Documentation Technique

Sommaire

Étudiants
Nolhan Biblocque
Léo Guinvarc'h
Victor Jockin
Mathys Laguilliez
Mucahit Lekesiz

Enseignant
André Péninou

Formation
BUT Informatique
2ème Année
Promotion 2024-2025

Établissement
IUT de Blagnac
Université Toulouse II – Jean Jaurès (31)

1. Présentation de l’application

2. Architecture de l'application

2.1 Architecture générale

2.1.1 Sous-systèmes de l'application

Application JavaFX

Programme Python

2.1.2 Fichiers utilisés

Fichier de configuration

Le fichier de configuration `configuration.ini` situé sous le répertoire `resources` contient les paramètres de la configuration créée par l'utilisateur au travers de l'interface de l'application Java. Ce fichier est lu par le programme Python à son lancement qui adapte ainsi son comportement en fonction des paramètres spécifiés.

STRUCTURE DU FICHIER

```
[MQTT] ; [1]
broker=mqtt.iut-blagnac.fr
port=1883
topic={{ PRÉFIXE DU TOPIC MQTT }}

[SUBJECTS] ; [2]
subject1={{ SUJET 1 }}
subject2={{ SUJET 2 }}
...

[DATA_TYPE] ; [3]
dataType1={{ TYPE DE DONNÉES 1 }}
dataType2={{ TYPE DE DONNÉES 2 }}
dataType3={{ TYPE DE DONNÉES 3 }}
...

[THRESHOLD] ; [4]
{{ TYPE DE DONNÉES 1 }}={{ SEUIL }}
{{ TYPE DE DONNÉES 2 }}={{ SEUIL }}
{{ TYPE DE DONNÉES 3 }}={{ SEUIL }}
...

[PARAMS] ; [5]
frequency={{ FRÉQUENCE }}
```

[1] Paramètres de connexion MQTT

- `broker` : Adresse du broker MQTT (valeur fixe).
- `port` : Port utilisé pour la connexion au broker (port standard MQTT, valeur fixe).
- `topic` : Préfixe des topics auxquels le programme Python doit s'abonner.
  - Pour accès aux capteurs AM107, le préfixe correspondant est `AM107/by-room/`.
  - Pour accès aux capteurs SOLAREDGE, le préfixe correspondant est `solaredge/blagnac/`.

[2] Liste des sujets à observer

- `subjectI` : I-ème sujet à observer.
  - Pour les capteurs AM107, le nombre de sujets à observer peut aller jusqu'au nombre total de salles disponibles, soit 53.
  - Pour les capteurs SOLAREDGE, le nombre de sujets à observer se limite à 1 : `overview`.

[3] Liste des types de données à récupérer

- `dataTypeI` : I-ème type de données à récupérer pour le type de capteurs consulté.

[4] Liste des seuils d'alerte par type de données (capteurs AM107 uniquement)

- Cette section indique, pour chaque type de données listé dans la section `DATA_TYPE`, le seuil dont le dépassement déclenchera une alerte.

[5] Paramètres avancés

- `frequency` : Fréquence de lecture des données.
  - **À noter :** La valeur pour ce paramètre n'a actuellement aucun impact sur le comportement du programme Python car non traitée. La fréquence définie lors du paramétrage de la configuration est cependant prise en compte par le processus de lecture des données de l'application Java.

Fichiers de données

Le fichier de configuration `configuration.ini` situé sous le répertoire `resources` contient les paramètres de la configuration créée par l'utilisateur au travers de l'interface de l'application Java.

Fichier de données globales

Fichier d'alertes

Fichiers de données par sujet

2.1.3 Interactions entre sous-systèmes et fichiers

1. **Écriture du fichier de configuration par l'application Java**
   - Après le paramétrage d'une configuration par l'utilisateur dans l'interface graphique, l'application Java crée un fichier `configuration.ini` sour le répertoire `resources` décrivant la configuration créée.

