DEPARTEMENT INFORMATIQUE - IUT 2 GRENOBLE

Année Universitaire 2008-2009

MEMOIRE DE STAGE

CONCEPTION ET REALISATION D'UN SERVICE WEB DE RECENSEMENT DE MAUVAIS CALIBRATEUR

Laboratoire d'Astrophysique de Grenoble

(Stage du 06 avril au 26 juin)

Présenté par

Nicolas Hofmann

Promotion (2A)

Jury

IUT : Mme Marie-Claude Caravel

IUT : Mr Pierre-François Dutot

Société : Mr Guillaume Mella

ref : *JMMC-TRA-2610-0001*

Remerciements :

Je tiens à exprimer ma reconnaissance à Guillaume Mella pour la qualité de l'encadrement du stage ainsi qu'à Sylvain Lafrasse pour ses conseils et Gilles Duvert pour les réponses qu'il a apportées à mes questions concernant l'astrophysique.

Je remercie par ailleurs tout le reste de l'équipe du Laboratoire d'Astrophysique de Grenoble pour l'accueil que j'y ai reçu.

Je remercierai enfin le Laboratoire d'Astrophysique de Grenoble de m'avoir offert l'opportunité de faire mon stage dans leur département.

Table des matières

Introduction

C'est pour clôturer mon DUT que j'ai passé ces deux derniers mois en tant que stagiaire au Laboratoire d'astrophysique de Grenoble. Ce rapport décrit le processus mis en place et les réflexions menés durant ce stage.

Le Laboratoire d'astrophysique de Grenoble[1] (LAOG) est un institut de recherche en astronomie, astrophysique et instrumentation. Le LAOG est situé à Saint-Martin d'Hères sur le campus de l'Université Joseph Fournier, près de Grenoble. Il est donc bien desservi par les transports en commun ce qui lui permet d'accueillir facilement stagiaires et invités, chose indispensable à la collaboration avec d'autres laboratoires. Administrativement, le LAOG est une unité mixte de recherche entre le CNRS et l'Université Joseph Fournier (UJF).

Le LAOG compte environ une centaine de personnes, et il a quatre équipes de recherches principales : ASTROMOL (astrophysique moléculaire), FOST (formation stellaire), les SHERPAS (études des phénomènes à hautes énergies, rayons gamma, rayons X, noyau actif de galaxie, quasars et micro-quasars) ainsi que GRRIL (Groupe de recherche et réalisation d'instruments du LAOG). Ce dernier développe des instruments interférométriques comme AMBER au VLT. C'est cette équipe que j'ai intégrée. Il est directement lié au Jean-Marie Mariotti Center (JMMC), qui développe des logiciels de traitement du signal interférométrique.

C'est en septembre 2000 qu'a été créé le JMMC[2] (pour Jean-Marie Mariotti Center) par l'Institut National des Sciences de l'Univers (INSU). Le JMMC est un réseau de laboratoires français plus ou moins spécialisés dans l'interférométrie optique coordonnée pas la LAOG.

La mission du JMMC est de coordonner les efforts des laboratoires membres pour offrir à tous les utilisateurs de services d'interférométrie potentiels le meilleur environnement de travail et le plus de capacité d'action possible. Sa mission consiste donc à :
-Développer, produire, documenter et maintenir les logiciels nécessaires à l'exploitation et le suivi de nouveaux équipements d'interférométrie.
-Stimuler et coordonner la formation des non spécialistes aux spécificités de ce domaine.
-Participer aux évènements concernant de nouveaux instruments destinés à l'interférométrie.
Le premier point constitue l'activité principale du JMMC et c'est celui qui justifie ma présence dans leur organisation.

A titre indicatif, le JMMC a participé à plusieurs projets d'envergure comme la conception de divers instruments destinés au VLTI (Very Large Telescope Interferometer) ou la mission DARWIN.

1 http://www-laog.obs.ujf-grenoble.fr/

2 http://www.jmmc.fr/

C'est dans ce cadre que mon projet s'inscrit. Il s'agit de mettre en place un nouveau service de base de données d'étoiles de calibration jugée mauvaise pour l'interférométrie optique[3]. Un certain nombre de services ayant cette fonction avait déjà été mis en place par d'autres organismes mais ils avaient été jugés trop peu souples pour correspondre parfaitement aux attentes des scientifiques. Il a donc été décidé d'accueillir un stagiaire pour travailler sur la question. J'ai eu la chance d'avoir été pris.

Dans la pratique, je travaillais dans la salle destinée aux visiteurs où un ordinateur de bureau m'était alloué. Je travaillais sous la supervision et avec l'aide de Guillaume Mella. Sylvain Lafrasse, bien que n'ayant pas directement participé au projet, nous exposait régulièrement ses remarques sur le travail qui était effectué. Avoir l'avis de quelqu'un d'extérieur est toujours utile dans la mesure où il peut vérifier la validité de notre raisonnement. Gille Duvert, astrophysicien, était là pour représenter les utilisateurs et répondre à nos questions concernant les préférences des utilisateurs, les degrés de précision à atteindre, le nombre d'utilisateurs estimé et toute autre donnée sur les détails du domaine. Enfin Samuel Prette, également en stage de fin de DUT, travaillait dans la même pièce que moi sur un projet différent. Nous nous connaissions déjà bien, Samuel et moi, avant le début du stage pour avoir effectué notre projet tuteuré de deuxième année ensemble (une application client serveur pour jouer au poker). Nous avions tout les deux Guillaume pour responsable. Nous nous sommes rendu mutuellement plusieurs services durant le développement. D'une manière plus générale, les membres du LAOG furent très accueillants et une petite présentation du travail qui y est effectué nous a été faite.

Concernant les conditions de vie, le LAOG n'étant pas une entreprise, elle n'a pas d'objectif de rentabilité ni besoin de planifier avec précision les taches. Rien n'est vraiment formalisé dès lors que chacun y met de la bonne volonté et que le travail avance à un rythme satisfaisant. Ainsi les horaires de travail sont très souples (pas de contrainte au niveau des pauses et un foot était même organisé le jeudi). Nous avions la possibilité de suivre les séminaires du LAOG. Nous avons assisté à la remise d'une médaille d'argent du CNRS ainsi qu'à la retransmission en direct du lancement du satellite d'observation PLANK le 14 mai depuis Kourou. Les conditions de travail n'étaient donc pas stressantes du tout. Une réunion était organisée tous les lundis avec toute l'équipe ainsi que, parfois, le directeur du LAOG pour discuter de ce qui avait était fait depuis la dernière réunion et des projets pour la suite du travail.

3 http://fr.wikipedia.org/wiki/Interf%C3%A9rom%C3%A8tre_optique_%C3%A0_longue_base

1)Analyse du Sujet

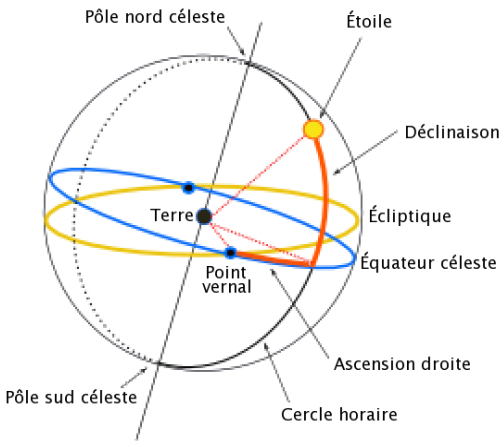
1.1)Contexte

1.1.1)Généralité du domaine

L'astrophysique, comme n'importe quel autre domaine, a certaines particularités non évidentes au néophyte qu'il convient de prendre en compte mais qui ne sont précisées que dans les cours et publications d'astrophysique. Cela est ainsi car il s'agit de notions de base qui sont supposées connues de tout astrophysicien et sont, par conséquent, sous-entendues.

1.1.1.1)Les coordonnées

Les Astrophysiciens utilisent historiquement le système de coordonnées équatoriales[4] pour localiser un astre dans le ciel. Ce système désigne un point dans le ciel en utilisant 3 valeurs: ascension droite, déclinaison et date. Il est nécessaire de préciser une date car dans l'espace tout bouge, y compris la terre. Pour faciliter la désignation d'un point, a été prise, par convention, la date « J2000 »[5]. L'utilisateur se charge des diverses conversions dont il a besoin. Ainsi, ne restent plus que l'ascension droite et la déclinaison qui sont abrégées respectivement par RA et DEC. Cette même notation sera utilisée pour tout le reste du rapport.



Ces coordonnées sont traditionnellement exprimées par les astrophysiciens en degrés sexagésimaux[6]. Notation qui subdivise l'angle en 3 champs: heure, minute, seconde. Or, ce format ne convient pas aux divers calculs et algorithmes de recherche. Il est nécessaire de disposer par un moyen ou un autre des coordonnées en degrés ou en radians pour automatiser le travail sur la base.

4 http://fr.wikipedia.org/wiki/Syst%C3%A8me_de_coordonn%C3%A9es_%C3%A9quatoriales
5 http://fr.wikipedia.org/wiki/J2000.0
6 http://fr.wikipedia.org/wiki/Num%C3%A9ration_sexag%C3%A9simale

Par ailleurs, les astrophysiciens qui travaillent dans le domaine du visible ne connaissent toujours pas avec précision les coordonnées des objets qu'ils désignent et doivent donc être assistés lors de la saisie de coordonnées s'ils le désirent.

Enfin, les Astrophysiciens ont pour habitude de rechercher des objets dans une base de donnée au travers de recherches coniques[7] (ou Cone Search), méthode qui consiste a rechercher tout les objets présents dans une base ce situant dans un certain morceau conique du ciel dont le sommet est la terre. Cette recherche utilise 3 paramètres: RA et DEC qui définissent la direction qui est le centre du cône, et l'angle de recherche ou SR (pour Search Radius) qui définit l'angle d'ouverture du cône, c'est a dire la largeur de la zone de recherche.

