Check-list installation

Pré-requis réseau

- [ ] Adresses IP planifiées et réservées
- [ ] Ports 5000 ouverts
- [ ] Accès SSH configuré sur tous les Pi

Installation matérielle (optionnel)

- [ ] Câblage électrique sécurisé
- [ ] Test boutons/LED avant fermeture

Configuration logicielle

- [ ] OS Raspberry Pi OS installé et mis à jour
- [ ] Environnement virtuel créé et activé
- [ ] Librairies installées
- [ ] Scripts Python déployés et testés
- [ ] Vérification des logs (fichiers créés, messages pertinents)
- [ ] Test de rotation des logs (fichier > 1 Mo)

Tests de validation

- [ ] Test unitaire chaque dispositif
- [ ] Test intégration serveur
- [ ] Simulation pannes réseau

Fiche de Test

Objectif : Valider le bon fonctionnement du système d'alerte sur chaque Raspberry Pi et le serveur central.

---

1. Tests unitaires (par Raspberry Pi)

Test 1 : Configuration et démarrage

- **Objectif** : Vérifier que la configuration est correctement chargée.
- **Étapes** :
  1. Lancer le script `systeme_alerte.py` sur le Raspberry Pi.
  2. Vérifier dans la console l'affichage des informations de démarrage :

```
=== SYSTÈME D'ALERTE D'URGENCE (SIMULATION) ===
Matériel : RPi-XX
Serveur : http://localhost:5000/alerte
IP locale : 192.168.1.XXX
```

3. **Validation** :
    - ☐ L'`APPAREIL_ID`, le nom et l'IP locale correspondent à la configuration.
    - ☐ Aucun message d'erreur au démarrage.

Test 2 : Simulation d'alerte

- **Objectif** : Vérifier que l'alerte est envoyée.
- **Étapes** :
    1. Appuyer sur le bouton d'urgence (ou simuler via l'interface clavier).
    2. Observer la LED :
        - **Succès** : 2 clignotements verts (code 200)
        - **Erreur serveur** : 4 clignotements rouges (code 400/503)
        - **Pas de connexion** : 6 clignotements rouges (timeout)
    3. **Validation** :
        - ☐ Le log confirme l'envoi de l'alerte.

Test 3 : Génération des logs

- **Objectif** : Vérifier que les logs sont bien générés et enregistrés.
- **Étapes** :
    1. Dans un terminal, suivre les logs en temps réel :

    ```
    tail -f logs/RPi-XX_application.log
    ```

    2. Déclencher une alerte (via bouton ou simulation clavier).
    3. **Validation** :
        - ☐ Un fichier log est créé dans `logs/RPi-XX_application.log`.
        - ☐ Les messages suivants apparaissent :
            - `INFO - Envoi de l'alerte de type : [TYPE_ALERTE_APPAREIL] depuis [APPAREIL_ID]`
            - `INFO - Alerte envoyée avec succès !` (ou un message d'erreur si problème)
        - ☐ Les logs sont lisibles et horodatés.

---

2. Tests d'intégration (Raspberry Pi + Serveur)

Test 4 : Réception des alertes par le serveur

- **Objectif** : Vérifier que le serveur reçoit et affiche correctement les alertes.
- **Étapes** :
    1. Accéder à l'interface du serveur : http://192.168.1.5:5000
    2. Déclencher une alerte depuis le Raspberry Pi.
    3. Rafraîchir la page `/alertes` du serveur.
    4. **Validation** :

- [ ] L'alerte apparaît dans la liste avec :
  - La source (`APPAREIL_ID`)
  - Le type (`TYPE_ALERTE_APPAREIL`)
  - La localisation (avec la bonne couleur)
  - L'IP source
- [ ] Le log du serveur (`serveur_alertes.py`) confirme la réception :

```
INFO - Alerte reçue de RPi-XX, type :
[TYPE_ALERTE_APPAREIL], localisation : [LOCALISATION]
```

Test 5 : Robustesse du système

- **Objectif** : Vérifier le comportement en cas de problème réseau ou serveur.
- **Étapes** :
  1. **Test 5.1 : Serveur indisponible**
     - Éteindre le serveur central.
     - Déclencher une alerte depuis le Raspberry Pi.
     - **Validation** :
       - [ ] La LED clignote 6 fois en rouge (timeout).
       - [ ] Le log indique : `ERROR - Erreur de connexion, serveur injoignable`.
  2. **Test 5.2 : Format d'alerte invalide**
     - Envoyer une requête mal formée au serveur :

```
curl -X POST http://192.168.1.5:5000/alerte -H "Content-
Type: application/json" -d '{"champ_invalide":"test"}'
```

     - **Validation** :
       - [ ] Le serveur répond avec un code 400.
       - [ ] Le log serveur indique : `WARNING - Données JSON manquantes dans la requête`.

---

3. Tests de performance et maintenance

Test 6 : Rotation des logs

- **Objectif** : Vérifier que la rotation des logs fonctionne.
- **Étapes** :
  1. Générer suffisamment de logs pour dépasser 1 Mo (ex : remplacer 1_000_000 dans le code `config.py` par 10_000 "10 ko").
  2. Vérifier dans le dossier `logs/` :

```
ls -lh logs/
```

3. **Validation** :
   - ☐ Plusieurs fichiers logs sont présents (ex : `RPi-XX_application.log`, `RPi-XX_application.log.1`, etc.).
   - ☐ Le fichier courant ne dépasse pas 1 Mo (ou 10 ko).

Test 7 : Procédure de reset

- **Objectif** : Vérifier que le reset des alertes fonctionne.
- **Étapes** :
  1. Envoyer plusieurs alertes au serveur.
  2. Effectuer un reset via la route `/reset` :

```
curl -X POST http://192.168.1.5:5000/reset
```

  3. **Validation** :
     - ☐ Le serveur répond : `{"status": "OK", "message": "X alertes supprimées"}`.
     - ☐ La liste des alertes est vide sur `/alertes`.

---

5. Check-list de validation

- ☐ Tous les Raspberry Pi démarrent sans erreur.
- ☐ Les logs sont générés et rotatifs.
- ☐ Les alertes sont envoyées et reçues correctement.
- ☐ Le serveur affiche les alertes avec les bonnes couleurs.
- ☐ Les tests de robustesse (timeout, format invalide) sont concluants.
- ☐ La rotation des logs et le reset fonctionnent.

---

6. Rapport de test (à remplir par les étudiants)

Test	Résultat	Observations/Problèmes
Test 1 : Configuration et démarrage	[OK/KO]	
Test 2 : Simulation d'alerte	[OK/KO]	
Test 3 : Génération des logs	[OK/KO]	
Test 4 : Réception des alertes	[OK/KO]	
Test 5.1 : Serveur indisponible	[OK/KO]	
Test 5.2 : Format invalide	[OK/KO]	

Test	Résultat	Observations/Problèmes
Test 6 : Rotation des logs	[OK/KO]	
Test 7 : Procédure de reset	[OK/KO]	

Commentaires généraux : (Ex : "Problème de connexion intermittente entre RPi-03 et le serveur", "Logs trop verbeux en mode DEBUG", etc.)