Groupe G2B-10 :

DOURLENT Maxime

STRAPUTICARI Luca

FERNANDEZ Mickaël

VIGNAL Alexandre

Documentation Java

Présentation rapide de l'application : **2**

Explication du Use Case global : **3**

Architecture : **5**

Architecture générale : 5
Ressources externes utilisées : 6
Structuration en packages de l'application documentée : 6
Eléments essentiels à connaître : 7

Fonctionnalités : **8**

Quitter l'application et fermeture de l'application 9
Visualiser les données des capteurs dans des graphiques 10
Visualiser une alerte de dépassement de seuil de valeur 11
Visualiser les 'logs' de relevés et d'alertes 12
Modifier la configuration de l'application de récupération des données 13

Procédures d'installation : **14**

Installation de l'environnement de développement 14
Installation et lancement de l'application 16

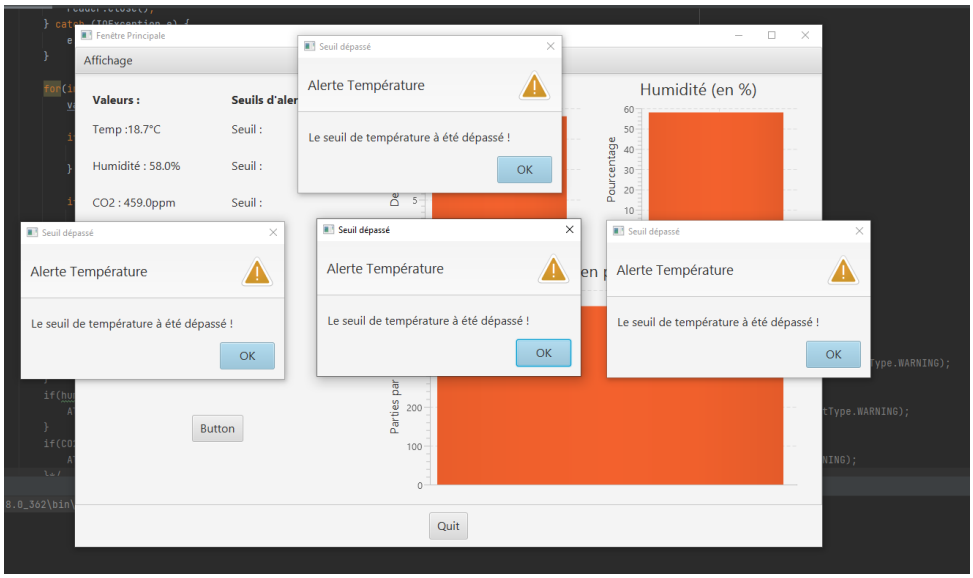
Présentation rapide de l'application :

L'application Java est une interface permettant à l'utilisateur de paramétrer un fichier de configuration qui servira au programme python associé. Ainsi en reliant ces deux programmes nous pourrons simuler la récupération de valeur depuis un entrepôt. Java étant la partie visuel de l'application et Python la partie fonctionnelle de récupération de données. Avec un timer pour la fréquence de lecture des données. Le fichier config.txt est sous le format suivant : "donnée:oui? 1 0 : seuil d'alerte.

