**Atelier 2 Raphael L3 IA**

Rapport **: Conception d’un système Smart Home basé sur des données météorologiques**

**1. Introduction au scénario choisi et aux objectifs de l'architecture**

L'objectif de cette architecture est de développer un système Smart Home intelligent capable d'ajuster automatiquement les systèmes de chauffage, climatisation, rideaux en fonction de deux principales sources d'information :

Les données météorologiques en temps réel, obtenues à partir de **l’API Open-Meteo**, pour surveiller la température et la luminosité (jour/nuit).

**Objectifs principaux :**

**Optimisation énergétique** : Minimiser la consommation d’énergie en activant uniquement les dispositifs nécessaires.

**Confort** : Améliorer l'expérience utilisateur grâce à l’automatisation.

**Efficacité** : Exploiter les données en temps réel pour une gestion dynamique des équipements.

**2. Description des technologies et services utilisés**

**a. Collecte des données météorologiques**

API Open-Meteo : Fournit des données météo en temps réel, telles que la température et l’état jour/nuit. Ces données alimentent le système de décision.

**b. Stockage des données**

Google Drive : Utilisé pour sauvegarder les données collectées sous la forme d’un fichier CSV. Google Drive offre une solution accessible et pratique pour gérer les données de manière persistante.

**c. Visualisation et analyses**

Google Colab : Sert d’environnement de développement pour exécuter le système, gérer les données et produire des graphiques.

Matplotlib/Seaborn : Génèrent des visualisations telles que l’évolution des températures, l’état des rideaux, ou l’utilisation du chauffage et de la climatisation.

**d. Automatisation des décisions**

Le système repose sur des algorithmes Python pour prendre des décisions basées sur des seuils prédéfinis pour la température (chauffage/climatisation) et la lumière (jour/nuit pour les rideaux).

**e. Interface utilisateur**

**ipywidgets** : Fournit une interface interactive pour l'authentification et l'affichage des données.

**3. Diagramme de l’architecture et explication de chaque composant**

Le diagramme illustre le flux des données entre les différents composants :

**Collecte des données météo :**

Les informations de l'API Open-Meteo sont récupérées à intervalles réguliers.

**Stockage des données :**

Les données collectées sont enregistrées dans un fichier CSV sur Google Drive.

**Traitement des données :**

Les seuils de décision sont appliqués (ex. température > 26 °C active la climatisation).

Le statut des équipements (chauffage, rideaux) est mis à jour.

**Visualisation :**

Les données stockées sont transformées en graphiques pour mieux analyser les tendances.

(Si besoin, intégrer un diagramme détaillé ici à partir de l'image que tu m’as partagée.)

**4. Avantages et défis de la solution**

**Avantage :**

**Scalabilité** : Le système peut facilement intégrer de nouveaux capteurs ou fonctionnalités, comme des capteurs de qualité de l’air.

**Efficacité énergétique** : Réduction de la consommation grâce à des décisions basées sur les données.

**Coût réduit** : L’utilisation d’un environnement local (Google Colab et Drive) évite des coûts liés à des services cloud avancés.

**Défis :**

**Sécurité :** Le stockage des données sur Google Drive pourrait nécessiter des mesures de chiffrement pour éviter des accès non autorisés.

**Temps réel :** Bien que les mises à jour soient périodiques (toutes les 5 secondes), un système entièrement temps réel pourrait nécessiter des services cloud plus robustes.

**Fiabilité :** L’API Open-Meteo est dépendante d’une connexion Internet, ce qui pourrait poser problème en cas de coupure.

**5. Raisonnement pour le choix des technologies**

**Open-Meteo :** API gratuite et simple pour obtenir des données météo en temps réel.

**Google Drive :** Permet de sauvegarder facilement les données sans nécessiter de déploiement complexe sur un cloud public.

**Google Colab :** Fournit un environnement accessible pour développer et exécuter le système sans installation locale.

**Python :** Langage adapté pour la manipulation de données, le développement de systèmes d’automatisation et la création de visualisations.

**6. Démystification des concepts complexes**

**Collecte des données en temps réel :** L’API Open-Meteo est interrogée périodiquement grâce à Python. Chaque réponse est une "photo instantanée" des conditions météorologiques actuelles, utilisée pour prendre des décisions intelligentes.

**Traitement des données :** Les décisions sont basées sur des seuils définis (température, luminosité). Par exemple, si la température dépasse 26 °C, la climatisation est activée pour maintenir le confort.

**Visualisation des données :** Les graphiques générés par Matplotlib et Seaborn permettent d’analyser les tendances (comme les pics de température) et d’identifier les améliorations possibles du système.

**Conclusion**

Ce projet démontre la puissance d’une architecture simple mais efficace pour une maison intelligente. Les technologies choisies permettent de combiner des fonctionnalités pratiques, des coûts réduits et une bonne scalabilité. Cependant, des améliorations, telles qu’une meilleure gestion de la sécurité et une véritable solution temps réel basée sur le cloud, pourraient renforcer encore le système.