



Université de Fribourg, Suisse

Département d'informatique

Bachelor en informatique de gestion

PMS QUOTES

Mise à niveau du programme d'importation de la BSU

Travail de séminaire

Pierre Vanhulst

Route de Combarod 5a, 1720 Corminboeuf

Dr. Stefan Husemann

Décembre 2009

Résumé

Ce travail a pour objectif de fournir à l'association d'étudiants Börsenspiel des Schweizer Universitäten (BSU) un programme d'importation des cours neuf et performant. À cette fin, seront traitées les questions des sources de données potentielles, des méthodes pour acheminer lesdites données jusqu'au Système d'Information (SI) de la BSU et de l'analyse des besoins du nouveau programme d'importation.

En fin de compte, toutes ces questions amèneront à la création d'une architecture pour le nouveau programme d'importation - appelé PMS Quotes - et à la rédaction d'un prototype fonctionnel amené à évoluer dans les mois à venir.

Mots-clés

BSU, CSV, Cours boursiers, fournisseurs de données financières, .Net, PMS, REST, Simulation boursière, SOA, UML, Web Services, XML

Table des matières

1.Introduction.....	1
2.Problématique.....	1
3.Objectifs.....	1
4.Méthodes de travail.....	2
5.Échange de données entre systèmes d'information.....	3
6.Sources de données.....	3
7.Historique des fournisseurs de données.....	3
8.Le Futur.....	4
9.Standards d'échange et Web Services.....	4
10.Définition et fonctionnement des Web Services.....	5
11.Différents types de Services.....	6
12.Choix du type de Service pour le PMS.....	8
13.PMS Quotes.....	9
14.Aperçu de la Base de Données de la BSU.....	9
15.L'ancêtre du PMS Quotes.....	11
16.Architecture du PMS Quotes.....	11
17.Analyse des besoins du PMS Quotes et cas d'utilisation.....	12
18.Modules du PMS Quotes.....	19
19.Architecture de la base de données du PMS Quotes.....	20
20.Extensions possibles et problèmes.....	21
21.Prototype PMS Quotes.....	23
22.Différences avec le PMS Quotes et limitations.....	23
23.Extraits notoires du Code Source.....	23
24.Importation du fichier CSV.....	23
25.Exportation vers "PMSQuotes_Kurs_Detail".....	25
26.Conclusion.....	27
27.Glossaire.....	28
28.Bibliographie.....	29

1. Introduction

1.1. *Problématique*

L'association d'étudiants **Börsenspiel der Schweizer Universitäten (BSU)** organise depuis plus de dix ans des simulations boursières accessibles à tous les étudiants des Universités et des Hautes Écoles de la Suisse. Ces simulations boursières - appelées **Portfolio Management Simulation (PMS)** - permettent aux membres de l'association de gérer un budget fictif de 1'000'000 CHF durant deux mois, à l'issue desquels les gagnants sont définis selon plusieurs critères. Dans ce contexte, la BSU propose aux joueurs d'investir dans environ 150 titres¹, dont les cours sont importés quotidiennement à la fermeture de la bourse².

L'importation desdits cours est possible grâce aux partenaires de la BSU et fonctionne à l'aide d'un programme qui introduit automatiquement les données dans le **Système d'Information (SI)** de la BSU. Malheureusement, l'association utilise depuis de nombreuses années un programme d'importation rudimentaire, écrit sans véritable méthodologie, ne respectant pas les recommandations suggérée par le génie logiciel et dénué de documentation. Il souffre également d'un certain nombre de bugs, rendant l'importation des cours hasardeuse, ce qui entraîna une succession de corrections palliatives qui n'ont que trop longtemps duré.

Dans l'optique d'améliorer ses services, la BSU a décidé de lancer le projet **PMS Top**, dont le but était de remettre à niveau toute l'infrastructure informatique de la BSU. Parmi les sous-projets, le **PMS Quotes** s'occupe de l'importation des données.

1.2. *Objectifs*

L'objectif de ce travail est de fournir une base exploitable pour offrir à la BSU un programme d'importation efficace.

Dans un premier temps, il sera question d'analyser les différentes possibilités qui sont offertes à la BSU pour se fournir les données. *D'où viennent les cours de la BSU ?* Cette première question s'intéresse directement aux liens que la BSU a pu tisser avec

¹ Ce chiffre peut passablement varier selon les années.

² Les négociations sont arrêtées tous les jours à 17:30, heure à laquelle la partie automatique de l'importation est lancée. Cependant, la véritable importation vers le jeu exige l'intervention d'un administrateur et peut parfois, dans le pire des cas, prendre toute une nuit.

ses partenaire au cours des années. Y sera fait une liste de nos précédents partenaires, et de ceux qui pourraient devenir nos fournisseurs de demain. *Existe-t-il des standards pour échanger les données entre différents SI ?* Une question qui mène tout naturellement à parler des **Web Services** et de l'**Architecture Orientée Services**. Ces questions préliminaires sont nécessaires pour préparer le terrain aux sections suivantes.

Dans un second temps, l'architecture prévue pour le PMS Quotes définitif sera présentée grâce à différents schémas visuels.

La dernière partie du travail de séminaire prendra la forme d'un programme prototype fonctionnel. Ce programme devra encore être retravaillé dans les mois à venir pour satisfaire pleinement les exigences de la BSU, mais il était néanmoins important d'atteindre un premier niveau utilisable à l'issue de ce travail de séminaire.

1.3. Méthodes de travail

Lorsqu'il s'agit de décrire un programme orienté objets, il existe un langage standard particulièrement efficace qui est à la programmation ce que le solfège est à la musique : **Unified Modeling Language (UML)**. C'est lui qui sera employé pour décrire l'architecture du PMS Quotes, ainsi que diverses parties du SI de la BSU en lien avec le projet. Parmi les nombreux diagrammes proposés par UML, les **Use Cases (cas d'utilisation)** ont été sélectionnés puisqu'ils couvrent tous les besoins descriptifs nécessaires dans le présent travail [Morley 2006]. Le **Modèle Relationnel** a été choisi afin de décrire les liens entre les différentes tables SQL utilisées.

Le prototype fonctionnel, enfin, est réalisé grâce au *framework* **.Net**, puisque la BSU a toujours favorisé les technologies Microsoft. Le SGBD employé est **SQL Server**, pour des raisons similaires - la BSU se reposant doré et déjà sur ce système - alors que le programme lui-même est rédigé en **C#**, préféré à Visual Basic en raison de sa ressemblance au langage Java.

2. Échange de données entre systèmes d'information

2.1. Sources de données

L'objectif de cette partie est de déterminer d'où viennent les cours employés par le PMS et surtout, comment ceux-ci sont amenés jusqu'au SI de la BSU.

2.1.1. Historique des fournisseurs de données

Voici un historique des partenaires de la BSU ayant offert, d'une manière ou d'une autre, l'accès aux valeurs de différents cours par le passé :

- À la "Préhistoire" du PMS, avant l'introduction d'une importation automatique des cours, ceux-ci étaient introduits manuellement dans le jeu par les Administrateurs qui tiraient leurs informations de leur lecture assidue des journaux.
- Une prémisses d'importation automatique vint de Quotelines³, qui faisait parvenir un fichier à la BSU qu'il fallait ensuite importer.
- Jusqu'en 2008, la BSU collaborait directement avec la **SWX Swiss Exchange**⁴, grâce à un accord plus ou moins tacite. La SWX nous permettait d'accéder à leur donnée via un programme appelé **Financial Quotes Service (FQS)**, dont les cours provenaient de la **Swiss Market Feed (SMF)**. FQS demandait de ses utilisateurs qu'ils se connectent aux serveurs de la SWX par une connexion telnet, utilisait une syntaxe propre [FQS 2009] et retournait les résultats via le terminal, dans un format qui n'avait rien de standard.

