RAPPORT DE STAGE

Centre de Données astronomiques de Strasbourg

Étude et prototypage d'applications Internet riches

Cédric CAPOULUN
Année universitaire 2007/2008

Université Nancy 2 – IUT Nancy-Charlemagne
Licence Professionnelle Concepteur et Intégrateur de Systèmes Internet / Intranet

RAPPORT DE STAGE
Avril – Juin 2008

Centre de Données astronomiques de Strasbourg
Observatoire Astronomique de Strasbourg
11, rue de l'Université
67000 STRASBOURG

Étude et prototypage d'applications Internet riches

Cédric CAPOULUN
Année universitaire 2007/2008

Tuteurs entreprise: André Schaaff, Thomas Boch
Tuteur enseignant: Denis Roegel

Université Nancy 2 – IUT Nancy-Charlemagne
Licence Professionnelle Concepteur et Intégrateur de Systèmes Internet / Intranet

Résumé

Ce rapport concerne le stage que j'ai effectué dans le cadre de la licence professionnelle Concepteur et Intégrateur de Systèmes Internet/Intranet (CISII), et qui s'est déroulé au Centre de Données astronomiques de Strasbourg, laboratoire de recherche de l'Observatoire Astronomique.

Ce stage a un double rôle : la réalisation d'un état de l'art des technologies visant à concevoir des applications Internet riches, puis le développement de différents prototypes dans le contexte de l'astronomie.

Nous présentons ici l'ensemble du travail effectué. Après une présentation du laboratoire et de l'organisation du déroulement du stage, nous verrons l'essentiel de l'état de l'art qui nous a amené à la réalisation d'un comparatif entre les différentes technologies. Ensuite nous exposerons les différents prototypes développés, avec, pour chacun, diverses explications sur les problèmes rencontrés ainsi que les solutions apportées. Enfin, nous dresserons un bilan et conclurons sur les apports du stage.

Remerciements

Je tiens tout d'abord à adresser mes sincères remerciements à Monsieur André SCHAAFF, Ingénieur de Recherche, pour m'avoir proposé ce stage, encadré, et chaleureusement accueilli au sein du Centre de Données astronomiques de Strasbourg. Mais je voudrais surtout lui exprimer ma plus vive reconnaissance pour m'avoir permis de me rapprocher du domaine de l'astronomie, dont je suis passionné depuis toujours.

Je voudrais aussi remercier Monsieur Thomas BOCH qui m'a également encadré durant ce stage, pour sa patience et sa compréhension, ainsi que toutes les personnes avec lesquelles j'ai pu travailler, pour leur bonne humeur et leur aide qui ont permis que ce stage se déroule dans les meilleures conditions.

Merci également à Monsieur Denis ROEGEL, mon responsable enseignant de l'Institut Universitaire de Technologies de Nancy-Charlemagne.

Enfin je tiens à remercier M. Mathias DEPRETZ ainsi que l'ensemble des membres du CDS que j'ai pu côtoyer et que je n'ai pas pu citer mais qui m'ont accueilli avec sympathie et ont contribué à mon enrichissement personnel et professionnel.

Table des matières

Introduction

Ce stage intitulé « étude et prototypage d'applications Internet riches dans le cadre de l'astronomie » s'est déroulé sur le site de l'Observatoire Astronomique de Strasbourg du 7 avril au 27 juin 2008, au sein du Centre de Données astronomiques de Strasbourg (CDS).

Ce stage s'inscrit dans l'aboutissement de la formation visant à l'obtention de la Licence Professionnelle. Cette période de 3 mois en entreprise permet de mettre en pratique et de valider les connaissances et compétences acquises au cours de la scolarité, l'objectif étant de participer à la réalisation d'une véritable mission pour le compte de l'entreprise d'accueil. Le stage apporte également une aide non négligeable pour s'orienter vers un futur projet professionnel, en étant confronté à un réel environnement de travail. Mais surtout, un tel stage permet d'acquérir de nouvelles compétences, tant sur le plan technique que sur le plan humain.

Le CDS développe plusieurs services (Simbad, VizieR, Aladin, ...) utilisés par les astronomes du monde entier. Le but de mon stage était d'explorer les possibilités offertes par les applications à interface riche dites RIA (pour « Rich Internet Application »). Il s'est alors naturellement divisé en deux grandes étapes : une première partie de recherche, puis une seconde partie de prototypage.

Nous présentons dans ce rapport toutes les étapes que nous avons suivies afin de réaliser le travail demandé. Après une présentation du laboratoire et du déroulement du stage, nous entrerons dans le vif du sujet en rapportant le résultat de nos recherches, qui a notamment permis de s'orienter vers un projet de réalisation de prototype, en présentant et en comparant différentes technologies. Ensuite nous expliquerons, étape par étape, le travail effectué, en exposant les problèmes rencontrés ainsi que les solutions apportées. Enfin, nous dresserons un bilan du travail réalisé ainsi qu'un bilan général de ce stage.

I. Présentation de l'entreprise

1. Structure juridique

L'Observatoire de Strasbourg est une Unité de Formation et de Recherche (UFR) de l'Université Louis Pasteur. Il s'agit également d'une Unité Mixte de Recherche du CNRS et de l'Université Louis Pasteur (UMR 7550).

2. Présentation générale

L'Observatoire est situé sur le campus de l'Esplanade ; ses bâtiments font partie du campus historique de l'université de Strasbourg, qui dispense les enseignements suivants :

- Master « Analyse et Traitement des Données sur les Milieux Astronomiques »
- Licence en Sciences et autres licences (enseignements d'ouverture)
- Licence de Géosciences
- Préparation au CAPES et à l'Agrégation
- Master Applications des Technologies Spatiales
- Diffusion de la Culture

La partie publique de l'Observatoire, le Planétarium, est ouvert pour la vulgarisation de l'astronomie. L'Observatoire se compose de quatre équipes de recherche : hautes énergies, étoiles et systèmes stellaires, galaxies, et le CDS.

2.1. Hautes énergies

L'équipe Astrophysique des Hautes énergies a pour thème l'étude des astres et sites de l'univers émetteurs de photons de haute énergie. Cette thématique générale recouvre des aspects variés, comme l'étude des astres compacts en fin d'évolution, la physique de leur activité, les phénomènes de haute énergie intéressant les étoiles jeunes ou le soleil, ou l'étude de ces phénomènes à l'échelle galactique. Ces recherches se sont largement appuyées sur les données acquises par le satellite ROSAT et s'appuieront dans l'avenir sur celles des satellites X de nouvelle génération, tout spécialement XMM[1].

2.2. Étoiles et systèmes stellaires

Les recherches menées par l'équipe « Étoiles et systèmes stellaires » recouvrent un domaine étendu, incluant les étoiles, les milieux interstellaires, la Galaxie et les galaxies proches. De l'étoile, objet individuel, l'intérêt s'est porté aux groupes d'étoiles, témoins de l'évolution stellaire mais aussi traceurs des grandes structures de la Voie Lactée.

2.3. Galaxies

Les activités de l'équipe sont centrées sur les problèmes de la structure du Groupe Local, de ses populations stellaires et sur la dynamique gravitationnelle. De plus, l'équipe possède un savoir-faire sur les outils statistiques d'analyse de données et sur les méthodes inverses non paramétriques. Un des objectifs consiste à combiner les informations d'évolution des populations stellaires et celles de dynamique afin de reconstituer les événements déterminants liés aux processus de formation et d'évolution galactique.

2.4. Le centre de données (CDS)

L'activité de recherche du CDS s'est concentrée sur l'étude de la dynamique galactique et des populations d'étoiles binaires, sur une participation importante à la mission HIPPARCOS[2] de l'Agence Spatiale Européenne, ainsi que sur le développement de méthodologies nouvelles applicables à l'analyse et au traitement de données astronomiques.

3. Organisation du personnel

Le CDS est un laboratoire de l'Institut National des Sciences de l'Univers (INSU), rattaché au CNRS. L'Observatoire de Strasbourg est un institut de l'Université Louis Pasteur. Le personnel du CDS comprend une douzaine de chercheurs, 12 ingénieurs, 3 techniciens, et plusieurs collaborateurs à contrat temporaire et invités.

4. Centre de Données astronomiques de Strasbourg (CDS)

J'ai effectué mon stage au sein du Centre de Données astronomiques de Strasbourg (CDS) qui est un centre de données dédié à la collection et à la distribution dans le monde entier de données astronomiques.

Le CDS héberge la base de données Simbad, la base de référence mondiale pour l'identification d'objets astronomiques. Le but du CDS est de :

- rassembler toutes les informations utiles, concernant les objets astronomiques, disponibles sous forme informatisée : données d'observations produites par les observatoires du monde entier, au sol ou dans l'espace
- mettre en valeur ces données par des évaluations et des comparaisons critiques
- distribuer les résultats dans la communauté astronomique
- conduire des recherches utilisant ces données

Le CDS joue, ou a joué, un rôle dans d'importantes missions astronomiques spatiales : contribuant aux catalogues d'étoiles guides, aidant à identifier les sources observées ou organisant l'accès aux archives, etc. Le CDS contribue au XMM SSC[3], sous la responsabilité

de l'équipe « Hautes-Énergies » de l'Observatoire de Strasbourg. Le service a également signé des accords d'échanges internationaux avec les organismes suivants :

- NASA
- National Astronomical Observatory (Tokyo, Japon)
- l'Académie des Sciences de Russie
- le réseau PPARC Starlink au Royaume-Uni
- l'Observatoire de Beijing (Chine)
- l'Université de Porto Allegre au Brésil
- l'Université de La Plata en Argentine
- InterUniversity Center for Astronomy and Astrophysics (Inde)

Le CDS est membre de la Fédération des Services d'Analyse de Données Astrophysiques et Géophysiques.

Le CDS coopère aussi avec l'Agence spatiale Européenne (transfert au CDS du service de catalogues du projet ESIS : le projet VizieR), et avec la NASA : en particulier le CDS abrite une copie miroir du Système de Données Astrophysiques (ADS) et ADS abrite une copie miroir de Simbad. Le CDS contribue aussi au projet NASA AstroBrowse. Le CDS abrite aussi les copies miroirs Européennes des journaux de l'American Astronomical Society (AAS).

5. Les Services du CDS

5.1. Simbad

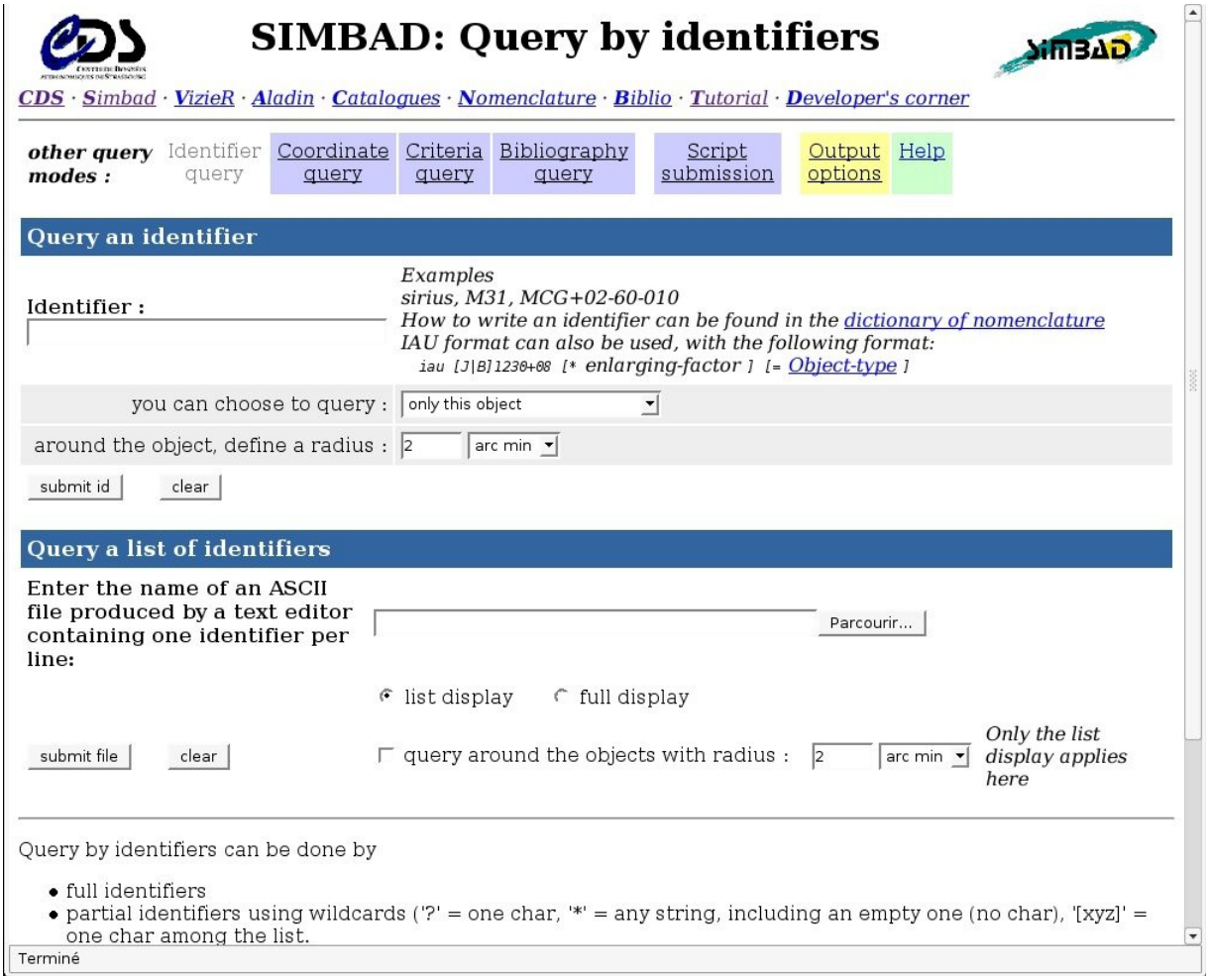


Figure 1.1 - Page Web permettant d'effectuer une requête sur Simbad.

Simbad est une base de données de référence pour les identifications et la bibliographie d'objets astronomiques. Simbad contient plus 11 millions d'identificateurs pour plus de 3 millions d'objets différents. Pour chaque objet figurent dans la base quelques mesures (position, magnitude dans différents domaines de longueurs d'ondes), ainsi que les références bibliographiques où l'objet est cité (plus de 110 000 articles sont concernés). L'utilisateur peut choisir le format du fichier où seront entreposés les résultats de la requête (Figure 1.1). En effet, Simbad peut générer des fichiers HTML[4], XML[5] ou XLS (fichiers Excel).

Cet ensemble de données résulte d'un long travail d'identification croisé entre de nombreux catalogues, listes d'objets et articles de journaux, entrepris au début des années 1980, et constamment développés et mis à jour depuis.

5.2. VizieR

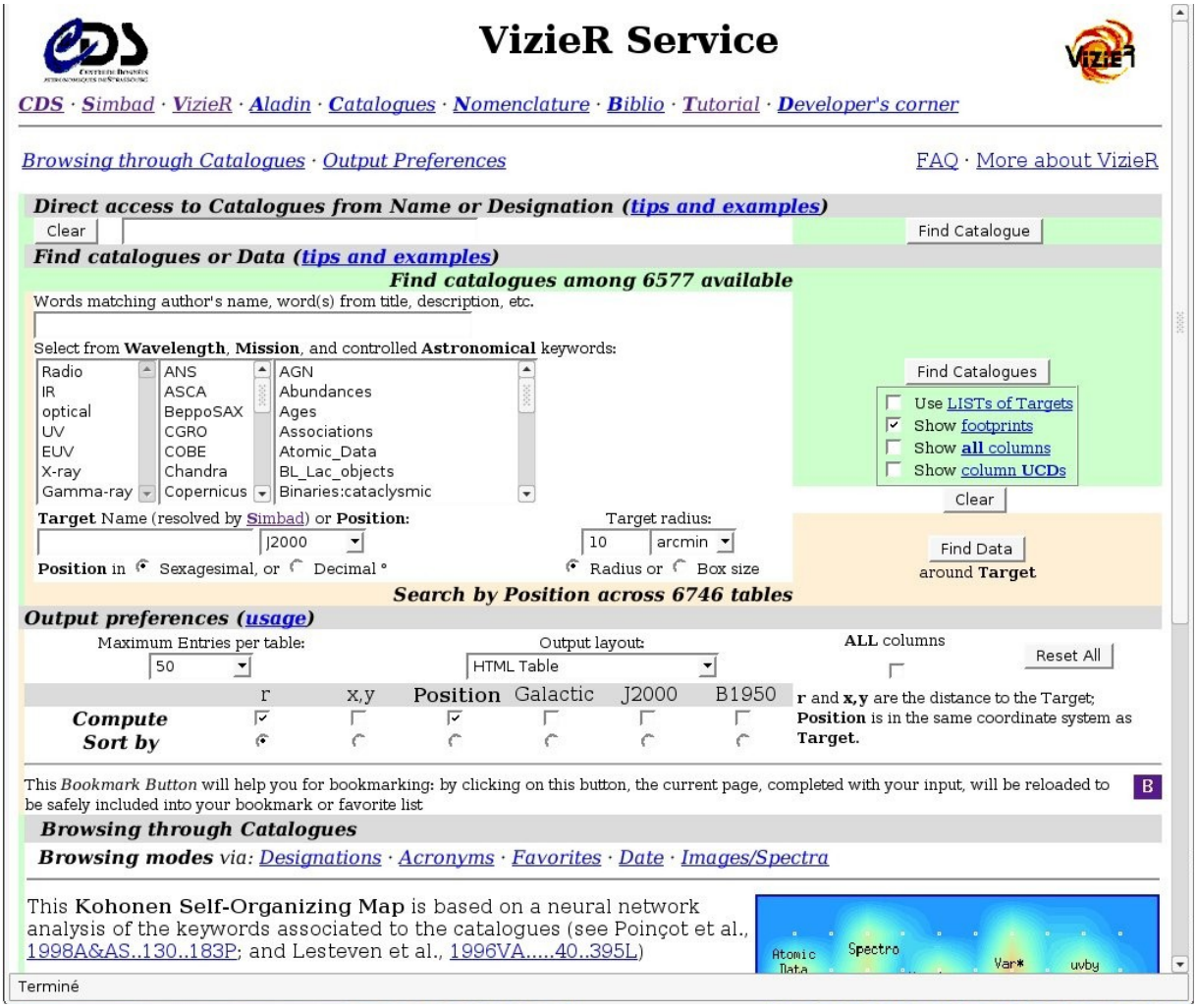


Figure 1.2 - Page Web permettant d'effectuer une requête sur VizieR.

VizieR est une base de données rassemblant plusieurs milliers de catalogues astronomiques sous un format homogène. Une description standardisée du contenu des catalogues permet leur inclusion dans un système de gestion de base de données (SGBD) relationnel. Un ensemble de liens, entre les tables de VizieR, et avec des services externes (bibliographiques, archives externes, serveurs d'images), permettent de naviguer entre les données des catalogues et d'autres données associées (Figure 1.2). Il faut noter que les très grands catalogues (plus de 10^7 enregistrements) ne peuvent pas être gérés par un SGBD relationnel pour des raisons de performances. Des outils spécifiques doivent être utilisés.

