Revisão de Literatura de *Frameworks* de Desenvolvimento Móvel Multiplataforma

CAPSI/2013

Pedro J. Freire [1], Rui Ribeiro [2]

1) SITILabs, Universidade Lusófona, Portugal

pedro.freire@ulusofona.pt

2) SITILabs, Universidade Lusófona, Portugal
rui.ribeiro@ulusofona.pt

Resumo

Num mundo de dispositivos móveis ubíquos, desenvolver a mesma aplicação para cada plataforma móvel pode ser uma tarefa economicamente desgastante, mesmo para o Estado. A solução habitual a este problema tem sido recorrer a *frameworks* de abstração de plataforma como o Qt, Titanium e PhoneGap, mas a qualidade resultante em termos de experiência de utilização pode desiludir bastante. Neste documento, fazemos uma revisão de literatura de *frameworks* de abstração de plataforma, nas suas várias incarnações, e damos a conhecer as suas características em termos de eficiência computacional e facilidade de desenvolvimento.

Palavras chave: revisão de literatura, *survey*, *framework* móvel, plataforma móvel, multiplataforma

1 Introdução

A informática está a tornar-se cada vez mais ubíqua. Atualizamos redes sociais no autocarro, lemos as últimas notícias no trabalho e mandamos *tweets* aos nossos amigos no restaurante.

Esta ubiquidade também se reflete no local de trabalho. De acordo com Gordon Thomson [Thomson 2012], um estudo da Cisco indica que a tendência atual no local de trabalho é de os colaboradores exigirem acesso aos dados empresariais pelos seus dispositivos móveis e a tendência dos consumidores é a mesma. As organizações estão a adaptar-se a estas exigências instituindo políticas de *Bring Your Own Device* (Traga o Seu Próprio Dispositivo, BYOD) e a portar as suas aplicações internas e públicas para múltiplas plataformas. A Administração Pública portuguesa não está alheia a este fenómeno e a Direção-Geral da Qualificação dos Trabalhadores em Funções Públicas, abreviadamente designada por INA, tem feito apresentações [Anjos 2010], [WMT] sobre este tema.

No entanto, este esforço é economicamente esgotante, tal como reportado por Charland e Leroux [Charland e Leroux 2011]. *Frameworks* multiplataforma para auxiliar o desenvolvimento são essenciais e o mercado respondeu rapidamente com múltiplas soluções.

Nas secções seguintes estabelecemos definições e alguns requisitos a que *frameworks* de desenvolvimento móvel multiplataforma deveriam almejar para atingir o objetivo de auxiliar o desenvolvimento. Fazemos em seguida uma revisão de literatura do atual estado da arte,

agrupando as *frameworks* pela linguagem ou tecnologia subjacente que assegura o desenvolvimento multiplataforma com um único código-fonte. Terminamos na secção 7 com as nossas conclusões.

1.1 Trabalho relacionado

Outros autores já fizeram revisões de literatura desta área [Singh e Palmieri 2011], [Hartmann et al. 2011], [Ribeiro e da Silva 2012], [Raj e Tolety 2012], [Rajendran 2013], embora de forma menos ampla.

1.2 Definições e requisitos

Definimos uma *framework* multiplataforma como um conjunto de ficheiros de código-fonte, bibliotecas e ferramentas que:

Suporta múltiplas plataformas. Pelo menos duas plataformas diferentes. Nesta definição, iOS e Android são duas plataformas diferentes. iOS 5 e iOS 6 não são.

Permite o desenvolvimento sem ramificações do código-fonte. Este código-fonte pode ocupar vários ficheiros. O que não pode é ter várias ramificações dependendo da plataforma, porque isso anularia a vantagem de se ter uma *framework* multiplataforma.

