ALKendi
CENTRE DE PRÉPARATION BTS

Développement Mobile

TP6

Nom et Prenom: Mohamed Amine BARAKAT

16 Janvier 2026

2025/2026

1. Introduction

Ce travail pratique a pour objectif de concevoir et développer une application mobile Android permettant à l'utilisateur de saisir le nom d'une ville et d'afficher les prévisions météorologiques correspondantes. L'application repose sur l'utilisation d'une **API REST**, en l'occurrence l'API **OpenWeather**, afin de récupérer les données météo à distance via le protocole **HTTP**.

Les données retournées par le serveur sont fournies au format **JSON**, puis analysées (parsing) et transformées en objets Java pour être affichées dynamiquement dans l'interface graphique à l'aide d'un **ListView** et d'un **Adapter personnalisé**.

Ce TP permet également de comprendre :

- la communication client–serveur,
- la gestion des permissions Android,
- l'utilisation de la bibliothèque **Volley**,
- le cycle de vie d'une **Activity Android**.

2. Réponses aux questions demandées

Permission Internet

Si la permission Internet n'est pas ajoutée, l'application ne pourra pas communiquer avec le serveur distant. Les requêtes réseau échoueront et aucune donnée météo ne sera affichée.
Android impose cette permission afin de protéger l'utilisateur contre l'utilisation abusive du réseau, la consommation non contrôlée des données mobiles et les risques liés à la sécurité.

Interface graphique

Le fichier `activity_main.xml` définit l'interface principale de l'application. Il décrit l'organisation des composants graphiques visibles à l'écran, tels que le champ de saisie, le bouton et la liste d'affichage.

Un layout séparé est utilisé pour les éléments de la liste afin de personnaliser l'affichage de chaque prévision météo. Cela permet d'afficher plusieurs informations par ligne avec une meilleure organisation.

Un `TextView` sert uniquement à afficher du texte, tandis qu'un `EditText` permet à l'utilisateur de saisir du texte.

Modèle de données

Il n'est pas recommandé d'utiliser directement le JSON dans l'interface car il est difficile à manipuler et à maintenir. La conversion en objets Java permet une meilleure organisation et une manipulation plus simple des données.

La programmation orientée objet permet de structurer les données, d'améliorer la lisibilité du code et de faciliter l'évolution de l'application.

Adapter personnalisé

Un Adapter permet de relier une source de données à une vue graphique.
La méthode `getView()` est utilisée pour créer et personnaliser l'affichage de chaque élément de la liste.
Un layout personnalisé est nécessaire afin d'afficher plusieurs informations pour chaque prévision météo avec une mise en forme adaptée.

Volley

La classe `RequestQueue` gère l'envoi et l'exécution des requêtes réseau.
Les requêtes sont asynchrones afin de ne pas bloquer l'interface utilisateur pendant la récupération des données.
En cas d'erreur réseau, une méthode spécifique est appelée pour gérer l'erreur et informer l'utilisateur.

Traitement JSON

Un `JSONObject` représente un objet contenant des paires clé–valeur, tandis qu'un `JSONArray` représente une liste d'objets.
Le bloc `try/catch` permet d'éviter le crash de l'application en cas d'erreur lors de la lecture des données.
Si une clé JSON n'existe pas, une exception est levée.

Affichage des données

Le `ListView` permet d'afficher une liste de données dynamiques.
La méthode `notifyDataSetChanged()` informe l'Adapter que les données ont été modifiées.
Sans cet appel, les nouvelles données ne s'affichent pas à l'écran.

3. Captures d’écran commentées

1. Interface principale : activity_main.xml

L’interface principale de l’application est définie dans le fichier **activity_main.xml**.
Elle représente l’écran principal avec lequel l’utilisateur interagit dès le lancement de l’application.