5.3. Aladin

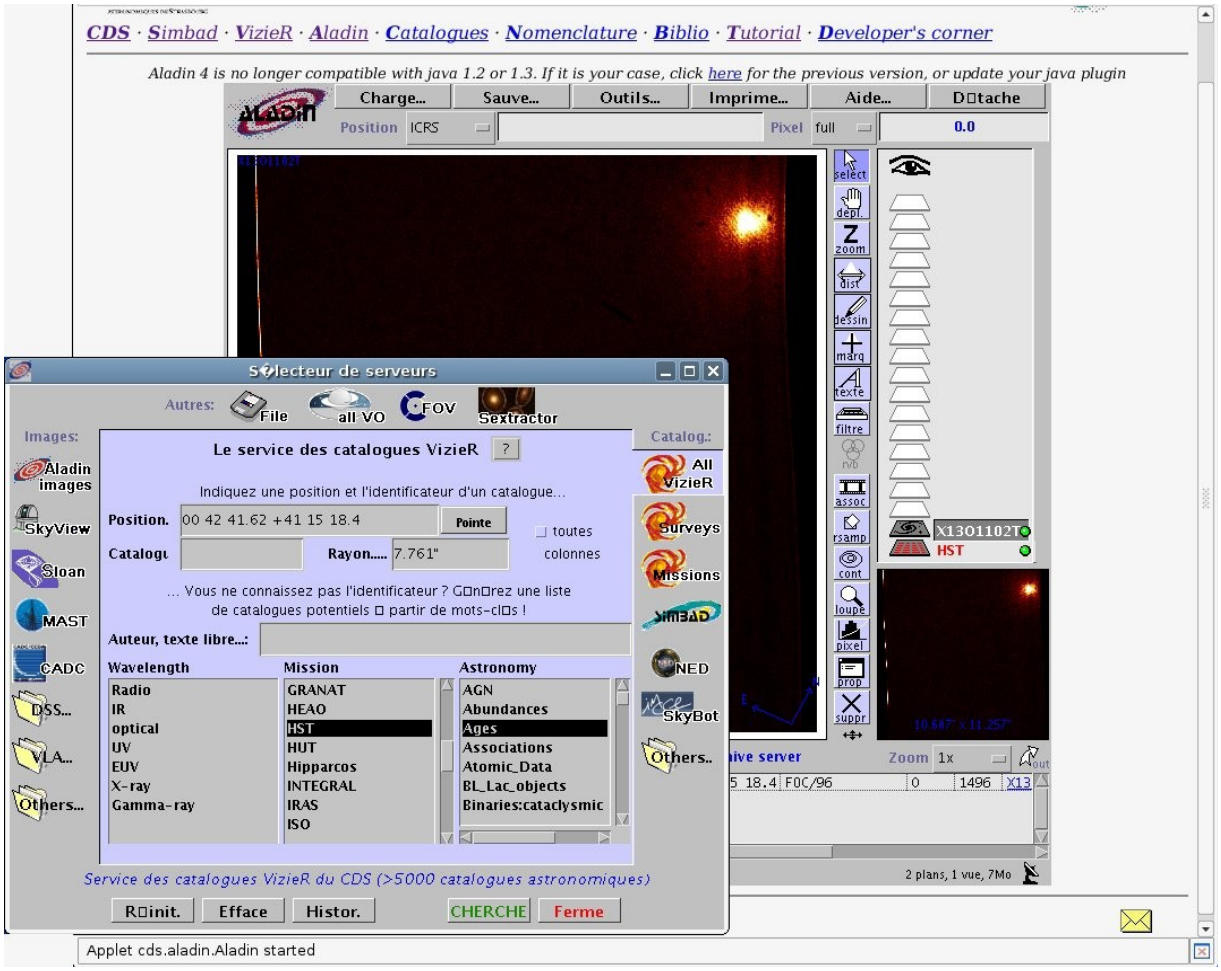


Figure 1.3 - Exemple d'utilisation d'Aladin.

Aladin (Figure 1.3) est un atlas interactif du ciel permettant d'accéder simultanément à des images numérisées du ciel, ainsi qu'à des catalogues et bases de données astronomiques. Cet outil permet de superposer, sur des images du ciel optique, les objets présents dans Simbad, des sources de catalogues contenus dans VizieR, mais aussi d'autres données, locales ou situées sur des serveurs distants.

5.4. Les autres services

Parmi les autres services offerts par le CDS, on peut citer le dictionnaire de nomenclature, les services bibliographiques et les services pages jaunes (AstroWeb, Sar's family, AstroGlu).

6. L'Observatoire Virtuel

Jusqu'à une époque récente, un grand nombre de projets ne prévoyaient pas de rendre accessibles les données issues des missions (spatiales, terrestres). Depuis l'avènement des nouvelles technologies, Internet et les réseaux rapides, d'importants efforts ont permis de mettre à disposition des astronomes de nombreuses sources de données (on peut citer par exemple les services VizieR, Simbad et Aladin du CDS).

Depuis quelques années, l'interopérabilité entre ces services est devenue une priorité car elle permet un accès aisé à l'information, un croisement des données de toutes natures et elle conduit au concept d'Observatoire Virtuel. L'astronome disposera à terme de nouveaux instruments utiles à sa recherche au travers d'une simple interface Web. Pour y parvenir, de nombreuses collaborations sont indispensables afin d'aboutir aux consensus indispensables à la pose des bases nécessaires à la « construction » de l'Observatoire Virtuel (OV). Les projets nationaux (OVF pour la France, GAVO pour l'Allemagne, le NVO pour les États-Unis, ...) et transnationaux (ESA VO, ESO VO, ...) sont réunis au sein de l'International Virtual Observatory Alliance. Ils participent à l'élaboration de Recommandations dans divers domaines (Modèles de données, Accès aux données, Sémantique, Grilles, etc.) au travers de groupes de travail se réunissant semestriellement. A titre d'exemple significatif, la première recommandation, « VOTable », décrit la formalisation de tables de données astronomiques au format XML. Celle-ci a été adoptée par de nombreux fournisseurs de données et intégrée dans les outils de l'OV.

II. Déroulement du stage

1. Sujet en détail et ses objectifs

Le stage a pour but d'étudier les apports éventuels des RIA[6] dans le cadre du CDS et des services qu'il propose. Ce stage a donc un double objectif : d'une part fournir un état de l'art sur les RIA, et d'autre part le développement d'un ou plusieurs prototypes d'application pour en mesurer la pertinence.

Nous nous sommes donc d'abord concentrés sur les RIA en général afin d'en comprendre le concept ainsi que d'en retirer une ou plusieurs définitions précises.

Ensuite, nous nous sommes penchés sur les diverses technologies actuelles et fonctionnelles (ou en passe de le devenir) permettant la réalisation d'applications à interface riche, pour réaliser un état de l'art. Cette étude, même si elle reste générale, a toutefois été orientée afin de pouvoir mettre en évidence le respect ou non de certaines contraintes liées au contexte du CDS, amenant à la réalisation d'un comparatif.

La rédaction de cet état de l'art doit donc permettre d'entrevoir les possibilités offertes par les applications à interface riche, mais pas seulement. En effet, son rôle est également de servir de support pour le CDS dans le cas où il serait décidé d'entreprendre le développement d'une application avec une technologie RIA. Cette documentation permet alors de savoir quels outils utiliser, les compatibilités, ou encore de visualiser des exemples concrets grâce à des références vers des sites ou blogs hébergeant ce type d'application. Il s'agit en fait de mettre à disposition une première base de connaissances sur le sujet avant tout travail de recherche ou d'application.

La partie « prototypage » de ce stage a quant à elle une approche plus pratique que théorique. En effet, elle a pour but de permettre le développement d'une ou de plusieurs applications dont le rôle est principalement de mettre en place une première expérimentation de la technologie choisie. Les prototypes à développer sont en relation avec les besoins du CDS afin de pouvoir les exploiter par la suite, s'ils s'avèrent pertinents. L'intérêt est donc d'explorer les fonctionnalités que met à disposition la technologie et d'en tirer des conclusions sur ses capacités et ses limites.

2. Organisation

Avant de commencer la présentation du travail effectué durant ce stage, je tiens à donner un bref aperçu de notre organisation afin de répondre au mieux au sujet donné.

Comme nous l'avons énoncé plus haut dans ce rapport, la première partie de ce stage (les trois premières semaines) était dédiée à la réalisation d'un état de l'art. Dans un premier temps il a fallu se familiariser avec le concept et les enjeux des applications Internet riches. Après un tri parmi les données recueillies, nous avons fait une liste, la plus exhaustive possible, des différentes technologies répondant aux critères des RIA.

Pour chacune de ces technologies, nous avons étudié son principe de fonctionnement. Dans les cas où cette technologie semblait intéressante, nous installions les différents outils permettant son développement seulement s'ils étaient gratuits ou présentaient une version d'évaluation, et s'ils étaient compatibles avec le système d'exploitation de la machine mise à ma disposition (Linux Ubuntu). Nous avons alors pu réaliser quelques petits tests qui nous ont permis de nous faire une idée sur la facilité de mise en oeuvre de cette technologie, ainsi que de l'explorer plus avant, notamment en ce qui concerne la richesse des composants qu'elle propose. Lorsqu'une technologie était propriétaire ou qu'il n'existait encore aucun outil de développement, nous nous rabattions sur les descriptions faites sur différents sites Internet ou, le cas échéant, sur des exemples existants. Une fois cette première expérimentation faite, je rédigeais une documentation présentant l'ensemble des résultats de mes recherches sur le Twiki.

Le Twiki est un site communautaire du même genre que Wikipédia. Ce site est interne au CDS et seuls les membres du CDS peuvent y accéder au moyen d'un nom d'utilisateur et d'un mot de passe. Une personne autorisée peut ainsi suivre l'évolution des projets du CDS et éventuellement ajouter ou modifier des articles sur le ou les projets auxquels il participe. Ainsi, il est possible de suivre l'évolution du travail effectué durant ce stage.

Il est arrivé que l'étude approfondie d'une technologie nous mette sur la piste d'une autre RIA qui n'était pas apparue lors de l'étude préalable, s'ajoutant alors à notre liste initiale. Par ailleurs, nous avons tout de même réalisé une courte documentation pour les technologies que nous n'avons pas approfondi, soit parce qu'elles avaient déjà été étudiées au sein du CDS (AJAX[7] par exemple) ou bien parce qu'elles semblaient nettement en retrait face aux autres RIA.

Nous avons conclu cet état de l'art par un comparatif entre ces différentes technologies. Celui-ci prend en compte de nombreux critères, notamment au niveau de leurs avantages et inconvénients, afin de déterminer l'intérêt d'une utilisation au CDS. Nous reviendrons sur le choix de ces critères dans la partie « Les applications Internet riche » de ce rapport.

J'ai ensuite présenté les conclusions de mon travail lors d'une première réunion entre toutes les personnes impliquées dans le suivi de ce stage. Nous avons décidé d'expérimenter certaines technologies et des premières idées de prototypes sont apparues.

La seconde partie de ce stage a été entièrement consacrée à la réalisation de prototypes, encadrée par des réunions régulières. Ces réunions se déroulaient ainsi : je présentais l'avancement de mon travail, puis nous discutions des modifications éventuelles à apporter, et enfin nous réfléchissions sur les étapes suivantes.

Bien que sans cahier des charges ni planning précis au départ, les premières réunions ont permis l'établissement d'une feuille de route pour mener à bien le stage. Nous avons ainsi pu réaliser trois prototypes, dont les rôles sont relativement divers.

Pour présenter le travail effectué durant ce stage nous évoquerons la réalisation de l'état de l'art puis le développement de prototypes.

III. Les applications Internet riches

1. Définition

Le terme « Rich Internet Application » (RIA) a été introduit dans une publication de Macromedia (désormais Adobe) en mars 2002 pour décrire un nouveau type d'application Web possédant une interface utilisateur plus poussée, plus conviviale, et surtout plus intuitive que les interfaces actuelles. Mais le concept n'est pas nouveau ; on retrouve ainsi le « Remote Scripting » de Microsoft (1998), le « X Internet » de Forrester Research (2000), ou encore le terme « Rich Web Client ».

Même si cette définition reste d'actualité, le terme RIA a depuis été largement repris et associé à divers concepts. Ainsi, on peut également considérer les RIA comme des applications dont les caractéristiques graphiques mais également applicatives sont semblables aux applications autonomes s'exécutant sur un ordinateur. À long terme, les RIA favoriseraient la fusion des logiciels traditionnels et des applications de type client-serveur, en permettant notamment à une application Web de fonctionner en local, c'est-à-dire hors ligne, en rapatriant toutes les informations dont elle a besoin sur le poste client, dans des fichiers de type cookie ou bien dans des petites bases de données temporaires.

Toutefois, il est utile de garder à l'esprit le fait que les standards Internet ont évolué lentement mais continuellement à travers le temps pour s'accommoder avec ces nouvelles techniques, ainsi il est difficile de définir clairement ce qui constitue une RIA. On peut cependant qualifier de RIA une application qui n'est limitée que par les capacités du système client, et c'est parce que l'application est déportée sur le client qu'on peut voir apparaître de nombreuses fonctionnalités - difficiles à mettre en œuvre jusque là - telles que le copier-déplacer, le glisser-déposer, des barres d'outils permettant la modification de données, ou encore des calculs effectués par le client sans avoir à transiter par le serveur.

Enfin, alors que certains, par abus de langage, désignent par RIA les diverses technologies, outils, ou plateformes qui permettent de développer des applications Internet dont l'interface homme-machine est riche en composants et en fonctionnalités, d'autres associent les RIA à la notion de « Write once, run everywhere », c'est-à-dire des applications portables qui peuvent s'exécuter sur toutes les plateformes, tous les périphériques et tous les systèmes.

2. État de l'art

Nous allons maintenant présenter succinctement chaque technologie étudiée, accompagnée d'une courte conclusion sur son apport du point de vue des RIA, pour finir par un tableau comparatif avec l'explication du choix des critères de comparaison.

2.1. Silverlight

Silverlight est une technologie Microsoft qui permet de développer des applications Web enrichies d'animations, de tracés vectoriels, de retransmission audio et vidéo. Il existe actuellement deux versions de Silverlight : la 1.0 et la 2.0. Côté client, cette technologie se présente sous la forme d'un plugin à installer sur son navigateur Web et qui permet d'exécuter le code produit. Du côté programmation, développer pour Silverlight repose sur le principe du « Code-Behind » : d'une part la conception du modèle graphique avec l'utilisation du langage à balises dérivé du XML : XAML[8] (le langage déclaratif de présentation des applications Windows Vista), et d'autre part l'utilisation d'un langage permettant d'implémenter la logique applicative. Le langage utilisé dépend alors de la version de Silverlight pour laquelle on souhaite développer. Pour la version 1.0, il faut utiliser du Javascript (avec la possibilité d'appliquer le concept AJAX), tandis que la version 2.0 (basée sur le framework .NET) accepte, en plus du Javascript, l'utilisation de PHP, Ruby, Python, C#, NET.

Silverlight a de nombreux avantages non négligeables tels que la possibilité d'utiliser plusieurs langages au sein d'un même projet (ce qui permet notamment une réutilisation de code ou encore l'utilisation du C# qui est un langage puissant), une volonté de devenir multi-plateformes et multi-navigateurs, gain de productivité de par la séparation du développement de la logique applicative et du développement de l'interface graphique, support de la HD[9], de la 2D, ou encore du streaming.

Mais cette technologie présente également des inconvénients majeurs comme celui de ne pas être disponible officiellement sur les plateformes Unix/Linux (mais un portage non officiel existe actuellement en version beta), de rendre obligatoire l'utilisation du framework .NET et des plateformes de développement associées payantes (Visual Studio) pour développer des applications, ou encore son faible taux de pénétration sur le marché actuel (même si l'on peut supposer qu'une diffusion massive et rapide serait envisageable via les mises à jours automatique de Windows).

2.2. Openlaszlo

Openlaszlo est une plateforme de développement (framework) open source pour applications Web RIA. Les applications développées avec Openlaszlo se basent sur une grammaire XML[5] appelée LZX. Il s'agit en fait d'un langage couplant XML et Javascript pour permettre de construire rapidement une interface graphique. Openlaszlo peut alors soit déployer l'application dans un servlet java traditionnel (dans ce cas le code LZX est compilé et envoyé au navigateur dynamiquement par le servlet, on parle de déploiement Proxy), soit compiler le code LZX en DHTML ou dans le format binaire propriétaire Flash SWF[10] (déploiement Solo).

Openlaszlo a l'avantage de permettre la génération aussi bien de code binaire Flash que de DHTML[11], on peut donc considérer que ces applications sont multi-plateformes et multi-navigateurs. Le fait que ce framework soit gratuit, open source et qu'il ne dépende pas d'une plateforme de développement particulière (et surtout payante) font de cette technologie un acteur intéressant sur le marché des RIA. Sa syntaxe simple, l'utilisation de standards reconnus, le support (en partie) des normes d'accessibilités de la spécification MSAA[12], sa stabilité, son ancienneté, et le support de Sun et IBM à cette technologie sont également des facteurs à prendre en compte.

Toutefois, Openlaszlo présente des inconvénients important à souligner : une technologie qui a du mal à s'imposer, malgré le fait qu'elle soit pionnière sur un marché présentant de nombreux concurrents, des applications moins performantes que celles produites avec la technologie Flex (concurrent direct) et le peu de réactivité de la communauté officielle. Le problème avec Openlaszlo, c'est que l'on ne perçoit pas de réelle évolution possible pour cette technologie qui est plutôt orientée multi-plateformes que RIA.

2.3. JavaFX

JavaFX est une famille de produits de Sun Microsystems, produits qui ont pour but de créer des applications internet riches. Actuellement, JavaFX est constitué de JavaFX Script et de JavaFX Mobile, d'autres produits sont également prévus. JavaFX Script est un langage de script déclaratif basé sur la plateforme Java et permettant de faciliter le développement d'interfaces graphiques, tandis que JavaFX Mobile est une plateforme mobile permettant l'exécution de code Java sur des téléphones portables, des PDA, et autres appareils mobiles, de manière plus flexible que l'actuel J2ME. Pour le moment sous licence, JavaFX Script devrait devenir Open Source (GNU Public License v2). Sa syntaxe est relativement différente de celle de Java, mais elle permet une interaction forte avec les classes Java, et beaucoup de classes de JavaFX implémentent les fonctionnalités Swing et Java2D, facilitant leur utilisation. De plus, du code JavaFX peut embarquer du code Java ou HTML.

JavaFX a l'avantage de s'exécuter dans une JVM[13], un plugin fortement présent sur les machines clientes et compatible multi-navigateurs et multi-plateformes. Le fait que cette technologie soit gratuite et repose sur une communauté très active n'est bien sûr pas négligeable, l'utilisation de la puissance du langage Java alliée à une syntaxe simplifiée l'est

encore moins, mais le réel atout de JavaFX est son interaction forte avec les classes de composants graphique de Java, ce qui permettrait de concevoir de vraies interfaces riches et proches des applications de bureau.

Mais JavaFX ne présente pas que des avantages. En effet, même si cette technologie est encore en cours de développement, elle doit faire face à des concurrents déjà en place. De plus, JavaFX n'apporte pas de réelle nouveauté puisque son principe repose sur celui des applets, une technologie éprouvée mais pour laquelle Java est déjà derrière ses concurrents. Finalement JavaFX est présenté comme une solution RIA, mais c'est en fait une nouvelle solution pour construire des applications à interface riche, qu'elles soient Web ou de bureau, et dans le cas d'une application Web, il faudrait utiliser les applets, ce qui donne l'impression de faire du neuf avec du vieux.

2.4. XUL

XUL (XML-based User interface Language) est un langage de description d'interfaces graphiques fondé sur le XML[5] et créé dans le cadre du projet Mozilla. XUL comprend un ensemble de balises permettant de définir tous les éléments d'une véritable interface utilisateur riche : des boutons, des listes, des menus, des zones d'édition, etc. De plus, il est possible de définir ses propres balises pour mettre en place des composants complexes et réutilisables en écrivant un fichier XBL (eXtensible Binding Language). Celui-ci comporte, pour chaque nouveau composant qu'il décrit, une partie spécifiant l'aspect graphique du composant, et plusieurs parties précisant son comportement, sous la forme de fonctions Javascript. Initialement, XUL a été développé afin de faciliter la création de l'interface graphique du navigateur Web Mozilla. Un moteur XUL est intégré au moteur Gecko de Mozilla, ce qui permet l'exécution de code XUL directement dans le navigateur.