Claro que, nativamente, plataformas diferentes obrigam ao uso de código-fonte diferente. Então, para tornar possível a ausência de ramificações do código-fonte, uma *framework* multiplataforma precisa de ter pelo menos:

Uma linguagem de programação unificadora. O iOS prefere Objective C, C++ e C, Android prefere Java, Blackberry prefere C e C++ e o Windows Phone prefere C#. Outras plataformas inclinam-se para o C e C++. Embora todas estas linguagens de alguma forma tenham tido origem no C, elas têm grandes diferenças entre si. Java e C# têm a sua própria máquina virtual. Classes e objetos declarados em Objective C diferem completamente de C++. Uma *framework* multiplataforma que permita um único código-fonte sem ramificações irá necessitar de uma linguagem unificadora que fará a ponte com a API subjacente.

Uma interface de aplicação *(Aplication Programming Interface: API)* unificadora. Cada plataforma móvel tem a sua própria API. Os conceitos que cada API utiliza e os requisitos que coloca sobre os programadores (formas de organização mental, habilidades, técnicas) diferem. Código escrito para uma API não pode ser facilmente convertido para outra, mesmo quando partilham a mesma linguagem de programação subjacente. Uma *framework* multiplataforma que permita um único código-fonte sem ramificações irá necessitar de uma API unificadora que fornece uma interface "virtual" ao programador, que é depois traduzida para cada uma das APIs subjacentes por bibliotecas de tempo real, ou pelo compilador. Esta API precisa de ser abrangente ao cobrir todos ou pelo menos a maior parte dos aspetos das APIs subjacentes.

APIs para lidar com diferenças entre dispositivos. Nem todos os dispositivos têm o mesmo tamanho ou densidade (dpi/ppp) de ecrã. Nem todos os dispositivos têm giroscópios, grande capacidade de armazenamento interna ou outras características que os programadores (e as APIs nativas) assumem como dados. Estas características (ou sua falta) devem ser expostos pela API unificadora.

A escolha de uma linguagem e API irá determinar:

1. Facilidade (rapidez) de desenvolvimento (quantidade de código necessário para resolver cada problema),

2. Ausência de erros (baixo número de *bugs)* durante o desenvolvimento,
3. Dependência em terceiros (licenças, ferramentas) para desenvolvimento futuro,
4. Custos de desenvolvimento (ferramentas),
5. Recursos computacionais usados pela aplicação (adiante referidos como "carga computacional"), que por seu turno poderão determinar a experiência do utilizador com a aplicação (tempo de resposta da interface do utilizador) e dispositivo (tempo de bateria).

No entanto, algumas soluções ao problema de desenvolvimento multiplataforma não recorrem à escrita de código, tal como iremos ver na secção 6, embora recorram à codificação da aplicação de forma visual. Iremos referir-nos a estas soluções como plataformas pelo que se torna importante distinguir os dois conceitos de plataforma:

Plataforma. Usamos a expressão plataforma para exprimir dois conceitos diferentes. O primeiro é o ambiente operacional (sistema operativo) onde se irá inserir a aplicação. Usamos expressões como "suporta a plataforma", "*framework* para a plataforma" ou "multiplataforma" quando nos referimos a este conceito. O segundo conceito é um conjunto de aplicações que gerando código de forma automática (e frequentemente invisível para o programador) permitem a criação de aplicações móveis. Usamos expressões como "plataforma de desenvolvimento" quando nos referimos a este conceito.

2 *Frameworks* baseadas em WebViews

Um WebView é um objecto nativo comum que pode ser adicionado a qualquer aplicação. Ele usa o *browser* do sistema operativo subjacente para exibir conteúdo Web dentro desse WebView. Uma vez que este tipo de objecto é suportado em todas as plataformas móveis (e *desktop)* e como as tecnologias Web são conhecidas da maior parte dos profissionais de TI, esta é uma excelente forma de se conseguir desenvolvimento multiplataforma.