    - **À noter :** À cette étape, si un fichier de configuration existe déjà, celui-ci est remplacé par le fichier de configuration nouvellement créé. Aucun mécanisme d'historisation ou de sauvegarde des fichiers de configurations n'a été mis en place.
2. **Lancement du programme Python par l'application Java**
    - Une fois le fichier de configuration créé, l'application Java démarre le processus de collecte des données en lançant en exécution le programme Python.
3. **Collecte des données par le programme Python**
    - Au lancement, le programme Python lis le fichier de configuration définissant son comportement.
    - Une fois lancé, il attend jusqu'à interruption les données envoyées par les sujets (capteurs).
    - À chaque réception de données, celles-ci sont enregistrées dans les fichiers de données correspondants.
4. **Lecture des fichiers de données par l'application Java**
    - En parallèle de l'exécution du programme Python, l'application Java lis à intervalle régulier (fréquence définie dans le fichier de configuration) les fichiers de données.
    - Les données lues sont ensuite stockées dans des structures de données puis transmises au tableau de bord de l'application pour affichage.
5. **Interruption du programme Python par l'application Java**
    - Lorsque le tableau de bord de l'application est fermé par l'utilisateur, le programme Python est automatiquement arrêté.
    - **À noter :** Après arrêt du processus de collecte des données, le fichier de configuration ainsi que les fichiers de données écrits sont conservés. Ils seront écrasés lors de la prochaine exécution de l'application.

2.2 Ressources externes

2.2.1 API utilisées

JavaFX

- **Rôles :**
    - Conception de l'IHM avec le module `javafx-fxml` (création d'interfaces utilisateur via des fichiers FXML).
    - Prise en charge et gestion de l'interface graphique dans l'application.
- **Version utilisée :** 17
- **Site officiel de JavaFX :** JavaFX - Home
- **Documentation officielle :** Oracle - JavaFX Documentation

OpenCSV

- **Rôle :** Lecture des fichiers de données au format `CSV` générés par le programme python

collecteur de données.

- **Version utilisée :** 5.5.2
- **Site officiel de JavaFX :** OpenCSV - About / Opencsv Users Guide
- **Documentation officielle :** OpenCSV - About / Developer Documentation

2.2.2 Plugins utilisés

JavaFX Maven Plugin

- **Rôle :** Packaging et exécution de l'application JavaFX.
- **Version utilisée :** 0.0.8
- **Site officiel de Maven Repository :** Maven Repository - JavaFX Maven Plugin Maven Mojo
- **Lien vers le dépôt GitHub du plugin :** GitHub - Maven plugin for JavaFX

Apache Maven Shade Plugin

- **Rôle :** Création d'un exécutable au format `JAR` contenant toutes les dépendances nécessaires au fonctionnement de l'application.
- **Version utilisée :** 3.4.1
- **Site officiel d'Apache Maven :** Apache Maven Project - Apache Maven Shade Plugin

Apache Maven Javadoc Plugin

- **Rôle :** Génération de la documentation du projet Java avec `Javadoc`.
- **Version utilisée :** 3.4.1
- **Site officiel d'Apache Maven :** Apache Maven Project - Apache Maven Javadoc Plugin

2.3 Structuration de l'application

2.4 Spécifications techniques

3. Conception et mise en oeuvre fonctionnalités

4. Procédures d'installation

4.1 Installation pour développement

4.1.1 Prérequis

1. **Installer l'environnement de développement Java**
   - Télécharger le **JDK 17** (ou version compatible) depuis le site officiel d'Oracle : Oracle - Java Downloads.
   - Installer le JDK en suivant les instructions indiquées par l'installateur.
   - Si nécessaire, ajouter le chemin vers le JDK à la variable d'environnement `PATH`.
   - Dans un terminal, vérifier l'installation avec la commande `java -version` ou `java --version`.
2. **Installer Apache Maven**
   - Télécharger **Maven** (archive ZIP) depuis le site officiel d'Apache Maven : Apache Maven Project - Downloading Apache Maven.
     - Pour une installation sur Linux ou Mac OS, télécharger la ***Binary tar.gz archive***.
     - Pour une installation sur Windows, télécharger la ***Binary zip archive***.
   - Ajouter le chemin vers Maven à la variable d'environnement `PATH`.
   - Dans un terminal, vérifier l'installation avec la commande `mvn -version`, `mvn --version` ou `mvn -v`.
3. **Configurer un IDE**
   - Si nécessaire, installer des plugins de prise en charge de **Maven** et **JavaFX** dans l'IDE utilisé pour le développement.

