Universidade do Minho
Mestrado em Engenharia Informática

Tópicos de Desenvolvimento de Software

Projeto de Pesquisa - Tecnologias de Desenvolvimento Cross-Platform

Grupo
Inês Ferreira - PG53879
Marta Sá - PG54084

8 de março de 2024

Conteúdo

1 Introdução

Este trabalho de pesquisa foi redigido no âmbito da unidade curricular de Tópicos de Desenvolvimento de Software e tem como principal objetivo estudar noções relacionadas com tecnologias de desenvolvimento *cross-platform*.

Em desenvolvimento de *software*, com o surgimento de múltiplas plataformas, tornou-se inviável o desenvolvimento de uma aplicação separada para cada uma delas, dado que esta abordagem envolve mais custos e tempo de desenvolvimento.

Desta forma, o desenvolvimento *cross-platform* é atualmente muito utilizado, uma vez que permite ao programador o desenvolvimento de aplicações que possam ser executadas em múltiplas plataformas, sem a necessidade de reescrever o código para cada uma delas.

Assim, ao longo deste trabalho iremos apresentar e analisar as características da *framework* Xamarin, nomeadamente, a sua arquitetura, as suas vantagens e desvantagens e a sua aplicabilidade. Por fim, o grupo irá comparar a *framework* escolhida com outras *frameworks* a nível, principalmente, de consumo de energia.

2 Xamarin

Ao longo deste capítulo iremos contextualizar e detalhar os conceitos da *framework* **Xamarin**. Selecionamos esta *framework*. uma vez que é das mais conhecidas e utilizadas no desenvolvimento de software multiplataforma.

2.1 Contextualização

O Xamarin é uma plataforma *open-source* que permite o desenvolvimento de aplicações para iOS, Android e Windows utilizando a plataforma .NET. Criada em 2011, a *framework* é uma das mais populares para o desenvolvimento de aplicações, sendo que faz atualmente parte da Microsoft. De facto, a sua aquisição foi incentivada pela sua capacidade de desenvolver *software* para múltiplas plataformas a partir do *Microsoft Visual Studio*, permitindo aos programadores com conhecimentos em .Net e C# o desenvolvimento de aplicações com desempenho e interface nativas.

2.2 Conhecimento Prévio

No que toca ao processo de aprendizagem, o Xamarin não exige necessariamente nenhum conhecimento prévio sobre outras *frameworks* ou linguagens de programação. No entanto, dado que esta *framework* está integrada na plataforma .NET e utiliza a linguagem de programação C#, poderá ser útil para programador adquirir conhecimentos acerca destas ferramentas.

No entanto, o Xamarin possui uma documentação detalhada acerca do seu funcionamento, bem como um conjunto diverso de tutorias e de exemplos práticos que demonstram a criação de aplicativos móveis em diferentes plataformas, pelo que o processo de aprendizagem poderá ser facilitado.

2.3 Tipo de Software Development Approach

O Xamarin é uma *framework* capaz de desenvolver aplicações através de uma compilação *cross-compiled*, uma vez que as aplicações são compiladas de forma nativa através da criação de uma versão especifica para cada plataforma alvo.

Esta abordagem apresenta algumas vantagens, nomeadamente, a possibilidade de aceder a APIs nativas, de apresentar, geralmente, uma boa performance e de ser independente de uma plataforma. No entanto, uma compilação *cross-compiled* pode apresentar algumas limitações no que toca, por exemplo, às dependências especificas de cada plataforma, à complexidade de teste e de *debugging* da aplicação e ao tempo de compilação.

2.4 Arquitetura

O diagrama da Figura 1 mostra a arquitetura geral de uma aplicação *cross-platform* Xamarin. O Xamarin permite criar uma user interface nativa em cada plataforma e escrever a lógica de negócio em C# que é, por sua vez, compartilhada entre plataformas, possibilitando a partilha de até 80% do código da aplicação. Para além disso, o Xamarin é criado com base no .NET, que lida automaticamente com tarefas como alocação de memória, recolha de lixo e interoperabilidade com plataformas subjacentes.

