

Titre courant : MobileQUIZ : une application mobile multiplateforme réalisée en Ajax avec le Google Web Toolkit

MobileQUIZ : une application mobile multiplateforme réalisée en Ajax avec le Google
Web Toolkit

Claude Coulombe, Nobry Phan Ouk

MATI Montréal

Résumé

Avec la prolifération des téléphones intelligents et l'émergence des tablettes se pose la question des technologies à privilégier pour la réalisation des applications mobiles dans le contexte de l'éducation.

Doit-on développer spécifiquement pour chaque plateforme une application native ou chercher à développer une application multiplateforme?

La règle de prise de décision est simple, si l'application n'est pas obligée d'accéder à des périphériques spécifiques à une plateforme ou de répondre à des contraintes de performances comme le « temps réel » ou le graphisme 3D, il est préférable de créer une application multiplateforme.

Parmi toutes les solutions de développement multiplateforme, la réalisation d'une application Web riche avec des technologies Ajax pour appareil mobile apparaît comme choix intéressant. Dans ce contexte, le fureteur est la plateforme et les technologies Web du fureteur, dites technologies Ajax comme JavaScript, HTML/HTML5 et CSS sont privilégiées pour réaliser l'application.

Encore là, de nombreux choix sont possibles dont des outils Ajax multiplateformes à base de JavaScript comme jQuery ou PhoneGap et GWT (Google Web Toolkit) un outil Ajax en logiciel libre basé sur Java.

Les auteurs ont donc réalisé un client multiplateforme avec GWT dans le cadre du projet MobileQuizz conçu pour contrôler à partir de questionnaires l'apprentissage de contenus scientifiques selon une approche conceptuelle.

À plus long terme, la généralisation de la norme HTML5, l'extension de ses APIs afin d'inclure de nouveaux périphériques et l'accroissement des performances des engins JavaScript devraient élargir les possibilités des applications mobiles multiplateformes réalisées avec les technologies Ajax.

Contexte

Avec la prolifération des plateformes mobiles, particulièrement les téléphones intelligents (Android [1], BlackBerry [2], iPhone [3], Palm Pré [4], Windows Phone [5], etc.) et l'émergence des tablettes du genre iPad [6] se pose la question de leur réalisation

technique et du choix des technologies à privilégier.

Les applications visées sont des applications Web comportant une interface riche désignées en anglais sous le terme RIA pour Rich Internet Applications. Une « interface riche » désigne une interface-usager non triviale et réactive semblable à une interface de logiciel bureautique classique (en anglais Desktop Like) en plus de l'utilisation de contenus multimédia comme le son, les images et la vidéo.

De plus, il faut tenir compte du contexte particulier des applications mobiles dans le monde de l'éducation. Ici, l'emploi du terme « monde de l'éducation » est préféré au terme plus commercial de « marché de l'éducation ».

Les contraintes des applications mobiles

Au niveau des applications mobiles, plusieurs contraintes existent. Par exemple, il faut concevoir des applications pour de petits écrans qui sont utiles, attractives et faciles à utiliser le plus souvent en touchant l'écran tactile. Cela représente de beaux défis au niveau de la conception des interfaces-usagers et de l'ergonomie.

Quant aux restrictions matérielles sur le plan de la mémoire vive et de la capacité des processeurs, elles disparaissent rapidement avec l'évolution de la technologie et l'accroissement des capacités des appareils mobiles. À moins d'une application très gourmande en ressources, les restrictions matérielles reliées à la mémoire ou au processeur ne devraient plus être des obstacles, du moins pour des matériels récents. Il en est de même pour l'emploi du multimédia comme le son ou la vidéo avec la généralisation de la norme HTML5 [7].

Il existe également des contraintes au niveau de l'emploi de la connexion sans fil, de l'usage de la bande passante et du fonctionnement de l'application hors-ligne. Encore là, l'apparition de mécanismes de persistance au niveau du fureteur comme Google Gears [8] et leurs généralisations avec la norme HTML5 fait disparaître cette contrainte.

La prolifération des plateformes mobiles

Au delà des contraintes communes à toutes les applications pour appareils mobiles se pose le problème grandissant de la diversité des plateformes mobiles. Ici le terme plateforme se réfère surtout au matériel (en anglais hardware) mais également au logiciel, principalement au système d'exploitation et aussi le fureteur web (ou navigateur).