1.1.1.2)Les noms et les catalogues

Premier point, une même étoile peut avoir plusieurs noms. Ceci vaut pour quasiment toutes les étoiles. Ceci est du au fait qu'elles sont présentes dans plusieurs catalogues et que chaque catalogue lui donne un identifiant qui lui est propre.

Qui plus est, les étoiles n'ont pas de nom principal. Si chaque étoile a plusieurs identifiants propres à chaque catalogue, aucun catalogue n'a en mémoire toutes les étoiles du ciel et ne donne un identifiant fixe et définitif à chaque étoile. Il est donc impossible de désigner un catalogue dont l'identifiant pourrait servir à identifier formellement une étoile. Seules les coordonnées nous permettent, alors, de désigner formellement une étoile, ce qui pose le problème de leur précision.

1.1.2)L'Observatoire Virtuel

L'Observatoire Virtuel, ou IVOA pour **I**nternational **V**irtual **O**bservatory **A**lliance[8], est un organisme formé en 2002 pour faciliter la collaboration et la coordination internationale nécessaires au développement et au déploiement des outils, systèmes et structures permettant l'utilisation des bases de données astronomiques et ceci de manière interopérable.

Il a donc été créé tout un ensemble de standards permettant la-dite interopérabilité qu'il convient de respecter si l'on est amené à fournir un service qui soit facilement utilisable par la communauté. On y retrouve des moyens standards d'accéder à des données, de formater des données, de formater des méta-données, d'informer sur l'existence d'un service, d'informer sur les capacités d'un service...

7 http://www.ivoa.net/Documents/latest/ConeSearch.html

8 http://www.ivoa.net/

Un certain nombre d'outils permettant de faciliter l'utilisation des standards ont été développés par les différentes équipes qui composent l'observatoire virtuel. Leur utilisation est donc une opportunité à étudier lorsque l'un d'entre eux propose un service qui nous intéresse.

Par ailleurs, le Centre de Données astronomiques de Strasbourg[9] avait déjà mis en place un ensemble de services avant l'apparition de l'observatoire virtuel. Il utilise donc ses propres standards mais la puissance de ses services font de lui un point clé dans l'observatoire virtuel.

L'observatoire virtuel sera appelé VO(**V**irtual **O**bservatory) dans la suite de ce mémoire.

1.2)Cahier des charges

1.2.1)Besoin

Dans le cadre de leurs travaux, les astrophysiciens sont amenés à se servir d'étoiles pour calibrer leurs instruments. Il existe un très grand nombre de calibrateurs potentiels mais tout les calibrateurs potentiels ne sont pas forcément de bons calibrateurs dans les faits. On ne saurait pas non-plus limiter la recherche de calibrateurs à une petite liste d'étoiles dont les propriétés seraient confirmées. Durant une observation, on ne constate pas instantanément qu'un calibrateur est mauvais, ce qui engendre une perte de temps. Il est donc nécessaire, ou pour le moins souhaitable, de mettre en place un service permettant le recensement des calibrateurs jugés mauvais (entre autre...) afin d'éviter qu'ils ne soient utilisés dans le futur.

La nature des informations à diffuser est, par ailleurs, susceptible de changer. Il est donc primordial de faire en sorte que toute extension de la base de données puisse être faite avec pas, ou peu, de modifications du code source de l'application. Ce besoin est important et aura de grandes répercutions sur l'architecture de l'outil.

Qui plus est, l'accès à la base doit être simple, aussi bien pour des humains que pour des machines. Il est donc nécessaire que l'outil obéisse au standard du VO et propose un accès destiné à des procédures automatiques. Cela permet, en effet, une utilisation facile de la base par les outils déjà existants du VO et l'intégration du service dans des structures plus grandes et plus puissantes, notamment le service SearchCal du JMMC.

Pour finir, il est préférable pour le JMMC de pouvoir maitriser l'intégralité des composants de l'application. L'application doit donc pouvoir fonctionner sans l'aide d'un quelconque composant propriétaire ou payant.

9 http://cdsweb.u-strasbg.fr/CDS-description.gml

1.2.2)Objectif

L'objectif est donc la mise en place d'une base de données d'étoiles de calibration. La base de données devra être consultable et modifiable à distance via des interfaces homme-machine et machine-machine respectant les méthodes de l'Observatoire Virtuel. Les modifications de la base doivent être faites dans un cadre sécurisé et l'ensemble du système doit être interopérable. L'outil doit supporter facilement l'ajout de champs dans la base. La finalité du projet est de mettre en place une base de données dynamique et facile à intégrer à d'autre applications, notamment l'application SearchCal[10] du JMMC.

1.2.2.1)Contenu de la base

La finalité de la base est de contenir des ensembles de commentaires. Chaque ensemble de commentaires est relatif à une étoile.

Une étoile a pour caractéristiques ses coordonnées RA et DEC et éventuellement un de ses nom.

Un commentaire a, au moins, pour caractéristiques l'étoile à laquelle il fait référence, l'identifiant de son auteur, sa date de création, les informations relatives à son état (valide ou autre éventuel), l'instrument avec lequel la mesure a été faite et un petit descriptif de sa raison d'être. Il est également nécessaire de préciser avec quel interféromètre a été effectuée la mesure. Cependant, les instruments sont liés à leur interféromètre (chaque instrument a un, et un seul, interféromètre). Une économie est donc envisageable.

Les données contenues dans la base doivent être pertinentes, avoir une utilité.

1.2.2.2)Extensibilité

Le contenu du champ commentaire n'étant connu qu'au minimum, il est nécessaire de réduire au maximum la quantité de code à réécrire en cas d'ajout d'un champ aux commentaires . On est cependant plus tolérant dans le cas d'une donnée dont la sémantique impose un traitement particulier (vérification automatique, champ calculé,...) comme c'est le cas pour la date de saisie ou l'instrument par exemple. L'ensemble des champs relatifs à une étoile n'est pas sensé changer à l'avenir.

10 http://www.jmmc.fr/searchcal_page.htm

1.2.2.3)Alimentation de la base

L'alimentation de la base est faite par les membres de la communauté scientifique. Il est nécessaire, à titre préventif, de limiter les possibilités d'abus quand à l'alimentation de la base. Ainsi, un système de comptes utilisateurs viendra contrôler l'accès en écriture dans la base.

La validité des champs est autant que possible vérifiée de manière automatique. Un administrateur est nécessaire pour les données dont la vérification ne peut pas être automatisée.

Afin de limiter au mieux les erreurs de saisie des coordonnées d'une étoile, il est nécessaire d'automatiser autant que possible leur saisie.

Étant donné le petit nombre de saisies potentielles, seuls des astronomes alimenteront la base. Il n'est donc pas nécessaire de faciliter l'automatisation de la saisie d'un commentaire.

1.2.2.4)Consultation de la base

L'accès à la base est laissé libre. La base doit être accessible à travers une interface compatible avec le VO et une interface facile à aborder par des astronomes.

L'interface homme/machine étant destinée à la communauté scientifique dans son ensemble, elle doit être intégralement en anglais, langage international par excellence.

La communauté scientifique ayant pour habitude d'effectuer ses recherches d'objets dans une base à travers une recherche conique, ce mode de consultation doit être fourni.

Les deux modes d'accès doivent permettre de retrouver facilement les informations les plus pertinentes. C'est à dire:

-Quelles sont les étoiles ayant un commentaire?
-Quelles sont les étoiles ayant un commentaire valide?
-Quels sont les commentaires correspondant à une étoile définie présente dans la base?
-Quels sont les commentaires valides correspondant à une étoile définie présente dans la base?

1.3)Moyens à disposition

1.3.1)Infrastructure informatique

Du point de vue matériel, on dispose pour cela d'un serveur suffisamment puissant pour accueillir ce genre de service. Le développement sera effectué sur des machines de bureau standard.

Du point de vue logiciel, le serveur (ainsi que les postes de travail) fonctionne sous un système d'exploitation Linux. On dispose de l'intégralité des logiciels libres et gratuits fonctionnant sous Linux. Le serveur dispose déjà d'un système de gestion de base de donnée MySQL[11] et d'un serveur d'application TomCat[12]. Un serveur TomCat permet de faire fonctionner des servlets Java. Les servlets java sont des classes java obéissant à une interface spécifique dont la fonction est de répondre à une requête HTTP[13]. Le SQL[14] est un langage de manipulation de Base de Données très populaire.

Le développement a été effectué sous Netbeans[15] 6.5, environnement de développement très polyvalent, avec les modules java web et associés. Netbeans et ces modules complémentaires permettent le développement de servlets java[16] et de JSP[17]. Netbeans et ces modules ont également la capacité de contrôler le serveur TomCat nécessaire aux tests de l'application, facilitant ainsi le développement.

Il a également été fait usage de phpMyAdmin[18] pour le contrôle de la base MySQL. Il s'agit d'un ensemble de script PHP permettant d'effectuer, via une interface web, toutes les opérations réalisables en ligne de commande. PhpMyAdmin fonctionnait sur un serveur apache classique que nous n'avons pas eu à manipuler.

Ces outils avaient déjà été pris en main précédemment, à l'exception de TomCat. La prise en main de TomCat n'a pas posé de problème étant donné le caractère simple des manipulations à effectuer, l'assistance de Netbeans et l'auto-documentation apportée par TomCat.