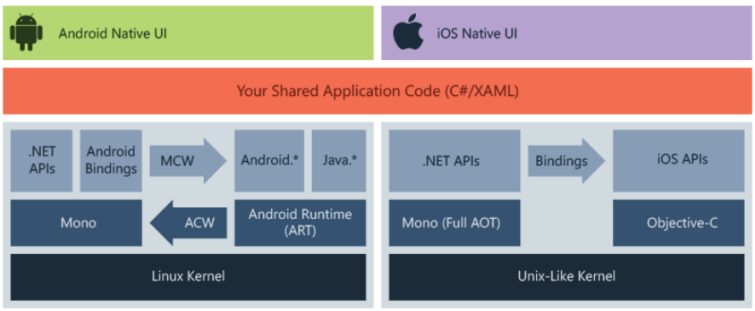

Figura 1: Arquitetura Xamarin.

2.4.1 Xamarin Android

As aplicações Xamarin.Android compilam de C# para uma IL (linguagem intermediária) que passa de seguida por uma compilação JIT (Just-In-Time) para assembly nativo quando a aplicação é iniciada. As aplicações Xamarin.Android são executadas no ambiente de execução Mono, lado a lado com a máquina virtual ART (Android Runtime). Para além disso, o Xamarin fornece .NET *bindings* para os *namespaces* Android.* e Java.*. Por sua vez, o ambiente de execução Mono chama estes *namespaces* por meio de MCW (Managed Callable Wrappers) e fornece ACW (Android Callable Wrappers) para o ART, permitindo que ambos os ambientes invoquem o código entre si.

2.4.2 Xamarin iOS

As aplicações Xamarin.iOS correm dentro do Mono que, por sua vez corre lado a lado com o Objective-C Runtime, e são completamente compiladas Ahead-of-Time (AOT) de C# para código assembly nativo ARM.

Para além disso, o Xamarin usa *Selectors* para expor o Objective-C ao managed C# e *Registrars* para expor o managed C# a Objective-C. Desta forma, *Selectors* e *Registrars* são coletivamente chamados de "*bindings*"e permitem ao Objective-C e ao C# comunicar.

Ambos os ambientes runtime correm em cima de um kernel UNIX, especificamente XNU.

2.5 Vantagens e Limitações

A utilização do Xamarin apresenta diferentes vantagens tais como:

- **Compatibilidade de código entre plataformas** - o facto de o código escrito em Xamarin poder ser usado em diferentes plataformas faz com que haja menos gastos em termos de tempo, dinheiro e esforço;
- **Manutenção simplificada** - uma vez que o mesmo código ser usado para diferentes plataformas resulta numa manutenção que requer menos tempo e esforço;
- **Suporte técnico da Microsoft** - uma vez que é uma tecnologia integrada na Microsoft, o Xamarin possui documentação, cursos e tutoriais muito abrangentes para ajudar os programadores;
- **Experiência do usuário nativo** - dado que o Xamarin garante acesso a kits de ferramentas nativas e APIs usadas para diferentes plataformas (Windows, iOS e Android). Para além disso, oferece desempenho e design nativos para cada aplicação desenvolvida com o mesmo.

Apesar de todas estas vantagens, como seria de esperar, o Xamarin apresenta também algumas desvantagens,nomeadamente:

- **Atualizações atrasadas** - pois quer o iOS como o Android realizam atualizações regularmente e os desenvolvedores do Xamarin não podem utilizar as mesmas imediatamente após o seu lançamento. Uma vez que a integração de novos recursos no ecosistema Xamarin requer algum tempo, pelo que um atraso é causado;
- **Aplicações pesadas** - aplicações desenvolvidas com a *framework* Xamarim tendem a ser maiores dado que é necessário lidar com diversas funcionalidades e bibliotecas, o que pode tornar o Xamarin inadequado para determinados projetos. De facto, podemos verificar na figura abaixo o tamanho total que uma aplicação "Hello, World!"de 6KB ocupa;

Figura 2: Aplicação "Hello, world!".

