Application Météo avec API REST et Retrofit

1. Introduction

Les applications modernes utilisent fréquemment des **API Web** pour récupérer des informations provenant de serveurs distants. Les applications de météo, de cartographie, de réseaux sociaux ou encore de commerce électronique reposent toutes sur ce principe de communication entre un client (application Android) et un serveur.

Dans cette application, nous allons développer une **application météo** capable de récupérer en temps réel les conditions météorologiques d'une ville grâce à une **API REST**. Pour simplifier les échanges entre Android et le serveur, nous utiliserons la bibliothèque **Retrofit**, qui est aujourd'hui l'une des solutions les plus populaires pour effectuer des requêtes HTTP.

> **Objectif pédagogique :** Cette application permet de découvrir le fonctionnement des API REST, la communication HTTP, le format JSON, ainsi que l'utilisation de Retrofit pour récupérer et afficher des données dynamiques dans une application Android.

2. Qu'est-ce qu'une API REST ?

Une **API (Application Programming Interface)** est un ensemble de services permettant à deux applications de communiquer entre elles.

Une **API REST** (Representational State Transfer) est un service Web qui utilise le protocole HTTP pour échanger des données. Les informations sont généralement transmises au format **JSON (JavaScript Object Notation)**, un format léger, lisible et largement utilisé sur Internet.

Le fonctionnement est simple :

1. L'application Android envoie une requête HTTP.
2. Le serveur traite cette requête.
3. Le serveur renvoie une réponse au format JSON.
4. Android interprète les données et les affiche à l'utilisateur.

3. Architecture de l'application

La communication entre Android et le serveur peut être représentée par le schéma suivant :

apcpedagogie.com

```
Utilisateur
      |
      ▼
Application Android
      |
      ▼
Retrofit
      |
      ▼
API REST
      |
      ▼
Serveur météo
      |
      ▼
Réponse JSON
      |
      ▼
Affichage des données
```

Cette architecture permet de séparer clairement la couche d'affichage, la couche réseau et la source de données.

4. Les dépendances Retrofit

Avant d'utiliser Retrofit, il est nécessaire d'ajouter les dépendances dans le fichier **build.gradle**.

```
implementation 'com.squareup.retrofit2:retrofit:2.9.0'
implementation 'com.squareup.retrofit2:converter-gson:2.9.0'
```

Explication :

- **Retrofit** gère les requêtes HTTP.
- **Gson Converter** convertit automatiquement les données JSON en objets Java.

5. Création de l'interface API

Retrofit utilise une interface Java pour décrire les requêtes HTTP.

```
public interface WeatherService {
```

apcpedagogie.com

```
    @GET("weather")
    Call<WeatherResponse> getWeather(
            @Query("q") String city,
            @Query("appid") String apiKey,
            @Query("units") String units
    );

}
```

Les annotations utilisées sont :

- `@GET`: indique une requête HTTP GET.
- `@Query`: ajoute un paramètre dans l'URL.
- `Call<WeatherResponse>`: représente la réponse attendue.

6. Création de Retrofit

Retrofit doit être initialisé avec l'adresse de l'API.

```
Retrofit retrofit = new Retrofit.Builder()
        .baseUrl("https://api.openweathermap.org/data/2.5/")
        .addConverterFactory(GsonConverterFactory.create())
        .build();

WeatherService service = retrofit.create(WeatherService.class);
```

La méthode `baseUrl()` indique l'adresse principale du serveur tandis que `GsonConverterFactory` permet de convertir automatiquement le JSON reçu.

7. Interface utilisateur

L'application météo possède une interface simple composée de :

- Un `EditText` pour saisir le nom d'une ville.
- Un bouton permettant de lancer la recherche.
- Des `TextView` affichant les informations météo.

```
<EditText
    android:id="@+id/editVille"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Nom de la ville"/>
```

apcpedagogie.com

```
<Button
    android:id="@+id/btnRechercher"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Rechercher"/>

<TextView
    android:id="@+id/txtTemperature"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Température"/>

<TextView
    android:id="@+id/txtDescription"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Description"/>
```

8. Envoi de la requête HTTP

Lorsque l'utilisateur clique sur le bouton, Retrofit envoie une requête vers le serveur.

```
service.getWeather(ville, API_KEY, "metric")
.enqueue(new Callback<WeatherResponse>() {

    @Override
    public void onResponse(Call<WeatherResponse> call,
                           Response<WeatherResponse> response) {

        if(response.isSuccessful()){

            WeatherResponse weather = response.body();

            txtTemperature.setText(
                    weather.getMain().getTemp() + " °C");

            txtDescription.setText(
                    weather.getWeather().get(0).getDescription());

        }

    }
```

apcpedagogie.com

```
    @Override
    public void onFailure(Call<WeatherResponse> call,
                          Throwable t) {

        txtDescription.setText("Erreur de connexion");

    }

});
```

Analyse :

- `enqueue()` exécute la requête de manière asynchrone.
- `onResponse()` est appelée lorsque le serveur répond correctement.
- `onFailure()` est exécutée en cas d'erreur réseau.

> **Important :** L'application doit disposer de la permission Internet dans le fichier `AndroidManifest.xml` afin de pouvoir communiquer avec une API distante.

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

9. Le format JSON

La réponse envoyée par le serveur est généralement au format JSON.

```
{
   "name":"Paris",
   "main":{
      "temp":26.8,
      "humidity":60
   },
   "weather":[
      {
         "description":"clear sky"
      }
   ]
}
```

apcpedagogie.com

Grâce à Gson, ce document JSON est automatiquement transformé en objets Java accessibles directement dans l'application.

10. Fonctionnement global

Le déroulement de l'application est le suivant :

1. L'utilisateur saisit le nom d'une ville.
2. Il clique sur **Rechercher**.
3. Retrofit envoie une requête HTTP GET.
4. Le serveur météo traite la demande.
5. Le serveur renvoie les données au format JSON.
6. Retrofit convertit automatiquement le JSON en objets Java.
7. L'application affiche la température, la description et les informations météo.

11. Avantages de Retrofit

- Code plus propre et plus lisible.
- Gestion automatique des requêtes HTTP.
- Conversion automatique du JSON avec Gson.
- Exécution asynchrone des requêtes.
- Maintenance facilitée.
- Très utilisé dans les applications Android professionnelles.

> **Bonnes pratiques :** Il est recommandé de vérifier la disponibilité de la connexion Internet, de gérer les erreurs HTTP (404, 500, etc.), d'afficher un indicateur de chargement (*ProgressBar*) pendant la récupération des données et de protéger la clé API en évitant de la stocker directement dans le code source.

12. Compétences acquises

À travers cette application, l'étudiant apprend à :

- Comprendre le fonctionnement des API REST.
- Utiliser Retrofit pour effectuer des requêtes HTTP.
- Manipuler les données au format JSON.
- Créer des interfaces réseau en Java.
- Afficher des données dynamiques provenant d'un serveur distant.

apcpedagogie.com

- Gérer les réponses et les erreurs réseau.
- Développer des applications Android connectées à des services Web.

Rédigé et structuré par : **Med Khalil Kribi**

Source : https://apcpedagogie.com/application-meteo-api-rest-retrofit-android · téléchargé le 07/11/2026 · © APC Pédagogie · apcpedagogie.com

apcpedagogie.com