Cahier des charges — Weather Analyzer

Barua Turjoy

1. Introduction

Le projet **Weather Analyzer** vise à créer une application en mode terminal permettant d'analyser et de visualiser des données météorologiques. L'utilisateur pourra :

- récupérer la météo d'une ville (température, humidité, conditions),
- stocker ces données dans une base de données,
- obtenir des analyses et des graphiques sur l'évolution du climat.

> **Une interface graphique (Tkinter) sera ajoutée après finalisation et stabilisation du backend.**

2. Fonctionnalités de l'application

- **Récupération des données** : Saisie du nom d'une ville pour obtenir la météo du jour.
- **Persistance** : Stockage des données dans une base (SQLite ou Supabase).
- **Statistiques** : Calcul de la moyenne, maximum, minimum des températures, nombre de jours de pluie, etc.
- **Visualisation** : Graphiques (courbes de tendance, histogrammes) via `matplotlib`.
- **Interface terminal** : Menu interactif et navigation claire.
- **Gestion des erreurs** : Gestion des cas de ville inconnue, absence de connexion, base vide, etc.

3. Conception de l'application

- **Langage** : Python.
- **Interface principle (Phase 1)** : Console (terminal).
- **Source des données** : API publique OpenWeatherMap.
- **Stockage** : SQLite (local) ou Supabase (cloud).
- **Analyse** : `pandas`.
- **Visualisation** : `matplotlib`.
- **Évolution prévue (Phase 2)** : Intégration d'une interface graphique Tkinter après achèvement du backend pour reproduire toutes les fonctionnalités (saisie de ville, analyses, graphiques) et tester l'ergonomie.

4. Contraintes techniques

Obligatoire : Python

Bibliothèques :

- `requests` : Appels API
- `sqlite3` ou `supabase-py` : Base de données
- `pandas` : Analyse
- `matplotlib` : Visualisation

Exécution : Applicatif utilisable en mode terminal.
Durabilité des données : Sauvegarde persistante pour analyses historiques.

Extensions possibles :

- Interface graphique (Tkinter) — Phase 2
- Prédictions basiques (Machine Learning)
- Comparaison multi-villes

5. Plan de développement

Phase 1 — Backend & Console

1. Connexion à l'API météo et récupération des données.
2. Création et gestion de la base de données (SQLite ou Supabase).
3. Mise en place du menu terminal.
4. Développement des fonctions d'analyse (moyennes, max/min, jours de pluie).
5. Génération et affichage de graphiques en terminal ou via fenêtres matplotlib.
6. Gestion des erreurs et messages clairs (logging, validations, cas hors-ligne).

Phase 2 — Interface graphique (Tkinter)

- Ajout d'une interface graphique après achèvement et stabilisation du backend.
- Intégration des mêmes fonctionnalités (saisie ville, analyses, graphiques).
- Tests d'ergonomie et robustesse.

Optionnel (post-Phase 2)

- Prédictions simples (régressions/ARIMA).
- Comparaison de plusieurs villes avec vues combinées.
- Export des rapports (CSV/PNG/PDF).