CHAPITRE 4

CONCEPTION

Ce chapitre décrit la conception d'une architecture de base dont le but tend vers les diverses spécifications énumérées dans la section 3.2. Étant donné que toutes les spécifications n'ont pu être implémentées, ce chapitre décrit d'abord les spécifications fonctionnelles qui ont été développées. Par la suite, l'architecture générale de QMA est suite présentée puis, chaques composantes de cette architecture sont décrites avec plus de précision. Ensuite, des diagrammes de séquences sont présentés afin de montrer l'interaction entre les diverses composantes lorsque l'utilisateur exécute des opérations particulières à la QdS. Finalement, comme QMA a été conçu, jusqu'à ce jour, pour interroger des routeurs Cisco uniquement, la dernière section du chapitre présente une approche pouvant être utilisée pour que QMA puisse interragir avec des équipements d'autres équipementiers que Cisco Systems.

4.1 Spécifications mises en oeuvre

Cette section énumère les spécifications fonctionnelles qui ont été implémentées dans QMA. QMA permet de visualiser la topologie du réseau et d'accéder à des fonctionnalités générales à l'ensemble du réseau et à d'autres plus spécifiques à un équipement précis. Par exemple, les options de QdS sont accessibles à l'utilisateur par le biais d'un menu contextuel obtenu lorsque l'utilisateur sélectionne un équipement spécifique.

Concernant les spécifications fonctionnelles, seules les fonctionnalités de visualisation des configurations et des statistiques des mécanismes de QdS ont été réalisées. Par conséquent, les fonctionnalités de configuration semi-automatique, de génération d'alertes et de génération de rapports de performance n'ont pu être réalisées et ont été

laissées pour le développement futur. Finalement, plus de la moitié des spécifications non-fonctionnelles énumérées dans la section 3.2.3 ont été atteintes.

4.2 Architecture générale

Cette section présente l'architecture générale de QMA (voir Figure 14). Elle commence tout d'abord par présenter brièvement le rôle des différentes composantes ainsi que l'interaction entre chacune d'elle. Cependant le choix des plateformes technologiques utilisées n'est pas abordé dans ce document puisqu'il a été expliqué plus en détail dans [34]. Par conséquent, ce mémoire prend pour acquis que la plateforme .Net a été utilisée pour réaliser les diverses composantes de l'architecture. De plus, pour les raisons évoquées dans la section 2.3, le protocole SNMP a été choisi pour l'interaction entre les composantes de l'architecture et les équipements réseaux.

Dans la Figure 14, on peut voir que *ModulesQMA* a deux interfaces : une pour interagir avec le réseau, l'autre avec la base de données. En fait, son rôle principal est d'interroger les équipements réseau à l'aide du protocole SNMP et de remplir la base de données en conséquence. Bien que ces deux fonctions soient ses principales, *ModulesQMA* est également utilisé à d'autres fins comme l'authentification, la gestion de la surveillance des tunnels MPLS, la gestion des Trap SNMP, etc. Il est généralement invoqué par le service Web *WS_QMA* ou par le contrôleur de serveur (*Server Controller*) et ce, de façon sporadique.

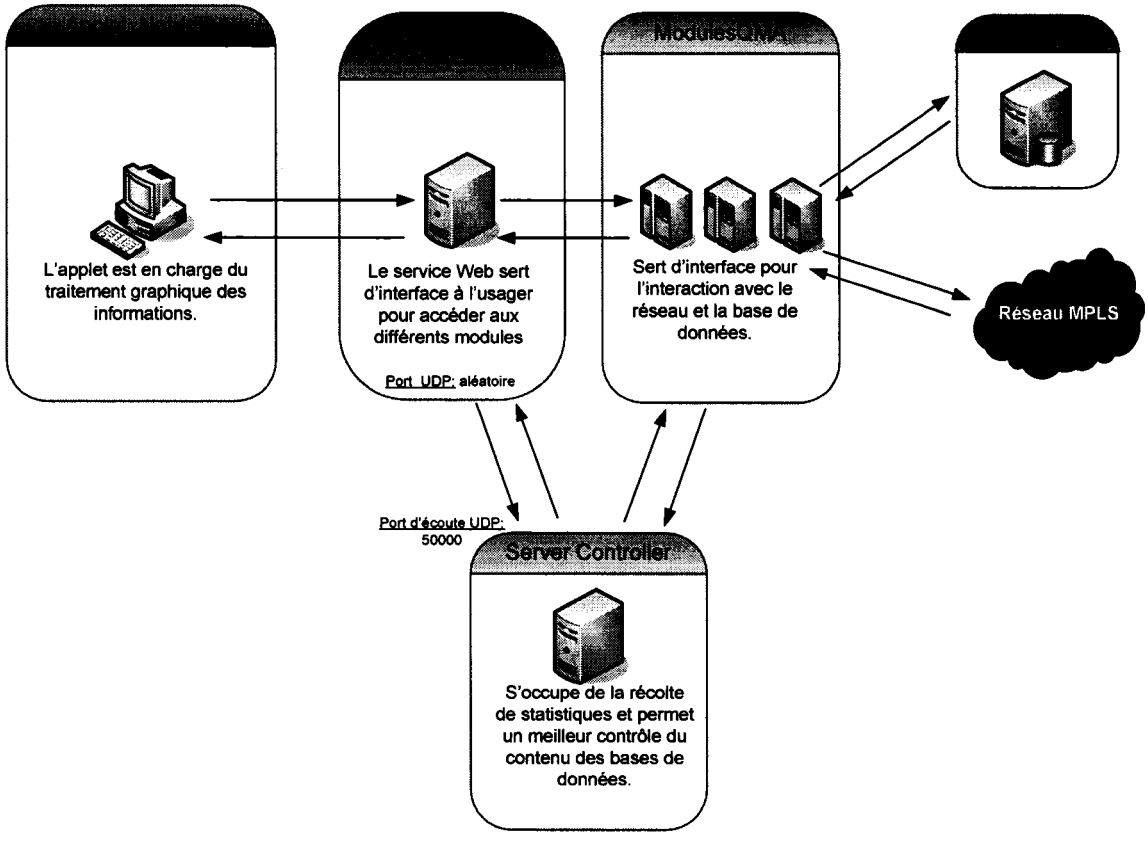

Figure 14 Architecture générale de QMA

La composante *Server Controller* est quant à elle utilisée afin de contrôler la base de données. Cette composante offre une interface utilisateur permettant de vider et/ou remplir une ou des tables de la base de données d'un équipement spécifique. Cette composante interagit avec *ModulesQMA* pour demander d'interroger le réseau et de remplir la base de données en conséquence ou tout simplement pour obtenir certaines informations de cette dernière. En plus de contrôler la base de données de configuration, cette composante est utilisée afin de récolter les statistiques des mécanismes de QdS. Par conséquent, *Server Controller* doit être créé qu'une seule fois et rester actif indéfiniment ou jusqu'à ce qu'un usager le termine. Les fonctionnalités de cette composante peuvent être invoquées soit par l'interface utilisateur ou encore par le biais du service Web.

L'applet constitue l'interface utilisateur typique du client et est en charge du traitement graphique des informations. Il communique avec le service Web (*WS_QMA*) de deux différentes façons. Il peut envoyer des commandes à *ModulesQMA* pour interroger les équipements réseau ou au contrôleur de serveur pour obtenir une configuration ou encore pour démarrer la collecte de statistiques. Ces commandes peuvent être lancées par l'utilisateur par le biais de boutons et/ou de menus. Par ailleurs, il peut communiquer avec le service Web pour obtenir des informations contenues dans la base de données. Ces informations peuvent dans certains cas subir un traitement par l'applet avant d'être affichées à l'utilisateur dans des *comboBox*, *listBox*, *textBox*, graphique ou autres composantes visuelles.

Finalement, le service Web *WS_QMA* est utilisé comme interface entre l'utilisateur de l'applet et le reste du système. Par conséquent, il peut invoquer des méthodes disponibles dans *ModulesQMA* pour interagir avec le réseau et/ou la base de données lorsque requis. Similairement, il communique avec le contrôleur de serveur afin d'obtenir une configuration de QdS ou encore afin d'ajouter et/ou de supprimer une instance d'un objet chargé de récolter les statistiques des mécanismes de QdS.

4.3 Descriptions des composantes de l'architecture générale

Cette section présente les diverses composantes du système QMA, présentées à la Figure 14, mais de façon plus détaillé. Elles seront en fait décrites par le biais de diagrammes de classes. Avant de commencer la lecture de cette section, il est recommandé d'aller voir l'ANNEXE 1 pour connaître les divers OIDs utilisés pour obtenir les configurations et les statistiques de QdS sur les équipements Cisco. Étant donné qu'il n'existe pas actuellement de MIB standard pour l'obtention de telles informations, QMA a donc dû être développé de façon à s'adapter à toutes les MIBs propriétaires. Étant donné que l'équipementier Cisco System est le plus gros vendeur mondial d'équipements IP, le système QMA a d'abord été développé pour supporter la MIB de cet équipementier.

L'ANNEXE 2 présente, quant à elle, la base de données de configurations et de statistiques de QdS. Cette base de données est principalement adaptée à la MIB de Cisco. Cependant, l'intégration d'autres équipementiers sera possible soit en adaptant la base de données afin de la rendre plus générique ou encore en créant une base de données différente dépendamment de l'équipementier utilisé. Pour plus de détails référez vous à la section 4.5. Ainsi l'aspect multi-vendeur de QMA est assuré.

4.3.1 ModulesQMA

Cette section présente les principales classes de *ModulesQMA* dont les classes de base *Snmp* et *Database* ainsi que deux classes de contrôle soit *Control_DB* et *Control_Auth*.

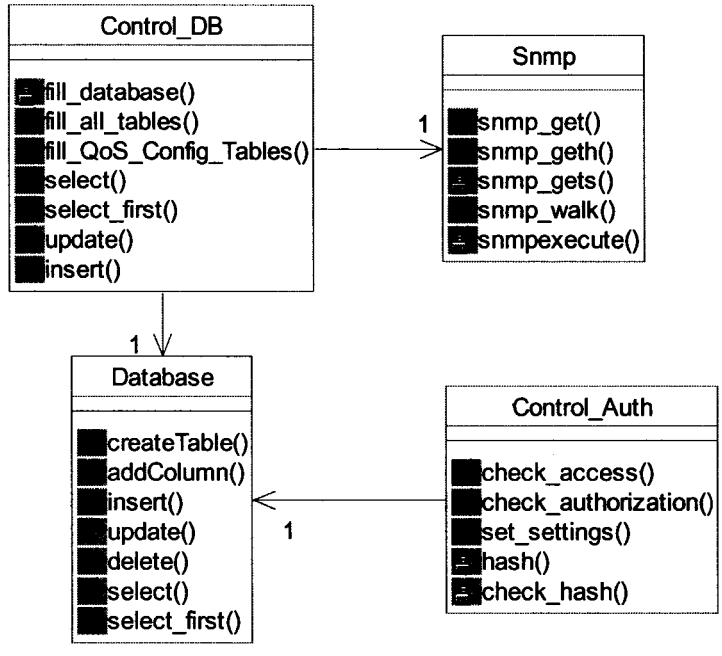

Figure 15 Diagramme de classes de ModulesQMA

4.3.1.1 Classe Snmp

Avant de comprendre le rôle des diverses méthodes, il est essentiel de savoir que la librairie Net-Snmp (http://net-snmp.sourceforge.net/) est utilisée afin d'effectuer les requêtes SNMP vers les équipements. Notez que les fonctions utilisées par cette librairie nous retournent les réponses en format *string* avec la structure qui suit :

OID.Feuille = Type: Valeur

Dans cette structure, *OID* représente l'objet demandé tandis que *Feuille* représente l'instance de l'objet demandé. Par la suite, *Type* correspond au type de la valeur retournée qui est représentée ici par *Valeur*.

La Figure 16 présente le détail de la classe *Snmp* qui ne sert en fait que d'interface d'accès à la librairie Net-Snmp. Par ailleurs, les diverses méthodes de la classe sont décrites ci-dessous.

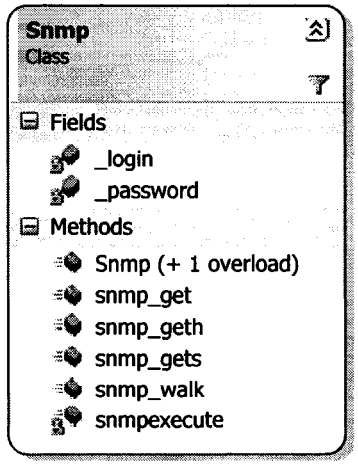

Figure 16 Détail de la classe Snmp de ModulesQMA

Méthode snmp_get

Cette méthode utilise la fonctionnalité GET de SNMP afin d'obtenir la valeur d'un seul *Object ID* (OID). La particularité de cette méthode consiste à ne retourner que la valeur de l'OID. De façon plus précise, elle retourne la valeur située à la fin du *string*, c'est-à-dire après le *Type:*. Dans l'exemple ci-dessous, l'équipement est interrogé pour connaître le nombre d'interfaces présentes dans l'équipement. En utilisant cette méthode, seule la valeur 10 sera retournée.

IF-MIB::ifNumber.0 = INTEGER: 10

Méthode snmp_gets

Cette méthode fonctionne de façon similaire à la précédente. Sa particularité consiste à traiter les OID dont la valeur de retour est de type *STRING*. En considérant l'exemple ci-dessous, cette méthode supprime les guillemets « " » pour ne retourner que la chaîne de caractère *out-ets1-parent*.

enterprises.9.9.166.1.6.1.1.1.1129 = STRING: "out-ets1-parent"

Méthode snmp_geth

Cette méthode fonctionne de façon similaire aux précédentes. Sa particularité consiste à traiter les OID dont la valeur de retour est de type *Hex-STRING*. Cette méthode a été créée spécifiquement pour convertir une chaîne hexadécimale en chaîne de caractère correspondant à une adresse IP. Par exemple, dans le cas où la valeur de retour est « AC 15 FD 0E », cette chaîne est reconvertie en adresse IP « 172.21.253.14 ».