1.3.2)Outil de l'observatoire virtuel

Le VO organise régulièrement des ateliers et séminaires au cours desquels sont présentés des méthodes et des outils. Ainsi nous disposons d'une très large gamme de solutions préfabriquées qu'il convient d'étudier avant de se lancer dans une quelconque réalisation. Il serait inutile d'implémenter des éléments déjà réalisés. Sont, par ailleurs, proposés des API[19] et du code source.

Les institutions membres du VO proposent également des services en ligne accessibles via les standards du VO ou, tout du moins, de manière interopérable.

11 http://www.mysql.fr/
12 http://tomcat.apache.org/
13 http://fr.wikipedia.org/wiki/Hypertext_Transfer_Protocol
14 http://fr.wikipedia.org/wiki/SQL
15 http://www.netbeans.org/
16 http://fr.wikipedia.org/wiki/Servlet
17 http://fr.wikipedia.org/wiki/JavaServer_Pages
18 http://www.phpmyadmin.net/home_page/index.php
19 http://fr.wikipedia.org/wiki/Interface_de_programmation

La liste de ces outils est longue. On peut cependant citer ceux du CDS (Centre de Données astronomiques de Strasbourg) qui fournit Simbad et Vizier. Vizier est une grande base de données de catalogue astronomique. Il regroupe les données de plusieurs catalogues différents de manière standardisée. Simbad, quand à lui, propose des services permettant la recherche et la consultation de tous ces catalogues de manière unifiée, comme s'il n'était qu'un. Références en la matière, Simbad et Vizier offrent les informations les plus fiables possible et sont donc des outils essentiels à beaucoup d'autres projets. Toutefois, certaines données peuvent changer sans préavis pour gagner en précision, notamment les coordonnées.

2)Solution

2.1)Organisation de la base

Comme vu précédemment, l'objet de la base de données est de contenir des ensembles de commentaires relatifs à des étoiles. On a donc deux éléments principaux: les étoiles et leurs commentaires.

Une étoile a plusieurs commentaires et un commentaires n'a qu'une étoile. La base de données a donc la structure de base suivante:

2.1.1)Les étoiles

Comme dit précédemment, une étoile n'a pas d'identifiant unique et fiable. Or, il nous faut un identifiant unique pour pouvoir les lier à leurs commentaires respectifs. Utiliser le couple de coordonnées RA et DEC comme identifiant a été envisagé mais il s'agit de nombres réels correspondant à des mesures susceptibles de gagner en précision et donc de changer. Un identifiant propre à la base a donc été arbitrairement créé. Il s'agit d'une variable initialisée à zéro et auto-incrémentée à chaque entrée d'une étoile dans la table.

Il nous faut, par ailleurs, les coordonnées des étoiles en degré ou en radian, afin de pouvoir exécuter une recherche conique sur la base. Les coordonnées en notation sexagésimal sont

également désirées par les utilisateurs. Des conversions sont possibles dans les deux sens, mais convertir les coordonnées sexagésimales en degré ou radian, à chaque fois que nécessaire, serait lourd de calculs étant donnée la fréquence d'utilisation de la recherche conique. Par ailleurs, les conversions sont de moins bonne qualité que les résultats fournis par les services du CDS. Or, la précision peut être un facteur clé. Il a donc était décidé de stocker les coordonnées aussi bien en degré/radian qu'en notation sexagésimale afin de ne pas perdre de temps à calculer ces données ou les obtenir depuis le CDS. Les coordonnées en notation sexagésimale n'étant destinées qu'a l'affichage, on peut se permettre de les stocker sous forme de chaines de caractères tel qu'elles seront affichées plus tard.

Dans un premier temps, le degré a été désigné arbitrairement plutôt que le radian car il semblait plus facile à appréhender et permettait de saisir plus facilement un jeu de test. Ce choix s'est avéré sans importance par la suite, en raison de l'adaptabilité d'un des outils utilisés.

Les-dites coordonnées pouvant gagner en précision sans préavis et n'étant, de ce fait, pas stables, on conserve également l'identifiant ayant servi à retrouver les coordonnées au CDS par pure précaution. Celui-la même qui a été rentré par l'utilisateur lors de l'ajout de premier commentaire référençant cette étoile (cf ajout d'élément).

Les étoiles ont donc pour caractéristiques: un identifiant unique arbitraire, la coordonné RA en degré, la coordonné DEC en degré, la coordonné RA en notation sexagésimale, la coordonné DEC en notation sexagésimale et l'identifiant ayant permis de la retrouver pour la première foi au CDS.

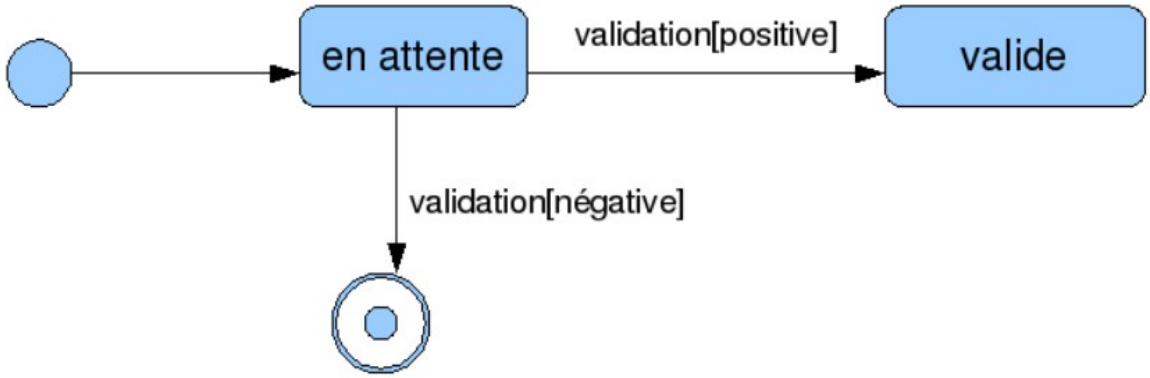
2.1.2)Les commentaires

Chaque commentaire étant lié a une étoile, il a, à minima, l'identifiant unique arbitraire de cette étoile. Il est également lié à un propriétaire. Il a donc l'identifiant de ce propriétaire. Nous avons donc voulu, dans un premier temps, prendre le couple identifiant étoile et identifiant utilisateur comme identifiant d'un commentaire, mais c'était sans compter sur le fait qu'un même utilisateur pouvait faire plusieurs commentaires au sujet d'une même étoile. Il a donc été décidé d'ajouter à ce couple la date de création du commentaire qui, par ailleurs, devait être stocké à d'autres fins.

Chaque commentaire comporte d'autres éléments utiles aux scientifiques mais n'ayant pas de sémantique particulière dans la base. Il peut s'agir de n'importe quel type de valeur. Leur nombre et leur nature sont susceptibles de changer. Ils sont la raison d'être de la contrainte d'extensibilité exprimée précédemment. Chaque commentaire comporte donc aussi des paramètres inconnus.

Il a été décidé que chaque commentaire avait un état qui décrive sa validité. En effet, le processus de validation d'un commentaire n'étant pas instantané de par le contrôle par un

modérateur (cf 2.2), il faut stoker le commentaire dans la base en attendant sa validation. Ceci suppose de pouvoir savoir si un commentaire a été validé ou s'il est en attente. Nous pensions donc utiliser la suite d'états suivante:

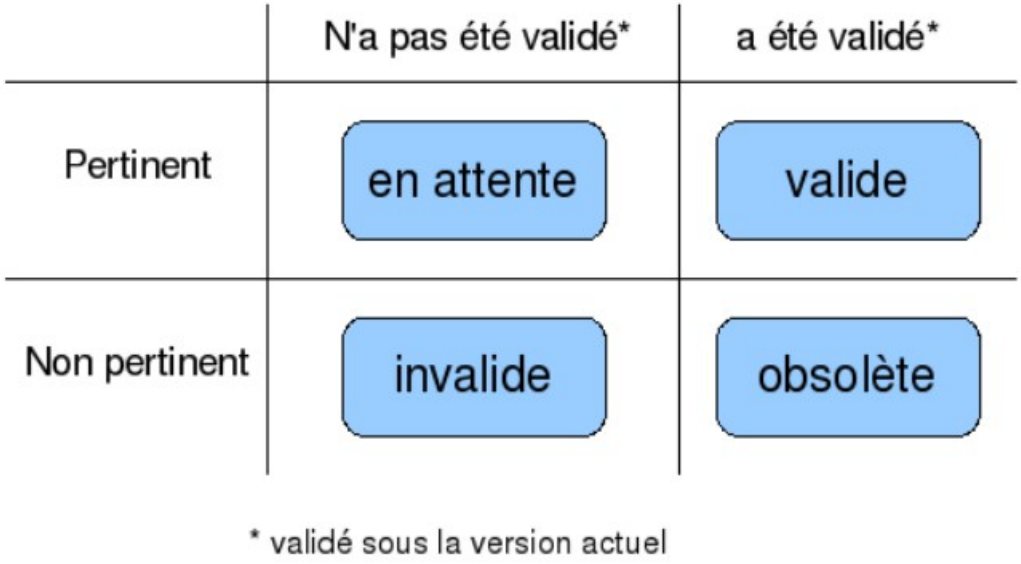


Mais il s'est avéré, au cours du projet, qu'un commentaire pouvait devenir obsolète, du fait d'une contestation ou d'une suppression volontaire. Or, il a été considéré comme perturbant et potentiellement litigieux pour les utilisateurs de supprimer purement et simplement les commentaires de la base. Il est donc nécessaire de garder une trace des commentaires jusqu'alors présent dans la base.