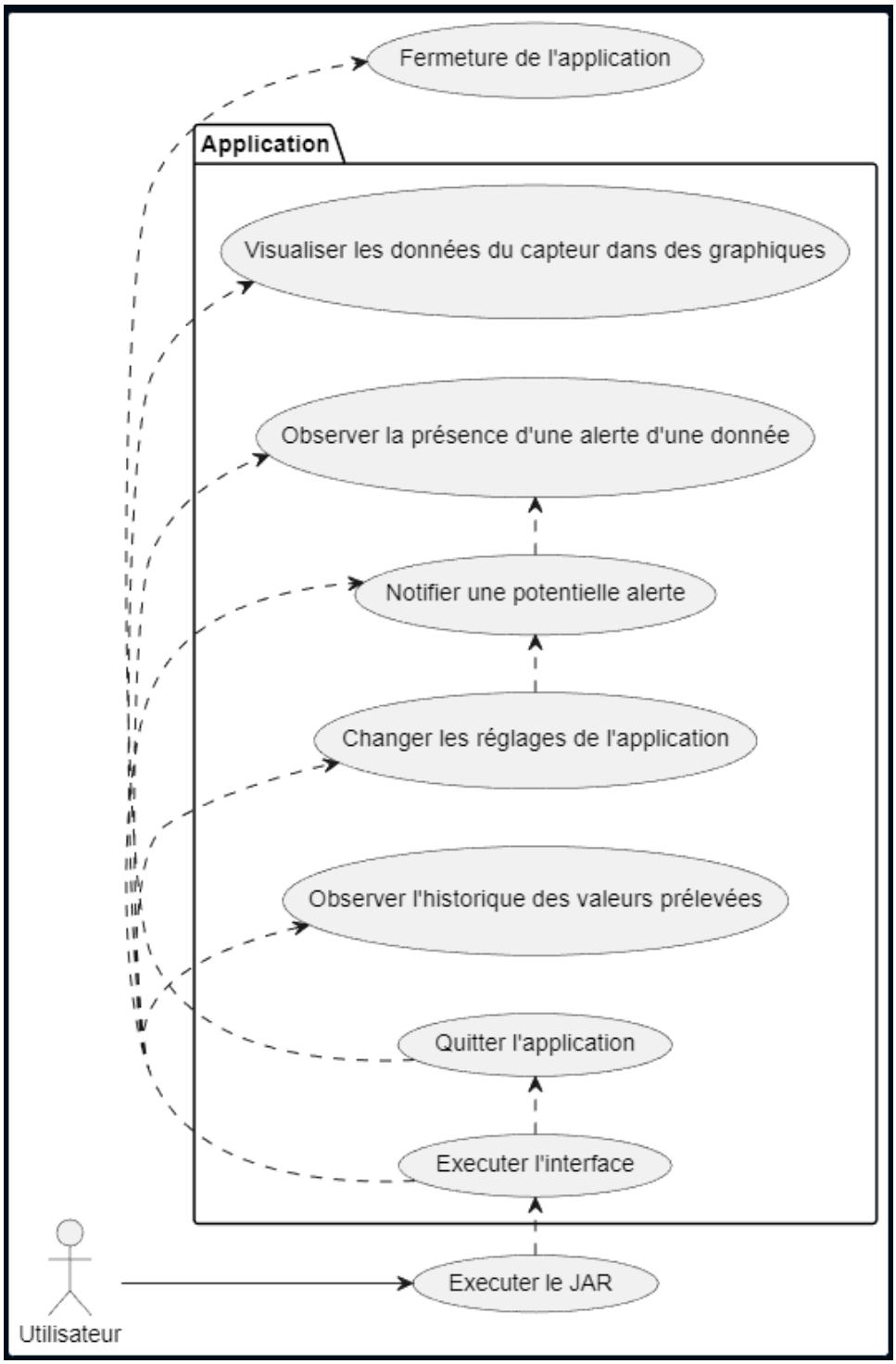
En résumé, cette application permet à un utilisateur de paramétrer le fichier configuration d'indiquer les données qu'il souhaite récupérer, leurs seuil d'alerte associé et les capteurs sur lequel il souhaite se connecter.

Grace à cela il récupère les données des capteurs des entrepôts qu'il dispose

Voici une capture d'écran de l'application en fonctionnement



Explication du Use Case global :



Ce Use Case représente un utilisateur de l'application que nous avons développée. Dans ce Use Case, nous pouvons voir les différentes intéractions que l'utilisateur peut avoir sur l'application en général.

L'utilisateur peut donc :

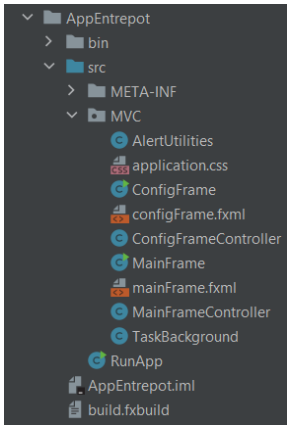
- Visualiser sous forme de graphique les données des capteurs notés dans les paramètres.
- Observer la présence d'une alerte si une valeur est supérieure au seuil qui lui est accordé.
- Changer les réglages de l'application, tel que modifier le serveur, le port, le timer, les capteurs qui sont impliqués, les données que l'on souhaite récupérer et leurs seuils respectifs. Ainsi les changements sont pris en compte dans le fichier "config.txt" pour les conserver.
- Possibilités de voir les différents Logs : LogsReleve.txt et LogsAlerte.txt

Architecture :

Architecture générale :

Au niveau de l'architecture générale de l'application nous avons utilisés le principe de Model View Controler (MVC), cependant les fichiers java ne sont pas rangés dans des packages correspondant à cette structure par soucis d'erreurs d'accès aux différentes ressources lors de l'exécution du jar.

