DomotIcApp - Rapport Technique

Diane Dewez, Iman Akabi,
Enora Lucas, Meven Moser, Corentin Chatellier

Encadrants : François Pasteau, Daniel Guillard

1 Architecture Globale

Le projet est divisé en deux parties distinctes : le modèle et l'interface graphique.

1.1 Modèle

Le modèle est composé de trois DLL :
- ModelDll.dll : c++, IDE : Visual Studio 2015
- RequeteHttp.dll : `C++`, IDE : Code : :Blocks
- Bluetooth_com.dll : `C++`, IDE : QtCreator

Ces DLL doivent être placées dans le même dossier que l'exécutable. Le modèle a été intégralement développé en C++, dans le but d'être portable pour permettre le portage de l'application sur un autre système rapidement. Ce code a ensuite été compilé sous la forme d'une DLL. De cette manière le code est réutilisable simplement au sein de l'application, et la partie modèle est bien indépendante de la partie IHM. Le code s'articule autour d'un objet Core, qui contient les paramètres de l'application et la liste des pièces de la maison. Il permet également de gérer ces mêmes paramètres, d'ajouter ou de retirer des pièces de l'application, et il s'occupe de la sauvegarde et du chargement de l'application à partir d'un fichier.

La modélisation des pièces, et des équipements contenus par ces pièces s'apparente à un arbre, dont la racine est l'objet Core. Nous avons donc défini un type Node, qui correspond à un nœud de l'arbre. Ces nœuds ont un nom et une icône pour permettre leur affichage dans l'IHM. Ce type Node est spécialisé ensuite par les classes Room et Equipment.

Un objet Room représente donc une pièce de la maison. En plus des attributs fournis par la classe Node dont ils héritent, ces objets possèdent une liste d'équipements. Là encore, tout comme la classe Core, la classe Room permet de gérer cette liste d'équipements. Ensuite vient la classe Equipment, qui représente un équipement. Les deux types d'équipements gérés par l'application sont les équipements liés à une box Kira ou à une box Fibaro, il y a donc deux sous-classes pour modéliser ces deux types d'équipements. Chacune de ces sous-classes apportent les éléments nécessaires à la gestion de ces types d'équipements (numéro de bouton et de page pour la Kira, identifiant de l'équipement et action à effectuer pour la Fibaro). Elles permettent ensuite, via la méthode sendRequest(), d'exécuter la requête HTTP permettant d'interagir avec l'équipement ciblé. Cette requête HTTP sera lancée via une deuxième DLL, qui s'occupe uniquement d'envoyer une requête pour un équipement Fibaro ou Kira avec les paramètres fournis. Cette DLL s'appuie sur la librairie HappyHttp.

Enfin, la communication avec le fauteuil de l'utilisateur via Bluetooth s'appuie sur les librairies de Qt. Bluetooth_com.dll et permet de récupérer directement les informations utiles, avec une méthode correspondant à une information du fauteuil.

Les points d'entrée utilisés dans le `C#` pour l'importation des méthodes des différentes DLLs ont été récupérés via l'utilitaire `dumpbin.exe`, (normalement situé à l'emplacement `{Repertoire de Visual Studio}/VC/bin/`), de la manière suivante :
`./dumpbin.exe /exports <chemin vers la DLL>`

1.2 Interface graphique

Cette partie correspond aux pages de l'application telles qu'on peut les voir et à leur interaction avec le modèle. Elle est développée avec XAML et .Net. Langages :

- XAML, pour l'interface utilisateur
- `C#`, pour le code.