FQS est un des services qui a été arrêté lors de la fusion de différentes institutions financières suisses et du changement du nom SWX Swiss Exchange pour **SIX Swiss Exchange**, en 2009.

- Pour le PMS 2007, la BSU avait eu accès à un nombre important de licences **Telekurs ID**, offertes par l'entreprise **Telekurs**. Telekurs ID est un programme professionnel d'information financière. L'idée était d'offrir aux joueurs un programme attractif et puissant pour leur permettre d'envisager leurs investissements virtuels, tout en les

³ Quoteline a lancé son propre jeu de simulation boursière. [Quoteline 2009]

⁴ La SWX Swiss Exchange était le nom de la place boursière suisse. Celle-ci a changé de nom pour SIX Swiss Exchange en 2008. [FQS 2009]

familiarisant avec les outils des professionnels du métier. L'édition 2010 du PMS renouvellera très probablement cette idée.

Bien qu'il aurait pu être envisageable d'employer Telekurs ID pour importer des cours, le programme est essentiellement destiné à un utilisateur humain, et n'est pas prévu pour l'interopérabilité entre programmes distants.

- Suite à la fermeture du service FQS en 2009, la BSU a cherché de nouveaux partenaires aptes à lui fournir les données. L'un d'entre eux fut le **VWDgroup**⁵ [VWDgroup 2009], qui proposait différents programmes d'information financière, à l'instar de Telekurs. Parmi ceux-ci, le service **iFeed** aurait pu aisément remplacer FQS, car il était prévu pour permettre à un SI de recevoir des données en format **Comma Separated Value (CSV)**⁶. Malheureusement, suite à diverses complications, la BSU n'a jamais réellement pu profiter d'iFeed, VWDgroup fournissant en lieu et place un fichier CSV contenant les cours de la veille chaque matin.

2.1.2. Le Futur

À l'aube de l'édition 2010, la BSU envisage de relancer le partenariat avec SIX Telekurs⁷ afin d'accéder aux données financières. Si les canaux d'échange sont encore à définir, il semble désormais acquis que le format de données employé sera du CSV, une décision qui n'est malheureusement pas des plus heureuses compte tenu du fait que les Web Services deviennent de plus en plus incontournables (cf. 2.1.2 : Standards d'échange et Web Services).

2.2. *Standards d'échange et Web Services*

Comme induit par l'historique présenté ci-dessus, les possibilités d'échange entre SI sont nombreuses, aussi le programmeur peine-t-il fréquemment lorsqu'il s'agit de faire fonctionner main dans la main deux applications logicielles distantes. C'est ce qui explique qu'aujourd'hui, l'importance d'un nombre limité de standards d'échange se fait de plus en plus ressentir.

⁵ Ancienne filiale de FIDES, rachetée intégralement par une société basée en Allemagne.

⁶ Un format de données simple, séparant chaque valeur par des virgules ou des point-virgules, et chaque entrée par des retours de ligne.

⁷ Après sa fusion avec SIX, Telekurs fut rebaptisée SIX Telekurs.

Les points suivants vont traiter d'un concept relativement récent, en constante évolution, qui est appelé à fédérer toutes les industries d'ici quelques années : les **Web Services** (Services Web, en français)...

2.2.1. Définition et fonctionnement des Web Services

Parler de Web Services, c'est d'abord parler de **Service Oriented Architecture (SOA - architecture orientée service)**. Cette architecture est une nouvelle étape vers la modularité et la diminution de l'interdépendance entre les systèmes informatiques, grâce à des couplages externes lâches. À la différence de la programmation orientée objet, SOA promeut la réutilisation des composants logiciels au niveau macro [Piette 2006, p.4]. L'implémentation d'une SOA implique différents prérequis pour une application (appelée "service" dans le cadre d'une SOA) qui offre ses fonctionnalités à une autre (appelée "client" dans le cadre d'une SOA) [Piette 2006] :

- Le service ne doit pas laisser voir la manière dont il est implémenté. Il peut être codé dans n'importe quel langage, s'exécuter sur n'importe quelle plateforme ; Cela ne doit être d'aucun intérêt pour le client.
- Le service doit implémenter une interface avec laquelle le client peut interagir.
- Le service doit être autonome, capable de fonctionner même si ses clients sont modifiés ou inaccessibles.
- Le service doit respecter différents contrats.

Les Services sont donc des applications mettant en oeuvre tous les concepts d'une SOA : deux applications s'échangent des données, sans que celle qui les demande ne sache quoique ce soit sur le fonctionnement de celle qui les lui offre, alors que cette dernière peut tout à fait fournir ses informations à différentes applications clientes. Tout ceci sans la moindre intervention humaine, comme l'indique la Figure I.

En quelques phrases, voici une définition claire de ce qu'est un Web Service : *“Les Web Services fournissent une manière standard de faire interagir différentes applications logicielles qui peuvent tourner sur diverses plateformes et/ou Frameworks. Les Web Services sont caractérisés par leur grande interopérabilité et leur extensibilité, ainsi que par leur description compréhensible par des machines grâce à l'emploi de XML. Ils peuvent être combinés par des couplages lâches pour réaliser des opérations*

complexes. Des programmes fournissant des services simples peuvent interagir les uns avec les autres pour offrir des services sophistiqués d'une valeur ajoutée" [W3C 2009].

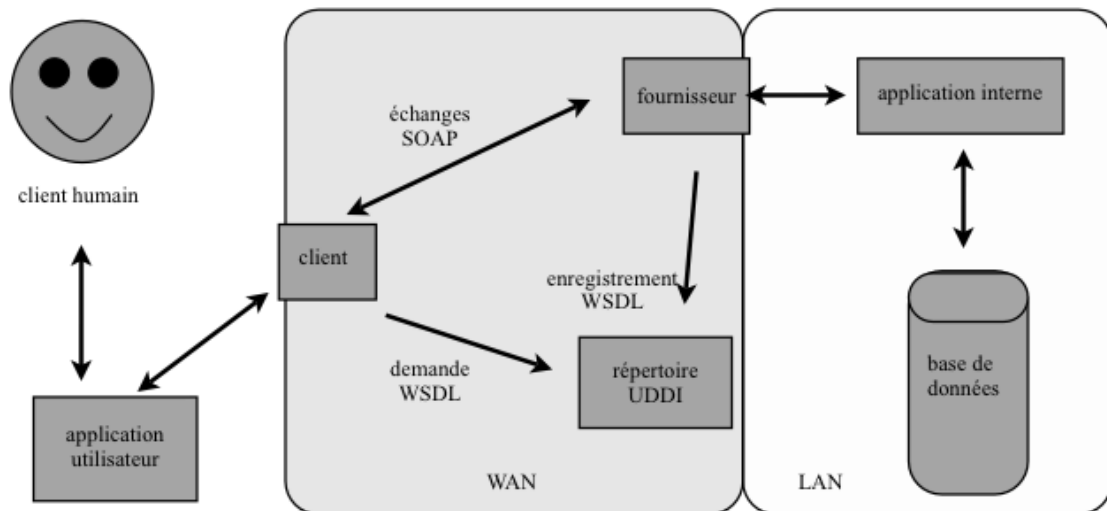


Figure I : Fonctionnement d'un Web Service WS-*. Une application cliente et une application service échange des données sur un réseau étendu.

2.2.2. Différents types de Services

Il existe plusieurs types de Services, utilisant des technologies différentes. Pour certains, ces deux types entrent directement en concurrence, là où d'autres y voient une sorte de complémentarité. Voici une description succincte de chacune d'entre elle.