As *frameworks* baseadas em WebViews aproveitam a expressividade do HTML5, CSS3 e JavaScript para desenvolver aplicações em múltiplas plataformas. Estas são as suas linguagens unificadoras. Elas têm curvas de aprendizagem pequenas e um tempo de implementação muito curto. Mas sofrem de problemas de performance e não têm um aspecto completamente nativo. A experiência do utilizador é assim negativamente afectada [Charland e Leroux 2011].

Estas *frameworks* são normalmente divididas em três camadas:

- Uma aplicação WebView que serve de ponte entre as tecnologias Web e os recursos nativos e que se compila nativamente para cada plataforma. Esta camada é essencial e obrigatória.
- Uma camada JavaScript de abstração de *browser*. Embora não obrigatória, esta camada permite que se escreva código JavaScript na nossa aplicação sem se ter a preocupação de lidar com as diferenças entre *browsers*. Esta camada também costuma trazer uma biblioteca que simplifica o código, e.g.: permite o uso de `$("id")` em vez de `document.getElementById("id")`. Ela pode ser usada sobre qualquer aplicação WebView.
- Uma camada JavaScript para criação de *widgets*. Também não é obrigatória, mas recorre a, e tem de ser compatível com, a camada anterior de abstração de *browser*.

2.1 Camada de aplicação WebView

Esta camada lança uma aplicação que consiste apenas de uma WebView que ocupa o ecrã inteiro e oferece objetos adicionais ao JavaScript. Estes objetos permitem o acesso a recursos nativos do dispositivo como acelerómetro, giroscópio, GPS, câmara e muitos outros.

São exemplos deste tipo de aplicações:

- PhoneGap [Pho], [Allenetal. 2010], [Charland e Leroux 2011], [Marinacci 2012], [Richardson 2012]
- MarmaladeSDK [Mar3]
- Trigger.io [Tri]
- Lively for Qt [Mikkonen et al. 2009]

2.2 Camada de abstração de *browser*

Numa aplicação WebView, o código do utilizador corre num *browser* diferente dependendo do dispositivo (Safari em iOS, Android Browser ou Chrome no Android, Internet Explorer no Windows Phone, etc.). *Browsers* diferentes têm APIs e semântica JavaScript ligeiramente diferentes. Assim estas aplicações normalmente empregam algum tipo de camada de abstração de *browser* – trata-se de código JavaScript que fornece uma API simples mas independente de *browser* para aceder ao *Document Object Model* (DOM) do HTML5 e outras características do JavaScript.

São exemplos deste tipo de bibliotecas JavaScript:

- jQuery [jQu1], [Richardson 2012]
- Zepto [Zep]
- XUI [XUI]
- ChocolateChip [ChU]
- Mobify.js [Mob1]

2.3 Camada de criação de *widgets*

Por fim, formulários Web (tal como exibidos por elementos de formulários do HTML5) têm um aspecto *(look-and-feel)* diferente dos formulários nativos do dispositivo. Algumas bibliotecas CSS3/JavaScript oferecem formas de gerar facilmente elementos de formulários *(widgets)* com aspecto nativo, em JavaScript.

São exemplos deste tipo de bibliotecas JavaScript:

- jQueryMobile [jQu2], [Richardson 2012]
- jQT [jQT], [Allen et al. 2010], anteriormente conhecido por jQTouch
- Jo [Jo2]
- WebApp.net [Web]
- The M Project [MPr]
- Wink Toolkit [Win]
- ChocolateChip-UI [ChU] também conhecida como ChUI
- Universal iPhone UI Kit (UiUIKit) [UiU]
- iWebKit [iWe], [Allen et al. 2010]
- qooxdoo [Qoo]
- iUI [iUI]
- CNET iPhone UI (CiUI) [CiU]
- Dojo Mobile [Doj]
- Kendo UI Mobile [Ken]
- DHTMLX Touch [DHT]

2.4 IDEs

Algumas *frameworks* incluem os seus próprios ambientes integrados de desenvolvimento *(Integrated Development Environments: IDEs)*. Isto permite que os programadores desenhem aplicações simplesmente arrastando elementos com o rato *(drag-and-drop)*, que tenham realce *(highlighting)* de sintaxe e/ou ferramentas de depuração *(debugging)*.

