Le client riche, toujours d'actualité ? Dans quelles technologies les entreprises doivent-elles investir ?

h e g

Haute école de gestion de Genève
Geneva School of Business Administration

Travail de diplôme réalisé en vue de l'obtention du diplôme HES

par :

Jean-Charles PLAGNARD

Conseiller au travail de diplôme :

Rolf HAURI, Professeur HES

Genève, le 16 Octobre 2009

Haute École de Gestion de Genève (HEG-GE)

Filière Informatique de Gestion

Déclaration

Ce travail de diplôme est réalisé dans le cadre de l'examen final de la Haute école de gestion de Genève, en vue de l'obtention du titre de Bachelor en Informatique de Gestion. L'étudiant accepte, le cas échéant, la clause de confidentialité. L'utilisation des conclusions et recommandations formulées dans le travail de diplôme, sans préjuger de leur valeur, n'engage ni la responsabilité de l'auteur, ni celle du conseiller au travail de diplôme, du juré et de la HEG.

« J'atteste avoir réalisé seul le présent travail, sans avoir utilisé des sources autres que celles citées dans la bibliographie. »

Fait à Genève, le 16 Octobre 2009

Jean-Charles Plagnard

Remerciements

En premier lieu, je tiens à remercier Mr Rolf Hauri pour ses conseils précieux et son suivi qui m'ont beaucoup aidé durant la réalisation de ce travail de diplôme.

Je tiens également à remercier tout particulièrement Mr Killian L'Herbette pour ses recommandations et pour m'avoir fait partager ses connaissances depuis de nombreuses années.

Je tiens ensuite à exprimer toute ma gratitude à Mr Daniel Baumgardt pour ses suggestions importantes.

Enfin, je tiens à témoigner toute ma reconnaissance à ma famille et à mes proches pour m'avoir soutenu et encouragé tout au long de ce travail.

Sommaire

L'objectif principal de ce travail de diplôme est de révéler que depuis l'apparition d'applications de type client riche[1] au début des années 2000, leur utilisation sur le Web[2] est toujours d'actualité.

Il sera question de traiter en premier lieu les particularités et les technologies qui existent pour mettre en œuvre ce type d'application. L'évolution des architectures client-serveur a permit un nouvel usage du Web permettant l'utilisation d'applications se rapprochant des applications de bureau et offrant une expérience utilisateur riche.[3] Les différentes technologies existantes sont nombreuses et offrent de beaucoup d'alternatives pour réaliser ces applications.

Ensuite, la mise en place d'un prototype utilisant deux technologies permettra de mettre en évidence les étapes nécessaires pour la création d'applications riches.

[1] Application Web fournissant les mêmes caractéristiques qu'une application de bureau

[2] World Wide Web

[3] Interaction entre l'utilisateur et l'application simple, intuitive

Table des matières

Déclaration i
Remerciements ii
Sommaire iii
Table des matières iv
Liste des Tableaux vi
Liste des Figures vi
Introduction 1
1. L'évolution des interfaces utilisateur 2
2. RIA – Rich Internet Application 7
2.1 Qu'est ce que RIA ? 7
2.2 Qu'apportent les RIAs aux utilisateurs 7
2.3 Un exemple avec Google Map et une carte traditionnelle (1bis.com) 8
2.4 Quelques chiffres 10
2.5 Qu'apportent les RIAs aux entreprises 11
2.6 Les inconvénients 14
3. Architecture typique des RIAs 16
4. Les technologies et les outils de développement 17
4.1 Ajax 17
4.2 Adobe 19
4.3 OpenLazslo 21
4.4 Microsoft 21
4.4.1 WPF 21
4.4.2 Silverlight 22
4.5 Mozilla 23
4.6 Sun Microsystem 24
4.7 Récapitulatif 25
5. Présentation de l'application mise en place 27
5.1 Choix des technologies 27
5.2 Objectifs et cahier des charges 27
5.3 Flex 28
5.3.1 Installation 28
5.3.2 Réalisation 28
5.3.3 Description des fichiers mxml 29
5.3.3.1 ClientRiche.mxml 29
5.3.3.2 SongRenderer.mxml 30
5.3.3.3 AutoCompleteSample.mxml 31
5.3.4 Déploiement de l'application 31
5.3.5 Application 31
5.3.6 Bilan 37
5.4 OpenLaszlo 37

5.4.1 Installation *37*
5.4.2 Réalisation *38*
5.4.3 Déploiement de l'application *40*
5.4.4 Application *40*
5.4.5 Bilan *44*

Conclusion **46**

Bibliographie **47**

Liste des Tableaux

Tableau 1 Récapitulatif solutions clients 6

Tableau 2 Récapitulatif pénétration technologies.....25

Tableau 3 Récapitulatif des technologies.....26

Tableau 4 Points positifs et négatifs de Flex 37

Tableau 5 Points positifs et négatifs d'OpenLaszlo 45

Liste des Figures

Figure 1 Classification des types de client-serveur selon Gartner Group.....3

Figure 2 Evolution des usages des architectures client-serveur 5

Figure 3 Du client léger au client lourd – les technologies mises en œuvre.....6

Figure 4 Carte interactive Google Map.....8

Figure 5 Carte interactive Google Map – photos 9

Figure 6 Carte statique – 1bis.com.....9

Figure 7 Les RIAs améliorent l'expérience Web.....10

Figure 8 Critères de préférence des RIAs 11

Figure 9 Atteinte des objectifs des RIAs.....13

Figure 10 Choix de développement de RIA.....14

Figure 11 Architecture et communication d'une RIA.....16

Figure 12 MCD de l'application 28

Figure 13 Balises MXML du fichier principal de l'application Flex.....29

Figure 14 Initialisation de l'application avec Flex.....32

Figure 15 Sélection d'un genre avec Flex 32

Figure 16 Sélection d'un artiste avec Flex.....33

Figure 17 Sélection d'un album avec Flex....33
Figure 18 Navigation sur la liste des chansons avec Flex....34
Figure 19 Lecture, pause et stop d'une chanson avec Flex....35
Figure 20 Albums reliés avec Flex....35
Figure 21 Recherche et auto-complétion avec Flex....36
Figure 22 Résultat d'une recherche avec Flex....36
Figure 23 Initialisation de l'application avec OpenLaszlo....40
Figure 24 Sélection d'un genre avec OpenLaszlo....41
Figure 25 Sélection d'un artiste avec OpenLaszlo....45
Figure 26 Sélection d'un album avec OpenLaszlo....42
Figure 27 Navigation sur la liste des chansons avec OpenLaszlo....43
Figure 28 Résultat d'une recherche avec OpenLaszlo....44

Introduction

L'évolution du Web[4] a commencé avec l'HTML qui à la base fournissait du contenu statique via un browser[5] Web. Au fil du temps, le Web n'a cessé d'évoluer permettant l'affichage de contenu riche – vidéos, audio, multimédia... – de plus en plus interactifs et dynamiques. De plus, le besoin des entreprises devenant grandissant quant aux fonctionnalités des applications, il fallait introduire la complexité des domaines métiers des entreprises sur les applications Web. La gestion dynamique de données a permis l'affichage d'informations variant suivant les demandes des utilisateurs et améliorant de ce fait l'expérience des utilisateurs. La capacité des navigateurs à été optimisée permettant l'amélioration des scripts – JavaScript – et des composants – entre autres Applet et ActiveX – toujours plus performants et supportant les attentes des utilisateurs. Les niveaux d'architectures client-serveur ont offert différents choix pour mettre en œuvre des applications. D'abord le *client léger* qui laisse les traitements du côté serveur, ensuite le *client lourd* qui réalise la quasi-totalité des traitements sur la machine cliente. L'apparition de la communication asynchrone entre serveur et client avec AJAX a permis de rendre l'utilisation du Web plus utilisable, plus fiable, réduisant le temps d'attente et les charges sur le serveur. Avec l'ensemble de ces évolutions, le comportement des utilisateurs a changé, les habituant à utiliser des applications avec du contenu riche et facilement manipulable. Le *client riche* – ou *RIA*[6] – représente cette évolution des applications Web, associant la puissante et intuitive expérience utilisateur d'une application de bureau avec la portée du Web. Leurs impacts sur les utilisateurs sont nombreux et l'adoption d'une solution RIA pour les entreprises leur permet d'ajouter de la valeur à leurs produits et services. Il existe des défis dans l'adoption ou non d'un client riche, et les bénéfices que peuvent gagner les entreprises sur le long terme est incontestable et ne peuvent pas être ignorés.

Le but de ce mémoire est de présenter les clients riches, de montrer que depuis leur introduction au début des années 2000, leur utilisation reste plus que jamais d'actualité et qu'ils fournissent des fonctionnalités évoluées permettant la manipulation de contenu complexe. Il sera aussi question de présenter les technologies et les différents acteurs du marché qui proposent des outils pour réaliser des applications riches. Enfin, une présentation d'une application simple réalisée avec deux technologies mettra en évidence les étapes et problèmes qui peuvent survenir lors de la construction d'une application riche.

---

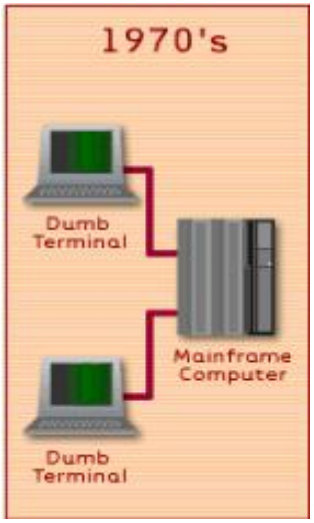
[4] World Wide Web

[5] Navigateur tel qu'Internet Explorer

[6] Rich Internet Application

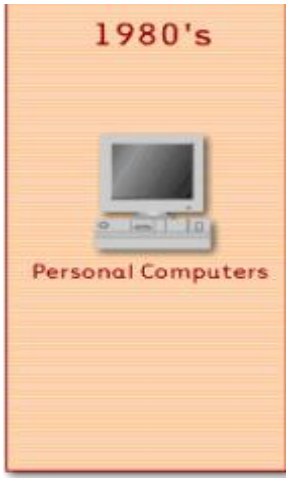
1. L'évolution des interfaces utilisateur

À l'origine des interfaces utilisateurs, les informations étaient centralisées sur des *mainframes* - « ordinateur central » en français - qui possédaient une forte puissance transactionnelle de telle sorte qu'un système tout entier reposait sur ses capacités de traitement. Les utilisateurs se connectaient sur cette machine centralisée à l'aide d'ordinateurs sans capacité de traitement local – uniquement saisie du clavier et affichage de l'écran.

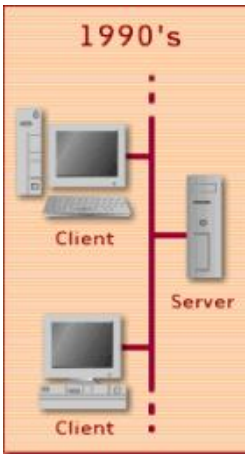


Source : http://www.automatedbuildings.com/news/jan02/art/alc/alc.htm

On peut par abus de langage appeler ces terminaux des *clients pauvres – ou clients légers* - car ils ne réalisent aucun traitement à part l'affichage d'informations envoyées par une machine tierce.



Ensuite sont apparus les premiers *ordinateurs de bureau* – ou *desktop, PC* – destinés à l'usage d'une personne dont les dimensions sont réduites. L'interface graphique permettait de modifier l'état du système à l'aide des fenêtres, menus, icônes, boutons, pointeurs, onglets…Cela rendait ces machines fortement interactives (Drag & Drop) et très flexibles (redimensionnables, facilement configurables). Elles possédaient une puissance de calcul suffisante pour pouvoir prendre en charge l'intégralité des traitements des applications installées sur la machine. Ces applications permettaient de réaliser des traitements évolués et possédaient une interface graphique sophistiquée – ce qui rendait leur utilisation plus fluide. On appelle ces applications des applications de type *client lourd*.

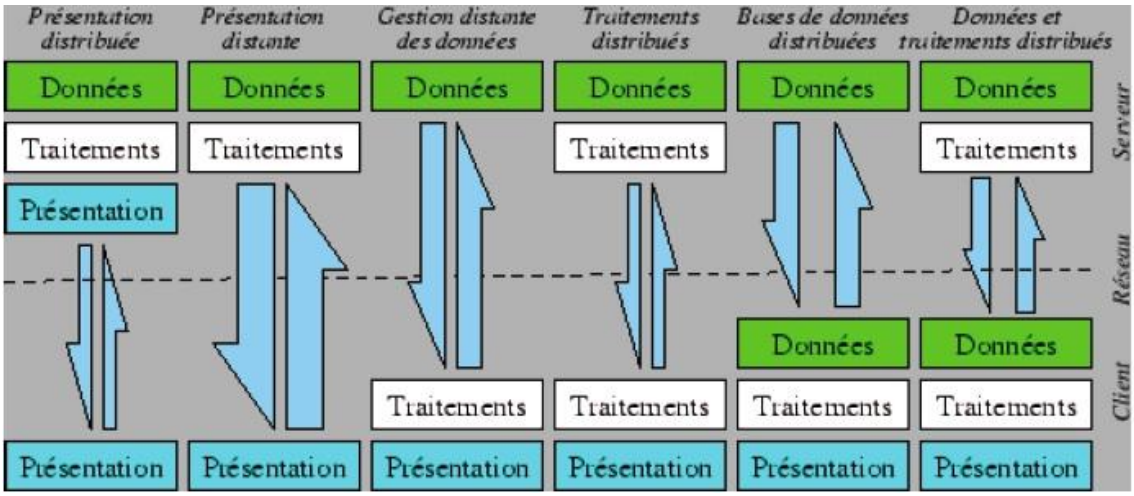


Pour permettre l'échange d'informations entre PCs, le stockage important de données et les traitements lourds, l'architecture de type *client-serveur* a évolué pour permettre une communication entre les ordinateurs de bureau et les serveurs. Les applications étaient installées sur les serveurs. Le client envoyait une requête au serveur ; celui-ci lui renvoyait les réponses et effectuait des

traitements. Les serveurs permettaient aussi le partage de données.

Cette architecture présente plusieurs types de client-serveur. Le cabinet de consultants américain, Gartner Group a publié un schéma des différents types existants.

Figure 1
Classification des types de client-serveur selon Gartner Group



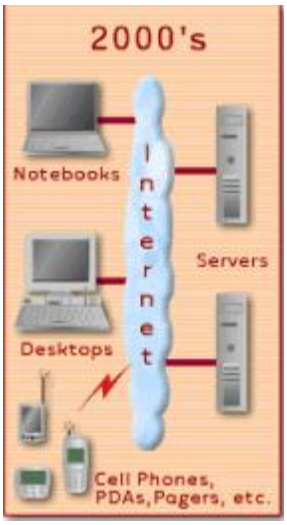
http://rangiroa.essi.fr/riveill/enseignement/2008-09/dotnet/remi.leblond.free.fr_probatoire_probatoire-1.pdf

Ce schéma représente les communications entre client et serveur à travers le réseau. Ce type d'architecture permet de faire fonctionner une application chez le client à travers le réseau. Une application est découpée en trois niveaux d'abstraction distincts :

- Les données qui regroupent l'ensemble des dispositifs qui permettent de gérer les informations stockées par l'application
- Les traitements qui contiennent les spécificités – ou couche métier – de l'application et qui constituent l'application elle-même.
- La présentation qui permet à l'utilisateur d'interagir avec l'application.

Les applications ayant évolué, on a vu apparaître les sites Web sur le réseau Internet. Un *site Web* représente un ensemble de pages Web hyperliées entre elles. La consultation d'un site Web est possible à l'aide d'un navigateur. Les premiers sites Web se résumaient principalement à la navigation entre les pages, les formulaires et l'affichage des données envoyées par le serveur. On se trouvait dans un type d'architecture de *client léger* – ou *présentation distribuée* – où la totalité de la logique métier était traitée du côté serveur.

