# Documentation Technique pour l'Application Smart Home Dashboard

## Réalisé par : Yassine Marnissi

## 1. Guide Utilisateur

### Introduction

L'application *Smart Home Dashboard* permet de surveiller et d'automatiser les systèmes de gestion de la température, de l'humidité et de la consommation d'énergie dans une maison connectée. Elle offre une interface interactive et des recommandations en temps réel pour ajuster ces paramètres, selon les seuils définis par l'utilisateur. L'utilisateur peut visualiser les données en temps réel à travers des graphiques interactifs et des cartes de valeurs.

### Installation

1. **Installer R et Python** :
   - R : Téléchargez et installez R depuis https://cran.r-project.org/.
   - Python : Téléchargez et installez Python depuis https://www.python.org/downloads/.
2. **Installer les packages nécessaires** : Ouvrir R et exécuter les commandes suivantes pour installer les bibliothèques nécessaires :

```
install.packages("shiny")
install.packages("shinyWidgets")
install.packages("plotly")
install.packages("data.table")
install.packages("bslib")
install.packages("shinydashboard")
install.packages("reticulate")
```

10. **Configurer l'environnement Python** : Assurez-vous que Python est installé et spécifiez le chemin d'accès dans le code R via `use_python()`. Le chemin doit correspondre à l'emplacement de votre interpréteur Python.

```
use_python("C:/path/to/your/python.exe")
```

12. **Importation du fichier Python** : Le fichier `analyse.py` contenant les fonctions de recommandations doit être accessible. Vous pouvez remplacer le chemin dans le code par celui où vous avez sauvegardé le fichier Python.

### Fonctionnalités Principales

1. **Tableaux de Bord en Temps Réel** :
   - Affiche la température, l'humidité et la consommation d'énergie actuelles.
   - Les cartes indiquent des seuils dynamiques (par exemple, "Température", "Humidité", "Consommation").
2. **Graphiques Interactifs** :
   - Visualisez l'évolution de la température, de l'humidité et de la consommation d'énergie en temps réel avec des graphiques interactifs via *Plotly*.
3. **Automatisation** :
   - Ajustez les seuils de température, d'humidité et de consommation via des

sliders pour personnaliser les règles d’automatisation.

- L'application ajustera automatiquement la configuration des appareils (chauffage, déshumidificateur, etc.).

4. **Recommandations en Temps Réel** :
   - Recevez des conseils directement dans l'application, générés par un script Python. Ces recommandations sont mises à jour en fonction des données d'entrée (température, humidité, consommation).

### Exemples d'Utilisation

1. **Surveiller la température et ajuster l'automatisation** :
   - L'utilisateur peut ajuster le seuil de température via un slider. Par exemple, si la température descend sous 18°C, l'automatisation allumera le chauffage.
   - L'application génère une recommandation, comme "Augmenter la température pour améliorer le confort", en fonction des valeurs observées.
2. **Analyser l'humidité et réagir à la consommation** :
   - Ajustez le seuil d'humidité et surveillez l'humidité actuelle. Si l'humidité est supérieure à 70%, un déshumidificateur pourrait être activé.
   - Recevez une alerte si la consommation d'énergie dépasse le seuil défini (par exemple, "Alerte : Consommation élevée").

## 2. Manuel Technique

### Architecture du Système

L'application est basée sur une architecture cliente-serveur où :

- **Frontend (Shiny)** : Interface interactive permettant à l'utilisateur de visualiser les données et de personnaliser les seuils.
- **Backend (Python via `reticulate`)** : L'application R appelle des fonctions Python pour générer des recommandations en fonction des données collectées.

### Modules de l'Application

1. **Interface Utilisateur (UI)** :
   - Comprend des cartes pour afficher les valeurs de température, humidité et consommation.
   - Contient des graphiques interactifs pour visualiser l'évolution des paramètres.
   - Des boutons et sliders permettent de définir les seuils d'automatisation et de régler la fréquence de mise à jour des données.
2. **Backend (Serveur)** :
   - Gestion des données d'entrée (température, humidité, consommation) et de l'affichage en temps réel via `reactiveVal()` et `reactiveTimer()`.
   - Des appels aux fonctions Python via `reticulate` pour obtenir des recommandations et ajuster l'automatisation.

3. **Appels Python** :
    - Le fichier Python `analyse.py` contient des fonctions de recommandation (`recommander_temperature`, `recommander_humidite`, etc.), qui sont appelées à chaque mise à jour des données.