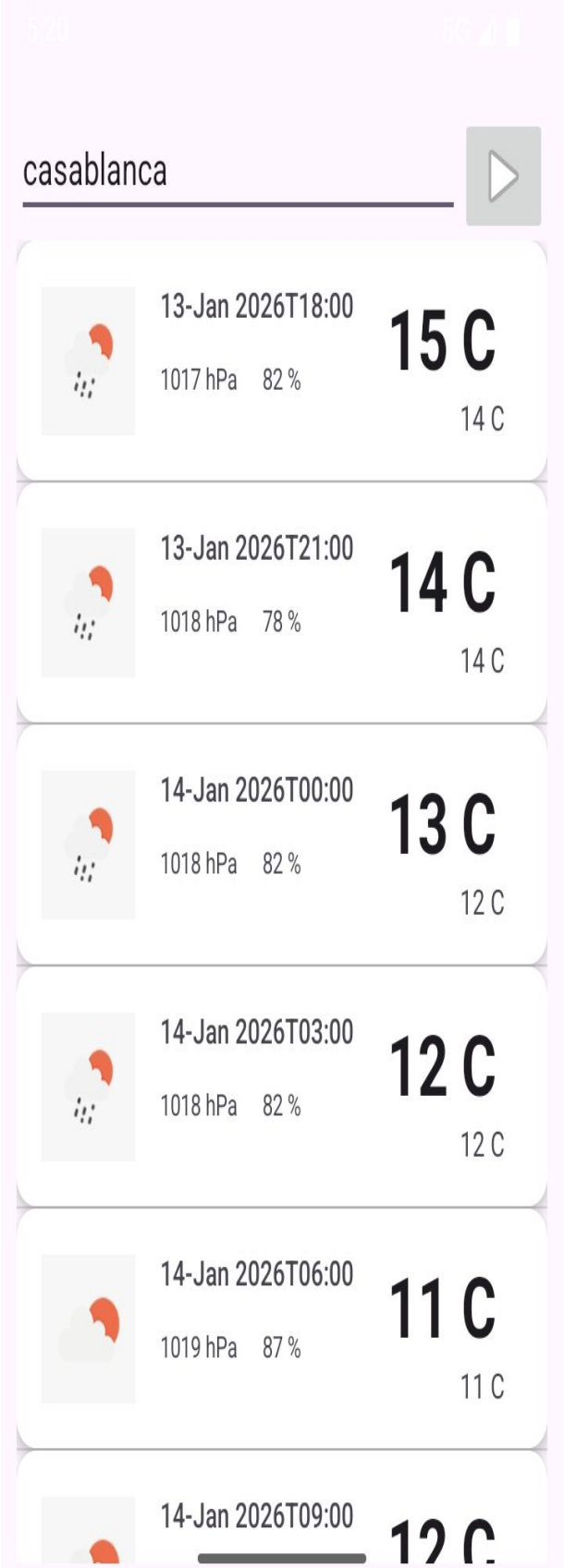
Cette interface est organisée à l’aide d’un **LinearLayout vertical**, ce qui permet d’afficher les composants les uns en dessous des autres de manière simple et lisible.

Elle contient :

- **Un champ de saisie (EditText)** permettant à l’utilisateur d’entrer le nom de la ville dont il souhaite consulter la météo.
- **Un bouton (ImageButton)** servant à lancer la recherche des prévisions météorologiques après la saisie de la ville.
- **Un ListView** destiné à afficher dynamiquement la liste des prévisions météo récupérées depuis l’API.

Cette interface a pour rôle principal :

- de permettre la saisie des données par l’utilisateur,
- de déclencher l’appel à l’API météo,
- d’afficher les résultats sous forme de liste claire et organisée.



2. Interface des éléments de la liste : list_item_layout.xml

La deuxième interface est définie dans le fichier **list_item_layout.xml**.
Elle correspond à **l'affichage d'un seul élément** de la liste des prévisions météo dans le `ListView`.

Chaque élément de la liste est composé de :

- **Une ImageView** affichant une icône représentant l'état météo (soleil, nuages, pluie, etc.).
- **Un TextView pour la date et l'heure** de la prévision.
- **Plusieurs TextView** affichant :
  - la température minimale,
  - la température maximale,
  - la pression atmosphérique,
  - le taux d'humidité.

Cette interface permet :

- une présentation détaillée et lisible de chaque prévision météo,
- une séparation claire entre la structure de la liste et l'interface principale,
- une personnalisation complète de l'affichage de chaque ligne grâce à l'Adapter personnalisé.

4. Conclusion

Ce TP a permis de développer une application Android complète intégrant une communication réseau, le traitement des données JSON et leur affichage dynamique dans une interface utilisateur. Il a renforcé la compréhension du fonctionnement des API, de la programmation orientée objet, de l'utilisation des Adapters et du cycle de vie des Activities Android.

Cette expérience constitue une base solide pour la réalisation d'applications mobiles plus avancées et professionnelles.