Cela a fait également apparaître qu'il pouvait être pertinent de permettre aux utilisateurs de voir plus largement les commentaires présents dans la base et pas seulement ceux validés par l'administrateur(cf 2.2). Un certain nombre de scénarios démontrant l'utilité d'une telle mesure ont été imaginés. Par exemple, un utilisateur humain étant capable d'évaluer lui même la pertinence d'une donnée, il peut vouloir voir les commentaires en attente de validation afin de compléter lui même la liste des calibrateurs qu'il aura à éviter.

Il a donc été décidé de permettre à l'utilisateur de voir, selon ses désirs, l'intégralité de la base ou seulement le contenu validé. Les commentaires peuvent donc être dans quatre états au lieu de deux : en attente de validation, validé, invalidé, et obsolète(c'est à dire, ayant été validé dans un premier temps mais ayant été retiré ensuite). L'état d'un commentaire peut donc être subdivisé en deux champs: le commentaire a t-il été validé sous sa forme actuelle? et le commentaire est il pertinent ?

D'où le diagramme suivant:



2.1.2.1)Les instruments

Parmi les informations récoltées dans un commentaire, les utilisateurs désirent savoir avec quel instrument ont été faites les mesures conduisant à déduire l'objet du commentaire. Ils désirent également savoir sur quel interféromètre a été faite la mesure. Or, chaque instrument est lié à un interféromètre bien précis. Les caractéristiques de chaque interféromètre et instrument sont réduites à leur nom. On stockera donc les instruments et leurs interféromètres respectifs dans une autre table. Le commentaire ne stockera, lui, que le nom de l'instrument qui le concerne. Le commentaire complet sera reconstitué à la consultation de la base.

2.1.2.2)Les utilisateurs

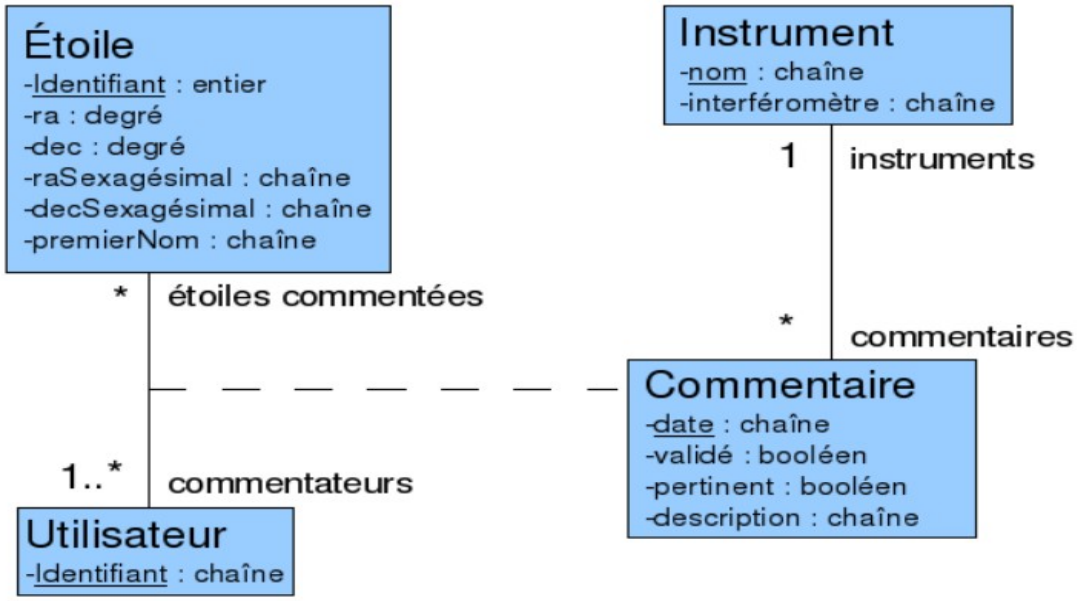
Comme vu précédemment, chaque commentaire a forcement un auteur et un même auteur peut avoir plusieurs commentaires. Il peut également n'en avoir aucun. Il est nécessaire de savoir qui a écrit quoi ne serait-ce que pour que chaque utilisateur sache ce qu'il a déjà écrit et pour pouvoir gérer les droits de modification des commentaires.

Un système de gestion des utilisateurs doit être mis en place, ce qui sous entend que pour chaque utilisateur, il va nous falloir stocker au moins son identifiant et son mot de passe. Mais, comme nous le verrons plus loin, ce système est externe à notre application et ne communique avec elle que via un module très simple (c'est la raison pour laquelle seule une sous partie de ce rapport est réservée aux utilisateurs). La seule chose que nous avons à stocker dans cette base est donc

l'identifiant de l'utilisateur. On a donc un lien « virtuel » avec les utilisateurs présents dans une autre base parmi les caractéristiques d'un commentaire.

Avant cette décision, s'était posé le problème de la déclaration à la CNIL[20] (Commission Nationale de l'Informatique et des Libertés). Car si cette solution n'avait pas été adoptée, le système aurait eu à stocker lui même des informations personnelles concernant les utilisateurs. Une petite étude a donc été faite et il s'est avéré qu'il n'était pas nécessaire d'être déclaré à la CNIL si les informations stockées ne l'étaient qu'à des fins de contact, ce qui était le cas dans le cadre de notre projet[21].

Les changements précédemment cités étendent la structure de la base de la manière suivante:



2.2)Comportement de la base

2.2.1)Droit d'accès au service.

La consultation de la base est laissée libre.

La modification de la base n'est laissée qu'au tiers de confiance. Un système de compte utilisateur est donc nécessaire. Il a été décide de laisser la fonction de gestion des comptes utilisateurs à une autre application du JMMC avec laquelle notre outil dialogue via une interface définie. Nous nous sommes limités, de notre coté, à vérifier que l'utilisateur existe et qu'il a le droit

20 http://www.cnil.fr/

21 http://www.cnil.fr/index.php?id=2016&delib[uid]=107&cHash=27269aa704

d'utiliser notre service. Le JMMC fournit par ailleurs les informations nécessaires au contact de l'utilisateur.

Nous avons fait ce choix afin de ne pas avoir à créer un module de gestion des utilisateurs complexe et propre à notre projet. Ce qui a, de plus, pour effet d'éviter à l'utilisateur d'avoir à retenir encore un mot de passe de plus, lui évitant ainsi une gêne. Le JMMC avait, par ailleurs, pour projet d'étendre son module de gestion des utilisateurs afin de lui permettre de gérer les accès à des services annexes, comme le notre, de manière unifiée. Ce choix a donc était fait pour correspondre à une stratégie plus large du JMMC.

Lorsqu'un tiers veut pouvoir alimenter la base, il demande à l'administrateur par email cette autorisation. L'administrateur doit alors juger du bien fondé de la demande d'accès et autorise alors, ou non, le droit d'accès au service. Chaque cas étant trop différant pour permettre l'automatisation de ce processus, il est nécessaire qu'un superviseur juge au cas par cas.

Cette mesure a pour but de filtrer l'accès en écriture à la base, afin de ne pas le laisser à des personnes n'en ayant pas la légitimité. Elle nous permet, par ailleurs, d'identifier formellement l'utilisateur-auteur d'un commentaire et, le cas échéant, de le contacter via les informations qu'il a laissées lors de son inscription au JMMC.

2.2.2)Ajout d'élément

Le contrôle des informations saisies est nécessaire afin de garantir la pertinence des informations contenues dans la base. Pour rappel, les informations saisies doivent permettre de comprendre pour quelle raison le calibrateur a été intégré à la base.

Il n'est pas possible de vérifier toutes les informations fournies par l'utilisateur lors de la saisie car certaines données proviennent d'observations qu'il n'est pas envisageable de reproduire pour la seule vérification des données de notre base. Seules les informations vérifiable sans observation sont donc contrôlées.

Concernant le problème des coordonnées mal connues des utilisateurs, sachant qu'ils connaissent en revanche au moins un nom de l'étoile qu'ils veulent designer, il a été décidé de ne demander à l'utilisateur que le nom de l'étoile et de consulter le service Simbad du CDS(Centre de Données astronomiques de Strasbourg) afin de récupérer les coordonnées de l'objet à partir du nom de l'étoile. Cela évite, par ailleurs, qu'un utilisateur ne rentre dans la base des données non vérifiées par inadvertance ou par malveillance.

Une fois les coordonnées obtenues, étant donné qu'elles peuvent être modifiées sans préavis, on ne peut pas, de façon simple, comparer leur égalité avec les autres coordonnées de la base pour savoir si le commentaire concerne une étoile déjà présente. Mais le degré de précision des

coordonnées fournies par le CDS est déjà très important et, par conséquent, si la précision venait à être augmentée, les nouvelles coordonnées resteraient extrêmement proches des anciennes.

La solution consiste donc à comparer les coordonnées retournées par le CDS avec celles de la base en effectuant une recherche conique dont l'angle d'ouverture serait légèrement supérieur au degré de précision minimum des coordonnées fourni par le CDS. Si la recherche donne un résultat, alors le résultat est considéré comme égal à l'élément recherché. Ces coordonnées sont mises à jour si nécessaire et c'est cet élément qui sera associé au nouveau commentaire. Si la recherche ne retourne rien, alors l'étoile est considérée comme nouvelle et créée dans la base.

2.2.3)Modification d'élément

Toute modification invalide l'élément qui doit être revalidé ensuite par l'administrateur. L'administrateur vérifie le contenu des champs contrôlables et notifie par mail l'utilisateur en cas de non validation. Ceci, toujours, afin de garantir la pertinence de l'information donnée par l'utilisateur.

Comme pour l'ajout d'élément, il n'est pas possible de vérifier toutes les informations fournies par l'utilisateur lors de la saisie. Seules les informations vérifiables sont donc validées, aussi bien de manière automatique que manuelle.