- **Representation State Transfer (REST)**, un système qui emploie les mêmes technologies que celle du web. Les échanges se font par l'intermédiaire du protocole HTTP et de l'emploi d'URI. En d'autres termes, un Service REST utilise des méthodes HTTP (GET, POST, PUT, DELETE) pour faire interagir deux applications logicielles distantes.
- **Web Services WS-***, utilisant entre autre trois composants principaux :
 - **Web Service Definition Language (WSDL)**, est un langage basé sur XML qui offre un modèle de définition des Web Services. Il fut développé par IBM, Microsoft et Arriba. La version 1.1 n'a pas été acceptée comme recommandation officielle du W3C, là où la version 1.2 y est parvenue. Cette dernière fut changée en 2.0 en raison du nombre important de modifications apportées [W3C 2007]. Ce langage propose des indications pour accéder à un Web Service donné (protocole, URI, etc). La Figure II propose un aperçu des différents composants du langage

WSDL : d'un côté, des informations abstraites telles que le type de données ou la description du Web Service, d'un autre, des informations concrètes telles que les protocoles à employer.

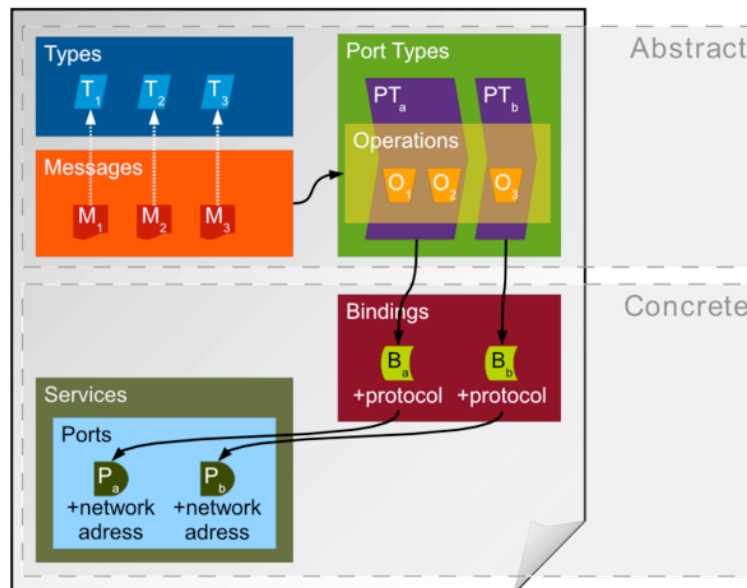


Figure II : Fonctionnement de WSDL (basé sur WSDL 1.1) [Wikipedia, 2008].

Les informations concrètes et abstraites sont séparées, chaque information abstraite menant à un lien concret réutilisable.

- **Universal Description Discovery and Integration (UDDI)**, est un annuaire basé sur le XML, créé par Organization for the Advancement of Structured Information Standards (OASIS) et recensant une liste de Web Services [OASIS 2007].
- **Simple Object Access Protocol (SOAP)**, est un protocole qui “*permet la transmission de messages entre objets distants, ce qui veut dire qu’il autorise un objet à invoquer des méthodes d’objets physiquement situés sur un autre serveur*” [Wikipedia, 2009]. Développé à l’origine par Microsoft, SOAP a été repris par le W3C et en est actuellement à sa version 1.2.

REST a l’avantage d’être plus facile à mettre en place que les Web Services utilisant SOAP, puisqu’il suffit d’avoir des connaissances web de base pour pouvoir l’utiliser. Par contre, les performances et la stabilité de SOAP sont généralement supérieures, d’autant que des outils de développement performants permettent de faciliter l’implémentation d’une telle architecture [Erl 2009].

2.2.3. Choix du type de Service pour le PMS

Aucun des partenaires de la BSU ne propose, à l'heure actuelle, des Web Services permettant le téléchargement de leurs cours. Une SOA est donc difficilement envisageable pour le PMS Quotes dans ces conditions.

Fort heureusement, si la partie externe du programme ne pourra pas être implémentée de la sorte, l'infrastructure interne du PMS va être repensée sous forme de Web Services. En effet, dans le cadre des évolutions du PMS Top, un des projets mis sur pied est de convertir l'ensemble des ressources du PMS en Web Services, offrant par conséquent un maximum d'indépendance entre chacune d'entre elle⁸.

Tout porte à croire que la BSU favorisera un Web Service type SOAP, en raison des outils de développement performants à disposition et de la robustesse des messages envoyés.

Le PMS Quotes fonctionnera par conséquent comme un service à l'égard de reste du PMS.

⁸ Voir travail de Bachelor de Mathias Büschi

3. PMS Quotes

Cette partie va aborder tout ce qui concerne directement le projet du PMS Quotes. Après avoir vu quels étaient les avantages des Web Services et quels étaient les problèmes posés par nos partenaires providers, il est possible de présenter l'architecture actuelle du PMS Quotes et ce vers quoi le projet va tendre dans les mois à venir.

3.1. Aperçu de la Base de Données de la BSU

Il est important d'avoir un aperçu de l'architecture de la base de données de la BSU pour saisir les sections suivantes. Celle-ci s'articule autour de 12 tables. La Figure III présente le modèle relationnel de la base de données de la BSU tel qu'il est vu dans SQL Server. Voici une description succincte de chaque table :

- **Kategorie** : Regroupe les différentes catégories de titres négociables dans le PMS.
- **Titel** : Recense l'intégralité des titres négociables pour le PMS en cours. Le champ "ID" sert de clé primaire, bien que les champs "ImportName" et "ValorenNr" proposent eux aussi des identifiants uniques.
- **Title_Option_Detail** : Cette table fournit des détails supplémentaires concernant les Options parmi les titres négociables de la BSU. En effet, FQS ne permettait pas de sélectionner les options directement grâce à un identifiant unique, il fallait par conséquent fournir toute une batterie d'informations pour accéder à une valeur en particulier.
- **Uni** : Enregistre les différentes Hautes Écoles et Universités d'où proviennent les membres.
- **Portfolio** : Enregistre les acquisitions de chaque joueur à chaque période.
- **Sprache** : Contient les différentes langues dans lequel la memberzone du PMS est accessible.
- **Gruppe** : Recense tous les joueurs du PMS.
- **Mutation** : Enregistre toutes les transactions effectuées par les joueurs pour chaque période.
- **Kurs** : Contient les valeurs de chaque titre pour chaque période.

- Kurs_Detail : Sert de passage intermédiaire entre l'importation des données depuis un provider et leur affichage sur le jeu du PMS.
- Periode : Compte tous les jours où la bourse suisse est ouverte lors du PMS.
- Bank : Tient à jour la somme d'argent totale de chaque joueur pour chaque jour.

