# Documentation Technique de l'Application de Monitoring Météorologique

## Table des matières

**Ce document détaille l'architecture, les fonctionnalités et le fonctionnement d'une application de monitoring en temps réel pour un système de prévision météorologique. L'objectif principal est de surveiller les performances du système, d'assurer une gestion efficace des requêtes des utilisateurs et de fournir une interface intuitive pour l'interaction avec le système. L'application utilise FastAPI pour le backend, Streamlit pour le frontend et Prometheus pour la collecte et l'analyse des métriques.**

# Architecture de l'application

**L'application est conçue avec une architecture à trois niveaux :**

- **Backend (FastAPI) : Gère les requêtes des utilisateurs, récupère les données météorologiques de l'API Open-Meteo, les traite et les stocke dans une base de données PostgreSQL.**
- **Frontend (Streamlit): Fournit une interface utilisateur interactive pour soumettre des requêtes, visualiser les prévisions météorologiques et afficher les métriques de performance.**
- **Monitoring (Prometheus et Grafana) : Collecte et analyse les métriques de performance du backend, telles que le nombre de requêtes, le temps de réponse et les erreurs, et les affiche dans des tableaux de bord Grafana.**

# Fonctionnalités :

- **Traitement des requêtes des utilisateurs : Analyse les commandes des utilisateurs (texte ou voix) pour extraire les informations de localisation et de durée des prévisions.**
- **Récupération des données météorologiques : Récupère les prévisions météorologiques de l'API Open-Meteo en fonction de la localisation et de la durée spécifiées.**
- **Stockage des données : Enregistre les requêtes des utilisateurs, les prévisions et les commentaires dans une base de données PostgreSQL.**
- **Visualisation des données : Affiche les prévisions météorologiques sous forme de graphiques et de tableaux interactifs à l'aide de Plotly.**
- **Monitoring des performances : Surveille le nombre de requêtes, le temps de réponse et les erreurs à l'aide de Prometheus et les affiche dans Grafana.**
- **Collecte des commentaires des utilisateurs : Permet aux utilisateurs de soumettre des commentaires sur l'application.**

# Modules Principaux

## Backend :

- **main.py : Initialise l'API FastAPI, définit les routes et les middlewares.**
- **process_command.py : Analyse les commandes des utilisateurs.**
- **weather.py : Interagit avec l'API Open-Meteo.**
- **database.py : Gère la base de données PostgreSQL.**
- **monitoring.py : Définit les métriques Prometheus.**
- **feedback.py : Gère les commentaires des utilisateurs.**

## Frontend :

- **main.py : Page principale de l'interface utilisateur.**
- **tabs/requests.py : Permet aux utilisateurs de soumettre des requêtes.**
- **tabs/results.py : Affiche les prévisions météorologiques.**
- **tabs/monitoring.py : Affiche les métriques de performance.**
- **tabs/feedback.py : Gère les commentaires des utilisateurs.**

# Technologies Utilisées

## FastAPI

- **Framework web Python moderne : FastAPI est un framework web moderne, haute performance pour la construction d'API avec Python 3.6+ basé sur les standards Python pour la définition des types (type hints). Il est conçu pour être facile à utiliser, rapide à coder et offrir des performances élevées.**
- **Performance : FastAPI est l'un des frameworks web Python les plus rapides, offrant des performances comparables à celles de NodeJS et Go. Cela est dû à l'utilisation de Starlette pour la partie ASGI et de Pydantic pour la validation des données.**
- **Validation des données : FastAPI utilise Pydantic pour la validation des données, ce qui permet de définir des schémas de données clairs et de garantir la validité des données reçues et renvoyées par l'API.**
- **Documentation automatique : FastAPI génère automatiquement une documentation interactive de l'API à l'aide de Swagger UI et Redoc, ce qui facilite la compréhension et l'utilisation de l'API.**

## Streamlit

- **Création d'applications web interactives : Streamlit est une bibliothèque Python open-source qui permet de créer des applications web interactives pour l'apprentissage automatique et la science des données.**
- **Facilité d'utilisation : Streamlit offre une syntaxe simple et intuitive, ce qui permet de créer rapidement des applications web sans avoir besoin de connaissances approfondies en développement web.**
- **Composants interactifs : Streamlit fournit une variété de composants interactifs, tels que des sliders, des boutons, des graphiques et des tableaux de données, qui permettent de créer des applications web dynamiques et engageantes.**
- **Intégration avec d'autres bibliothèques : Streamlit s'intègre facilement avec d'autres bibliothèques Python populaires pour la science des données, telles que Pandas, NumPy, Scikit-learn et Plotly.**