Méthode snmp_walk

Cette méthode utilise la fonctionnalité BULKWALK de SNMP. En considérant que la MIB est structurée en arbre, cette fonctionnalité consiste à spécifier un noeud de l'arbre (ou OID) auquel toute la structure sous-jacente sera retournée. Cette méthode est

pratique dans le cas où un OID a plusieurs feuilles qui s'y rattache. Par exemple, si un usager désire obtenir la liste des noms d'interfaces, il peut demander l'OID *ifDescr* (ou 1.3.6.1.2.1.2.2.1.2) qui retournera ifDescr.ifIndex pour tous les index d'interface (ifIndex). Dans une telle situation, il est donc utile d'obtenir tout le retour de la fonction SNMP-BULKWALK et non seulement la liste des valeurs. Il devient alors plus simple, pour un programmeur, d'associer un index d'interface avec son nom.

Méthode snmp_execute

Cette méthode est une méthode générique qui prend comme paramètre le nom de la fonction SNMP à appeler. Ainsi elle peut être utilisée dans plusieurs situations, comme par exemple par les méthodes snmp_walk, snmp_get, etc. Elle retourne un *string* contenant la totalité de la réponse obtenue.

4.3.1.2 Classe Database

Cette classe est utilisée pour interagir avec la base de données. Elle comprend des méthodes lui permettant de manipuler la base de données de toute sorte de façon. La Figure 17 présente le détail de cette classe tandis que la liste des méthodes ainsi que leurs descriptions sont résumées dans le Tableau X.

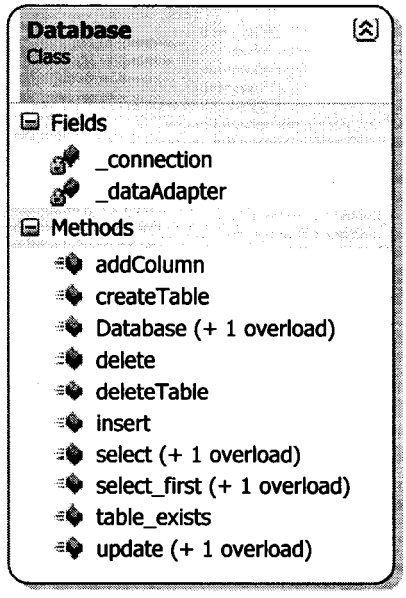

Figure 17 Détail de la classe Database de ModulesQMA

Tableau X

Description des méthodes de la classe Database dans ModulesQMA

Méthodes	**Descriptions**
createTable	Utilisée pour créer une nouvelle table avec les colonnes spécifiées dans les paramètres de la méthode.
deleteTable	Utilisée pour supprimer une table de la base de données.
table_exists	Retourne un booléen indiquant si la table spécifiée existe ou non.
addColumn	Ajoute une ou des colonnes à une table.
insert	Insère une nouvelle entrée dans une table.
update	Modifie une ou plusieurs entrées d'une table selon les conditions spécifiées dans les paramètres de la méthode.
delete	Supprime une ou plusieurs entrées d'une table selon les conditions spécifiées dans les paramètres de la méthode.

Tableau X (suite)

Méthodes	Descriptions
select	Sélectionne une ou plusieurs entrées d'une table suivant les conditions spécifiées dans les paramètres de la méthode. Cette méthode retourne une structure de donnée de type DataSet.
select_first	Retourne la première occurrence des critères de sélection spécifiés dans les paramètres de la méthode.

4.3.1.3 Control_DB

Cette classe est utilisée pour remplir la base de données selon les informations récoltées des équipements. Elle utilise principalement la classe *Snmp* pour interroger les équipements, puis la classe *Database* pour interagir avec la base de données. Elle comporte une foule de méthodes pour remplir ou vider les tables de façon individuelle ou en groupe. Toutes les méthodes (58 au total) n'ont pas été insérées dans le diagramme de classe dans le but de l'alléger. Il est cependant important de noter que la table Router doit être remplie au préalable avant d'utiliser les méthodes de remplissage des tables relatives à la QdS. Certaines méthodes ont été définies pour remplir et vider la table Router (*fill_Router_Table* et *empty_Router_Table*). Elles sont accessibles à un usager via deux boutons tel que présenté dans l'ANNEXE 3. De plus, la classe *Control_DB* possède certaines méthodes utilisées pour manipuler la base de données. Elles possèdent le même nom et syntaxe que les méthodes de la classe *Database* et sont utilisées comme interface à cette dernière classe. En fait, ces méthodes on été créées afin qu'un objet qui possède une instance de *Control_DB* n'ait pas besoin d'instance de *Database* pour obtenir ou modifier des informations de la base de données. La Figure 18 présente le détail de la classe tandis que les principales méthodes de *Control_DB* sont résumées dans le Tableau XI.

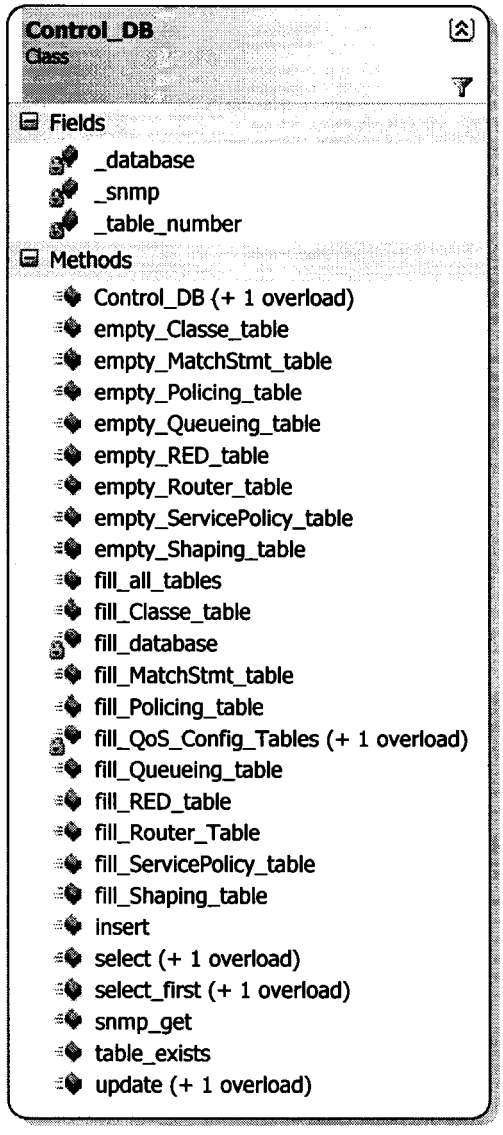

Figure 18 Détail de la classe Control_DB de ModulesQMA

Tableau XI

Description des méthodes de la classe Control_DB dans ModulesQMA

Méthodes	Descriptions
fill_database	Utilisée pour remplir toute les tables relatives à MPLS et à la QdS. Elle appelle entre autre la méthode fill_QoS_Config_Tables.
fill_QoS_Config_Tables	Remplit, pour un ou tous les routeurs, toutes les tables relatives aux configurations de la QdS. Ces tables sont présentées à l'ANNEXE 2 et doivent être remplit dans l'ordre suivant : ServicePolicy, Class, MatchStmt, Queueing, Shaping, Policing puis RED.
select	Appelle la méthode *select* de la classe *Database.*
select_first	Appelle la méthode *select_first* de la classe *Database.*
update	Appelle la méthode *update* de la classe *Database.*
insert	Appelle la méthode *insert* de la classe *Database.*

4.3.1.4 Control_Auth

La Figure 19 présente le détail de la classe *Control_Auth*. Cette classe a deux principaux rôles. Elle est utilisée, dans un premier temps, afin de valider si un usager est autorisé à faire quelque opération que ce soit dans le système ou non. Dans un deuxième temps, lorsque l'usager a été authentifié, cette méthode retourne des informations générales au bon fonctionnement du système. Ces informations sont composées comme suit :

a. Nom du serveur SQL
b. Nom de la base de données de configuration
c. Nom de la base de données de statistiques
d. Nom d'usager et mot de passe de la base de données
e. Nom d'usager et mot de passe SNMP

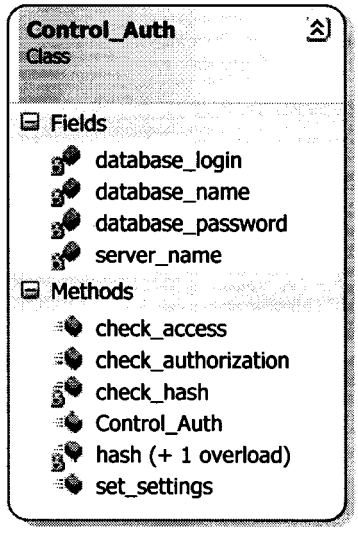

Figure 19 Détail de la classe Control_Auth de ModulesQMA

Le Tableau XII présente une description des méthodes de cette classe.

Tableau XII

Description des méthodes de la classe Control_Auth dans ModulesQMA

Méthodes	Descriptions
check_access	Retourne un booléen indiquant si l'usager est autorisé ou non dans le système.
check_authorization	Une fois que l'usager a été authentifié, cette méthode est utilisée pour retourner les informations générales relatives à la base de données utilisée ainsi qu'aux configurations SNMP. Notez que l'utilisateur peut spécifier les informations qu'il désire recevoir.
set_settings	Configure les informations générales pour un utilisateur spécifique.
hash	Cette méthode prend en paramètre le texte à hacher et retourne une chaîne de caractère contenant le texte haché.

Tableau XII (suite)

Méthodes	Descriptions
check_hash	Cette méthode prend en paramètre le mot de passe non haché (donné par l'usager) ainsi que le mot de passe haché (relevé de la base de données) et retourne un booléen indiquant s'ils correspondent. Cette méthode est principalement utilisée afin de savoir si un usager a entré le bon mot de passe.

4.3.2 Applet

Cette section présente une description de l'implémentation de l'applet, qui est l'interface utilisateur de QMA. La Figure 20 présente un diagramme de classe qui illustre l'interaction entre les diverses classes de l'applet. Dans ce diagramme, seules les classes relatives à la QdS sont affichées. De plus, aucun attribut ni aucune méthode ne sont affichés afin d'alléger la complexité de ce diagramme. Notez cependant que chacune de ces classes sera discutée plus en détail dans les sections qui suivent.

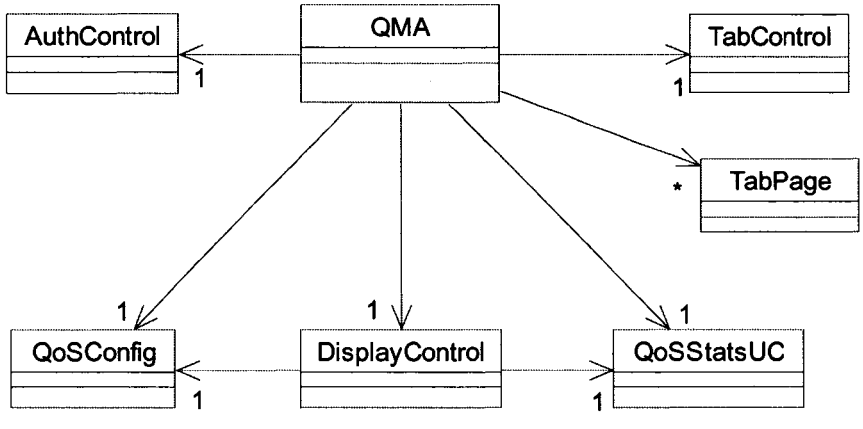

Figure 20 Diagramme de classe de l'applet QMA

Dans la Figure 20, on peut voir que la classe *QMA* est la classe principale. Elle possède les instances des autres classes et possède, du même coup, un certain contrôle sur ces dernières. Dans ce diagramme, les classes *QMA*, *DisplayControl*, *AuthControl*, *QoSConfig* et *QoSStatsUC* sont toutes des contrôles utilisateur (*User Control*). C'est-à-dire qu'elles sont « des blocs de code réutilisables dont le but cible un interface utilisateur »[35]. Tous ces contrôles seront décris dans les lignes qui suivent.

4.3.2.1 Le contrôle utilisateur QMA

Tel que spécifié précédemment, le contrôle utilisateur *QMA* est une classe principale qui détient les instances des autres contrôles utilisateurs. La Figure 21 illustre la composition de cette classe. Notez que seul ce qui a trait à la QdS y a été affiché, le reste étant caché afin de ne pas nuire à la compréhension.

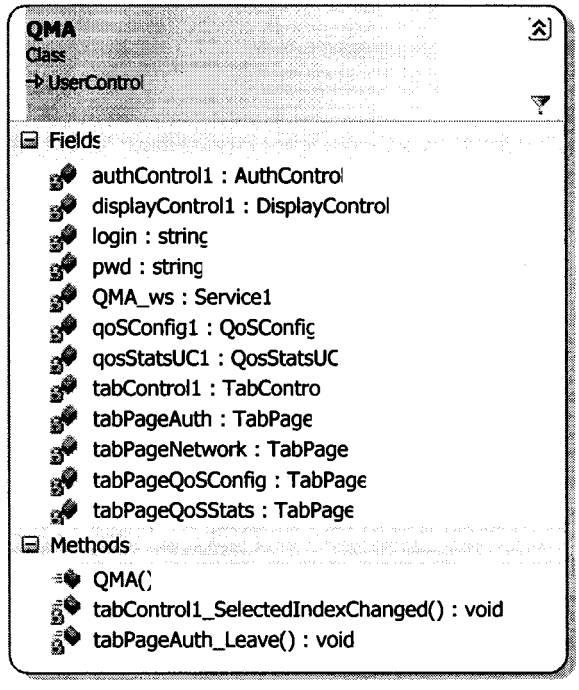

Figure 21 Détails du contrôle utilisateur QMA

On peut voir dans cette figure que le nombre de méthodes et d'attributs est plutôt restreint puisque la majorité du traitement se fait dans les autres contrôles utilisateur. Le contrôle utilisateur QMA est plutôt utilisé afin d'assembler tous les contrôles utilisateurs dans une seule fenêtre. Ce contrôle utilisateur possède la référence au service Web qu'il référera aux autres contrôles utilisateurs qui le nécessite.

Afin de pouvoir afficher plusieurs contrôles utilisateurs dans une seule fenêtre, des onglets ont été réalisés. Par conséquent, *tabControl1* est utilisé afin de gérer les différents onglets tandis que *tabPageAuth*, *tabPageNetwork*, *tabPageQoSConfig* et *tabPageQoSStats* constituent en fait les différents onglets. Il suffit d'insérer les contrôles utilisateur *authControl1*, *displayCotnrol1*, *qoSConfig1* et *qosStatsUC1* dans chaque onglet respectif.