4.1.2 Étapes d'installation

1. **Cloner le dépôt du projet**
   - Accéder au dépôt GitHub du projet : GitHub - SAE S3.01 DevApp
   - Cloner le dépôt du projet via la commande :

     ```
     git clone https://github.com/IUT-Blagnac/sae-3-01-devapp-2024-2025-g2b12.git
     ```

   - Accéder au répertoire du projet Java situé sous `solution iot/application_iot` via la commande :

     ```
     cd solution\ iot/application_iot
     ```

2. **Construire le projet avec Maven**
   - Supprimer les fichiers et ressources précédemment compilés avec la commande `mvn clean`

puis compiler le projet Java via la commande `mvn install`. Il est également possible d'utiliser directement la commande `mvn clean install`.

3. **Exécuter l'application depuis Maven**
    - Exécuter le projet JavaFX via la commande `mvn javafx:run`.

4.2 Installation pour utilisation

4.2.1 Prérequis

1. **Installer le Java Runtime Environement (JRE)**
    - Vérifier que Java est installé sur la machine en exécutant la commande `java -version` dans un terminal.
    - Si Java n'est pas installé, télécharger et installer le **JRE 8** ou version ultérieure depuis le site officiel de Java : Java - Télécharger Java.
2. **Installer Python 3**
    - Vérifier que Python en version 3 est installé sur la machine en exécutant la commande `python -version` ou `python3 -version` dans un terminal.
    - Si Python n'est pas installé, télécharger et installer la dernière version disponible sur le site officiel de Python : Python - Downloads.

4.2.2 Étapes d'installation

1. **Télécharger l'application**
    - Télécharger l'archive de l'application (fichier ZIP) située sous le répertoire `livrables/IoT` du dépôt GitHub du projet : GitHub - Livrables IoT
        - Pour une installation sur Mac OS, préférer l'archive `application_jar_mac_os.zip`.
        - Pour une installation sur Windows ou Linux, préférer l'archive `application_jar_windows.zip`.
2. **Décompresser l'archive de l'application**
    - Décompresser l'archive téléchargée dans un répertoire à l'aide d'un outil de décompression tel que **WinRAR** ou **7-Zip**.
    - L'arborescence de l'application après décompression doit ressembler à ceci :

```
application/
|-- ressources/
|   |-- data/
|   |-- configuration.ini
|   |-- mqtt.py
|-- application_iot-1.0-SNAPSHOT-shaded.jar
```

4.2.3 Étapes de lancement

1. **Lancer l'application dans le gestionnaire de fichiers**
   - Lancer l'exécutable `application_iot-1.0-SNAPSHOT-shaded.jar` en double-cliquant sur celui-ci.
   - *Le menu principal de l'application devrait alors apparaître à l'écran.*
2. **Lancer l'application en ligne de commande**
   - Ouvrir un terminal et se placer dans le répertoire `application` à l'aide de la commande `cd`.
   - Lancer ensuite l'exécutable de l'application via la commande :

     ```
     java -jar application_iot-1.0-SNAPSHOT-shaded.jar
     ```

   - *Le menu principal de l'application devrait alors apparaître à l'écran.*

Étudiants
Nolhan Biblocque
Léo Guinvarc’h
Victor Jockin
Mathys Laguilliez
Mucahit Lekesiz

Enseignant
André Péninou

Formation
BUT Informatique
2ème Année
Promotion 2024-2025

Établissement
IUT de Blagnac
Université Toulouse II – Jean Jaurès (31)