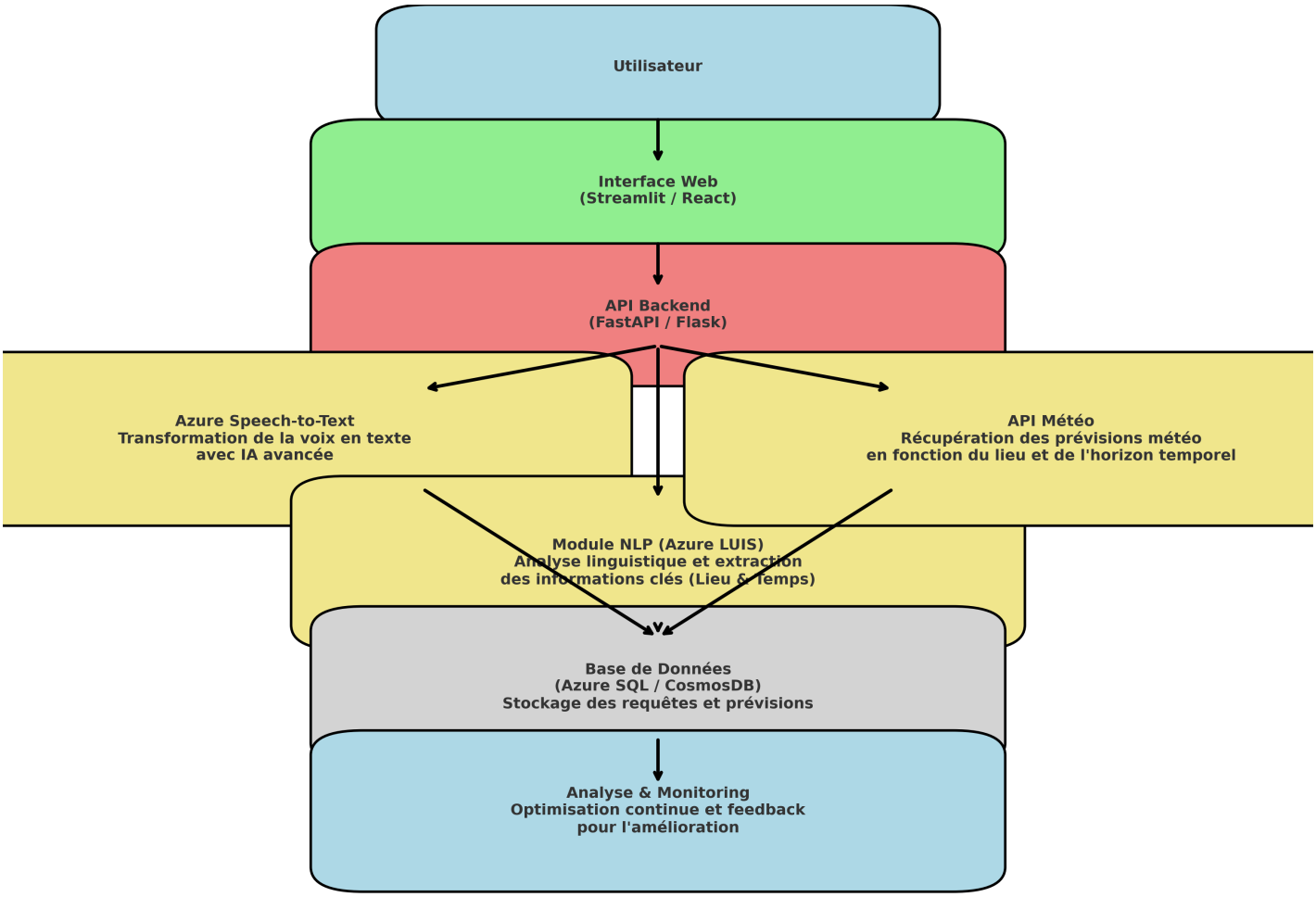
Architecture fonctionnelle du projet :

1. Schéma d'architecture fonctionnelle



2. Décomposition :

L'application repose sur plusieurs couches interconnectées :

- **Interface utilisateur (UI)**
    - Accueil et champ de recherche vocale.
    - Affichage des prévisions météorologiques sous forme de texte et de graphiques.
    - Affichage d'une carte interactive pour sélectionner un lieu.
- **Traitement de la commande vocale**
    - Azure Speech-to-Text : Transcription du signal vocal en texte.
    - Module NLP (Azure LUIS ou autre) : Extraction des entités (lieu et horizon temporel).
    - Enrichissement des requêtes (validation et correction éventuelle du texte).
- **Moteur de recherche météo**
    - Appel à une API externe (OpenWeatherMap, WeatherAPI, MeteoFrance, etc.).
    - Paramétrage des requêtes en fonction des informations extraites.
    - Gestion des erreurs en cas d'indisponibilité ou d'erreur de réponse.
- **Base de données et stockage**
    - Stockage des requêtes et résultats en **Azure SQL Database** ou **CosmosDB**.
    - Historisation des prévisions pour analyse ultérieure.
    - Enregistrement des logs pour monitoring et amélioration de la reconnaissance vocale.
- **API Backend**
    - Serveur FastAPI ou Flask gérant l'ensemble des requêtes.
    - Sécurisation des appels avec authentification et contrôle des entrées.
    - Exposition des données sous forme d'API REST pour le frontend.
- **Analyse et monitoring**
    - Comparaison des prévisions avec les valeurs réelles (feedback loop).
    - Détection des erreurs et ajustements automatiques.
    - Tableaux de bord analytiques pour améliorer la précision des prédictions.

3. Flux de traitement

1. L'utilisateur **parle** dans le micro.
2. Azure **convertit l'audio en texte** (Speech-to-Text).
3. Le **module NLP** analyse la phrase et **extrait le lieu et l'horizon temporel**.
4. Le **backend formate la requête** et interroge une **API météo**.
5. La **réponse est traitée et stockée** dans la base Azure.
6. L'interface **affiche les prévisions** sous forme de texte, graphiques et carte interactive.
7. Les résultats sont **monitorés** et un **feedback loop** ajuste la reconnaissance en cas d'erreurs.

4. Technologies utilisées

- **Azure Speech-to-Text**: Reconnaissance vocale.
- **Azure LUIS (Language Understanding)** : Analyse de texte et extraction des intentions.
- **API météo** : OpenWeatherMap, WeatherAPI ou MeteoFrance.
- **Base de données Azure** : Stockage des requêtes et logs.
- **FastAPI/Flask** : Backend pour orchestrer les services.
- **Streamlit ou React** : Interface utilisateur.