Étape 1 : Planification et Conception

- **Définir les Exigences**
  - Combien de pièces voulez-vous surveiller ?
  - Quelle est la distance entre les Raspberry Pis et le Pi central ?
  - Avez-vous besoin de fonctionnalités spécifiques pour l'application web (comme des alertes, des rapports historiques, etc.) ?
- **Choisir les Composants**
  - Sélectionner les modèles de Raspberry Pi pour le central et les clients.
  - Choisir les capteurs de température (comme le DHT22 pour la température et l'humidité).
- **Concevoir l'Architecture du Système**
  - Décider comment les Pis communiqueront entre eux (MQTT, HTTP, etc.).
  - Établir un schéma de base de données pour stocker les lectures de température.

Étape 2 : Configuration de l'Environnement

- **Configurer les Raspberry Pis**
  - Installer le système d'exploitation (Raspberry Pi OS recommandé).
  - Configurer le réseau pour qu'ils puissent communiquer entre eux.
  - Mettre en place le Raspberry Pi central avec un serveur MQTT si vous utilisez MQTT, ou préparer un endpoint API si vous utilisez HTTP.
- **Configurer l'Environnement de Développement**
  - Installer les langages de programmation nécessaires (comme Python).
  - Installer les bibliothèques pour interagir avec les capteurs de température et les protocoles de communication (comme paho-mqtt pour MQTT).

Étape 3 : Développement

- **Programmer les Clients (Raspberry Pis dans chaque pièce)**
  - Écrire le script pour lire les données des capteurs de température.
  - Envoyer les données au Pi central via le protocole choisi.
- **Programmer le Serveur Central (Raspberry Pi Central)**
  - Mettre en place la réception des données de température.
  - Stocker les données reçues dans la base de données.
- **Développer l'Application Web**
  - Créer une interface utilisateur pour afficher les données.
  - Interfacer l'application avec la base de données pour récupérer et afficher les données.

Étape 4 : Tests et Déploiement

- **Tester chaque Composant**
  - Vérifier que les capteurs fonctionnent correctement.
  - S'assurer que les données sont correctement envoyées au Pi central.
  - Tester l'application web localement.
- **Tester le Système dans son Ensemble**
  - Vérifier la communication entre tous les appareils.
  - S'assurer que l'application web affiche correctement les données en temps réel.
- **Déploiement**
  - Installer les Raspberry Pis dans les pièces.
  - Lancer l'application web sur le Pi central pour qu'elle soit toujours accessible.

Étape 5 : Maintenance et Évolution

- **Surveiller le système** pour s'assurer qu'il fonctionne sans interruption.
- **Mettre à jour le logiciel** au besoin.
- **Ajouter des fonctionnalités** ou des améliorations au fil du temps.

Étape 1 : Planification et Conception

Définir les exigences.

a. Nombre de Pièces à Surveiller: 5

- **Capteurs**: Vous aurez besoin de 5 ensembles de capteurs de température et d'humidité. Le DHT22 est un choix populaire car il fournit à la fois des lectures de température et d'humidité avec une bonne précision.
- **Raspberry Pis Clients**: Un Raspberry Pi par pièce, de préférence un modèle moins coûteux et économe en énergie comme le Raspberry Pi Zero W, serait adéquat pour cette tâche.

b. Connexion Wi-Fi via le Routeur de la Maison

- Puisque vous avez une bonne connexion Wi-Fi dans toutes les pièces, le Wi-Fi semble être le moyen de communication le plus approprié. Assurez-vous que chaque Raspberry Pi a une carte Wi-Fi ou est capable de se connecter au Wi-Fi (le Raspberry Pi Zero W a le Wi-Fi intégré).
- **MQTT** pourrait être un bon choix pour la communication entre les Raspberry Pis clients et le Raspberry Pi central, car il est léger et conçu pour les réseaux instables ou de faible bande passante.

c. Fonctionnalités de l'Application Web

- **Affichage en Temps Réel**: Vous aurez besoin d'une interface frontend dynamique. Des technologies comme WebSocket peuvent être utilisées pour mettre à jour l'interface utilisateur en temps réel.
- **Alertes pour Taux d'Humidité / Température**: Implémenter un système de notification qui vérifie continuellement les lectures des capteurs et envoie une alerte si les valeurs dépassent les seuils préconfigurés.
- **Graphiques Historiques**: Stocker les lectures de température et d'humidité dans une base de données. Utiliser une bibliothèque graphique côté client (comme Chart.js) pour afficher l'historique des températures et des taux d'humidité.
- **Pas d'Accès Sécurisé Requis**: Simplifie le développement de l'application web, mais assurez-vous que les données ne sont pas sensibles et que l'absence de sécurisation est acceptable.