Rapidement, il a fallu aller plus loin que les simples sites Web. Le besoin grandissant des utilisateurs et la constante évolution des technologies ont permis la prolifération des *sites Web dynamiques*. Les données provenaient alors de base de données et l'affichage des pages consultées variait suivant la demande des utilisateurs. L'utilisation de ces outils est devenue incontournable – aussi pour les entreprises – et le développement de solutions Web a fortement augmenté. Les entreprises réalisaient des applications Web pour gérer leurs processus métiers internes à l'entreprise ou réalisaient des plates-formes de boutiques en ligne.

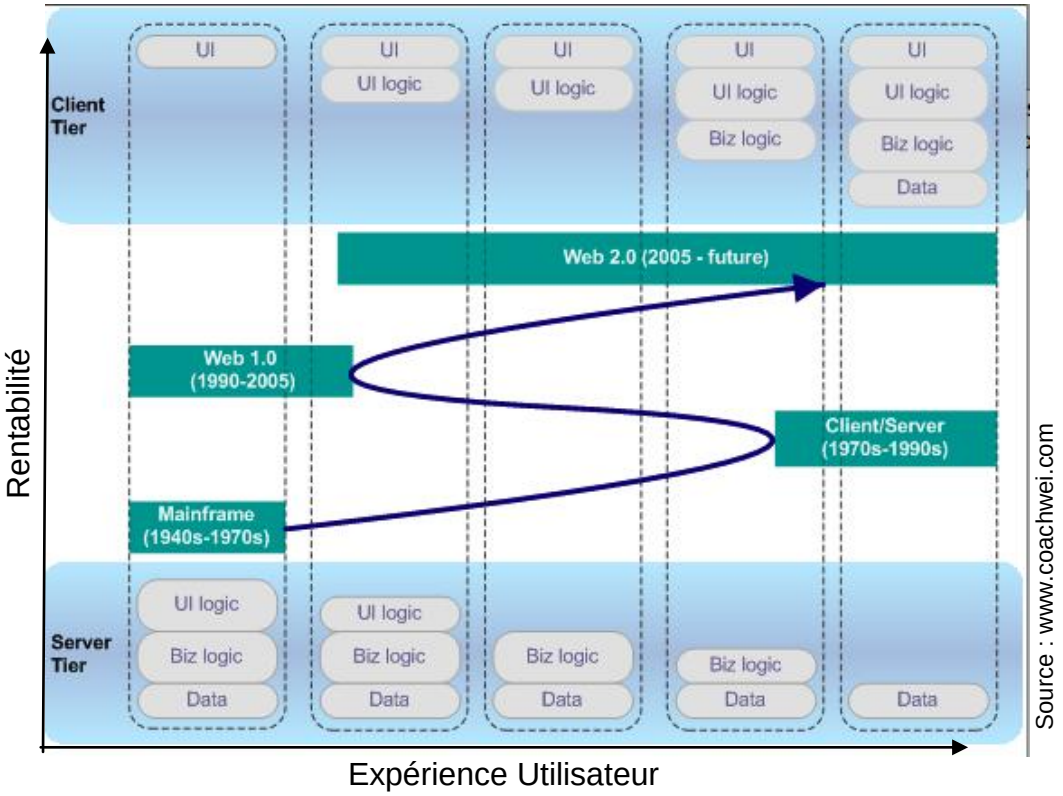


Les années 2000 ont été le siège du changement des comportements utilisateurs vis-à-vis du Web. L'augmentation des utilisateurs, des sites Web de plus en plus dynamiques et performants, du trafic et de la bande passante, a fait que le Web est devenu plus interactif. L'expérience utilisateur a été fortement améliorée pour fournir maintenant une navigation fluide. L'accès au contenu riche - vidéos, audio, animations …- devient désormais incontournable pour tous les utilisateurs du Web – entreprises, consommateurs, particuliers… La notion de *client riche* a fait son apparition dès lors que les traitements des applications Web n'étaient plus réalisés uniquement sur les serveurs. Le *client riche ou RIA* – Rich Internet Application – est un compromis entre le *client léger* – tous les traitements sont réalisés sur le serveur et le client s'occupe de l'affichage – et le *client lourd* – tous les traitements sont réalisés sur la machine de l'utilisateur. Les applications de type client riche sont des applications distantes avec les avantages des applications locales, c'est-à-dire qu'elles possèdent la rapidité de réaction des applications installées et la facilité de distributions des sites Web. Ces applications offrent des interactions riches à un utilisateur, les traitements sont à la fois effectués du côté serveur et du côté client. Elles permettent à l'utilisateur de manipuler les données.

Les architectures *client-serveur* ont subi un mouvement de balancier entre les *clients légers* et les *clients lourds*. La figure ci-dessous représente l'évolution des usages des architectures client-serveur, ainsi que leur portée au niveau de l'expérience utilisateur et rentabilité. Auparavant, ni les architectures centrées serveur ou clients n'étaient

appropriées. Les architectures centrées serveurs comme les mainframes ou les simples sites Web statiques n'offraient pas une réelle expérience utilisateur.

Figure 2
Evolution des usages des architectures client-serveur



L'évolution des technologies a permis une constante augmentation de la rentabilité des applications car les besoins ont évolué, mais cela n'a pas été le cas de l'expérience utilisateur. Elle a augmenté en passant du client léger au client lourd, diminué en passant du client lourd au client léger, et maintenant augmente avec le client riche. Le client riche représente donc la transition des applications Web depuis le client léger à un client qui fournit une expérience utilisateur d'une application de bureau de type client lourd.

Il existe différentes technologies pour mettre en exergue des applications riches. La figure ci-dessous représente les différentes technologies – non exhaustives - existantes pour réaliser des applications pour tout type de client.

Figure 3
Du client léger au client lourd – les technologies mises en œuvre

Source : http://webilus.com/illustration/interfaces-riches-du-client-leger-au-client-lourd-ria-rda

Les différentes architectures clients offrent des possibilités qui ne sont pas exploitées de la même manière suivant le type de client choisi. Le tableau ci-dessous représente les caractéristiques de chaque solution et leur valeur.

Tableau 1
Récapitulatif solutions clients

	Client Léger	**Client Lourd**	**Client Riche**
Facilité de déploiement	+++	+	+++
Interactivité	+	+++	+++
Facilité de MAJ[7] et maintenance	+++	+	+++
Productivité	+	+++	+++
Traitements	Serveur	Client	Client & serveur

La suite de ce mémoire présentera en détail les clients riches et mettra en évidence leurs caractéristiques qui font leur réussite. La présentation des technologies associées et de leur fonctionnement démontrera de plus que la mise en place de solutions riches apporte un succès incontestable.

[7] Mise à jour

2. RIA – Rich Internet Application

2.1 Qu'est ce que RIA ?

Les applications de type client riche ou RIA sont des applications Web qui permettent aux utilisateurs d'avoir une réelle expérience utilisateur, leur offrant une interaction riche, simple et intuitive. Elles fournissent les mêmes fonctionnalités avancées qu'une application de bureau et l'accessibilité des sites Web. L'activité liée à l'interface utilisateur est prise en charge par le client et les opérations sur les données sont réalisées par le serveur.

Les avancées technologiques ont changé les pratiques des utilisateurs vis-à-vis du Web. L'accès à du contenu multimédia – documents, vidéos, audios… -, l'interaction et la facilité de déploiement ont défini un nouvel usage du Web.

Nous allons donc voir en quoi les applications riches jouent un rôle important dans la définition des nouveaux usages du Web, apportant une nouvelle expérience applicative utilisateur.

2.2 Qu'apportent les RIAs aux utilisateurs

Le cabinet d'analyste Forrester Research spécialisé en technologies et système d'information a réalisé plusieurs études concernant les Rich Internet Applications. Une de leur étude[8] fait ressortir que les RIA améliorent l'expérience utilisateur sur plusieurs niveaux lorsque celles-ci sont bien conçues :

Premièrement, elles permettent **la manipulation directe du contenu** : elles aident l'utilisateur à trouver, manipuler et afficher du contenu pertinent sans avoir à constamment rafraîchir la page. Elles lui permettent donc de voir instantanément l'impact de ses changements sur les différentes options accessibles, et ce de manière aisée. Par exemple glisser librement des données sur l'écran, revenir en arrière sur des étapes de processus.

Elles **interagissent** avec les médias : elles permettent de lire, mais aussi d'interagir avec les contenus multimédias. De plus en plus de formats sont supportés par les différentes technologies, augmentant la prise en charge de ces médias.

[8] Rich Internet Application : Why And How

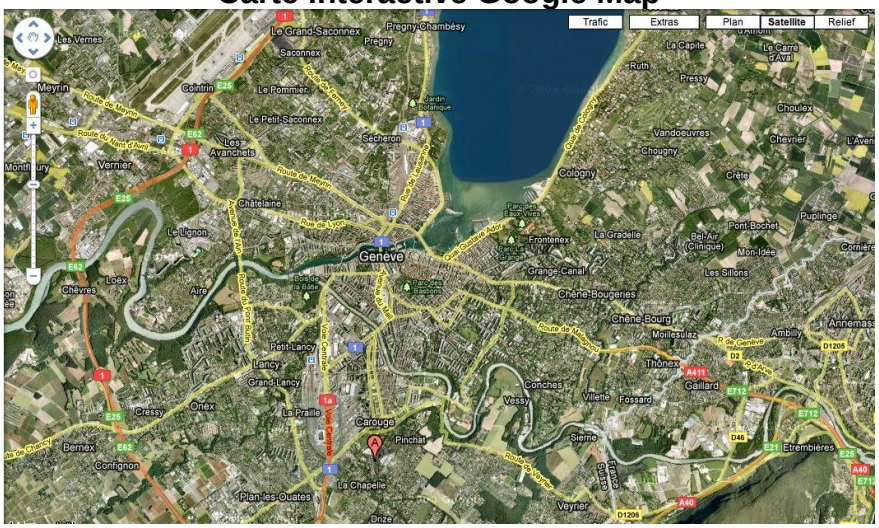
Elles **évitent les erreurs de traitement et les oublis** car les formulaires de saisie sont validés en temps réel et sont soumis lorsque le formulaire est validé.

Elles offrent une **nouvelle manière de réaliser les achats**[9]. Elles éliminent les étapes traditionnelles d'une commande en diminuant les appels au serveur typiquement imposé par l'HTML. Le processus reste le même – remplissage des informations nécessaires pour la commande – sauf que l'envoi des données au serveur se réalise à lors du paiement de la commande. On peut remplir ses informations personnelles et continuer ses achats, puis finir de remplir ses informations personnelles. Cette fluidité de navigation n'est pas possible dans les applications traditionnelles.

2.3 Un exemple avec Google Map et une carte traditionnelle (1bis.com)

Google Map[10] est une application cartographique qui a beaucoup de succès auprès des utilisateurs en raison de sa facilité d'usage. Elle permet lors de la navigation de rafraîchir les zones visitées sans recharger la page.

Figure 4
Carte interactive Google Map



Source : http://maps.google.fr

De nombreuses fonctionnalités sont possibles avec cette application. Il est possible d'avoir une vue satellite, une vue en plan ou encore en relief. De même il est possible de consulter le trafic routier, calculer un itinéraire et y redéfinir les étapes au simple

[9] Par exemple http://www.eticweb.net/FlexEcommerce/FlexEcommerce.html

[10] http://maps.google.ch/

clique de souris. Des photos – ou vidéos, webcams et articles wikipédias – sont aussi disponibles lorsque l'on clique sur une zone de la carte.

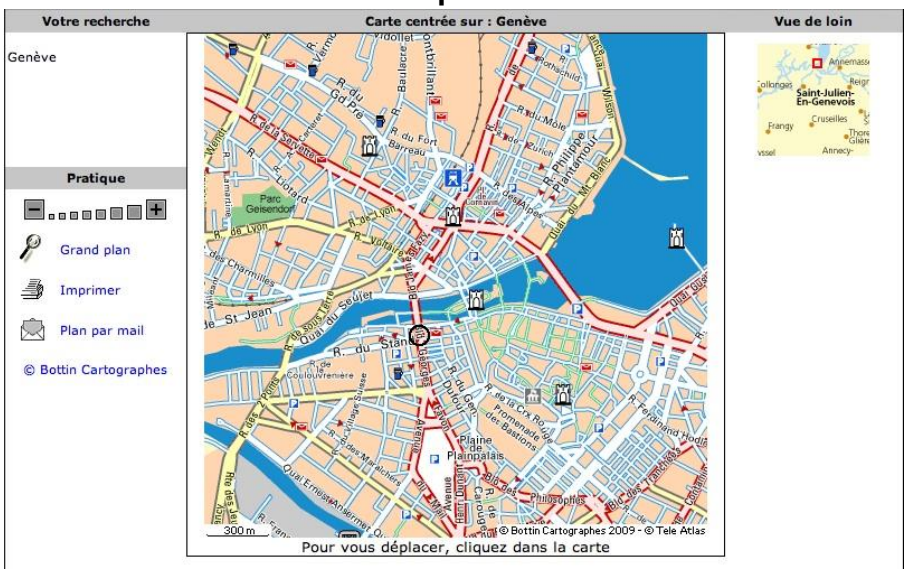
Figure 5
Carte interactive Google Map – photos

Source : http://maps.google.fr

Une application telle que celle-ci offre une interaction fluide et réactive pour les utilisateurs.

Une autre application de cartographie[11], cette fois statique rend son utilisation moins fluide. Il n'est pas possible de se déplacer dans la carte car elle est rechargée entièrement lorsqu'on interagit avec elle, causant un temps d'attente beaucoup plus long et n'offrant pas une forte expérience utilisateur.

Figure 6
Carte statique – 1bis.com



Source : http://www.1bis.com/

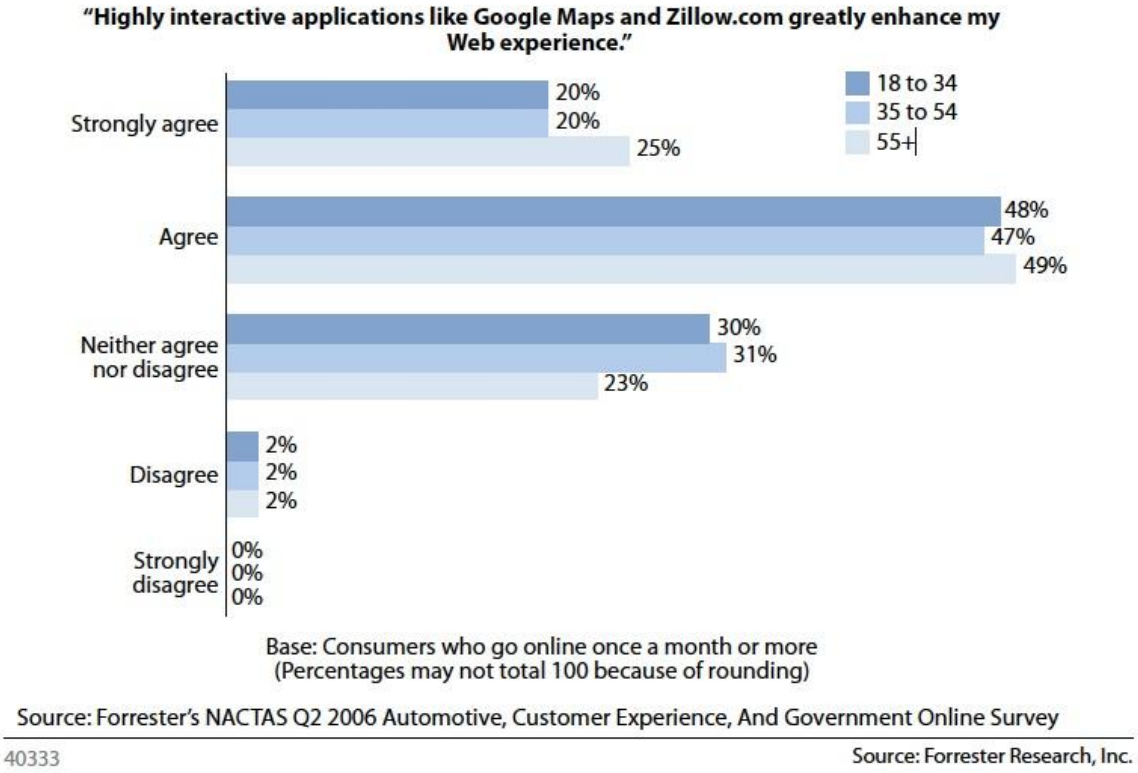
[11] http://www.1bis.com/

La différence entre ces deux applications montre qu'une application riche offre des fonctionnalités puissantes par rapport à une application statique, le tout en offrant une utilisation simple et intuitive.