## Bibliothèques Utilisées

- **R** :
    - `shiny` : Pour créer l'interface utilisateur interactive.
    - `shinyWidgets`, `shinydashboard`, `bslib` : Pour l'interface utilisateur avec des cartes et des thèmes modernes.
    - `plotly` : Pour les graphiques interactifs.
    - `reticulate` : Pour interagir avec Python.
- **Python** :
    - Le script Python (`analyse.py`) génère des recommandations en fonction des données d'entrée et des seuils définis.

## Gestion des Erreurs et Débogage

- L'application vérifie si des données manquent avant de mettre à jour les graphiques ou les cartes, garantissant ainsi qu'il n'y ait pas d'erreurs d'affichage.
- Si les fonctions Python ne retournent pas de résultats valides, un message d'erreur est affiché et l'utilisateur est informé de la situation.

# 3. Rapport de Développement

## Objectifs du Projet

L'objectif principal de cette application est de créer une interface intelligente permettant de gérer en temps réel les conditions d'une maison connectée en fonction des paramètres suivants : température, humidité et consommation d'énergie. L'application doit être capable de générer des recommandations et d'adapter les seuils d'automatisation en fonction des données collectées.

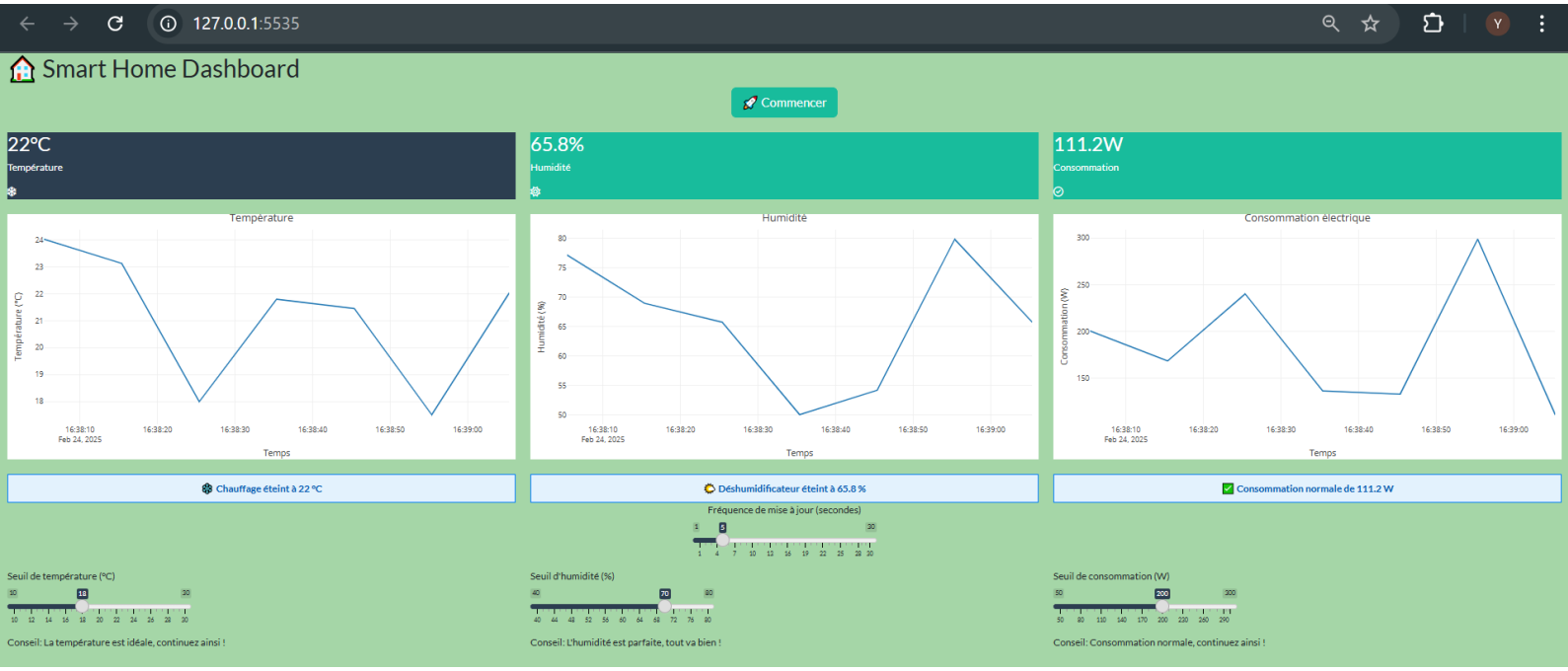
## Conception et Développement

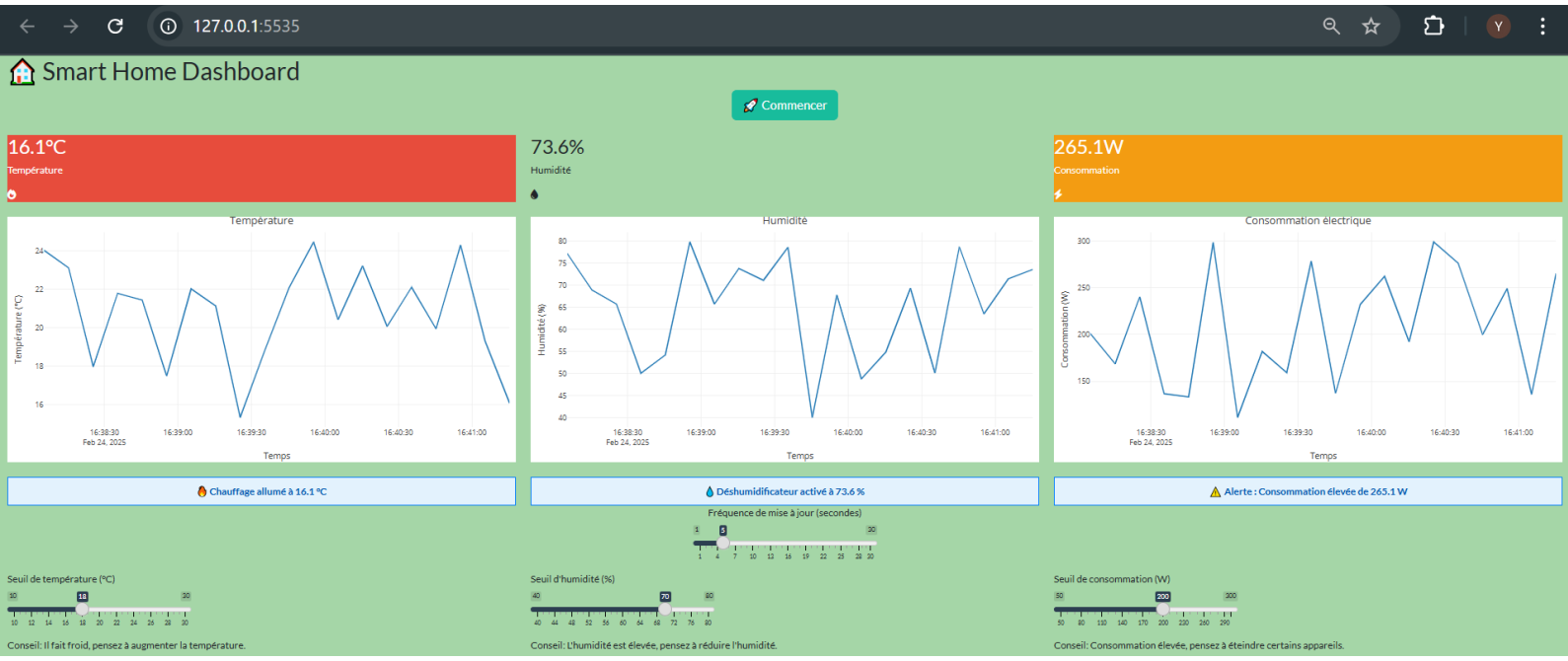
Le développement s'est concentré sur l'intégration fluide entre R et Python, permettant d'afficher des données en temps réel et de donner des recommandations personnalisées. La logique de gestion des seuils d'automatisation a été construite en fonction des données d'entrée.