Cette technologie a l'avantage de permettre la création de réelles interfaces riches équivalentes à celles des applications de bureau. De plus, le support d'un ensemble de technologies standardisées par le W3C[14] est un plus qui ne peut être nié. Enfin, de part la possibilité d'exécuter une application XUL de manière identique à l'intérieur d'un navigateur ou directement sur le bureau de la machine cliente, certains y voient l'avenir des RIA.

Mais il subsiste un inconvénient de taille. En effet, l'exécution d'une application XUL nécessite la possession d'un navigateur utilisant le moteur Gecko. Cette technologie est donc loin d'être multi-navigateurs. A cela on peut ajouter une communauté peu active et des outils de développement peu évolués.

2.5. Volta

Volta est une nouvelle technologie RIA de Microsoft, qui est encore en développement. Volta se décompose en deux fonctionnalités importantes : un outil de « lean programming » et un compilateur de code MSIL[15].

Volta est un outil de « lean programming », ce qui signifie que les développeurs n'ont plus à se soucier de ce qui constitue les parties clientes et serveurs pendant leurs développements. Ils décident ensuite quelle partie de l'application doit être prise en charge par le serveur, et quelle partie doit résider sur le poste client, Volta s'occupant du refactoring du code.

Mais le but premier de Volta est de permettre la conversion de n'importe quelle application développée sur la plateforme .NET en application Web. Tous les langages supportés par la plateforme .NET produisent, à la compilation, un code MSIL. Volta permettrait alors de convertir n'importe quel code MSIL en code Javascript structuré (classes totalement réécrites en Javascript) à l'image de GWT[16] de Google qui produit du code Javascript à partir d'un développement en Java. Une application .NET passerait alors à une application Web 2-tiers : formation de classes serveur (en .NET) et des classes client (en Javascript). Pour aller encore plus loin, les concepteurs de Volta parlent d'un outil qui permettra de développer en .NET et de choisir la plateforme vers laquelle compiler (AJAX, Flash/Flex, Silverlight) à l'instar d'Openlaszlo (qui se limite lui au DHTML et au Flash). Dans le cas où le CLR[17] choisi ne serait pas disponible sur la machine du client, Volta pourra de lui-même compiler l'application vers un autre CLR qu'il aurait détecté sur le client. Côté serveur, Volta pourrait donc compiler une application vers n'importe quel langage à la volée, et serait donc 100% compatible avec les différents systèmes et navigateurs.

A l'heure actuelle, une version expérimentale de Volta est téléchargeable mais très peu de fonctionnalités énoncées sont implémentées : la compilation ne se fait que vers du Javascript et il ne supporte que quelques fragments des bibliothèques du framework .NET. Il n'est donc pas possible d'étudier cette technologie, mais si ses concepteurs tiennent leurs promesses, Volta pourrait devenir la meilleure solution RIA, alliant performances et compatibilités.

2.6. HTML 5

HTML est en constante évolution depuis son introduction sur Internet au début des années 1990. Certaines des modifications de ses caractéristiques ont été introduites dans des spécifications, tandis que d'autres ont été apportées, au fur et à mesure, par des acteurs du Web, et notamment les sociétés mettant à disposition des logiciels d'édition de ce langage.

La dernière spécification de HTML, HTML4, a été publiée en 1999 ; et depuis le W3C s'est tourné vers XHTML[18]. Mais XHTML 2 suscite la controverse, outre le fait qu'il est incompatible avec les précédents standards, on lui reproche son orientation purement document alors que le Web, qui devient 2.0, a besoin de plus en plus d'applications et utilise des médias variés. De cette opposition naît en 2004 un groupe de travail indépendant soutenu par Apple, Mozilla et Opera, le « WHATWG[19] » qui entreprend la définition d'un successeur au HTML. Le 3 Mars 2007, le W3C annonce qu'il reprend le travail concernant HTML pour lui donner un successeur.

HTML5 est la cinquième révision majeure du langage cœur du World Wide Web avec pour but de remplacer à la fois HTML4 et XHTML. Le 22 janvier 2008 est publié le « W3C Working Draft », qui pose les bases de HTML5. Le W3C estime que la recommandation pour ce langage sera disponible pour 2010.

Ce nouveau format ne se veut plus être un simple format de document, mais aussi un support pour les applications Web courantes telles que forum, wiki, recherches, e-mail, etc. et veut faciliter l'interopérabilité. Les possibilités de HTML5 incluent également le graphisme et l'image, ce qui permettra de faire des pages animées, des jeux en ligne multi-joueurs, etc.

En HTML5 il est possible d'utiliser deux syntaxes: l'une proche du HTML actuel, l'autre conforme à une syntaxe XML[5]. Le langage subit des modifications au niveau des éléments et des attributs (ajout, suppression, modification), ainsi que des extensions (ajout d'API[20]). Il intégrera, on outre, une autre spécification en cours: WebForms 2.0.

La plus grande partie de ces nouvelles fonctionnalités n'étant pas encore implémentées dans les navigateurs actuels, il est difficile à l'heure actuelle d'évaluer la pertinence et l'influence qu'aura HTML5 sur le marché des RIA. Toutefois on peut souligner le soutien des acteurs majeurs du Web (Apple, Mozilla, Opera, Google), ainsi que la volonté d'en faire un standard, gratuit et multi-plateformes.

Toutefois HTML5 devra faire face à une concurrence qui sera alors bien en place. De plus, le manque de soutien de Microsoft (Internet Explorer étant le navigateur le plus utilisé) peut également porter préjudice à cette technologie encore peu approfondie et peu implémentée.

2.7. Flex

Flex est le framework d'Adobe permettant de créer des applications Web dites RIA. Disponible depuis fin février 2008 dans sa version 3, Flex est une technologie qui permet de réaliser des applications au format SWF facilement, grâce à deux langages : une grammaire MXML, basée sur le langage XML, et Actionscript 3. MXML sert principalement à définir l'interface graphique de l'application en gérant l'agencement des composants dont deux types sont disponibles : les conteneurs (boîtes, panneaux, fenêtres, etc.) et les éléments de contrôle (champs texte, listes, tree, etc.) ; mais MXML peut également être utilisé pour définir de manière déclarative des aspects non graphiques d'une application, comme l'accès à des sources de données côté serveur par exemple. Les balises MXML correspondent à des classes ou des propriétés de classe Actionscript. Actionscript 3 permet donc d'étendre le périmètre fonctionnel de l'application notamment en enrichissant les contrôles. Même si à l'instar de Flash, Flex permet de générer des fichiers SWF, il s'agit tout de même d'une approche vraiment différente entre ces deux technologies (qui restent néanmoins compatibles). Flex est un outil plutôt orienté pour les développeurs, permettant de s'affranchir des aspects purement graphiques de Flash et de développer très rapidement des applications riches en composant, tandis que Flash demeure l'outil le plus approprié pour la réalisation d'animations.

Flex s'appuie sur le Flash Player dans sa version 9 pour s'exécuter, un plugin qui serait installé sur plus de 800 millions de postes connectés à Internet (soit 95% du parc informatique mondial, d'après une étude commandée par Adobe et publiée sur son site officiel). On peut donc objectivement dire qu'une application Flex peut être facilement déployée, et c'est un de ses atouts majeurs. Mais ce n'est pas le seul. En effet, Flex (en partie open source), permet l'augmentation du nombre d'interactions avec l'utilisateur grâce à des interfaces très riches en composant, interfaces qui peuvent se reposer sur l'existence de nombreuses bibliothèques et composants gratuits diffusés par une communauté très active. Il s'agit d'un produit stable, respectueux des standards, qui semblent se tailler une part du marché des RIA.

On trouve peu d'inconvénients à Flex, si ce n'est le manque de puissance du Flash Player face à la JVM de Java. De plus, il n'existe pas de rétrocompatibilité avec les versions antérieures des Flash Player pour les applications développées en Flex 3.

2.8. Les autres technologies

Nous avons étudié d'autres technologies uniquement « en surface » sans réellement les approfondir, soit parce qu'elles ne présentaient pas d'intérêt du point du vue des RIA, soit parce que ces technologies ont déjà fait l'objet d'une étude sein du CDS.

2.8.1 GWT

GWT[16] est un framework qui permet le développement d'applications AJAX en utilisant Java : le principe est de développer en Java avec des composants dont le fonctionnement est proche de Swing. Dans la phase de déploiement, le compilateur GWT traduit l'application vers une application Web utilisant Javascript, HTML, et DOM[21].

Il existe de nombreux composants ainsi que des librairies tierces. On peut également souligner la compatibilité avec Ajax ou encore la simplicité de développement et la productivité accrue. Mais il faut savoir que le développement est limité à la version 1.4.2 de java et que le compilateur manque de fiabilité.

2.8.2 EclipseRAP

EclipseRAP est un framework qui permet le développement d'applications AJAX en utilisant Java, à l'instar de GWT. Mais à la différence de GWT, la majeure partie des traitements est confinée sur le serveur, ce qui permet d'accéder à la totalité de l'API Java, notamment les composants SWT[22] et JFace.

Cette technologie permet d'exploiter entièrement les librairies Java d'où une richesse des composants SWT. Toutefois les nombreux échanges avec le serveur ne permettent pas de considérer une application EclipseRAP comme réellement une RIA.

2.8.3 Wazaabi

Wazaabi est un framework qui permet le développement d'applications J2EE basées sur EclipseRCP et dont les interfaces graphiques sont déclarées dans des fichiers XML utilisant la norme ZUL.

Wazaabi a pour avantages d'être open source, de permettre une séparation du développement IHM[23]/logique applicative et de réduire au maximum les échanges client/serveur. Mais c'est une technologie trop jeune et assez lourde à mettre en œuvre.

2.8.4 HaXe

HaXe est un nouveau langage. Côté client, il peut être compilé en SWF ou en DHTML. Côté serveur, il est exporté en bytecode Neko qui peut être interprété par un serveur particulier.

HaXe est un langage intéressant, notamment grâce à une syntaxe simplifiée et à l'existence d'outils de développement. Mais les bibliothèques sont très limitées pour cette technologie encore jeune.

2.8.5 Curl

Curl est un langage qui permet, entre autres, de développer des applets à interface riche avec une syntaxe qui lui est propre.

Curl est stable et compatible Windows/Linux, présente des outils de développement, et est un des pionniers des RIA. Mais la nécessité d'installer un plugin peu répandu et d'utiliser une technologie peu connue ne permet pas d'envisager de réelles évolutions pour ce langage.

3. Conclusion

Finalement, avec toutes les définitions et tous les concepts que nous avons vus, on peut être sûr que les RIA apportent une dimension nouvelle au terme « Application ». A l'heure où le Web qui devient 2.0 place l'utilisateur au centre des préoccupations, les RIA se situent dans cette mouvance technologique qui va permettre de rendre très attractive une application client-serveur. Cette approche permet d'apporter une interactivité et une ergonomie plus poussées, plus fluides, plus intégrées, et surtout une expérience utilisateur proche de celle que l'on retrouve avec des applications de bureau, au point de pouvoir faire oublier à l'internaute qu'il est en train d'exécuter une application dans son navigateur Web.

4. Comparatif

4.1. Le choix des critères

Afin d'effectuer une comparaison entre les différentes technologies issues de l'état de l'art, nous avons fixé des critères de comparaison. Ceux-ci prennent en compte les attentes du CDS. Nous allons maintenant présenter l'ensemble de ces points :

- Compatibilité : la compatibilité fait partie des critères les plus importants. En effet, pour qu'une application Web puisse devenir un service du CDS, il faut qu'elle soit compatible avec les navigateurs les plus populaires ainsi que la majorité des systèmes d'exploitation courants. De même, les outils de développement doivent être multi-plateformes, ou tout du moins fonctionner sous un environnement libre Linux.

- Gratuité : il est important que la technologie soit gratuite. Le CDS veut éviter les technologies dont il faut s'acquitter d'une licence coûteuse. Dans le même état d'esprit, il faut qu'il existe des outils de développement gratuits.

- Stabilité : le CDS met à disposition des services Web très utilisés, il faut donc que la technologie choisie permette de réaliser des applications stables.

- Performances : la rapidité d'exécution d'une application est un point primordial dans le choix d'une technologie.

- Maturité : la maturité d'une technologie traduit son sérieux et son implantation sur le marché, il s'agit donc d'un critère non négligeable.

- Maintenance et capacité d'évolution : dernier critère important pour le CDS, la capacité d'évolution d'une technologie. Une technologie soutenue par une communauté active aura plus de poids qu'une technologie peu suivie. De même, une technologie qui permet une maintenance aisée sera plus facilement adoptée.

- Le plus : il s'agit d'un critère supplémentaire, pas forcément très important, mais qui permettrait le cas échéant de départager plusieurs technologies.

4.2. Le tableau comparatif

		Compatibilité des systèmes		Compatibilité des navigateurs	Licence et gratuité		Performance & stabilité	Maturité & capacité d'évolution	Le plus
		Développement	Utilisateur		Développement	Utilisateur			
Silverlight	1.0	Systèmes les plus courant *	Windows Mac OS X	Internet Explorer 6/7 Firefox 2 Safari	aucun outil particulier (il faut disposer du fichier Silverlight.js qui est gratuit)	plugin gratuit sous licence commercial «Go Live» (Microsoft)	performance correcte et sorties régulières de patchs correctifs	version qui a été développée puis reprise dans la version 2.0	développement facile à mettre en place
	2.0	Windows	Windows Mac OS X	Internet Explorer 6/7 Firefox 2 Safari	uniquement une solution payante			toujours en version beta, communauté active	puissance du langage C#
Openlaszlo		Systèmes les plus courant	Systèmes les plus courant	ensemble des navigateurs les plus populaires **	outils gratuits	technologie Open Source sous licence CPL (IBM)	performances moyennes	technologie ancienne, très faible taux de pénétration sur le marché, communauté peu active	choix dans le format de sortie
HTML5		Systèmes les plus courant	Systèmes les plus courant	Firefox 2/3 Opera 9	outils gratuits	pas de licence	trop peu implémenté pour pouvoir se prononcer	volonté d'en faire un standard, encore à l'état de WorkingDraft	standard W3C
Flex		Systèmes les plus courant	Systèmes les plus courant	ensemble des navigateurs les plus populaires	SDK[4] gratuit (partie Open Source), outils payants & gratuits	plugin « freeware »	technologie performante et stable	longue expérience,fort taux de pénétration sur le marché (98%), standardisation du langage, communauté très active	forte compatibilité et très implanté sur le marché
JavaFX		Systèmes les plus courant	Systèmes les plus courant	ensemble des navigateurs les plus populaires	JDK[23] gratuit et Open Source, compilateur sous licence GPL, éditeurs gratuits	plugin (JVM) gratuit en licence GNU GPL	technologie très performante et stable	très grande expérience, communauté active, fort taux de pénétration sur le marché (84%)	expérience de Sun dans les technologies d'applets

* Windows, Unix/Linux, MacOS, Solaris

** Internet Explorer, Firefox, Netscape, Safari, Opera

	Compatibilité des systèmes		Compatibilité des navigateurs	Licence et gratuité		Performance & stabilité	Maturité & capacité d'évolution	Le plus
	Développement	Utilisateur		Développement	Utilisateur			
Volta	Windows	dépend du format de sortie choisi	dépend du format de sortie choisi	solution payante (version expérimentale gratuite)	encore en cours de développement, aucune licence évoquée	projet expérimental très peu avancé à l'heure actuelle		utilisation du C# et possibilité de convertir vers n'importe quel langage Web
XUL	Systèmes les plus courant *	Systèmes les plus courant	Firefox SeaMonkey Netscape	aucun outil particulier, framework sous licence MPL (Mozilla)	pas de licence	solution très stable (utilisée pour l'interface des composants du navigateur Firefox)	Peu d'évolution possible tant que cette technologie restera centrée sur Firefox	Interface très riche et proche des applications de bureau
GWT	Systèmes les plus courant	Systèmes les plus courant	ensemble des navigateurs les plus populaires **	aucun outil particulier, framework sous licence Apache 2.0	pas de licence	performant	projet soutenu par Google	écrire du javascript à partir du langage Java
EclipseRAP	Systèmes les plus courant	Systèmes les plus courant	ensemble des navigateurs les plus populaires	outils gratuits et framework sous licence EPL (Eclipse)	pas de licence	performances moyennes, produit stable	projet jeune et peu développé	utilisation de l'ensemble des bibliothèques Java
Wazaabi	Systèmes les plus courant	Systèmes les plus courant	ensemble des navigateurs les plus populaires	outils gratuits et framework sous licence EPL (Eclipse)	plugin (JVM) gratuit en licence GNU GPL	technologie très performante et stable	projet jeune et peu soutenu	utilisation des composants graphiques de EclipseRCP
HaXe	Systèmes les plus courant	dépend du format de sortie choisi	dépend du format de sortie choisi	outils gratuits et framework sous licence GPL	pas de licence	problème de stabilité	technologie jeune et richesse des composants limitée	une syntaxe simplifiée
Curl	Windows Linux (RedHat 9 et suSE 9)	Windows Linux (RedHat 9 et suSE 9)	Internet Explorer Netscape Firefox Safari	outils payants et framework sous licence commerciale	plugin (Curl RTE) gratuit sous licence	performances moyennes, produit stable	produit mature (version 6), mais très peu soutenue	pionnier sur le marché des RIA

* Windows, Unix/Linux, MacOS, Solaris

** Internet Explorer, Firefox, Netscape, Safari, Opera

5. Solutions retenues

Dans un premier temps, nous avons décidé d'effectuer une expérimentation de HTML5 par pure curiosité. La technologie n'étant pas encore au point, il n'est pas envisageable de réaliser un prototype utilisable, mais le but ici est d'entrevoir ce que pourra nous fournir cette nouvelle version du langage lorsqu'elle sera disponible.

Ensuite, nous avons décidé d'utiliser la technologie qui tire son épingle du jeu face à ses concurrentes, à savoir Flex, et plus précisément Flex dans sa version 3. En effet, il s'agit de la technologie RIA qui répond à tous les critères de choix : c'est une technologie disponible sur les principaux navigateurs et la majorité des systèmes d'exploitation avec son plugin « Flash Player », stable, mature et performant, installé sur plus de 98% du parc informatique mondial des ordinateurs connectés à Internet (source : site officiel d'Adobe). La version 3 apporte de nombreux composants encore plus performants. Enfin, l'ouverture partielle de cette technologie en licence OpenSource et l'existence d'outils de développement gratuits en font une technologie de choix.

Nous avons alors envisagé plusieurs projets de prototypage en Flex : dans un premier temps nous nous sommes penchés sur la réalisation d'un prototype de carte cliquable, puis nous nous sommes ensuite attachés à développer une application permettant de créer un dictionnaire de nomenclature d'objets célestes.

Nous allons donc maintenant, pour chaque prototype, expliquer son rôle et les différents moyens que nous avons mis en oeuvre pour y arriver.

IV. Les prototypes réalisés

1. Les WebForms 2.0

1.1. Conception

Comme nous l'avons vu précédemment dans ce rapport, HTML5 est encore très partiellement implémenté et seulement dans certains navigateurs. En fait, seules quelques fonctionnalités sont aujourd'hui exploitables, dont les canvas et les WebForms 2.0.

Les canvas sont des objets qui permettent d'effectuer du dessin vectoriel dessus. Mais l'absence d'API[20] permettant de prendre en main le canvas oblige pour le moment à utiliser le langage Javascript, ce qui n'est pas le but de HTML5.

Les WebForms quant à elles, sont un moyen de développer des formulaires avancés facilement et sans écrire de code Javascript. Cette technologie est développée depuis 2003 par le WATHWG[19] (groupe formé de Opera, Apple, et Mozilla) et a été incorporée à HTML5. Aujourd'hui, seul le navigateur Opera permet de comprendre les balises de la spécification WebForms 2.0.