2.4 Quelques chiffres

L'exemple de la partie précédente démontre que les RIAs permettent de manipuler du contenu sans avoir à rafraîchir entièrement la page, améliorant l'expérience des utilisateurs. Une étude de Forrester[12] fait ressortir que l'utilisation de RIA chez les utilisateurs **améliore leur expérience Web**, du fait qu'ils les trouvent **simple d'utilisation**, et ce toute tranche d'âge confondue.

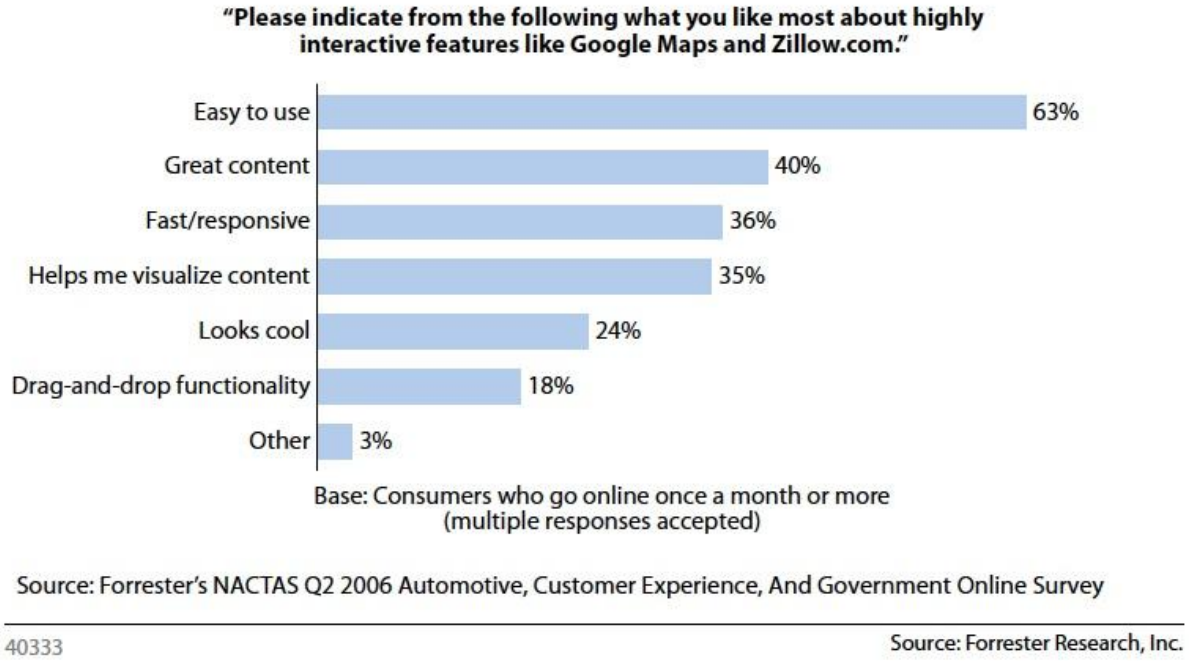
Figure 7
Les RIAs améliorent l'expérience Web



Seulement 2% des personnes interrogées n'admettent pas que les RIAs améliorent leur expérience Web. L'impact des applications riches chez les utilisateurs est important et pour différentes raisons, comme le montre la figure ci-dessous :

[12] Web Users Want Rich Internet Applications

Figure 8
Critères de préférence des RIAs



La majorité des utilisateurs préfèrent les RIAs car elles sont simples d'utilisation, ensuite parce qu'elles affichent du contenu remarquable et parce qu'elles ont un temps de réponse rapide. D'un autre côté, elles aident à visualiser du contenu et sont très ergonomiques lorsqu'elles sont bien construites. Les fonctionnalités de Drag & Drop apportent des interactions fortes avec l'utilisateur, ce qui améliore l'utilisation de ces IHM[13].

2.5 Qu'apportent les RIAs aux entreprises

Pour élargir le mode d'interaction entre les utilisateurs finaux et les applications, les RIAs percent une nouvelle voix dans laquelle les entreprises peuvent ajouter de la valeur à leurs services et leurs produits.

En effet, les RIAs **augmentent la productivité** des utilisateurs. Il n'existe plus d'interface multifenêtres. Elles sont remplacées par une fenêtre unique qui allège et simplifie l'affichage des informations permettant un gain de temps dans les procédures métiers.

Elles permettent ainsi de fournir un feedback aux utilisateurs afin qu'ils puissent directement voir le résultat de leurs actions. Cette manipulation directe permet aussi de

[13] Interface Homme-Machine : interaction d'un utilisateur avec un programme informatique

personnaliser les produits ou les services, ce qui réduit le temps que prend un utilisateur à effectuer une tâche.

Ces applications permettent de **visualiser un large ensemble de données**, ce qui aide les utilisateurs à trouver plus facilement ce qu'ils recherchent. Il est également possible de manipuler les images – illustrations – et de véhiculer plus d'informations pour les voir sous différentes versions - transitions, vidéos, animations – de manière plus riche.

Les applications riches représentent une réelle opportunité pour combler le fossé entre les individus et les applications. Les utilisateurs attendent désormais des applications intuitives proposant une multitude de fonctionnalités avancées.

Les RIAs **remplacent les applications de bureau** car elles possèdent leur robustesse et leur stabilité. Elles possèdent aussi la souplesse du contenu Web car on utilise seulement les informations nécessaires. De plus les informations sont toujours à jour car elles sont situées au même endroit.

Les sites commerciaux

L'achat en ligne est devenu le moyen préféré des consommateurs pour faire leurs achats. De grandes innovations ont été réalisées en vue d'améliorer l'expérience des utilisateurs et d'enrichir l'industrie d'e-commerce. Les revendeurs ont développé des contrôles sur les commandes qui fournissent aux consommateurs un environnement avec une unique fenêtre de contrôle. Cette fenêtre permet de :

- Contrôler la commande et les étapes du processus. Les utilisateurs peuvent rapidement et facilement revenir en arrière et en plein milieu des achats, enregistrer les détails de livraisons ou de paiement, le tout sans changer de page et sans perdre de données ou sans attendre que les données soient rafraîchies par le serveur.
- Calculer rapidement les effets de ventes croisées, vente de gamme supérieure, et les options de transport. Ce qui a pour effet d'encourager le consommateur à explorer l'ajout d'un produit ou de rendre la vente plus rapide.
- Saisir rapidement les données personnelles et les options de paiement. La rapidité des taux de réponses rend le consommateur plus confiant sur la sécurité du site. Cela rend aussi le site plus intéressant et crédible, ce qui gagne la confiance du consommateur.

La fidélité des consommateurs est aussi améliorée, car l'échange des informations est géré de manière plus efficace (gestion des comptes, contrôle des commandes…) Les entreprises peuvent offrir une expérience très sophistiquée aux utilisateurs ce qui accroît la rétention des utilisateurs et qui mène à une augmentation des revenus. De plus, l'entreprise acquiert de nouveaux consommateurs grâce à une première bonne impression. Elle réduit de ce fait ses coûts au niveau des centres d'appels car les consommateurs ont de moins en moins besoin d'assistance téléphonique.

Au niveau des coûts, l'entreprise réduit ses coûts opérationnels au niveau de la bande passante. En effet, les RIAs diminuent l'usage de la bande passante ainsi que les chargements vers les serveurs en déplaçant une partie des processus sur le browser Web.

Une étude de Forrest[14] montre que les entreprises qui mesurent l'impact des RIAs reconnaissent dans la plupart des cas que leur mise en place atteint leurs objectifs, voire plus.

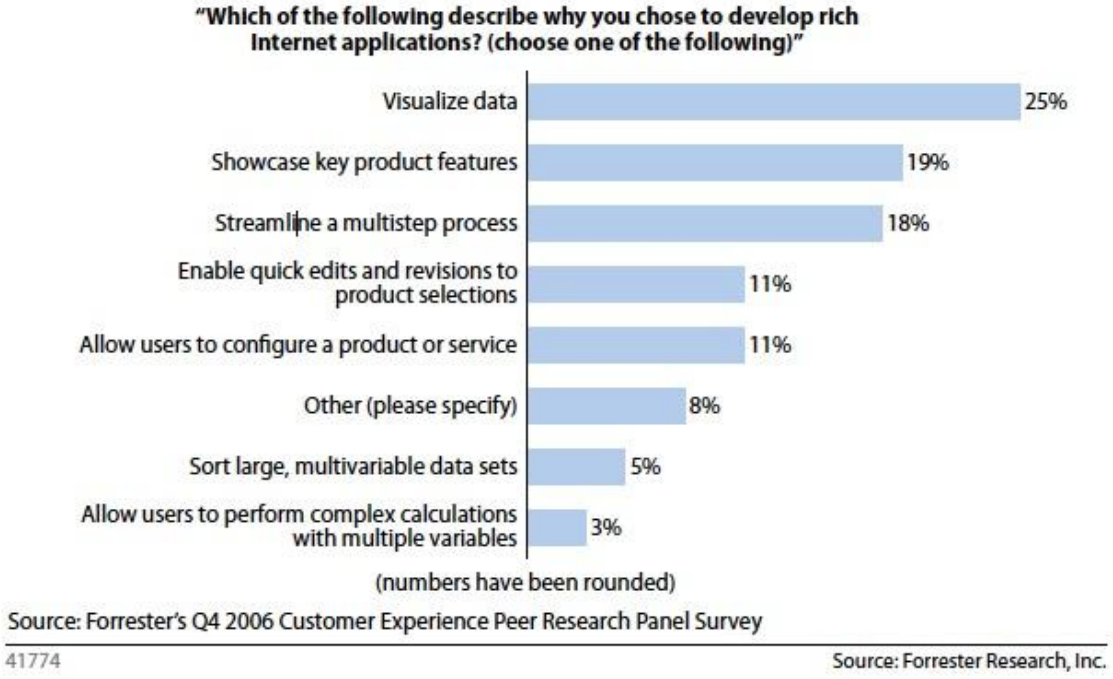
Figure 9
Atteinte des objectifs des RIAs

Les succès des RIAs sont profondément connus et utilisés pour augmenter l'efficacité en ligne.

La même étude montre les différents choix d'une entreprise dans le développement des RIAs.

[14] The Business Case For Rich Internet Applications

Figure 10
Choix de développement de RIA



Les RIAs offrent une alternative à l'HTML et fournissent un meilleur impact aux utilisateurs au niveau des interactions fluides et des traitements. Elles fournissent une large suite de fonctionnalités, principalement la visualisation de données, l'affichage efficient de fonctionnalités, la mise en place efficace de processus multiples.

2.6 Les inconvénients

Différents inconvénients subsistent dans les applications riches, souvent dépendants des technologies utilisées. Nous allons ici traiter les inconvénients généraux des clients riches.

Tout d'abord, les applications riches offrent une nouvelle utilisation du Web et la **navigation est devenue différente** des sites Web dont on a l'habitude de faire usage. L'apprentissage chez les utilisateurs sera plus ou moins long car il faut radicalement changer les mœurs et les habitudes de navigation.

Un problème non-géré nativement par les navigateurs est le **manque d'historique**. En effet, les navigateurs ne sont pas en mesure de connaître l'historique de navigation

dans une application riche. Il est nécessaire qu'il soit développé dans l'application, augmentant le temps de développement. De même il n'est pas possible de faire un **favori** vers une page dans un certain état.

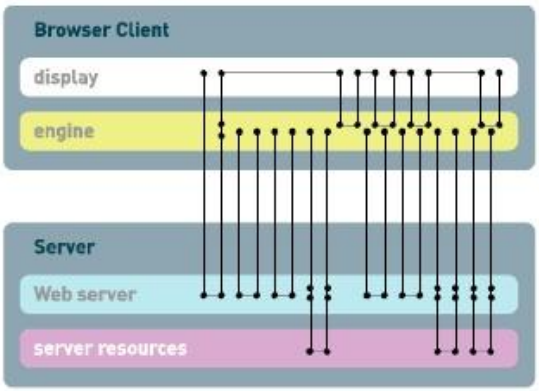
Un autre problème lié aux RIAs est le **référencement** des pages. Les grands moteurs de recherche ne peuvent pas facilement indexer le contenu dynamique, ce qui limite la portée des applications.

Il existe toujours des problèmes de **sécurité**, existants bien avant les RIAs. Les attaques peuvent se cibler sur le serveur, avec les injections SQL, le cross-site scripting (XSS) et le cross-site request forgeries (CSRF) qui sont les attaques les plus communes contre les serveurs. Les attaques ciblées chez le client proviennent souvent du code malveillant ajouté sur la machine et qui est capable de lire les informations transmises entre le client et le serveur. La sécurité varie suivant les Frameworks RIA utilisés.

3. Architecture typique des RIAs

Bien qu'il y ait plusieurs technologies qui rivalisent les unes avec les autres pour devenir standard et implémenter des RIAs, elles ont toutes des caractéristiques communes comme le présente la figure ci-dessus.

Figure 11
Architecture et communication d'une RIA



Source : http://www.keynote.com/

Indépendamment de l'implémentation, toutes les RIAs introduisent dans la partie client un moteur de rendu (« rendering engine ») qui agit comme un médiateur entre le serveur Web et l'utilisateur final. Ce moteur est généralement téléchargé sur la machine cliente au tout début de la session et c'est lui qui interagit avec le serveur Web.

Du côté serveur, la plupart des RIAs possèdent un composant serveur qui fournit au moteur de présentation du client le résultat des invocations faites en aval - il agit comme un broker[15].

Le moteur de rendu permet d'avoir ses processus en mode asynchrone, et ce indépendamment de la communication avec le serveur. Ceci offre donc plusieurs avantages tels que :

- L'information peut être rapportée du serveur et affichée en anticipation de la réponse de l'utilisateur
- Les pages peuvent être mise à jour au fur et à mesure des saisies de l'utilisateur
- Les informations cachées peuvent être utilisées en réponse de la saisie utilisateur, et ce sans communication avec les serveurs.

Nous allons maintenant présenter les différentes technologies qui permettent de créer des RIAs. Elles répondent généralement à la description ci-dessus.

[15] Intermédiaire

4. Les technologies et les outils de développement

De nombreuses technologies existent, relativement jeunes pour la plupart et venant de différents acteurs de l'Internet. Les RIAs représentent un investissement d'avenir car ce marché est en pleine construction. Les principaux acteurs sont Adobe, Microsoft, Sun Microsystem et Mozilla qui ont chacun développé des technologies différentes et non interopérables. De plus en plus d'acteurs se positionnent sur ce marché – Lazslo Systems, Google, AOL …

Parmi les différentes solutions existantes, les choix pour implanter une solution RIA sont nombreux et afin d'évaluer correctement les technologies, il est important de prendre en compte plusieurs critères. Ces critères dépendent des différents acteurs qui sont les utilisateurs, les développeurs et les entreprises.

Les différents critères évalués sont :

- L'expérience utilisateur : quel degré de richesse apportera l'application
- Le prix
- Les performances
- La sécurité
- Le support multimédia – vidéos, audios…
- L'accessibilité depuis d'autres appareils – PDA, smartphones…
- La facilité et la rapidité de mise en œuvre - langage, GUI controls (librairies), complétion du code, IDE – et de maintenance; la maturité du Framework
- Les exigences au niveau de la plate-forme client : installation d'un plugin, support des navigateurs et systèmes d'exploitation
- Le support communautaire, la documentation

4.1 Ajax

Ajax est l'acronyme d'Asynchronous Javascript and XML. Ce n'est pas une technologie à proprement parlé mais un regroupement de technologies utilisées depuis longtemps sur le Web qui sont :

- L'HTML et les feuilles de styles CSS
- Le XML – eXtensible Markup Language. C'est un langage de balisage générique. Il permet de structurer les informations afin de faciliter les échanges à travers Internet.
- Le Javascript. C'est un langage orienté objet destiné à s'exécuter uniquement sur le browser du client.
- L'objet XMLHttpRequest. Il permet de communiquer avec le serveur de manière asynchrone et de récupérer des données.