Après le lancement du premier iPhone [3] par Apple en 2007, la question se posait moins

car il n'y avait qu'une seule marque d'appareil et l'on développait spécifiquement pour cette plateforme. Or depuis, des alternatives sont apparues comme la plateforme Android [1] lancée par Google en 2008, le pionnier Blackberry [2] (depuis 1999) qui a modernisé son offre, le Pré [4] de Palm (2009) racheté par Hewlett-Packard en 2010, le Windows Phone [5] (2010) et bien d'autres encore. Or en novembre 2010 le nombre d'utilisateurs de la plateforme Android aux États-Unis a dépassé le nombre d'utilisateurs du iPhone [9] et en février 2011 il dépassait le nombre d'utilisateurs de Blackberry [10].

Prolifération des langages et des outils de développement

On assiste également à une prolifération des langages et des outils pour réaliser des applications mobiles.

Les applications pour le iPhone [3] d'Apple doivent être réalisées pour l'iOS (système d'exploitation pseudo multitâche propriétaire) avec le langage Objective-C dans l'environnement de développement intégré Xcode et la boîte à outils (kit de développement ou SDK) spécifique au iPhone.

Les applications pour le BlackBerry [2] de Research In Motion qui roulent sur un système d'exploitation multitâche propriétaire sont programmées ou bien en Java avec une bibliothèque de composants spécifiques et un plugiciel Eclipse (le principal environnement intégré de programmation pour Java) comprenant un émulateur de BlackBerry ou bien avec l'outil Ajax [11] WebWorks.

C'est une solution similaire pour la plateforme Android [1] de Google, incluant le plugiciel Eclipse et le simulateur à la différence que les outils d'une technologie plus récente sont pour la plupart en logiciels libres (licence Apache) et roulent sur un système d'exploitation multitâche dérivé du noyau de Linux.

De son côté, le Pré [4] de Palm-HP est basé sur le système multitâche webOS dérivé de Linux qui a remplacé le Palm OS et des outils Ajax [11] regroupés sous le nom de Palm webOS SDK avec un plugiciel Eclipse.

Enfin, la plateforme Windows Phone [5] de Microsoft qui tourne sous Windows Phone 7 (système d'exploitation multitâche propriétaire basé sur le noyau de Windows CE 6) exige le développement en C# ou en Visual Basic dans l'atelier Visual Studio 2010 avec les outils du Windows Phone Developer Tools incluant un émulateur.

Rareté des ressources humaines

Pire, le développement d'applications natives sur ces différentes plateformes mobiles requiert une expertise rare.

Malgré des efforts de simplification, la maîtrise de chacune des plateformes et de leurs outils de développement représente une courbe d'apprentissage importante. Par conséquent, les ressources humaines compétentes sont rares, coûteuses et fragmentées selon la popularité des plateformes.

La diffusion

Il existe des contraintes au niveau de la diffusion de l'application. En effet, certaines plateformes comme le iPhone d'Apple limitent expressément les technologies de réalisation des applications pouvant être diffusées dans leur boutique en ligne.

Par exemple, seules les applications réalisées avec le langage Objective-C peuvent accéder au App Store d'Apple. Donc pas d'applications Java, ni Flash, ni d'applications réalisées en technologies Ajax dans le App Store qui est directement intégré au iPhone.

Pour la plupart des entreprises commerciales, l'avantage du App Store est évident même si elles doivent laisser à Apple une part importante (30 %) du prix de vente de leurs logiciels.

Dans le contexte non commercial de l'éducation, la diffusion via une boutique en ligne n'est peut-être pas essentielle. Il faut donc mettre dans la balance la difficulté relative et surtout le coût plus élevé de développer une application native versus l'avantage que représente la diffusion dans une boutique « commerciale » versus l'accessibilité élargie d'un site web mobile.

Généralement, les revenus de la vente ne font pas partie de l'équation car la plupart des applications éducatives sont distribuées gratuitement.

Une institution publique recevra beaucoup de pression afin de diffuser sur le plus grand nombre de plateformes possibles afin d'en augmenter l'accessibilité et afin de ne pas être accusée de favoriser une entreprise commerciale au dépend des autres.

Quelque part, il faut également apprendre à résister à un effet de mode qui consiste à avoir son application mobile dans le App Store d'Apple.

Application native versus application multiplateforme

Avant tout, il faut bien distinguer entre une application native qui implique généralement une dépendance directe sur le matériel et/ou le système d'exploitation et une application portable (ou multiplateforme) qui fonctionnera sur différentes plateformes matérielles et/ou systèmes d'exploitation.