IDE : Visual Studio 2015
Versions : .Net Framework : v2.0.5072

Le projet est composé de neuf pages, c'est-à-dire neuf interfaces différentes avec lesquelles l'utilisateur peut interagir :

- MainPage : page principale correspondant à la partie "Utilisateur". Elle est notamment composée d'une grille de pièces. Les pièces sont récupérées à partir de ModelDll.dll. Lors du clic sur une pièce, on affiche la grille des équipements associés à cette pièce (qu'on récupère aussi à partir de ModelDll.dll). Cette page est lancée à l'ouverture de l'application.
- AdminPage : page permettant d'accéder aux diverses options de configuration (gestion pièces, équipements, réseau, interface).
- RegaglesReseau, ReglagesCouleur ,ReglagesModeSelection, ReglagesTailleIcones : ces quatre pages permettent de modifier certains paramètres réseau (pour la KIRA et la FIBARO) et certains paramètres de l'interface graphique (tailles des icônes, couleurs du thème, etc. . .). Les nouveaux paramètres sont alors enregistrés grâce à ModelDll.dll.
- GestionPieces : page permettant d'ajouter des pièces, de les renommer, de les supprimer ou d'y ajouter des équipements. Les modifications sont alors enregistrées à l'aide de ModelDll.dll.
- GestionEquipments : page permettant d'ajouter des équipements (selon type : FIBARO ou KIRA), de les supprimer ou de les renommer. Les modifications sont alors enregistrées à l'aide de ModelDll.dll
- WheelChairFeedback : page affichant les données de certains capteurs du fauteuil comme la vitesse ou la position du joystick par exemple. Ces informations sont récupérées à partir de Bluetooth_com.dll.

En plus de ces pages, on retrouve la classe Affichage, codée en `C#`. Celle-ci permet de créer des grilles de pièces ou d'équipements.