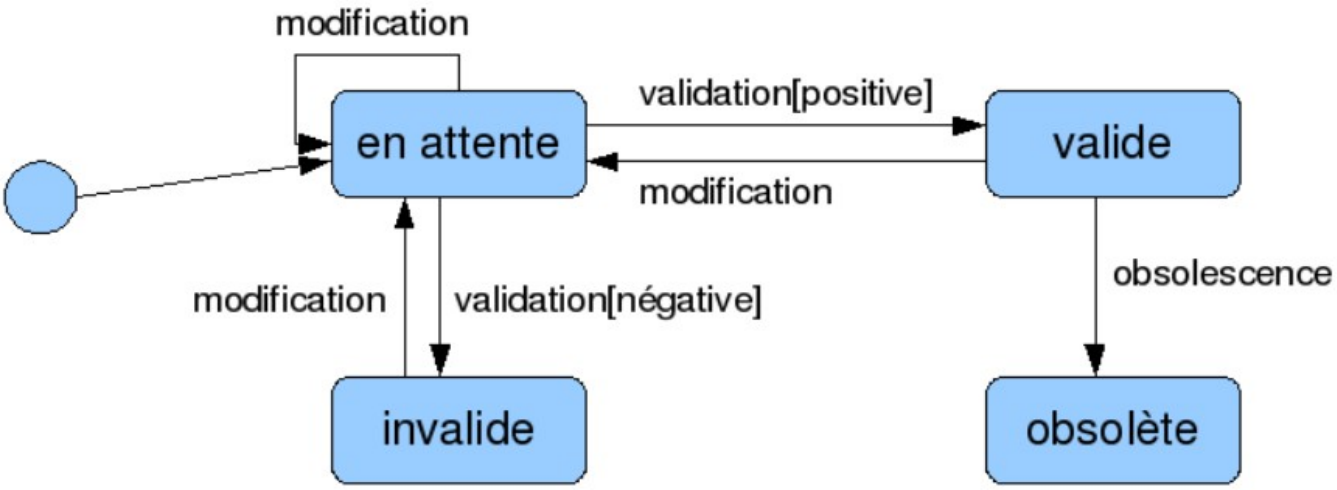
Les commentaires obsolètes ayant pour vocation de rester tels qu'ils sont, indéfiniment, il n'est pas possible de les modifier. Chaque propriétaire d'un commentaire valide peut, par ailleurs, décider de le rendre obsolète s'il le désire. Ceci car il est tout a fait possible qu'un utilisateur découvre, par lui même, l'obsolescence de son commentaire. Laisser cette option à l'utilisateur permet de ne pas avoir à saisir l'administrateur si ce cas se produit.

Un commentaire invalidé peut également être modifié, auquel cas il sera réexaminé par l'administrateur. A noter que la possibilité de modifier un commentaire validé est actuellement remise en question en vertu de la volonté de conserver les éléments ayant été validés mais devenu non pertinents sous l'état « obsolète ».

2.2.4)Contestation d'élément

Étant donné le petit nombre d'utilisateurs, la probabilité d'occurrence d'une contestation a été jugée suffisamment faible pour ne pas prévoir d'outil automatisant la démarche. La personne contestant une donnée doit donc utiliser la fonction de contact de l'administrateur et formuler manuellement sa demande.

Au terme de cette réflexion, nous en sommes donc arrivés à modéliser l'état d'un commentaire et son évolution comme résumé dans le diagramme qui suit:



2.3)Interaction avec la base

L'interface homme/machine reprenant, entre autres, des fonctions fournies par l'interface machine/machine de manière plus présentable, il a été décidé de réaliser d'abord cette dernière puis de se servir des services qu'elle offre pour réaliser l'interface homme/machine afin d'éviter la redondance du code.

2.3.1)Interface machine/machine

A la base, cette interface devait pouvoir fournir tous les services fournis par l'interface manuelle, à des fins d'automatisation facile, et ce, en respectant les standards de l'Observatoire Virtuel[22]. Il s'est avéré par la suite qu'il n'était pas nécessaire d'automatiser la modification de la base car le nombre d'éléments à saisir a été jugé trop faible pour justifier du temps passé à le faire. Ceci d'autant qu'aucun standard concernant les opérations de modification que nous avons à implémenter n'existe à l'heure actuelle.

De plus, si une automatisation devait tout de même être réalisée, l'interface homme/machine ayant la forme d'un site internet, il serait toujours possible de le faire par ce moyen. Le web repose, en effet, uniquement sur des interfaces machine/machine standard. Seul le navigateur rend l'information lisible par un humain en bout de course. Il est donc tout à fait possible d'automatiser le processus de modification de la base en analysant de prés l'interface homme/machine.

22 http://www.ivoa.net/Documents/

La seule fonction dont l'utilisation doit être facilement automatisable est donc la consultation de la base. Il aurait été possible, mais pas envisageable, de faire comme pour les fonctions de modification car l'Observatoire Virtuel a défini des méthodes standards pour cette fonction.

2.3.1.1)Le standard

Il existe 3 standards définis par l'Observatoire Virtuel concernant l'accès aux données d'une base. Le premier est pour les spectres, le deuxième est pour les images et le troisième est pour tout le reste ayant au moins un couple de coordonnées RA et DEC. Ce dernier standard est appelé « Simple Cone Search »[23] et fait partie des DAL (pour Data Access Layer, ou couche d'accès au données). On y sélectionne les données voulues via une recherche conique sur les coordonnées. Il est adapté à toute les données concernant un point du ciel en général. C'est ce standard qui nous intéresse car notre base référence des étoiles et qu'on y accède au travers d'une recherche conique. En l'implémentant, on implémente aussi, du même coup, la fonction de recherche conique.

Dans les faits, ce standard définit les paramètres et le format de réponse d'une requête HTTP interrogée avec les méthode GET et POST. Il s'agit d'un standard obéissant à l'architecture REST[24]. Sont passés en paramètres, tous les éléments nécessaires à une recherche conique c'est à dire RA, DEC (les coordonnées du point visé) et SR (l'ouverture par rapport au point visé). Est alors renvoyé un texte au format VOTable correspondant au résultat de la requête. Le format VOTable[25] est un autre standard de l'Observatoire Virtuel. Il s'agit d'un document XML[26] permettant de transporter un tableau de données et ses méta-données (nom des colonnes, type …). Une VOTable peut aussi transporter un message d'erreur.

2.3.1.1)Les outils

A cette étape du projet ont été évalués les outils fournis par la communauté aux développeurs et autres personnes ayant à mettre en place un service obéissant au standard du VO. Ceci afin de voir dans quelle mesure nous pouvions les utiliser pour notre projet. Notamment pour ce qui est du calcul des recherches coniques. Cette démarche a, d'ailleurs, été fructueuse et a permis de faire totalement abstraction des détails de la recherche conique.

Ont été étudiés:

23 http://www.ivoa.net/Documents/latest/ConeSearch.html
24 http://fr.wikipedia.org/wiki/Representational_State_Transfer
25 http://www.ivoa.net/Documents/latest/VOT.html
26 http://fr.wikipedia.org/wiki/Extensible_Markup_Language

-DALToolKit : Il semblait fournir les 3 services standards cités plus haut en s'adaptant sur une base SQL mais ne supportait pas encore celui qui nous intéresse sous sa version actuelle. L'outil a donc été abandonné.

-Saada : Il fournit le service standard qui nous intéresse en plus de fabriquer des pages de consultation manuelle mais obtient ce résultat de manière peu documentée et n'est pas adaptable facilement à une base dynamique. Cela pose le problème d'y adapter notre interface d'alimentation. Ce produit est plus adapté à la mise en ligne de petits catalogues fixes. Ainsi, malgré une prestation de service présentée comme alléchante, l'idée d'utiliser cet outil a été abandonnée.

-DSA Astrogrid[27] : Il fournit le service standard demandé. Il est également bien maintenu et assurera d'autres services standards du VO dans l'avenir. Il peut s'adapter à tout type de base SQL. Il s'est avéré être très bien auto-documenté (étant un service web il héberge sa documentation sur place). Il ne fourni pas de fonction inutile à notre projet ce qui est gage de simplicité et donc d'efficacité. Cette outil a été sélectionné pour toutes ces raisons. (signification du nom non trouvé)

Ces trois solutions fonctionnent toutes avec des servlets Java (cf 1.3.1). Ce choix s'explique par leur simplicité de mise en place et leur capacité à utiliser toutes les fonctions fournies par la librairie standard Java et donc à être polyvalentes.

DSA fonctionne donc sur un serveur d'application TomCat. Il est nécessaire de configurer trois point de DSA pour l'adapter à son environnement.
Premier point, remplir le ficher de configuration. Il est auto-documenté et fournit quelques paramètres simples à DSA comme, par exemple, son nom d'utilisateur et son mot de passe d'accès à la base.
Deuxième point, ajouter le pilote de la base de données dans ses librairies. La manipulation en elle même est très simple mais elle nous a montré, en partie, comment utiliser une base SQL facilement depuis une servlet ou, plus généralement, tout autre application Java.
Enfin, dernier point, générer le fichier de méta-données. La sémantique associée à chaque élément de la base y est donnée ici. Il s'agit d'un document XML et peut être partiellement auto-généré par DSA. Il supporte donc parfaitement l'extensibilité de la base.

Enfin, DSA ne fait pas la distinction entre une vue d'une table et la table en elle même. Ce qui nous permet de faire tous les tris et les combinaisons permis par le SQL pour restreindre les résultats à certaines données. Il ne gère cependant pas les clés composées.

