes les télécharger à chaque fois (économie de temps, accès hors connexion, économie de batterie ...)

4. Quelle solution de stockage vous parait la plus adaptée ?

- Implémenter le nécessaire pour créer une base de donnée SQLite (Rappel : créer un SQLiteOpenHelper, implémenter onCreate et éventuellement onUpgrade)
- Dans onCreate, éxecuter les requêtes de création de tables pour créer une table Station avec tout ou une partie des attributs de la classe Station

5. Imaginons que l'API star change et fournisse des données en plus. Comment modifier la base de données SQLite si elle a déjà été crée sur l'appareil ?

- Exécuter une insertion avec des données de test

- Exécuter une “query” pour récupérer les données stockées et instancier des objets “Station” et les afficher dans la log
- Insérer les données récupérées via le webservice

6 Il est temps de faire les choses proprement

6. Quel point, vu dans les bonus, a été complétement ignoré depuis le début de ce TP ?

```
StrictMode.setThreadPolicy(new ThreadPolicy.Builder().detectNetwork().penaltyDeathOnNetwork().
    build());
```

- StrictMode est un outil d'aide aux développement permettant de détecter les méthodes longues susceptibles de bloquer un Thread.
- Le code ci-dessus crash l'application dès qu'un appel réseau est fait dans le Thread actif (ici, le Thread main / UI)
- Ajouer le code ci-dessus avant la requête HTTP, pleurer.
- Laisser le code ci-dessus et dépacer la requête HTTP dans un Thread séparé (utiliser new Thread(Runnable)), sourire.

```
StrictMode.setThreadPolicy(new ThreadPolicy.Builder().detectDiskWrites().penaltyDeath().build
    ());
```

- Anticiper les dégâts de l'ajout du code ci-dessus
- L'ajouter, pleurer.
- Déplacer les insertions en base de données dans un Thread séparé, sourire.

7 Afficher les données

7. Quel view / viewgroup vous parait le plus adapté pour afficher les stations ?
8. Quel adapter vous parait le plus adapté sachant que les données proviendront d'une base SQLite (et donc d'un Cursor) ?

- Implémenter votre choix
- Toutes les données ne rentreront pas dans l'affichage, choisir celles qui vous paraissent pertinentes
- Implémenter, comme dans le TP 2, une activity de visualisaition d'un station
- Faire le lien entre l'activity principale et l'activity de visualisation
- Dans l'activity de visualisation, afficher les données disponibles sur cette station
- Ajouter un bouton dans l'activity de visualisation pour mettre à jour les données sur la station
- Utiliser le filtre station de l'API star pour ne récupérer que les données de cette station

8 Pour aller plus loin

Wow, déjà tout fait ? Voici quelques pistes d'améliorations :

- Ajouter un bouton dans l'activity principale pour mettre à jour toutes les données
- Déplacer les boutons dans une Actionbar
- Utiliser la géolocalisation android et l'API de géolocalisation des stations pour proposer à l'utilisateur les x stations de vélo les plus proches de lui
- Afficher, dans l'activity de visualisation d'une station, une map googlemap (nécessite une clé google API et de tester sur des vrais appareils ou sur des émulateurs munis des google APIs)
- Les bonnes pratiques (guidelines) recommandent d'informer l'utilisateur lors des chargements et en particulier lors de l'accès au réseau. Implémenter un ProgressDialog pour informer l'utilisateur des chargements.