Dans la liste des méthodes on voit bien sûr le constructeur de la classe. La méthode *tabControl1_SelectedIndexChanged* est appelée lorsque l'utilisateur tente de changer d'onglet. Son nom d'utilisateur et son mot de passe sont alors vérifiés afin de s'assurer qu'il s'agit bien d'un utilisateur autorisé. Si ce n'est pas le cas, l'onglet retourne à l'onglet d'authentification. Lorsque l'usager est dans l'onglet d'authentification et clique sur un autre onglet, la méthode *tabPageAuth_Leave* est appelée. Cette méthode sert à valider les paramètres d'authentification entrés par l'usager en appelant une méthode du service Web. Une fois validés, les paramètres d'authentification sont alors configurés dans les autres contrôles utilisateurs qui les nécessitent.

4.3.2.2 Le contrôle utilisateur AuthControl

Le contrôle utilisateur présenté dans cette section est utilisé pour qu'un usager puisse entrer ses paramètres d'authentification. C'est-à-dire qu'il doit entrer son nom d'usager et son mot de passe dans *login_textBox* et *password_textBox* respectivement. S'il désire supprimer le contenu de ces deux *textBox*, l'utilisateur n'a qu'à appuyer sur *button1* afin

que la méthode *button1_Click* soit appelée pour exécuter cette tâche. Quant aux méthodes *get_login_textBox* et *get_password_textbox*, elles sont utilisée par le contrôle utilisateur *QMA* afin qu'il puisse aller chercher le contenu de *login_textBox* et *password_textBox* respectivement (voir Figure 22).

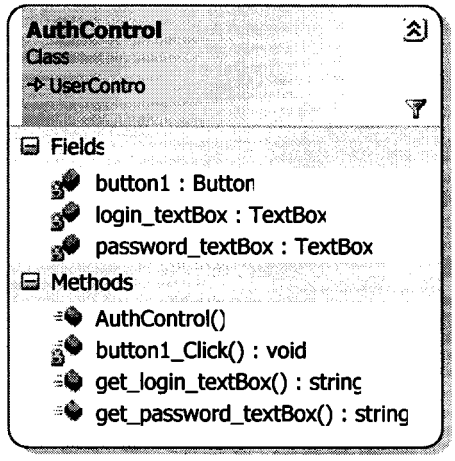

Figure 22 Détails du contrôle utilisateur AuthControl

4.3.2.3 Le contrôle utilisateur DisplayControl

Le contrôle utilisateur *DisplayControl* est celui utilisé pour visualiser les éléments du réseau et pour passer des commandes. Il est cependant important de mentionner que le schéma du réseau ne permet présentement pas d'afficher une grande quantité d'équipements et que du travail reste à faire afin de pouvoir afficher la totalité du réseau d'un fournisseur de service. Veuillez prendre note qu'afin de rendre la figure lisible, la Figure 23 présente une vue sommaire de la classe et que les méthodes et attributs présentés correspondent à ceux liés à la QdS. Dans un même ordre d'idée, les arguments des méthodes ont été omis. Par exemple, les composantes graphiques utilisées pour représenter le réseau (*GraphRouter*) ne sont pas affichés.

Comme *DisplayControl* doit interagir avec le service Web, une référence à celui-ci a été insérée par le contrôle utilisateur *QMA*. Ensuite, celui-ci a également configuré les noms d'usager (*_login*) et mot de passe (*_password*) requis, et ce par le biais des méthodes *set_login* et *set_password*. De plus, *DisplayControl* doit posséder une référence vers certains objets du contrôle utilisateur *QMA* afin de pouvoir interagir sur eux lors de la sélection de certaines options (*qosConfigObject* et *qosStatsObject*), ou encore afin d'imposer un changement d'onglet (*tabControlQMA*, *qosConfigPage* et *qosStatsPage*). Par conséquent la méthode *setQoSObjects* a été créée de sorte que le contrôle utilisateur QMA puisse fournir la référence vers ces instances à *DisplayControl*.

Étant donné que les options de QdS sont accessibles par le biais d'un menu contextuel obtenu en cliquant avec le bouton droit sur un objet *GraphRouter*, l'objet *contextMenuRouter1* a été créé. Ce menu contextuel détient plusieurs items de menu, dont les plus importants pour la QdS sont :

a. menuItemShowPolicy : correspond à l'option de QdS *Show Service Policy*.
b. menuItemRenewQos : correspond à l'option de QdS *ReNew QoS Configurations*.
c. menuItemQosStats : correspond à l'option de QdS *Begin Statistic Collection*.
d. menuItemTelnet : correspond à l'option *Start Telnet*.

Lorsqu'un des *menuItem* est sélectionné, la méthode *menuItemX_Click* correspondante est appelée. Lorsqu'un des *menuItem* correspondant à une option de QdS est sélectionné trois opérations générales à ces options sont réalisées. D'abord le nom du routeur sélectionné est conservé, dans l'attribut *selectedRouter*, pour un usage futur. Par la suite l'attribut *qosSelected* est mis à *true* afin de signaler qu'une option de QdS est sélectionnée puis l'option sélectionnée est conservée temporairement dans l'attribut *qosOptionSelected*.

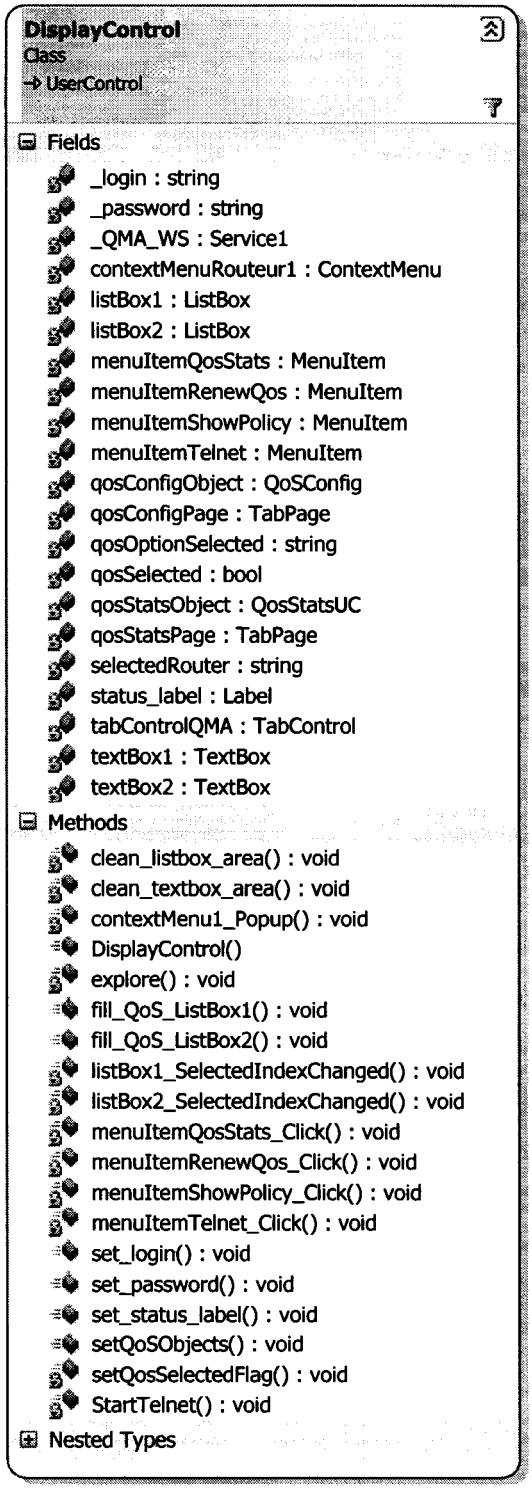

Figure 23 Détail du contrôle utilisateur DisplayControl

Lorsque l'une des options *Show Service Policy* ou *Begin Statistic Collection* est sélectionnée, les méthodes *clean_listbox_area* et *clean_textbox_area* sont appelées afin de vider le contenu des *listBox* et *textBox* respectivement. Par la suite la méthode *fill_QoS_ListBox1* est appelée afin de lister, dans le *listBox1*, les interfaces du routeur sélectionné (*selectedRouter*) sur lesquelles une politique de service est appliquée. Lorsqu'un usager sélectionne alors une interface, la méthode *listBox1_SelectedIndexChanged* est appelée afin qu'elle puisse elle-même lancer la méthode *fill_QoS_ListBox2*. Ainsi le *listBox2* dresse la liste des politiques de service présentes sur l'interface sélectionnée. Une fois que l'usager sélectionne une de ces politiques, la méthode *listBox2_SelectedIndexChanged* est lancée. Cette méthode vérifie d'abord si une option de QdS a été sélectionnée (*qosSelected*). Si tel est le cas, la méthode vérifie quel type d'option a été sélectionné (*qosOptionSelected*) afin d'appeler la méthode correspondante du service Web. Dans le cas de l'option *Show Service Policy*, une réponse, contenant la configuration demandée, est attendue du service Web. *DisplayControl* appelle alors la méthode *addConfig* de l'objet *qosConfigObject* puis change d'onglet afin de diriger l'utilisateur directement à l'onglet *qosConfigPage*. Dans le cas de l'option *Begin Statistic Collection*, seule la méthode *addStats*, de l'objet *qosStatsObject*, a été appelée. Par la suite, l'utilisateur est redirigé vers l'onglet *qosStatsPage*.

Dans le cas où l'utilisateur aurait choisi l'option *ReNew QoS Configurations*, une méthode correspondante est appelée sur le service Web.

Finalement, lorsque l'utilisateur sélectionne l'option *Start Telnet*, le service Web est interrogé afin de retrouver l'adresse IP du routeur sélectionné. Par la suite, la méthode *StartTelnet* est appelée. Cette méthode prend en paramètre l'adresse IP à laquelle la session Telnet doit être établie. Pour terminer, l'application Telnet est lancée à l'adresse IP spécifiée.

4.3.2.4 Le contrôle utilisateur QoSConfig

Le contrôle utilisateur présenté dans cette section (Figure 24), permet de lister et d'afficher les configurations demandées par l'utilisateur. L'un des principaux attributs de cette classe est *configList* qui est en fait une liste triée contenant la liste des configurations, en format texte, que l'utilisateur a demandé de visualiser. Ces configurations sont indexées par une chaîne de caractères identifiant, de façon unique, une politique de service. En fait, la chaîne de caractère est structurée comme suit : *NomRouteur-NomInterface-Direction-NomDeLaPolitique*. Le *comboBox1* dresse la liste des index de la liste triée. Lorsque l'utilisateur sélectionne un de ces éléments de la liste, la méthode *comboBox1_SelectedIndexChanged* est appelée, laquelle retrouve la configuration correspondante de la liste et l'affiche dans le *textBox1*.

L'utilisateur peut supprimer les configurations de la liste à sa guise en utilisant le bouton *buttonRemove* qui appelle la méthode *buttonRemove_Click* qui supprime alors l'entrée correspondante de la liste triée et du *comboBox1*.

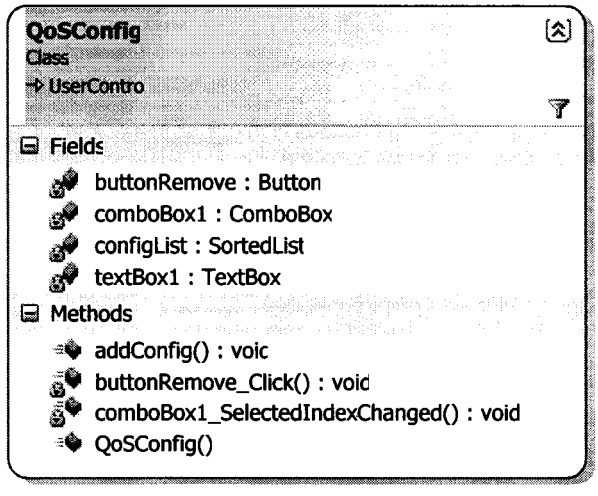

Figure 24 Détail du contrôle utilisateur QoSConfig

4.3.2.5 Le contrôle utilisateur QoSStatsUC

Le principal but de ce contrôle utilisateur est d'afficher les statistiques de QdS à l'usager. Le détail de sa classe est présenté à la Figure 25 à l'exception de certains *Label* qui ont été omis afin d'alléger la figure sans toutefois compromettre la compréhension de la classe. Comme ce contrôle doit interroger la base de données régulièrement, la référence au service Web *QMA_ws* a été insérée et configurée par le contrôle utilisateur *QMA*. Par ailleurs, afin que le service Web puisse authentifier l'utilisateur, le contrôle utilisateur *QoSStatsUC* doit avoir connaissance du nom d'usager (*login*) et du mot de passe (*password*) qui sont tous deux configurés par le contrôle utilisateur *QMA* via les méthodes *setLogin* et *setPassword*.

Tel que vu dans la description du contrôle utilisateur *DisplayControl*, la méthode *AddStats* est utilisée pour ajouter une politique de service à la liste des politiques surveillées si elle n'y est pas déjà. Cette méthode appelle la méthode *addQoSStats* du service Web.