Nous avons donc réalisé une page HTML (Figure 2.1), sous forme de tableau de trois colonnes. Dans la première on peut voir le code utilisé, dans la seconde le résultat graphique, puis dans la troisième les explications sur la fonctionnalité. Nous nous sommes basés sur les spécifications énoncées sur le site du W3C[14] qui référence toutes les fonctionnalités existantes, puis nous avons réalisé une implémentation de chacune d'elles.

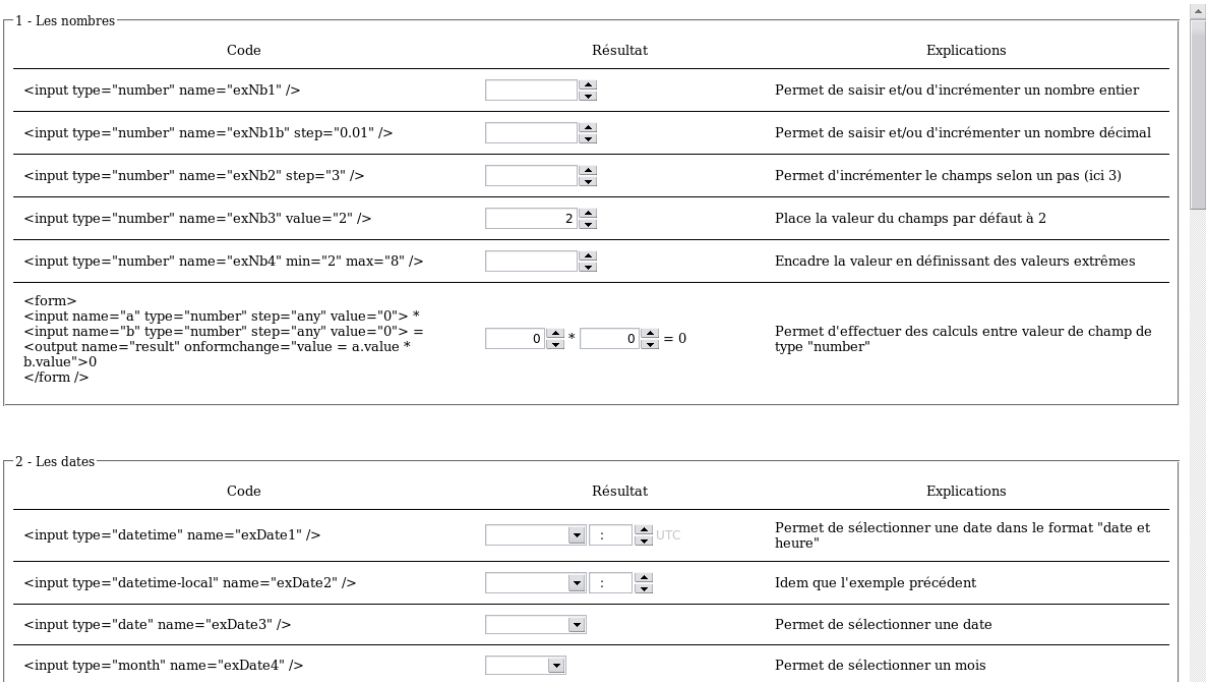


Figure 2.1 – Extrait de la page Web que nous avons réalisée.

1.2. Réalisation

1.2.1 Les nombres

Les WebForms 2.0 permettent de manipuler les nombres, ce qui était impossible jusque là sans utiliser un langage de programmation supplémentaire. Il est désormais possible d'effectuer des calculs au niveau du navigateur, avec des échanges client/serveur réduits au maximum. L'apparition du type « number » permet de modifier un nombre de manière incrémentale. Le seul type existant est le type décimal, mais il est possible d'utiliser des nombres réels en spécifiant comme pas d'incrémentation un nombre à virgule.

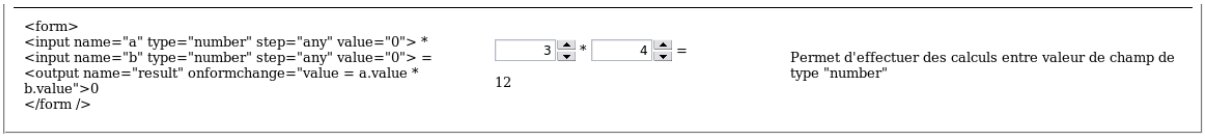


Figure 2.2 – Exemple de calcul: multiplication entre deux nombres entiers

1.2.2 Les dates

Le type « date » fait également son apparition. Il permet, comme son nom l'indique, de gérer des dates. La saisie d'une date se fait alors par un calendrier intégré. Il est possible de personnaliser ce calendrier, en permettant une sélection journalière, hebdomadaire, ou bien mensuelle, ou encore en empêchant la sélection de certaines dates.

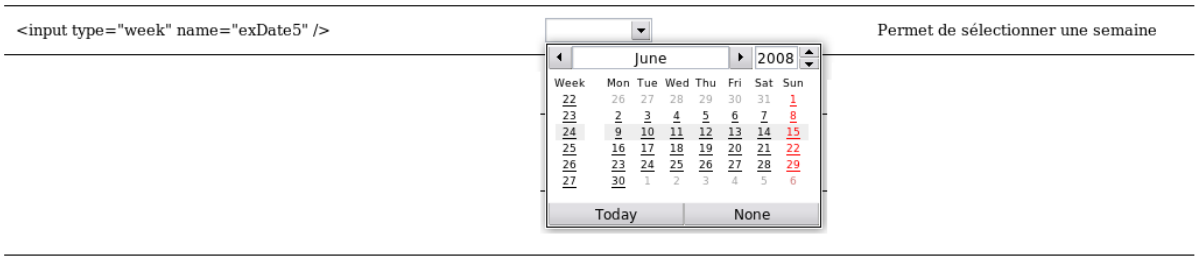


Figure 2.3 – Exemple de calendrier permettant de sélectionner une semaine entière

1.2.3 Les expressions régulières

Une des nouveautés est la possibilité de faire des vérifications grâce à des expressions régulières standardisées (ECMAScript[26]) de manière très simple. Il suffit d'ajouter un attribut « pattern » et de spécifier une expression régulière. À la validation du formulaire, une vérification sera effectuée afin d'analyser le contenu du champ comportant le type « pattern ». Si le format ne correspond pas, le formulaire n'est pas envoyé et un message d'erreur informe l'utilisateur de la saisie erronée.

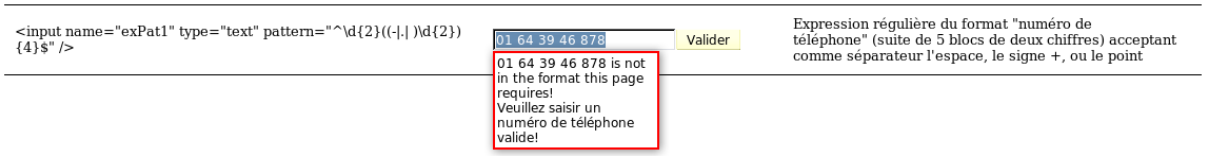


Figure 2.4 – Exemple d'utilisation des expressions régulières: vérifie que la saisie correspond bien au format du numéro de téléphone français

1.2.4 Les listes de données et XML

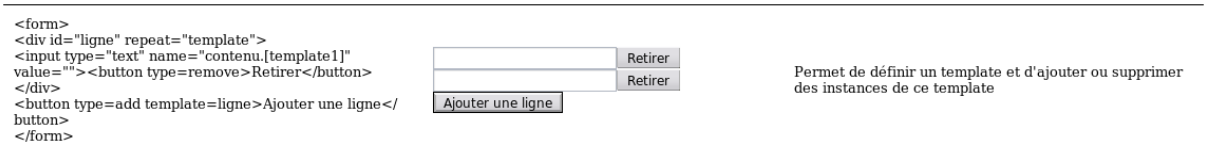
Les types « dataList » (ou liste de données) permettent de récupérer des données de manière dynamique et asynchrone, c'est-à-dire sans aucun rechargement de la page. Il est alors possible d'afficher des informations directement en provenance d'un fichier XML, seulement si celui-ci respecte un format particulier (à l'heure actuelle, ce format n'a pas encore été précisé).

Figure 2.5 – Exemple d'utilisation des liste de données: récupération de données dans une liste à partir d'un fichier XML

1.2.5 Les « templates » et les répétitions

Les templates permettent de définir des « schémas », c'est-à-dire des composants graphiques avancés construits à partir de plusieurs composants de base. L'intérêt des templates est de les utiliser en combinaison avec la fonctionnalité de répétition, qui permet d'ajouter dynamiquement une ou plusieurs fois un même template. On voit vite l'intérêt d'un tel mécanisme, notamment pour les sites de vente en ligne : il suffirait de cliquer sur un bouton pour ajouter un nouveau champ au formulaire d'achat.

Figure 2.6 – Exemple d'utilisation des templates avec répétition: affichage d'un template avec un



bouton permettant d'ajouter des instances de ce template

1.2.6 Les vérifications

Plusieurs types de vérifications sont implémentés en natif avec les WebForms 2.0, et en particulier les vérifications d'adresse e-mail ou encore de numéro de téléphone. Il suffit alors de préciser que le champ est d'un certain type prédéfini, et la vérification s'effectuera automatiquement au moment de la validation du formulaire. Même si ces vérifications sont relativement basiques à l'heure actuelle, elles vont sans aucun doute faciliter la vie des développeurs qui n'auront plus à écrire de nombreuses lignes de code dans le but de vérifier que la saisie est dans le bon format.

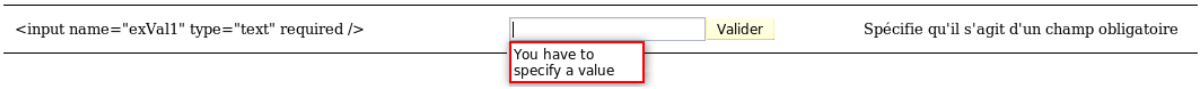


Figure 2.7 – Exemple d'utilisation des validations: un message d'erreur est affiché lors de la validation d'un champ obligatoire si celui-ci est laissé vide

1.3. Conclusion

Bien que HTML5 et à fortiori les WebForms 2.0 ne soient qu'en cours de développement, cette première expérimentation laisse envisager le potentiel de cette nouvelle version du langage HTML, notamment du point de vue de la facilité de mise en oeuvre et surtout du gain de temps non négligeable au niveau du développement.

2. La carte cliquable

2.1. Pré-requis

Le développement en Flex a nécessité une initiation à deux nouveaux langages de programmation que sont Actionscript 3 et MXML[27] ainsi que la prise en main des différents outils de développement.

2.1.1 Flex Builder 3

Travaillant sous un environnement Linux, nous avons pu profiter de la version beta de l'outil officiel de développement pour Flex : Flex Builder 3 d'Adobe. Cette version permet une utilisation gratuite de l'environnement de développement pendant une période de 120 jours. Mais d'autres outils gratuits peuvent également être utiliser au lieu de Flex Builder tels que Flashdevelop, SpketIDE, ou encore MXML Editor 1.0. La prise en main a été très rapide, puisque Flex Builder n'est en réalité qu'une version modifiée de l'IDE[28] OpenSource Eclipse. Ayant déjà pour habitude de travailler avec cet outil, je n'ai eu aucun mal à m'adapter au développement de projet Flex. La réelle nouveauté pour moi a été l'utilisation des deux langages permettant le développement des applications Internet riches en Flex.

2.1.2 Actionscript 3 & MXML

Comme je l'ai déjà expliqué dans la partie consacrée à l'état de l'art de ce rapport, Flex utilise une grammaire XML[5], le MXML, développé par Adobe pour définir l'interface graphique de l'application, et le langage de script Actionscript pour motoriser l'application. Ne connaissant aucun de ces deux langages, j'ai dû en apprendre les bases.

Ces deux langages peuvent être utilisés séparément, dans des fichiers différents. Dans ce cas, le fichier contenant le code MXML portera l'extension .mxml et le fichier contenant le code Actionscript portera quant à lui l'extension .as. Chacun de ces deux types de fichiers est ensuite traduit en classe Actionscript avant d'être compilé dans le format binaire exécutable SWF[10]. Il faut savoir qu'il est également possible d'utiliser de l'Actionscript au sein d'un document MXML (étant donné qu'il sera traduit en classe Actionscript), dans ce cas le code est dit « embarqué ».

La syntaxe du MXML est très simple et rapide à prendre en main, et ne m'a posé aucun problème lors de mon apprentissage. Ce langage permet de créer une interface graphique très rapidement en utilisant les composants existants et présents dans le kit de développement de Flex, dont voici un exemple relativement simple :

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application
  xmlns:mx="http://www.adobe.com/2006/mxml"
  layout="horizontal"
  width="500"
  height="500"
  >
  <mx:Panel
    id="monPanel"
    x="0"
    y="0"
    width="450"
    height="450"
    verticalAlign="middle"
    horizontalAlign="center"
    title="Le titre de mon panel"
    >
    <mx:Button
      x="0"
     y="0"
     id="monBouton"
     label="Valider"
    />
    <mx:Canvas
     id="monCanvas"
      x="50"
     y="50"
     width="350"
     height="350"
     backgroundColor="black"
    />
  </mx:Panel>
</mx:Application>
```

Cet exemple de code source MXML permet dans un premier temps de créer l'application générale grâce à la balise `</mx:Application>` en lui spécifiant une taille et un mode de disposition des composants (ici de manière horizontale). Cette application (figure 7.1) se compose alors d'un panel ayant pour titre « le titre de mon panel » comprenant lui même deux éléments : un bouton nommé « Valider » et un canvas dont le fond est de couleur noire.

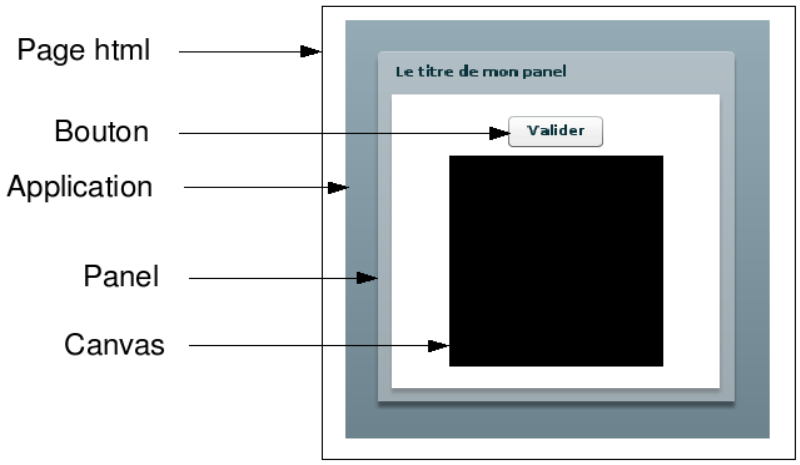


Figure 3.1 – Rendu graphique du code source MXML

En ce qui concerne l'Actionscript, la version utilisée avec Flex 3 est l'Actionscript 3. C'est un langage de script développé par Adobe et qui tente, notamment dans sa version 3, de se rapprocher un peu plus du standard ECMAScript du W3C. Syntaxiquement, l'Actionscript est relativement semblable au langage Java (programmation orientée objet, notation pointée, utilisation des mêmes mot-clefs, ...) ainsi qu'au langage Javascript (particulièrement dans les déclarations de types). Mais la réelle nouveauté avec Actionscript réside dans son mode de fonctionnement en partie asynchrone.

En effet, ce langage est principalement basé sur les événements, c'est-à-dire que pour une action particulière (un clic sur un bouton par exemple), un événement est propagé. Il faut ensuite écrire la partie du code qui permettra de capter l'événement et d'effectuer l'action associée. Il faut savoir que cette fonctionnalité n'est pas bloquante, on est donc capable d'exécuter du code entre le moment où l'événement est diffusé et le moment où il est capté. Mais Actionscript est également capable d'effectuer des opérations de manière séquentielle, les unes après les autres.

Pour exemple, voici le code source Actionscript permettant de capter l'événement lors du clic de la souris sur un bouton et qui dessine 10 points sur un canvas dont les coordonnées et les couleurs sont calculées de manière aléatoire. Pour ce faire, reprenons l'exemple précédent (code MXML) en modifiant le code du bouton comme ceci :

```
<mx:Button
  x="0"
  y="0"
  id="monBouton"
  label="Valider"
  click="drawPoints(event)"
/>
```

Ainsi, lorsque l'utilisateur clique sur le bouton « valider », la fonction `drawPoints` est appelée. Cette fonction est une fonction Actionscript dont voici le code (fichier complet est en annexe A1) :

```
import mx.core.UIComponent;

private function drawPoints(event:MouseEvent) : void
{
  var uic : UIComponent = new UIComponent();
  uic.width = monCanvas.width;
  uic.height = monCanvas.height;

  for(var i:int=0; i<10; i++)
  {
    uic.graphics.beginFill(Math.random() * 0xffffff);
    uic.graphics.drawCircle(
                        Math.random() * 150,
                        Math.random() * 150,
                        4);
    uic.graphics.endFill();
  }

