AVIGNON UNIVERSITÉ

TP3 : Interrogation de service web

uapv2001561
BOUDOUNT Youssef

19/03/2024

Licence Informatique L3
Parcours LININGLOG : INGENIERIE LOGICIELLE
UE Applications Mobiles

Responsable
MERLIN Eric
VERNET Mathilde

UFR
SCIENCES
TECHNOLOGIES
SANTÉ

CERI

CENTRE
D'**ENSEIGNEMENT**
ET DE **RECHERCHE**
EN **INFORMATIQUE**
ceri.univ-avignon.fr

Sommaire

1 Introduction

Ce rapport documente le développement et la fonctionnalité de l'application Android créée dans le cadre du TP3 de ue applications mobiles. L'objectif principal de ce travail pratique est de développer une application permettant d'interroger un service web pour collecter des informations météorologiques actuelles pour différentes villes à partir d'une base de données. Ce projet permet d'approfondir la compréhension de l'analyse des réponses JSON à l'aide de la bibliothèque Retrofit et de son convertisseur Moshi.

2 Étapes de réalisation de l'application

La réalisation de l'application s'est déroulée en plusieurs étapes :

1. Préparation de l'environnement de développement : Téléchargement et extraction de l'archive TP3.zip contenant le code source Java et les fichiers ressources.
2. Configuration des dépendances : Ajout des permissions internet dans le fichier Manifest et inclusion des bibliothèques Retrofit et Moshi dans le script Gradle.
3. Création de l'interface OWMInterface : Définition des méthodes nécessaires pour interroger le service web openweathermap à l'aide de Retrofit.
4. Analyse JSON avec Moshi : Création de la classe WeatherResponse pour traiter les données météorologiques reçues en format JSON.
5. Implémentation de la logique métier : Développement des classes WeatherResult et WeatherRepository pour gérer les interactions avec le service web et la base de données.
6. Mise à jour de l'interface utilisateur : Codage des fonctionnalités permettant la mise à jour des écrans de l'application en réponse aux données reçues du service web.

3 Présentation de l'application

Cette section fournit des captures d'écran illustrant différentes fonctionnalités et interfaces de l'application. Chaque capture d'écran est accompagnée d'une brève description pour aider à comprendre le contexte et la fonctionnalité qu'elle représente.

3.1 Fonctionnalités de l'application

L'application offre plusieurs fonctionnalités clés :

- Affichage de la liste des villes : La base de données pré-remplie affiche les villes et leurs pays associés.

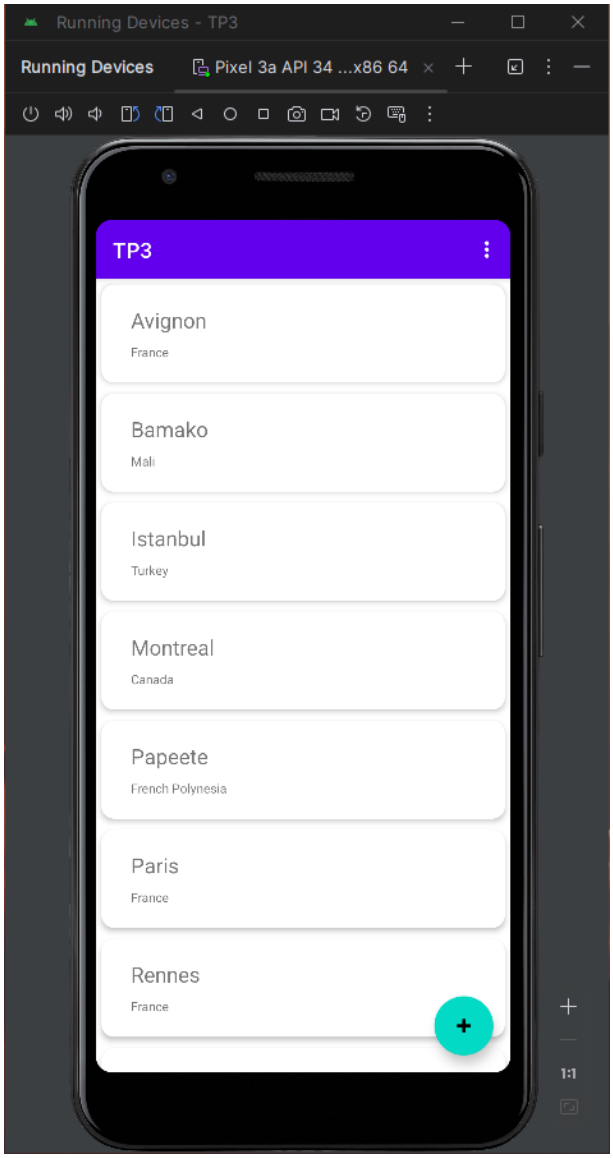

Figure 1. L'écran d'accueil de l'application montrant la liste des villes.(avant actualisation)

- Mise à jour des informations météorologiques : Les utilisateurs peuvent actualiser les informations météorologiques pour une ville sélectionnée ou pour toutes les villes via le service web.