2 Diagrammes UML

2.1 Diagramme de classe de l'interface graphique

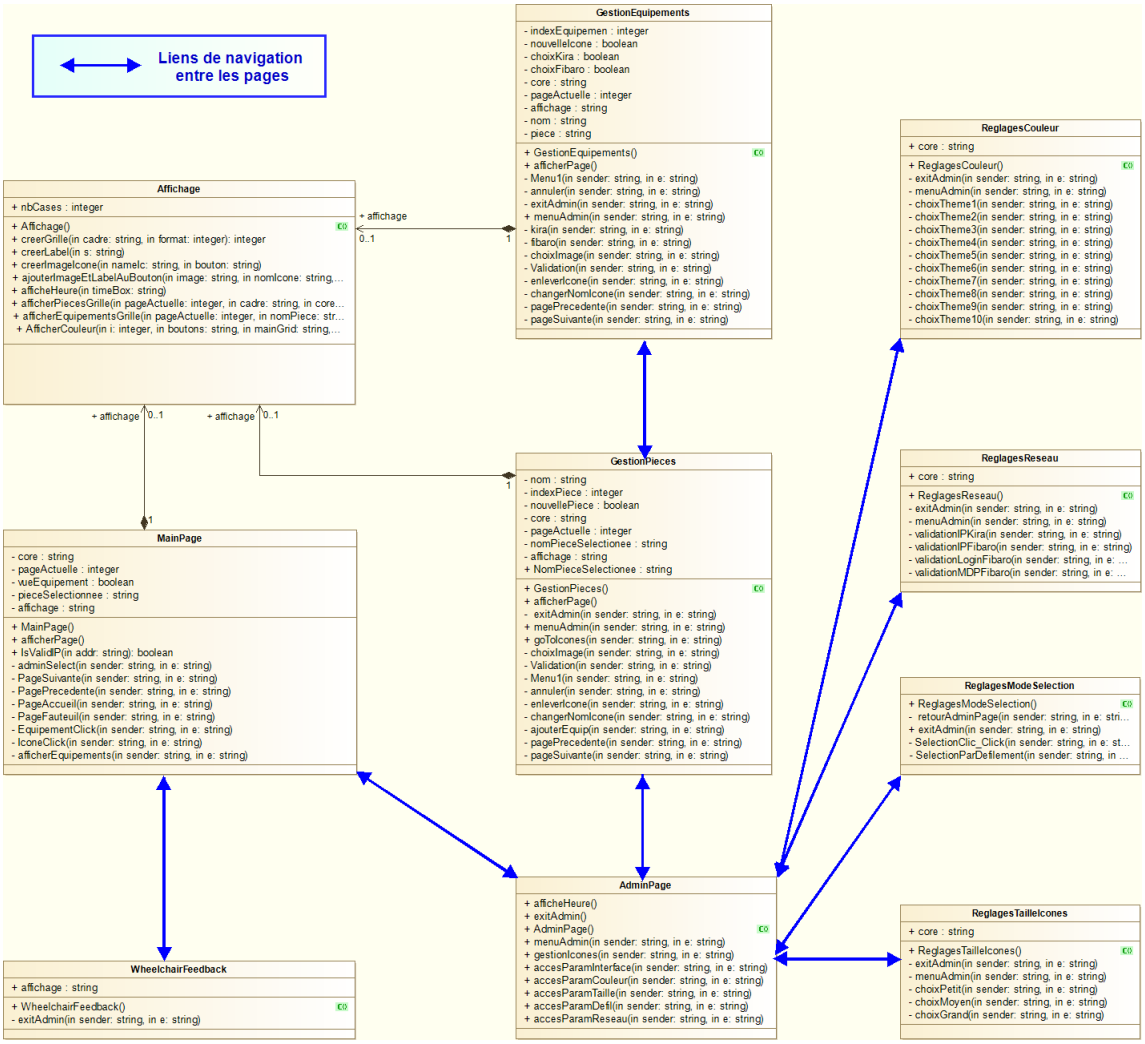

FIGURE 1 – Diagramme de classe de l'interface graphique

2.2 Diagramme de classe du modèle

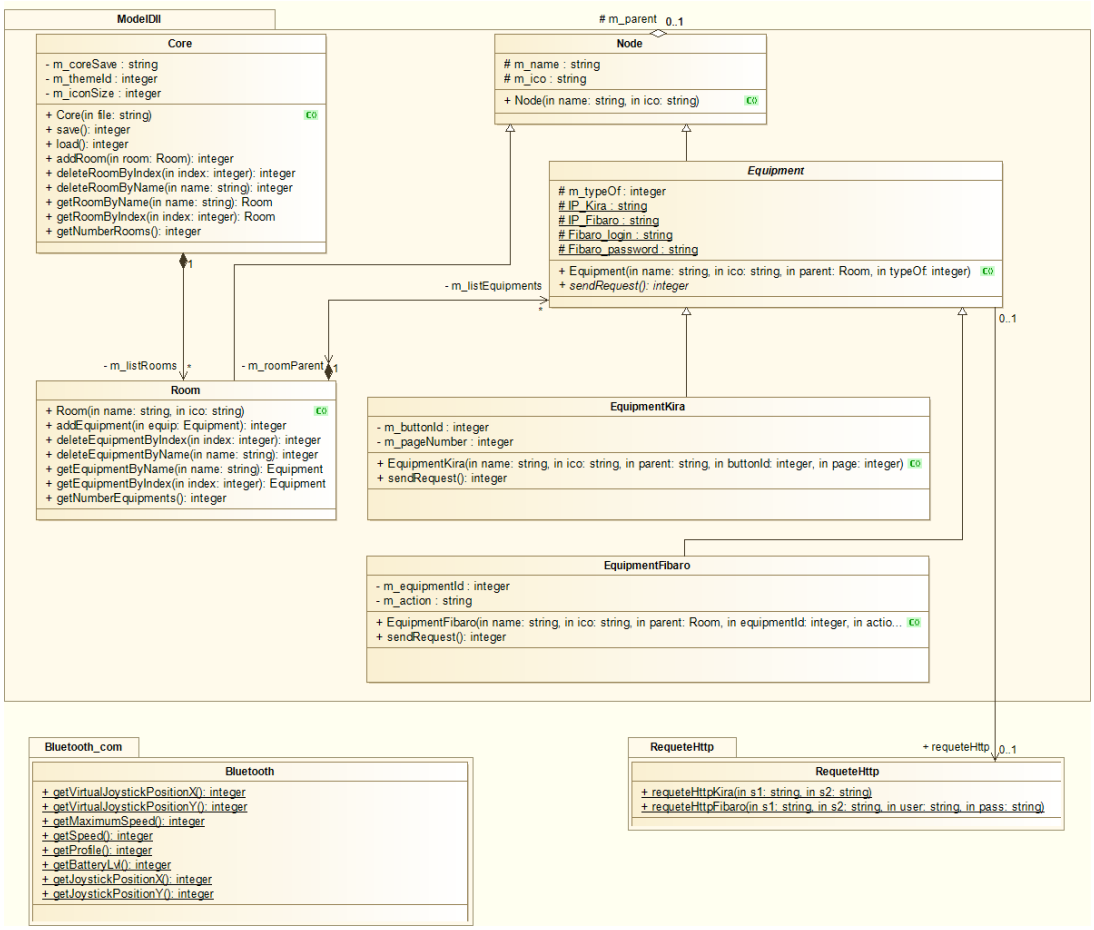

FIGURE 2 – Diagramme de classe du modèle

2.3 Diagramme de classe du modèle et de l'interface graphique