L'architecture est donc arrangée comme ci dessous :



On retrouve dans cette architecture trois types de fichier :

- Les modèles : MainFrame, ConfigFrame, TaskBackground
- Les vues : configFrame.fxml, mainFrame.fxml, application.css
- Les contrôleurs : ConfigFrameController, MainFrameController, TaskBackground

Un modèle (Model) contient les données à afficher.

Une vue (View) contient la présentation de l'interface graphique.

Un contrôleur (Controller) contient la logique concernant les actions effectuées par l'utilisateur.

Ressources externes utilisées :

Nous avons utilisés l'extension .jar pour l'éxecutable de l'application.

Cette extension est très pratique, répandue. De plus nous l'avons deja utilisé lors de projets précédents.

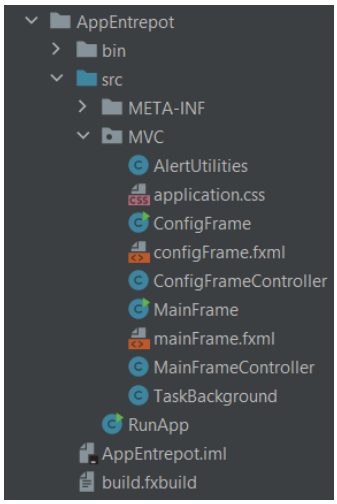
Nous avons utilisé le JRE Java 1.8 car l'application utilise JavaFX.

Concernant l'interface, nous avons optés pour des fichiers FXML avec le logiciel SceneBuilder que nous avions déjà utilisé lors de projets antérieurs. De ce fait, l'interface du logiciel nous est familière et nous permet de ne pas nous attarder sur la partie interface de l'application.

Bien évidemment, afin que l'application fonctionne il faut que le fichier "config.txt" soit présent et qu'il soit au même endroit que le .jar de l'application.

Structuration en packages de l'application documentée :

Nos fichiers sont contenus dans un paquetage nommé "MVC", cependant comme mentionné précédemment, une erreur au niveau du lancement de l'application nous empêchait d'organiser l'architecture sur le mode l: Model View Controller avec les paquetages correspondant.



Chaque méthode de l'application est commentée afin que d'autres développeurs puissent rapidement comprendre le code. De plus, les points importants sont bien expliqués dans tous nos fichiers.

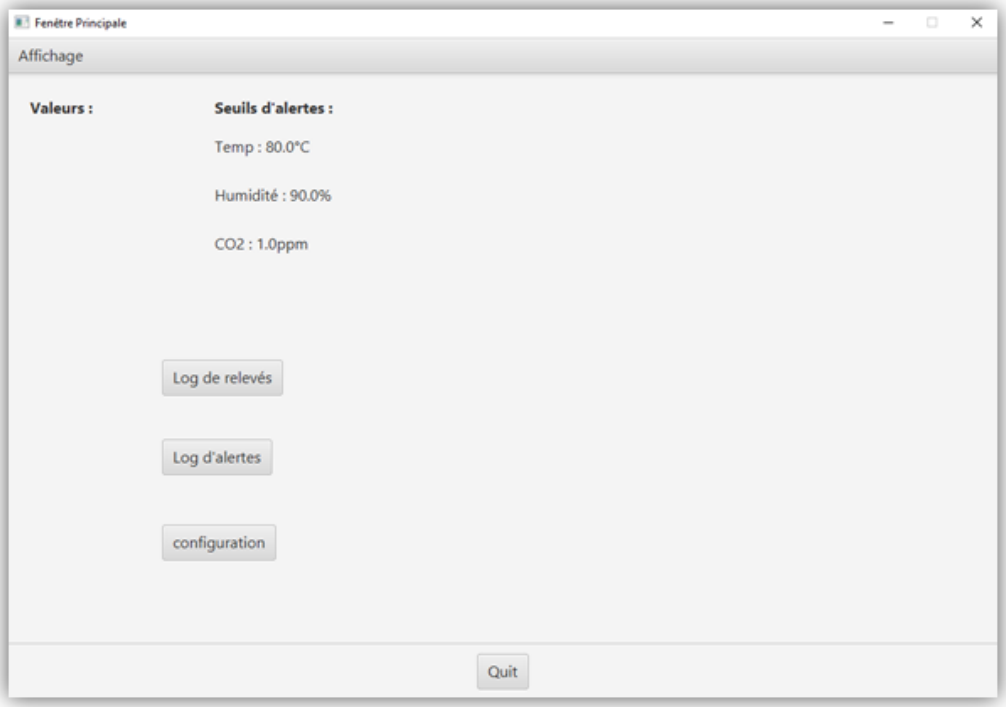
Eléments essentiels à connaître :