Choisir les composant

1. Raspberry Pi

a. Raspberry Pi Central (Serveur)

- **Modèle recommandé**: Raspberry Pi 4. Ce modèle offre des performances supérieures, plus de ports USB pour connecter des périphériques, et une meilleure gestion réseau, ce qui est essentiel pour gérer les communications avec plusieurs clients et héberger l'application web.

b. Raspberry Pis Clients (dans chaque pièce)

- **Modèle recommandé**: Raspberry Pi Zero W. C'est un choix économique et économe en énergie, parfait pour des tâches simples comme la lecture de capteurs et l'envoi de données.
-

2. Capteurs

a. Température et Humidité

- **DHT22** ou **BME280**:
- **DHT22** est un choix populaire pour obtenir des lectures de température et d'humidité.
- **BME280** est un peu plus cher mais offre une meilleure précision et inclut également un capteur de pression atmosphérique.

b. Qualité de l'Air

Pour mesurer la qualité de l'air, vous pouvez considérer des capteurs qui mesurent des composants spécifiques comme les Composés Organiques Volatils (COV), le dioxyde de carbone (CO2), ou les particules fines (PM2.5 et PM10). Voici quelques options :

- **CCS811**: Mesure les COV et le CO2. Relativement simple à utiliser avec une bonne intégration dans les systèmes à base de Raspberry Pi.
- **BME680**: Fournit des mesures de température, d'humidité, de pression atmosphérique, et de qualité de l'air (COV). C'est un choix tout-en-un si vous voulez minimiser le nombre de capteurs différents.
- **PMS5003** ou **SDS011**: Spécialisés dans la mesure des particules fines (PM2.5 et PM10).
-

3. Communication

- **Cartes Wi-Fi** (si nécessaire) : Assurez-vous que chaque Raspberry Pi client a une connectivité Wi-Fi. Le Raspberry Pi Zero W a déjà le Wi-Fi intégré.
-

4. Câbles et Alimentation

- **Câbles** : Câbles GPIO pour connecter les capteurs aux Raspberry Pis.
- **Alimentation** : Des adaptateurs secteur pour chaque Raspberry Pi, avec une puissance suffisante pour les modèles choisis (par exemple, 2.5A pour le Raspberry Pi 3, 3A pour le Raspberry Pi 4).

5. Boîtiers et Montage

- **Boîtiers** : Pour protéger chaque Raspberry Pi et ses capteurs. Choisissez des boîtiers qui permettent une bonne circulation de l'air pour des lectures précises des capteurs.
- **Options de montage** : Envisagez comment vous allez monter ou positionner vos Raspberry Pis et capteurs dans chaque pièce pour une mesure optimale.

Concevoir l'Architecture du Système

1. Communication entre les Raspberry Pis

a. Protocole de Communication

- **MQTT (Message Queuing Telemetry Transport)** :
- **Pourquoi MQTT ?** : C'est un choix populaire pour l'IoT car il est léger, conçu pour des réseaux de bande passante limitée, et permet une communication efficace entre appareils. Il utilise le modèle de publication/abonnement, ce qui rend la distribution des messages à plusieurs clients facile et flexible.
- **Broker MQTT**: Vous aurez besoin d'un broker MQTT sur votre Raspberry Pi central. Mosquitto est un choix populaire et facile à configurer.
- **Clients MQTT**: Chaque Raspberry Pi dans les pièces agira en tant que client MQTT, publiant les données des capteurs au broker MQTT.

b. Format des Messages

- **JSON (JavaScript Object Notation)** :
- **Pourquoi JSON ?** : Il est léger, facile à lire, et peut être facilement interprété par la plupart des langages de programmation, ce qui le rend idéal pour l'envoi de données structurées comme celles provenant de vos capteurs.