27 http://www.astrogrid.org/maven/docs/HEAD/pal/index.html

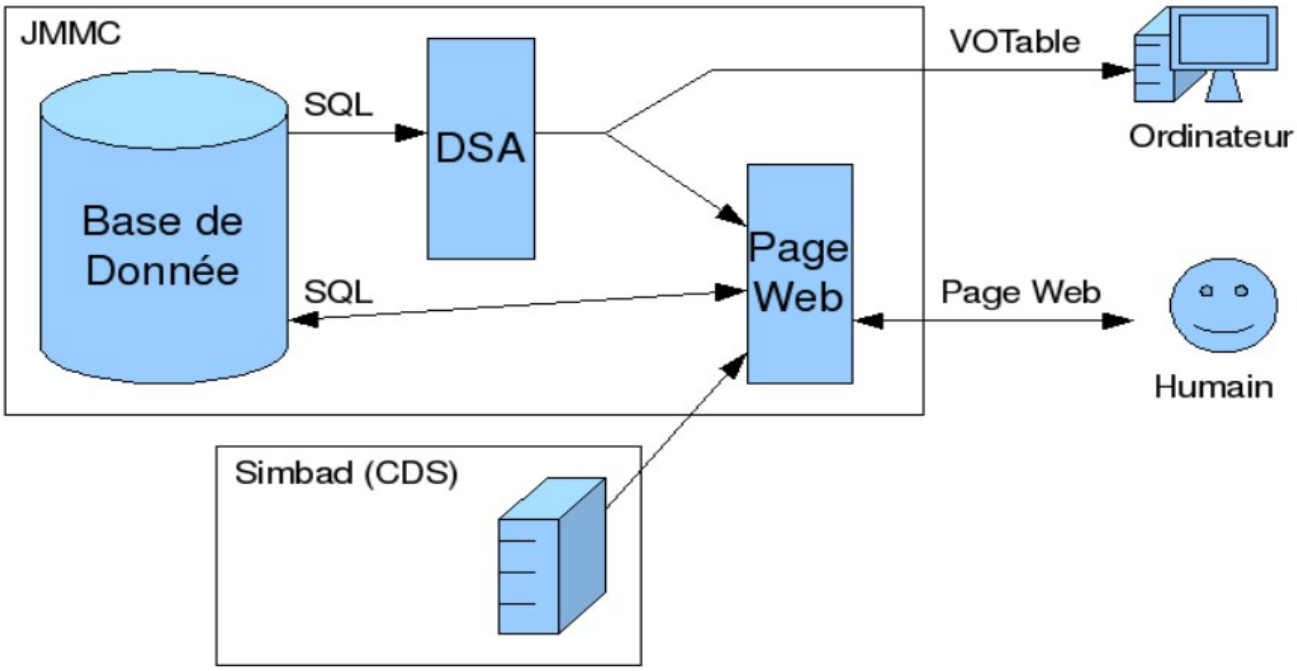
2.3.2)Interface homme/machine

Cette interface se présente sous la forme de page web car elle a les fonction suivante: elle sert à faire des opérations sur une base de manière distante; elle doit être simple d'utilisation; elle doit être lisible par des humain et elle doit le faire de manière interopérable. La solution du site internet s'impose donc de par sa suprématie vis a vis des autres solutions possibles qu'aurait été, par exemple, la mise en place de notre propre protocole accompagné d'un client spécifique à notre application.

L'application ayant a terme pour objectif d'être intégré au site du JMMC, l'architecture et le style de la page ont été définis par ce dernier. A savoir, une entête commune contenant le logo et la navigation générale et un menu de navigation locale sur la gauche.

On en arrive donc à devoir faire des pages web devant, à minima, interagir avec une base de donnée, gérer des sessions utilisateurs(demander les identifiants à chaque fois n'est pas envisageable), aller chercher des données via des requêtes HTTP (DSA, Simbad), interpréter des documents XML (VOTable) et éventuellement aller chercher des informations dans un fichier de configuration (pour l'extensibilité). Pour générer des pages web dynamiques, plusieurs concurrents sérieux se font face.

On veut donc construire une application qui ressemble à cela:



3)Réalisation

3.1)Détail de l'Implémentation et Problème Rencontré

3.1.1)choix du langage

Comme dit précédemment, nous avons besoin d'un outil capable de générer dynamiquement des pages web disposant de multiples fonctionnalités allant du simple accès à une base SQL, à l'interprétation de documents XML en passant par l'envoi de requête HTTP.

S'opposent actuellement sur le marché de l'application web:

-les ASP[28] (pour Active Server Pages) : Solution propriétaire de chez Microsoft. Cette technologie été éliminée d'emblée faute de portabilité. Elle a les même fonction qu'un script CGI.

-les servlets Java : Il s'agit d'un ensemble de classes et d'interfaces Java permettant de gérer une requête HTTP (ou autre,tell que FTP, mais ces cas ne seront pas étudiés ici). Elles permettent donc d'écrire des réponses dynamiques au format HTML[29]. Une servlets java n'est pas fonctionnelle en elle même. Elle doit être intégrée à un serveur d'application. Il existe de nombreux serveurs d'applications mais le serveur le plus utilisé est TomCat, logiciel open source et libre d'utilisation.

Les servlets Java existent, par ailleurs, aussi sous la forme de JSP (pour JavaServer Pages). Une JSP est un document ayant une architecture proche du XML permettant de générer automatiquement une partie du code d'une servlets java. Un peu moins fonctionnelles qu'une simple servlets, elles ont pour vocation de faciliter l'écriture de page web dynamique sans empêcher l'utilisation de servlets classiques en cas de besoin. Cette variante a donc été sélectionnée.

-le PHP[30] (pour Pretty Hypertext Processor) : c'est un langage permettant d'écrire des scripts qui seront ensuite interprétés par un logiciel (on appelle PHP la réunion des deux). C'est un langage qui fonctionne, entre autres, sur des plateformes libres. Il est l'un des plus employés pour générer dynamiquement des pages web. C'est une technologie très portable. L'étendue de sa bibliothèque standard nous étant inconnue, elle a été supposée moins importante que celle du Java et plus

28 http://fr.wikipedia.org/wiki/Active_server_pages
29 http://fr.wikipedia.org/wiki/Hypertext_Markup_Language
30 http://fr.wikipedia.org/wiki/PHP

specialisée. L'utilisation de ce langage n'était par ailleurs pas dans les habitudes de l'équipe de développement.

-les script CGI[31] (pour Common Gateway Interface) : c'est un ensemble de règles permettant de faire communiquer un serveur Web avec des applications externes. Les CGI peuvent être écrits dans n'importe quel langage (C, C++, Perl, Python ...). Le seul avantage fourni par cette technologie par rapport aux autres solutions étant le choix du langage, elle n'a d'intérêt que si l'on veut programmer dans un langage autre que le Java et le PHP car ces deux langages ont déjà des solutions d'intégration au web qui leur sont spécifiques. Java ayant la librairie standard la plus étendue, la mieux documentée et l'équipe ayant l'habitude de ce langage, la solution du script CGI a été estimée moins intéressante que celle de la servlet Java.

C'est donc les servlets Java qui ont été sélectionnés pour générer les pages de l'interface homme/machine de notre application. Cela permet par ailleurs d'alléger le travail du serveur. En effet, DSA et notre projet fonctionnant avec le même serveur d'application (TomCat), il ne sera pas nécessaire d'installer du contenu supplémentaire pour faire fonctionner l'interface homme/machine.

On peut leur passer des paramètres globaux via un fichier de configuration. C'est par ce moyen qu'on adapte une servlet à son environnement spécifique. Dans notre cas, leur ont été passés des paramètre d'usage comme le chemin d'accès à la base de données, le nom et le mot de passe associé, localisation de DSA, localisation du CDS, … et aussi, dans un premier temps, certaines informations sur la signification de la base non contenue dans le fichier de méta-données originel.

L'accès à la base de donnée se faisant via une API standard commune à toutes les application de ce langage, cette question ne sera pas traitée dans ce rapport.

3.1.2)Interprétation des méta-données

Sachant que la base est extensible, et l'application devant y accèder au travers de requête SQL, elle va devoir composer, elle même, dynamiquement des requêtes SQL. Elle va aussi devoir générer dynamiquement les entêtes des tableaux présentant les résultats ainsi que des formulaires de saisie. Pour toutes ces taches, l'application a besoin d'informations sur les informations qu'elle a à traiter, c'est à dire des méta-données(des données sur les données). Ceci afin de pouvoir afficher les méta-données intéressantes aux yeux des utilisateurs et pour pouvoir associer une sémantique à ce qui n'est, à la base, que des noms de colonne pour pouvoir leur réserver un traitement particulier.

31 http://fr.wikipedia.org/wiki/Common_Gateway_Interface

Il nous fallait donc définir une structure représentant l'architecture de la base. Arrivés à ce point, nous nous sommes aperçus que quelqu'un d'autre avait déjà fait une partie de ce travail à notre place. En effet, comme vu dans la partie où nous avons traité DSA, cette application fonctionne déjà avec fichier de méta-données qu'elle peut, « cerise sur la gâteau », en partie auto-générer.

Ces méta-données étant stockées sous format XML, nous pouvions y accéder facilement, mais avant cela, il nous a fallu trouver un interpréteur ou « parseur » depuis l'anglais.

3.1.2.1)Choix de l'analyseur syntaxique XML

Il existe deux types de parseurs XML:
-les parseurs SAX[32] (pour Simple API for XML) : SAX traite les documents élément par élément et appelle la fonction qui lui correspond au fur et à mesure des rencontres. SAX ne stocke pas la structure du document mais le programmeur qui l'utilise est libre de le faire. Cette méthode impose de parcourir le document dans l'ordre.
-les parseurs DOM[33] (pour Document Object Model) : DOM charge l'intégralité d'un document XML dans une structure de données, qui peut alors être manipulée puis reconvertie en XML, mais, se faisant, cette méthode peut être gourmande en mémoire ce qui peut poser problème quand le document est de taille importante. Cependant, avoir tout le document en mémoire apporte l'avantage de pouvoir le parcourir sans contrainte.