São exemplos deste tipo de IDEs:

- ApplicationCraft [App2]. Usa PhoneGap.
- HandheldDesigner [Han]. Apenas para iOS com a biblioteca jQuery Mobile.
- Icenium [Ice]. Usa PhoneGap.

2.5 *Mobile Enterprise Application Platforms* (MEAP)

A firma Gartner criou o termo Plataformas de Aplicações Empresariais Móveis (MEAP) [King et al. 2009], [King e Clark 2011]. MEAP especifica dois componentes para permitir o desenvolvimento multiplataforma:

Servidor *middleware* móvel. Este é o elemento unificador que converte os serviços empresariais existentes num protocolo que os clientes móveis conseguem compreender.

Aplicação cliente móvel. Esta aplicação corre em cada dispositivo móvel e liga-se ao servidor *middleware* móvel. Exibe a interface do utilizador e corre alguma lógica de negócio. Pode ser implementada como uma aplicação nativa ou usando o *browser* do dispositivo.

A maior parte destas plataformas na realidade usa o PhoneGap (ou uma *framework* WebView semelhante – i.e., o *browser* do dispositivo móvel) como tecnologia subjacente.

São exemplos deste tipo de plataformas:

- 5app Javascript Communications Library [Fiv]
- Antenna AMP Client [Ant]
- Appear IQ [App1]
- Convertigo Mobilizer [Con], [Picciotto 2012]
- IBM Worklight [IBM]
- Mendix App Platform [Men]
- MobiOne [Mob2]
- Motocol Platform [Mot]
- Motorola RhoMobile Suite [Rho], [Allen et al. 2010]
- Nexcore Mobile Platform [Nex]
- NSB/AppStudio [NSB]
- OpenMEAP [Ope]
- Outsystems [Out]
- SAP Mobile [SAP]
- Servoy Mobile [Ser]
- Verivo Enterprise Mobility Platform [Ver]

3 *Frameworks* baseadas em linguagens interpretadas / *Just-in-Time (JIT)*

Estas são *frameworks* que usam os interpretadores ou compiladores JIT incorporados nas aplicações ou nos sistemas operativos subjacentes para correr o mesmo código fonte em múltiplas plataformas. Elas deixam para trás o *browser* com a sua interpretação e descodificação de HTML5 e CSS3 e focam-se apenas na linguagem de *scripting* (normalmente JavaScript, mas Lua também é comum nestas *frameworks)*. Estas *frameworks* são computacionalmente mais leves, mas com a falta de uma camada de descrição da disposição visual (HTML5 e CSS3) elas necessitam de uma API multiplataforma específica. Esta API é semelhante, mas não igual, a

cada uma das APIs das plataformas, necessitando assim de uma biblioteca subjacente para converter as chamadas entre APIs.

Estas bibliotecas e sistemas ponte entre linguagens para interface com as APIs nativas no entanto, ainda causam uma carga computacional significativa.

São exemplos deste tipo de *frameworks:*

- TitaniumSDK [Tit], [Allen et al. 2010]
- Sencha Touch [Sen], [Allen et al. 2010]
- Corona SDK [Cor]
- MOAI SDK [MOA]
- ShiVa3D [Shi]
- KonyOne Platform [Kon]

4 *Frameworks* baseadas em linguagens com máquinas virtuais (VMs)

Interpretar linguagens, ou compilá-las *just-in-time*, consome recursos computacionais que podem ser necessários a uma aplicação ou para o tempo de uso (bateria) de um dispositivo. Algumas *frameworks* resolvem esta questão usando uma linguagem baseada em máquina virtual *(Virtual Machine: VM)* como Java ou o *Common Language Runtime* (CLR) do .NET.