2. Architecture du Système

a. Raspberry Pis Clients (dans chaque pièce)

- **Tâches** :
- Lire les données des capteurs de température, d'humidité et de qualité de l'air.
- Envoyer les données collectées au Raspberry Pi central à intervalles réguliers sous forme de messages JSON via MQTT.
- Envoyer des signaux de présence (heartbeats) pour permettre la surveillance de l'état de connexion.

b. Raspberry Pi Central (Serveur)

- **Tâches** :
- Agir en tant que broker MQTT pour recevoir les messages des clients.
- Stocker les données reçues dans une base de données.
- Fournir une interface web pour l'affichage des données en temps réel, l'historique des températures, des taux d'humidité, de la qualité de l'air, et l'état de connexion des clients.
- Gérer les alertes en cas de dépassement de seuils ou de déconnexion des clients.

c. Base de Données

- **Type de Base de Données** : SQLite (pour sa simplicité et sa facilité d'utilisation) ou PostgreSQL/MySQL (si vous avez besoin de plus de robustesse et de fonctionnalités).
  - **Stockage des Données** :
  - Température, humidité, qualité de l'air, timestamp de chaque lecture.
  - Timestamps des signaux de présence de chaque Raspberry Pi client.

d. Application Web

- **Backend** : Node.js avec Express, Python avec Flask, ou un autre framework léger pour construire l'API et gérer les interactions avec la base de données.
- **Frontend** : Frameworks modernes comme React ou Vue.js pour construire une interface utilisateur dynamique. Utilisation de bibliothèques comme Chart.js pour les graphiques.
- **WebSockets** (optionnel) : Pour l'affichage en temps réel des données sur l'interface web.

3. Sécurité et Fiabilité

- **MQTT Sécurisé** : Utiliser MQTT sur TLS/SSL pour sécuriser la communication entre les clients et le serveur.

- **Sauvegarde de la Base de Données** : Mettre en place des routines de sauvegarde pour éviter la perte de données.
- **Surveillance du Système** : Implémenter des logs et des systèmes de monitoring pour surveiller l'état de santé de votre système.

En suivant cette architecture, vous aurez un système robuste et évolutif pour surveiller la température, l'humidité, et la qualité de l'air dans différentes pièces, avec une interface web pour visualiser les données. Ce plan peut être ajusté en fonction de vos besoins spécifiques ou des contraintes que vous rencontrez lors de la mise en place. N'hésitez pas à poser des questions ou à demander des éclaircissements sur certains aspects !

Schéma de Base de Données

1. Table `readings`

Cette table stocke les lectures individuelles de chaque capteur. Chaque ligne représente une lecture à un moment donné.

Colonne	Type de Donnée	Description
`id`	INT PRIMARY KEY AUTO_INCREM	Un identifiant unique pour chaque lecture.
`sensor_id`	INT	L'identifiant du capteur qui a effectué la lecture.
`temperature`	FLOAT	La température relevée par le capteur.
`humidity`	FLOAT	Le taux d'humidité relevé par le capteur.
`air_quality`	FLOAT	La qualité de l'air relevée par le capteur (peut être un indice de qualité de l'air ou une mesure spécifique comme le
`reading_tim`	TIMESTAMP	Le moment exact de la lecture.

2. Table `sensors`

Cette table contient des informations sur chaque capteur. Si vous utilisez différents types de capteurs ou si vous souhaitez garder des informations comme l'emplacement du capteur, cette table sera utile.

Colonne	Type de Donnée	Description
`sensor_id`	INT PRIMARY KEY AUTO_INCREMENT	Un identifiant unique pour chaque capteur.
`type`	VARCHAR	Le type de capteur (par exemple, "DHT22", "CCS811", "BME680", etc.).
`location`	VARCHAR	L'emplacement du capteur (par exemple, "Salon", "Chambre", etc.).
`installation_date`	DATE	La date d'installation du capteur.

3. Table `alerts`

Cette table peut être utilisée pour stocker des informations sur les alertes générées lorsque certaines conditions sont remplies (par exemple, température trop élevée, mauvaise qualité de l'air, etc.).

Colonne	Type de Donnée	Description
`alert_id`	INT PRIMARY KEY AUTO_INCREMENT	Un identifiant unique pour chaque alerte.
`sensor_id`	INT	L'identifiant du capteur qui a déclenché l'alerte.
`alert_type`	VARCHAR	Le type d'alerte (par exemple, "Température élevée", "Mauvaise qualité de l'air").
`alert_messag`	TEXT	Un message décrivant l'alerte.
`alert_time`	TIMESTAMP	Le moment exact où l'alerte a été générée.

Relations entre les Tables

- `readings.sensor_id` fait référence à `sensors.sensor_id`.
- `alerts.sensor_id` fait référence à `sensors.sensor_id`.

Cela vous permet de relier les lectures et les alertes à des informations spécifiques sur le capteur.

Indexation

Pour améliorer les performances des requêtes, surtout si votre base de données devient très grande, pensez à indexer les colonnes qui sont souvent utilisées dans les requêtes de recherche, comme `sensor_id` et `reading_time` dans la table `readings`.