- L'objet DOM – Document Object Model. C'est une directive créée par le W3C qui fournit une représentation structurée pour la programmation des documents HTML et XML. Elle connecte notamment les pages Web aux scripts Javascript.

La technologie Ajax est simple à mettre en œuvre et est la meilleure solution pour améliorer rapidement le design d'une application Web existante. En effet, il n'implique pas au développeur de changer complètement son code ou ses outils de développement. De plus Ajax procure un impact minimal sur la productivité du développeur si le temps de conception est court.[16]

Une particularité qui fait la force d'Ajax est qu'il est gratuit et il n'est pas nécessaire d'installer de composant pour faire fonctionner l'Ajax. Il suffit d'autoriser le Javascript sur le navigateur.

Les navigateurs tendent à interpréter le langage de la même façon, ce qui n'était pas toujours le cas les années précédentes. Mais néanmoins, le temps de réaction d'une application réalisée en Ajax est limité par rapport à une application Flex.

Le code source est visible depuis le navigateur, ce qui peut poser problème et le polymorphisme n'est pas possible en JavaScript.

Concernant la sécurité, Ajax est soumis aux mêmes problèmes de sécurité cités précédemment dans le point 2.6 – Les Inconvénients.

Les Frameworks

Une large communauté utilise Ajax et de nombreux Frameworks ont vu le jour. Ils fournissent des ensembles de librairies qui optimisent les performances, l'accessibilité, l'intégration, la sécurité, qui sont multi plate-forme – ou cross-browser – et qui se conforment à des normes.

Voici une liste non exhaustive des différents Frameworks Ajax :

- GWT – Google Web Toolkit - : c'est un Framework pour créer des applications « ajaxifiées » en Java.
- Dojo Toolkit : il est écrit entièrement en JavaScript et possède une librairie graphique, des effets de transitions, des outils d'internationalisations (formatage des dates, monnaies…)…Il est

[16] Forrester research : Ajax or Flex : How to Select RIA technologies

également possible de générer une documentation à partir du code source ainsi que de l'analyser, le compresser et l'optimiser.

- MooTool : rassemble des fonctionnalités évoluées telles que le drag-N-Drop, les animations graphiques...
- Prototype : ce Framework permet de bénéficier des fonctionnalités avancées de la POO.
- Jquery : bibliothèque javascript qui facilite le développement de sites Web avec la gestion des interactions asynchrones, des évènements, des animations…

4.2 Adobe

La technologie Flash a été créée par Macromedia en 1996 et permettait de créer des contenus multimédias – dessin vectoriels, Bitmap, audio, vidéo. En décembre 2005, Adobe racheta Macromedia ce qui lui permit de diffuser « Flash Player », un plugin installé sur le navigateur du client. Ce plugin est le plus répandu - 98,8% de pénétration en juin 2009 -, il permettait à l'origine de jouer des animations et des objets interactifs sur une page Web. La majeure partie des navigateurs et systèmes d'exploitation peuvent afficher du Flash. Cette technologie a évolué et il est possible de créer des applications riches interactives pouvant être lues par le plugin Flash.

Il existe deux solutions qui permettent de développer des applications pouvant être interprétées par ce plugin : Flex et Laszlo.

Flex

Flex est une excellente solution pour créer des RIAs et il est possible de créer des applications avec un niveau de complexité plus ou moins grand. Elle possède une certaine maturité par rapport à ses concurrents, et a le soutien d'une large communauté.

Flex 1.0 a été conçu à l'origine par Macromedia comme un produit commercial. Adobe a rendu une partie du Framework open source et sa dernière version commerciale - Flex Builer 3 - fournit tous les outils nécessaires pour développer plus rapidement des RIA grâce à des fonctionnalités avancées – gestions multimédia, données… – ce qui en fait une technologie robuste.

Une application Flex est composée :

- Du langage Actionscript 3 – AS3. Comme JavaScript, l'AS3 est une implémentation d'ECMAScript[17] . C'est un langage orienté objet, qui tend à se rapprocher du langage Java.
- Du MXML – Macromedia eXtensible Markup Language – c'est un langage de balise basé sur le XML. Il permet de décrire les interfaces utilisateur et de contrôler l'aspect d'une application – positionnement des composants, contrôle des évènements…

La combinaison de ces deux langages fournit les outils nécessaires pour construire de robustes applications riches. La documentation et des exemples de mise œuvre sont facilement disponibles sur le Web.

Flex Builder permet aux développeurs d'avoir une expérience WYSIWYG[18]. Flex Builder est en constante évolution – il y a déjà eu 3 version en 5 ans - et la prochaine version de Flex Builder sortira prochainement cette année sous le nom de Flash Builder 4. Il est cependant possible d'ajouter le plugin pour Eclipse.

Flex builder peut fournir deux formats de sortie : soit un swf qui sera exécuté depuis e Web - la navigation de l'application se fera à partir d'un navigateur – ou aussi générer des applications en AIR – installées sur l'ordinateur de l'utilisateur – et qui ne sont pas utilisées par les navigateurs. Les applications AIR offrent donc tous les avantages d'une application de bureau.

Il est seulement nécessaire pour les utilisateurs de posséder le plugin Flash. Les applications créées à partir de Flex fonctionnent avec les versions 9 ou ultérieures du plugin (environ 97% des ordinateurs sont équipés de ces plugins). Ce qui fait une application Flex multi plate-forme.

La plate-forme est facilement extensible et il est possible de l'intégrer du côté serveur avec des technologies Java, PHP, ASP, Ruby…

Au niveau de l'expérience utilisateur, les solutions Flex offrent les mêmes richesses qu'une application de bureau, sans rafraîchissement de page, tous les accès aux données et l'affichage sont réalisés en arrière plan.

---

[17] ECMAScript est le standard international de langage de programmation de script.

[18] What You See Is What You Get

Le modèle de sécurité de Flex protège le client, tout comme le serveur car le player Flash opère dans une sandbox chez le client et l'utilisateur qui accède aux ressources sur le serveur et reçoit une autorisation et une authentification.[19]

4.3 OpenLazslo

Open Lazslo a été développé à l'origine en 2004 par Lazslo Systems, la plate-forme Open Source principale pour construire et déployer des applications Web 2.0. La technologie Open Lazslo est profondément adoptée par les applications et les fournisseurs de services comme les entreprises, les consommateurs, l'éducation et les gouvernements. Lazslo Systems fournit des mises à jour, des tutoriaux et du support de qualité pour Open Lazslo et offre une expérience Web riche.

Cette technologie en est maintenant à sa 4ème version et est sous licence CPL - Common Public Licence. Cette plate-forme fournit tous les composants pour construire des RIAs et supporte le déploiement simultané d'applications en Flash et en DHTML. Cela la rend portable sur tous les browsers, à condition de posséder le plugin Flash si la version compilée de l'application est le FlashPlayer.

OpenLazslo fournit un ensemble de classes et composants réutilisables – grilles, contrôles d'arborescences, fenêtres…. – et inclut des comportements typiques comme le drag-n-drop, des animations, le databinding… Une application est écrite en LZX, un langage XML qui inclut du JavaScript et un plugin pour Eclipse est disponible pour développer des applications WYSIWYG.

OpenLazslo tire profit des nouvelles technologies et de celles existantes. Le problème principal est que toute la logique applicative des applications doit être écrite en JavaScript.

4.4 Microsoft

4.4.1 WPF

Microsoft WPF – Windows Presentation Foundation – est disponible depuis la sortie de Windows Vista et est mise en place à travers le Framework .NET 3. WPF se comporte un peu comme Adobe Flex car cette technologie permet de réaliser des interfaces

[19] http://livedocs.adobe.com/flex/3/html/help.html?content=security2_01.html

utilisant les ressources du Web de manière standalone (sur le desktop) ou dans le navigateur Internet Explorer.

Les applications réalisées avec WPF ne fonctionnent qu'avec Windows comme système d'exploitation. Windows est le système d'exploitation préféré des utilisateurs, il est intégré sur plus de 98% des ordinateurs.

Les applications WPF sont réalisées à l'aide de Microsoft Visual Studio 2005 avec les extensions installées du Framework .NET 3, ou avec Microsoft Visual Studio 2008. Étant basé sur .NET 3, on peut réaliser des applications à l'aide de plusieurs langages : VB .NET, C#... La création est simplifiée avec le mode de développement WYSIWYG.

Pour créer des applications WPF, il faudra utiliser le langage XAML – eXtensible Application Markup Language -, un langage de balises basé sur le XML qui permet de contrôle l'aspect des composants. Une libraire avancée de composants permet de réaliser des applications mixant en toute liberté de la vidéo, de la 3D, des images et des flux issus du Web.

Cette technologie exploite la plate-forme et les services de sécurité de Miscosoft .NET pour sécuriser les applications générées avec WPF[20]. Il est possible de paramétrer la sécurité pour les applications réalisées.

4.4.2 Silverlight

Silverlight est une extension allégée de WPF – de nom original WPF/E. Elle permet à l'aide du Framework .NET 3 de réaliser des RIAs. Le langage utilisé est le même que WPF, c'est-à-dire du XAML conjugué à un langage supporté par le Framework .NET 3. L'environnement de développement Visual Studio est bien connu des développeurs et est très utilisé.

Cette technologie a été mise en place par Microsoft dans le but de faire face à Adobe avec son FlashPlayer. Elle intègre les contenus multimédias de qualité – vidéos, audios… - et permet aussi de visionner des animations vectorielles désormais en 3D.

Silverlight est un plugin qu'on installe sur le poste du client. Aujourd'hui cette technologie toute jeune est peu présente sur les navigateurs – environ 28% de

[20] http://msdn.microsoft.com/fr-fr/library/aa349158.aspx

pénétration[21] - et son déploiement prendra du temps. Silverlight est comme Flash, c'est-à-dire cross-browser et cross-platform. Il peut fonctionner sur la plupart des systèmes d'exploitation et des navigateurs.

Silverlight n'est pas encore mature et ne propose pas une gamme complète de composants, à l'inverse de Flex. De même que les sources et les exemples sur le Web qui sont peu nombreux comparés aux autres technologies.

Silverlight est un concurrent sérieux à Adobe et s'améliore rapidement. La première sortie du plugin date du 05 Septembre 2007, et la version 3 – sortie le 10 Juillet 2009 – a amélioré plusieurs fonctionnalités telles que le databinding, l'ajout de contrôles, le format HD pour la vidéo, et la possibilité de lancer les applications hors du navigateur. De plus, ce plugin prend en charge l'accélération GPU, c'est-à-dire qu'il utilise la carte graphique de l'ordinateur client pour s'occuper du rendu d'éléments graphiques et du décodage des vidéos, ce qui n'est pas encore le cas du plugin Flash.

4.5 Mozilla

La fondation Mozilla a créé XUL – XML User Interface Language – en 1998 pour réaliser des applications riches.

L'inconvénient majeur des applications XUL est qu'elles ne fonctionnent que sous le navigateur Firefox ou ceux basés sur Mozilla – Thunderbird, SeaMonkey -, ce qui représente une pénétration d'environ 18% chez les utilisateurs. Autrement il faut installer un plugin pour Internet Explorer.

Le XUL se base sur l'environnement XulRunner pour s'exécuter. Tous les outils de Mozilla sont Open Source. XUL utilise différentes technologies pour créer des RIA, telles que le JavaScript pour associer des actions aux commandes de l'interface, l'HTML, le CSS, le DOM. Il est possible de créer toutes sortes d'applications, utilisant la vidéo.

D'autres technologies ont été introduites par Mozilla pour compléter XUL :

- Le XBL – eXtensible Binding Language, c'est un langage de balisage qui permet aux développeurs d'étendre XUL et de créer ses propres balises.

[21] Microsoft Silverlight Version Support : http://www.statowl.com/silverlight.php

- XPCOM et XPConnect. Ce sont des technologies qui permettent aux applications XUL d'importer des bibliothèques externes (XPCOM) écrites en différents langages – C, C++, JavaScript, Python, Java, Perl. XPConnect permet l'interaction entre les objets XPCOM et le JavaScript.

Il existe pour le moment deux IDE pour le XUL – XulBooster et Xul Dev – qui fournissent une interface WYSIWYG pour construire des applications en XUL.

Tous les composants XUL sont gratuits et respectent les standards W3C.

4.6 Sun Microsystem

JavaFX

JavaFx est la réponse de Sun Microsystem pour concurrencer les Frameworks RIAs. Elle est apparue tardivement - en 2007. JavaFX 1.0 est sortie le 4 décembre 2008 – et n'est pour le moment pas encore mûre. La version 1.0 ne possède pas de composants graphiques de haut niveau.

Cette technologie offre plusieurs possibilités au niveau de la portabilité : il est possible de créer des RIAs, des applications de bureau ou encore des applications portables sur téléphones.

JavaFX nécessite la *JRE* – Java Runtime Environment - pour fonctionner. Si c'est une application basée sur un navigateur, l'application sera déployée sous forme d'applet qui sera téléchargé sur la machine du client et lue à l'aide de la *JVM* – Java Virtual Machine. LA JRE inclut la JVM et est présente sur la plupart des machines - plus de 80% des ordinateurs, et une large partie des téléphones portables. Cela rend les applications multi plates-formes.

Les applications sont codées avec JavaFX Script. C'est un langage de script déclaratif – comme le SQL - construit sur la plate-forme de développement Java. Il génère des interfaces utilisateur à partir de Java Swing.

JavaFX est sous licence GPL. Pour coder des applications JavaFX, Sun propose un plugin à installer sur son IDE Netbeans, il sera bientôt open source pour le porter sur différents IDE.

Java est le langage de programmation le plus populaire dans les entreprises d'IT. Sun espère obtenir le soutien de la communauté Java pour implanter JavaFX.

4.7 Récapitulatif

Tableau 2

Récapitulatif pénétration technologies

	Windows	**Mac**	**Autres**
Pénétration systèmes d'exploitation	**96,5%**	**3,3%**	**0,2%**
BROWSER - pénétration			
Internet – 78%	**Oui**	**Oui**	
Firefox – 18,2%	**Oui**	**Oui**	
Chrome – 2%	**Oui**	**Non (version béta)**	
Safari – 1,4%	**Oui**	**Oui**	
Autres – 0,4%			

Tableau 3

Récapitulatif des technologies

	Flex	**Silverlight**	**JavaFX**	**XUL**	**OpenLazslo**	**Ajax**	**WPF**
Vendeur	Adobe	Microsoft	Sun Microsystem	Mozilla	Lazslo Systems		Microsoft
Support communautaire	oui - importante	oui - peu importante	oui - très peu	oui	oui	oui - importante	oui
Open Source	Oui : SDK	Non	Oui	Oui	Oui	Oui	Non
Moteur d'exécution	Flash Player	Silverlight Player	JRE	Xul Runtime	Flash Player	Aucun	.NET 3
Pénétration	98%	20%	98%	18%	98%		50%
Syntaxe	MXML + ActionScript	XAML + .NET	JavaFX Script	XUL+ JavaScript	LZX + JavaScript	Ajax	XAML + .NET
Présent depuis	2001	2007	2007	1998	2002	2004	
IDE	Flex Builder ou plugin	Visual Studio	Plugin	Plugin	Plugin		Visual Studio
Expérience utilisateur	Tres riche	Riche	Riche	Riche	Très Riche	Riche	Riche
Support Audio/Video	Excellent	Excellent	Bien	Moyen	Bon	Pauvre	Bien
Librairies	Avancée	Avancée	Peu mature	Avancée	Avancée	Avancée avec un Framework	Avancée
Performances	Très bonnes	Très bonnes	Bonnes	Bonnes	Très bonnes	Bonnes	Très bonnes

5. Présentation de l'application mise en place

5.1 Choix des technologies

Après avoir présenté les différentes technologies existantes pour créer des applications de type client riche, il est temps de présenter l'application réalisée. Le but consistait à vérifier si la théorie et les résultats présentés précédemment correspondent avec une application réalisée.