Le document à analyser étant relativement petit et pouvant ne pas être analysé à chaque chargement de page, nous avons préféré utiliser la méthode DOM car elle est moins contraignante que la méthode SAX. Par ailleurs, reconstituer une structure à partir de SAX semblait complexe.

3.1.2.2)Organisation des méta-données

Une fois le document mis en mémoire, on le parcourt pour organiser notre propre structure de données, plus simple et implémentant des fonctions adaptées à nos besoins.

Notre structure se construit d'elle même par récurrence à partir du chemin du fichier qu'elle doit analyser. Le premier niveau de la structure interprète le document original pour le transformer en DOM, après quoi il collecte les informations le concernant, extrait chaque sous partie du document correspondant au niveau inférieur et passe cette sous partie en paramètre au constructeur de l'élément de niveau inférieur. Chaque sous partie en fait de même jusqu'au niveau le plus bas. A

32 http://fr.wikipedia.org/wiki/Simple_API_for_XML
33 http://fr.wikipedia.org/wiki/Document_Object_Model

chaque appel d'une servlets, le champs est testé, s'il est a NULL alors le constructeur est appelé. Les méta-données générées étant les mêmes, nous n'avons pas de problème d'accès concurrent à résoudre.

Il y a 4 niveaux de hiérarchie dans les méta-données:
-méta-données : celui-ci ne correspond à rien si ce n'est à un groupe de catalogue. Il s'agit du document en lui même.
-le catalogue : correspond à la base de données.
-la table : correspond à une table dans une base de données.
-la colonne : correspond à une colonne d'une table.
-l'UCD[34] (Unified Content Descriptors) : correspond à un des UCD d'une colonne. Un UCD est un vecteur de sens, il a vocation de définir ce qu'est la donnée. Il s'agit d'un standard du VO.

Les informations collectées sont dans tous les cas sauf l'UCD le nom de publication de l'élément (appelé nom par la suite), et le nom réel dans la base de l'élément (appelé ID par la suite). Dans le cas de la colonne sont ajoutés le type Java et l'unité. Un UCD a une version et un libellé. Les nom et les ID sont uniques dans l'ensemble dont fait partie l'élément.

En plus des fonction de parcours et de consultation habituels, on pouvait éventuellement avoir besoin des fonctions suivantes car elle portent sur des éléments ayant une sémantique:
-savoir quel catalogue a tel ID.
-savoir quel catalogue a tel nom.
-pour un catalogue donné, savoir quelle table a tel ID.
-pour un catalogue donné, savoir quelle table a tel nom.
-pour une table donnée, savoir quelle colonne a tel ID.
-pour une table donnée, savoir quelle colonne a tel nom.
-pour une colonne donnée, Obtenir l'UCD de telle version.
Elles ont donc été implémentée comme présenté en annexe.

Il s'est avéré, au cours du développement, que le fichier de méta-données ne contenait pas à lui seul assez d'informations sur la base pour la faire fonctionner. Nous avons d'abord comblé cette lacune en passant les informations supplémentaires en paramètre mais avons très vite été débordés par leur nombre. Il a donc été décidé de surcharger les méta-données de base.

Pour cela, nous avons utilisé un champ actuellement inutilisé par DSA présent a tout les niveaux: le champs description. Il a été décidé d'y introduire une chaine de caractère facile à identifier. Cette solution est provisoire mais nous permet d'avancer sur des points plus importants

34 http://www.ivoa.net/Documents/latest/UCD.html

du projet en attendant qu'ils aient été implémenté (cf 3.3). Pour pouvoir bien distinguer la description de l'identifiant ainsi introduit, il est encadré par les caractères '[' et ']'. Il nous suffit alors de rechercher la chaine « [identifiant] » dans la description pour savoir si l'élément pointé en est porteur. Ce mode permet, par ailleurs, de bien distinguer le contenu destiné aux humain du contenu destiné à notre application.

Ces méta-données servent à générer automatiquement les requêtes et le code HTML indépendamment du détail de la base de donnée.

3.1.3)Consultation de service web

Nous avons besoin, pour notre application, de venir consulter le service Simbad du CDS et le Simple Cone Search fourni par DSA. Ces deux service sont des services web. On y accède via une requête HTTP et le résultat nous est retourné sous forme de chaine de caractères.

Un outil du VO proposait ces fonctions mais il a été jugé trop lourd a mettre en place pour des taches aussi simple. Il était, par ailleurs, plutôt orienté vers le développement de clients destiné à fonctionner sur des machines de bureau. Il n'était donc pas conçu pour l'usage que nous voulions lui donner.

Ainsi il a été décidé d'implémenter nous même l'ensemble du processus. La bibliothèque standard Java proposait déjà un ensemble de classes destinées a exécuter des requête HTTP. Elles prennent le maximum de paramètre par défaut disponible. Leur utilisation fut donc très simple et sans surprise: création de l'objet, configuration supplémentaire éventuelle, lancement de la connexion. Après cela, il est possible de récupérer les flux d'entrée et de sortie éventuellement nécessaire. Comme nous utilisions une requête de type GET et que les paramètres y sont contenu dans l'URL, seul le flux entrant nous a été utile.

Une fois la chaine de caractères de réponse obtenu, il convient de l'interpréter.
Dans le cas du CDS, il est possible de personnaliser complètement le format de sortie(nature et unité des information,séparateurs, ...). Il est on ne peut plus facile d'interpréter un résultat dans ces conditions. Dans le cas du CDS, nous avons encapsulé toute cette procédure dans une classe (consultation et interprétation).

Dans le cas de DSA, deux chose peuvent nous intéresser : afficher directement les données ou récupérer les valeurs correspondantes à la colonne des identifiants afin de les réexploiter dans une requête SQL (pour rappel, DSA ne fonctionne pas sans une et une seule colonne d'identifiants). Dans les deux cas, on utilise un parseur DOM pour analyser le résultat car le temps de calcul et la place prise en mémoire ont été jugé suffisamment faible pour être acceptable. Pour le deuxième cas, le format VOTable transportant des méta-données en plus des données elle même, cherche dans ces

méta-données la colonne portant l'UCD correspondant à l'identifiant principale puis on parcourt les données pour récupérer la colonne correspondante. Le premier cas est traité dans la partie suivante.

3.1.4)Fabrication automatique de code HTML

Pour facilité l'écriture du site et garantir son extensibilité, il est nécessaire de générer automatiquement la partie du code correspondant à la base. Pour y parvenir, nous avons créé des procédures génériques utilisables quelque soit les données et les méta-données (du moment que ces dernières sont valides).

Pour générer les formulaires de saisie, il n'est besoin que des méta-données. Le seul formulaire susceptible de changé est celui de saisie d'un commentaire. Ceci ne concerne donc qu'une seule table de la base et non pas une combinaisons au autre chose venant complexifié d'avantage le problème.

Pour générer le formulaire, on parcourt la partie des méta-données correspondant à la table des commentaires et on fabrique un champs en fonction de la sémantique qui lui est associé. Ils peuvent être ignoré si remplie automatiquement(date de saisie,utilisateur), modifiés si il s'agit de pointeur vers autre chose (étoiles, instruments) et personnalisés en fonction du type de la donnée(un grand champs pour un texte, 3 champs pour une date...). Le champs prend pour identifiant celui de la colonne à partir duquel il a été généré.

Le contrôle des informations renvoyées par le formulaire se fait de la même manière. A partir des méta-données, on va cherché la valeur du ou des arguments qui sont associés à une colonne et on effectue un contrôle si sa sémantique le permet ou l'exige. La conformité avec le type de la donnée est vérifié d'une manière générale(un entier doit avoir le format d'un entier, un réel celui d'un réel,...) et plus si la sémantique de la donnée ne s'y limite pas (un instrument doit faire parti de la liste des instrument de la base,...).

Pour générer un tableau représentant le résultat d'une consultation, on peut avoir pour source soit une VOTable, soit un objet ResultSet issue d'une requête SQL. Dans les deux cas, sont transportées les donnée et les méta-données. Nous avons, au minimum, le nom des tables et des colonnes d'origine des données. Dans les deux cas, les seuls méta-données fourni par l'objet sont insuffisantes et nécessiteront de faire appel au notre. Il n'y a donc aucune différence entre les VOTable et les ResultSet vis a vis du traitement que nous devons leur appliquer. Seul le mode de parcours change. Les deux cas ne seront donc pas distingués.

Nous commençons par générer l'entête du tableau en parcourant le document résultat pour y récupérer les tables et colonnes d'origine des champs à présenter. A partir de cela nous venons chercher dans nos propres méta-donnée (celle issue du XML de DSA) les informations relatives au

champs et nous les affichons selon notre convenance. Construire le corps relève du jeu d'enfant : nous parcourons le document en séparant les champs par les balises adéquates. Nous évitons autant que possible de retirer des éléments en fonction de leur sémantique à cette étape du processus afin de garder l'outil le plus puissant et universel possible. Nous essayons autant que possible de traiter l'information devant subir un traitement particulier en amont de cette étape.

3.2)État actuel du projet

A l'heure actuelle, l'interface machine/machine fonctionne parfaitement et dispose d'un jeu d'essai. L'ensemble des outils nécessaires à la construction des pages ont été implémentés à l'exception de la fonction d'envoi d'email à l'administrateur.