## Prometheus

- **Système de monitoring et d'alerte : Prometheus est un système de monitoring et d'alerte open-source qui collecte des métriques à partir de cibles configurées en récupérant des données via HTTP.**
- **Modèle de données multidimensionnel : Prometheus stocke les métriques sous forme de séries temporelles avec des identifiants uniques et des paires clé-valeur pour les étiquettes, permettant une analyse flexible et des requêtes puissantes.**
- **Langage de requête PromQL : Prometheus fournit un langage de requête puissant, PromQL, pour interroger et agréger les données de séries temporelles.**
- **Alertes : Prometheus prend en charge la définition d'alertes basées sur des règles pour notifier les utilisateurs en cas de conditions spécifiques, telles que des seuils dépassés ou des erreurs.**

## PostgreSQL

- **Système de gestion de base de données relationnelle : PostgreSQL est un système de gestion de base de données relationnelle (SGBDR) open-source puissant et robuste, connu pour sa fiabilité, sa conformité aux normes SQL et ses fonctionnalités avancées.**
- **Fiabilité et intégrité des données : PostgreSQL met l'accent sur l'intégrité des données et offre des fonctionnalités telles que les transactions ACID, les contraintes d'intégrité référentielle et la réplication pour garantir la cohérence et la fiabilité des données.**
- **Extensibilité : PostgreSQL est hautement extensible et prend en charge un large éventail de types de données, d'index, de fonctions et de langages de programmation procéduraux, ce qui permet de l'adapter à diverses applications.**
- **Communauté active : PostgreSQL bénéficie d'une communauté open-source active et d'un développement continu, garantissant un support, des mises à jour régulières et une amélioration constante du système.**

## Open-Météo API

- **API météorologique gratuite : Open-Meteo API est une API gratuite et open-source qui fournit des données météorologiques mondiales, y compris des prévisions horaires et quotidiennes pour diverses variables météorologiques.**
- **Couverture mondiale : L'API couvre l'ensemble du globe, offrant des données pour un large éventail d'emplacements.**
- **Données variées : Open-Meteo API fournit des données pour diverses variables météorologiques, telles que la température, les précipitations, la couverture nuageuse, la vitesse du vent et plus encore.**
- **Facilité d'utilisation : L'API est facile à utiliser avec une documentation claire et des exemples de code.**

**Pydantic**

- **Validation des données et gestion des schémas : Pydantic est une bibliothèque Python qui utilise des annotations de type Python pour la validation des données et la gestion des schémas.**
- **Définition de schémas : Pydantic permet de définir des schémas de données clairs et concis à l'aide de classes Python et d'annotations de type.**
- **Validation automatique : Pydantic valide automatiquement les données par rapport au schéma défini, ce qui permet de garantir la validité des données reçues et renvoyées par l'API.**
- **Sérialisation et désérialisation : Pydantic prend en charge la sérialisation et la désérialisation des données entre différents formats, tels que JSON, YAML et les dictionnaires Python.**

## Requests

- **Bibliothèque HTTP pour Python : Requests est une bibliothèque HTTP élégante et simple pour Python, qui permet d'envoyer des requêtes HTTP de manière simple et intuitive.**
- **Fonctionnalités : Requests prend en charge diverses fonctionnalités HTTP, telles que l'authentification, les cookies, les sessions, les timeouts et le téléchargement de fichiers.**
- **Facilité d'utilisation : Requests est conçu pour être facile à utiliser, avec une API simple et intuitive.**

## Requests-Cache

- **Mise en cache des requêtes HTTP : Requests-Cache est une bibliothèque Python qui permet de mettre en cache les réponses des requêtes HTTP, ce qui permet d'accélérer les applications et de réduire la charge sur les serveurs distants.**
- **Configuration flexible : Requests-Cache offre une configuration flexible pour contrôler le comportement de la mise en cache, tels que la durée de vie du cache, les méthodes HTTP à mettre en cache et les en-têtes à inclure dans la clé de cache.**
- **Intégration avec Requests : Requests-Cache s'intègre facilement avec la bibliothèque Requests, ce qui permet de l'utiliser avec un minimum de modifications de code.**