Le choix des technologies s'est basé sur Adobe Flex et OpenLaszlo. Il ne m'a pas été possible de réaliser l'application avec d'autres technologies supplémentaires par manque de temps et d'apprentissage. Il m'aurait été très intéressant de réaliser l'application avec Silverlight car cette technologie encore récente semble très appropriée pour ce type d'application.

Flex est la technologie la plus mature, elle possède une large communauté de développeurs et de nombreux exemples et forums existent sur le Web. Le choix de cette technologie me paraissait évident.

OpenLaszlo est une technologie très intéressante car elle permet de générer des applications en DHTML ou en Flash. Je voulais voir le comportement de l'application pour les deux types de déploiement.

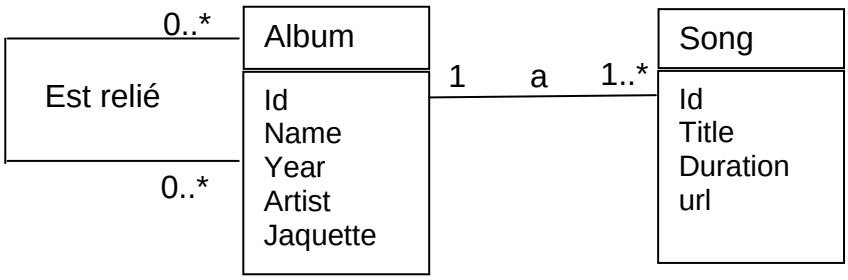
5.2 Objectifs et cahier des charges

Avant de décrire l'application et ses fonctionnalités, le mandat consistait à réaliser une application simple mettant en avant le client riche et entre autres son côté asynchrone. Il sera question de traiter pas à pas du point de vue développeur la mise en place des technologies, de l'installation au déploiement de l'application. Ensuite une présentation de l'application elle-même.

L'application est une bibliothèque musicale. Elle permet de gérer des fichiers musicaux depuis une base de données. Il est possible de :

- Rechercher avec auto-complétion des chansons à partir d'un champ texte.
- Consulter les titres musicaux à partir de listes de genres, artistes et albums. Ces listes se mettent à jour lorsqu'un élément d'une liste est sélectionné.
- Afficher une liste de chansons et ses détails depuis la recherche ou la sélection sur les listes.
- Jouer les mp3.

Figure 12
MCD de l'application



Un album possède comme informations un nom, une année, un artiste et une jaquette. Un album possède une ou plusieurs chansons, dont chacune possède un titre, une durée et l'url du titre mp3. Un album peut être lié à zéro ou plusieurs autres albums du même genre.

5.3 Flex

5.3.1 Installation

Il est possible de réaliser des applications Flex avec Flex Builder ou avec un plugin Eclipse. J'ai réalisé l'application avec Flex Builder. L'installation se fait à l'aide d'un fichier exécutable qui installe l'IDE et tous les composants nécessaires au développement d'applications.

J'ai aussi installé amfphp pour permettre la communication entre la base de données mysql et le langage ActionScript utilisé dans le projet Flex. Amfphp est open-source et permet d'échanger des données entre client et serveur sous la forme de libraire PHP.

L'installation d'amfphp est très simple car il suffit de télécharger une bibliothèque et de l'installer par copier-coller sur le serveur. Il suffit ensuite de rajouter dans le projet Flex un fichier de configuration XML qui spécifie l'adresse du serveur. La communication entre Flex et le serveur est aisée : on dispose sur le serveur d'une classe php contenant les différents services et méthodes nécessaires. On appelle depuis Flex le nom du service qui renvoie un tableau avec les résultats.

5.3.2 Réalisation

L'application se compose de trois fichiers mxml :

- le premier ClientRiche.mxml est le fichier principal de l'application. C'est lui qui gère l'ensemble des composants
- Un deuxième SongRenderer.mxml qui est un composant générique qui gère les fichiers musicaux et leur affichage.

- Enfin AutoCompleteSample.mxml qui s'occupe de la recherche avec auto-complétion.[22] Sa mise en place est très rapide, il suffit lors de l'initialisation de l'application fournir un tableau avec les mots qui seront auto-complétés. Ici, le tableau contient les noms des genres, albums, artistes et chansons.

Il y a également un dossier com qui contient deux sous-dossiers représentants respectivement :

- Un dossier adobe qui contient les classes AS3 qui gèrent l'auto-complétion.
- Un dossier busfront qui contient la bibliothèque qui gère les fichiers musicaux. C'est un Framework open-source avec un ensemble de classes AS3 qui s'occupent de medias. Le fonctionnement du Framework est disponible à l'adresse http://www.busfront.com/EPP

Enfin il y a un fichier xml qui fait la passerelle entre Flex et amfphp sur le serveur.

5.3.3 Description des fichiers mxml

Le langage mxml permet de produire une interface graphique à partir de balises XML spécifiques.

5.3.3.1 ClientRiche.mxml

Voici le code mxml du fichier principal de l'application.

Figure 13
Balises MXML du fichier principal de l'application Flex

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" xmlns:comp="com.adobe.flex.extras.controls.*"
     layout="absolute" addedToStage="init()" height="697" width="1191" backgroundColor="#869ca7">