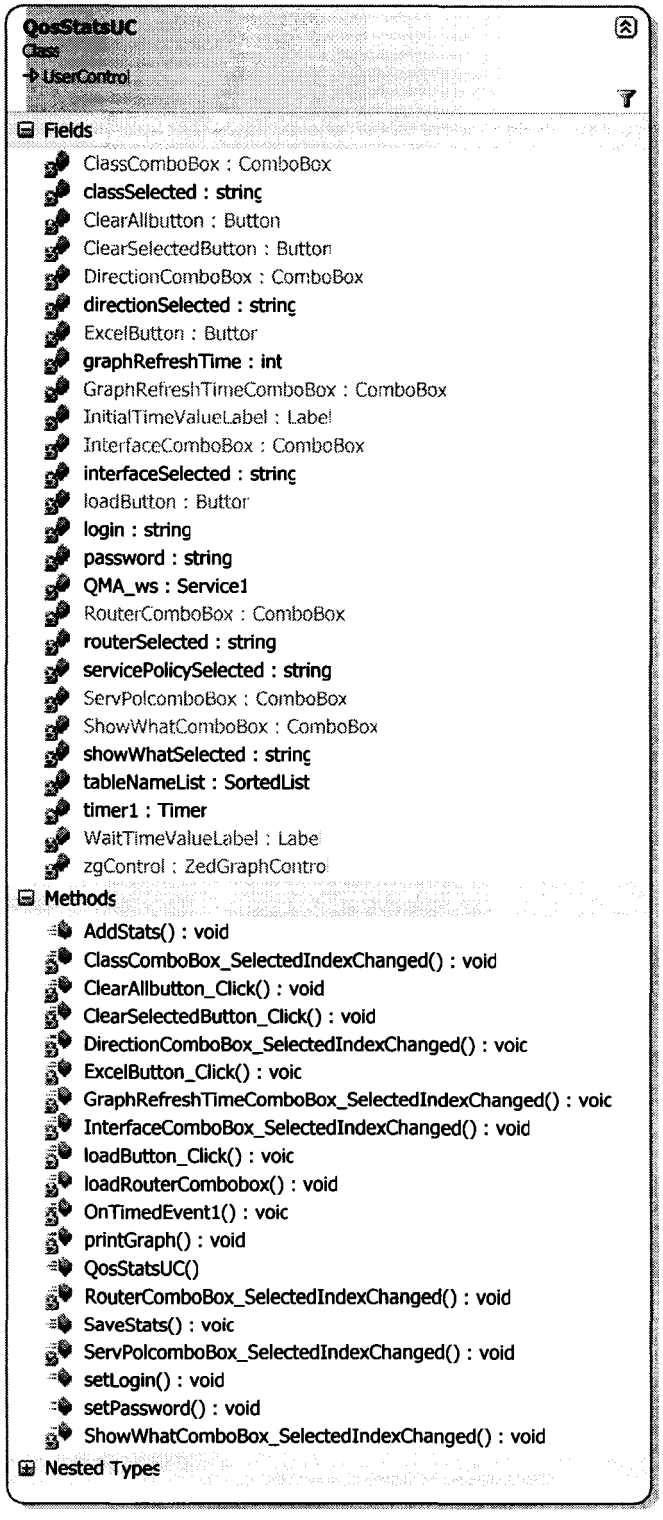

Figure 25 Détail du contrôle utilisateur QosStatsUC

Les *comboBox*

Plusieurs *comboBox* ont été insérés afin de permettre à l'utilisateur de sélectionner les statistiques à visualiser. C'est *comboBox* sont listés et décrits brièvement ci-dessous.

a. routerComboBox :
   - Permet la sélection de l'équipement sur lequel la politique désirée est appliquée.

b. InterfaceComboBox :
   - Permet la sélection de l'interface de l'équipement sur laquelle la politique de service désirée est appliquée.

c. DirectionComboBox :
   - Permet de sélectionner la direction de l'interface et du routeur sélectionné sur laquelle la politique de service désirée est appliquée.

d. ServPolcomboBox :
   - Permet de sélectionner la politique de service désirée.

e. ClassComboBox :
   - Permet de sélectionner la classe de la politique de service dont l'usager désire voir des statistiques. Notez que le nom de la classe représenté dans ce *comboBox* a été structuré en insérant le nom de la politique et le nom de la classe, tous deux séparés d'un « _ ». Cette structure a été adoptée afin de différencier les classe pouvant porter le même nom dans une politique parent et enfant.

f. ShowWhatComboBox :
   - Permet de sélectionner les statistiques de la classe que l'usager désire visualiser.

Les objets des *comboBox* doivent être sélectionnés dans la séquence suivante :

Router→Interface→Direction→Politique de service→Classe→ShowWhat

Lorsqu'un un objet est sélectionné dans l'une ou l'autre des listes précédentes, la méthode *xyzComboBox_SelectedIndexChanged* correspondante est appelée. Par conséquent, lorsqu'un objet est sélectionné dans un *comboBox*, la méthode *xyzComboBox_SelectedIndexChanged* correspondante est utilisée pour remplir le *comboBox* suivant de la séquence à l'exception de *ShowWhatComboBox*. En effet lorsqu'un objet de la liste de ce *comboBox* est sélectionné, la méthode *printGraph* est appelée afin d'afficher les statistiques correspondantes.

Étant donné qu'une table est créée dans la base de données pour chacune des classes, la liste triée *tableNameList* conserve la liste des noms de table indexé avec le nom de la politique de service et le nom de la classe. Par conséquent, cette liste est remplie lorsque l'usager sélectionne une politique de service dans *ServPolcomboBox*.

De plus, des attributs ont été créés afin de conserver la sélection réalisée par l'utilisateur, soit les attributs :

a. routerSelected : Conserve l'identification du routeur sélectionné.
b. interfaceSelected : Conserve l'identification de l'interface sélectionnée.
c. directionSelected : Conserve l'identification de la direction sélectionnée.
d. servicePolicySelected : Conserve l'identification de la politique de service sélectionnée.
e. classSelected : Conserve l'identification de la classe sélectionnée.
f. showWhatSelected : Conserve l'identification de la statistique sélectionnée.

L'affichage graphique

Une fois que l'usager a sélectionné la statistique qu'il désire visualiser, la méthode *printGraph* est appelée. Cette méthode utilise le nom de la table, prise dans *tableNameList*, afin de savoir quelle table interroger. Par la suite elle recherche, dans la table *StatsTablesInfo* (voir ANNEXE 2), les divers noms de colonnes, à aller chercher, qui corresponds aux statistiques demandées. Par la suite elle crée une liste de points pour

chacune de ces colonnes et l'ajoute à l'objet *zgControl* (voir http://zedgraph.org/ pour plus d'informations sur cette librairie) puis défini l'élément de la légende correspondant. Ainsi il est possible d'avoir plusieurs courbes sur un seul graphique. Finalement elle recalcule l'échelle de l'abscisse, rafraîchit le graphique, le rend visible s'il ne l'était pas déjà et démarre *timer1* qui est utilisé pour le rafraîchissement graphique. En fait, à chaque fois que le minuteur *timer1* expire, la méthode *OnTimedEvent1* est lancée. Cette méthode lance à son tour la méthode *printgraph* afin de rafraîchir le graphique. Notez que l'intervalle de rafraîchissement graphique est défini par l'attribut *graphRefreshTime* qui peut être modifié par l'utilisateur à l'aide du *comboBox* *GraphRefreshTimeComboBox*. Si l'usager sélectionne un nouveau temps de rafraîchissement, la méthode *GraphRefreshTimeComboBox_SelectedIndexChanged* est lancée afin de modifier la valeur *graphRefreshTime* et de réinitialiser le minuteur.

Les boutons

Quatre boutons ont été mis à la disposition de l'utilisateur afin de lui permettre un plus grand nombre de fonctionnalités. D'abord le bouton *loadButton* est probablement un des plus utiles puisqu'il permet à un usager d'accéder aux statistiques de toutes les politiques de service surveillées. Lorsque ce bouton est appuyé, la méthode *loadButton_Click* est alors appelée. Cette méthode lance à son tour la méthode *loadRouterComboBox* afin de remplir *RouterComboBox* avec la liste des routeurs du réseau sur lesquels une politique de service est surveillée.

Le bouton *ClearSelectedButton* est quant à lui utilisé pour stopper la collecte d'information de la politique sélectionnée. Lorsque ce bouton est appuyé, la méthode *ClearSelectedButton_Click* est lancée afin de supprimer toutes les entrées de la base de données correspondant à cette politique. Elle fait appel à la méthode *clearQoSStats* du service Web.

Le bouton *ClearAllButton* est quant à lui utilisé pour stopper la collecte de statistiques de toutes les politiques de service surveillées. Elle vide également la base de données des statistiques de ces politiques.

Finalement le bouton *ExcelButton* est utilisé pour sauvegarder toutes les statistiques dans un fichier Microsoft Excel. Lorsque ce bouton est appuyé, la méthode *ExcelButton_Click* est lancée. Cette méthode appelle à son tour la méthode *SaveStats* qui se charge du formatage et de la sauvegarde des données. Cette méthode enregistre toutes les tables listées dans *tableNameList* une à la fois. Pour chacune de ces tables, les colonnes sont ajoutées une à une.

4.3.3 Service Web

Tel que mentionné précédemment, cette composante a été créée afin de servir d'interface entre l'applet du client et le reste du système. De plus, elle offre une connexion sécurisée (avec HTTPS) afin de protéger les données échangées. La Figure 26 présente le diagramme de classe du service Web. Bien qu'on ne voit qu'une seule classe, une classe (Global.asax) supplémentaire est ajoutée par défaut, afin de gérer les connexions TCP. Quant à la classe *QMA_WebService.asmx*, les diverses méthodes présentées dans la figure peuvent se séparer en trois catégories soit l'authentification et vérification d'accès, l'interaction avec la base de données puis l'interaction avec le contrôleur du serveur. De plus, certaines classes de *ModulesQMA* sont parfois instanciés directement dans les méthodes de *QMA_WebService*.

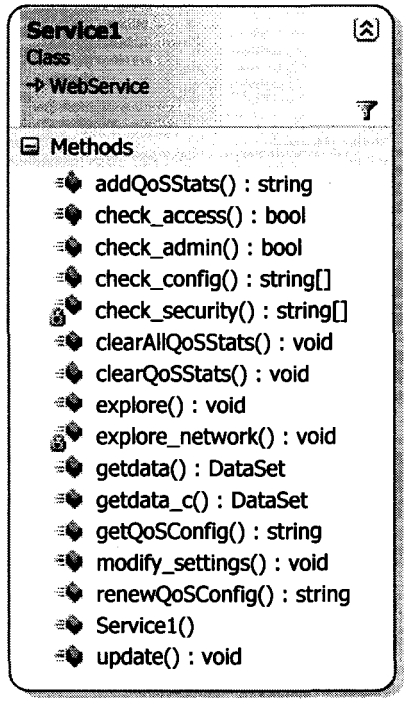

Figure 26 Diagramme de classe du service Web

4.3.3.1 Authentification et obtention des informations générales

Cette catégorie englobe les cinq premières méthodes présentées dans la Figure 26 soit : *check_access*, *check_admin*, *check_security*, *check_config* et *modify_config*. Elles sont entre autre utilisées afin de valider l'authentification d'un utilisateur et d'obtenir ou de modifier ses paramètres généraux. Ces méthodes sont résumées dans le Tableau XIII.

Tableau XIII

Méthodes du service Web utilisée pour l'authentification ou l'obtention/modification des informations générales

Méthodes	Descriptions
check_access	Appelle la méthode *check_access* de la classe *Control_Auth* de *ModulesQMA*.
check_admin	Vérifie si l'utilisateur est de type administrateur ou non et retourne un booléen indiquant s'il en est un ou non.
check_security	Appelle la méthode *check_authorization* de la classe *Control_Auth* de *ModulesQMA* et retourne les paramètres demandés par l'utilisateur.
check_config	Appelle la méthode *check_security* et retourne la liste complète des informations générales.
modify_settings	Appelle la méthode *set_settings* de la classe *Control_Auth* de *ModulesQMA* afin de configurer les informations générales pour un utilisateur particulier.

4.3.3.2 Interaction avec la base de données

Les méthodes de cette catégorie sont utilisées par l'utilisateur de l'applet afin de demander ou de modifier des entrées dans la base de données. Elles sont également utilisées par l'utilisateur de l'applet afin de remplir les bases de données de configuration. Le Tableau XIV résume ces méthodes.

Tableau XIV

Méthodes du service Web utilisées pour l'interaction avec la base de données

Méthodes	Descriptions
getdata_c	Crée une instance de *Control_DB* puis appelle la méthode *select* de cette instance afin de pouvoir retourner à l'utilisateur un *Dataset* correspondant à la table, aux colonnes et aux conditions demandées.
getdata	Appelle la méthode *getdata_c* mais sans spécifier de condition.
update	Crée une instance de *Control_DB* et appelle la méthode *update* de cette instance afin de modifier une entrée de la base de données.
explore	Crée une instance de *Control_DB* et démarre un *thread* qui exécute la méthode *explore_network.*
explore_network	Appelle la méthode *fille_all_tables* de l'instance de *Control_DB* afin de remplir les tables de configurations pour l'ensemble du réseau.
renewQoSConfig	Crée une instance de *Control_DB* et appelle la méthode *fill_QoS_Config_Tables* de cette instance afin de rafraîchir la base de données de configuration de QdS pour un équipement spécifique.

4.3.3.3 Interaction avec le contrôleur du serveur

Les méthodes de cette catégorie sont appelées par l'utilisateur de l'applet et sont exécutées par le contrôleur de serveur. Chacune de ces méthodes établi un *socket* UDP afin d'envoyer les commandes au contrôleur de serveur. Ce *socket* est détruit à la fin de la méthode.

Tableau XV

Méthodes du service Web utilisées pour l'interaction avec le contrôleur de serveur

Méthodes	Descriptions
getQoSConfig	Utilisée pour demander une configuration de QdS et la retourner à l'applet.
addQoSStats	Utilisée pour démarrer la collecte de statistiques pour une politique de service données.
clearQoSStats	Utilisée pour stopper la collecte de statistiques pour une politique de service donnée. De plus toutes les statistiques, précédemment récoltées, sont supprimées.
clearAllQoSStats	Utilisée pour stopper toutes collectes de statistiques ainsi que pour supprimer toutes les statistiques précédemment récoltées.

4.3.4 Contrôleur de serveur

Le contrôleur de serveur est une application qui est constamment active et qui fonctionne sur la même machine physique que le serveur web. Tel qu'illustré à la Figure 14, cette application utilise le protocole UDP pour communiquer avec le service Web. Afin de conserver une architecture sécuritaire, cette application doit obligatoirement fonctionner sur la même machine physique que le service Web afin que les échanges d'information ne puissent être interceptés. Cependant le protocole UDP peut éventuellement être remplacé par un autre, plus sécuritaire, afin de pouvoir séparer physiquement ces deux composantes de l'architecture (serveur Web et contrôleur de serveur).

Cette application est utilisée dans deux buts. Premièrement, elle permet une surveillance des politiques de QdS ainsi qu'une surveillance de certains aspects qui ont trait à MPLS partie traitée dans un autre mémoire [2]. Deuxièmement, elle permet de contrôler certains paramètres généraux.