São exemplos deste tipo de *frameworks:*

- Xamarin [Xam]
- Unity [Uni]

5 *Frameworks* baseadas ficheiros binários compilados

Um uso ainda mais eficiente de recursos computacionais é compilar as aplicações adiantadamente *(Ahead-of-Time: AOT)*, i.e. durante o desenvolvimento, para que aquilo que é instalado no dispositivo e utilizado seja código optimizado [Stilkerich et al. 2012]. Linguagens típicas de compilação AOT para desenvolvimento móvel são C, C++ e Objective C.

Contudo, diferentes plataformas móveis usam linguagens nativas diferentes. Ou algum tipo de ponte entre linguagens é necessária para aceder às APIs da plataforma ou será necessário desenvolver código (nativo) diferente para cada plataforma de forma a suportar algumas das características da plataforma, eliminando assim a vantagem de uma *framework* multiplataforma. Mais ainda, outros investigadores chegaram à conclusão que desenvolver em C/C++ tem uma taxa mais alta de bugs (por um factor de até 2) do que desenvolver em Java [Phipps 1999], [Vivanco e Pizzi 2002], [Vivanco e Pizzi 2005], [English e McCreanor 2009], [Bhattacharya e Neamtiu 2011], pelo que apesar destas *frameworks* serem das mais eficientes computacionalmente, também são as que têm maior dificuldade e maiores taxas de erros *(bugs)*.

São exemplos deste tipo de *frameworks:*

- Qt 5 [Qt2]
- V-Play Game Engine [VPl]
- Apache Flex [Apa]
- Adobe AIR [Tucker et al. 2008]
- Microsoft Silverlight [Moroney 2009], [Moroney 2010]
- JavaFX [Clarke et al. 2009]
- Google Web Toolkit (GWT) [Goo], [Marinacci 2012]
- GWT Mobile WebKit [GWT]
- Marmalade Juice [Mar1]
- XMLVM [XML], [Puder 2010]

6 Plataformas sem escrita de código e IDEs

Estas plataformas dispensam a escrita de código. Em vez disso, a aplicação é prototipada com recurso ao arrasto visual *(drag-and-drop)* de objetos, desenhar fluxos de trabalho *(workflows)* e configurar propriedades de objetos (cores, conjuntos de dados, etc.). Uma aplicação nativa em cada plataforma irá então interpretar um ficheiro proprietário que contém estes objetos e fluxos de trabalho.

Embora não sejam *frameworks* em si, estas plataformas são relevantes ao desenvolvimento multiplataforma pois resolvem o problema recorrendo à codificação visual da aplicação.

O desenvolvimento neste tipo de plataformas é habitualmente muito rápido e simples. Mas a sua configurabilidade implicitamente limitada torna-as restritas a alguns casos de uso específicos, e a sua carga computacional está intrinsecamente dependente da expressividade dos ficheiros proprietários.

São exemplos deste tipo de plataformas:

- Magic xpa Application Platform [Mag]
- TapLynx [Tap]
- eMobc [eMo]
- ViziApps[Viz]
- 7 Weptun AppConKit [App3]
- Marmalade Quick [Mar2]
- DragonRAD [Dra]
- Enprovia Mobile Enabler [Enp]

7 Conclusões

Nos últimos anos assistimos a uma explosão no número de *frameworks* de desenvolvimento móvel. Estas tornaram-se extremamente sofisticadas: algumas são baseadas em tecnologias computacionalmente eficientes enquanto outras aproveitam tecnologias de desenvolvimento simples para garantir um tempo de implementação muito curto.

A escolha de *framework* de desenvolvimento móvel por parte de uma organização como o Estado estará dependente de múltiplos fatores. No entanto, sem a previsão de que um único fabricante e sistema operativo móvel dominem o mercado, o uso de uma *framework* destas é inevitável.