En réalité, il est tout à fait possible de créer des applications hybrides qui comporteront des parties « natives » et des parties « portables ». Une approche hybride peut être dans un premier temps, de réaliser une application Web mobile qui roule sur toutes les plateformes puis de réécrire certaines parties très circonscrites pour spécialiser l'application afin de tirer profit de la plateforme matérielle spécifique. Par exemple, remplacer les contrôles HTML (boutons, menus, etc.) par des contrôles natifs (boutons physiques).

Une première question se pose, doit-on développer spécifiquement pour chaque plateforme une application native ou chercher à développer une application multiplateforme qui sera portable sur les différentes plateformes?

Evidemment, il faut d'abord évaluer les besoins de l'application spécifique et le contexte de sa diffusion.

La règle de prise de décision est simple, si l'application n'est pas obligée d'accéder à des périphériques spécifiques à la plateforme (géolocalisation, caméra, accéléromètre, système téléphonique, boutons ou manettes physiques, etc.) ou de répondre à des contraintes de performances comme le « temps réel » ou le graphisme 3D, il est généralement préférable de créer une application multiplateforme portable.

Les contenus multimédia qui jusqu'ici justifiait l'emploi de la technologie Flash n'est plus une contrainte avec l'émergence des balises audio et video de HTML5 qui sont supportées par la plupart des fureteurs récents.

Les contraintes de performances de type temps réel se retrouvent typiquement dans les jeux d'action très réactifs, le graphisme 3D et les simulations complexes. En général, les applications impliquant des simulation discrètes simples et du graphisme 2D sont réalisables avec des technologies Ajax [11] comme ceux pouvant être réalisés avec la technologie Flash [12] par exemple

Technologies Ajax

Comme nous parlons d'une application comportant un client Web riche et que nous écartons les solutions propriétaires (comme Flash [12] ou Silverlight [13]) et celles qui ne

roulent pas sur l'ensemble des appareils mobiles (comme Applet / JVM [14], Flash [12] Silverlight [13]), il nous reste essentiellement des technologies basées sur JavaScript, la plupart en logiciel libre, connues sous la dénomination technologies Ajax ou parfois technologies de l'Open Web. En effet Ajax [11] est le terme utilisé pour référer à la combinaison de technologies : JavaScript, DOM, HTML/HTML5 et CSS. De plus, on assume généralement que l'application ne fait que des communications asynchrones via l'emploi de l'objet JavaScript XMLHttpRequest.

Dans ce contexte, le fureteur est la plateforme et les technologies du fureteur, dites technologies Ajax, comme JavaScript, DOM, HTML/HTML5 et CSS sont privilégiées pour réaliser l'application.

Encore là, de nombreux choix sont possibles dont des outils Ajax multiplateformes à base de JavaScript comme jQuery [15] ou PhoneGap [16] et GWT (Google Web Toolkit) [17] un outil Ajax basée sur Java. De plus ces technologies ont l'avantage d'être en logiciel libre.

L'application MobileQuizz

Le projet MobileQuizz [18] consiste en l'administration de tests ou quizz comportant des choix multiples, des images, des graphiques et des équations mathématiques pour contrôler l'apprentissage de contenus scientifiques selon une approche conceptuelle. MobileQuiz est donc une application typique nécessitant l'emploi d'un client riche.

Dirigé par M. Jesus Vazquez-Abad, le projet MobileQuizz [18] implique également Mme Odile Martial et deux informaticiens, Mme Thi Lan Anh Dinh et M. Anh Tuan Nguyen, qui ont réalisé la partie serveur, la banque de données, le module d'analyse des résultats et une première version de l'application cliente avec une applet Java [14].

Réalisation d'un client riche multiplateforme avec GWT

Les auteurs, MM Claude Coulombe (architecture) et Nobry Phan Ouk (programmation) ont réalisé un nouvelle version du client multiplateforme sur la base d'une application web pour mobile avec la boîte à outils Ajax GWT (Google Web Toolkit) [17].

Nous avons opté pour GWT car nous maîtrisons bien cette technologie et qu'elle offre de nombreux avantages. Pour des raisons de performances, de portabilité multifureteur et de gestion de la complexité, nous avons choisi GWT qui permet de réaliser les interfaces-usagers entièrement en Java, de profiter des avantages d'un langage à typage

statique, d'utiliser les nombreux outils de développement Java dont Eclipse et finalement de produire le code du client en JavaScript optimisé.