Étant donné que les différentes instances des modules définis dans *ModulesQMA* sont créées uniquement lorsque le système en a besoin et détruites lorsqu'il en a finies, ces modules ne pouvaient être utilisés pour pouvoir effectuer la récolte de statistiques. Le contrôleur de serveur, qui est une application toujours en fonction, a donc été créé afin de pallier à cette problématique.

Il est important de mentionner que l'interface graphique offerte par le contrôleur de serveur est dédiée à l'administrateur du système QMA et non aux administrateurs réseau qui utilisent QMA pour surveiller le réseau. Ces derniers utiliseront plutôt l'interface offerte par l'applet.

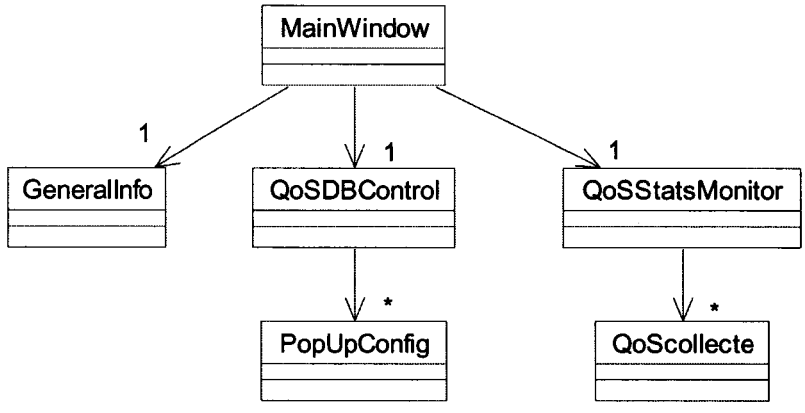

Figure 27 Diagramme de classe du contrôleur du serveur

La Figure 27 présente le diagramme de classe du contrôleur de serveur pour les options qui concernent la QdS uniquement. Cette figure montre que la forme Windows *MainWindow* possède une instance des contrôles utilisateurs *GeneralInfo*, *QoSDBControl* et *QoSStatsMonitor*. Le premier est utilisé pour modifier les paramètres généraux concernant les bases de données et les paramètres SNMP, le second est utilisé pour contrôler la base de données de configuration de la QdS ainsi que pour afficher une politique de service en utilisant la classe *PopUpConfig*, le troisième est utilisé pour gérer

les objets *QoScollecte* qui sont chargés d'effectuer la surveillance d'une politique de service. Les sections qui suivent présenteront plus en détail chacune de ces classes.

4.3.4.1 La forme Windows MainWindow

Cette forme est en quelque sorte la fenêtre principale de l'application. La Figure 28 représente le détail de cette classe. Cette fenêtre possède les instances des différents contrôles utilisateur (*generalInfo1*, *qoSDBControl1* et *qoSStatsMonitor1*). De plus, cette forme gère le changement d'onglet pour passer d'une page à une autre. Ce changement d'onglet est réalisé en utilisant les objets *tabControl1*, *tabPage1*, *tabPage2* et *tabPage3*. La méthode *setGeneralInfo* est utilisée par le contrôle utilisateur *generalInfo1* afin que les informations générales modifiées dans ce contrôle utilisateur soient modifiées également dans les autres contrôles utilisateurs.

Le rôle principal de *MainWindow* consiste à gérer la communication entre cette application et le service Web. Il exécute cette tâche en appelant les méthodes appropriées, du contrôle utilisateur adéquat, lors de la réception d'une commande du service Web. En fait, l'attribut *listener* constitue l'interface de connexion (*socket*) qui écoute sur le port UDP défini par la constante *listenPort* (50000). À la fin de l'initialisation, un *thread* est lancé afin d'exécuter la méthode *StartListener* qui est en charge d'attendre la réception de commandes provenant du service Web et d'appeler les méthodes appropriées.

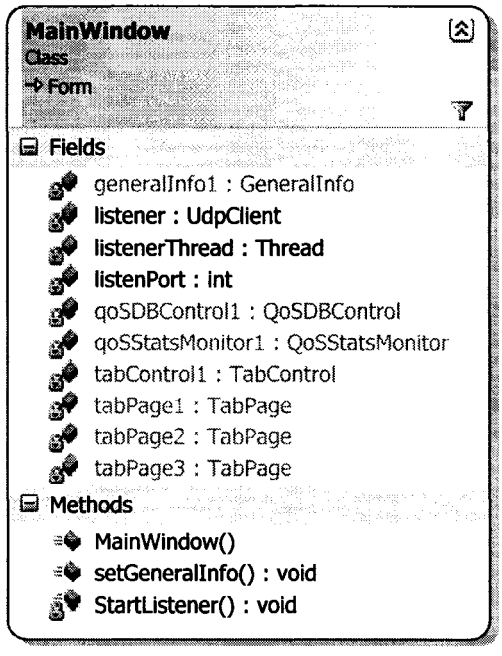

Figure 28 Détail du contrôle utilisateur MainWindow

L'organigramme de la méthode *StartListener* est représenté à la Figure 29. Dans cette figure, on voit qu'une boucle est exécutée indéfiniment. D'abord un message est attendu dont la structure est la suivante :

NomDeLaCommande Paramètre1 Paramètre2 Paramètre3 ...

Cette structure débute par le nom de la commande suivi des paramètres tous séparés par un espace. Le nombre de paramètres transmis dépend de la commande utilisée. Les messages sont de format chaîne de caractères. Une fois qu'un message est reçu, une vérification du type de message est faite puis la méthode adéquate est lancée. Il y a deux principaux types de message, ceux utilisés pour agir sur les statistiques et celui pour agir sur les configurations. Comme il y a des contrôles utilisateurs conçus spécifiquement pour agir sur les statistiques (*qoSStatsMonitor1*) et sur les configurations (*qoSDBControl1*), les méthodes de ces contrôles utilisateurs, correspondants aux commandes reçues dans les messages, seront lancées.

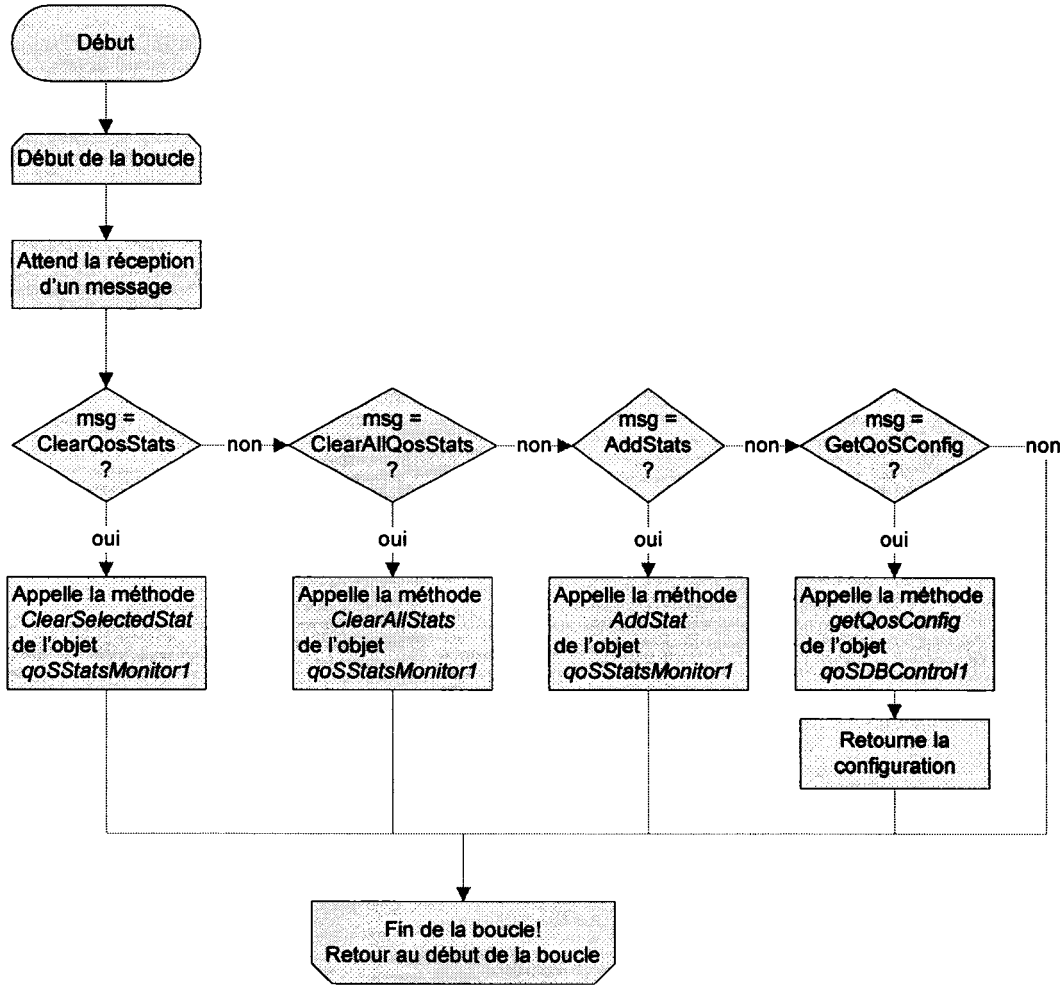

Figure 29 Organigramme de la méthode *StartListener*

En se basant sur les explications ci-dessus, la commande *ClearQosStats* est utilisée pour supprimer les statistiques d'une politique de service précise, identifiée par les divers paramètres. Par conséquent la méthode *ClearSelectedStat* de l'objet *qoSStatsMonitor1* est appelée. Similairement, lorsque la commande *ClearAllQosStats* est reçue, la méthode *ClearAllStats* du même objet est lancée. Cette commande n'inclut aucun paramètre. Lorsque la commande *AddStats* est reçue avec les paramètres qui identifient la politique de façon unique, la méthode *AddStat* du même objet est lancée. Finalement, lorsque la méthode *GetQoSConfig* est reçue, suivie des paramètres qui identifient la politique de

façon unique, la méthode *getQosConfig* de l'objet *qoSDBControl1* est lancée. Cette méthode retourne la configuration, en format chaîne de caractères, qui sera alors retournée à l'entité qui est à l'origine de la commande et ce, en utilisant la même interface de connexion, c'est-à-dire les mêmes adresses IP et ports UDP source et destination.

4.3.4.2 Le contrôle utilisateur GeneralInfo

Ce contrôle utilisateur (voir Figure 30) a été créé afin de centraliser les informations générales utilisées par la majorité des classes de cette application. Si des modifications s'imposent, celles-ci sont réalisées à un seul endroit et sont diffusées à toutes les entités qui les nécessitent. Ces informations générales, codées en dur via des chaînes de caractères, sont présentées à l'utilisateur dans des *textBox*. Le Tableau XVI présente la liste des informations générales ainsi que leurs variables et descriptions associées.

Tableau XVI

Descriptions des informations générales et de leurs variables associées

Nom des chaînes de caractères	**Nom des *textBox***	**Descriptions**
SQLServer	SQLServerTextBox	Nom du serveur SQL.
ConfigDBName	ConfigDBNameTextBox	Nom de la base de données utilisée pour les configurations.
StatsDBName	StatsDBNameTextBox	Nom de la base de données utilisée pour les statistiques.
DBUser	DBLoginTextBox	Nom d'utilisateur utilisé pour l'accès aux bases de données.

Tableau XVI (suite)

Nom des chaînes de caractères	**Nom des *textBox***	**Descriptions**
DBPwd	DBPassTextBox	Mot de passe utilisé pour l'accès aux bases de données.
snmpUser	SNMPLoginTextBox	Nom d'utilisateur utilisé pour les requêtes SNMP.
snmpPwd	SNMPPassTextBox	Mot de passe utilisé pour les requêtes SNMP.

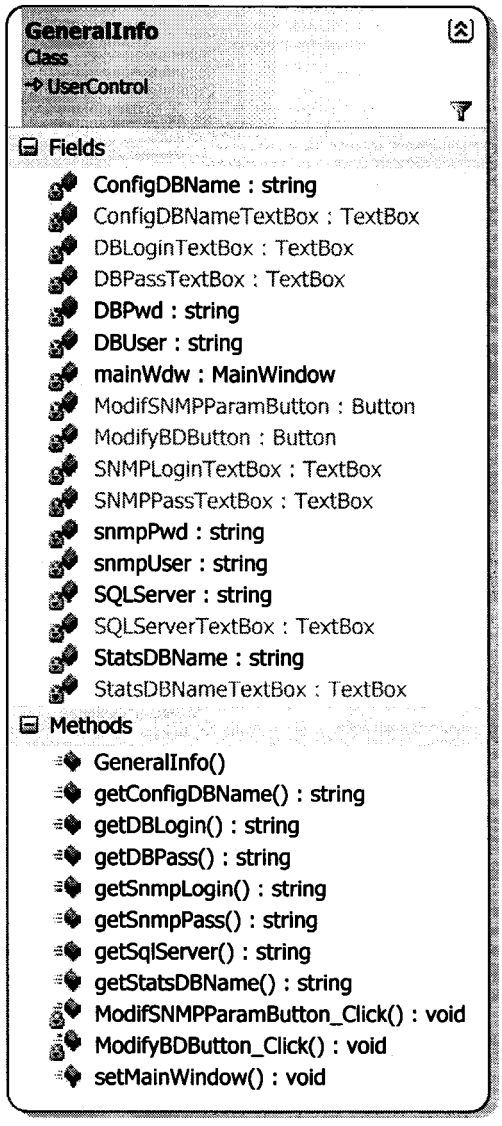

Figure 30 Détail du contrôle utilisateur GeneralInfo du contrôleur de serveur

Comme il peut être vu dans la Figure 30, ce contrôle utilisateur contient une référence vers le contrôle utilisateur *MainWindow*. Cette référence est configurée par le *MainWindow* lui-même, via la méthode *setMainWindow*, une fois que ce dernier a initialisé l'objet *generalInfo1*. Elle est utilisée lorsque l'usager modifie une des informations et clique sur le bouton *ModifyDBButton* ou sur le bouton *ModifSNMPParamButton*. La méthode *setGeneralInfo* du *MainWindow* est alors appelée

afin que les modifications soient diffusées aux autres contrôles utilisateurs. Cette méthode du *MainWindow* utilise d'abord les accesseurs *getSqlServer*, *getConfigDBName*, *getStatsDBName*, *getDBLogin*, *getDBPass*, *getSnmpLogin* et *getSnmpPass* pour accéder aux nouvelles informations puis les diffuse par la suite.