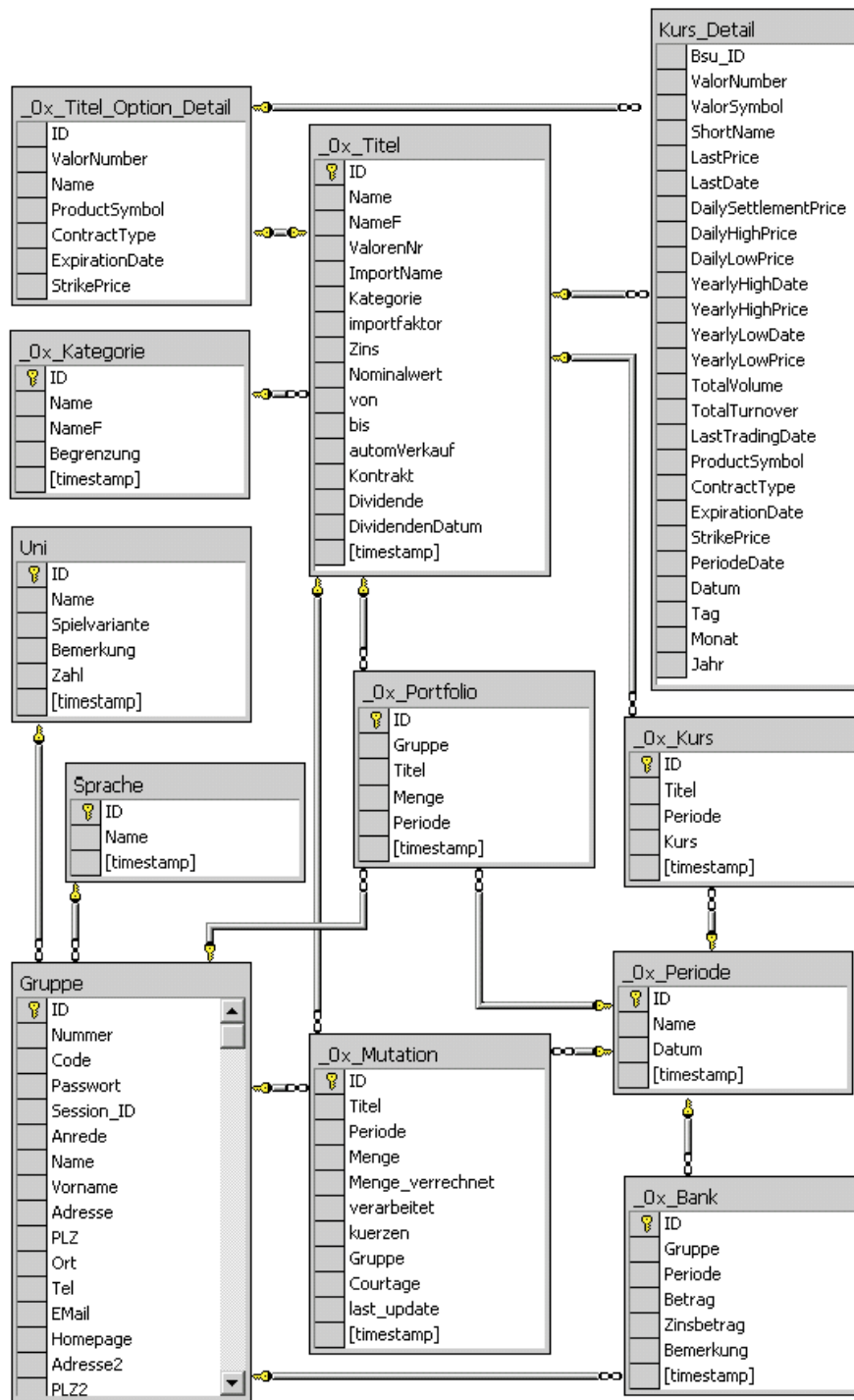


Figure III : Schéma Relationnel de la base de données BSU [Hüsemann 2007].

3.2. L'ancêtre du PMS Quotes

L'ancien programme d'importation de la BSU a été rédigé en Java. Il était activé tous les soirs à la fermeture de la bourse à l'aide d'une Scheduled Task⁹. Voici, dans les grandes lignes, les étapes qu'effectuait le programme :

- Connexion Telnet au serveur de la SWX
- Recherche de la liste des cours négociables pour le PMS
- Envoi des requêtes pour chaque cours, conformément à la syntaxe de FQS
- Réception et traitement des données
- Insertion du résultat dans la table intermédiaire "Kurs_Detail"

Les données renvoyées par la SWX prenaient la forme de nombreuses lignes de texte, dont les différentes valeurs étaient séparées par des tabulations. Le programme scindait donc les lignes reçues à chaque tabulation, et insérait le contenu obtenu dans les champs de la table Kurs_Detail.

Une fois les données insérées dans la table Kurs_Detail, un administrateur devait actionner manuellement la conversion et le transfère des données vers la table Kurs. La séparation entre ces deux phases de l'importation permettait une vérification humaine systématique des valeurs importées par le programme Java.

Malheureusement, certains cours n'étaient pas importés correctement : entre les importations en monnaie étrangère et l'absence de certaine valeur pour certains cours, les bugs étaient relativement fréquents.

Mais outre d'un programme dont la robustesse le rendrait capable de détecter les erreurs de cours, la BSU avait également besoin d'un programme permettant l'importation capable de s'adapter si jamais son fournisseur de données devait changer.

3.3. Architecture du PMS Quotes

Ce chapitre touche directement au coeur du PMS Quotes : la base de données de la BSU ayant été présentée et les différents problèmes du précédent programme d'importation

⁹ Tâche programmable, une fonctionnalité de Windows permettant de lancer un fichier à une ou plusieurs dates précises.

identifiés, il est désormais possible de poser les besoins du nouveau programme et d'établir ce à quoi il est appelé à ressembler.

3.3.1. Analyse des besoins du PMS Quotes et cas d'utilisation

La BSU propose déjà un modèle des cas d'utilisation relativement poussé¹⁰. Parmi les nombreux cas d'utilisation, seuls deux concernaient directement l'importation des cours : le cas d'utilisation UC3 "Configurer Importation" et le cas d'utilisation UC26 "Importer les cours de la SWX". Ces deux cas ont été renommés respectivement en "Configurer PMS Quotes" et "Exporter les cours". Ensuite, un troisième cas vient se greffer aux deux premiers : le cas d'utilisation UC27 "Importer les cours depuis les proviers". De ces trois cas d'utilisation découlent toutes les fonctionnalités du PMS Quotes, qui vont être analysés plus en profondeur dans les points suivants.

Dans les définitions qui suivent, deux acteurs sont différenciés :

- **Administrateur** : Membre de l'équipe d'administration BSU, en charge de la maintenance quotidienne du jeu. Un des administrateurs aura pour responsabilité de configurer le jeu à son commencement.
- **PMS Core** : Programme central du PMS, qui sert de relais pour toutes les autres applications nécessaires au bon fonctionnement du jeu.
- **Automate** : Programme ou extension du PMS Quotes qui lancera quotidiennement l'importation locale des données.
- **Providers** : Fournisseur des informations concernant les titres jouables dans le PMS.

Ces trois cas d'utilisation, liés entre eux comme le montre la Figure IV, décrivent la totalité des fonctionnalités nécessaires au PMS Quotes.

Les cas d'utilisation qui suivent décrivent les scénarios alternatifs en indiquant le pas alternatif par lequel ils commencent dans leur titre. La numérotation reprend alors à partir de 1, et la dernière ligne du cas précise si le cas d'utilisation s'arrête là ou reprend à une autre étape du scénario principal.

¹⁰ La liste complète des cas d'utilisation du PMS sont présentés dans le travail de séminaire d'Elira Shehu. [Shehu 2008]

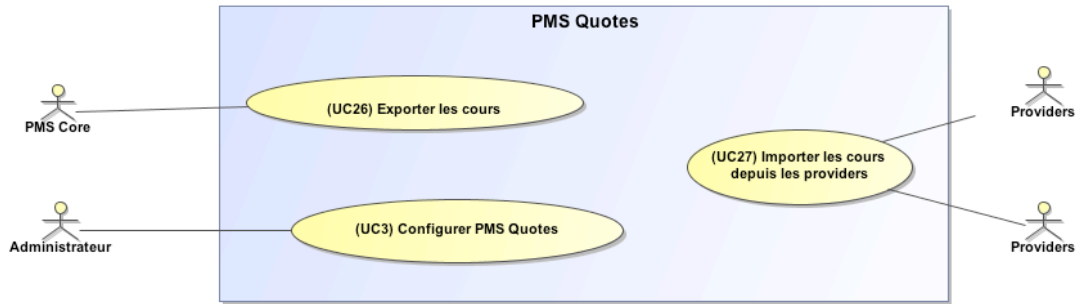


Figure IV : Diagramme de cas d'utilisation “PMS Quotes” au sein du paquet “Calcul”.