    <mx:RemoteObject id="serviceAmf" fault="faultHandler(event)" source="library" destination="amfphp">
        <mx:method name="getAlbums" result="askInitAlbumsHandler(event)" fault="faultHandler(event)"/>
        <mx:method name="getSongs" result="askInitSongsHandler(event)" fault="faultHandler(event)"/>
        <mx:method name="getArtistes" result="askInitArtistesHandler(event)" fault="faultHandler(event)"/>
        <mx:method name="getGenres" result="askInitGenresHandler(event)" fault="faultHandler(event)"/>
        <mx:method name="selectGenresArtist" result="askArtistesHandler(event)" fault="faultHandler(event)"/>
        <mx:method name="selectGenresAlbum" result="askAlbumsHandler(event)" fault="faultHandler(event)"/>
        <mx:method name="selectArtistAlbum" result="askAlbumsHandler(event)" fault="faultHandler(event)"/>
        <mx:method name="rechercheInput" result="askRechercheInputHandler(event)" fault="faultHandler(event)"/>
        <mx:method name="getSongFromLists" result="askSongsFromListsHandler(event)" fault="faultHandler(event)"/>
    </mx:RemoteObject>
    <mx:Label x="43" y="48" text="Recherche :" width="80" height="20" id="lblRech"/>
    <mx:List x="43" y="91" width="271" height="143" id="lstGenre" itemClick="selectGenre()" />
    <mx:List x="313" y="91" width="296" height="143" id="lstArtist" itemClick="selectArtist()" />
    <mx:List x="608" y="91" width="288" height="143" id="lstAlbum" itemClick="selectAlbum()" />
    <comp:AutoComplete id="inputRech"
        width="246"
        dataProvider="{rechercheString}"
        labelField="name"  x="131" y="46"/>
    <mx:Button x="385" y="46" label="ok" id="btnRech" click="btnRecherche()"/>
    <mx:Panel x="43" y="262" width="846" height="375" layout="absolute">
        <mx:Accordion width="800" id="adgSong" color="#0B333C" horizontalCenter="-13" y="0" click="accordionClick()" />
    </mx:Panel>
</mx:Application>
```

[22] Ce modèle est disponible à l'adresse : http://www.websector.de/blog/2008/04/30/quick-tip-avoid-issues-using-adobes-autocomplete-input-component-using-flex-3/

La balise `<mx:Application>` est la base du fichier mxml.

La balise `<mx:RemoteObject>` contient des sous balises `<mx:method>` qui sont les services que l'on demande au serveur. On nomme chaque service et on défini les méthodes result lorsqu'on reçoit les résultats. Il y a aussi une méthode qui doit gérer une erreur lors de l'invocation d'un service.

On a ensuite trois balises `<mx:List>` qui sont les listes de genres, artistes et albums. Chacune des listes appelle lors d'un clique un service spécifique qui mettra à jour l'application – mise à jours des listes, affichage des chansons… C'est lors de la sélection des genres, des artistes ou des albums que la communication asynchrone se réalise. Elle se fait de manière transparente aux yeux de l'utilisateur qui peut continuer d'utiliser l'application.

La balise `<comp:AutoComplete>` est le composant qui va servir à l'auto-complétion. Il ne commence pas par mx car c'est un composant qui n'est pas de base avec Flex. Il a été réalisé pour des besoins spécifiques. C'est à ce composant que l'on fournit un tableau avec les mots que l'on souhaite utiliser pour la complétion de la recherche.

La balise `<mx:Accordion>` sert à afficher les chansons sous forme de liste. Le changement de chansons fera défiler la liste de la même manière qu'un accordéon.

Le reste des balises est basique car il s'agit de Label, de Panel et de bouton.

Les fichiers mxml peuvent aussi contenir du code ActionScript. Le fichier ClientRiche.mxml contient la majorité de la logique applicative. Il contient les différentes méthodes appelées lors de l'interaction avec l'utilisateur. Il suffit d'ajouter les balises `<mx:Script><![CDATA[ ]]></mx:Script>` qui contiennent les différentes méthodes nécessaires au fonctionnement de l'application.

5.3.3.2 SongRenderer.mxml

Ce fichier fonctionne comme un Renderer. Un Renderer sert à donner un rendu spécifique : image, label, texte…. Ils sont plus simples à maintenir car ils fonctionnent comme un composant générique.

Ce fichier se compose des éléments nécessaires à l'affichage des informations d'une chanson. On y retrouve le nom de la chanson, son album, son artiste, son genre, sa durée, sa jaquette et son player.

Lorsque l'on affiche une liste de chansons, le fichier principal ClientRiche.mxml va construire un SongRenderer pour chaque chanson avec ses données spécifiques. Tout le traitement lié à la lecture, le téléchargement de la chanson va se trouver dans ce fichier. C'est ce fichier qui va utiliser la librairie busfront et qui va utiliser le Framework de gestion des médias.

Il se compose donc comme le fichier ClientRiche.mxml, c'est-à-dire avec des balises XML et du code ActionScript.

5.3.3.3 AutoCompleteSample.mxml

Ce fichier gère l'auto-complétion. Il va utiliser la bibliothèque adobe qui se trouve dans le dossier com pour gérer l'auto-complétion de la barre de texte construite dans ClientRiche.mxml.

5.3.4 Déploiement de l'application

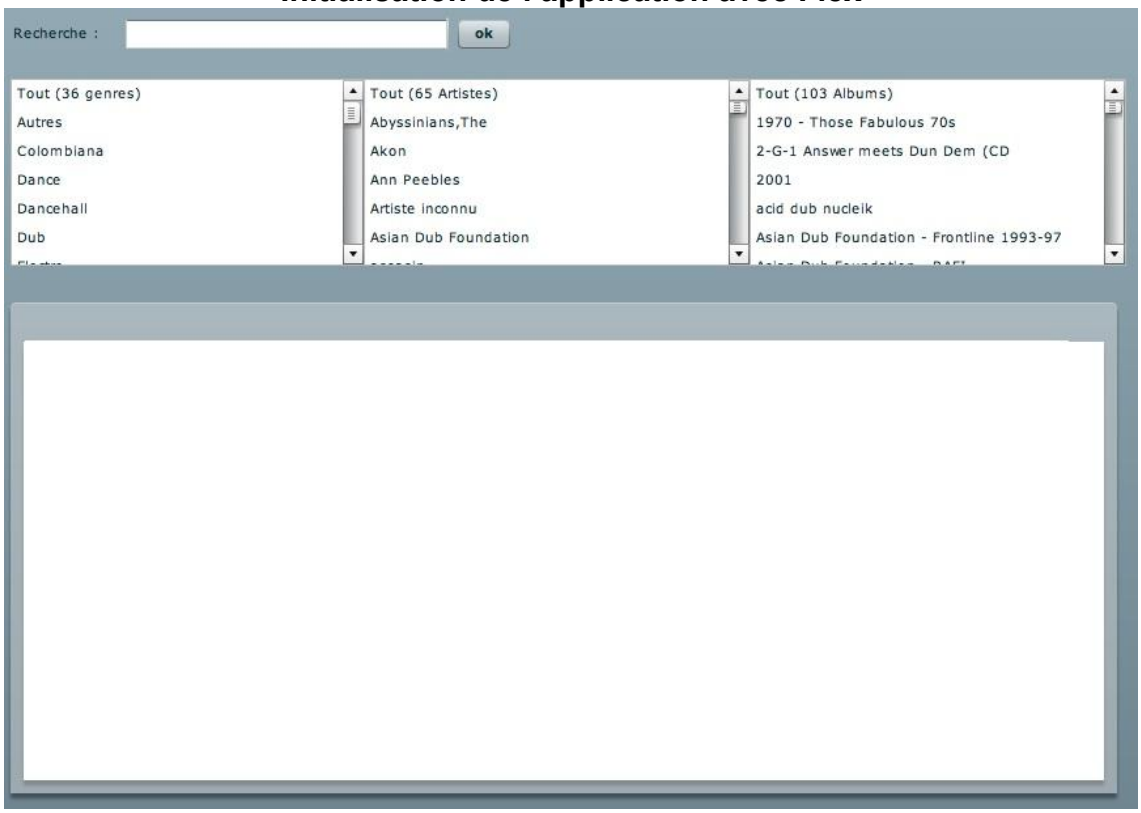
Une fois que l'application est finie d'être codée, Flex Builder nous génère un fichier swf compilé de l'application. Il suffit ensuite de mettre le fichier sur le serveur ou sera accessible l'application.

5.3.5 Application

À l'aide de ces trois fichiers mxml, la réalisation de l'application à été simple et rapide. Il n'y a pas de gros traitements qui ont été effectués sur la base de données, seulement de la consultation - SELECT.

Voici différentes captures d'écran qui montre le fonctionnement de l'application :

Figure 14
Initialisation de l'application avec Flex



Lors de l'initialisation de l'application, les listes de genres, artistes et albums se chargent en interrogeant la base de données grâce aux différents services.

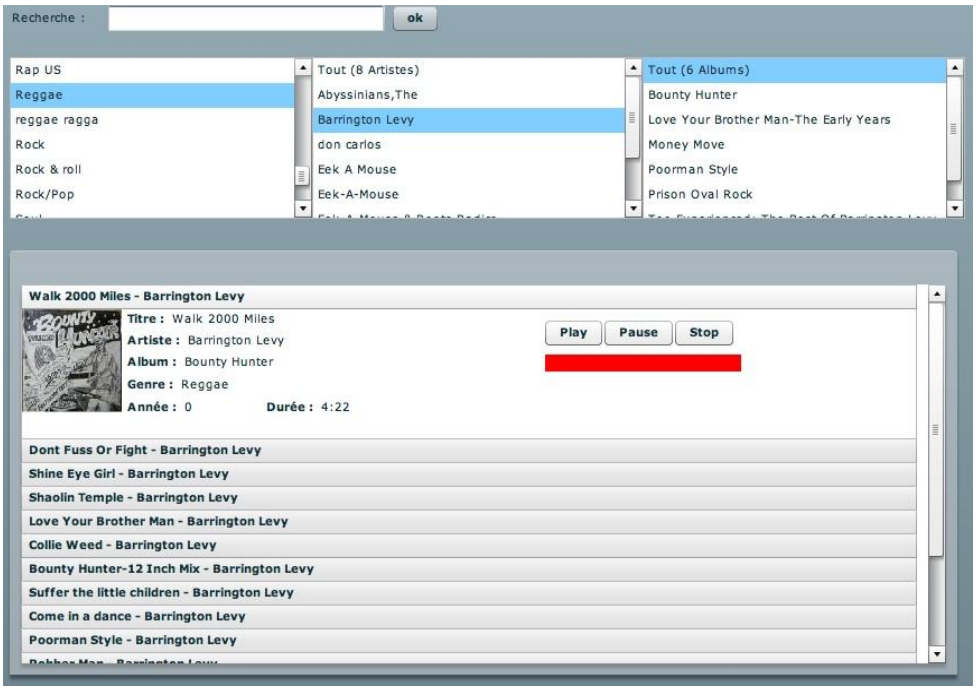
Figure 15
Sélection d'un genre avec Flex

Lorsqu'on sélectionne un genre dans la liste des genres, cela a pour effet d'appeler les services sur le serveur de manière asynchrone. On met à jour presque instantanément les listes des artistes et des albums qui correspondent au genre sélectionné.

Figure 16
Sélection d'un artiste avec Flex

De la même manière que les genres, lorsqu'on sélectionne un artiste on met à jour la liste des albums.

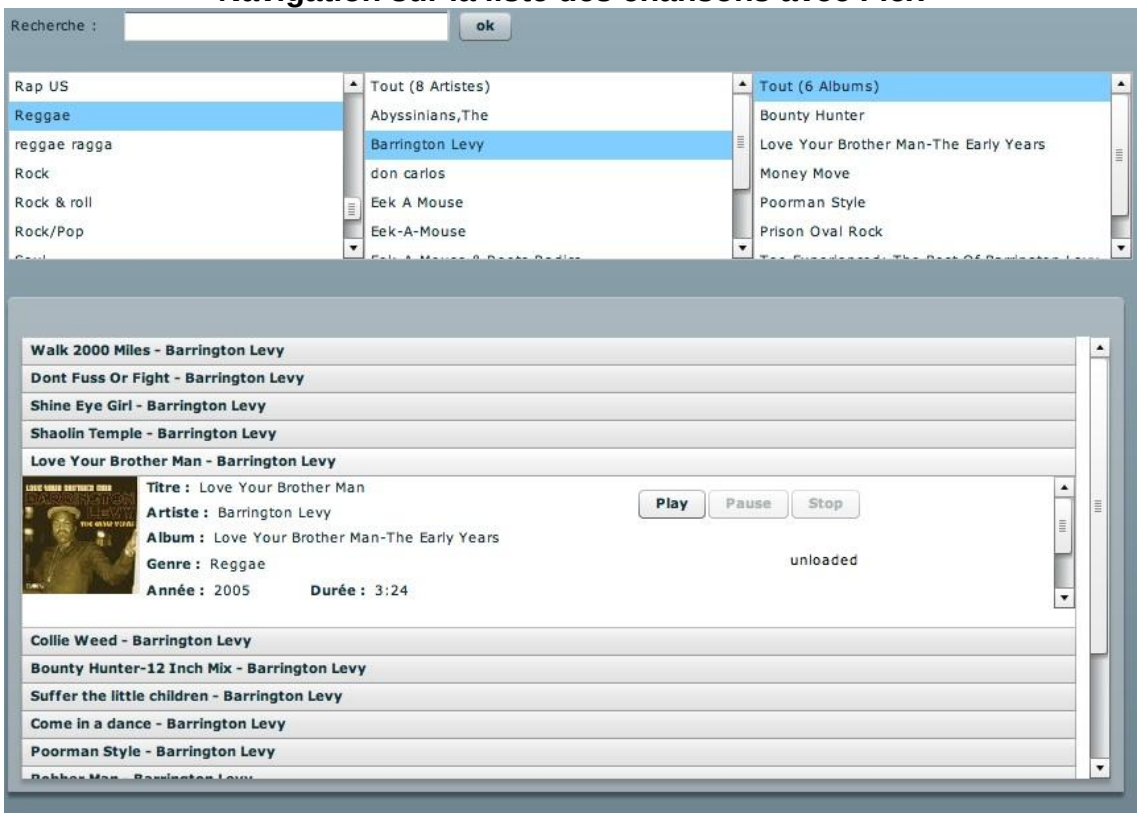
Figure 17
Sélection d'un album avec Flex



La sélection d'un album dans la liste va mettre à jour la liste « accordéon » des chansons. Elle va afficher toutes les chansons qui correspondent aux critères des trois listes du dessus.

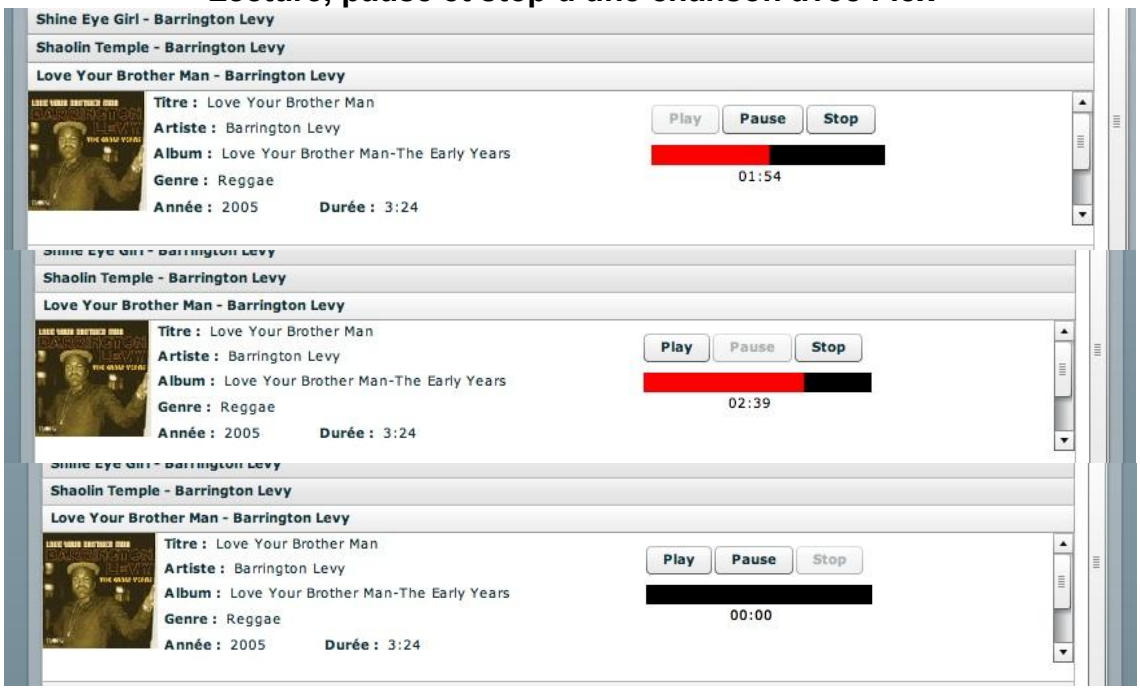
Ensuite la navigation est intuitive car on peut immédiatement voir l'effet d'un clique sur une chanson de la liste. Les informations de la première chanson disparaissent avec une mini animation tandis que les informations de la chanson sélectionnée apparaissent avec la même mini animation.

Figure 18
Navigation sur la liste des chansons avec Flex



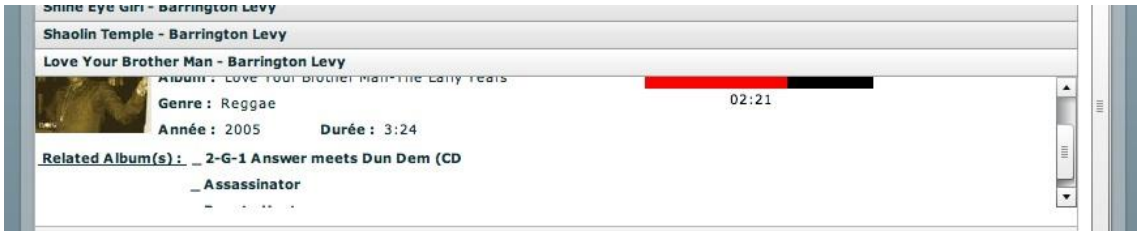
On distingue que le player est prêt et nous donne l'état de la chanson, on peut lancer la lecture. Le Framework qui gère les mp3 est puissant et offre au développeur une gestion simple des médias. Il n'est pas rare de trouver des bibliothèques AS3 pour Flex (payantes ou open source) qui offrent d'énormes possibilités de traitements pour tout types d'applications – statistiques, gestion et retouches d'images, traitements de données… - et qui simplifient grandement le travail du développeur.

Figure 19
Lecture, pause et stop d'une chanson avec Flex



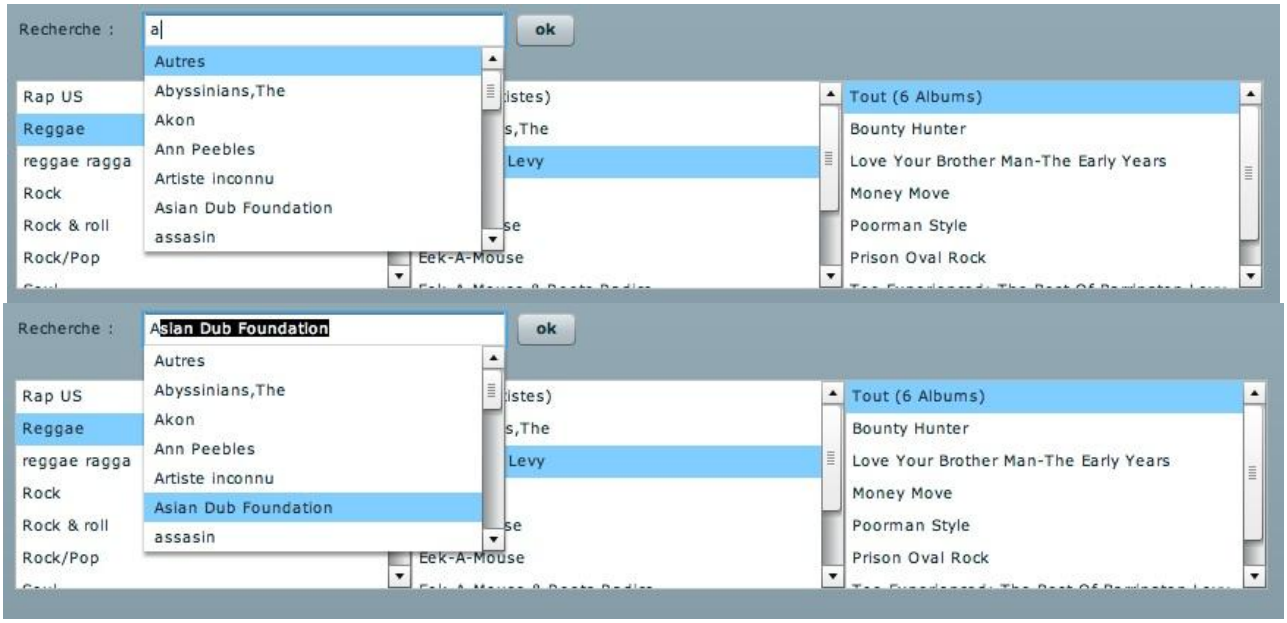
Lors de la lecture d'un titre mp3, des contrôles sont effectués sur les boutons ce qui permet à l'utilisateur d'éviter de faire des erreurs.

Figure 20
Albums reliés avec Flex



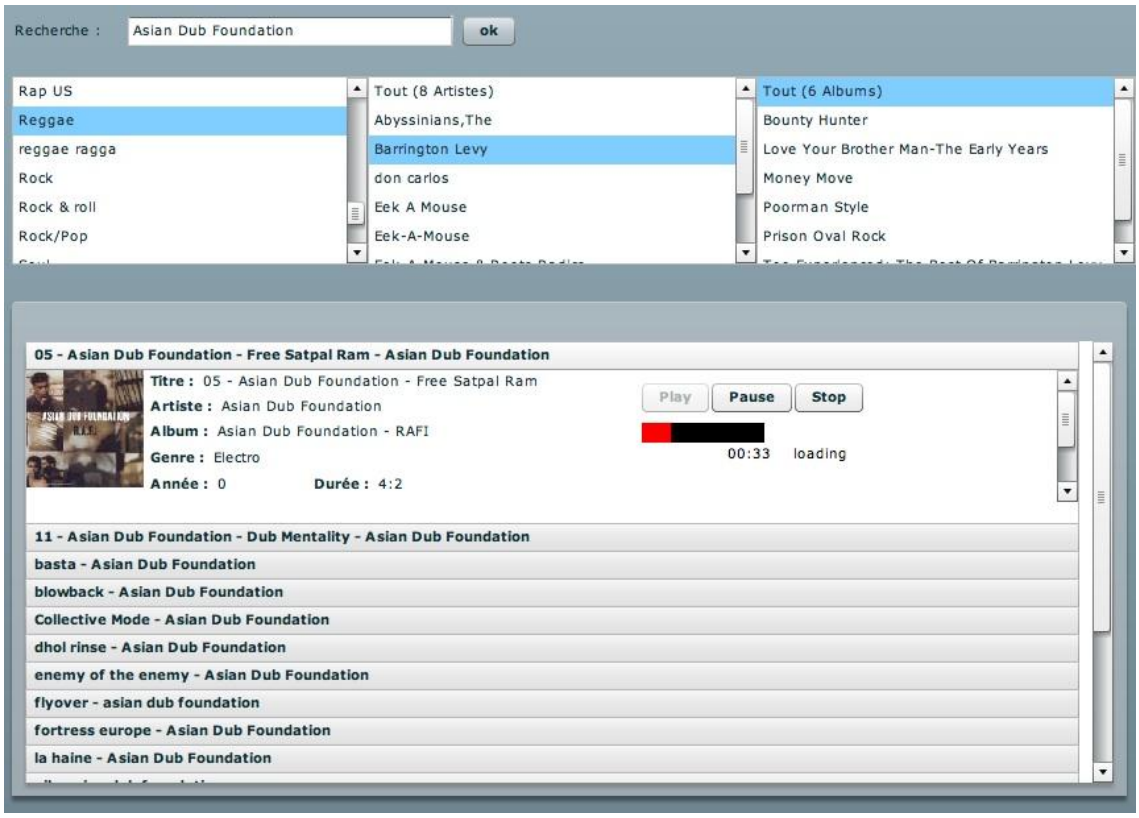
Lors de l'affichage des informations d'une chanson, une liste apparaît pour afficher un ensemble d'albums qui semble similaire à la chanson jouée.

Figure 21
Recherche et auto-complétion avec Flex



La recherche de chansons grâce au champ texte est aisée. Une liste de mots apparaît lors de la saisie de l'utilisateur et on peut sélectionner dans la liste générée une recherche prédéfinie.

Figure 22
Résultat d'une recherche avec Flex