4.3.4.3 Le contrôle utilisateur QoSDBControl

Ce contrôle utilisateur a deux principaux rôles. D'abord il comporte une panoplie de boutons permettant de remplir les tables de configurations de QdS pour un équipement donné. Par ailleurs, il peut être utilisé pour afficher la configuration d'une politique de service pour un équipement donné. La Figure 31 présente le détail de la classe. Certains attributs n'y ont pas été affichés afin de rendre la figure plus lisible. On retrouve, parmi ces attributs, les informations générales qui peuvent être configurées par le *MainWindow* via la méthode *SetGeneralInfo*.

D'abord le *ListBox3* dresse la liste des équipements présents dans le réseau. Il est utilisé afin que l'usager puisse sélectionner un équipement pour lequel il désire effectuer une action. Lorsqu'il sélectionne un équipement, la méthode *listBox3_SelectedIndexChanged* est lancée. Cette méthode lance alors l'autre méthode *Fill_ListBox1_QoSConfig* afin que le *listBox1* dresse la liste des interfaces de cet équipement, pour lesquelles une politique de service a été appliquée. L'usager peut alors soit cliquer sur un des boutons pour effectuer une action sur l'équipement en question, soit cliquer sur une des interfaces affichées dans le *listBox1*. Lorsqu'il sélectionne une interface, la méthode *listBox1_SelectedIndexChanged* est lancée. Cette méthode lance par la suite l'autre méthode *Fill_ListBox2_ServicePolicy* afin d'afficher les politiques de services présentes sur l'interface sélectionnée. Lorsque l'usager sélectionne une des politiques affichées dans le *listBox2*, la méthode *listBox2_Selected_IndexChanged* est lancée. Cette méthode lance alors la méthode *getQosConfig* qui retourne la configuration. Cette méthode est décrite par l'organigramme présenté et décrit à la

Figure 65 de l'ANNEXE 2. Par la suite, un objet *PopUpConfig* est créé afin d'afficher la configuration dans une fenêtre de type *pop up*.

Les boutons

En tout temps, l'usager peut cliquer sur le bouton *Clear_All* afin que la méthode *Clear_All_Click* soit lancée. Cette méthode est utilisée afin de vider toutes les tables de configuration de QdS de tous les équipements listés dans le *listBox3*. Les autres boutons nécessitent la sélection d'un équipement spécifique dans le *listBox3*. Leur liste et descriptions sont présentées dans les points qui suivent. Notez qu'avant de remplir, pour un équipement donné, les tables relatives aux configurations de la QdS, la table *Router* doit impérativement être remplie. Cette table peut être remplie en cliquant sur le bouton *Fill_Router_Table.*

a. Le bouton *Fill_Router_Table* lance la méthode *Fill_Router_Table_Click.*
   - Remplit la table *Router* du routeur sélectionné.

b. Le bouton *Empty_Router_Table* lance la méthode *Empty_Router_Table_Click.*
   - Vide la table *Router* du routeur sélectionné.

c. Le bouton *Fill_All* lance la méthode *Fill_All_Click.*
   - Remplit toutes les tables de QdS du routeur sélectionné en appelant la méthode *Fill_All_QoS_Table.* Suit la séquence de remplissage suivante :

ServicePolicy → *Class* → *MatchStmt* → *Queueing* → *Shaping* → *Policing* → *RED*

d. Le bouton *Remove_All* lance la méthode *Remove_All_Click.*
   - Vide toutes les tables de configuration de QdS du routeur sélectionné. Suit la séquence inverse de celle citée dans le point précédent.

e. Le bouton *Fill_ServicePolicyTable* lance *Fill_ServicePolicyTable_Click.*
   - Remplit la table *ServicePolicy* du routeur sélectionné.

f. Le bouton *Empty_ServicePolicyTable* lance *Empty_ServicePolicyTable_Click.*
   - Vide la table *ServicePolicy* du routeur sélectionné.

g. Le bouton *Fill_ClasseTable* lance la méthode *Fill_ClasseTable_Click.*

- Remplit la table *Class* du routeur sélectionné.

h. Le bouton *Empty_ClasseTable* lance la méthode *Empty_ClasseTable_Click.*
   - Vide la table *Class* du routeur sélectionné.

i. Le bouton *Fill_MatchStmt_Table* lance la méthode *Fill_MatchStmt_Table_Click.*
   - Remplit la table *MatchStmt* du routeur sélectionné.

j. Le bouton *Empty_MatchStmt_Table* lance *Empty_MatchStmt_Table_Click.*
   - Vide la table *MatchStmt* du routeur sélectionné.

k. Le bouton *Fill_Queueing_Table* lance la méthode *Fill_Queueing_Table_Click.*
   - Remplit la table *Queueing* du routeur sélectionné.

l. Le bouton *Empty_Queueing_Table* lance *Empty_Queueing_Table_Click.*
   - Vide la table *Queueing* du routeur sélectionné.

m. Le bouton *Fill_Shaping_Table* lance la méthode *Fill_Shaping_Table_Click.*
   - Remplit la table *Shaping* du routeur sélectionné.

n. Le bouton *Empty_Shaping_Table* lance *Empty_Shaping_Table_Click.*
   - Vide la table *Shaping* du routeur sélectionné.

o. Le bouton *Fill_Policing_Table* lance la méthode *Fill_Policing_Table_Click.*
   - Remplit la table *Policing* du routeur sélectionné.

p. Le bouton *Empty_Policing_table* lance la méthode *Empty_Policing_table_Click.*
   - Vide la table *Policing* du routeur sélectionné.

q. Le bouton *Fill_RED_table* lance la méthode *Fill_RED_table_Click.*
   - Remplit la table *RED* du routeur sélectionné.

r. Le bouton *Empty_RED_Table* lance la méthode *Empty_RED_Table_Click.*
   - Vide la table *RED* du routeur sélectionné.

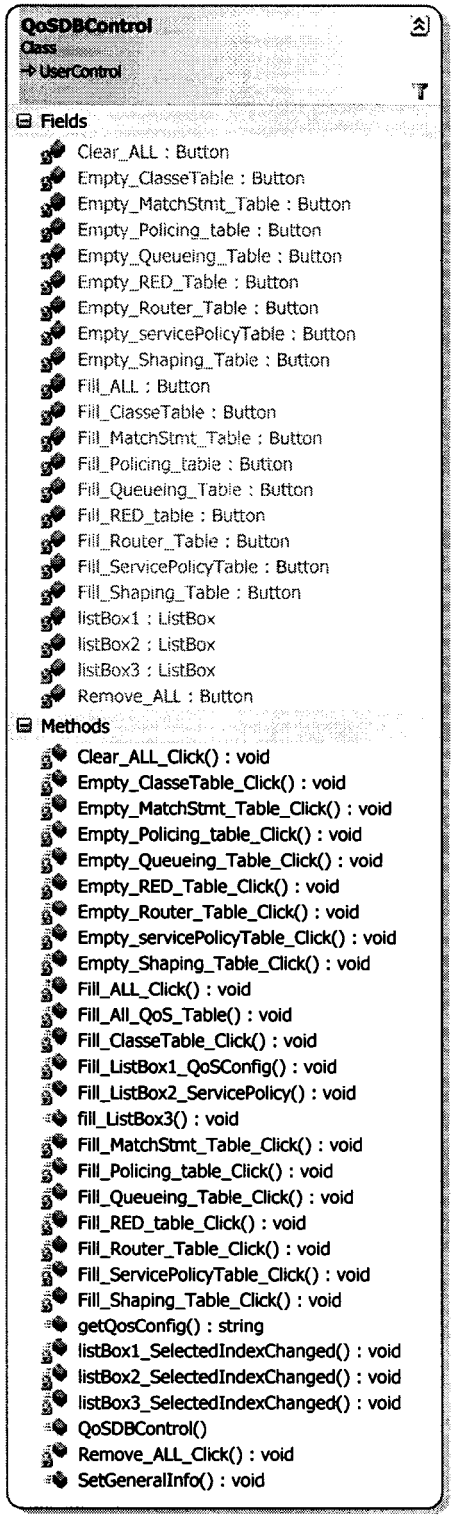

Figure 31 Détail du contrôle utilisateur QoSDBControl du contrôleur de serveur

4.3.4.4 La classe PopUpConfig

Cette classe est utilisée uniquement afin d'afficher une configuration de QdS. Par conséquent elle hérite de *WindowsForm* et ne détient qu'un seul *textBox*. Sa seule méthode est le constructeur de la classe. Elle prend comme paramètre une chaîne de caractères correspondant à la politique de service ainsi qu'une chaîne de caractères correspondant à l'identificateur de la politique de service. La première chaîne de caractères est donc insérée dans le *textbox* tandis que la seconde est insérée comme titre de la fenêtre.

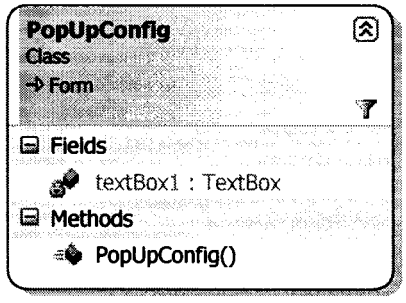

Figure 32 Détail de la classe PopUpConfig

4.3.4.5 Le contrôle utilisateur QoSStatsMonitor

Ce contrôle utilisateur a principalement les mêmes contrôles et fonctionnalités que ceux du contrôle utilisateur *QoSStatsUC* de l'applet. Vous pouvez donc référez à la section 4.3.2.5 pour mieux comprendre les fonctionnalités des *comboBox* et des boutons. La Figure 33 présente le détail de la classe du contrôle utilisateur *QoSStatsMonitor*.

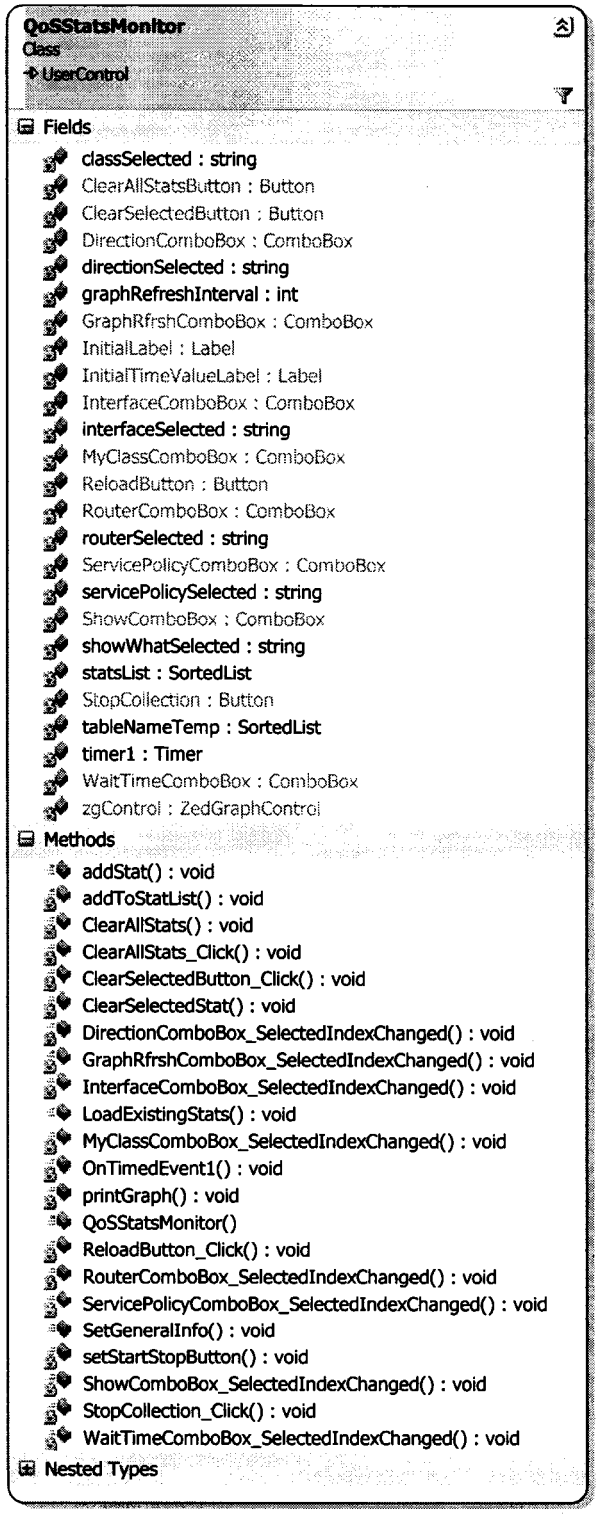

Figure 33 Détail du contrôle utilisateur QoSStatsMonitor

Ce contrôle utilisateur a cependant certaines fonctionnalités que le contrôle utilisateur *QoSStatsUC* de l'applet n'a pas. Dans un premier temps, l'usager a la possibilité de stopper et de redémarrer à sa guise la collecte d'information pour une politique de service donnée. Cette fonctionnalité est principalement utilisée pour le débogage et n'est absolument pas utile pour un utilisateur de l'applet. Dans un deuxième temps, l'usager a la possibilité de modifier le temps d'attente entre deux collectes de statistiques consécutives pour une politique de service donnée. Cette fonctionnalité ne doit pas être accessible à un utilisateur de l'applet étant donné qu'elle peut compromettre les performances du système. Il faut donc qu'elle soit utilisée avec précautions, par un utilisateur averti.

Dans un troisième temps, *QoSStatsMonitor* possède une autre fonctionnalité très importante et qui constitue en fait, sa principale raison d'être. En effet, *QoSStatsMonitor* est en charge de la gestion des instances de surveillance *QoScollecte*. Une telle instance est créée pour chaque politique de service surveillée. Il est important de mentionner que jusqu'à présent, aucun test de performance n'a été effectué afin de déterminer le nombre maximal de politique de service que QMA peut surveiller en concurrence. Mais il est évident que QMA ne serait pas en mesure de surveiller tout un réseau à lui seul. Référez-vous à la section RECOMMANDATIONS pour une architecture plus performante.