  monCanvas.addChild(uic);
}
```

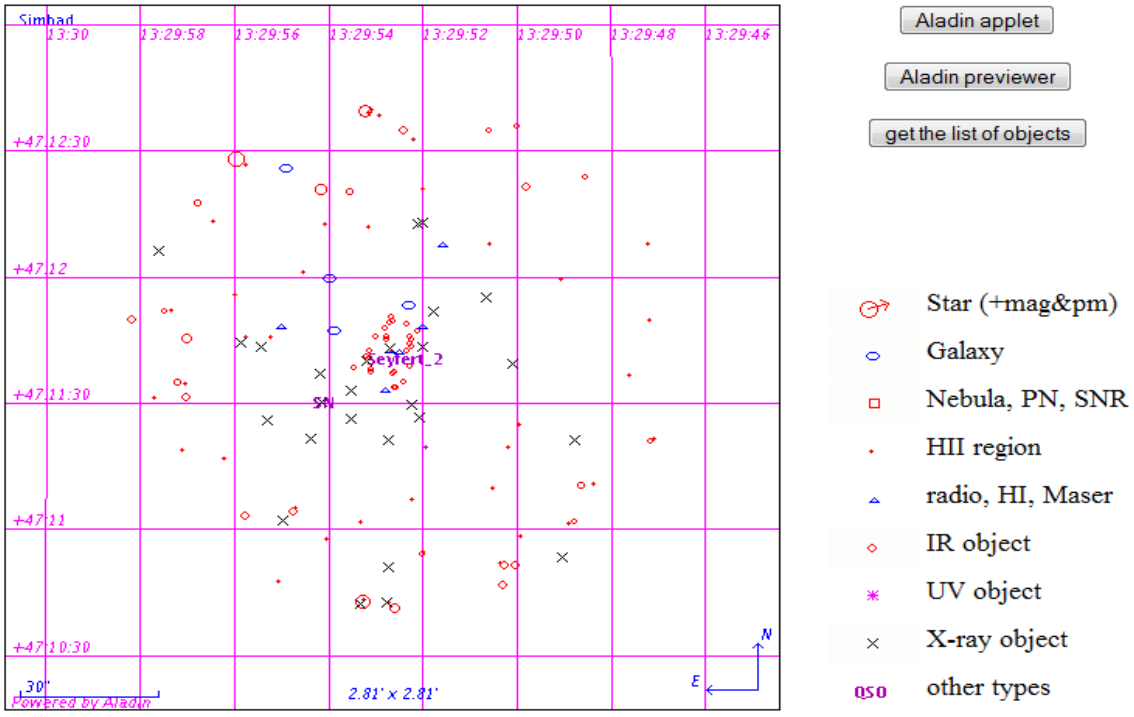
Cette fonction crée un objet de type UIComponent qui prend le même taille que l'objet canvas. Ensuite, on dessine un cercle sur le UIComponent dont la couleur ainsi que les coordonnées sont définies aléatoirement grâce à la fonction mathématique « random ». Cette opération est effectuée 10 fois avant d'ajouter le UIComponent sur canvas (Figure 3.2).

Figure 3.2 – Rendu graphique de l'application après appel à la fonction Actionscript

2.2. Conception

2.2.1 Présentation de l'existant

Le service de la carte cliquable existe déjà (Figure 4.1). Il s'agit d'un service permettant d'afficher graphiquement des données astronomiques relatives à une cible que l'on indique au préalable. Une image est générée et renvoyée par un serveur. Cette image représente une grille traduisant les coordonnées sexagésimales ainsi que des symboles (rond, carré, losange, etc.) qui s'y superposent. Ces symboles représentent des types d'objets astronomiques, dont chaque instance caractérise un objet particulier identifié par des cordonnées (x,y) sur l'image. On peut assimiler cet objet à un point. Lorsque la souris survole cette image, il est possible de cliquer, dans ce cas une nouvelle page se charge, affichant des informations complémentaires sur le point sélectionné. Si l'utilisateur a cliqué quelque part sur la carte mais pas sur un point, un message est alors affiché sur une nouvelle page, l'informant qu'il n'y a aucune donnée associée à cette position.



2.2.2 Objectifs & contraintes

L'objectif majeur de ce prototype est de permettre une première expérimentation de la technologie Flex afin d'en mesurer sa pertinence. Nous allons donc aborder les fonctionnalités générales de cette RIA.

Le second objectif est de réaliser un prototype viable qui pourrait remplacer le service existant s'il s'avère plus intéressant. Le but est de rendre le service plus convivial mais également de lui apporter un certain nombre de nouvelles fonctionnalités :

- la possibilité d'afficher une image au lieu d'une grille
- représenter les données astronomiques par des points cliquables
- pouvoir zoomer sur l'image de fond sans affecter la taille et la position des points du premier plan sur l'image
- pouvoir déplacer l'image (notion de glisser/déposer)
- permettre l'affichage d'informations complémentaires au survol d'un point
- permettre de dessiner les points d'après leur type
- pouvoir afficher dans un tableau sur la même page les données supplémentaires correspondantes aux points affichés, avec la possibilité de mettre en évidence un point sur la carte si ses données dans le tableau ont été sélectionnées, ou bien mettre en évidence les données associés lorsqu'un point est cliqué sur la carte
- possibilité de copier dans le presse-papier les données du tableau

Le prototype doit être performant dans le sens où l'affichage doit être fluide et réactif. Le poids final de l'application est également une contrainte important à prendre en compte et le maximum doit être fait afin de limiter cette taille. Enfin, par souci de réutilisation, le développement sera fait de manière modulaire, afin de permettre la réutilisation d'un certain nombre de fonctionnalités pour d'autres services du CDS. Pour ce faire, nous avons développé cette application en plusieurs modules auxquels nous avons appliqué le modèle de programmation MVC[29] en respectant la séparation entre la logique applicative et la couche graphique de l'application.

2.2.3 Fonctionnement général de l'application développée

L'application s'exécute dans un navigateur et prend en paramètre dans son URL le nom de la cible. Si la cible n'existe pas, un message d'erreur est affiché et permet à l'utilisateur d'effectuer une nouvelle saisie. Si la cible est correcte, l'application récupère alors un fichier XML contenant 2 URLs : l'une pointant vers l'image à afficher, l'autre vers un fichier XML contenant les données astronomiques à afficher sur l'image. Ce fichier XML est au format VOTable (pour Virtual Observatory Table, un exemple de ce format peut être retrouvé en annexe A2), format qui a été défini pour l'échange de données astronomiques dans le cadre de l'Observatoire Virtuel.

L'application se charge ensuite d'afficher l'image qu'elle a téléchargée, puis de parcourir le fichier XML afin de créer tous les points correspondants aux données astronomiques associées. Les points créés sont ensuite affichés par dessus l'image (Figure 4.2).

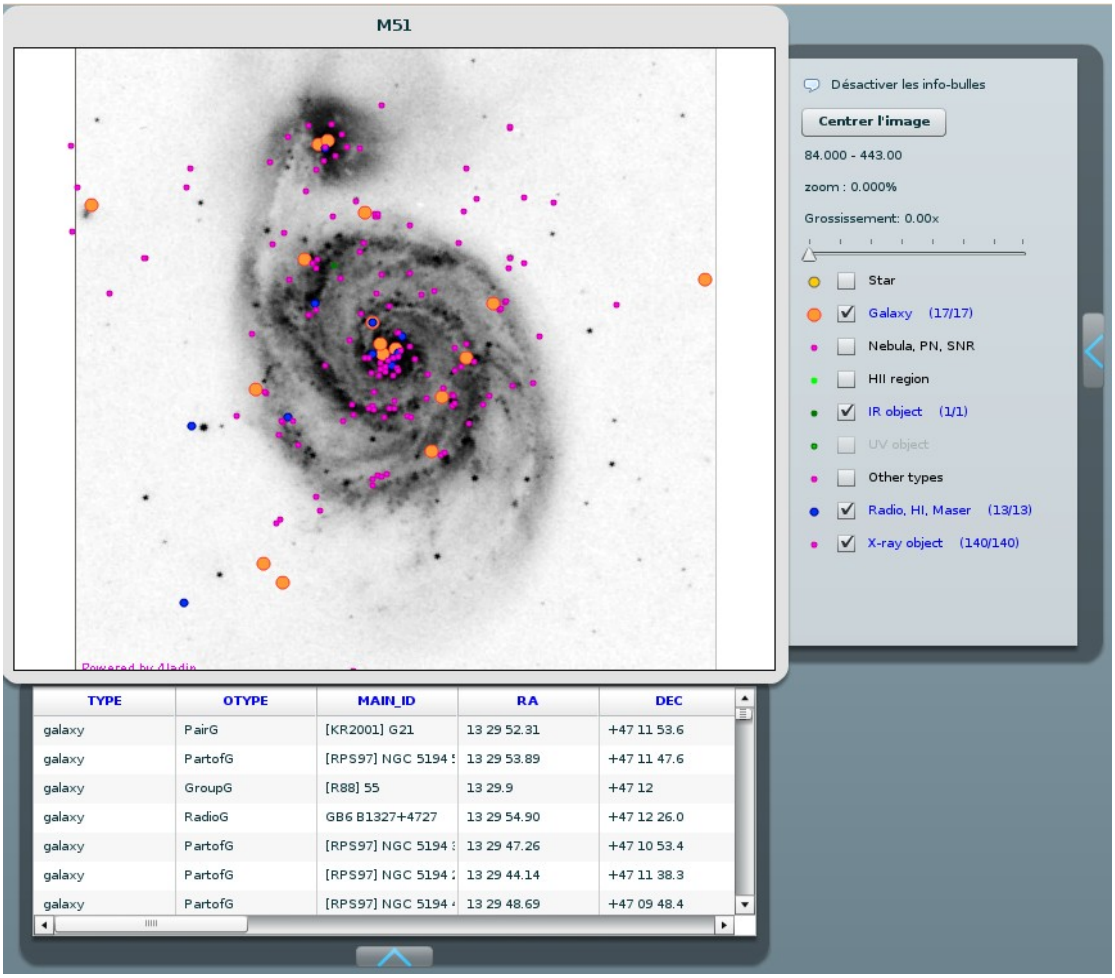


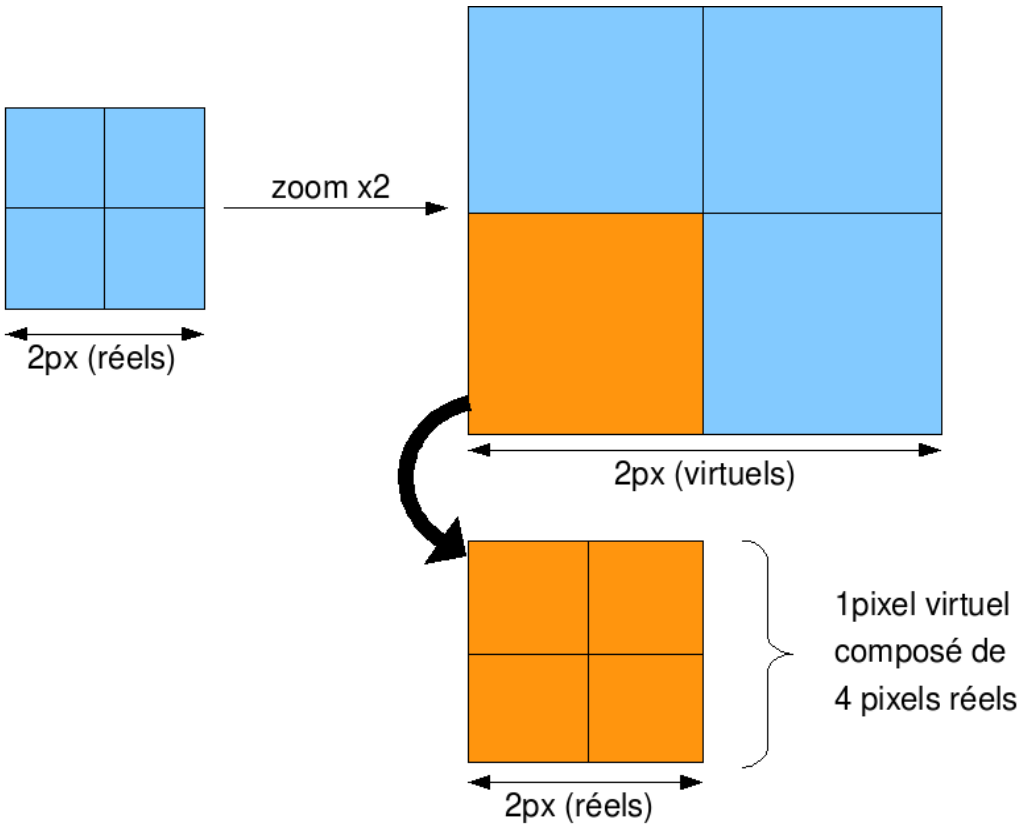
Figure 4.2 – Exemple du service développé en Flex

2.3. Réalisation

2.3.1 Le zoom

Pour permettre le zoom sur une image, nous utilisons une fonction mathématique fournie dans le framework Flex. Cette fonction permet donc d'agrandir l'image d'après un pas, mais aussi de la centrer en fonction des coordonnées qui lui sont passées. Le zoom s'effectuant avec la molette de la souris, ces coordonnées correspondent donc à celles du pointeur de la souris. Cette méthode utilisée pour accomplir le zoom est assez particulière. En effet, si visuellement la taille de l'image semble avoir augmenté, dans la réalité son nombre de pixels n'a pas bougé.

Le zoom sur une image n'augmente en réalité sa taille que virtuellement. C'est-à-dire que pour une image de 2px sur 2px et pour un pas de 2, nous devrions obtenir une nouvelle image, plus grande, de 4px par 4px. Or pour le lecteur Flash, après le zoom, cette image a toujours une taille de 2px par 2px. Ceci s'explique par le fait que la fonction de zoom crée des pixels virtuels, plus gros que les pixels originaux, et qui contiennent eux même un certain nombre de pixels réels (Figure 5.1).



Les composants de l'application voient donc l'image comme ayant une taille fixe puisque le lecteur ne prend en compte que la taille virtuelle, alors que graphiquement, on voit bien que l'image a doublé de taille. Il est facile de calculer la taille réelle de l'image, mais il est impossible de forcer le lecteur à prendre en compte la taille réelle et non la taille virtuelle. Ceci pose donc plusieurs problèmes.

En effet, l'image est dessinée sur un composant appelé canvas. Ce canvas permet de lui appliquer d'autres composants graphiques, en les empilant les uns par dessus les autres. Nous utilisons donc un canvas comme conteneur de l'image afin de pouvoir superposer par la suite les points correspondant aux données astronomiques. Le canvas étant un objet de type « conteneur », il sait normalement gérer les tailles des objets qu'il contient (implémentation en natif). Ainsi, lorsqu'un composant atteint une taille qui dépasse celle du conteneur, le canvas fait en sorte de « cacher » les parties des composants qui débordent hors de celui-ci (Figure 5.2).

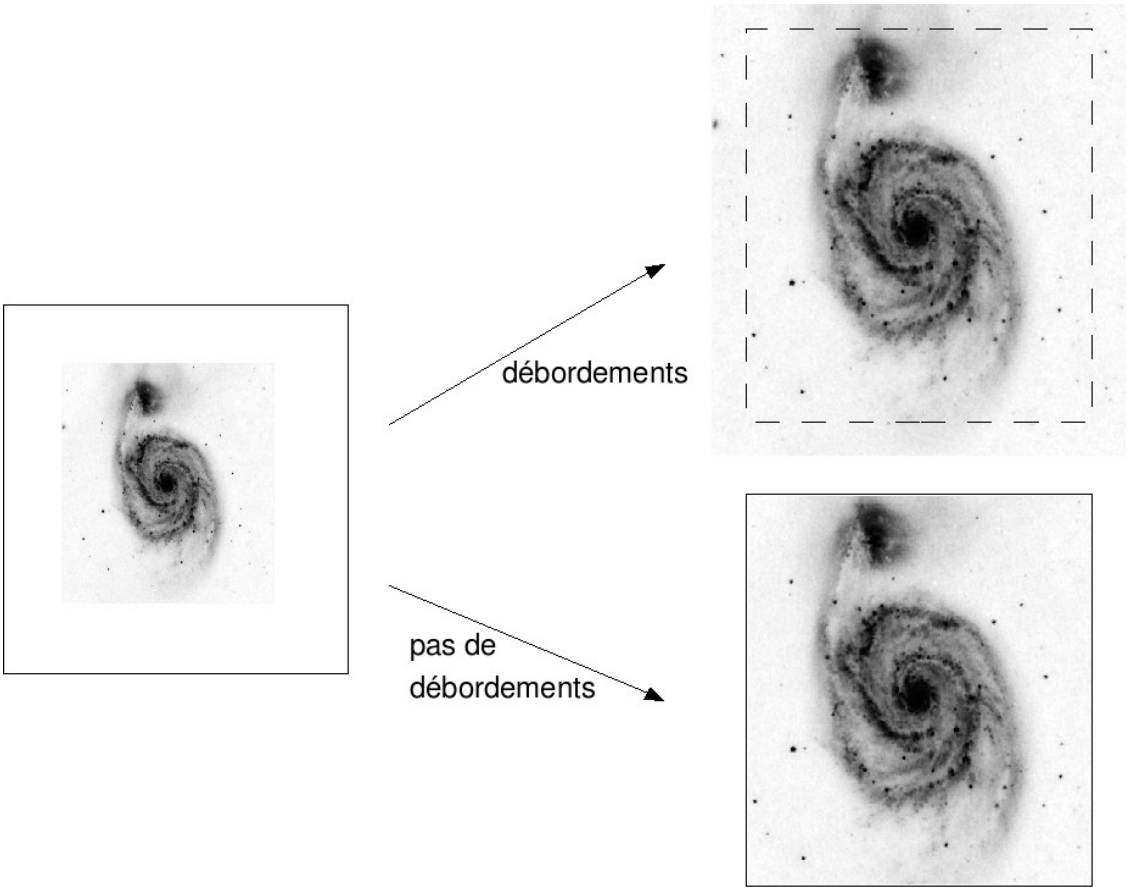


Figure 5.2 – Schéma explicatif du mécanisme de gestion du débordement

Comme nous venons de le voir, la taille réelle de l'image n'est vue en tant que telle par les autres composants de l'interface graphique. Le problème est donc que le canvas ne peut pas gérer les débordements de l'image puisque, de son point de vue, la taille de l'image n'a pas varié entre sa taille initiale et sa taille zoomée. La solution est donc de simuler ce comportement. Pour ce faire, la seule solution trouvée est de créer un autre composant, qui serait lié à l'image, et qui prendrait comme taille la taille réelle de l'image. Ce composant, un canvas, est donc placé entre le canvas principal et l'image. Ainsi, lorsque l'on zoome sur l'image, le canvas caché sous l'image prend les nouvelles tailles de l'image. Ainsi, le canvas principal peut gérer les débordements du canvas caché et donc indirectement ceux de l'image (Figure 5.3).

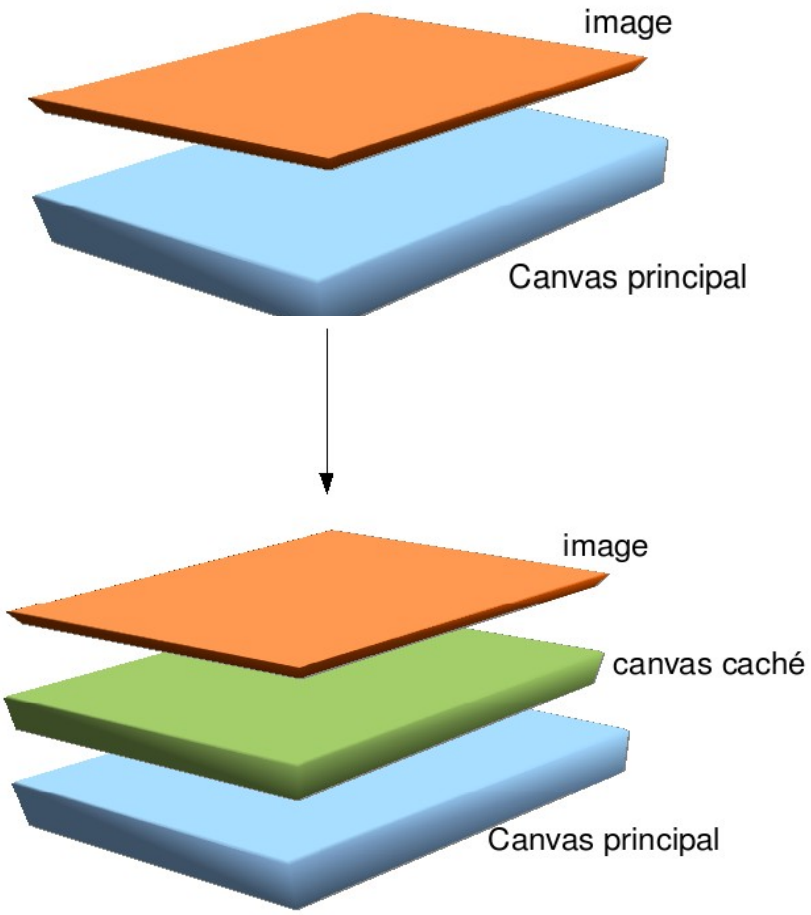


Figure 5.3 – Schéma explicatif du mécanisme de simulation des débordements

2.3.2 Le glisser/déposer

L'une des fonctionnalités de base de cette application est le glisser/déposer. Ce mécanisme consiste à sélectionner un objet avec la souris puis, tout en maintenant le clic gauche enfoncé, à déplacer le pointeur de la souris. Ceci permet alors de bouger visuellement l'objet choisi. On retrouve largement ce comportement sur les systèmes d'exploitation courants utilisant une interface graphique, comme par exemple lorsqu'on glisse des documents dans la corbeille ou encore lorsqu'on change un fichier de répertoire. Dans notre application, le but est de pouvoir effectuer la même action sur l'image, lui permettant ainsi d'être déplacée sur toute la surface de son conteneur.

La solution semble relativement simple car Flex intègre cette notion dans certains de ses composants. En effet, pour un composant spécifique, ici notre image, il suffit d'activer un paramètre renseignant le lecteur que l'objet en question peut être déplacé sur son conteneur. Il est donc possible de bouger chaque point qui constitue l'image sur chaque point du conteneur. Il est alors possible de sortir tout ou partie de l'image, et même de la perdre si l'utilisateur bouge son image trop loin par rapport au conteneur (Figure 6.1).

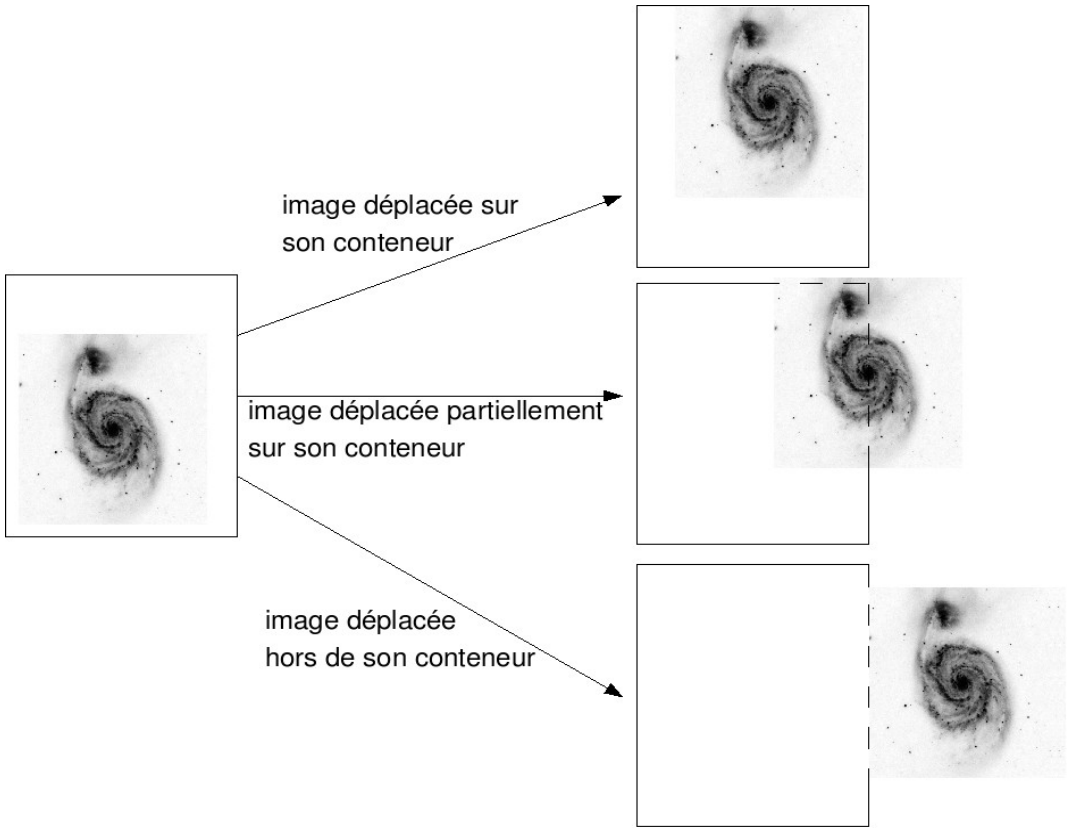


Figure 6.1 – Schéma explicatif du mécanisme de glisser/déposer avec perte de l'objet

Comme on peut le voir sur cette illustration, dans le cas où l'image est déplacée hors de son conteneur, l'extrémité gauche de l'image est placée au niveau de l'extrémité droite de son conteneur, dans ce cas il devient impossible pour l'utilisateur de « récupérer son image ». En effet, il n'est plus possible de cliquer sur l'image, car la partie encore sur le conteneur se confond avec le bord de ce dernier.

Pour empêcher toute perte de l'image ou de la sortir trop de son conteneur, la solution mise en place consiste à borner manuellement les zones de déplacements de l'image.

Dans un premier temps, nous avions pensé simplement restreindre la zone de glisser/déposer à la taille du conteneur. Pour se faire, il suffit de prendre les coordonnées de l'image et de vérifier :

- que la coordonnée x de l'image est supérieure à celle du conteneur
- que la coordonnée y de l'image est supérieure à celle du conteneur
- que la coordonnée x' (= x_{image} + largeur de l'image) est inférieure à celle du conteneur ($x_{conteneur}$ + largeur du conteneur)
- que la coordonnée y' (= y_{image} + longueur de l'image) est inférieure à celle du conteneur ($y_{conteneur}$ + longueur du conteneur)

On peut comme cela détecter dès qu'un bord de l'image entre en contact avec un bord du conteneur et désactiver la fonction de glisser/déposer. Ainsi, il est strictement impossible de sortir le moindre pixel de l'image de son conteneur. Mais cette solution entre en conflit avec une autre fonctionnalité de l'application qu'est le zoom sur l'image (se référer au 2.3.1 de ce rapport). L'image pouvant être zoomée, elle finira forcément par sortir de son conteneur. Dans ce cas il faut gérer le fait que la partie visible de l'image puisse être glissée de manière à permettre l'affichage des parties qui débordent, tout en gardant toujours la totalité du conteneur « rempli » par l'image (Figure 6.2).

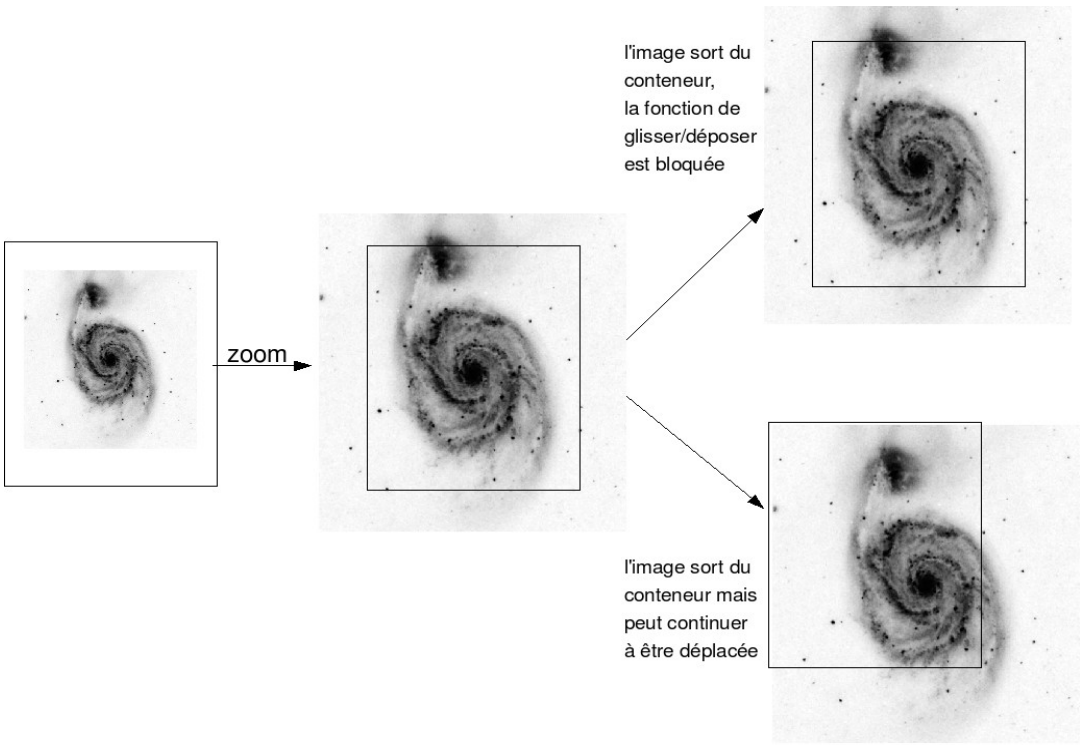


Figure 6.2 – Schéma illustrant une image qui déborde du conteneur, lequel est toujours « rempli » par l'image.

Pour y parvenir, nous avons opté pour la mise en place d'une zone de « dragabilité », c'est-à-dire une zone prédéfinie en fonction de la taille de l'image (qui varie à chaque action de zoom) et qui délimite le périmètre dans lequel l'image peut être déplacée. Le concept est le suivant : créer un cadre dont le point de coordonnées (x,y) de l'image est « prisonnier » (Figure 6.3).

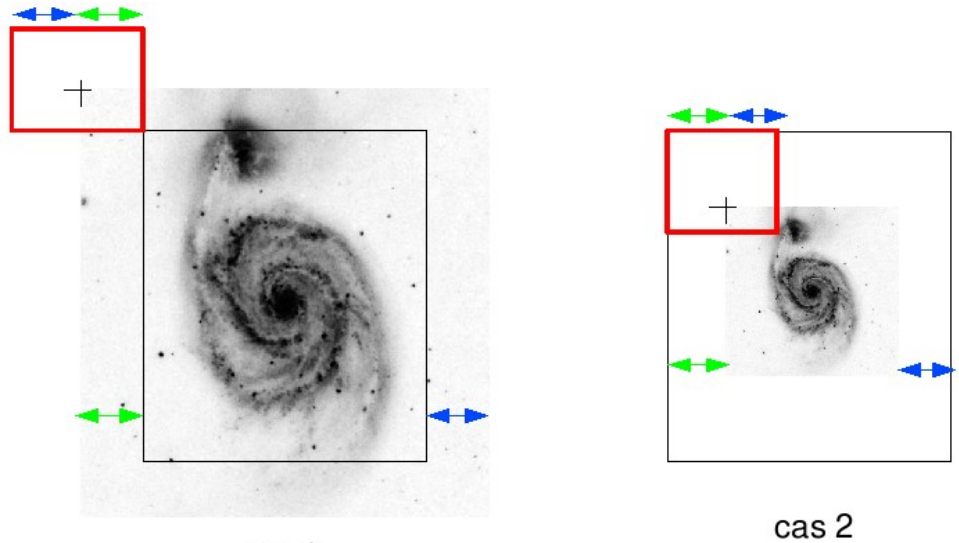


Figure 6.3 : Illustration du concept de dragabilité.

Comme nous pouvons le voir sur l'illustration ci-dessus, un cadre est créé (cadre rouge), dont la taille dépend des zones de débordement de l'image sur son conteneur (cas 1) ou bien des zones de débordement du conteneur sur l'image (cas 2). Dans les deux cas, le périmètre de déplacement du point (x,y) de l'image est restreint de manière à pouvoir afficher toute l'image sans jamais la sortir du conteneur. La zone représentée en rouge se calcule facilement puisque l'on connaît les tailles du conteneur et de l'image.

2.3.3 La fluidité des interactions et la rapidité d'exécution

Le Flash Player, qui exécute le code binaire SWF, effectue en permanence des calculs, permettant de générer le rendu graphique (l'affichage). Le fonctionnement de ce lecteur est alors basé sur une liste des actions à faire (la displayList). Chaque tâche est mise en file, puis lorsque tous les calculs sont effectués, l'affichage est déclenché. Ce fonctionnement permet au lecteur de consigner ce qui a déjà été affiché pour ne pas avoir à retracer la totalité de l'application lorsqu'un changement survient. Le problème majeur vient du fait que l'on ne peut pas agir sur cette liste ni forcer le rendu lorsqu'on le souhaite.