Le résultat de la recherche s'affiche dans la liste « accordéon » et la navigation fonctionne de la même manière que celle définie précédemment.

5.3.6 Bilan

Adobe Flex est un outil puissant qui fournit tous les éléments nécessaires pour réaliser des applications de type client riche. Le fonctionnement des applications en Flex est fortement basé sur les évènements et il peut être difficile au premier abord de savoir que plusieurs évènements sont nécessaires pour gérer un appel sur un serveur. Cependant la documentation est complète, la communauté est présente et de nombreuses sources simples d'utilisation se trouvent facilement sur le Web. Cela offre au développeur une diminution de charge de travail dans le sens ou de nombreux cas ont déjà été réalisés. Il devient rare de devoir partir de zéro.

La prise en main du langage ActionScript nécessite tout de même un temps d'apprentissage non négligeable qu'il faut prendre en compte. Cela peut rebuter une partie des développeurs Web qui utilisent des outils plus simples comme des CMS. Cependant ce langage offre de nombreuses possibilités qui sont très proches du Java.

Tableau 4
Points positifs et négatifs de Flex

Points positifs	Point négatifs
Installation	Temps d'apprentissage
Documentation – communauté	
Maintenance	
Déploiement	

5.4 OpenLaszlo

5.4.1 Installation

OpenLaszlo existe officiellement depuis 2002. Il n'existe pas d'installation simple pour OpenLaszlo. En effet, le projet OpenLaszlo a eu des périodes où il a été abandonné, ne laissant pas beaucoup de supports. J'ai testé l'installation sur Macintosh. Il faut d'abord télécharger et installer OpenLaszlo Server pour que les applications fonctionnent. Pour ce qui est de l'IDE, il faut aussi télécharger un plugin. Il faut plusieurs prérequis : posséder Eclipse 3.2 et non une version supérieure, avoir installé préalablement le package WTP – Web Tool Platform. Ensuite il faut trouver le bon

plugin OpenLaszlo car celui-ci change souvent de location.[23] Une fois que le plugin est installé dans Eclipse, on peut créer des projets Laszlo. Seulement, il ne m'a pas été possible d'utiliser le mode Design et de construire l'application avec les composants par glisser-déplacer. Cela n'a pas été très gênant lors du développement de l'application. Il faut aussi configurer OpenLaszlo Server et lui indiquer la JAVA_HOME installée sur l'ordinateur.

Même si l'installation d'OpenLaszlo n'est pas évidente, il n'est plus nécessaire d'utiliser d'autres outils pour par exemple communiquer avec le serveur et accéder à la base de données. Il n'est cependant pas possible d'utiliser des classes PHP pour obtenir les informations de la base de données. La gestion des données externes se réalise grâce à des fichiers XML. Le fichier PHP doit afficher les données comme si c'était du XML. Afin d'éviter d'avoir plusieurs conditions dans le fichier php et de traiter les multiples POST, j'ai créé un fichier php par service : un pour l'affichage des listes de genres, artistes et albums, un pour l'affichage des chansons dans la liste « accordéon » et un pour la recherche.

5.4.2 Réalisation

L'application se compose d'un seul fichier LZX. Comme le MXML, LZX permet grâce à des balises XML de produire une interface graphique. Chaque technologie nomme les balises XML à leur manière, bien quelles produisent le même effet. Il n'y a pas de standardisation sur le nommage des balises. Les autres technologies qui fonctionnent grâce à des balises XML – qui n'ont pas été testées dans ce mémoire - ont elles aussi un nommage particulier. Pour chaque technologie, le temps d'apprentissage est important, mais en général – pour les technologies que j'ai testé - il est relativement rapide.

Pour OpenLaszlo, Ii n'a pas été aussi facile d'obtenir le même résultat que l'application en Flex. Par exemple, lorsqu'on a une liste avec beaucoup d'enregistrements et que sa largeur est déjà définie, on pourrait penser qu'une scrollbar s'ajouterait automatiquement - comme dans Flex. Et bien non, il faut l'ajouter à la main. C'est autant de petits détails qui font que le début de l'utilisation d'OpenLaszlo paraît difficile. Mais à force d'utilisation et de manipulations, on comprend son fonctionnement qui est finalement logique. De même, les composants suivent une hiérarchie au sein de l'application comme des répertoires et des dossiers. Il était au début difficile de cibler

[23] On trouve sur le site http://laszlodev.fr un bon tutoriel pour la mise en place d'applications Laszlo

d'autres composant qui n'étaient pas au même niveau que celui où on se trouvait. On ne peut pas cibler directement un composant ou commencer le ciblage depuis la racine. Par exemple un élément d'une liste de genre se trouve au 4ème niveau depuis la racine, alors qu'on pourrait penser qu'il se trouve au deuxième ou troisième niveau.

La recherche avec auto-complétion ne fonctionne pas. Il existe dans la bibliothèque d'OpenLaszlo un composant avec auto-complétion, il s'agit d'une liste box. Malheureusement, ce composant ne trie pas automatiquement les données récupérées et va chercher les résultats qui se trouvent après le premier élément trouvé même s'ils ne correspondent pas avec la recherche. Par exemple, ce composant a un dataset avec – 1) Alpha, 2) Artiste Inconnu, 3) Beat, lorsqu'on tape dans la liste Ar, la liste qu'il va afficher sera 1) Artiste Inconnu et 2) Beat. Ce composant n'était pas adapté pour notre cas, et les différentes recherches effectuées sur les forums et sites communautaires ne mentionnent pas d'autres composants ou de corrections pour ce cas particulier.

La communauté autour de cet outil n'est pas très grande et les sites et forums ne sont pas nombreux. Néanmoins on trouve sur le site d'OpenLaszo[24] des tutoriaux qui permettent rapidement de le prendre en main. Le forum lui aussi présente beaucoup de solutions efficaces pour des problèmes souvent rencontrés. Il m'a été d'une grande aide lors du développement.

Un des gros défaut d'OpenLaszlo est la gestion des évènements. Ceux-ci ne sont pas bien gérés : lorsque l'on veut remplir et mettre à jour une liste avec des données provenant d'un fichier XML, on ne peut pas savoir si l'application a bien chargé le fichier ou s'il y a eu une erreur. La communication est transparente, elle est faite automatiquement et il n'y a pas de moyen de contrôle. De plus, quand on défini un dataset qui va récupérer un fichier XML, il n'y a pas de moyen de savoir si la requête a été lancée et si elle va aboutir à un résultat qui va s'afficher.

Certains caractères spéciaux ne sont pas gérés par OpenLaszlo. Par exemple, le caractère « & » ne peut pas être affiché dans l'application. Il n'est pas pris en charge, ce qui a provoqué des problèmes lorsqu'on affichait dans les listes des genres, artistes ou albums une ligne qui possédait ce caractère. Rien ne s'affichait. Il faut donc, lorsque l'on génère le fichier XML remplacer ce caractère par un autre, ou par une suite de caractères et effectuer un traitement en plus.

---

[24] http://www.openlaszlo.org/lps/docs/guide/

Le code de l'application est faible et compte environ 150 lignes (Flex en compte plus du double). Il est tout de même relativement simple de réaliser une application efficace avec peu de code.

5.4.3 Déploiement de l'application

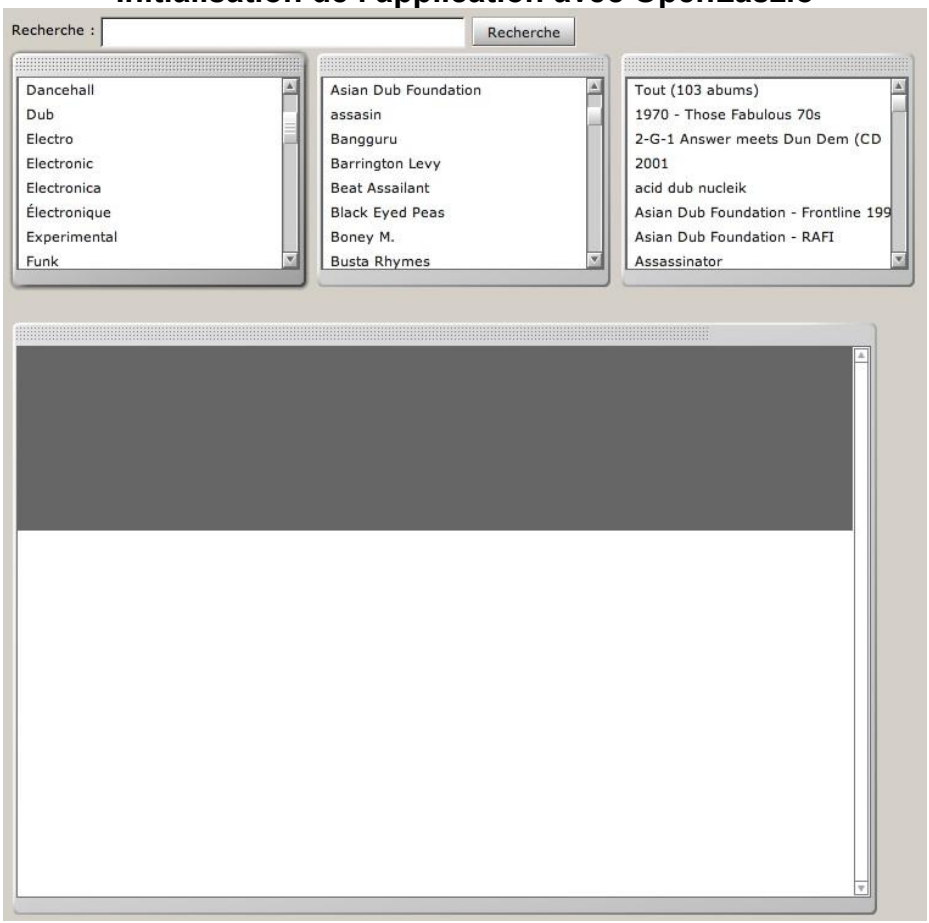
OpenLaszlo permet le déploiement d'application en Flash ou en DHTML. Avec Flash, un fichier SWF est automatiquement généré et on peut le déployer sur un serveur ou sera accessible l'application.

Quant au DHTML, l'application ne fonctionne pas pour lire les titres musicaux. Il n'a donc pas été nécessaire de générer l'application car il n'y a pas de solution pour lire les médias en DHTML avec OpenLaszlo.

5.4.4 Application

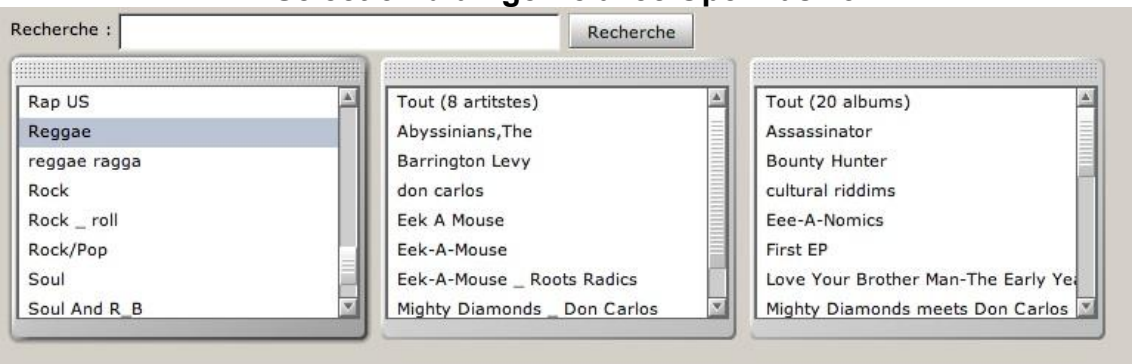
Voici les différentes captures d'écran qui montrent le fonctionnement de l'application avec le plugin Flash.

Figure 23
Initialisation de l'application avec OpenLaszlo



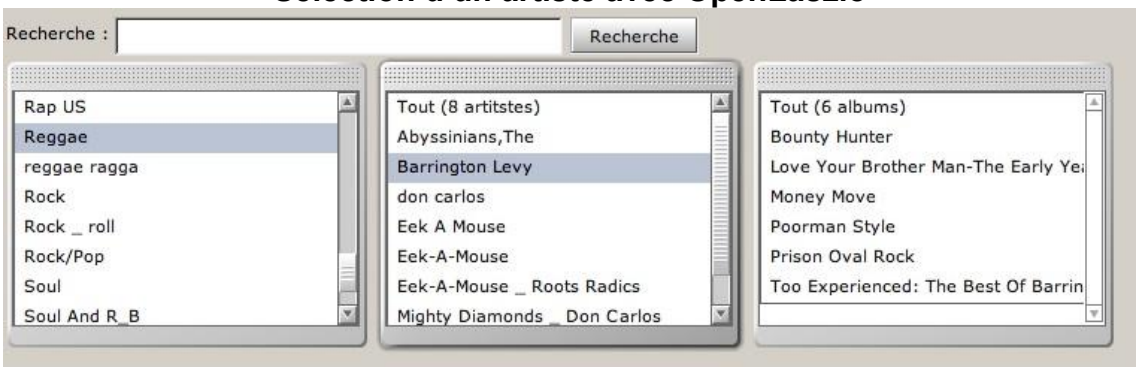
Lors de l'initialisation de l'application, on trouve les listes de genres, artistes et albums chargées. On peut sélectionner un genre :

Figure 24
Sélection d'un genre avec OpenLaszlo



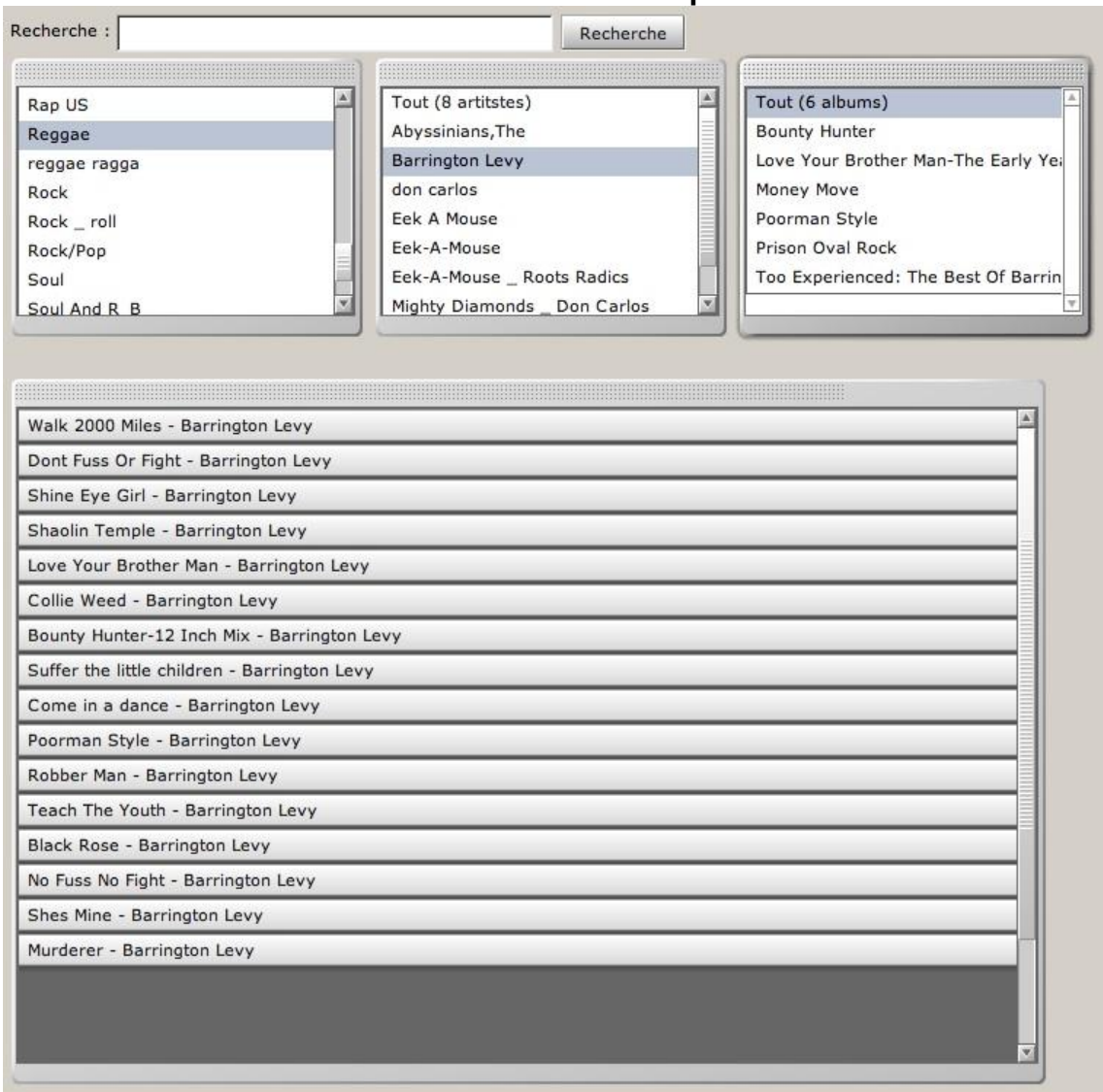