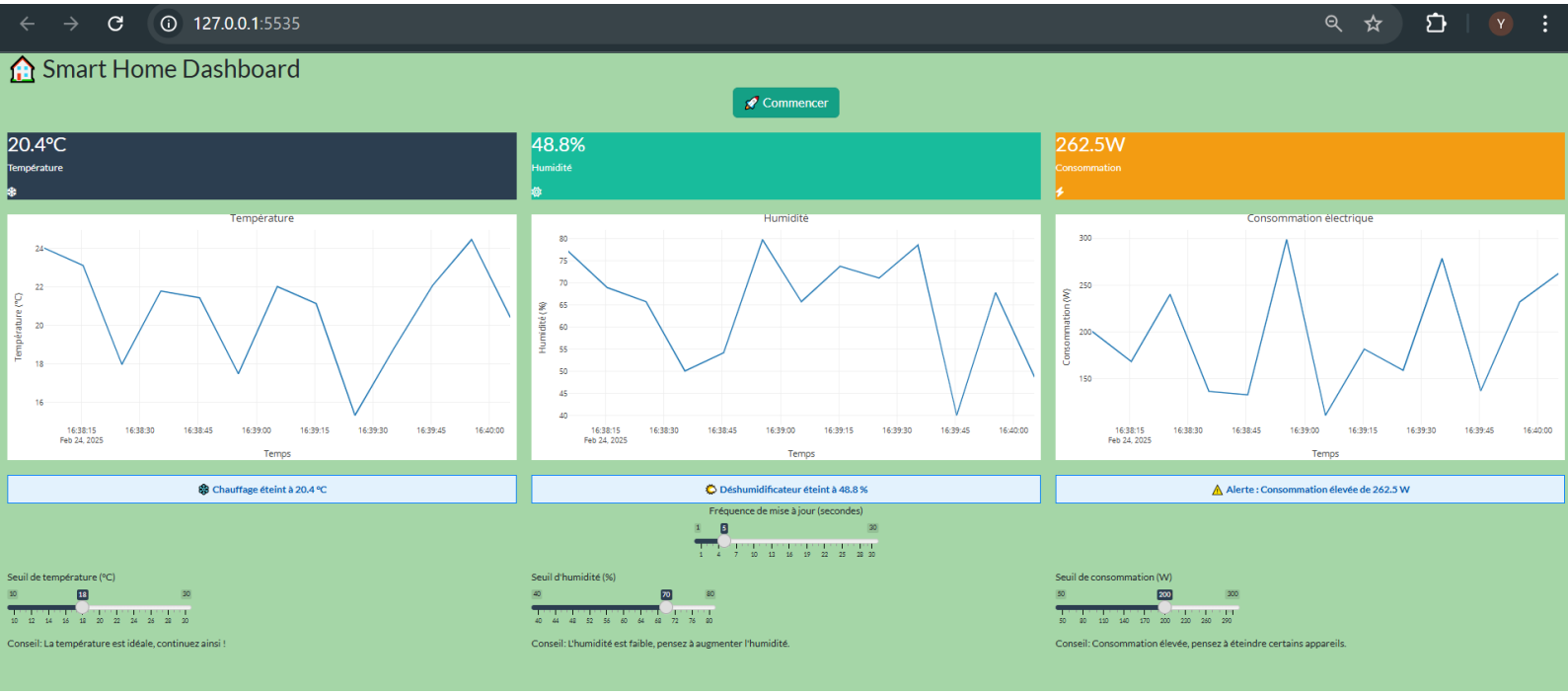
## Résultats obtenus

L'application fonctionne comme prévu, offrant une interface interactive avec des graphiques en temps réel et des recommandations personnalisées. Les utilisateurs peuvent définir des seuils pour chaque paramètre et recevoir des alertes ou des conseils basés sur l'analyse des données.

# Interface d’application






Ici, nous avons défini la fréquence de mise à jour à 5 secondes et fixé les seuils suivants : la température à 18°C, l'humidité à 70 % et la consommation à 200. Nous remarquons que les couleurs changent en fonction des valeurs, avec des conseils recommandés pour l'utilisateur en bas.

## Conclusion

L'application Smart Home Dashboard offre une solution interactive et automatisée pour surveiller la température, l'humidité et la consommation d'énergie d'une maison connectée. Grâce à une interface développée sous R Shiny et une intégration avec Python, elle permet d'afficher des données en temps réel, de générer des recommandations personnalisées et d'adapter automatiquement les seuils d'automatisation. L'architecture client-serveur garantit une communication fluide entre l'interface utilisateur et le moteur de recommandations. L'approche modulaire et l'intégration de bibliothèques avancées comme Plotly et Reticulate assurent une expérience utilisateur optimale.

Ce projet représente une avancée dans la gestion intelligente de l'énergie domestique, offrant un outil efficace pour améliorer le confort et l'efficacité énergétique.