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 de visualisation de l'emploi du temps personnel pamplemousse.
Ce TP fera appel aux notions suivantes :

- Réseau / requêtes HTTP (webservice)
- Parsing JSON
- 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 Récupération des données pamplemousse

Sur pamplemousse, les étudiants consultent habituellement leur emploi du temps sous la forme d'un tableau affiché en HTML sur le site.
Le HTML mélange les données et leur présentation ce qui rend très fastidieuse son analyse dynamique (=parsing).
Pamplemousse propose heuresement aussi un fichier calendrier .ics contenant uniquement les données dans un format simple et standard. Il est possible d'obtenir ce calendrier sur cette page une fois connécté.
Ce fichier se révèle assez facile à analyser dynamiquement.
On peut donc récupérer automatiquement l'emploi du temps d'un étudiant en 3 étapes :

- S'authentifier sur pamplemousse (l'authentification se fait via le service d'authentification unique -SSO- de l'école)
- Télécharger le fichier .ics
- Analyser le fichier .ics pour extraire les informations sur les cours

Dans ce TP on va, pour simplifier, utiliser un webservice non officiel qui se chargera de la récupération et de la mise en forme des données.
Pour les curieux, le code utilisé par le webservice pour faire ces 3 étapes est disponible à cette adresse.

2 Utilisation du webservice

Le webservice va renvoyer la liste des prochains cours d'un étudiant ainsi que les informations sur ces cours (salle, professeur, horaires ...).

2.1 Principe d'un webservice

Un webservice est, comme son nom l'indique, un service accessible sur le web.
Techniquement, il s'agit d'un programme tournant sur un serveur et déstiné à diffuser des données brutes (contrairement au HTML, il n'y a pas d'informations sur la façon d'afficher ces données) via le protocole HTTP.

2.2 Appel au webservice

Un appel au webservice se fait via une requête HTTP GET à l'adresse
http://chessdiags.com/pamplemousse?login=XXX&mdp=YYY
XXX doît être remplacé par l'identifiant de l'étudiant (ex : id3182)
YYY doît être remplacé par le mot de passe de l'étudiant

2.3 Résultat du webservice

Le webservice renvoie des données JSON.
Un exemple du résultat d'un appel au webservice pour un étudiant de 1ère année est disponible à cette adresse.
Un exemple de résultat en cas d'erreur (login / mot de passe invalide, pamplemousse inaccessible ...) est disponible à cette adresse.

2.4 Visualiser le JSON

Même si le format JSON est relativement clair et simple, il reste assez difficile d'obtenir une vision globale des données présentes dans un résultat JSON d'un seul coup d'oeil.
Des outils de visualisation JSON existent et sont très utiles dès lors que l'on manipule des fichiers JSON.
Vous pouvez par exemple utiliser http://jsonviewer.stack.hu/

2.5 Webservice de test

Afin de ne pas surcharger le serveur pamplemousse (chaque requête au webservice donne lieu à 3 requêtes sur pamplemousse), il vous est demandé d'utiliser autant que possible l'url de test http://chessdiags.com/exemple.json qui renvoie toujours l'emploi du temps d'un étudiant de 1A.

2.6 Mise en garde

Lors de l'utilisation du webservice, le mot de passe de l'étudiant circule en clair. Il vous est donc conseillé, afin d'éviter tout problème, soit de changer votre mot de passe avant d'utiliser le webservice soit de n'utiliser que l'url de test.

3 Hello World !

- Créer un nouveau projet avec une activity basique (ex : blankactivity)
- Ajouter la permission INTERNET au manifest
- Dans un premier temps nous n'allons pas nous occuper de l'UI et simplement afficher l'emploi du temps dans la log
- Dans le onCreate de votre Activity, ajouter la ligne suivante

```
1 StrictMode.setThreadPolicy(new ThreadPolicy.Builder().permitAll().build());
```

  Cette ligne sera expliquée plus tard dans le TP

- Utiliser HttpURLConnection pour faire une requête HTTP vers l'adresse du webservice de test

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

```
public String readStream(InputStream inputStream) throws IOException {
    BufferedReader reader = new BufferedReader(new
    InputStreamReader(inputStream));
    String ligne = null;
    String contenu = "";
    while ((ligne = reader.readLine()) != null) {
        contenu += ligne;
    }
    return contenu;
}
```

- Afficher dans la log le contenu de la réponse.