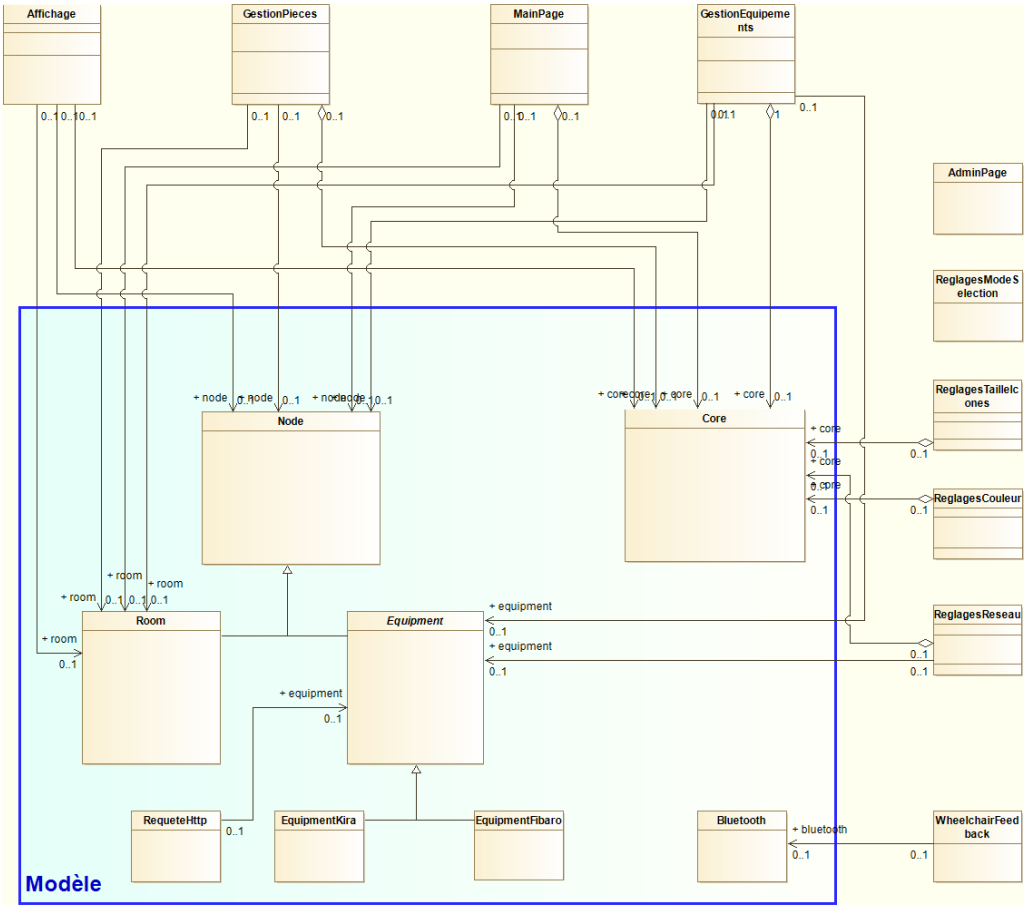

FIGURE 3 – Diagramme de classe du modèle et de l'interface graphique

3 Documentation

Voir Vue_Controleur.pdf, Dll_Modele.pdf Dll_Requete_Http.pdf, Bluetooth_com.pdf.

4 Installation

Lancer le script PowerShell Add-AppDevPackage.ps1 situé dans DomoticApp/AppPackages/DomoticApp_1.0.0.0_Test.