Na nossa revisão de literatura, verificamos que não existem *frameworks* que cumpram todos os objetivos enunciados na secção 1.2. PhoneGap é uma das *frameworks* mais populares e o Marmalade SDK e o Qt 5, assim como algumas plataformas sem escrita de código, são os que mostram as maiores promessas em termos de carga computacional da aplicação final. Contudo, todas obrigam a algum compromisso de um dos vetores de análise, obrigando a escolha de uma *framework* ter de pesar uma boa experiência de utilizador versus um baixo custo de desenvolvimento.

8 Referências

[Fiv] *5app Javascript Communications Library*. http://www.fiveapp.com/products/library.

[Ant] *Antenna AMP Client.* http://www.antennasoftware.com/products/amp-client.

[Apa] *Apache Flex.* http://flex.apache.org.

[App1] *Appear.* http://www.appearnetworks.com/create-and-manage/.

[App2] *ApplicationCraft.* http://www.applicationcraft.com.

[ChU] *ChocolateChip-UI (ChUI).* http://chocolatechip-ui.com.

[CiU] *CNET iPhone UI (CiUI).* http://code.google.com/p/ciui-dev/.

[Con] *Convertigo Mobilizer.* http://www.convertigo.com/en/crm/convertigo-mobilizer.html.

[Cor] *Corona SDK.* http://www.coronalabs.com.

[DHT] *DHTMLX Touch.* http://www.dhtmlx.com/touch/.

[Doj] *Dojo Mobile.* http://dojotoolkit.org/features/mobile.

[Dra] *DragonRAD.* http://dragonrad.com.

[eMo] *eMobc.* http://www.emobc.com.

[Enp] *Enprovia Mobile Enabler.* http://www.enprovia.com/products/mobile/enabler/index.html.

[Goo] *Google Web Toolkit (GWT).* https://developers.google.com/web-toolkit/.

[GWT] *GWT Mobile WebKit.* http://code.google.com/p/gwt-mobile-webkit/.

[Han] *Handheld Designer.* http://handhelddesigner.com.

[IBM] *IBM Worklight.* http://www.ibm.com/software/products/us/en/worklight.

[Ice] *Icenium.* http://www.icenium.com.

[iUI] *iUI.* http://www.iui-js.org.

[iWe] *iWebKit.* http://snippetspace.com/portfolio/iwebkit/.

[Jo2] *Jo.* http://joapp.com.

[jQT] *jQT.* http://jqtjs.com.

[jQu1] *jQuery.* http://jquery.com.

[jQu2] *jQuery Mobile.* http://jquerymobile.com.

[Ken] *Kendo UI Mobile.* http://www.kendoui.com/mobile.aspx.

[Kon] *KonyOne Platform.* http://www.kony.com/apps/build.

[Mag] *Magic xpa Application Platform.* http://www.magicsoftware.com/magic-xpa-application-platform.

[Mar1] *Marmalade Juice.* http://www.madewithmarmalade.com/juice.

[Mar2] *Marmalade Quick.* http://www.madewithmarmalade.com/quick.

[Mar3] *Marmalade SDK.* http://www.madewithmarmalade.com/sdk.

[Men] *MendixAppPlatform.* http://www.mendix.com/application-platform-as-a-service/.

[MOA] *MOAI SDK.* http://getmoai.com/products/sdk/.

[Mob1] *Mobify.js.* http://www.mobify.com/mobifyjs/.

[Mob2] *MobiOne.* http://www.genuitec.com/mobile/.

[Mot] *Motocol Platform.* http://motocol.com/platform.

[Rho] *Motorola RhoMobile Suite.* http://www.motorolasolutions.com/US-EN/Business+Product+and+Services/Software+and+Applications/RhoMobile+Suite.

[Nex] *Nexcore Mobile Platform.* http://nexcore.skcc.com/mobile/html/nmp/nmp.html.

[NSB] *NSB/AppStudio.* http://www.nsbasic.com/app/.