La carte cliquable doit être capable d'afficher un grand nombre de points (de plusieurs centaines à plusieurs milliers) tout en gardant sa fluidité et sa réactivité pour l'utilisateur. Étant donné le fonctionnement du Flash Player, il va sans dire que les performances déclinent rapidement avec un tel nombre d'objets à afficher. De plus, pendant que Flash génère son rendu, l'écran reste bloqué, c'est-à-dire que l'utilisateur ne peut plus interagir avec l'interface graphique de l'application. Ce phénomène peut porter à confusion et faire croire à l'utilisateur que l'application ne fonctionne pas normalement.

A titre d'exemple, afficher 3000 points prend approximativement une quinzaine de secondes. Il a donc fallut trouver une solution qui permette d'une part de laisser l'interface toujours réactive, et d'afficher les points beaucoup plus rapidement d'autre part. Plusieurs solutions ont été envisagées, et notamment l'utilisation d'objets graphiques plus légers que ceux utilisés initialement. Le problème est que les objets plus légers ne permettent pas d'interagir avec eux. Or le but est de pouvoir cliquer sur les points afin d'obtenir les informations complémentaires. Cette solution ferait donc perdre une partie des fonctionnalités souhaitées.

Nous avons également sérieusement envisagé la création d'un calque transparent qui contiendrait les points. Ce calque est alors créé pendant l'initialisation de l'application ce qui rend transparent le mécanisme de placement des points aux yeux de l'utilisateur. Outre le problème que les calques soulèvent (voir la partie 2.3.1), le temps d'attente avant rendu ne subit pas de modification significative. Nous nous sommes en fait rendu compte en mettant en place cette solution, que ce n'est pas réellement le fait de devoir créer plusieurs milliers d'instances d'objets lourds qui prend du temps, mais uniquement le fait de devoir ajouter les points graphiquement sur l'écran.

Nous avons finalement opté pour une solution qui contourne le problème. En effet, s'il est impossible de forcer le moteur de rendu de Flash, il est cependant possible de simuler la fin des calculs. Il s'agit de placer des « timers », c'est-à-dire des horloges qui se déclenchent à intervalles réguliers. Par exemple, sur une liste de calcul contenant 30 opérations, nous pouvons diviser cette liste en plaçant un timer qui, toutes les 40ms, exécutera une opération. Il suffit ensuite de répéter cette opération autant de fois qu'il y a de calculs à effectuer (ici 30). Nous avons donc un programme qui exécute un calcul puis qui attend 40ms avant de lancer le deuxième. Ceci est interprété par le lecteur Flash comme des listes séparées. Ainsi, au lieu d'attendre d'avoir effectué les 30 calculs et de générer le rendu, Flash va pouvoir générer le rendu à chaque fin de calcul, soit 30 rendus. Cette solution permet donc d'avoir un affichage progressif des points, un point étant affiché toutes les 40ms. En fait, cette solution n'est pas si bien que cela, au contraire même.

En effet, même si l'utilisateur peut patienter en voyant les points s'afficher progressivement et continuer à interagir avec les autres composants de l'application, le temps total d'affichage (pour un exemple de 3000 points) peut prendre jusqu'à 50 secondes. Ceci s'explique simplement : plus le nombre de points affichés augmente, plus le lecteur Flash met de temps à générer le prochain rendu. En fait, il n'est pas capable de seulement ajouter un point, mais il doit tout re-générer, donc à l'instant n il doit générer l'affichage du point n ainsi que les $n-1$ points précédents.

Ce résultat peut être amélioré en influençant sur la vitesse d'affichage du lecteur. Le player fonctionne sur un principe « d'affichage par seconde », c'est-à-dire qui effectue un rendu à intervalles réguliers, appelé FPS pour Frames Par Seconde. Initialement, le fps est réglé à 25, ce qui permet

d'afficher 25 images (et donc 25 rendu) à la seconde. Lorsque l'on utilise un timer, sa fréquence ne peut être inférieure à l'inverse du framerate (le nombre de fps). Ce qui explique que pour un framerate de 25, un timer ne peut se déclencher que toutes les 40ms au minimum. Heureusement, il est possible de personnaliser ce framerate. Ainsi, en le passant à 50 fps, nous pouvons afficher un point toutes les 20ms. En terme de performance, on pourrait s'attendre à ce que le temps d'attente pour l'affichage soit diminué de moitié, mais en réalité le gain n'est que de 10 à 30% selon la machine. En fait, l'utilisation du framerate peut être à double tranchants, car il dépend essentiellement des caractéristiques de la machine cliente. Ainsi, si la machine d'un utilisateur ne peut faire tourner l'application avec un framerate de 50, le flash player le détectera et placera dans ce cas le framerate au plus bas possible (20 fps), ce qui peut alors être catastrophique en ce qui concerne les temps d'affichage.

Nous avons trouvé une autre solution tout récemment et nous n'avons pas encore eu le temps de la mettre en place, mais d'après nos premiers tests, il pourrait s'agir d'une solution viable (affichage de 10000 points en 125ms). Il ne faudrait alors pas utiliser de composants graphiques mais plutôt dessiner directement les points sur l'image. Ainsi, le lecteur n'aurait qu'un seul rendu à faire : celui de l'image, ce qui est très léger. Cependant, on perd les fonctionnalités d'interactions avec les points. Il suffirait alors de simuler ces comportements en gardant en mémoire un tableau des coordonnées des points, puis en détectant les différents comportements utilisateur (clic, survol avec la souris, etc.) ainsi que leur emplacement (c'est-à-dire les coordonnées de la souris sur l'image). Il faudrait ensuite comparer ces coordonnées avec celles présentes dans le tableau et vérifier qu'il y a bien une action qui y est associée. Tout se ferait alors uniquement par les coordonnées (x,y) sans utiliser de composants Flex.

2.4. Conclusion

Dans l'état actuel des choses, la carte cliquable que nous avons développée en Flex ne satisfait pas aux exigences de performances énoncées dans les objectifs, à cause du temps d'attente trop long pour l'affichage des points. Néanmoins, le problème devrait être réglé avec l'apparition d'une nouvelle solution que nous mettrons en place à la suite du stage.

3. Le dictionnaire de nomenclature d'objets célestes

3.1. Conception

3.1.1 Présentation de l'existant

Le service de dictionnaire de nomenclature d'objets célestes est un service très utilisé par la communauté des astronomes, qui permet d'obtenir des informations avancées (synonymes, descriptions, etc.) sur des objets astronomiques. Actuellement, une première page HTML permet à un utilisateur, via un formulaire de saisie, d'effectuer une recherche dans le dictionnaire. Il existe divers critères que l'on peut spécifier afin d'affiner la recherche. Cette page donne également des informations sur ce qu'est le dictionnaire de nomenclature et comment l'utiliser (Figure 7.1).

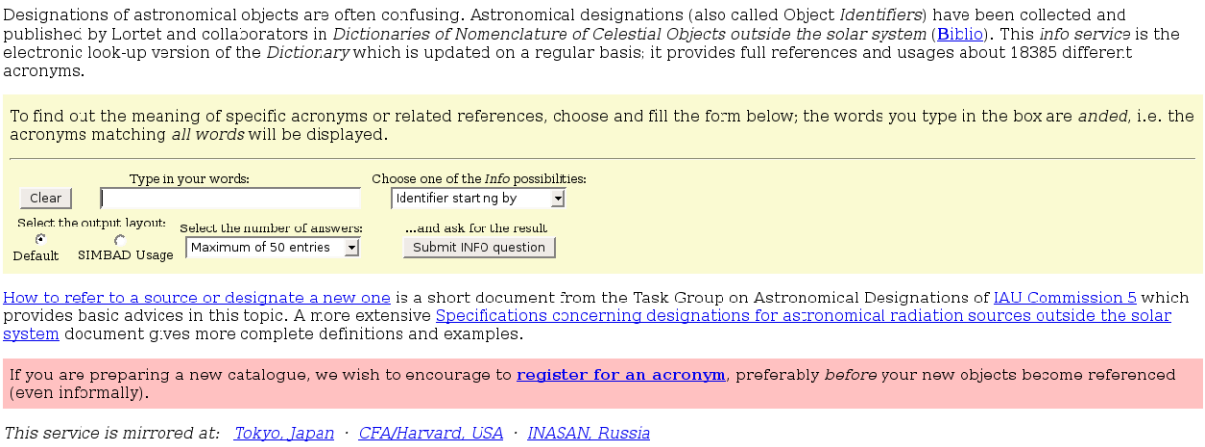


Figure 7.1 – Première page du service : le formulaire

Ensuite, une seconde page affiche les résultats sous forme tabulaire (Figure 7.2) puis, lorsque l'utilisateur clique sur un résultat, il obtient des informations complémentaires. Ces informations sont alors affichées sur une troisième page (Figure 7.3).

Acronym	Use	Format	Year	1st Author	Obj. Type
(A)	[GVC73]	{A} R.N	1973	GIOVANELLI R.+	Concentration
(AG)	[GB2001]	{AG} N	2001	GENOVA R.+	Interstelar Cloud
AGN	GN	HH.MM.m HH.MM.m.NN	-	NECKEL Th.+	Neb. Obj.
(ALFALFA)	[GHK2005]	{HI} JHHMMSS.s+DDMMSS	2005	GIOVANELLI R.+	HI
(ALFALFA)	[GHK2007]	1-NNN	2007	GIOVANELLI R.+	HI
(An)	[GG70]	{An} N	1970	GEORGELIN Y.P.+	HII
(AO)	Gat	NNNN	1973	GATEWOOD G.+	*
(ATCA)	[GGM98]	{ATCA} JHHMMSS+DDMMSS JHHMMSS+DDMMSS	1998	GAENSLER B.M.+	?
(ATCA)	[GSF98]	{ATCA} JHHMMSS+DDMMSS JHHMMSS+DDMMSS	1998	GAENSLER B.M.+	Neb
(B)	GC	NNNNN	1937	BOSS B.	*
(Be)	[GC2000]	{Be}NN	2000	GREBEL E.K.+	Be*
(BG)	[GWC99]	{BG} JHHMMSS+DDMMSS	1999	GARNETT D.R.+	Bok Globule
(BGC)	GC	NNNNN	1937	BOSS B.	*

Figure 7.2 – Seconde page du service : les résultats

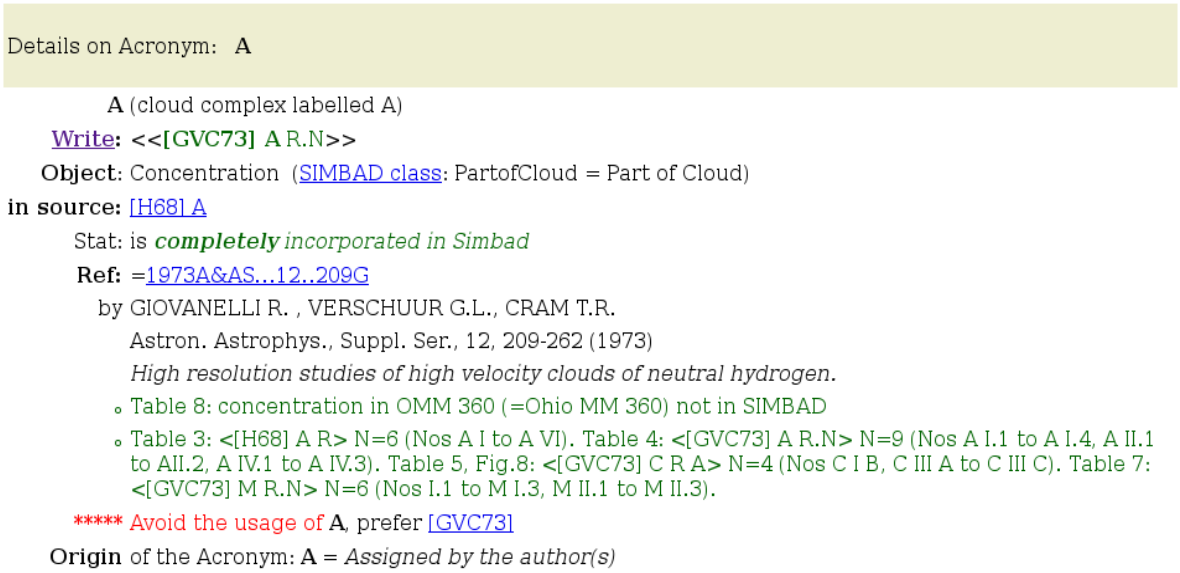


Figure 7.3 – Troisième page du service : des informations complémentaires

3.1.2 Objectifs & contraintes

L'objectif de ce prototypage est de réaliser une application plus conviviale que celle existante, et surtout moins lourde, en évitant le chargement de trois pages différentes.

Nous avons donc convenu que la totalité des informations devrait être affichée sur une seule page, sans faire intervenir de rechargement. L'affichage du formulaire de recherche ainsi que des résultats sur le même écran permet dans un même temps d'éviter les va-et-vient entre plusieurs pages dans le cas où l'utilisateur serait intéressé par faire des recherches sur des thèmes différents.

L'autre avantage de tout afficher sur une seule et même page est de permettre la comparaison visuelle entre plusieurs résultats d'une même recherche très rapidement. En effet, la version Flex de l'application doit permettre d'afficher les informations complémentaires sans pour autant perdre de vue les résultats de la recherche, pouvant ainsi comparer le contenu de plusieurs résultats.

L'application doit clairement permettre la gestion des données par l'utilisateur, c'est-à-dire leur tri (automatique par des fonctions de tri ascendant et descendant), leur déplacement (manuel par glisser/déposer), ou encore l'affinage de la liste des résultats (garder ou cacher des éléments sélectionnés et/ou filtrer grâce à un terme saisi). Toutes ces manipulations doivent être effectuées côté client, pour éviter au maximum le trafic réseau.

3.1.3 Fonctionnement général de l'application développée

L'application se compose de trois grands « blocs ». Chaque bloc est rétractable, ce qui permet de cacher les informations dont l'utilisateur n'a plus besoin (les informations sur le fonctionnement du dictionnaire par exemple). Nous retrouvons donc les informations diverses dans un premier temps, puis le formulaire de recherche. Lorsqu'une recherche est effectuée, le bloc d'informations se rétracte, laissant de la place pour le bloc des résultats (Figure 8.1). Lorsqu'un bloc est affiché ou rétracté, la taille des autres blocs varie en fonction de la place restante, ce qui évite ainsi d'avoir un ascenseur sur la page.

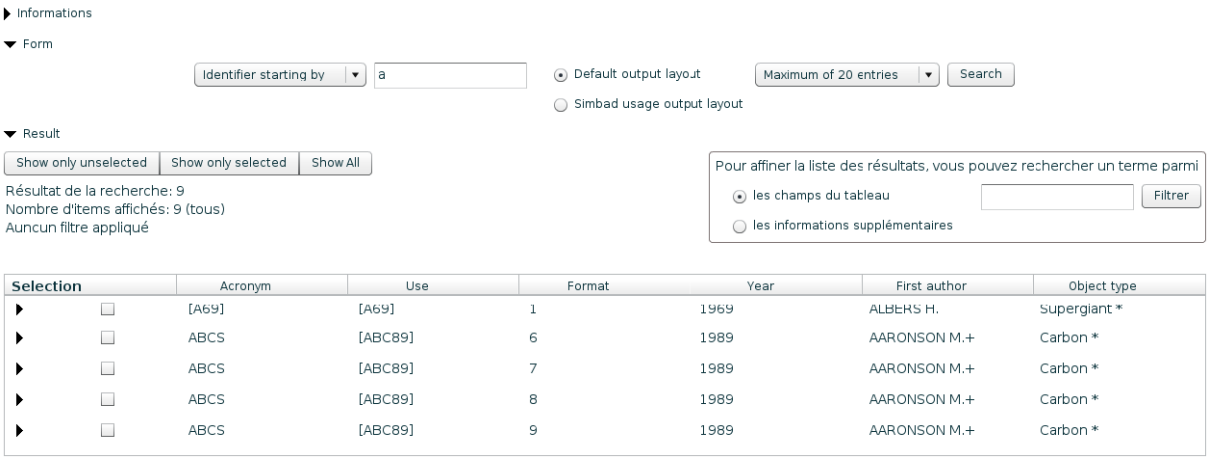


Figure 8.1 – Exemple d'utilisation avec affichage du bloc des résultats (bloc d'informations rétracté)

En ce qui concerne la fonction de recherche, nous avons décidé de modifier la fonction existante qui retournait du code HTML généré en fonction des résultats. Désormais, l'application s'occupe de communiquer, via l'url, la saisie de l'utilisateur et récupère un fichier XML contenant la liste des résultats trouvés avec, pour chaque résultat, les informations complémentaires qui sont associées. L'application parcourt ensuite le document XML et affiche graphiquement les résultats.

Ces résultats sont affichés sous la forme d'une liste, dont chaque ligne peut-être développée (Figure 8.2). Il s'agit d'un comportement courant et familier pour l'utilisateur, on le retrouve généralement sur les outils d'exploration de fichiers présents sur les systèmes d'exploitation, où une petit sigle « + » ou « - » permet de développer ou au contraire de rétracter le contenu sous-jacent.

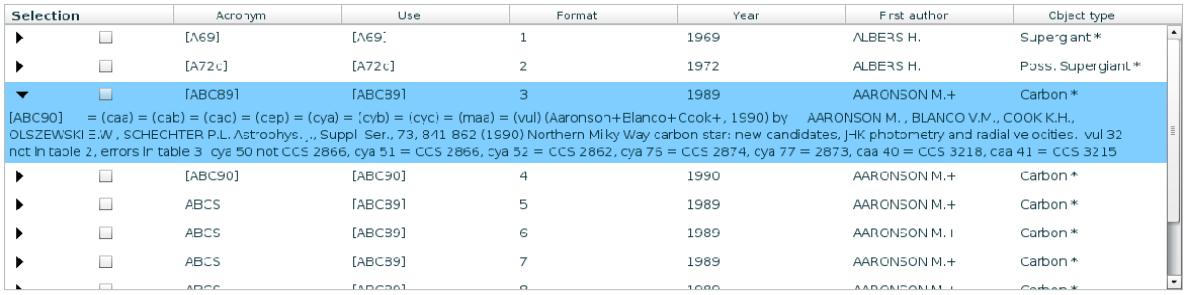


Figure 8.2 – Exemple d'utilisation de la liste des résultats avec une ligne développée.

3.2. Réalisation

3.2.1 La réalisation d'un composant avancé

La réalisation de ce prototype ne soulève pas de problème particulier, si ce n'est l'utilisation du composant DataGridAdvanced. Nous voulons afficher la liste des résultats sous format tabulaire, c'est-à-dire placée dans un tableau avec des fonctions de tri et de redimensionnement des colonnes. Un composant Flex existe et gère très bien toutes ces fonctionnalités, il s'agit du DataGrid (Figure 9.1).

Acronym	Use	Format	Year	First author	Object type
[A69]	[A69]	1	1969	ALBERS H.	Supergiant *
[A72c]	[A72c]	2	1972	ALBERS H.	Poss. Supergiant *
[ABC89]	[ABC89]	3	1989	AARONSON M.+	Carbon *
[ABC90]	[ABC90]	4	1990	AARONSON M.+	Carbon *
ABCS	[ABC89]	5	1989	AARONSON M.+	Carbon *
ABCS	[ABC89]	6	1989	AARONSON M.+	Carbon *

Figure 9.1 – Exemple d'utilisation de DataGrid

Mais nous souhaitons également attacher un bloc d'informations (les informations complémentaires) sous une ligne du tableau, chaque ligne ayant un bloc associé. Le problème est que ce composant gère les données uniquement cellule par cellule, et l'enchaînement de cellules placées côte à côte et formant une ligne n'est pas interprété par le DataGrid. Il faut donc utiliser un composant plus poussé, permettant cette distinction : le DataGridAdvanced.

Ce composant permet de « joindre » des cellules les unes aux autres, formant ainsi une ligne de *n* cellules. Mais malheureusement ce composant est contenu dans la partie payante du framework Flex. Il fallait alors trouver un moyen de simuler le comportement de ce composant sans l'utiliser.

