**Rapport NeoCampus**

Goupe : WASHBROOK Thomas, SALAMA Karim, JOUANNEAU Gatien, LARRY Guillaume

**Première partie : interface de simulation**

**1 .Répartition des taches**

Pour ce projet nous avons décidé de travailler avec un groupe de 4 personnes. Au début nous avions du mal à nous organiser, nous ne savions pas par où commencer le projet. Ensuite Karim s’est imposé en tant que leader du projet, il a réparti la charge de travaille afin de pouvoir finir l’interface de simulation pour le début des vacances de fin d’année.

-Thomas a fait la partie graphique.

-Karim a fait la connexion au serveur.

-Gatien a réalisé la génération des données et la barre de progression.

-Guillaume a modélisé les capteurs via les différentes classes.

**2. Choix et problèmes lors du code**

On a eu des problèmes sur l’utilisation du gridBagLayout pour faire des fenêtres dynamiques, nous voulions que lorsque l’utilisateur redimensionne la fenêtre les différents éléments s’adaptent aux dimensions de la fenêtre mais nous avions pas le temps de réussir à maitriser le gestionnaire de répartition, c’est pour cela que la première interface de simulation est statique.

Un deuxième problème s’est posé à nous quand nous avons dû lire les différentes valeurs pour la position d’un capteur intérieur (salle, étage, bâtiment) depuis le fichier contenant ces valeurs -format\_bat-.

Nous avons aussi décidé d’implémenter une barre de progression pour l’interface de simulation afin d’améliorer le rendu visuel pour l’utilisateur afin qu’il sache quand une valeur est envoyée au serveur.

Lors de cette première partie nous avons utilisé un scanner pour lire les données envoyées par le serveur ce qui nous n’a pas posé de problème avant la deuxième partie, où nous avons plutôt opté pour un buffer-reader car il y avait des problèmes des récupération des données envoyées par le serveur.

**Deuxième partie : interface de visualisation**

**1 .Répartition des taches**

Pour cette deuxième partie nous nous sommes plus facilement réparti la charge de travail qu’à la première partie. Nous avons organisé un Git Hub pour nous permettre à chacun de s’occuper de sa partie et de travailler efficacement tout en regroupant sous Git Hub .

-Thomas a réalisé le renderer des tableaux, les alertes et la partie graphique.

- Karim a fait la connexion au serveur, l’inscription des différents capteurs et le Controller.

-Guillaume a fait le gestionnaire de fichiers et les graphiques.

-Gatien a fait le TreeModel et le TableModel.

**2.Choix et problème lors du code**

Pour les alertes nous avons décidé de créer un onglet alerte dans l’interface de simulation afin que l’utilisateur puisse configurer facilement ses différentes alertes. Il peut ainsi sélectionner le capteur sur le quel il veut appliquer l’alerte, avec le tableau représentant les capteurs inscrits, il peut ensuite configurer le seuil de son alerte et choisir si il veut l’alerte pour les valeurs inférieures ou supérieures à celui-ci. Puis sur le second tableau il peut visualiser les alertes précédemment créer ainsi que les supprimer en sélectionnant l’alerte.

Nous avons essayé de respecter le motif d’architecture : Modèle View Controller (MVC). Pour ce faire nous avons créé une classe JTableRender qui permet de faire les couleurs pour les alertes (View). Le TableModel et le TreeModel permettent de savoir quels éléments à afficher dans l’arborescence des capteurs et dans les tableaux (Model). Le Controller lui est dans l’interface de simulation qui permet de gérer les évènements des différents éléments graphiques à la disposition de l’utilisateur.

Pour les graphiques nous avons décidé d’utiliser la librairie JFreeChart car c’est une bibliothèque libre de droit et facile d’utilisation.

Pour enregistrer les valeurs des capteurs nous avons décidé d’enregistrer directement les objets java dans les fichiers.

Au début nous n’avions pas utilisé le MVC mais cela nous a posé beaucoup de problèmes et de lisibilité de code. Nous avons donc décidé de revenir en arrière pour implémenter ce motif d’architecture afin de faciliter grandement le code ainsi que la lisibilité pour peut être une maintenance futur.

Nous avons rencontré un problème final quand nous avons dû rassembler nos différentes versions, pour le résoudre nous avons modifié plusieurs parties du code afin qu’ils puissent marcher ensemble.

**Bilan**

Au début du projet nous avions du mal à nous organiser et à répartir les différentes tâches. Au fur et à mesure de l’avancement du projet nous nous sommes de mieux en mieux organisés pour résoudre les problèmes. Nous avons même dû consulter plusieurs fois le tuteur du projet pour résoudre les problèmes les plus ardus. La gestion du temps était aussi un problème surtout pour la première partie dû aux partiels que nous avions.