[Ope] *OpenMEAP*. http://www.openmeap.com/.

[Out] *Outsystems*. http://www.outsystems.com/.

[Pho] *PhoneGap (Apache Cordova)*. http://phonegap.com.

[Qoo] *qooxdoo*. http://qooxdoo.org.

[Qt2] *Qt*. http://qt-project.org.

[SAP] *SAP Mobile*. http://www.sap.com/pc/tech/mobile/software/solutions/overview/index.html.

[Sen] *Sencha Touch*. http://www.sencha.com/products/touch/.

[Ser] *Servoy Mobile*. http://www.servoy.com/.

[Shi] *ShiVa3D*. http://www.shivaengine.com.

[Tap] *TapLynx*. http://www.taplynx.com.

[MPr] *The M Project*. http://www.the-m-project.org.

[Tit] *Titanium SDK*. http://www.appcelerator.com/platform/titanium-sdk/.

[Tri] *Trigger.io*. https://trigger.io.

[Uni] *Unity*. http://unity3d.com.

[UiU] *Universal iPhone UI Kit (UiUIKit)*. http://code.google.com/p/iphone-universal/.

[VPl] *V-Play Game Engine*. http://v-play.net.

[Ver] *Verivo Enterprise Mobility Platform*. http://www.verivo.com/mobile-platform/overview/.

[Viz] *ViziApps*. http://www.viziapps.com.

[Web] *WebApp.net*. http://trywebapp.net.

[App3] *Weptun AppConKit*. http://www.weptun.de/appconkit/architektur/.

[Win] *Wink Toolkit*. http://www.winktoolkit.org.

[Xam] *Xamarin*. http://xamarin.com.

[XML] *XMLVM*. http://xmlvm.org/overview/.

[XUI] *XUI*. http://xuijs.com.

[Zep] *Zepto*. http://zeptojs.com.

[Allen et al., 2010] Allen, S., Graupera, V., e Lundrigan, L. (2010). *Pro Smartphone Cross-Platform Development*. Apress.

[Bhattacharya e Neamtiu, 2011] Bhattacharya, P. e Neamtiu, I. (2011). "Assessing Programming Language Impact on Development and Maintenance." *Proceeding of the 33rd international conference on Software engineering - ICSE '11*, (2):171.

[Charland e Leroux, 2011] Charland, A. e Leroux, B. (2011). "Mobile Application Development: Web vs. Native." *Communications of the ACM*, 54(5):49–53.

[Clarke et al., 2009] Clarke, J., Connors, J., e Bruno, E. J. (2009). *JavaFX: Developing Rich Internet Applications*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition.

[English e McCreanor, 2009] English, M. e McCreanor, P. (2009). "Exploring the Differing Usages of Programming Language Features in Systems Developed in C++ and Java." *Technical report, University of Limerick*.

[Hartmann et al., 2011] Hartmann, G., Stead, G., e DeGani, A. (2011). "Cross-Platform Mobile Development." *Technical Report March.*

[King e Clark, 2011] King, M. J. e Clark, W. (2011). "Magic Quadrant for Mobile Enterprise Application Platforms." *Technical report.*

[King et al., 2009] King, M. J., Clark, W., e Jones, N. (2009). "Magic Quadrant for Mobile Consumer Application Platforms." *Technical report.*

[Marinacci, 2012] Marinacci, J. (2012). *Building Mobile Applications with Java: Using the Google Web Toolkit and PhoneGap.* O'Reilly Media, Inc.

[Mikkonen et al., 2009] Mikkonen, T., Taivalsaari, A., e Terho, M. (2009). "Lively for Qt: A Platform for Mobile Web Applications." *In Proceedings of the 6th International Conference on Mobile Technology, Application & Systems - Mobility '09*, artigo nº 24.

[Moroney, 2009] Moroney, L. (2009). *Introducing Microsoft Silverlight 3*. Microsoft Press, 3rd edition.