Le contrôle utilisateur *QoSStatsMonitor* possède une liste triée (*statsList*) qui contient la liste des instances de *QoScollecte* indexées par un identificateur qui permet de reconnaître une politique de façon unique dans tout le réseau. Cet identificateur est constitué du nom de l'équipement, du nom de l'interface, de la direction ainsi que du nom de la politique tous séparés par un « _ ». Pour ajouter un élément à cette liste, la méthode *addToStatList* doit être appelée. Cette méthode vérifie d'abord si la statistique à ajouter existe déjà dans la liste. Si elle n'y est pas, elle crée une nouvelle instance de *QoScollecte* et l'ajoute à la liste tout simplement. La section 4.3.4.6 présente plus en détail cette dernière classe.

4.3.4.6 La classe QoScollecte

Cette classe est utilisée afin de récolter les statistiques d'une politique de services de façon périodique. La Figure 34 en présente le détail.

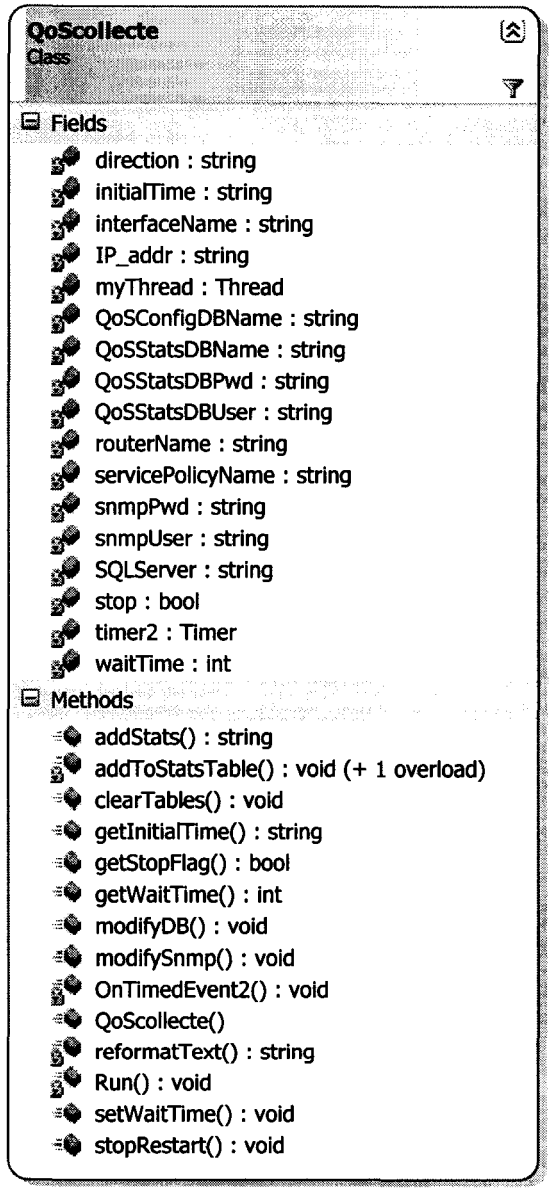

Figure 34 Détail de la classe QoScollecte

D'abord on peut voir dans cette figure que cette classe nécessite la connaissance des informations générales au sujet des bases de données et des configurations SNMP. Ces divers attributs sont introduits dans les paramètres du constructeur de la classe. Les méthodes *modifyDB* et *modifySnmp* ont été créées afin de pouvoir modifier ces attributs. De plus, les informations permettant d'identifier la politique de service de façon unique (Nom du routeur, nom de l'interface, direction et nom de la politique) sont également données dans les paramètres du constructeur. Lors de l'exécution de ce dernier, la méthode *addStats* est appelée afin de créer les tables requises, si nécessaire, dans la base de données (voir ANNEXE 2 pour plus de détail sur ces tables). Par la suite, l'adresse IP requise pour pouvoir communiquer avec l'équipement en question, est récupérée de la table *Router*. Puis, la date et l'heure de la création de l'objet sont conservées dans l'attribut *initialTime*. Finalement un *thread* est créé et lancé afin d'exécuter la méthode *Run* dont l'organigramme est illustré à la Figure 35.

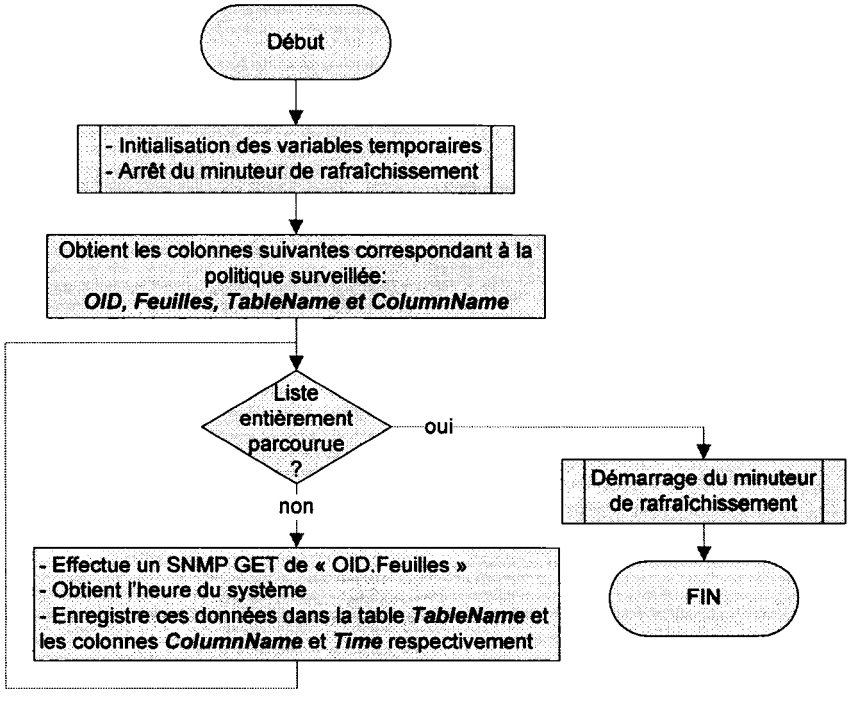

Figure 35 Organigramme de la méthode Run

Dans cette figure, on voit que le *thread* (*myThread*) débute en arrêtant le minuteur (*Timer2*) et en obtenant les colonnes *OID*, *Feuilles*, *TableName* et *ColumnName* depuis la table *StatsTablesInfo*. Par la suite chaque ligne du *DataSet* obtenu sera traitée en boucle. Pour chaque combinaison *OID.Feuilles* de la liste, une requête SNMP GET sera envoyée à l'équipement approprié afin d'obtenir la statistique désirée. Par la suite, la référence temporelle sera prise en se basant sur la date et l'heure du système à laquelle la réponse SNMP aura été obtenue. Finalement, la nouvelle statistique sera introduite dans la colonne *ColumnName* et la référence temporelle dans la colonne *Time* de la table *TableName*. Lorsque toute la liste aura été parcourue, le minuteur sera redémarré afin d'attendre un certain temps avant d'effectuer cette routine à nouveau. À chaque fois que le minuteur expire, la méthode *OnTimedEvent2* est lancée afin de démarrer à nouveau l'exécution de la méthode *Run* par *myThread*.

Le reste des méthodes est décrit dans le Tableau XVII.

Tableau XVII

Descriptions de quelques méthodes de la classe QoScollecte

Méthodes	**Descriptions**
clearTables	Supprime toutes les tables de la base de données qui correspondent à la politique de service surveillée par cet objet. Supprime également toutes les entrées correspondantes dans la table *StatsTablesInfo*.
getInitialtime	Retourne la date et l'heure de création de l'objet de surveillance.
getStopFlag	Retourne l'attribut *stop* qui informe si la collecte de statistiques est stoppée ou non.
getWaitTime	Retourne l'attribut *initialTime* qui correspond à l'intervalle d'attente entre deux collectes de statistiques.

Tableau XVII (suite)

Méthodes	Descriptions
reformatText	Reformate le texte passé en paramètre afin de remplacer les caractères '.', '/' et '-' par le caractère '_'. Cette méthode est particulièrement utile pour donner des noms significatifs aux tables et aux colonnes de la base de données étant donné qu'il s'agit de caractères interdits. Par exemple, les chaînes de caractères « FastEthernet0/0.50 » ou « class-default » ne seraient pas utilisables.
setWaitTime	Reconfigure l'attribut *waitTime* qui définit l'intervalle d'attente entre deux collectes de statistiques consécutives (utilisé par le minuteur *Timer2*).
stopRestart	Arrête ou redémarre la collecte de statistiques et configure l'attribut *stop* en conséquence.

4.4 Interaction entre les composantes de l'architecture générale

Cette section présente les interactions entre les diverses composantes de l'architecture. D'abord, la première sous-section présentera un diagramme de classes général afin de voir comment chacune des classes est utilisée. Quant à la seconde sous-section, elle présentera les diagrammes de séquences qui permettent de voir l'interaction entre les composantes de l'architecture et ce, pour les principales opérations pouvant être initiées par l'utilisateur.

4.4.1 Diagramme de classes

Cette section présente le diagramme de classe général, présenté à la Figure 36, dans lequel l'interaction entre les diverses classes du système est représentée. On voit que la classe *QMA* de l'applet possède une référence vers le service Web *QMA_WebService*.

Tel que mentionné dans la section 4.3.2, cette référence est par la suite distribuée aux contrôles utilisateur de l'applet qui la nécessite.

Par ailleurs, le service Web utilise une instance de la classe *Control_Auth* de *ModulesQMA* afin d'authentifier un usager. Cette classe possède également une instance de *Control_DB* et de *Database* de *ModulesQMA* afin d'interagir avec le réseau et/ou la base de données. Ce service Web peut créer, lorsque requis, une instance de la classe *Socket* afin de communiquer avec le contrôleur de serveur.

Quant aux différents modules de *ModulesQMA*, ceux-ci sont créés de façon sporadique par le service Web ou encore par le contrôleur de serveur. En fait, leurs instances sont créées à chaque fois que le service Web ou le contrôleur de serveur désire exécuter une opération d'authentification ou encore une opération sur la base de données ou sur le réseau.

Finalement, le contrôleur de serveur possède des instances des classes de *ModulesQMA* dont *Database*, *Control_DB* et *Snmp*. La première étant utilisée lorsque le contrôleur de serveur désire effectuer des opérations sur la base de données uniquement. La seconde étant utilisée lorsque le contrôleur de serveur désire remplir la base de données selon les configurations du réseau. La troisième étant utilisée par la classe *QoScollecte* afin que le contrôleur de serveur puisse interroger le réseau pour obtenir les statistiques qu'il désire. Finalement, cette composante de l'architecture possède une instance de la classe *ClientUdp*, qui hérite de la classe *Socket*, utilisée pour recevoir des commandes du service Web.

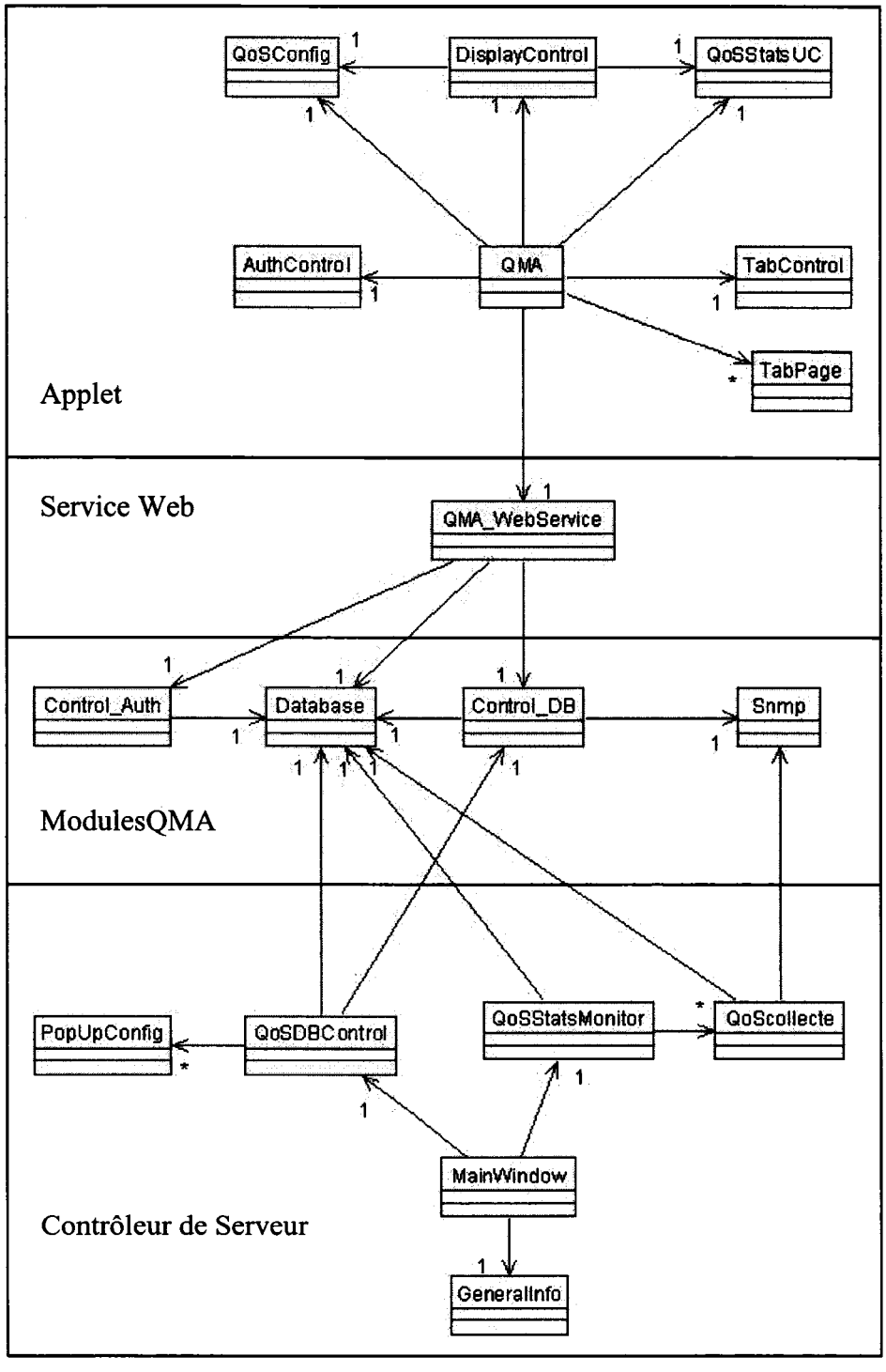

Figure 36 Diagramme de classe général montrant l'interaction entre les modules

4.4.2 Diagrammes de séquences

Cette section présente les diagrammes de séquences des principales actions pouvant être exécutée par un utilisateur. Le principal but de cette section est de représenter, pour ces actions, les interactions entre les diverses composantes de l'architecture. Encore une fois, seules les actions concernant la QdS seront représentées. Tel qu'il peut être vu dans le manuel de l'utilisateur (voir ANNEXE 3), ces actions consistent à demander, pour un équipement donné, un renouvellement des configurations de QdS. Par ailleurs, l'usager peut demander de visualiser une configuration, ou encore de démarrer une collecte de statistique.