Détails du cas d'utilisation UC3 Configurer PMS Quotes

Description : Un Administrateur souhaite configurer le PMS Quotes de sorte à ce que le programme puisse déterminer les titres à importer et les providers à interroger. Il se connecte sur l’interface graphique le lui permettant (vraisemblablement une page web) et fait ses choix à l’aide de menus déroulants.

Les choix sont alors enregistrées dans la base de données du PMS Quotes.

Acteur : Administrateur

Préconditions :

- La Table “Titel” du PMS est configurée correctement
- La Table “Date” du PMS est également configurée
- La Table “PMSQuotes_Providers” du PMS Quotes est également configurée

Postcondition :

En cas de succès

- Les données sélectionnées par l’Administrateur sont enregistrées dans la table “PMSQuotes_Configuration”.

En cas d’échec à cause d’une table non remplie

- Le système reste dans son état précédent.

Actions :

Scénario principal, en cas de succès

1. L’Administrateur se connecte sur l’administration du PMS Quotes.

2. Le PMS Quotes retourne une page d'administration.
3. L'Administrateur choisit le provider dans un menu déroulant.
4. L'Administrateur choisit les types de titres voulus pour chaque provider.
5. L'Administrateur valide les choix en appuyant sur le bouton "Valider".
6. Le PMS Quotes enregistre les choix de l'Administrateur dans la base de données et retourne un message de confirmation.
7. L'Administrateur valide le message de confirmation. Fin du cas d'utilisation.

Scénario alternatif, si une table n'est pas configurée. Début au pas 2.

1. Le PMS Quotes affiche page d'erreur, stipulant que la base de données nécessaire à son fonctionnement n'est pas correctement configurée.
2. Le PMS Core reçoit un message d'erreur à afficher sur la page principale de l'administration.
3. L'Administrateur confirme le message d'erreur et retourne sur la page principale de l'administration. Fin du cas d'utilisation.

Scénario alternatif, si le jeu est déjà lancé. Début au pas 2.

1. Le PMS Quotes affiche la page de configuration, accompagnée d'un avertissement stipulant que le jeu est déjà lancé et que des modifications pourraient entraîner des dysfonctionnements du système.
2. L'Administrateur confirme qu'il souhaite bien faire les modifications. Retour à l'étape 3 du scénario principal. L'utilisateur peut également refuser, ce qui met un terme au cas d'utilisation.

Besoins non-fonctionnels :

- Le système doit être capable de gérer des changements en cours de jeu, dans l'optique où un provider viendrait à bloquer (c'est l'un des objectifs principaux du PMS Quotes).
- L'Administrateur choisit d'abord un provider, puis sélectionne le type de titres que ce dernier peut lui proposer.

Détails du cas d'utilisation UC27 Importer les cours depuis les providers

Description : l'Automate déclenche quotidiennement l'importation des cours depuis les providers sélectionnées lors de la réalisation du cas d'utilisation UC3. Le PMS Quotes contacte alors les providers et leur fournit la liste des titres désirés. Il les reçoit, les vérifie et les traite de sorte à les exporter vers sa base de données. L'importation peut être lancée plusieurs fois au cours d'une seule journée.

Acteur : Automate

Acteur secondaire : Providers

Préconditions :

- L'UC3 a été effectué

Postcondition :

En cas de succès

- Les valeurs des cours sont enregistrées dans la table "PMSQuotes_Kurs_Detail" et prêtes pour une exportation vers le PMS.

En cas d'échec à cause de l'impossibilité de joindre un ou plusieurs provider(s)

- Un message d'erreur stipulant le problème est envoyé vers l'administration du PMS.
- Les titres des providers contactables sont importés dans la base de données.

En cas d'échec à cause d'une corruption d'un ou de plusieurs titres

- Un message d'erreur stipulant le problème est envoyé vers l'administration du PMS.
- Les titres ayant pu être correctement importés sont traités et enregistrés dans la base de données.

En cas d'échec à cause de l'impossibilité de joindre la base de données du PMS Quotes

- Un message d'erreur stipulant le problème est envoyé vers l'administration du PMS.

Actions :

Scénario principal, en cas de succès

1. L'Automate appelle l'importation des cours depuis les providers du PMS Quotes.
2. Le PMS Quotes sélectionne la liste des titres désirés.
3. Le PMS Quotes sélectionner les providers désirés.
4. Le PMS Quotes se connecte aux différents providers en suivant les indications fournies à leur sujet dans la base de données.
5. Les providers valident la connexion et offrent au PMS Quotes un accès à leurs informations.
6. Le PMS Quotes envoie la liste des titres désirés aux providers.
7. Les Providers traitent la liste en cherchant les titres demandés.
8. Le PMS Quotes réceptionne la liste et vérifie son contenu.
9. Le PMS Quotes enregistre le contenu dans la base de données. Fin du cas d'utilisation.

Scénario alternatif, si un ou plusieurs providers ne sont pas contactables. Début, au pas 4.

1. Le PMS Quotes émet un message d'erreur pour chaque provider impossible à joindre.
2. Le PMS Quotes réceptionne la liste des providers ayant pu être joints et vérifie son contenu. Retour au pas 9 du scénario principal.

Scénario alternatif, si un ou plusieurs titres sont corrompus. Début au pas 8.

1. Le PMS Quotes émet un message d'erreur pour chaque titre corrompu.
2. Le PMS Core reçoit des messages d'erreur à afficher sur la page principale de l'administration du jeu. Retour au pas 9 du scénario principal.

Scénario alternatif, s'il est impossible d'enregistrer les données dans la base de données du PMS Quotes. Début au pas 9.

1. Le PMS Quotes émet un message d'erreur urgent pour indiquer que l'enregistrement des données est impossible.

2. Le PMS Core reçoit un message d'erreur à afficher sur la page principale de l'administration. Fin du cas d'utilisation.

Besoins non-fonctionnels :

- Le traitement des données doit s'effectuer en moins de 5 minutes.
- L'importation doit pouvoir fonctionner à n'importe quelle heure.
- L'importation doit pouvoir être effectuée plusieurs fois par jour.

Détails du cas d'utilisation UC26 Exporter les cours

Description : Un Administrateur souhaite effectuer le calcul du jeu. Le PMS contacte alors le PMS Quotes afin d'obtenir les cours des titres du jeu pour une date donnée. Le PMS Quotes convertit les données demandées et les exporte vers le PMS.

Acteur : PMS Core

Préconditions :

- La Table "Date" du PMS est correctement configurée
- L'UC27 a été effectué pour la date demandée

Postcondition :

En cas de succès

- Les données demandées par l'Administrateur sont enregistrées dans la base de données du PMS.

En cas d'échec, si l'Administrateur indique une date erronée

- Un message d'erreur est retourné à l'administration du PMS.

En cas d'échec, si un ou plusieurs des cours importés ont été signalés comme erroné(s) lors de l'UC27.

- Un message d'avertissement est retourné à l'administration du PMS.
- Les données demandées par l'Administrateur sont enregistrées dans la base de données du PMS.