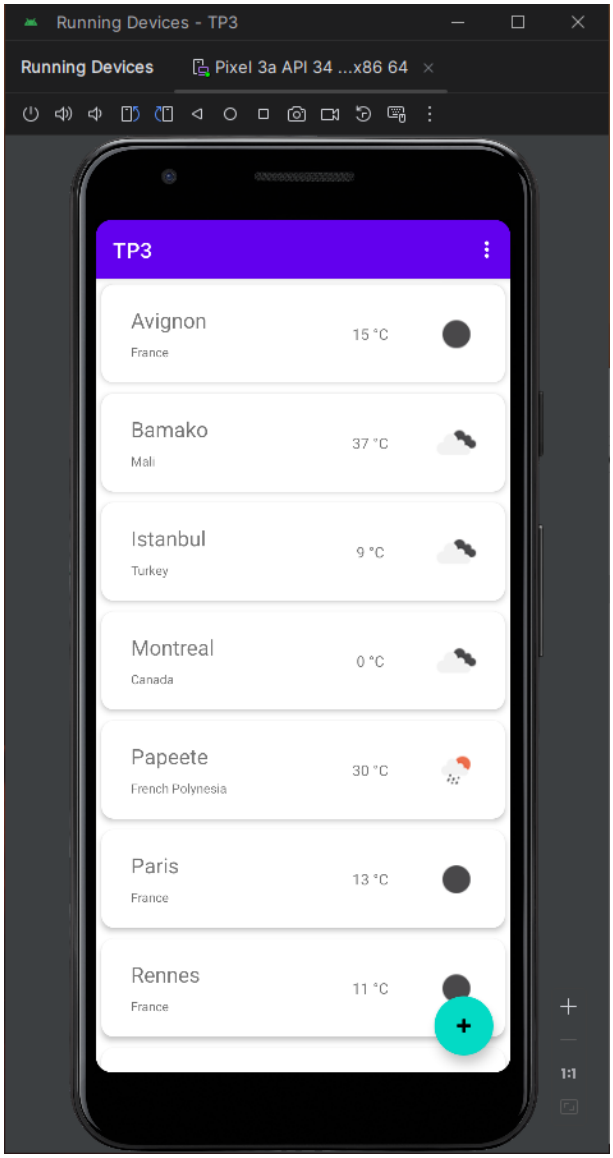

Figure 2. L'écran affichant les détails météorologiques pour une ville sélectionnée.

- Visualisation détaillée : Les détails de la météo, tels que la description et l'icône, sont affichés pour chaque ville après la mise à jour.

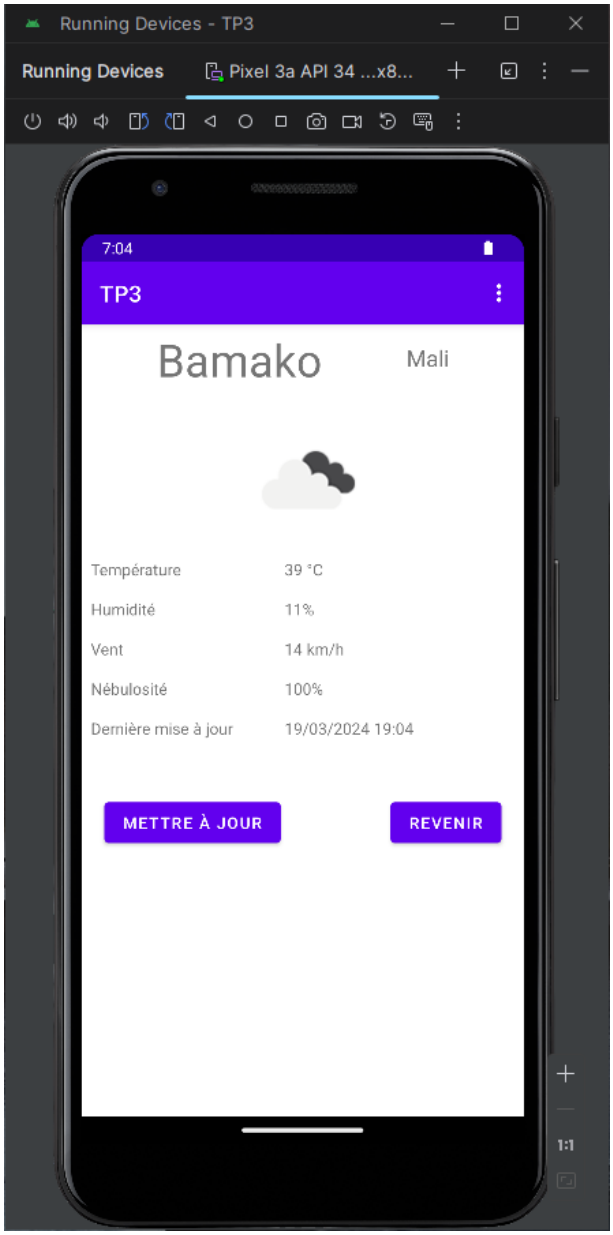

Figure 3. Visualisation détaillée de la météo

- Gestion des états de chargement : Des indicateurs de progression signalent à l'utilisateur que les données sont en cours de chargement.

