Exercice — Collecte, stockage et exploitation automatisées (Open-Meteo)

Objectif : Mettre en place un flux automatisé qui interroge l'API Open-Meteo pour plusieurs villes, stocke l'historique, calcule des indicateurs et produit des alertes.

API

Point d'accès :

```
https://api.open-meteo.com/v1/forecast?latitude=<LAT>&longitude=<LON>&current_weather=true
```

Exemples de villes

- Paris — 48.8566, 2.3522
- Lille — 50.6292, 3.0573
- Lyon — 45.7640, 4.8357
- Marseille — 43.2965, 5.3698
- Casablanca — 33.5731, -7.5898

Champs utiles

- `current_weather.time` (ISO)
- `current_weather.temperature` (°C)
- `current_weather.windspeed` (km/h)
- `current_weather.winddirection` (°)
- `current_weather.weathercode` (WMO)
- Contexte : `latitude`, `longitude`, `timezone`

> L'implémentation est agnostique : OS et langage au choix de l'étudiant. L'important est l'industrialisation du flux (collecte, stockage, exploitation, planification).

Partie A — Collecte

1. Lire une liste de 3 à 5 villes (nom, latitude, longitude) depuis un fichier de configuration ou une structure interne.
2. Pour chaque ville, appeler l'API et extraire au minimum les champs listés ci-dessus.
3. Gérer les erreurs : en cas d'échec HTTP ou JSON invalide, effectuer jusqu'à deux nouvelles tentatives avec un court délai. Journaliser l'échec si persistant.
4. Tracer chaque exécution avec horodatage, villes traitées et durée.

Partie B — Stockage

Choisir l'un des formats persistants (append-only) :

- CSV avec en-tête : `city,ts,latitude,longitude,temperature_c,windspeed_kmh,wind_dir_deg,weathercode,timezone`
- JSON Lines (un objet JSON par ligne) avec les mêmes champs et, si souhaité, la réponse brute pour traçabilité.

- SQLite avec table : `measurements(city, ts, latitude, longitude, temperature_c, windspeed_kmh, wind_dir_deg, weathercode, timezone, raw_json)`

Chaque exécution ajoute des enregistrements sans écraser l'historique.

Partie C — Exploitation

1. Sur les dernières 24 h et par ville, calculer : température minimale, maximale et moyenne ; vitesse de vent maximale ; nombre de mesures.
2. Générer un rapport lisible (texte ou Markdown) avec un tableau récapitulatif et un bloc « anomalies » listant les exécutions en échec.
3. Mise en alerte : si `temperature_c > 30` ou `windspeed_kmh > 60`, consigner une alerte dans un fichier dédié ou dans un canal choisi.

Partie D — Automatisation

- Planifier la collecte à un intervalle régulier (par exemple toutes les heures).
- Planifier la génération du rapport une fois par jour (par exemple 07:00).
- Centraliser la configuration (liste des villes, seuils d'alerte, chemins de stockage et de logs).

Contraintes et attendus

- Append uniquement : ne jamais écraser les données existantes.
- Journalisation horodatée des succès/échecs.
- Possibilité de régénérer un rapport sans rappeler l'API (travail sur l'historique).