Actions :

Scénario principal

1. Le PMS Core envoie la date désirée au PMS Quotes.
2. Le PMS Quotes réceptionne les informations et cherche la liste des titres.
3. Le PMS Quotes convertit les données de la table en message SOAP.
4. Le PMS Quotes envoie le message contenant la dernière valeur de chaque cours pour la date correspondante au PMS Core. Fin du cas d'utilisation.

Scénario alternatif, si le PMS Quotes n'est pas joignable. Début au pas 2.

1. Le PMS Core affiche un message d'erreur urgent sur la page principale de l'administration. Fin du cas d'utilisation.

Scénario alternatif, si la date envoyée par l'Administrateur est erronée. Début au pas 3.

1. Le PMS Quotes retourne une erreur au PMS Core si aucune valeur ne peut être trouvée pour la date spécifiée.
2. Le PMS Core affiche un message d'erreur sur la page principale de l'administration. Fin du cas d'utilisation.

Scénario alternatif, si un ou plusieurs cours ont été marqués comme erronés lors de l'UC27. Début au pas 3.

1. Le PMS Quotes retourne un avertissement pour prévenir du fait que certains cours sont probablement erronés et communique la liste des titres concernés.
2. Le PMS Core affiche un message d'erreur contenant la liste des titres corrompus sur la page principale de l'administration. Retour à l'étape 3 du scénario principal.

Besoins non-fonctionnels :

- Un Administrateur peut demander l'importation des titres à n'importe quelle date déjà passée (dans l'optique de restaurer des anciennes données, par exemple).
- Le PMS Quotes fonctionne comme un Web Service vis-à-vis du PMS. Les messages sont donc envoyés au format SOAP.

- Le PMS Quotes doit pouvoir transmettre ses données plusieurs fois par jour, à n'importe quelle heure.

3.3.2. Modules du PMS Quotes

Après la description des cas d'utilisation ci-dessus, on comprend que le PMS Quotes pourra être scindé en divers modules, comme présenté dans la Figure V. Ces modules représentent des objets facilement identifiables dans le cadre d'une programmation orientée objets.

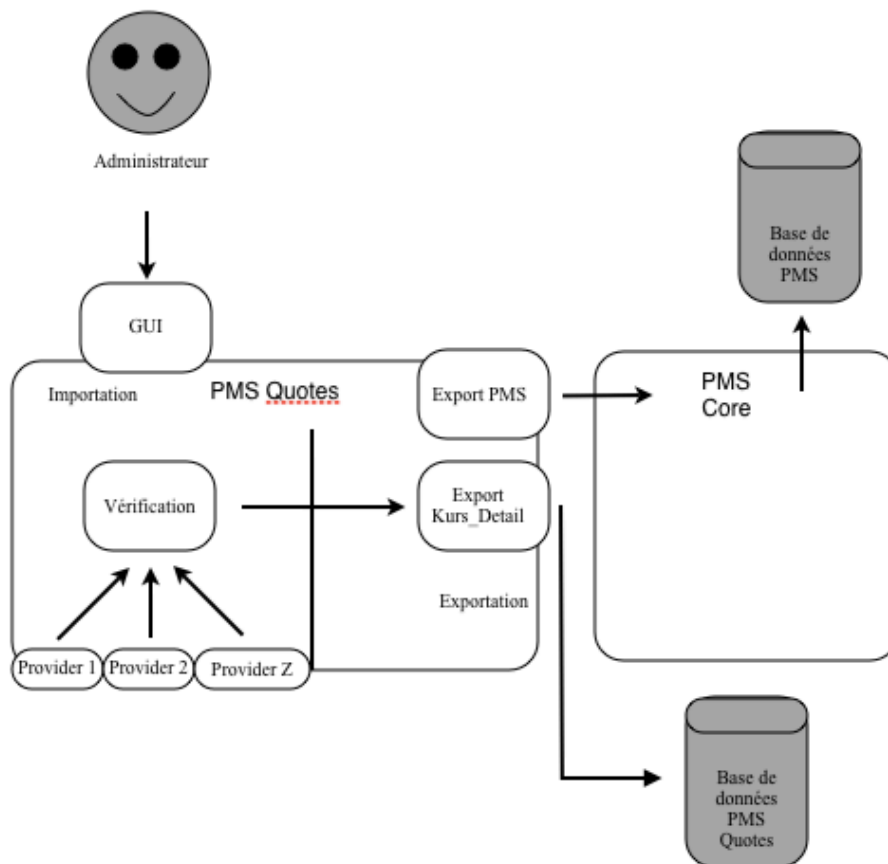


Figure V : Illustration des différents modules du PMS Quotes et de leur intégration au sein du PMS.

- **GUI de configuration** : Page ASP.Net permettant la configuration du PMS Quotes : providers utilisés, titres importés, etc. Il implémente le cas d'utilisation UC3.
- **Module "Providers"** : Ce module implémente la première partie du cas d'utilisation UC27 en se chargeant de l'importation des données depuis les providers choisis. Il se connecte à ceux-ci selon les protocoles spécifiques à chaque provider, réceptionne les

données et les encapsule, champ par champ, dans un DataSet¹¹ qu'il envoie au module de vérification.

- **Module “Vérification”** : Ce module est la deuxième partie de l'implémentation du cas d'utilisation UC27. Il réceptionne le DataSet envoyé par le module “Providers”, et applique une série de vérifications pour s'assurer qu'aucune valeur n'est manquante ou erronée (en se connectant notamment à la table Kurs_Detail pour vérifier si les titres précédents n'ont pas trop augmenté). Dans le cas contraire, le module génère un message d'erreur s'affichant sur le tableau de bord de l'administration du PMS.
- **Module “Exportation vers Kurs_Detail”** : Ce module implémente la troisième et dernière partie du cas d'utilisation UC27. Une fois les données validées par le précédent module, le module “Exportation vers Kurs_Detail” introduit les données dans la table “PMSQuotes_Kurs_Detail”.
- **Module “Exportation vers PMS”** : Ce module intervient en dernier, et n'est pas appelé par un autre module du PMS Quotes. Il implémente le cas d'utilisation UC26. Il entre en jeu lorsque le programme du PMS effectue l'importation quotidienne vers la base de données finale. Les données désirées sont alors extraites de la table Kurs_Detail, et transformées en message SOAP, réceptionné par le programme client comme le veut la logique des Web Services (voir 2.1.2.1, “Définition et Fonctionnement des Web Services”).

3.3..3 Architecture de la base de données du PMS Quotes

Le PMS Quotes nécessite la création de cinq tables afin de fonctionner correctement. La Figure VI montre le Schéma Relationnel les reliant les unes aux autres, même si deux des cinq tables fonctionnent indépendamment des autres.

Une table “**PMSQuotes_Configuration**” doit être générée afin de contenir tous les choix des administrateurs de la BSU pour le PMS en cours. Elle contient différents champs permettant le bon fonctionnement de l'importation : taux maximal acceptable d'augmentation de la valeur d'un titre, chemin du dossier où réceptionner les données transmises fournies par les fournisseurs, etc.

¹¹ Classe propre au framework .Net, permettant de traiter facilement une liste de données comme s'il s'agissait d'une table. [Microsoft 2009]

La table “**PMSQuotes_Providers**” renferme toutes les informations sur tous les providers. Son rôle ressemble quelque peu à celui de WSDL (cf. 2.1..2.2) : il s’agit d’indiquer quel protocole, quel port et quelle adresse employer pour se connecter au Provider. En outre, pour fonctionner, l’importation doit savoir quels types de cours propose chaque Provider.

C’est pour cette raison que les tables “**PMSQuotes_Type_Titles**” et “**PMSQuotes_Provider_Types**” ont été créées : la première recense chaque type de titres alors que la seconde fait le lien entre les providers et les types qu’ils se proposent de fournir.