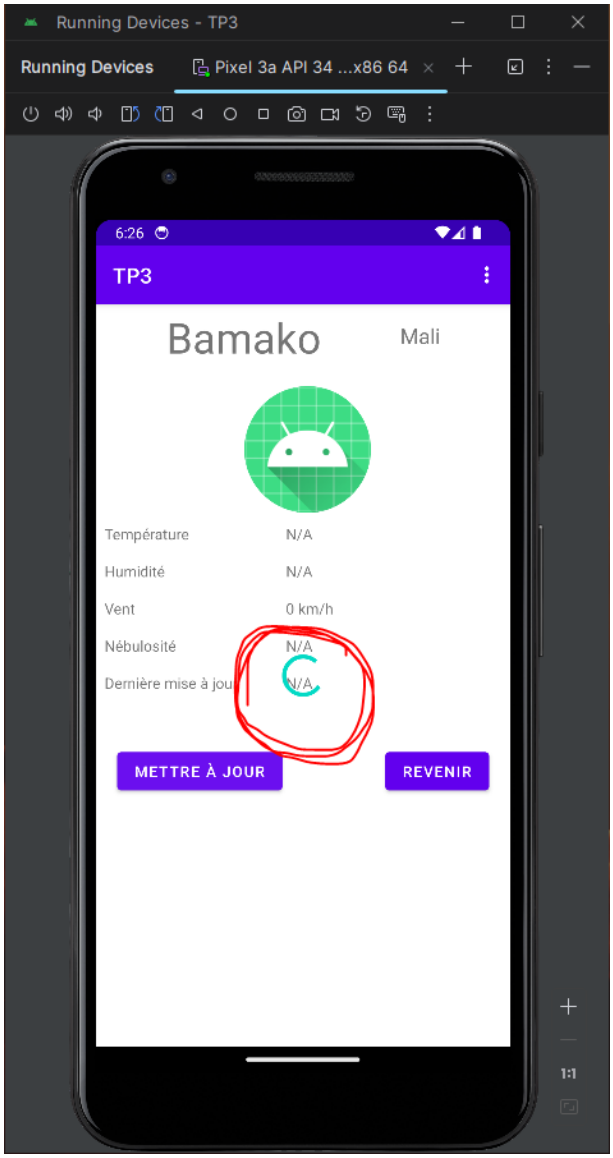

Figure 4. Gestion des états de chargement

- Traitement des erreurs : L'application gère les éventuelles erreurs de connexion ou de réponse du service web et affiche des messages appropriés.