Le client riche Ajax a été programmé et mis au point en Java à partir des composants de la bibliothèque de GWT dans l'environnement de développement Eclipse. Le code ressemble beaucoup au code de la librairie Swing de Java et fait appel à la notion d'événements (action d'un acteur dans le système, le plus souvent une interaction de l'utilisateur) et de programmation événementielle (en anglais event-driven programming) [19].

Pour l'interface-utilisateur nous avons utilisé UiBinder [20], un outil XML de création d'interface qui permet un découplage facile du contenu de l'interface (en XML), de son comportement via un fichier Java associé et de son apparence via des CSS dans un fichier externe qui peut être spécifique à chaque plateforme.

Les composants créés avec UiBinder suivent le patron de conception « composite » ou « objet composite » [21]. En effet, un élément d'interface complexe est construit ou composé d'éléments d'interface plus simples dans un conteneur englobant.

Ci-dessous un bout de code de l'interface-client de MobileQuizz dans le formalisme déclaratif propre à UiBinder. On voit qu'un panneau vertical contient un panneau horizontal lui-même contenant deux autres panneaux horizontaux qui contiennent des étiquettes. Cela fonctionne un peu comme des blocs de Lego.

```

<!DOCTYPE ui:UiBinder SYSTEM "http://dl.google.com/gwt/DTD/xhtml.ent">
<ui:UiBinder xmlns:ui="urn:ui:com.google.gwt.uibinder"
  xmlns:g="urn:import:com.google.gwt.user.client.ui">
  <g:VerticalPanel>
    <g:HorizontalPanel styleName="topMenu">
      <g:HorizontalPanel styleName="leftpart">
        <g:Label>Réponses réussies: </g:Label>
        <g:Label ui:field="success"></g:Label>
        <g:Label>% </g:Label>
        <g:Label ui:field="aok"></g:Label>
        <g:Label>/</g:Label>
        <g:Label ui:field="qp"></g:Label>
        <g:Label>)</g:Label>
      </g:HorizontalPanel>
      <g:HorizontalPanel styleName="rightpart">
        <g:Label>Questions maîtrisées: </g:Label>
        <g:Label ui:field="mastered"></g:Label>
        <g:Label>% </g:Label>
        <g:Label ui:field="nb_question"></g:Label>
        <g:Label>/</g:Label>
        <g:Label ui:field="q"></g:Label>
        <g:Label>)</g:Label>
      </g:HorizontalPanel>
    </g:HorizontalPanel>
    <g:VerticalPanel ui:field="middle" styleName="middlePart">
      <g:HTML ui:field="question" styleName="question"></g:HTML>
      <g:HTML ui:field="choices" styleName="choices"></g:HTML>
    </g:VerticalPanel>
    <g:VerticalPanel styleName="bottomMenu">
      <g:HTML ui:field="commentDisplay" styleName="comment"></g:HTML>
      <g:HorizontalPanel styleName="bottomWindow">
        <g:HorizontalPanel styleName="leftpart">
          <g:MenuBar>
            <g:MenuItem ui:field="user">Usager:</g:MenuItem>
          </g:MenuBar>
          <!-- <g:Label ui:field="username"></g:Label -->
        </g:HorizontalPanel>
        <g:HorizontalPanel styleName="rightpart">
          <g:Button
            styleName="hint">Indice</g:Button>
          ui:field="hint"

```

```

        <g:Button                                ui:field="check"
styleName="check">Valider</g:Button>
        <g:Button                                ui:field="skip"
styleName="skip">Avancer</g:Button>
        </g:HorizontalPane>
    </g:HorizontalPane>
</g:VerticalPane>
</g:VerticalPanel>
</ui:UiBinder>

```

Le code du comportement associé à l'interface-usager utilise le patron de conception de la « Commande » (en anglais Command Design Pattern) [22]. Ci-dessous, un extrait du code du comportement de l'interface montrant deux commandes « getOnline » et « getOffline ».

```

public Client() {
    initWidget(uiBinder.createAndBindUi(this));

    getOnline = new Command() {
        @Override
        public void execute() {
            // Test server connection with RPC
            AsyncCallback<Integer> callback = new
AsyncCallback<Integer>() {
                @Override
                public void onFailure(Throwable caught) {
                    popupLine("No connection to server, staying
offline.", false);
                }
                @Override
                public void onSuccess(Integer result) {
                    // if it's OK, go online
                    if ( result == 1 ) { // Should be OK by default
                        popupLine("Local data cleared, back
online.", true);
                    }
                    goOnline();
                    UpdateState();
                }
            }
        };
        QuizServiceAsync testService = GWT.create(QuizService.class);
        testService.testServer(callback);
    }
};

getOffline = new Command() {
    @Override
    public void execute() {
        goOffline();
        UpdateState();
    }
};
...