[Moroney, 2010] Moroney, L. (2010). *Microsoft Silverlight 4 Step by Step.* Microsoft Press, 1st edition.

[Phipps,1999] Phipps, G. (1999). "Comparing Observed Bug and Productivity Rates for Java and C++." *Software Practice and Experience*, 29(Janeiro de 1998):345–358.

[Picciotto, 2012] Picciotto, O. (2012). *Solve Your Many-Device-to-Many-Platform Mobile Application Integration Challenges.* http://www.ibm.com/developerworks/cloud/library/cl-mobileapplatform/index.html

[Puder, 2010] Puder, A. (2010). "Cross-Compiling Android Applications to the iPhone." *In Proceedings of the 8th International Conference on the Principles and Practice of Programming in Java - PPPJ '10*, page 69, New York, New York, USA. ACM Press.

[Raj e Tolety, 2012] Raj, R. e Tolety, S. B. (2012). "A Study on Approaches to Build Cross-Platform Mobile Applications and Criteria to Select Appropriate Approach." *In India Conference (INDICON), 2012 Annual IEEE*, pages 625–629, Kochi, India.

[Rajendran, 2013] Rajendran, D. K. (2013). *Attaining Uniformity in User Interfaces across Mobile Platforms - A Developer's Perspective.* Masters, Dalhousie University.

[Ribeiro e da Silva, 2012] Ribeiro, A. e da Silva, A. R. (2012). "Survey on Cross-Platforms and Languages for Mobile Apps." *In 2012 Eighth International Conference on the Quality of Information and Communications Technology*, pages 255–260.

[Richardson, 2012] Richardson, C. (2012). *Lucast Cross Platform Development with Web Technologies.* PhD thesis.

[Singh e Palmieri, 2011] Singh, I. e Palmieri, M. (2011). "Comparison of Cross-Platform Mobile Development Tools." *IDT: Malardalen University.*

[Stilkerich et al., 2012] Stilkerich, M., Thomm, I., Wawersich, C., e Schro¨der-Preikschat, W. (2012). "Tailor-made JVMs for Statically Configured Embedded Systems." *Concurrency and Computation: Practice and Experience*, 24(8):789–812.

[Taivalsaari et al., 2008] Taivalsaari, A., Mikkonen, T., Ingalls, D., e Palacz, K. (2008). "Web Browser as an Application Platform: The Lively Kernel Experience." *Technical report, Sun Microsystems, Inc.*, Mountain View, CA, USA.

[Thomson, 2012] Thomson, G. (2012). "BYOD: Enabling the Chaos." *Network Security,* 2012(2):5–8.

[Tucker et al., 2008] Tucker, D., Casario, M., Weggheleire, K. D., e Tretola, R. (2008). *Adobe AIR 1.5 Cookbook: Solutions and Examples for Rich Internet Application Developers.* O'Reilly Media, Sebastopol, CA, USA, 1st edition.

[Vivanco e Pizzi, 2002] Vivanco, R. e Pizzi, N. (2002). "Computational Performance of Java and C++ in Processing Large Biomedical Datasets." *In Electrical and Computer Engineering, 2002 - IEEE CCECE 2002, Canadian Conference on*, pages 691–696.

[Vivanco e Pizzi, 2005] Vivanco, R. A. e Pizzi, N. J. (2005). "Scientific Computing with Java and C++: A Case Study Using Functional Magnetic Resonance Neuroimages." *Software: Practice and Experience*, 35(3):237–254.

[Anjos, 2010] Anjos, André (2010). *m-Gov: Administração Pública em Mobilidade.* http://comunidades.ina.pt/content/m-gov-administração-pública-em-mobilidade.

[WMT] *Ciclo de Workshops "A Mobilidade e as Tecnologias".* http://comunidades.ina.pt/content/ciclo-de-workshops-mobilidade-e-tecnologias.