## SpaCy

- **Bibliothèque de traitement du langage nature l: SpaCy est une bibliothèque Python open-source pour le traitement avancé du langage naturel (NLP).**
- **Fonctionnalités : SpaCy offre une variété de fonctionnalités NLP, telles que la tokenisation, l'analyse morpho-syntaxique, la reconnaissance d'entités nommées et l'analyse des sentiments.**
- **Performance : SpaCy est conçu pour être rapide et efficace, ce qui le rend adapté aux applications NLP à grande échelle.**
- **Modèles pré-entraînés : SpaCy fournit des modèles pré-entraînés pour diverses langues, ce qui permet de démarrer rapidement avec des tâches NLP courantes.**

## Plotly

- **Bibliothèque de visualisation interactive : Plotly est une bibliothèque Python open-source pour créer des graphiques interactifs.**
- **Graphiques variés : Plotly prend en charge une variété de types de graphiques, tels que les graphiques linéaires, les graphiques à barres, les camemberts, les nuages de points et les cartes.**
- **Interactivité : Les graphiques Plotly sont interactifs, ce qui permet aux utilisateurs de zoomer, de dézoomer, de survoler des points de données et d'explorer les données de manière plus approfondie.**
- **Intégration avec d'autres bibliothèques : Plotly s'intègre facilement avec d'autres bibliothèques Python pour la science des données, telles que Pandas et Streamlit.**

## Uvicorn

- **Serveur ASGI pour Python : Uvicorn est un serveur ASGI (Asynchronous Server Gateway Interface) rapide et léger pour Python, qui permet d'exécuter des applications web asynchrones.**

- **Performance : Uvicorn est l'un des serveurs ASGI les plus rapides, offrant des performances comparables à celles de NodeJS et Go.**
- **Compatibilité avec asyncio : Uvicorn est conçu pour fonctionner avec asyncio, la bibliothèque de programmation asynchrone de Python.**
- **Facilité d'utilisation : Uvicorn est facile à configurer et à utiliser, ce qui en fait un choix populaire pour le développement et le déploiement d'applications web Python asynchrones.**

# APIs

## Open-Meteo API

- **Fonctionnalités :**
    - **Fournit des données météorologiques historiques et prévisionnelles pour n'importe quel endroit dans le monde.**
    - **Offre une variété de variables météorologiques, y compris la température, les précipitations, la couverture nuageuse, la vitesse du vent, etc.**
    - **Permet de spécifier la résolution temporelle des données (horaire, quotidienne).**
    - **Propose différentes options de format de sortie (JSON, CSV).**
- **Utilisation dans l'application :**
    - **Le module weather.py utilise l'API Open-Meteo pour récupérer les prévisions météorologiques en fonction de la localisation et de la durée spécifiées par l'utilisateur.**
    - **Les requêtes à l'API sont effectuées à l'aide de la bibliothèque requests.**
    - **Les réponses de l'API sont mises en cache à l'aide de requests-cache pour optimiser les performances.**
    - **Les données météorologiques récupérées sont traitées et formatées pour l'affichage dans l'interface utilisateur Streamlit et le stockage dans la base de données PostgreSQL.**

## Azure Speech to Text

- **Fonctionnalités :**
    - **Service de reconnaissance vocale basé sur le cloud qui convertit la parole en texte.**
    - **Prend en charge plusieurs langues et dialectes.**
    - **Offre différentes options de personnalisation, telles que la spécification du vocabulaire et du domaine.**
    - **Peut être utilisé avec différents types d'entrée audio, tels que des fichiers audios et des flux audios en direct.**

- **Utilisation dans l'application :**
    - **L'application utilise Azure Speech to Text pour permettre aux utilisateurs de soumettre des requêtes vocales.**
    - **Le module process_command.py appelle la fonction azure_speech_to_text pour convertir les fichiers audios ou les flux audios en direct en texte.**
    - **Le texte transcrit est ensuite analysé par SpaCy pour extraire les informations de localisation et de durée des prévisions.**

# Utilisation d'Azure dans l'application

**Outre Azure Speech to Text, l'application pourrait potentiellement utiliser d'autres services Azure, tels que :**

- **Azure Cognitive Services : Pour des fonctionnalités d'IA supplémentaires, telles que l'analyse des sentiments, la traduction et la reconnaissance d'images.**
- **Azure Machine Learning : Pour entraîner et déployer des modèles d'IA pour améliorer les prévisions météorologiques ou personnaliser l'expérience utilisateur.**
- **Azure App Service : Pour héberger l'application web Streamlit et l'API FastAPI dans le cloud.**
- **Azure Database for PostgreSQL : Pour héberger la base de données PostgreSQL dans le cloud.**