```

Remarquez l'appel asynchrone au service RPC (Remote Procedure Call) [23] dans la commande `getOnline` avec la fonction de retour asynchrone `AsyncCallback` qui procède à la mise à jour de l'interface-usager en fonction du résultat de la communication asynchrone avec le serveur.

Le code Java a été compilé pour produire un code JavaScript optimisé par le transcompilateur (en anglais `cross compiler`) intégré à GWT. Ceci est un gros atout pour GWT. Selon Google, à partir de 100 lignes de code le transcompilateur de GWT génère un code JavaScript plus performant qu'un code JavaScript fait à la main.

Le fonctionnement hors ligne de `MobileQuizz` est assuré par l'emploi de la technologies `Gears` [8] de Google. De plus l'affichage des équations mathématiques a nécessité l'inclusion de `ASCIIMathML.js` [24], une bibliothèque JavaScript en code source libre spécialisée dans l'affichage du formalisme mathématique selon la norme `MathML`.

L'harmonisation des feuilles de style CSS aux différents appareils mobiles a également constitué un travail plutôt fastidieux.

Un autre défi important a été de « faire communiquer » la nouvelle interface-usager avec le serveur existant. Heureusement, les capacités d'intégration du code JavaScript de l'API `JSNI` (JavaScript Native Interface) [25] de GWT ont permis de manipuler relativement facilement les flux de données échangés avec le serveurs qui étaient en `JSON` (JavaScript Object Notation) [26].

Des expériences ont été réalisées avec `MobileQuizz` sur des contenus en chimie et en thermodynamique. Dans un premier temps, les appareils utilisés étaient des `IPaq 210` de HP sur lesquels nous avons installés un fureteur `Opera` reconnu pour les plateformes mobiles et l'extension `Google Gears`. `MobileQuizz` a donc été utilisé avec succès sur différents appareils mobiles dont `iPhone`, `IPaq` et `Android`.

Afin d'améliorer `MobileQuizz` au niveau de son apparence ou pour ajouter de nouvelles fonctionnalités nous pourrions utiliser des codes des bibliothèques JavaScript `PhoneGap` [16], `GwtMobile` [27] ou `Gwt-mobile-webkit` [28] en exploitant l'API `JSNI` [25] de GWT.

Application Ajax dans le App Store

Il est « théoriquement » possible de créer une application « acceptable » pour le App Store d'Apple en partant d'un squelette d'application écrit en `Objective-C` et en lui incorporant un code `Ajax`. Pour cela, il faut utiliser `XCode` et payer 99 \$ (USD) pour

devenir « officiellement » un développeur Apple.

Cependant, votre application est soumise au jugement arbitraire des employés d'Apple qui passent en revue les applications pour les autoriser dans le App Store. Certains employés d'Apple considèrent que GWT ou PhoneGap sont des APIs privées (private API) et par conséquent ils rejettent les applications créées avec ces outils.

Bien que certains blogueurs affirment que des applications Ajax sont désormais acceptées par le App Store [29], [30] d'autres recommandent de maquiller ou adapter le code produit par ces outils pour les rendre « acceptable » [31]. Il existe même des scripts pour transformer « automatiquement » le code produit.

Conclusion

Dans le contexte « non commercial » du monde de l'éducation, nous avons montré qu'à moins de contraintes matérielles spéciales, il était avantageux de développer une application mobile multiplateforme en utilisant des technologies Ajax en logiciel libre. Nous avons également montré que le Google Web Toolkit présentait des atouts importants.

À plus long terme, la généralisation de la norme HTML5, l'extension de ses APIs afin d'inclure de nouveaux périphériques et l'accroissement des performances des engins JavaScript devraient élargir les possibilités des applications web multiplateformes réalisées avec les technologies Ajax.