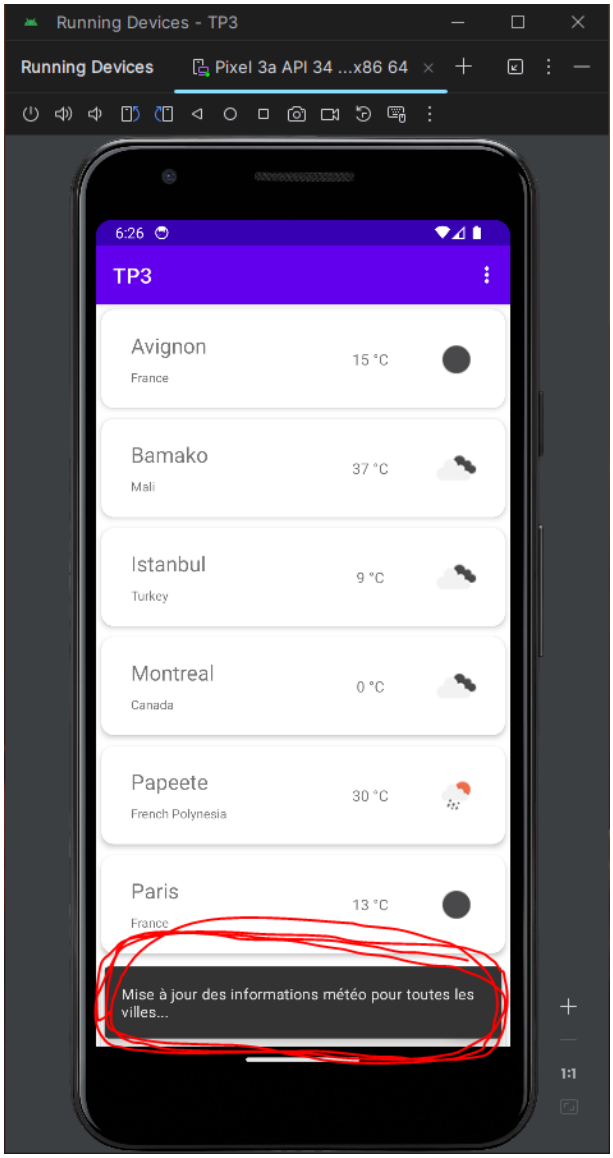

Figure 5. Traitement des erreurs

4 Problèmes rencontrés et solutions

Durant le développement de l'application Android pour le TP3, divers défis sont survenus, impactant des aspects techniques variés. Voici une analyse approfondie de ces problèmes et des solutions adoptées pour les résoudre.

4.1 Problème 1 : Problèmes de configuration avec Retrofit

Symptôme : Les premières tentatives d'appels API à l'aide de Retrofit échouaient systématiquement, renvoyant soit des erreurs de connexion, soit des réponses mal formées.

Cause : Le problème résidait dans la configuration incorrecte des dépendances Retrofit dans le fichier build.gradle, ainsi que dans l'omission de la permission d'accès à Internet dans le fichier AndroidManifest.xml.

Solution : La résolution a nécessité plusieurs étapes. D'abord, revérification du fichier build.gradle pour m'assurer que toutes les dépendances nécessaires pour Retrofit et Moshi étaient correctement intégrées et à jour. Ensuite, ajout de la permission `<uses-permission android:name="android.permission.INTERNET"/>` dans le fichier AndroidManifest.xml pour

autoriser l'application à accéder à Internet. Enfin, consultation de la documentation officielle de Retrofit pour valider ma compréhension de la configuration et de l'utilisation de la bibliothèque.

4.2 Problème 2 : Erreurs d'analyse JSON avec Moshi

Symptôme : L'application ne parvenait pas à parser correctement les réponses JSON reçues du service web, entraînant des crashes ou des données incorrectes affichées.

Cause : Ce problème était dû à une mauvaise correspondance entre les noms des champs dans les classes de données Java et les clés JSON fournies par le service web. De plus, certaines données JSON étaient parfois null et non prises en compte correctement dans le code.

Solution : Pour corriger ces erreurs, j'ai utilisé des annotations Moshi pour mapper explicitement les champs JSON aux variables de la classe Java, permettant ainsi de gérer les différences de nomenclature. J'ai également mis en place des valeurs par défaut pour les champs pouvant être null et utilisé des types optionnels pour éviter les NullPointerException.

Figure 6. Visualisation détaillée de la météo avec des valeurs nulles

4.3 Problème 3 : Gestion des erreurs de réseau

Symptôme : L'application se fermait brusquement lorsqu'il y avait des problèmes de connexion réseau ou que le service web était inaccessible.

Cause : La gestion des exceptions liées au réseau était insuffisante. L'application n'était pas préparée à gérer les scénarios où le service web était lent ou non disponible.

Solution : J'ai amélioré la robustesse de l'application en implémentant une gestion d'exception plus complète autour des appels API. J'ai utilisé le mécanisme de Callback de Retrofit pour distinguer les différents types d'erreurs (comme les erreurs de connexion et les erreurs HTTP). Des messages d'erreur appropriés sont désormais affichés à l'utilisateur via des Snackbars, au lieu de fermer l'application.

4.4 Problème 4 : Performances de l'application lors de la mise à jour des données pour toutes les villes

Symptôme : L'application devenait extrêmement lente et peu réactive lors de la mise à jour des informations météorologiques pour toutes les villes enregistrées dans la base de données.

Cause : Le processus de mise à jour était effectué de manière synchrone et bloquante, ce qui empêchait l'interface utilisateur de rester fluide et réactive pendant le traitement des données.

Solution : Pour résoudre ce problème, j'ai modifié la logique de mise à jour pour qu'elle soit asynchrone. J'ai utilisé des appels Retrofit asynchrones et j'ai veillé à ce que les opérations de base de données soient également effectuées sur des threads séparés du thread principal de l'application. Cela a permis de maintenir l'interface utilisateur fluide et réactive pendant que les données étaient mises à jour en arrière-plan.

5 Conclusion

Le TP3 a été une opportunité significative pour approfondir la connaissance du développement Android, en particulier en ce qui concerne l'interrogation de services web et l'analyse des données JSON. Bien que plusieurs défis aient été rencontrés, chacun a été résolu de manière systématique, contribuant ainsi à une meilleure compréhension du processus de développement d'applications mobiles. L'application finale répond aux exigences énoncées et fournit une base solide pour de futures extensions et améliorations.