Enfin, une table “**PMSQuotes_Kurs_Detail**” contient les données temporaires automatiquement importées chaque jour. Il s’agit simplement d’une copie de la table “Kurs_Detail” de la base de données de la BSU.

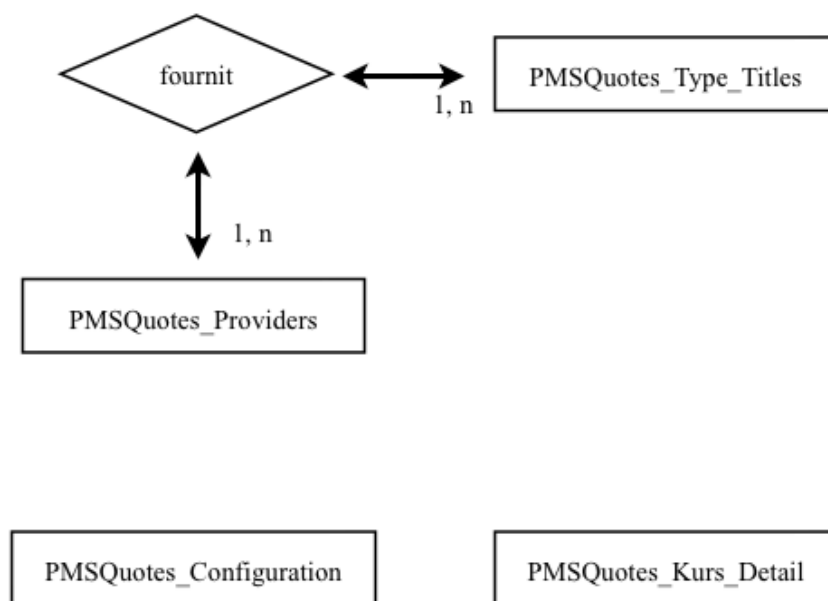


Figure VI : Schéma relationnel de la base de données du PMS Quotes.

3.3.4. Extensions possibles et problèmes

Comme indiqué dans les cas d’utilisation, l’importation des cours depuis le PMS Quotes n’est pas entièrement indépendante de la base de données de la PMS. Cette dépendance remet en cause la définition du PMS Quotes en tant que Web Service, puisque le programme serait incapable de fonctionner si jamais la base de données de la BSU devait tomber en panne. Parmi les solutions possibles, on pourrait imaginer que

l'importation ne se fasse que lorsque le PMS Core le demande, et qu'il envoie lui-même, sous forme de message SOAP, la liste des titres qu'il souhaite importer. Cependant, cette proposition suggérerait une lourde modification du modèle présenté dans ce travail, puisque l'importation temporaire vers la table "PMSQuotes_Kurs_Detail" deviendrait obsolète.

Une autre fonctionnalité intéressante aurait été de permettre l'importation des titres plusieurs fois par jour. Une telle possibilité serait relativement facile à implémenter sur le modèle présenté ci-dessus, en modifiant par exemple la table "PMSQuotes_Configuration".

Finalement, ce travail fait souvent référence à un panneau d'administration du PMS. Celui-ci n'existe pas encore, et risque de voir le jour dans le cadre d'un des sous-projets du PMS Top. Tant que sa forme n'a pas été décidée, il est impossible de décrire réellement par quelle méthode le PMS Quotes le contactera, mais il se peut que ladite méthode influence passablement le modèle présenté dans ce travail de séminaire (en rajoutant, par exemple, un module supplémentaire).

4. Prototype PMS Quotes

Le PMS Quotes étant un projet dont la taille est conséquente, sa réalisation complète dans le cadre de ce travail de séminaire aurait été impensable. Au lieu de cela, un prototype a été créé, sur la base duquel le vrai programme pourra être développé dans les mois à venir (les détails sur les technologies employées pour la réalisation du programme sont exposés au chapitre 1.3).

4.1. Différences avec le PMS Quotes et limitations

Ce prototype implémente les modules “Providers” et “Exportation vers Kurs_Detail”, ainsi que les tables “Kurs_Detail”. Les autres parties du programme seront implémentées hors du cadre du travail de séminaire.

À l’heure actuelle, le prototype ne prend en charge qu’un seul type de fichier - des fichiers CSV modifiés à partir de ceux envoyés par le VWDgroup pour le PMS 2009.

4.2. Extraits notoires du Code Source

4.2.1. Importation du fichier CSV

Cette partie traite de la première partie du prototype, à savoir, l’importation du contenu d’un fichier CSV. D’autres types de fichiers seront pris en compte dans la version définitive du PMS Quotes.

Architecture d’un fichier CSV

Voici à quoi ressemble une série de ligne d’un des fichiers CSV utilisés par le Prototype¹².

```
3. TIC,EXC,TITLE,ISIN,DOMNSIN,TCUR,CLOSE,DATEC
4. AAM,SWX,ANGLO PLC,GB00B1XZS820,CH3186826,CHF,27,08/05/09
5. ABBN,SWX,ABB LTD N,CH0012221716,CH1222171,CHF,18.06,08/05/09
6. ABBNE,SWX,ABB LTD N 2. LINIE,CH0037934822,CH3793482,CHF,10,16/04/09
7. ABSIE,SWX,ABSOLUTE INVEST I 2L,CH0021119927,CH2111992,CHF,32.15,08/04/09
8. ABSPE,SWX,APE I 2. LINIE,CH0021971509,CH2197150,CHF,5.75,19/12/08
```

La première ligne sert d’en-tête. “TIC” représente le nom abrégé du titre, “EXC” décrit le marché auquel le titre appartient, “TITLE” fait évidemment référence au nom réel du titre, “ISIN” à son ISIN¹³, “DOMNSIN” à l’identifiant local du titre sur son marché

¹² Les fichiers originaux, émis par le VWDgroup, utilisent des point-virgules plutôt que des virgules pour séparer leurs valeurs.

¹³ “International Securities Identification Number”, un identifiant unique donné à chaque titre.

d'origine, "TCUR" à la monnaie dans laquelle le titre est côté, "CLOSE" au Last Price dudit titre et finalement, "DATEC" pour la date de la dernière négociation.

Utilisation de OLE DB

Pour ce prototype, l'importation des fichiers CSV se fait par l'intermédiaire de l'interface "OLE DB", fournie dans le Framework .Net. Il s'agit en fait est un ensemble d'interfaces permettant d'interagir avec de nombreuses sources de données en y accédant uniformément [Microsoft 2009].

Voici un extrait du code source, concernant la méthode qui importe un fichier CSV grâce à OLE DB.

```
1. string dir = Path.GetDirectoryName(path);
2. string file = Path.GetFileName(path);
3.
4. string strConnexion = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=\" + dir + "\"
  \";Extended Properties=\"text;HDR=Yes;FMT=Delimited\"";
5. string ImportQuery = "SELECT * FROM " + file;
6.
7. DataTable ImportTable = new DataTable();
8.
9. OleDbDataAdapter ImportAdapter = new OleDbDataAdapter(ImportQuery,
  strConnexion);
10.
11. try
12. {
13.     ImportAdapter.Fill(ImportTable);
14. }
15. catch (Exception e)
16. {
17.     // Envoie un message d'erreur sur le panneau d'administration PMS (à créer)
18. }
19.
20. ImportAdapter.Dispose();
21.
22. return ImportTable;
23.
```

La connexion OLE DB fonctionne ainsi :

- Les lignes 1 et 2 récupèrent le nom du dossier et du fichier, dont le chemin est envoyé en argument à la méthode courante.
- La ligne 4 est une *connexion string* formatée pour ouvrir un fichier CSV. La partie "Data Source" fait appel au chemin du dossier où se trouve le fichier spécifié en argument. La partie "HDR=Yes" précise que le fichier CSV possède une en-tête, alors que la partie "FMT=Delimited" est nécessaire pour qu'OLE DB sache qu'il traite un fichier CSV.