4 Parsing des données

4.1 Lire le JSON

- Créer un objet JSONObject à partir des données récupérées précedemment
- En utilisant la méthode getJSONArray, récupérer le tableau des cours
- Afficher, dans un Toast, le nombre de cours

4.2 Instancier les objets correspondants

- Créer une classe JAVA "Cours" représentant un cours de l'emploi du temps
- Donner à cette classe les mêmes attributs que les données disponibles dans la réponse JSON
- Instancier, à partir du JSONObject, une liste (ou un tableau) d'objet JAVA "Cours"

5 Stockage des données

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

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

- Implémenter le nécessaire pour créer une base de données SQLite (utiliser SQLiteOpenHelper)
- Dans onCreate, éxecuter une requête de création de table pour créer une table Cours avec tous les attributs de la classe Cours
- Exécuter une insertion avec des données fictives
- Exécuter une "query" pour récupérer les données stockées et instancier les objets "Cours" correspondant aux données, les afficher dans la log
- Insérer les données récupérées via le webservice

2. Imaginons qu'un jour pamplemousse change et fournisse des données en plus.
   Comment modifier la base de données SQLite si elle a déjà été crée sur l'appareil ?

6 Il est temps de faire les choses proprement

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

- Remplacer la ligne StrictMode ajoutée au début du TP par celle ci

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

- Relancer l'application, pleurer
- StrictMode est un outil d'aide au développement permettant de détecter les opérations longues susceptibles de bloquer un Thread.
- Le code ci-dessus indique à StrictMode de faire crasher l'application dès qu'un appel réseau est fait dans le Thread actif (ici, le Thread principal : main / UI). Depuis le niveau d'API 11, c'est le comportement par défaut.
  La ligne ajoutée en début de TP disait à StrictMode de tout laisser passer.
- Déplacer le bloc de code correspondant à la requête HTTP dans un Thread séparé (utiliser new Thread(Runnable)), sourire.

7 Afficher les données

7.1 Afficher la liste des cours

4. Quel view / viewgroup vous parait le plus adapté pour afficher les cours ?

- Ajouter cet élément au layout en lui faisant prendre toute la place en largeur / hauteur
- Il nous faut maintenant un adapter : créer une nouvelle classe "CoursAdapter" qui extends BaseAdapter
- Hériter de BaseAdapter implique d'écrire 4 méthodes
- Avant d'implémenter ces méthodes, construire un constructeur prenant un Context et une List de Cours et stocker ces valeurs dans des attributs de classe

getCount : renvoyer le nombre d'éléments de la liste

getItem : renvoyer le n-ième élément de la liste

getItemId : renvoyer position (il suffit que l'id de chaque item soit unique)

getView : ci-dessous un squelette d'une méthode getView. Il faut au préalable créer un layout "layoutcours" représentant un élement de la liste.

```
1   public View getView(int position, View convertView, ViewGroup parent) {
2           View view;
3
4           if (convertView == null) {
5               view = ((LayoutInflater) context.getSystemService(Context.
                    LAYOUT_INFLATER_SERVICE)).inflate(R.layout.layoutcours, parent, false);
6           } else {
7               view = convertView;
8           }
9           Cours cours = (Cours) getItem(position);
10
11          TextView nomCours = (TextView) view.findViewById(R.id.nomcours);
12          nomCours.setText(cours.getNom());
13
14          return view;
15      }
```

- Instancier un CoursAdapter et l'affecter à la ListView via listView.setAdapter
- Il est possible d'affiner le layout de chaque item en rajoutant des champs et d'autres informations. Rajouter la date de début du cours.

7.2 Afficher le détail d'un cours

- Créer une activity de visualisation d'un cours en particulier
- En utilisant le bon listener, récupérer les clicks faits sur un élément de la ListView
- Lors d'un click sur un élément, ouvrir l'activity de visualisation

6. Il y a t'il des paramètres à passer à l'activity de visualisation ?

- Rajouter dans l'intent les données nécessaires à l'activity de visualisation
- Dans le onCreate de l'activity de visualisation, récupérer les données de l'intent
- A partir des données de l'intent, récupérer en base les données correspondant à ce cours
- Afficher ces données (vous êtes libres du layout)

8 Pour aller plus loin

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

- Ajouter un bouton dans l'activity principale pour recharger les données
- Créer un écran de gestion des préférences pour que l'utilisateur puisse renseigner son identifiant / mot de passe
- 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.