Nous avons dès lors opté pour le composant List (Figure 9.2). Il permet de représenter des données sous format linéaire. Chaque ligne qui le compose peut être personnalisée, on peut donc afficher les données de la manière souhaitée en disposant des blocs de données. De plus, ce composant gère en natif le déplacement par glisser/déposer des différentes lignes qui le compose, permettant ainsi d'intervertir les lignes entre elles.

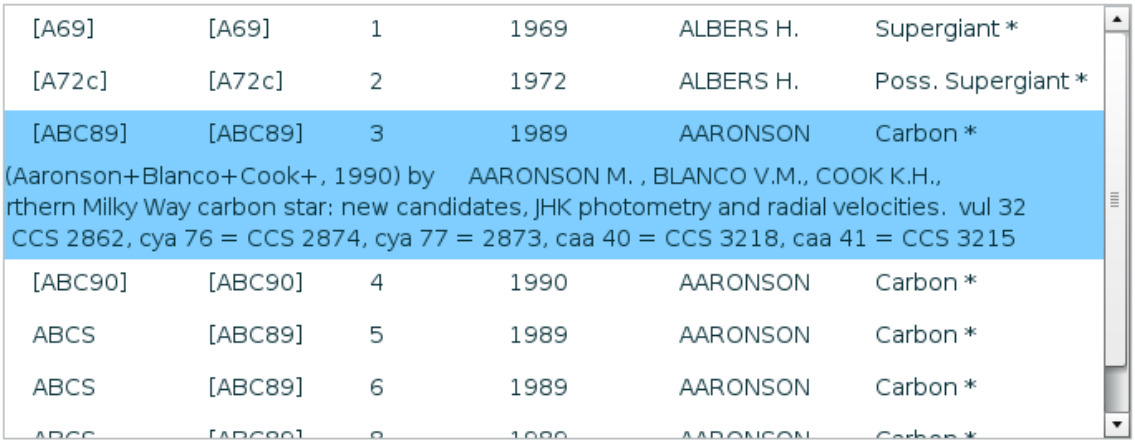


Figure 9.2 – Exemple d'utilisation de List

Le seul problème avec ce composant, c'est qu'il gère les données uniquement en ligne et ne perçoit pas la notion de colonne. On remarque d'ailleurs qu'il n'existe pas d'en-tête comme pour le DataGrid permettant de faire du tri ou du redimensionnement de colonne.

La solution que nous avons trouvé consiste à créer un composant avancé, gérant à la fois les données de manière linéaire comme l'objet List, mais également en colonne comme le DataGrid. Ce composant est en réalité une combinaison des deux précédents (Figure 9.3), en superposant l'objet List par-dessus le DataGrid, ne laissant apparaître que son en-tête. Il s'agit ensuite de les faire communiquer en instaurant la notion de « binding » entre les deux. Il faut préciser que seule la List contient des données, le DataGrid reste quant à lui vide, il n'y a donc pas de duplication inutile de données.

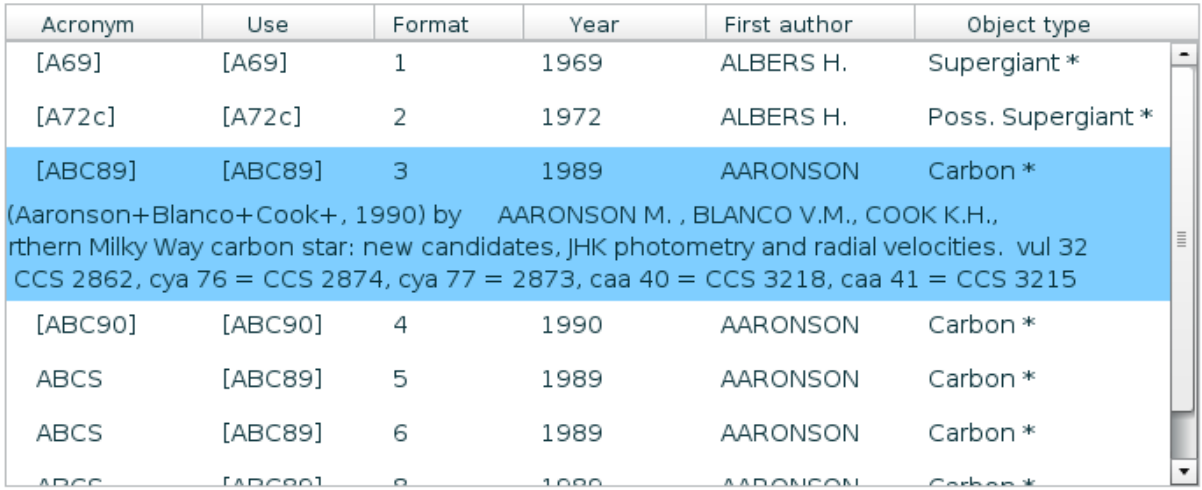


Figure 9.3 – Exemple d'utilisation du composant avancé, assemblage de deux composants

Le binding est un moyen rapide de relier différents composants. Il permet une sorte de communication entre les deux, les changements de l'un influençant sur l'état de l'autre. Nous avons vu précédemment qu'une ligne d'une List peut être considérée comme un assemblage de blocs (Figure 9.4). Il suffit alors de relier la taille d'une colonne avec la taille du bloc qui se situe graphiquement sous l'en-tête de cette colonne. Lorsque l'utilisateur modifiera la taille de la colonne, la taille du bloc en sera immédiatement affectée grâce au mécanisme de binding. En appliquant ce principe à toutes les lignes de la List, tous les blocs se retrouvent redimensionnés en même temps, simulant ainsi un comportement en colonne.

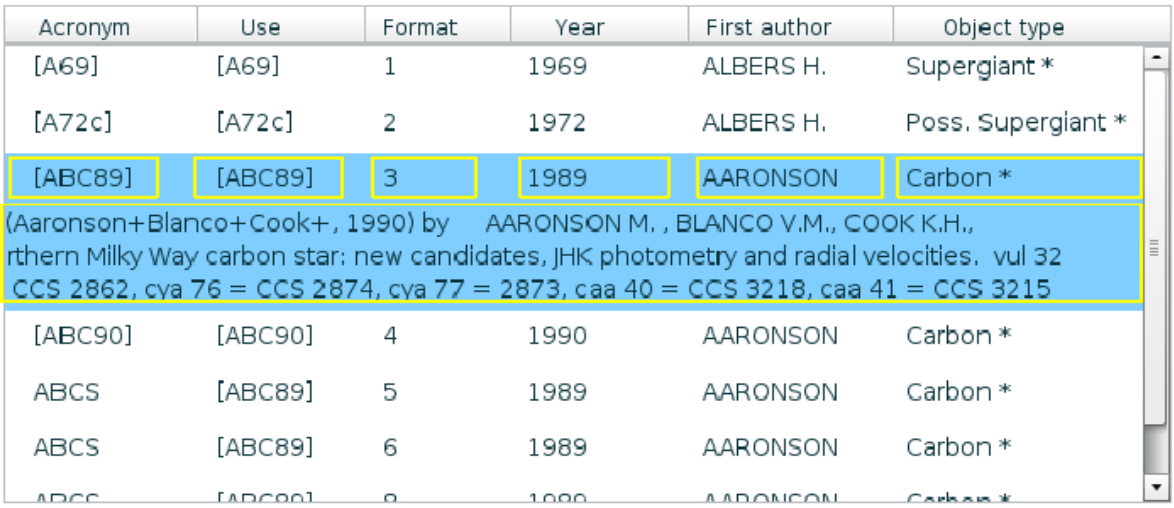


Figure 9.4 – Affichage des différents blocs qui composent une ligne

Concernant le tri sur une colonne, le DataGrid étant vide, aucune fonction de tri par défaut ne peut être utilisée. Il a donc fallut capter l'événement lorsque l'utilisateur clique sur l'en-tête. Puis, selon la colonne cliquée, une fonction de tri personnalisée est appelée. Nous agissons donc directement sur la liste en lui spécifiant un nouvel ordre d'affichage, ce qui implique que toute la List soit vidée puis re-remplie avec les données réagencées. L'objet List étant déjà généré et affiché, et les objets composant la liste étant légers, cette manipulation n'affecte pas la vitesse de rendu, et l'application reste totalement fluide.

3.3. Conclusion

Ce prototype, encore en cours de développement, devrait trouver sa place au sein des services du CDS en remplaçant le dictionnaire de nomenclature actuel. Des améliorations seront également apportées, notamment au niveau du formulaire de recherche qui proposera plus de critères afin de préciser le plus possible la recherche.

Cette application, même si elle reste relativement différente de la précédente, m'a permis de mettre en pratique les connaissances et les méthodes acquises lors de la réalisation du premier prototype. Nous avons également pu étudier et développer notre propre composant Flex, qui pourrait être réutilisé au besoin dans une autre application.

V. Bilan

1. Planning final

Avril	Mai	Juin
◆ 7 : installation et découverte des services ◆ 8-18 : réalisation de l'état de l'art ◆ 22 : réunion 1 – présentation du travail réalisé ◆ 23-25 : finalisation de l'état de l'art et développement du premier prototype (HTML5) ◆ 28 : réunion 2 – prise de connaissance des premiers cas à implémenter et début du développement du second prototype	◆ 7 : réunion 3 – suivi de l'avancement, présentation des problèmes rencontrés et réflexion sur les solutions possible ◆ 14 : mise en place d'une solution permettant l'optimisation du temps d'affichage ◆ 15 : réunion 4 – suivi de l'avancement	◆ 3 : réunion 5 – prise de connaissance du troisième prototype à réalisé ◆ 12 : réunion 6 – suivi de l'avancement ◆ 16-27 : rédaction du rapport et de la présentation

2. Bilan professionnel

Pendant ce stage, j'ai eu l'occasion de me former à des nouvelles technologies et des nouveaux langages (Actionscript, MXML, ...). J'ai également vu, au travers de la réalisation de l'état de l'art, ce qu'est réellement le métier de développeur informatique : il est nécessaire de toujours rester « à la page », de connaître les nouvelles technologies, mais surtout de savoir s'auto-former sur un marché où les technologies sont en constante évolution.

Je me suis également rendu compte de l'importance de la maîtrise de la langue anglaise en informatique. Lorsqu'il s'agit de techniques nouvelles, il est très difficile de trouver des informations dans une autre langue que l'anglais. Mais si lire une documentation technique et comprendre des concepts est une chose, savoir présenter un problème et demander des conseils, sur des forums par exemple, en est une autre.

Enfin, ce stage m'a permis d'être confronté avec le monde du travail, qui est relativement différent de l'environnement scolaire. Par exemple, les réunions ne sont pas seulement des « points de passage » permettant de juger de l'avancement du travail, mais également le moyen de discuter des problèmes rencontrés et de réfléchir tous ensemble aux solutions possibles. Les réunions sont aussi importantes, notamment en phase de prototypage, pour permettre l'évolution du sujet et la mise en place des nouvelles directives de développement.

3. Bilan personnel

Je ressens une réelle satisfaction à avoir mené plusieurs projets d'un bout à l'autre de leur développement, sachant qu'aucun d'entre eux n'a été abandonné en cours de route. Je suis également reconnaissant envers mes encadrants pour l'intérêt qu'ils ont porté à mon travail tout au long du stage.

Pouvoir travailler dans un laboratoire et ainsi être en contact avec le monde de la recherche fut très enrichissant. Outre les compétences techniques que j'ai acquises, j'ai également beaucoup appris sur le monde du travail grâce aux gens que j'ai pu côtoyer.

Enfin, pouvoir me rapprocher du domaine de l'astronomie fut pour moi un réel plaisir. Fasciné depuis toujours par cet univers, j'ai longtemps hésité sur l'orientation de mon parcours scolaire avant de finalement opter pour l'informatique. C'est un choix que je ne regrette pas car il m'a permis, en travaillant au CDS, de concilier mes deux passions.

Conclusion

Ce stage m'a beaucoup apporté, tant sur le plan professionnel que personnel. Il m'a également permis de mettre en pratique bon nombre de connaissances acquises tout au long de mon cursus, et notamment les compétences techniques assimilées avec la licence professionnelle.

En outre, les problèmes que j'ai pu rencontrer m'ont permis de progresser et d'acquérir des connaissances dans des nouvelles technologies informatiques. Ceci me sera, j'en suis sûr, très utile pour mes futures expériences professionnelles. J'ai particulièrement apprécié ce premier contact avec le monde de l'astronomie et je suis heureux de pouvoir prolonger cette expérience suite à un contrat qui m'a été proposé.

Enfin, ce stage m'a amené à revoir ma décision concernant mon projet professionnel et mon choix d'entrer dans le monde du travail dès la fin de ma formation. J'envisage sérieusement de poursuivre mon parcours universitaire afin d'obtenir un niveau de compétences suffisant me permettant, par la suite, de travailler dans le domaine de l'astronomie.

Mots-clés

RIA, XML, Flex, HTML, interface homme-machine, Web 2.0, prototypage, application Web.

Références

Site Internet de l'Observatoire Astronomique de Strasbourg :
http://astro.u-strasbg.fr

Site Internet du Centre de Données astronomiques de Strasbourg :
http://cdsweb.u-strasbg.fr

Wiki du WATHWG :
http://wiki.whatwg.org/wiki/

Le « Working Draft » du W3C (différences avec HTML 4) :
http://www.w3.org/TR/2008/WD-html5-diff-20080122/

Le « cookbook » officiel de Flex :
http://www.adobe.com/cfusion/communityengine/index.cfm?event=homepage&productId=2

Site Internet qui permet de tester les composants Flex :
http://examples.adobe.com/flex2/inproduct/sdk/explorer/explorer.html

Lexique

[1] XMM : *X-ray Multi Mirror*
Le projet XMM-Newton est un télescope spatial de l'Agence Spatiale Européenne, destiné à l'observation des rayons X.

[2] HIPPARCOS : *HIgh Precision PARallax COllecting Satellite*
C'est un satellite de mesure à haute précision.

[3] XMM SSC : X-*ray Multi Mirror Survey Science Center*
Consortium de 10 institutions européenne participant au projet XMM[1].

[4] HTML : *HyperText Markup Language*
Langage de balise qui permet de structurer et de mettre en forme des données, interprété par un navigateur Web. HTML est un langage basé sur le méta-langage SGML[30].

[5] XML : *eXtensible Markup Language*
Langage de balise dont le rôle est de faciliter l'échange de contenus. XML est une simplification de SGML.

[6] RIA : *Rich Internet Application*
Terme pour désigner une application Internet dont l'interface utilisateur est riche en composants.

[7] AJAX : *Asynchronous JAvascript and XML*
Approche de développement d'interfaces Web riches.

[8] XAML : *eXtensible Application Markup Language*
Langage déclaratif d'interface graphique basé sur les composants de l'interface de Windows Vista.

[9] HD : *Haute Définition*
Sigle souvent utilisé pour désigner un ensemble de normes de vidéo numérique permettant d'avoir une image de meilleure qualité par rapport à la référence dite SD (Standard Definition).

[10] SWF : *ShockWave Flash*
Format de fichier binaire exécutable par une lecteur Flash.

[11] DHTML : *Dynamic HTML*
Ensemble de techniques permettant de réaliser des pages Web qui peuvent être modifiées (par opposition à du contenu sur une page statique).

[12] MSAA : *Microsoft Active Accessibility*
Normes d'accessibilités des applications Web selon Microsoft.

[13] JVM : *Java Virtual Machine*
Machine virtuelle permettant d'interpréter et d'exécuter du code binaire Java.

[14] W3C : *World Wide Web Consortium*
Organisme de normalisation chargé de promouvoir la compatibilité des technologies du Web.

[15] MSIL : *MicroSoft Intermediate Language*
Langage entre code machine et langage de programmation, compilé à la volée.

[16] GWT : *Google Web Toolkit*
Framework développé par Google, permettant de créer des pages web dynamiques en utilisant la technologie AJAX[7].

[17] CLR : *Common Language Runtime*
Composant de machine virtuelle du framework .NET qui définit l'environnement d'exécution des codes de programmes.

[18] XHTML : *eXtensible HTML*
Langage de balisage servant à l'écriture de pages Web qui se fonde sur la syntaxe définie par XML[5].

[19] WHATWG : *Web Hypertext Application Technology Working Group*
Collaboration non officielle des différents développeurs de navigateurs Web ayant pour but le développement de nouvelles technologies destinées à faciliter l'écriture et le déploiement d'applications à travers le Web.

[20] API : *Application Programming Interface*
Interface de programmation qui définit la manière dont un composant informatique peut communiquer avec un autre.

[21] DOM : *Document Object Model*
Interface indépendante de tout langage de programmation permettant à des programmes d'accéder ou de mettre à jour le contenu, la structure ou le style de documents.
[22] SWT : *Standard Widget Toolkit*
Bibliothèque graphique libre pour Java.

[23] IHM : *Interface Homme-Machine*
Étudie la façon de concevoir des systèmes informatiques qui soient ergonomiques, c'est-à-dire efficaces, faciles à utiliser ou plus généralement adaptés à leur contexte d'utilisation.

[24] SDK : *Software Development Kit*
Ensemble d'outils permettant aux développeurs de créer des applications de type défini.

[25] JDK : *Java Development Kit*
Environnement dans lequel le code Java est compilé pour être transformé en code machine afin que la JVM[13] puisse l'interpréter.

[26] ECMAScript : *European Computer Manufacturers AssociationScript*
Il s'agit d'un standard dont les spécifications sont implémentées dans la plupart des langages de script.

[27] MXML : *Macromedia XML*
Langage à balises de description d'interface utilisateur pouvant embarquer des instructions ActionScript.

[28] IDE : *Integrated Development Environment*
Logiciel regroupant un éditeur de texte, un compilateur, des outils automatiques de fabrication, et souvent un débogueur, permettant de simplifier le travail du développeur.

[29] MVC : *Model-View-Controller*
Architecture et méthode de conception qui organise l'IHM[23] d'une application.

[30] SGML : *Standard Generalized Markup Language*
Langage de description à balises, de norme ISO.

Annexes

Annexe A1 – Exemple de code source d'une application Flex

(fichier MXML avec code Actionscript embarqué)

