Application **MyMeteo**

L'objectif de ce TP est de découvrir les fonctionnalités de la librairie Volley qui a été conçue afin de gêrer efficacement les connexions réseaux effectuées par une application. Attention ! cette librairie n'est pas adaptée si votre application doit télécharger des fichiers volumineux ou si votre objectif est de faire du streaming. La librairie Volley est appropriée pour les requêtes réseaux de type interrogation d'une BD et récupération d'un résultat concis.

Le manuel d'utilisation de la librairie Volley est disponible sur le site Android developer :

Transmitting Network Data Using Volley

Votre application sera constituée d'un **RelativeLayout** contenant une zone de type **TextView**, une zone de type **Button** et une zone de type **ImageView**. En cliquant sur le bouton **Go**, votre application affichera la température actuelle de la ville de Toulon, ainsi qu'une image représentant l'état du ciel (cf. Fig.1).

Afin de recueillir ces données, votre application devra interroger le site www.prevision-meteo.ch qui permet de récupérer au format JSON des données météo.

Ouvrez dans un navigateur l'url suivante afin d'observer le format des résultats renvoyés :

http://www.prevision-meteo.ch/services/json/toulon

PARTIE 1

Dans un premier temps, on ne s'intéressera qu'à l'affichage de la température et du texte associé sur les conditions météo. Afin que votre application puisse effectuer une connexion réseau il faut ajouter une autorisation dans le fichier **AndroidManifest.xml** :

```
<application
…..
</application>
<uses-permission android:name="android.permission.INTERNET" />
```

La librairie Volley gêre pour votre application les connexions réseau via une file d'attente qu'il faut initialiser :

```
RequestQueue file = Volley.newRequestQueue(this);
```

Il faut ensuite construire votre requête. Volley a été developpé pour prendre en compte les requêtes de type JSON , i.e. les requêtes qui envoient des données au format JSON et les requêtes qui reçoivent des données au format JSON (ce qui est notre cas). Vous devrez pour cela déclarer un objet de type **`JsonObjectRequest`** dont les paramètres du constructeur sont :

- *method* : the HTTP method to use (use the **`Request.Method.xxx`** `constant`).
- *url* : URL to fetch the JSON from (String).
- *jsonRequest* : A JSONObject to post with the request. **`null`** is allowed and indicates no parameters will be posted along with request.

- *listener* : Listener to receive the JSON response
- *errorListener* : Error listener, or null to ignore errors.

Pour les deux écouteurs (les deux dernières paramètres), vous spécifierez **this** et vous préciserez que votre classe principale implémente les écouteurs **`Response.Listener`**, et **`Response.ErrorListener`**.

Il vous faudra donc, au sein de votre application, surcharger les méthodes **onErrorResponse(VolleyError error)** et **onResponse(JSONObject response)**.

C'est dans la méthode **onResponse** que vous devrez afficher la température récupérée. Le paramètre de cette méthode correspond à la réponse JSON reçue suite à votre requête. Utilisez alors les méthodes **getJSONObject** et **getString** de la classe **JSONObject** pour récupérer la valeur de la température.

Une fois la requête construite, il faut la rajouter à la file d'attente initialement créée via la méthode **add** de l'objet **RequestQueue**.