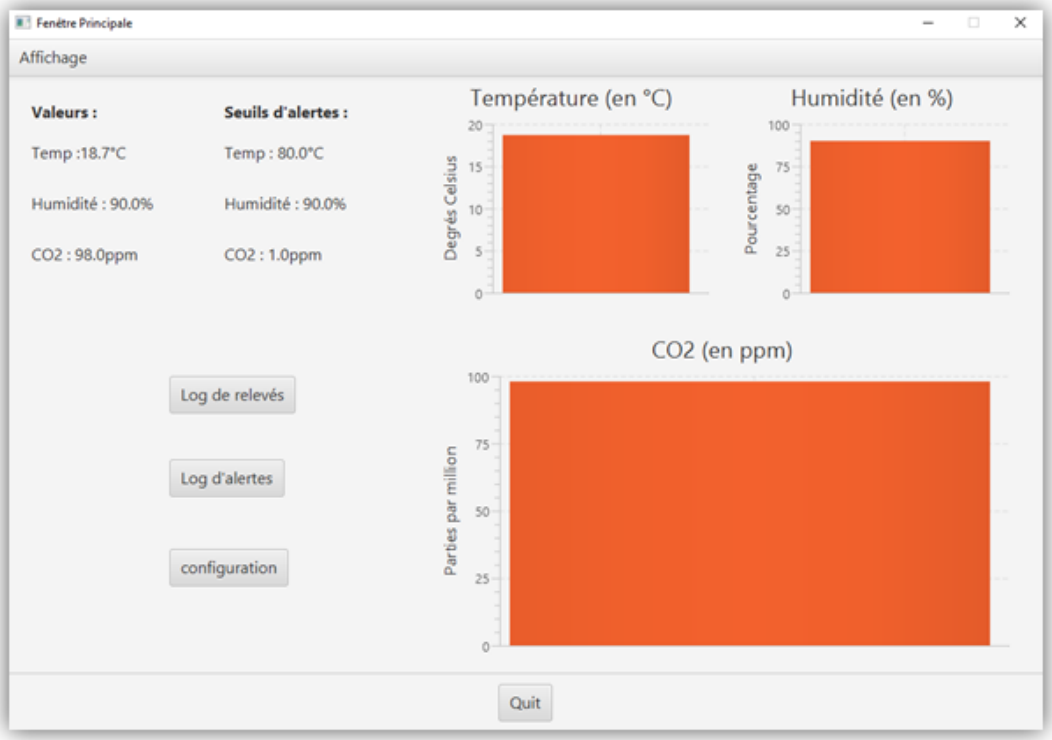
Dans l'application nous utilisons un thread pour le côté dynamique de l'application .

L'application modifie et lit les données du fichier "config.txt" .

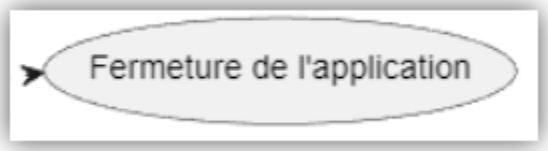
Les données stockées dans "config.txt" sont de la forme clé:valeur

Fonctionnalités :