```
<?xml version="1.0" encoding="utf-8"?>
<!--L'en-tête doit être la première ligne du code-->

<!--Déclaration des caractéristiques de l'application-->
<mx:Application
  xmlns:mx="http://www.adobe.com/2006/mxml"
  layout="horizontal"
  width="500"
  height="500"
  >

  <!--
      La balise Script permet d'incorporer du code Actionscript
      au sein d'un fichier MXML
  -->
  <mx:Script>
    <![CDATA[
      import mx.core.UIComponent;

       /*
            Fonction qui permet de dessiner 10 points sur un objet
      de type UIComponent puis d'ajouter cet objet sur
      l'objet Canvas pour afficher graphiquement les points
       */
       private function drawPoints(event:MouseEvent) : void
       {
         //Instanciation d'un objet UIComponent qui prend la taille
         //(longueur et largeur) du canvas
         var uic : UIComponent = new UIComponent();
         uic.width = monCanvas.width;
         uic.height = monCanvas.height;

         //Boucle qui permet de dessiner un point sur le UIComponent
         for(var i:int=0; i<10; i++)
         {
          //Défini la couleur du point aléatoirement
           uic.graphics.beginFill(Math.random()*0xffffff);
          //Défini la position du point aléatoirement sur son
          //conteneur (l'objet UIComponent)
           uic.graphics.drawCircle(Math.random()*150, Math.random()*150,4);
           uic.graphics.endFill();
         }

         //Ajout du UIComponent sur le canvas
         monCanvas.addChild(uic);
       }
    ]]>
  </mx:Script>


  <!--
      Déclaration du panel
  -->
  <mx:Panel
    id="monPanel"
    x="0"
```

```
    y="0"
    width="450"
    height="450"
    verticalAlign="middle"
    horizontalAlign="center"
    title="Le titre de mon panel"
  >

  <!--
      Déclaration du bouton de déclenchement d'affichage des points
  -->
  <mx:Button
    x="0"
    y="0"
    id="monBouton"
    label="Valider"
    click="drawPoints(event)"
  />

  <!--
      Déclaration du canvas
  -->
  <mx:Canvas
    id="monCanvas"
    x="50"
    y="50"
    width="400"
    height="400"
    backgroundColor="black"
  />
  </mx:Panel>

</mx:Application>
```

Annexe A2 – Extrait d'un fichier XML au format VOTable

(fichier contenant à l'origine les données de 1090 objets astronomiques tronqué ici à 4 objets)

```
<?xml version='1.0'?>
<VOTABLE version="1.1"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.ivoa.net/xml/VOTable/v1.1
http://www.ivoa.net/xml/VOTable/v1.1"
>

<!--
 !  VOTable written by STIL version 2.6-2
(uk.ac.starlink.votable.VOTableWriter)
 !  at 2008-04-30T09:09:14
 !-->

<RESOURCE>
<TABLE name="Joined" nrows="1087">
<PARAM arraysize="*" datatype="char" name="Match type" value="Pair match,
best matched pairs only">
<DESCRIPTION>Type of match which created this table</DESCRIPTION>
</PARAM>
<PARAM arraysize="*" datatype="char" name="Match algorithm" value="Exact
Value">
<DESCRIPTION>Matching algorithm which created this table</DESCRIPTION>
</PARAM>
<PARAM arraysize="*" datatype="char" name="Join Type" value="1 and 2 (An
output row for each row represented in both input tables)">
<DESCRIPTION>Determines which rows appear in output table</DESCRIPTION>
</PARAM>
<PARAM arraysize="*" datatype="char" name="First input table" value="1:
m51.csv"/>
<PARAM arraysize="*" datatype="char" name="Second input table" value="2:
Simbad.xml"/>
<FIELD datatype="short" name="X">
<VALUES null='-32768'/>
</FIELD>
<FIELD datatype="short" name="Y">
<VALUES null='-32768'/>
</FIELD>
<FIELD datatype="double" name="_RAJ2000" ucd="pos.eq.ra;meta.main"/>
<FIELD datatype="double" name="_DEJ2000" ucd="pos.eq.dec;meta.main"/>
<FIELD ID="MAIN_ID" arraysize="*" datatype="char" name="MAIN_ID"
ucd="meta.id;meta.main" width="22">
<DESCRIPTION>Main identifier for an object</DESCRIPTION>
</FIELD>
<FIELD ID="OTYPE" arraysize="*" datatype="char" name="OTYPE"
ucd="src.class" width="6">
<DESCRIPTION>Object type</DESCRIPTION>
</FIELD>
<FIELD ID="RA_s" arraysize="13" datatype="char" name="RA" precision="8"
ucd="pos.eq.ra;meta.main" width="13">
<DESCRIPTION>Right ascension</DESCRIPTION>
</FIELD>
<FIELD ID="DEC_s" arraysize="13" datatype="char" name="DEC" precision="8"
ucd="pos.eq.dec;meta.main" width="13">
<DESCRIPTION>Declination</DESCRIPTION>
</FIELD>
```

```
<FIELD ID="COO_ERR_MAJA" datatype="float" name="COO_ERR_MAJA" precision="3"
ucd="phys.angSize.smajAxis;pos.errorEllipse;pos.eq" unit="mas" width="6">
<DESCRIPTION>Coordinate error major axis</DESCRIPTION>
</FIELD>
<FIELD ID="COO_ERR_MINA" datatype="float" name="COO_ERR_MINA" precision="3"
ucd="phys.angSize.sminAxis;pos.errorEllipse;pos.eq" unit="mas" width="6">
<DESCRIPTION>Coordinate error minor axis</DESCRIPTION>
</FIELD>
<FIELD ID="COO_ERR_ANGLE" datatype="short" name="COO_ERR_ANGLE"
ucd="pos.posAng;pos.errorEllipse;pos.eq" unit="deg" width="3">
<DESCRIPTION>Coordinate error angle</DESCRIPTION>
<VALUES null='-32768'/>
</FIELD>
<FIELD ID="PMRA" datatype="double" name="PMRA" precision="3"
ucd="pos.pm;pos.eq.ra" unit="mas.yr-1" width="9">
<DESCRIPTION>Proper motion in RA</DESCRIPTION>
</FIELD>
<FIELD ID="PMDEC" datatype="double" name="PMDEC" precision="3"
ucd="pos.pm;pos.eq.dec" unit="mas.yr-1" width="9">
<DESCRIPTION>Proper motion in DEC</DESCRIPTION>
</FIELD>
<FIELD ID="FLUX_B" datatype="float" name="B" ucd="phot.mag;em.opt.B"
unit="mag" width="10">
<DESCRIPTION>Magnitude B</DESCRIPTION>
</FIELD>
<FIELD ID="FLUX_V" datatype="float" name="V" ucd="phot.mag;em.opt.V"
unit="mag" width="10">
<DESCRIPTION>Magnitude V</DESCRIPTION>
</FIELD>
<FIELD ID="SP_TYPE" arraysize="*" datatype="char" name="SP_TYPE"
ucd="src.spType" width="6">
<DESCRIPTION>MK spectral type</DESCRIPTION>
</FIELD>
<FIELD ID="GALDIM_MAJAXIS" datatype="float" name="GALDIM_MAJAXIS"
ucd="phys.angSize.smajAxis" unit="arcmin" width="4">
<DESCRIPTION>Angular size major axis</DESCRIPTION>
</FIELD>
<FIELD ID="GALDIM_MINAXIS" datatype="float" name="GALDIM_MINAXIS"
ucd="phys.angSize.sminAxis" unit="arcmin" width="4">
<DESCRIPTION>Angular size minor axis</DESCRIPTION>
</FIELD>
<FIELD ID="GALDIM_ANGLE" datatype="short" name="GALDIM_ANGLE"
ucd="pos.posAng" unit="deg" width="3">
<DESCRIPTION>Galaxy ellipse angle</DESCRIPTION>
<VALUES null='-32768'/>
</FIELD>
<FIELD ID="BIBLIST_N" datatype="double" name="BIBLIST" ucd="meta.bib"
width="10">
<DESCRIPTION>List of Bibcodes</DESCRIPTION>
</FIELD>

<DATA>
<TABLEDATA>
    <TR>
    <TD>351</TD>
    <TD>387</TD>
    <TD>202.46820833333334</TD>
    <TD>47.19466666666666</TD>
    <TD>4C 47.36A</TD>
    <TD>Seyfert_2</TD>
    <TD>13 29 52.37</TD>
```

```
    <TD>+47 11 40.8</TD>
    <TD>10800.0</TD>
    <TD>10800.0</TD>
    <TD>90</TD>
    <TD></TD>
    <TD></TD>
    <TD>8.8</TD>
    <TD></TD>
    <TD> </TD>
    <TD>8.317</TD>
    <TD>7.079</TD>
    <TD>-32768</TD>
    <TD>1952.0</TD>
  </TR>
  <TR>
    <TD>351</TD>
    <TD>389</TD>
    <TD>202.4679583333333</TD>
    <TD>47.194222222222216</TD>
    <TD>[LPS2002] 6</TD>
    <TD>Star</TD>
    <TD>13 29 52.31</TD>
    <TD>+47 11 39.2</TD>
    <TD></TD>
    <TD></TD>
    <TD>-32768</TD>
    <TD></TD>
    <TD></TD>
    <TD></TD>
    <TD></TD>
    <TD> </TD>
    <TD></TD>
    <TD></TD>
    <TD>-32768</TD>
    <TD>1.0</TD>
  </TR>
  <TR>
    <TD>349</TD>
    <TD>386</TD>
    <TD>202.46875</TD>
    <TD>47.19499999999999</TD>
    <TD>[CPF88] 132746.2+472710.0</TD>
    <TD>Maser</TD>
    <TD>13 29 52.5</TD>
    <TD>+47 11 42</TD>
    <TD></TD>
    <TD></TD>
    <TD>-32768</TD>
    <TD></TD>
    <TD></TD>
    <TD></TD>
    <TD></TD>
    <TD> </TD>
    <TD></TD>
    <TD></TD>
    <TD>-32768</TD>
    <TD>4.0</TD>
  </TR>
  <TR>
    <TD>351</TD>
    <TD>385</TD>
```

```
    <TD>202.46816666666666</TD>
    <TD>47.19522222222221</TD>
    <TD>[LPS2002] 15</TD>
    <TD>Star</TD>
    <TD>13 29 52.36</TD>
    <TD>+47 11 42.8</TD>
    <TD></TD>
    <TD></TD>
    <TD>-32768</TD>
    <TD></TD>
    <TD></TD>
    <TD></TD>
    <TD></TD>
    <TD> </TD>
    <TD></TD>
    <TD></TD>
    <TD>-32768</TD>
    <TD>1.0</TD>
  </TR>
</TABLEDATA>
</DATA>
</TABLE>
</RESOURCE>
</VOTABLE>
```