**L'utilisation de ces services Azure pourrait améliorer les fonctionnalités, les performances et la scalabilité de l'application.**

# Sécurité

- **Validation des entrées : L'application valide les entrées utilisateur pour prévenir les injections SQL et autres failles de sécurité.**
- **Variables d'environnement : Les informations sensibles, telles que les clés d'API et les mots de passe, sont stockées dans des variables d'environnement pour des raisons de sécurité.**
- **Journalisation : L'application enregistre les événements importants et les erreurs pour faciliter le débogage et la surveillance de la sécurité.**

**Cette application de monitoring fournit une solution complète pour la gestion et la surveillance d'un système de prévision météorologique. Elle combine une interface utilisateur conviviale, des fonctionnalités de traitement des requêtes robustes, une intégration avec**

# Glossaire

- **API (Application Programming Interface) :** Interface de programmation d'application. Un ensemble de définitions et de protocoles permettant à des applications de communiquer entre elles.
- **ASGI (Asynchronous Server Gateway Interface) :** Standard Python pour la construction d'applications web asynchrones.
- **Backend :** Partie d'une application qui s'exécute sur un serveur et gère la logique métier, le stockage des données et la sécurité.
- **Docker :** Plateforme de conteneurisation permettant d'empaqueter une application et ses dépendances dans une unité standardisée pour le développement, le déploiement et l'exécution.
- **Endpoint :** Point d'accès à une API, généralement une URL spécifique, qui permet d'effectuer une action ou de récupérer des données.
- **Frontend :** Partie d'une application qui interagit directement avec l'utilisateur, généralement une interface graphique.
- **Grafana :** Plateforme open-source pour la visualisation et l'analyse de données de séries temporelles, souvent utilisée pour le monitoring.
- **HTTP (Hypertext Transfer Protocol) :** Protocole de communication utilisé pour transférer des informations sur le web.
- **JSON (JavaScript Object Notation) :** Format de données textuel léger et lisible par l'homme, couramment utilisé pour l'échange de données sur le web.
- **Kubernetes :** Plateforme open-source pour automatiser le déploiement, la mise à l'échelle et la gestion d'applications conteneurisées.
- **Middleware :** Logiciel qui se situe entre un système d'exploitation et les applications qui s'exécutent sur celui-ci.
- **Monitoring :** Surveillance continue des performances, de la disponibilité et de la santé d'un système.
- **NLP (Natural Language Processing) :** Traitement du langage naturel. Domaine de l'intelligence artificielle qui vise à permettre aux ordinateurs de comprendre, d'interpréter et de générer du langage humain.
- **Pydantic :** Bibliothèque Python pour la validation des données et la gestion des schémas.
- **Prometheus :** Système de monitoring et d'alerte open-source.
- **REST (Representational State Transfer) :** Style d'architecture logicielle pour les systèmes hypermédias distribués, comme le web.

**Structured Query Language) :** Langage de programmation standardisé pour la gestion des bases de données relationnelles.

- **Streamlit :** Bibliothèque Python open-source pour créer des applications web interactives pour l'apprentissage automatique et la science des données.
- **Uvicorn :** Serveur ASGI pour Python.

# Bibliothèque de liens

- **FastAPI :** https://fastapi.tiangolo.com/
- **Streamlit:** https://docs.streamlit.io/
- **Prometheus Client :** https://github.com/prometheus/client_python
- **Pandas:** https://pandas.pydata.org/docs/
- **Requests:** https://docs.python-requests.org/
- **Requests-Cache :** https://requests-cache.readthedocs.io/
- **SpaCy :** https://spacy.io/
- **Plotly:** https://plotly.com/python/
- **Uvicorn:** https://www.uvicorn.org/
- **PostgreSQL :** https://www.postgresql.org/docs/
- **Open-Meteo:** https://open-meteo.com/
- **Grafana :** https://grafana.com/
- **Docker:** https://www.docker.com/
- **Kubernetes :** https://kubernetes.io/
- **Azure Speech to Text:** https://azure.microsoft.com/en-us/services/cognitive-services/speech-to-text/