La suite du travail consiste donc à assembler tous ces éléments pour construire le site. Les outils utilisés ayant été conçus de manière à être les plus génériques possible, ils sont, par conséquent, puissants et cette tache est donc relativement simple.

Si l'opportunité se présente, certaines fonctions décrites dans la partie suivante seront implémentées.

3.3)Amélioration possible

Une fois le projet prêt pour la phase de production, une interface d'administration plus poussée pourra éventuellement être implémentée. En effet, elle est actuellement réduite au strict minimum et ne gère que les cas courants, les cas plus complexes devant être gérés manuellement. Il a cependant été décidé que manipuler directement la base via phpMyAdmin était une solution de substitution suffisante. Améliorer l'interface administrateur n'est donc pas une priorité.

Le projet ayant été conçu depuis le début dans le but de pouvoir supporter l'extension de la base, aucune modification mineure de cette dernière ne devrait demander de réécrire quoi que ce soit dans le code de l'application. Seul l'ajout d'un élément, dont la sémantique nécessite qu'il subisse un traitement particulier, demanderait la modification du code. L'application utilise un maximum de méthodes génériques pour permettre de la réécrire le plus facilement possible et donc parer, au mieux, à ce genre d'éventualité. Ainsi, le projet est conçu afin de supporter au mieux toute amélioration de la base.

Les astronomes aimeraient, par ailleurs, pouvoir se voir retourner des résultats au format VOTable à travers l'interface manuelle. Ceci ne devrait pas être trop complexe. En effet, il ne s'agit que de modifier l'aspect avec lequel sont retournés les résultats des requêtes. Le format VOTable étant très proche du format des tableaux en HTML, il nous suffira de reprendre la fonction et de modifier légèrement la méthode de construction de certaines balises. Les structures des algorithmes devraient rester exactement les mêmes.

Conclusion

Ce stage a donc été pour moi une première initiation au monde du travail. Mon temps et mon énergie ont servi à évaluer les besoins des astrophysiciens puis à concevoir et développer des solutions. Pour cela, il m'a fallu comprendre certains détails du domaine, analyser les standards du VO, analyser toutes ces données pour concevoir une solution, apprendre à me servir de TomCat, DSA et servlets Java puis de réaliser l'objet de ma présence.

La principale difficulté fut la contrainte d'extensibilité. En effet, contrairement à un site classique, rien qui ne concerne directement les données ne doit être écrit en dur dans le code car elles ne sont pas stables. Or il est pourtant nécessaire d'y faire référence. On est donc obligé de parcourir les méta-données afin de recomposer ce qui aurait été écrit en dur dans un autre cas (code HTML, requêtes SQL,...). Ceci s'énonce simplement mais suppose de mettre en place des mécanismes complexes pouvant s'adapter parfaitement à toutes les situations. Il en résulte cependant des outils extrêmement puissants de par leur adaptabilité.

En plus d'étendre mes connaissances sur les technologies que j'ai utilisées, j ai donc appris au cours de ce stage à construire du code entièrement configurable et ai éprouvé beaucoup de satisfaction à user de mes capacités pour concevoir des solutions élégantes dans ce cadre. J'ai également appris des méthodes d'interrogation permettant de récupérer des informations très précises de personnes n'ayant qu'une idée floue des problèmes afin de pouvoir comprendre leur besoin. De même, j'ai amélioré ma capacité à vulgariser l'informatique pour la rendre accessible à des personnes non expertes afin de leur faire comprendre aussi mes contraintes.

A noter que, du fait de l'organisation très ouverte, la méthode de travail a pris un aspect très proche de l'« Extreme Programing ». Cette pratique est longue à décrire et a fait l'objet de plusieurs livres. Je me contenterais donc de rappeler les principes qui font de cette méthode ce qu'elle est: la communication, la simplicité, le retour d'informations et le courage face au changement (ainsi que le respect mais cela est plus universel). Principe desquels sont issus un certain nombre de pratiques « extrêmes ». Sans le vouloir particulièrement, tous ces principes, à l'exception de la revue permanente du code, ont été appliqués. Ainsi, sans le savoir, nous avons appliqué cette méthode car elle s'adaptait le mieux à notre environnement et s'est invitée d'elle même. Ce mode de développement est également proche de celui que l'on a connu à l'IUT dans le cadre de nos projets et de certains de nos travaux pratiques.

Le caractère itératif de cette démarche a posé quelques problèmes pour la rédaction du présent rapport. La pratique de la conception continue engendre le fait que certains besoins sont révélés suite à la mise en place de certaines fonctionnalités. Ceci n'est pas en accord avec une approche logique et donc bien compartimentée du problème. Par exemple, la mise en place de DSA a été faite dès la deuxième semaine de stage alors qu'aucun cahier des charges complet et bien structuré n'existait. Ce n'est, d'ailleurs, toujours pas le cas et le présent rapport a pour vocation de documenter cette partie.

Une sorte de « journal du développeur » aurait donc été plus adapté mais incompatible avec les contraintes liées au rapport. J'ai donc décidé de présenter les versions majeures de chaque étapes du projet bien qu'en réalité, il en ait existé une multitude, plus ou moins proches les une des autres et plus ou moins floues, qui étaient chacune pertinentes au moment de leur existence et dont il aurait donc fallu parler.

Le flou n'a pas posé de problèmes dans la mesure ou il est réduit dès que nécessaire. Ce stage s'est donc globalement bien déroulé.

Webographie

documentation sur les standard du VO
http://www.ivoa.net/Documents
du VO

documentation sur les standard du CDS
http://aladin.u-strasbg.fr/tutorials/cds.gml
du CDS

Informations diverse sur les fait scientifique ainsi que les caractéristique des produits utilisés.
http://fr.wikipedia.org/wiki/Accueil
combinaison sources multiples

Informations sur les API standards Java
http://java.sun.com/j2se/javadoc/
de Sun Microsystems

Informations et tutoriels divers
http://www.developpez.com/
combinaison sources multiples

auxquels il faut ajouter de multiples autres sources moins signifiantes.
Aucune ressource papier n'a été utilisée.

Annexes

A noté qu'il n'y a pas de planning du fait de la nature de l'organisation du projet: Toute organisation dans le temps n'est qu'indicative. Par ailleurs plusieurs taches étant effectuées simultanément, un planning détaillé ne saurait être fourni et il n'est pas possible non plus de synthétiser de manière objective.

Diagramme de classe de la partie de l'application gérant les méta-données:


MetaData
Attributes
private boolean isNew = true
Operations
public MetaData(String filepath)
public Catalog getCatalog(int i)
public Catalog getCatalogId(String id)
public Catalog getCatalogName(String name)
public int getCatalogSize( )
public boolean isNew( )
catalogs
0..*
Catalog
Attributes
private String id = ""
private String name = ""
Operations
public Catalog(Element elem)
public String getId( )
public String getName( )
public Table getTable(int i)
public Table getTableId(String id)
public Table getTableName(String name)
public Table getTableFlag(String flag)
public int getTableSize( )
public String myPrint( )
tables 0..*
UCD
Attributes
private String version = ""
private String content = ""
Operations
public UCD(Element elem)
public String getContent( )
public String getVersion( )
public String myPrint( )
UCDs
0..*
Column
Attributes
private String id = ""
private String name = ""
private String dataType = ""
private String units = ""
private String description = ""
Operations
public Column(Element elem)
public UCD getUCDversion(String version)
public UCD getUCDAt(int i)
public int getUCDSize( )
public String getDataType( )
public String getId( )
public String getName( )
public String getUnits( )
public String getDescription( )
public String myPrint( )
columns
0..*
Table
Attributes
private String id = ""
private String name = ""
private String description = ""
Operations
public Table(Element elem)
public Column getColumn(int i)
public String getDescription( )
public Column getColumnId(String id)
public Column getColumnName(String name)
public Column getColumnFlag(String flag)
public int getColumnSize( )
public String getId( )
public String getName( )
public String myPrint( )

Mots Clef: Astronomie, base de données, site web dynamique, requêtes HTTP, Observatoire Virtuel, XML, servlet Java, standard, étoile, méta-données.

Résumé:

Durant ce stage, ma mission a été de concevoir et implémenter un web service. Ce web service avait pour objectif de fournir un accès a un base de données dynamique de mauvais calibrateurs pour l'astronomie. La base doit être facilement alimentable par la communauté scientifique tout en évitant les détournements dont elle pourrait être l'objet. Le web service a aussi à fournir un accès destiné au machine obéissant au standard de l'Observatoire Virtuel. Une partie de l'architecture de la base de donnée nous était inconnue ce qui a eu pour conséquence de transformer le simple fait d'écrire un site web dynamique en la tache autrement plus complexe qu'est celle de créer des méthodes capable de faire le même travaille sans que rien ne soit codé en dure. Nous avons donc due faire usage de méta-données. Le service avait aussi à contrôler les nouvelles données saisies en utilisant les service du VO ce qui suppose de formuler des requete HTTP et d'interpréter le résultat.

Key Words: astronomy , database, dynamic web site, HTTP query, International Virtual Observatory Alliance, XML, Java servlet, standard , stars, meta-data.

Abstract:

During this work placement, my mission consisted in design and program a web service. This web service's goal is providing an access to a dynamic bad calibrator database for astronomers all around the world. The database must be easily feed by astronomers by a process that also avoid unexpected use. The web service have also to provide an access for computers compatible with the International Virtual Observatory Alliance standards. A part of the architecture of the database was not stable that who has for consequence to transform the simple job of dynamic web site creation into the harder problem of creating methods able to make the same job without hard coded elements. So we had to use meta-data. The application has also to use IVOA services to check new data. It suppose being able to process a HTTP query and being able to parse the result.