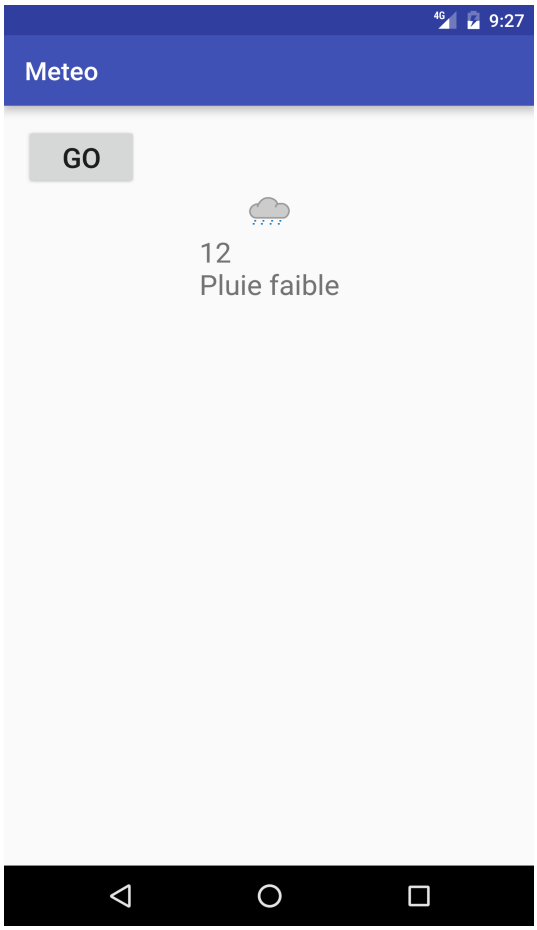

Fig. 1

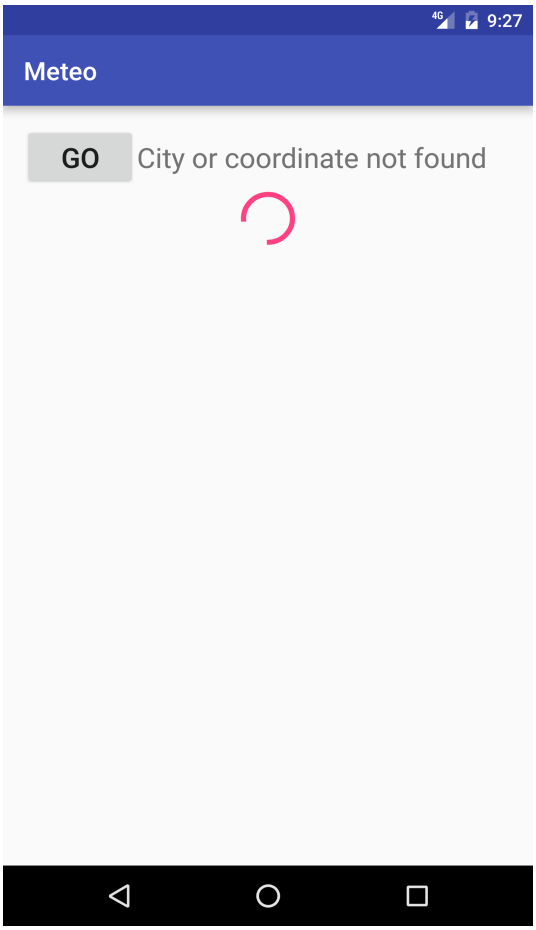

Fig. 2

PARTIE 2

Il s'agit dans cette partie de rajouter dynamiquement une barre de progression (sous forme circulaire) afin d'indiquer à l'utilisateur qu'une connexion réseau s'effectue. Cette barre disparaîtra une fois le résultat de la requête JSON reçue.

Pour créer une barre de progression qui n'a pas été déclarée au préalable dans le fichier XML qui décrit l'aspect de votre application, il faut :

- déclarer un nouvel objet de type `ProgressBar` en spécifiant pour contexte votre classe principale (**this**),
- appliquer des paramètres de layout à ce nouvel objet afin de spécifier qu'il doit se trouver sous le bouton « Go »,
- spécifier que votre barre doit être visible via la méthode `setVisibility`,
- ajouter votre barre de progression à votre layout principal.

Une fois la réponse JSON reçue, il faut penser à faire disparaître la barre de progression avant de faire apparaître la température.

PARTIE 3

La réponse JSON reçue contient l'url d'une image représentant l'état courant du ciel (valeur associée à la clé « icon_big »). Pour télécharger cette image, il va falloir utiliser un objet de type **`ImageRequest`** dont les paramètres du constructeur sont :

- *url :* URL of the image,
- *listener* : Listener to receive the decoded bitmap,
- *maxWidth* : Maximum width to decode this bitmap to, or zero for none,
- *maxHeight* : Maximum height to decode this bitmap to, or zero for none,
- *scaleType* : The ImageViews ScaleType used to calculate the needed image size,
- *decodeConfig* : Format to decode the bitmap to,
- *errorListener* : Error listener, or null to ignore errors.

Les paramètres *maxWidth* et *maxHeight*, s'ils ne sont pas nuls, précise à la librairie Volley qu'il faut redimensionner l'image une fois qu'elle aura été téléchargée.

Le paramètre *scaleType* indique quelle option appliquer pour placer l'image dans la vue la contenant, cf :

https://developer.android.com/reference/android/widget/ImageView.ScaleType.html)

On placera **null** dans ce champ étant donné que dans le fichier XML on aura déclaré une **ImageView** dont la taille devra s'adapter à son contenu.

Le paramètre *decodeConfig* indique comment interpréter l'encodage des pixels de l'image. Les images étant encodées en RGB avec un canal de transparence (donc encodée sur 4 octets), il faut utiliser pour ce paramètre la constante : **Bitmap.Config.ARGB_8888**, cf.

https://developer.android.com/reference/android/graphics/Bitmap.Config.html

Concernant le *listener* vous spécifierez que ce doit être votre application (**this**). Il faudra donc que votre méthode implémente **onResponse**. Or celle-ci a déjà été implémenté pour gêrer le résultat de la requête JSON. Vous allez donc changer le paramètre de cette méthode en spécifiant qu'elle ne

traite plus un objet de type **JSONObject** mais un objet général de type **Object**. En utilisant l'opérateur **instanceof** de Java, vous testerez à l'intérieur de votre méthode **OnResponse**, si l'objet reçu est de type **JSONObject** ou de type **Bitmap**. S'il est de type **Bitmap**, alors vous appliquerez la méthode **setImageBitmap** de l'objet **ImageView** associé à la vue **ImageView** de votre layout pour afficher l'image reçue.

PARTIE 4

Pour terminer, on souhaite gêrer les erreurs d'interrogation du serveur. Attention, il ne s'agit pas de considérer les erreurs dûes à un problème de connexion au serveur, ces erreurs sont à gêrer dans la méthode **onErrorResponse**. Il s'agit de prendre en compte les erreurs dues au fait que l'interrogation porte sur une ville non référencée par le site (cf. Fig. 2) . Interrogez directement (via un navigateur) le serveur avec une ville inconnue et observez le résultat obtenu. Vous constaterez que vous obtenez une réponse au format JSON. Il va donc falloir gêrer ces erreurs via la méthode **OnResponse**. Vous afficherez le message d'erreur reçu dans la Vue prévue initialement pour l'affichage de la température. Vous aurez modifié dynamiquement les paramètres de layout de cette vue afin que le message d'erreur s'affiche à droite du bouton et non pas en dessous.