- La ligne 5 est une simple requête SQL, dont le nom de la table est remplacé par le nom du fichier, au sein du dossier “dir”.
- Les lignes suivantes lancent la connexion au fichier, insère chaque valeur de chaque ligne dans un objet “DataTable” et retourne ce dernier.
- Le “catch” des lignes 15 à 18 enverra un message d’erreur sur le panneau d’administration du PMS dès que celui-ci sera créé (ce qui sort du cadre du PMS Quotes).

À l’heure actuelle, seuls les fichiers CSV délimités par des virgules peuvent être traités par le prototype, même s’il existe un moyen fastidieux de faire en sorte qu’OLE DB puisse gérer d’autres délimiteurs. Mais si l’emploi d’OLE DB est une solution acceptable et fonctionnelle, tout porte à croire que la solution idéale pour l’importation de fichiers CSV serait de créer, à terme, un Parser dédié. Certains prétendent qu’il est possible d’obtenir des résultats jusqu’à quinze fois plus rapides suivant la qualité du *parser* [Lorion 2008].

4.2.2. Exportation vers “PMSQuotes_Kurs_Detail”

Cette dernière section traite de la seconde partie du prototype : le module “Exportation vers Kurs_Detail”, et la table qui va avec.

Table “PMSQuotes_Kurs_Detail”

La table “PMSQuotes_Kurs_Detail” est similaire à “Kurs_Detail” du PMS (voir Figure III), à ceci près qu’un champ a été rajouté pour permettre d’identifier les cours suspects d’être corrompus sans affecter le moindre autre champ. Il s’agit d’un simple champ “integer”, contraint à deux valeurs : 0 ou 1.

Après une importation depuis un fichier CSV, la plupart des champs restent vides. Les détails tels que “YearlyHighPrice” ne sont malheureusement pas livrés avec ledit fichier, aussi sera-t-il important, lors de la réalisation du programme définitif, de réfléchir à l’hypothèse de retirer ces champs si d’aventure nul provider n’était à même de nous les fournir.

Détail de la méthode “ExportToKursDetail”

Cette méthode traite le DataSet reçu en modifiant ses champs de manière à les rendre compatibles avec ceux de la table “Kurs_Detail”. Une fois cette opération terminée, elle utilise la classe SqlConnection pour se connecter à la base de données et y injecte simplement le résultat.

5. Conclusion

Le besoin de mettre à jour l'infrastructure de la BSU se fait de plus en plus pressant au fur et à mesure des années. La concurrence dans le domaine des jeux de simulation boursière - bien que n'étant réellement directe ni menaçante - commence à se faire sentir, et il est temps pour l'association de faire évoluer sa plateforme vers un niveau supérieur. Le PMS Quotes, et le PMS Top dans son ensemble, ne sont qu'une étape vers cette modernisation, une étape dont l'utilisateur final n'aura pas nécessairement conscience puisqu'elle ne fera que rendre les opérations internes plus performantes. Néanmoins, c'est bel et bien par une refonte de celles-ci que tout doit commencer.

Ce document a mis en évidence les fournisseurs de données à l'aide desquels la BSU va pouvoir faire fonctionner le PMS (Telekurs), ainsi qu'il brosse rapidement une description des méthodes les plus efficaces pour communiquer ces données. Sur la base de ces informations, l'architecture finale du PMS Quotes a finalement été définie, et un prototype fonctionnel sert d'exemple et de fondation pour une implémentation complète.

Même avec ce travail de séminaire terminé, il reste encore beaucoup de cheminement à réaliser pour que la BSU puisse finalement bénéficier d'un programme d'importation efficace. Néanmoins, cette étude offrira à l'équipe informatique de la BSU une orientation quant aux prochains efforts à effectuer.

6. Glossaire

- **BSU** : Börsenspiel der Schweizer Universitäten, association d'étudiants de l'Université de Fribourg.
- **CSV** : Comma Separated Value. Langage informatique générique, dont les valeurs sont séparées par des virgules.
- **.Net** : Framework de Microsoft, regroupant diverses technologies.
- **PMS** : Portfolio Management Simulation, le jeu de simulation boursière organisé annuellement par la BSU. Dans ce travail, lors des parties plus techniques, "PMS" se réfère également à l'infrastructure nécessaire au fonctionnement du PMS.
- **REST** : Type de services informatiques basé sur les technologies web. Souvent en concurrence avec les Web Services.
- **SOA** : Service Oriented Architecture. Architecture d'un groupe de programmes pouvant travailler en collaboration tout en profitant de couplages lâches.
- **UML** : Unified Modelling Language. Normes graphiques permettant de décrire des programmes orientés objets.
- **Web Services** : Type de services informatique basé sur diverses technologies.
- **XML** : Extensible Markup Language. Langage informatique générique et standard, dont les valeurs sont encadrées dans des balises.

7. Bibliographie

[Erl 2009] Erl, Thomas : *Web Service Contract Design and versioning for SOA*, 2009.

[FQS 2009] SIX : *FQS User Guide*, accessible à : <http://fqs.swx.com>, accédé le 5 janvier 2010.

[Hüsemann 2007] Hüsemann, Stefan: *BSU IT Architecture*, décembre 2007.

[Lorion 2008] Lorion, Sebastien : *CodeProject : A Fast CSV Reader. Free source code and programming help*, accessible à : <http://www.codeproject.com/KB/database/CsvReader.aspx>, accédé le 5 janvier 2010.

[Microsoft 2009] Microsoft : *MSDN France*, accessible à : <http://msdn.microsoft.com/fr-fr/default.aspx>, accédé le 31 décembre 2009.

[Morley 2006] Morley, Chantal ; Hugues, Jean ; Leblanc, Bernand : *UML 2 - Pour l'analyse d'un système d'information*, Dunod, 3e édition, 2006.

[OASIS 2007] OASIS : *OASIS UDDI Specification TC*, accessible à : http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=uddi-spec, accédé le 31 décembre 2009.

[Quoteline 2009] Quoteline : *Quoteline - Kurse, Charts, Boerseninformationen aus aller Welt*, accessible à : <http://www.quoteline.ch/default.asp?l=E>, accédé le 5 janvier 2010.

[Piette 2006] Piette, Benoit : *Introduction aux Web Services*, accessible à : <http://benoitpiette.com/labo/introduction-aux-web-services.html#page2>, accédé le 31 décembre 2009.

[Shehu 2008] Shehu, Elira : *Analyse d'un système d'information et extension de celui-ci par de nouvelles fonctionnalités*.

[VWDgroup 2009] VWDgroup : *VWD information solutions AG*, accessible à : <http://www.vwdgroup.ch/fides/markt.htm?u=0&k=0&language=GB>, accédé le 5 janvier 2010.

[W3C 2009], W3C : *Web Services Activity Statement*, accessible à : <http://www.w3.org/2002/ws/Activity.html>, accédé le 5 janvier 2010.

[W3C 2007] W3C : *Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language*, accessible à : <http://www.w3.org/TR/wsd120/>, accédé le 31 décembre 2009.

[Wikipedia 2008] Wikipedia : *Web Services Description Language - Wikipédia*, accessible à : http://fr.wikipedia.org/wiki/Web_Services_Description_Language, accédé le 31 décembre 2009.

[Wikipedia 2009] Wikipedia : *Architecture orientée services - Wikipédia*, accessible à : http://fr.wikipedia.org/wiki/Architecture_orientée_services, accédé le 31 décembre 2009.