On met à jour les listes des artistes et des albums. On ne remarque pas le temps que l'application prend pour interroger le serveur et lui demander les informations, celui-ci est quasi instantané.

Figure 25
Sélection d'un artiste avec OpenLaszlo



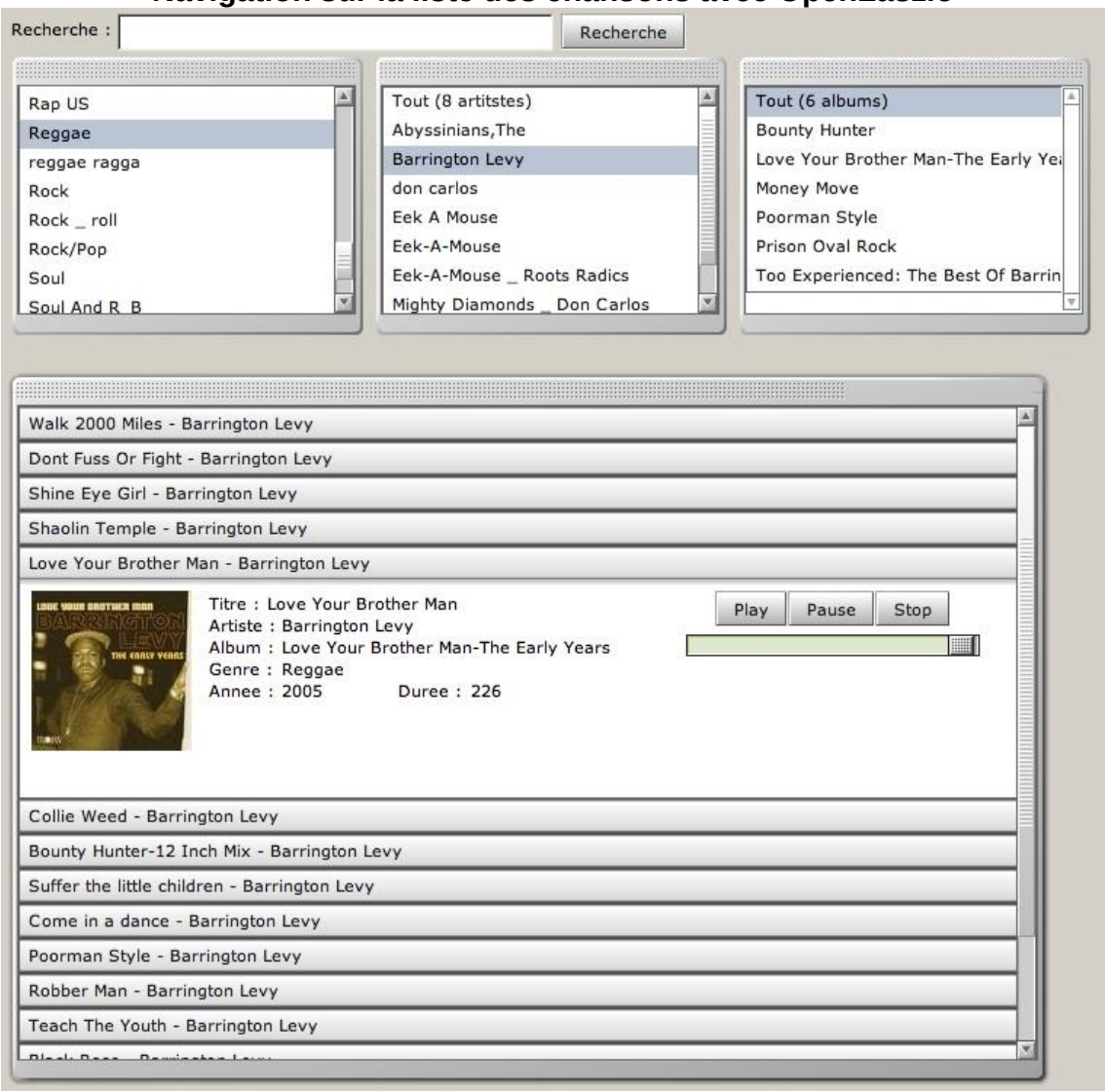
La sélection d'un artiste met à jour la liste des albums.

Figure 26
Sélection d'un album avec OpenLaszlo



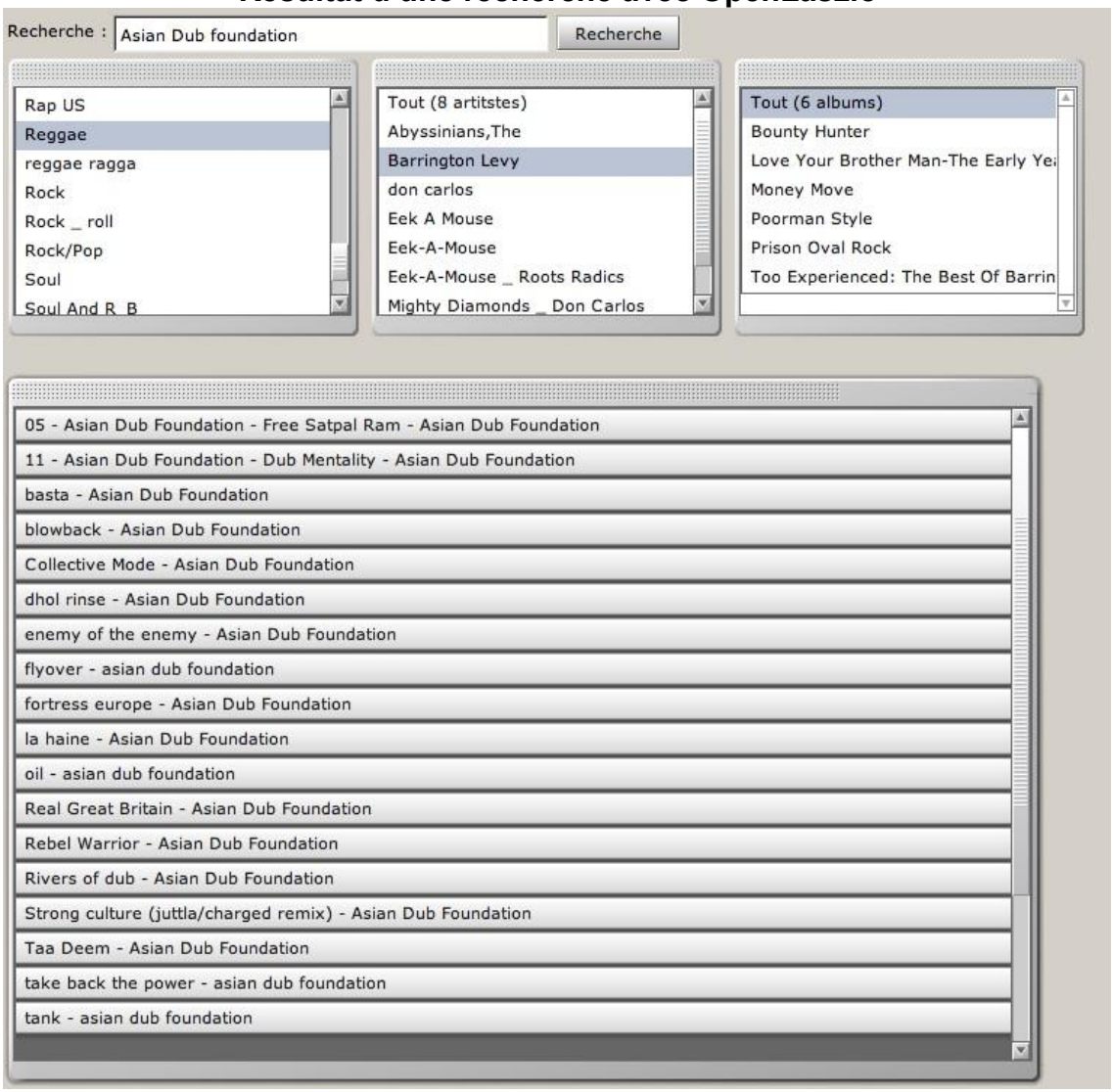
Lorsqu'on clique sur un album, la liste « accordéon » se charge et affiche les chansons. La navigation est la même que pour l'application Flex, elle est intuitive :

Figure 27
Navigation sur la liste des chansons avec OpenLaszlo



On sélectionne une chanson et on accède à ses informations. On peut lire la chanson, la mettre en pause ou la stopper. Les outils pour lire le contenu multimédia sont fournis de base avec OpenLaszlo. Il n'existe pas encore de Framework qui gère les médias de façon performante. On ne peut pas savoir si le titre musical a totalement été chargé ou encore si l'adresse url de la chanson est correcte.

Figure 28
Résultat d'une recherche avec OpenLaszlo



La recherche fonctionne de la même manière que dans l'application Flex, mais elle ne supporte pas l'auto-complétion.

5.4.5 Bilan

OpenLaszlo souffre de ne pas avoir une très grande communauté. Ce qui est dommage puisque c'est un outil très puissant qui est facile à prendre en main. Il est possible de générer du Flash ou du DHTML pour le même code source, seulement dans certains cas - comme celui-ci par exemple – cela ne fonctionne pas. Il faut donc choisir avant de programmer sous quelle forme l'application devra être déployée. Même si l'application réalisée ne fonctionne pas en DHTML, d'autres applications sans gestion de médias peuvent être réalisées rapidement grâce à OpenLaszlo. De cette

façon, elles peuvent être déployées à la fois sous Flash (versions 8, 9 et 10) et DHTML et lors de la visite d'un utilisateur afficher la version qui convient le mieux pour l'utilisateur (détection du plugin Flash ou non).

Tableau 5
Points positifs et négatifs d'OpenLaszlo

Points positifs	Point négatifs
Facilité de mise en œuvre	Documentation – communauté
Déploiement	Installation
	Maintenance
	Gestion des médias

Conclusion

Depuis leur introduction, les applications de type client riche n'ont cessé d'être mises en avant et d'être de plus en plus présentes dans notre navigation Web. Le client riche est clairement toujours d'actualité et il est fort probable que leur utilisation devienne de plus en plus commune.

Les différents acteurs du marché et les entreprises qui utilisent des outils pour mettre en place ces RIAs l'ont bien compris et ne cessent de fournir des services visant à rendre l'interaction et la manipulation de contenu entre l'utilisateur et les applications toujours plus effectives.

De nombreux choix existent sur les technologies à adopter pour réaliser ce genre d'application et il ne faut pas négliger les différents aspects utilisateurs, développeurs et entreprises. Une application doit répondre aux critères de chacune des parties et les efforts réalisés ces dernières années ont contribué à combler les manques dans ce domaine qui pouvaient encore subsister. Les technologies existantes fournissent d'énormes possibilités qui apportent de la valeur ajoutée dont principalement la satisfaction du client et la réduction des coûts.

Chaque technologie possède ses qualités et ses défauts et il est primordial d'en être informé avant de se lancer dans le développement d'applications de type client riche. Le choix d'une technologie dépend fortement de la nature de l'application car elles ne fournissent pas toutes les mêmes résultats. L'apprentissage d'une technologie et de son langage de programmation nécessite un temps d'adaptation et de maîtrise plus ou moins long qui sera déterminant lors de la décision de son utilisation. C'est aux entreprises et aux développeurs de choisir celle qui convient le mieux pour leurs besoins.

Les usages des utilisateurs sur le Web ont radicalement été changés ces dernières années et nous pouvons nous demander dans quelles mesures les RIAs continueront d'évoluer, quel sera l'aboutissement de l'expérience utilisateur sur le Web ? De plus, quels changements permettront d'aboutir à une mise en place plus rapide et moins coûteuse ? Est-il possible que des standards entre les acteurs du marché voient le jour et que l'interopérabilité entre langages devienne réalisable ?

Bibliographie

ROGOWSKI, Ron, *The Business Case For Rich Internet Applications*. Forrester Research, 12 Mars 2007. 11p. 41774

HAMMOND, Jeffrey, *Ajax Or Flex?: How To Select RIA Technologies*. Forrester Research, 28 Décembre 2006. 7p. 40989

RIED, Stefan, *Ajax Disappoints Power Users Looking For Web 2.0 - Style Business Apps*. Forrester Research, 20 Mars 2008. 8p. 45536

MENDEL, Thomas, *Enterprise Platform Trends, H2 2008: Chrome Arrives And Vista Departs*. Forrester Research, 16 Mars 2009. 5p. 45899

ROGOWSKI, Ron, *Rich Internet Applications: Not Just For Customers*. Forrester Research, 1 Mars 2007. 4p. 41717

ROGOWSKI, Ron, *Rich Internet Applications: Why & How*. Forrester Research, 1 Septembre 2006. 12p. 39816

ROGOWSKI, Ron, Smackdown : *Rich Internet Applications Versus HTML*. Forrester Research, 7 Décembre 2006. 13p. 40566

ROGOWSKI, Ron, *Web Users Want Ruch Internet Applications*. Forrester Research, 13 Septembre 2006. 6p. 40333

Laszlo System, *OpenLaszlo : An Open Architecture Framework for Advanced Ajax Application*. [en ligne] Laszlo System : Technology White Paper, Novembre 2006. http://www.openlaszlo.org/whitepaper/LaszloWhitePaper.pdf (consulté le 15.10.2009)

KLEIN, Norman, CARLSON, Max, *Laszlo In Action*. Manning Publications Co., 2008

NODA, Tom, HELWIG Shawn, *Rich Internet Applications*. [en ligne] UW E-Business Consortium, 16 Novembre 2005. http://www.uwebc.org/opinionpapers/docs/RIA.pdf (consulté le 15.10.2009)

KEYNOTE, *Rich Internet Applications : Design, Measurement, and Management Challenges*. [en ligne], http://keynote.com/docs/whitepapers/RichInternet_5.pdf (consulté le 15.10.2009)

COZIC, Frédéric, *blog.aysoon.com* [en ligne] http://blog.aysoon.com/Que-sont-les-applications-riches-RIA-Partie-1-Definition-et-usages-195 (consulté le 15.10.2009)

DESHAYES, Fabien, *blog.tangane.com* [en ligne] http://blog.tangane.com/tag/ria/ (consulté le 15.10.2009)

ZORITCHACK, Gaetan, MARIAC, Pierre, *itintegrans.com* [en ligne] http://itintegrans.com/le-blog-itintegrans/11-technologie/32-developpement-application-ria-choisir-entre-gwt-et-flex.html (consulté le 15.10.2009)

HES-SO VALAIS, INSTITUT INFORMATIQUE DE GESTION, *InterfaNews* [en ligne], http://www.interfanews.ch/wiki/index.php?title=Accueil (consulté le 15.10.2009)

BORDERIE, Xavier, *Les technologies multiplate-forme de XUL* [en ligne], 7 Mai 2004 http://www.journaldunet.com/developpeur/tutoriel/xml/040506-xml-xul-xpcom-xpconnect.shtml (consulté le 15.10.2009)

REUVEN, Cohen, MARINELLO, Roberto, KETABCHI, Farshid, ROMANSKI, Pat, MCLAUGHLAN, Ted, *Web 2.0 Re-Examined : Nexaweb's Coach Wei On The Paradigm Shift, Technology Stack and Business Value* [en ligne], 11 Janvier 2007

http://web2.sys-con.com/node/313618&usg=__NUq65MSjRPYwkgdNqTj_irvqqJc=&h=450&w=545&sz=44&hl=fr&start=78&sig2=WOW_14S2Rv23B6iHeeYNhA&um=1&tbnid=Sq8FL7aq4vMa2M:&tbnh=110&tbnw=133&prev=/images%3Fq%3Dhistorique%2Bweb%2Bmainframe%2Bclient%2Bserveur%26ndsp%3D21%26hl%3Dfr%26lr%3D%26client%3Dsafari%26rls%3Den%26sa%3DN%26start%3D63%26um%3D1&ei=6jWmSvrxCY74-QaBmMnxDw (consulté le 15.10.2009)

OBJET DIRECT, *Choisir une solution RIA* [en ligne], http://wiki.objetdirect.com/wiki/index.php?title=Choisir_une_solution_RIA#Les_technologies_.C2.AB_applet-like_.C2.BB (consulté le 15.10.2009)

BRIJESH, Deb, *Rich Internet Applications : A Look Into Available Technology Choices* [en ligne], http://www.jaxmag.com/itr/online_artikel/psecom,id,828,nodeid,147.html (consulté le 15.10.2009)

VAIBHAV, V. Gadge, *Technology options for Rich Internet Applications* [en ligne], 25 Juillet 2006, http://www.ibm.com/developerworks/library/wa-richiapp/#compare (consulté le 15.10.2009)

KAVA, Akash, *akashkava.com* [en ligne] http://akashkava.com/blog/41/ria-why-which-one-how/ (consulté le 15.10.2009)