Quitter l'application et fermeture de l'application

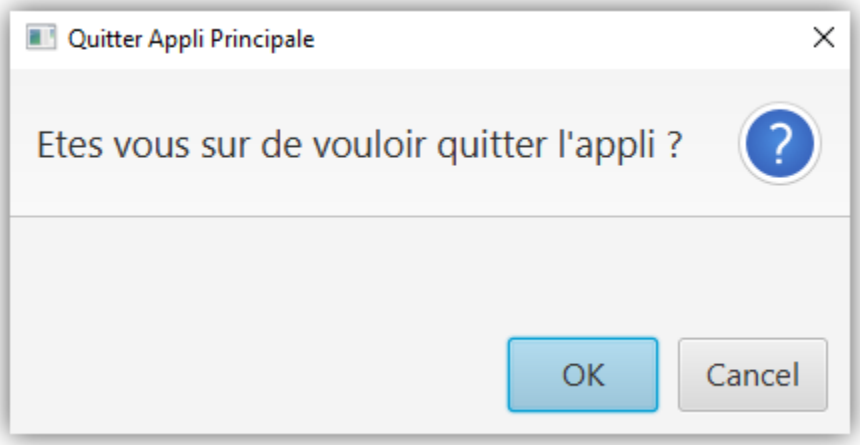


Fichiers concernés :

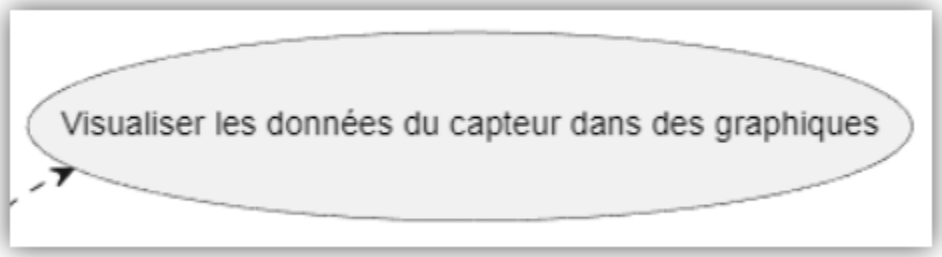
-AlertUtilities : confirmYesCancel()

-MainFrameController : doQuit()

Dans la fenêtre, on peut utiliser le bouton 'Quit' ou la croix pour lancer une alerte de confirmation pour ensuite fermer l'application



Visualiser les données des capteurs dans des graphiques

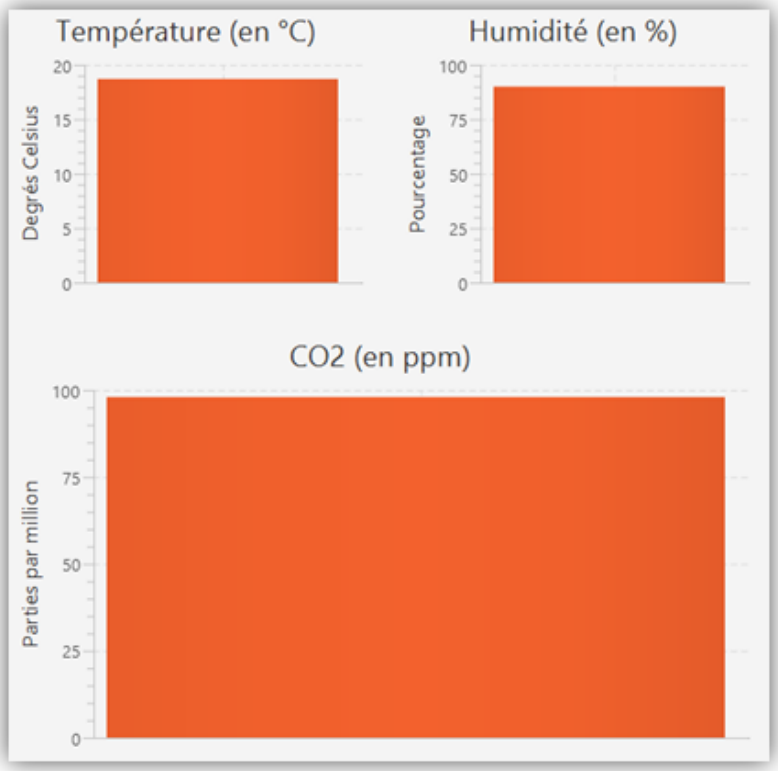


Fichiers concernés :

-MainFrameController : miseAJourCharts()

-TaskBackGround : run()

Dans la fenêtre, On peut voir des BarCharts visualisant les valeurs du dernier relevé, les valeurs changent dynamiquement grâce à un 'Timer'.