Références

[1] Android par Google

www.android.com

Android SDK

<http://developer.android.com/index.html>

Plugiciel ADT (Android Development Tool) pour Eclipse

<http://developer.android.com/sdk/eclipse-adt.html>

[2] BlackBerry par Research In Motion

ca.blackberry.com/fr

Java Application Development

<http://us.blackberry.com/developers/javaappdev>

BlackBerry WebWorks Development

<http://us.blackberry.com/developers/browserdev>

[3] iPhone par Apple

www.apple.com/ca/fr/iphone

Xcode

<http://developer.apple.com/technologies/tools>

iOS Developer Program

<http://developer.apple.com/programs/ios/>

[4] Palm Pré par Hewlett-Packard

www.palm.com/us/products/phones/pre/index.html

Plugiciel webOS pour Eclipse

<https://developer.palm.com/content/index.php?id=4152>

[5] Windows Phone par Microsoft

www.microsoft.com/windowsphone/fr-fr/default.aspx

Windows Phone Development

<http://msdn.microsoft.com/en-us/library/ff402535%28v=VS.92%29.aspx>

[6] iPad par Apple

www.apple.com/ca/fr/ipad/

Xcode

<http://developer.apple.com/technologies/tools>

[7] HTML5

Article général dans Wikipedia

<http://fr.wikipedia.org/wiki/HTML5>

La norme HTML5 sur le site de W3C

<http://www.w3.org/TR/html5/>

[8] Google Gears

<http://gears.google.com>

[9] Android passes iPhone in the US

www.techspot.com/news/41896-android-passes-iphone-in-the-us.html

[10] Android passes BlackBerry in the US

<http://www.techspot.com/news/43105-android-passes-blackberry-in-the-us.html>

[11] Ajax (Asynchronous JavaScript and XML), Ajax est maintenant le terme utilisé pour

référer à la combinaison de technologies : JavaScript, DOM, HTML/HTML5 et CSS.

http://fr.wikipedia.org/wiki/Ajax_informatique

L'article de Jesse James Garrett qui a lancé le terme Ajax

<http://www.adaptivepath.com/ideas/e000385>

[12] Adobe Flash

www.adobe.com/products/flash/

[13] Microsoft Silverlight

www.silverlight.net

[14] Applet Java : Les applets Java sont des applications en bytecode, téléchargées depuis un serveur et qui s'exécutent du côté client dans une machine virtuelle Java ou JVM (Java Virtual Machine).

<http://java.sun.com/applets>

[15] jQuery

<http://jquery.com>

[16] PhoneGap

www.phonegap.com

[17] GWT (Google Web Toolkit)

<http://code.google.com/intl/fr/webtoolkit/overview.html>

[18] Programmation événementielle ou programmation par événements

http://fr.wikipedia.org/wiki/Programmation_%C3%A9v%C3%A9nementielle

[19] UiBinder

http://code.google.com/intl/fr/eclipse/docs/gwt_uibinder.html

[20] Patron de conception « commande »

http://fr.wikipedia.org/wiki/Commande_patron_de_conception

<http://www.blackwasp.co.uk/Command.aspx>

[21] Patron de conception « composite »

http://fr.wikipedia.org/wiki/Objet_composite

[22] RPC (Remote Procedure Call)

<http://code.google.com/intl/fr/webtoolkit/doc/latest/DevGuideServerCommunication.html>

[#DevGuideRemoteProcedureCalls](#)

[23] JSNI (JavaScript Native Interface)

<http://code.google.com/intl/fr/webtoolkit/doc/latest/DevGuideCodingBasicsJSNI.html>

[24] JSON (JavaScript Object Notation)

www.json.org

[25] Odile Martial, Jesús Vázquez-Abad, *Didactique des sciences et ergonomie pour l'innovation grâce aux technologies mobiles d'apprentissage*, Ergo'IA, Proceedings de la conférence Ergonomie et Informatique Avancée, New York 2010.

[26] ASCIIMathML.js, une bibliothèque JavaScript en code source libre spécialisée dans l'affichage du formalisme mathématique selon la norme MathML.

<http://mathcs.chapman.edu/~jipsen/mathml/asciimath.html>

<http://mathcs.chapman.edu/~jipsen/mathml/asciimathdownload.html>

[27] GwtMobile

<https://github.com/dennisjzh/GwtMobile>

[28] Gwt-mobile-webkit

<http://code.google.com/p/gwt-mobile-webkit/>

[29] PhoneGap officially permitted on the App Store

<http://nachbaur.com/blog/phone-gap-officially-permitted-on-the-app-store>

[30] PhoneGApp Store Approval

<http://blogs.nitobi.com/jesse/2009/11/20/phone-gapp-store-approval/>

[31] Remove PhoneGap References

<http://wiki.phonegap.com/w/page/16494817/Remove-PhoneGap-References>