Application TodoList

Electif PMR 2019

Afin d'éviter des lourdeurs inutiles, je vous présente ici les modifications que j'ai apporté au précédent projet.

MainActivity :

En utilisant la fonction fournie sur Moodle, on met en place dans OnCreate() :

```
if(verifReseau()){
    btnOK.setVisibility(View.VISIBLE);
}
```

On ajoute au layout la barre pour entrer le mot de passe, en textPassword :

```
<EditText
    android:id="@+id/edt_password"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_weight="1"
    android:ems="10"
    android:hint="Mot de passe"
    android:inputType="textPassword" />
```

On ajoute également une progressbar, dont on lance l'affichage au début de l'asynctask, et qu'on désactive à sa fin.

```
<ProgressBar
    android:id="@+id/progess"
    android:layout_width="64dp"
    android:layout_height="74dp"
    android:visibility="gone"/>
```

En suivant les morceaux de code donnés dans Moodle, on créer une classe AsyncTask interne à MainActivity. Lors du doInBackground, on appelle une fonction getHash que l'on a coder dans DataProvider.

```
//On renvoie le hash si l'utilisateur existe, sinon on renvoi ""
public String getHash(String myUrl, String pseudo, String motDePasse, String methode) {
    String URL = myUrl + "/authenticate?user=" + pseudo + "&password=" + motDePasse;
    String sJson = requete(URL,methode);
    JSONObject oJson = null;
    try {
        oJson = new JSONObject(sJson);
        if (oJson.getBoolean( name: "success")) {
            return oJson.getString( name: "hash");
        }
        else {
            return "";
        }
    } catch (JSONException e) {
        e.printStackTrace();
        return "erreur avec l'objet Json";
    }
}
```

Cette méthode sert à renvoyer le hash de l'utilisateur, si ses identifiants sont reconnu par l'API. On s'en sert donc lors de la vérification de la validité des identifiants.

Cette méthode fait appel à la fonction « requete() » que l'on a modifié :

```
public String requete(String qs,String methode) {
    if (qs != null)
    {
        try {
            URL url = new URL(qs);
            HttpURLConnection urlConnection = (HttpURLConnection) url.openConnection();
            urlConnection.setRequestMethod(methode);
            InputStream in = new BufferedInputStream(urlConnection.getInputStream());
            String txtReponse = convertStreamToString(in);
            urlConnection.disconnect();
            return txtReponse;
        } catch (MalformedURLException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }

    }
    return "";
}
```

Ainsi, on peut choisir la méthode que l'on va employer lors de la requête faite à l'API. De plus, l'url de l'api est directement donnée à la fonction. Ce qui la rend très générale est utilisable dans toute situation.

Une fois le hash récupéré, on le met dans les settings et on lance ChoixListActivity.

SettingsActivity :

La gestion de SettingsActivity une fois encore se lance dans le menu que l'on crée de la même manière que pour l'ancienne application. Là où précédemment j'avais fait le choix de coder SettingsActivity qui extends AppCompatActivity, cette fois ci j'ai essayé de la faire extends PreferenceActivity afin de pouvoir y mettre la gestion de l'url à suivre pour l'API, comme le demande la consigne.

```
<PreferenceScreen xmlns:android="http://schemas.android.com/apk/res/a
    <EditTextPreference
        android:title="URL API REST"
        android:key="url"
        android:defaultValue="http://tomnab.fr/todo-api"
        />

</PreferenceScreen>
```

Malheureusement la gestion des préférences, entre cette classe et l'utilisation de SharedPreferences m'ont perdu et je n'ai pas compris l'utilisation de cette fonctionnalité. Je n'ai pas trouvé de documentation suffisamment claire sur internet pour pouvoir m'éclairer là dessus. Le problème que je n'arrive pas à résoudre, est de savoir comment récupérer ce que l'utilisateur entre dans la fenêtre ouverte dans préférence. En récupérant cela, on pourrait le récupérer dans chaque activité et ainsi simplement changer l'adresse de l'API.

ChoixListActivity :

J'ai souhaité garder la même « architecture » des activités ChoixListActivity et ShowListActivity. En effet, ainsi la manière d'implémenter le RecyclerView n'est pas modifié et les changements que j'ai apporté au code sont uniquement la communication entre l'application et l'API ainsi que la gestion des informations récupérées.

```
@Override protected JSONArray doInBackground(Object... objects) {
    return  (new DataProvider()).getLists(myUrl,mHash);
}

@Override protected void onPostExecute(JSONArray myJson) {
    super.onPostExecute(myJson);
    myJsonList = myJson;
    //on ajoute a liste de todos (qu'on enverra à l'adapteur) chacune
    // des todos envoyés par le serveur. On a crée un nouvel attribut dans la classe ListeTodo
    //et on a crée un nouveau constructeur.
    try {
        for(int i=0; i<myJsonList.length();i++){
            myListeTodo.add(new ListeToDo(myJsonList.getJSONObject(i).getString( name: "id"),
                    myJsonList.getJSONObject(i).getString( name: "label")));
        }
    } catch (JSONException e) {e.printStackTrace();}
    // on ajoute a l'adapter les todolist propre au pseudo précédemment séléctionné
    mAdapter = new PseudoItem(myListeTodo);
    mRecyclerView.setLayoutManager(mLayoutManager);
    mRecyclerView.setAdapter(mAdapter);
```

On lance dès la création de l'activity une asynctask qui va demander la liste des TodoList associé au hash précédemment reçu. Dans ce thread, on ira récupérer d'abord la liste JSONArray renvoyé par le serveur grâce à la fonction getLists, très similaire à getHash.

On crée ensuite en postExecute une liste de ListeToDo, que l'on vient mettre dans l'adapteur, afin que le RecyclerView puisse l'exploiter.

En mettant en place un onClickListener sur le bouton pour ajouter une todolist, on met en place une nouvelle classe AsyncTask qui servira à demander au serveur de créer une nouvelle liste associé au hash.

J'ai rencontré un problème à ce moment. Une asynctask ne peut être appelé qu'une seule fois. Cela implique que le bouton n'est « fonctionel » qu'une seule fois. De plus, une fois que l'on a crée la liste, l'affichage dans le recyclerview n'est pas mis à jour automatiquement.

Pour parer à ce problème que j'avais déjà rencontré lors de la première application, j'ai choisi de relancer l'activité grâce à un intent dirigé vers elle-même. J'ai cependant désactivé l'animation d'apparition d'activité afin de réduire la nuisance.

```
protected void onPostExecute(String string) {
    super.onPostExecute(string);
    Intent intent = getIntent();
    overridePendingTransition( enterAnim: 0,  exitAnim: 0);
    intent.addFlags(Intent.FLAG_ACTIVITY_NO_ANIMATION);
    finish();
    overridePendingTransition( enterAnim: 0,  exitAnim: 0);
    startActivity(intent);
}
```

Le procédé mis en place est exactement le même dans ShowListActivity.

ShowListActivity :

Pour ShowListActivity, j'ai mis en place une troisième classe Asyntask, qui elle sert à prévenir le serveur lorsqu'il faut changer le checked d'une classe lorsque la checkbox est touchée par l'utilisateur. La requête est directement faite depuis le doInBackground. Une fois encore, il faut relancer l'activité de la même manière pour pouvoir mettre à jour l'affichage.

Conclusion :

L'application est fonctionnelle et les problèmes rencontrés lors de la première version sont résolu par la simplification de gestion qu'à apporter l'API. Cependant, il reste comme optique d'amélioration la meilleure gestion de PreferenceActivity afin de pouvoir laisser la possibilité à l'utilisateur de changer l'adresse de l'API à souhait.