4.4.2.1 Opération *ReNew QoS Configurations*

La Figure 37 représente le diagramme de séquence réalisé lorsque l'usager sélectionne l'option *ReNew QoS Configurations*. La classe *DisplayControl* de l'applet appelle alors la méthode *renewQoSConfig* du service Web. Par la suite, ce dernier appelle la méthode *fill_QoS_Config_Tables* qui est en charge d'interroger l'équipement en question et de remplir la base de données de configurations de QdS.

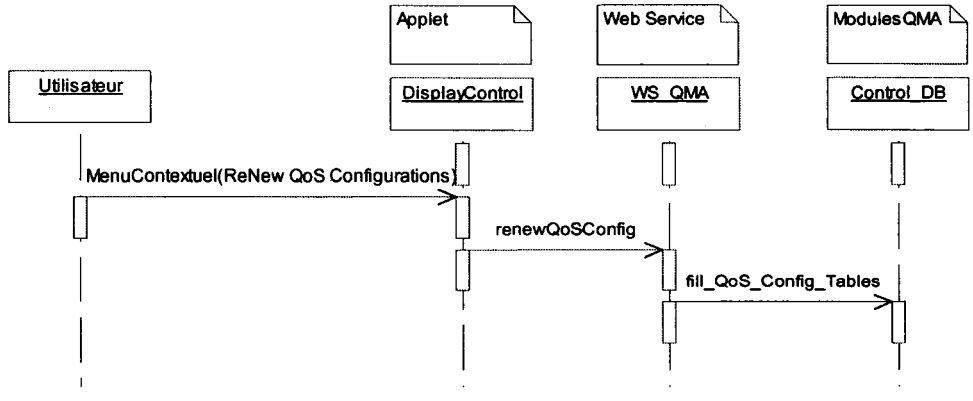

Figure 37 Diagramme de séquence de l'opération *ReNew QoS Configurations*

4.4.2.2 Opération *Show Service Policy*

La Figure 38 représente, quant à elle, la séquence d'exécution lorsque l'utilisateur a sélectionné l'option *Show Service Policy* du menu contextuel. Le contrôle utilisateur obtient la liste des interfaces, via le service Web et la classe *Database*, sur lesquelles une configuration de QdS a été appliquée. Cette liste est alors affichée dans le *listBox1*. Lorsque l'utilisateur sélectionne une interface, le contrôle utilisateur interroge la base de données, via le service Web, afin d'obtenir la liste des politiques de service appliquées sur l'interface sélectionnée. Cette liste est alors affichée dans le *listBox2*.

Lorsque l'utilisateur sélectionne la politique de service dont il désire voir la configuration, le contrôle utilisateur *DisplayControl* appelle la méthode *getQoSConfig* du service Web. Ce dernier communique avec le contrôleur de serveur pour lui envoyer la commande *GetQoSConfig*. Lorsque le contrôle utilisateur *MainWindow* du contrôleur de serveur intercepte la commande, il lance alors la méthode *getQosConfig* du contrôle utilisateur *QoSDBControl*. Ce dernier formate, dans une chaîne de caractères, les informations de configuration obtenues de la base de données et retourne cette chaîne de caractères au *MainWindow*. Ce dernier retourne la réponse au service Web qui la retourne également au contrôle utilisateur *DisplayControl* de l'applet. Finalement, la configuration est ajoutée à la liste de configuration du contrôle utilisateur *QoSConfig*.

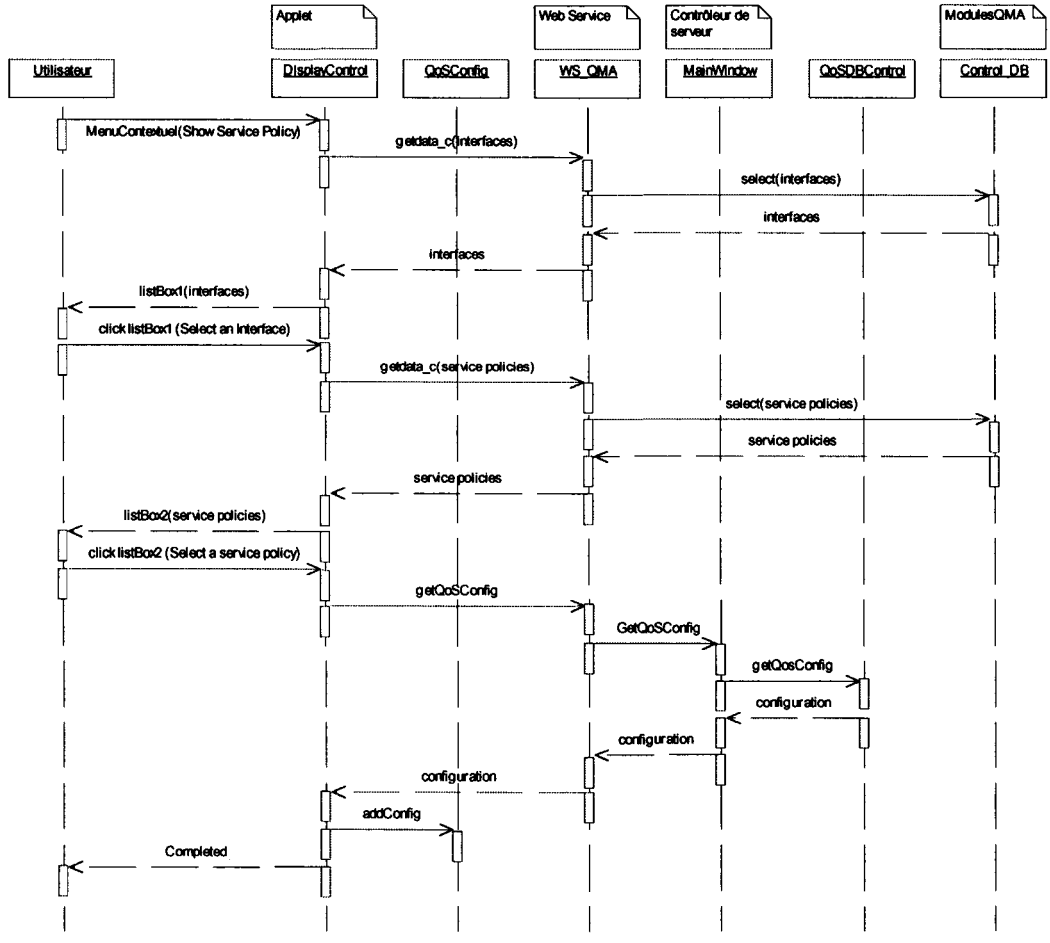

Figure 38 Diagramme de séquence de l'opération *Show Service Policy*

4.4.2.3 Opération *Begin Statistic Collection*

La Figure 39 représente la séquence d'exécution réalisée lorsque l'utilisateur sélectionne l'option *Begin Statistic Collection* du menu contextuel. Tel que spécifié lors de la séquence d'exécution de l'option *Show Service Policy*, le *listBox1* dresse la liste des interfaces sur lesquelles une politique de service est appliquée. Lorsque l'utilisateur

sélectionne une interface, le *listBox2* dresse alors la liste des politiques de service appliquées sur l'interface.

Lorsque l'utilisateur sélectionne une politique de service, le contrôle utilisateur *DisplayControl* de l'applet lance la méthode *addStats* du contrôle utilisateur *QoSStatsUC*. Ce dernier appelle la méthode *addQoSStats* du service Web lequel envoie un message au contrôleur de serveur dont le nom de commande est *addStats*. Ce message est alors intercepté par le contrôle utilisateur *MainWindow* du contrôleur de serveur lequel lance la méthode *addStat* de *QoSStatsMonitor*. Cette méthode vérifie dans sa liste si elle possède une instance de *QoScollecte* correspondant à celle demandée. Si non, un nouvel objet *QoScollecte* est créé. Trois messages peuvent être retournés par le *QoSStatsMonitor*. Le message *OK* sera retourné si l'opération a bien été exécutée, le message *Error* sera retourné si l'opération a échouée et le message *Already Exist* sera retourné si les tables existaient déjà dans la base de données.

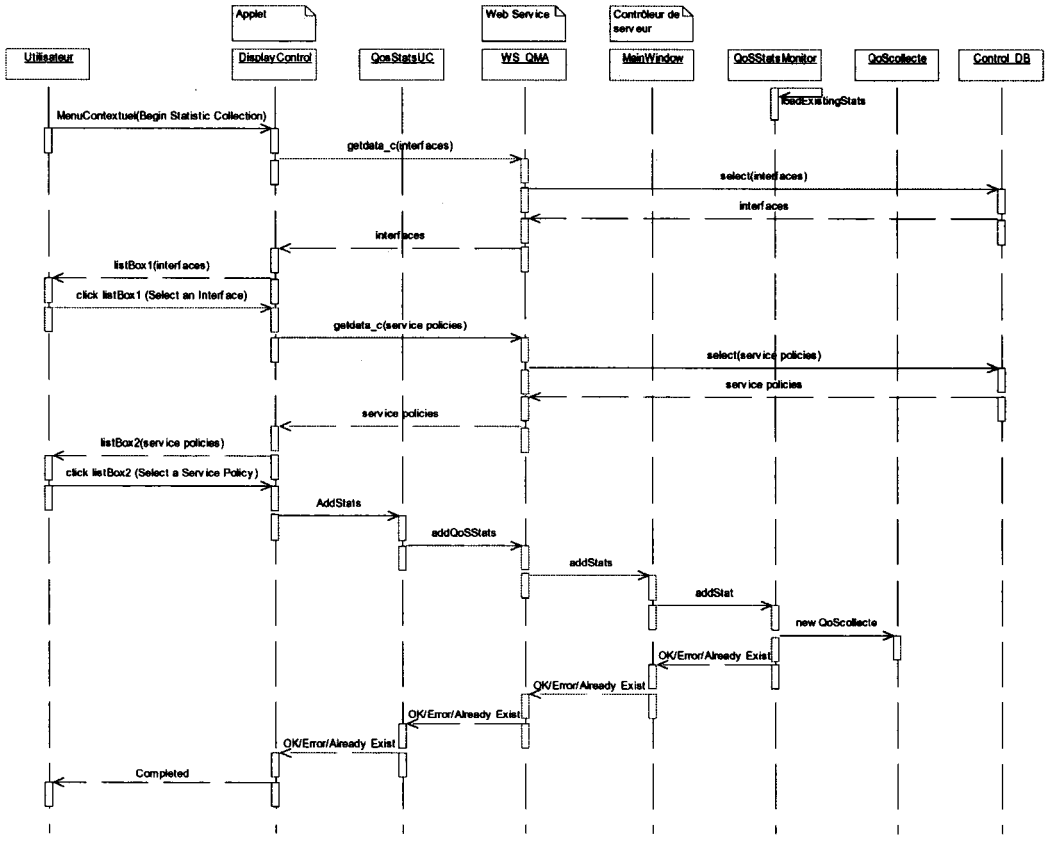

Figure 39 Diagramme de séquence de l'opération *Begin Statistic Collection*

4.5 Intégration d'autres équipementiers à QMA

Cette section présente une approche pouvant être utilisée, pour le développement futur de QMA, afin que le système puisse interragir avec des équipements d'un équipementier autre que Cisco Systems. La base de données actuellement créée et présentée à l'ANNEXE 2 utilise un format général pouvant être adapté, s'il y a lieu, pour qu'il puisse supporter de nouvelles MIB. Dans le cas où les modifications à cette base de données seraient trop importantes pour simplement y interfacer une nouvelle MIB, une nouvelle base de données pourrait être créée pour cette MIB spécifique. Dans une telle

situation, une table devra être créée pour conserver la liste des équipements et de la base de données qu'ils utilisent. Cette table pourrait, par exemple, inclure les colonnes suivantes :

a. ***HostName*** : Nom d'hôte de l'équipement.
b. ***IP*** : Adresse IP utilisée pour rejoindre l'équipement.
c. ***Constructor*** : Nom de l'équipementier.
d. ***Model*** : Nom ou numéro du modèle de l'équipement.
e. ***QoSConfigDatabase*** : Nom de la base de données utilisée pour emmagasiner les configurations de QdS de cet équipement.

Une fois ceci réalisé, certaines modifications devront être apportées à la classe *ControlDB* de *ModulesQMA* afin que QMA aille d'abord obtenir les informations sur l'équipementier et le modèle de l'équipement à interroger. Une fois que *ControlDB* connaît ces informations, il n'aurait donc qu'à utiliser les OID adéquats et à sauvegarder les réponses dans la base de données correspondante. Similairement, à chaque fois qu'une classe aurait besoin d'accéder à la base de données de configuration de QdS, cette classe devrait préalablement aller voir dans cette table quelle base de données interroger.

Pour conclure, après avoir décrit l'architecture générale du système QMA, ce chapitre a détaillé chaque composante de cette architecture. Pour chaque composante, les classes utilisées ainsi que leurs principaux attributs et méthodes ont été listés et décrits. Par la suite, ce chapitre a expliqué l'interaction entre les diverses composantes de l'architecture et ce, pour les principales commandes pouvant être lancées par l'utilisateur. Pour terminer, ce chapitre a présenté une approche pouvant être utilisée pour permettre au système QMA d'interragir avec plusieurs équipementiers. Le chapitre qui suit valide le fonctionnement de ces commandes et, du fait même, l'interaction entre les diverses composantes de l'architecture générale.