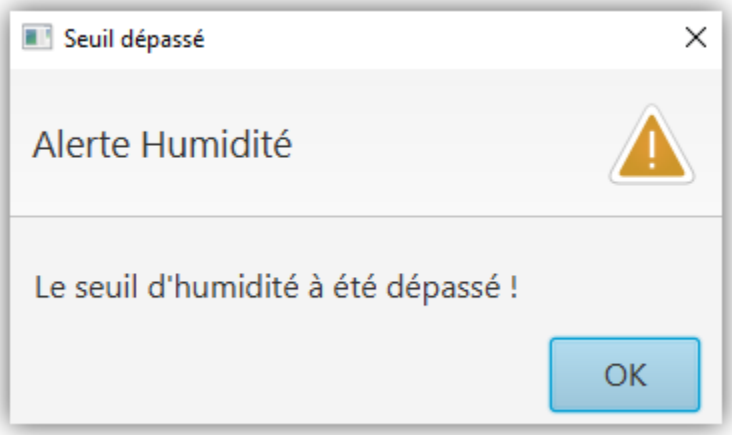
Visualiser une alerte de dépassement de seuil de valeur

Fichiers concernés :

-AlertUtilities : showAlert()

-TaskBackGround : run()

Lorsque les valeurs récupérées dynamiquement dépassent le seuil donné par le fichier ‘config.txt’, une alerte est déclenchée à l’écran.

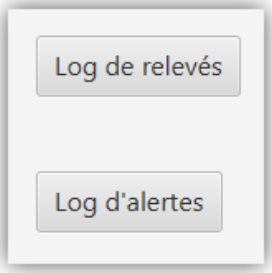


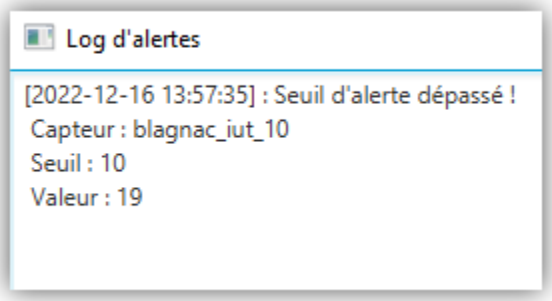
Visualiser les ‘logs’ de relevés et d’alertes

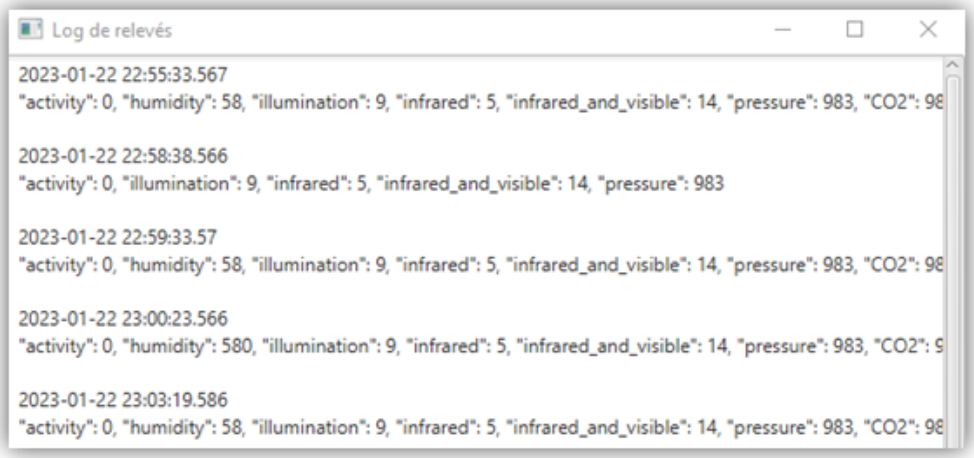
Fichiers concernés :

-MainFrameController : openLogWindow()

Comme les valeurs affichés ne sont que les plus récentes, il est possible de regarder un historique des valeurs et alertes enregistrées par l’application.







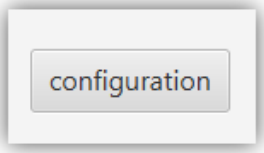
Modifier la configuration de l'application de récupération des données

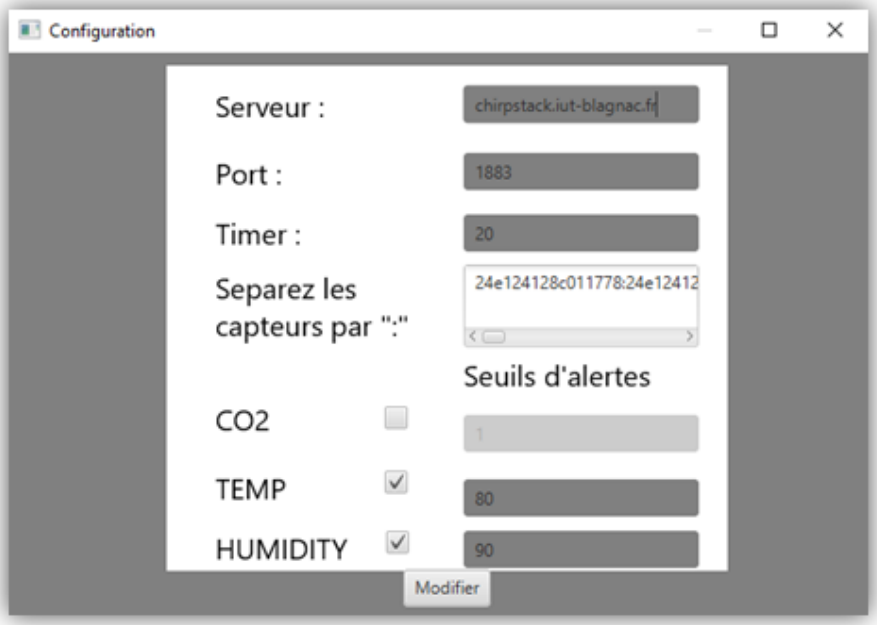
Fichiers concernés :

-ConfigFrameController : initialize() saveConfiguration()

En appuyant sur le bouton 'Configuration', on accède à une page formulaire qui nous permet de modifier le fichier 'config.txt', modifiant les paramètres de récupération des données.

Le formulaire est pré-rempli avec les informations déjà présentes dans le fichier.





Procédures d'installation :

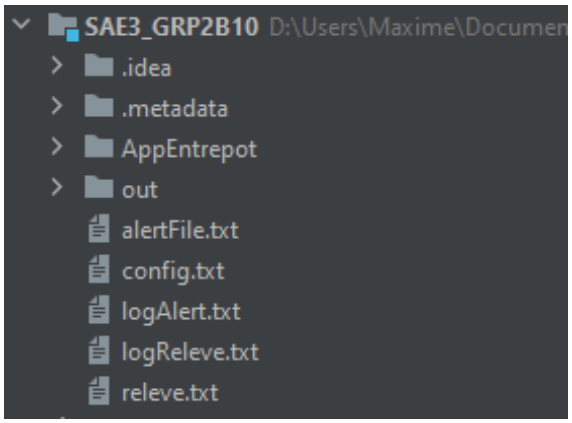
Installation de l'environnement de développement

Récupérer le workspace 'SAE3_GRP2B10' et l'ouvrir avec l'IDE de votre choix, celui que nous avons utilisés pour le développement est IntelliJ.

Il faut utiliser un SDK java 1.8 pour continuer le développement.

Nous recommandons l'outil 'SceneBuilder' pour travailler sur les fichiers FXML.

Les fichiers utilisés par l'application tel que 'config.txt' doivent se trouver au premier niveau du workspace lors du développement :



Installation et lancement de l'application

Vérifier que la version de java soit en 1.8 avec 'java -version'

Si ce n'est pas le cas, faire 'sudo update-alternatives –config java' pour vérifier les versions installées et choisir la 1.8 si elle est disponible.

Si elle n'est pas disponible, l'installer avec 'sudo apt-get install openjdk-8-jdk' puis vérifier a nouveau la version actuelle.

Pour lancer l'application, il est nécessaire d'avoir les fichiers « config.txt, AlertFile.txt et Releve.txt » dans le même répertoire que le fichier.jar.

Il suffit ensuite d'utiliser la commande 'java -jar SAE3_GRP2B10.jar' pour lancer l'application.