- **Comunidade Pequena** - devido à comunidade Xamarin ser pequena às vezes é difícil encontrar ajuda em novas tecnologias e recursos da comunidade.

2.6 Casos de Uso

Relativamente a casos de uso, o Xamarin é uma *framework* que integra vários recursos e que permitem a sua utilização nos seguintes contextos:

- acesso a funcionalidades nativas, tais tais como, ecrã de bloqueio, acelerómetro e gestor de ficheiros através das APIs *cross-platform* que biblioteca *Xamarin.Essentials* providencia;
- desenvolvimento de UI para iOS e Android com funcionalidades nativas a partir de código partilhado através da plataforma *Xamarin.Forms*, permitindo que um programador partilhe, em média, 90% da sua aplicação em múltiplas plataformas;
- desenvolvimento de aplicações que integram *features* como XML, Base de Dados, suporte a *networks*, entre outras, a partir de uma Biblioteca de Classes Base (BCL).

2.7 Aplicabilidade

O Xamarin é aplicável a, principalmente, aplicações móveis e a projetos que necessitam de uma maneira fácil de compartilhar código entre diferentes plataformas. Para além disso, o Xamarin poderá ser utilizado quando o intuito é desenvolver uma aplicação nativa permitindo aos *end users* uma experiência familiar com a aplicação.

Por outro lado, o Xamarin possui algumas limitações que impedem a sua aplicabilidade em projetos com as seguintes características:

- requerem a utilização de ferramentas e bibliotecas de terceiros, dado que poderá haver incompatibilidades;
- requerem uma baixa ocupação no dispositivo, já que as aplicações ocupam tipicamente mais espaço do que as nativas devido, principalmente, às bibliotecas associadas;
- requerem um elevado e complexo processamento gráfico, uma vez que neste caso uma aplicação nativa poderá ter um melhor desempenho.

2.8 Sistemas e Aplicações Que Utilizam o Xamarin

Existem diversas aplicações que utilizam o Xamarin, neste tópico iremos referir algumas delas:

- **Alaska Airlines** - Esta aplicação trata-se de um guia de viagens aéreas que permite resolver problemas relacionados a informações do voo desde a reserva das passagens, check-in, seleção de assento, mudança de assento, pedido de comida, etc.

- **Microsoft news** - Trata-se de um site de noticias que disponibiliza noticias de diversas categorias, apresenta uma boa user interface e um download gratuito.
- **UPS Mobile** - Consiste numa aplicação que permite rastrear encomendas desde o envio até à entrega. Para além disso, esta aplicação permite estimar os custos e o tempo de envio, encontrar os pontos de recolha UPS mais próximos e trata-se de uma aplicação fácil de usar.
- **Outback** - Trata-se de uma app criada para o restaurante *Outback* que permite preencher previamente os detalhes do pedido junto com o endereço de finalização da compra para fazer pedidos de entrega ao domicilio.

2.9 Comparação com Outras Abordagens

Nesta secção, iremos discutir e comparar a *framework* Xamarin com outras *frameworks* a nível do consumo de energia, tempo de execução e utilização de CPU, tendo como base o estudo "Development Frameworks for Mobile Devices: A Comparative Study about Energy Consumption".

O estudo em questão foca-se na análise do impacto de *frameworks* e abordagens de desenvolvimento no consumo de energia de aplicações Android. Desta forma, foram selecionadas as seguintes *frameworks* e respetiva abordagem:

1. Android SDK API 23 (nativa);
2. Apache Cordova versão 7 (híbrida);
3. Appcelerator Titanium versão 5 (interpreted);
4. NativeScript versão 3 (interpreted);
5. Xamarin versão 6 (cross-compilation);
6. Corona versão 2016 (cross-compilation);
7. Android NDK revision 15c (cross-compilation).

Para além disso, para realizar esta análise, foram selecionados três tipos de aplicações, que indicam características presentes frequentemente nas aplicações, com o objetivo de medir o consumo de energia provocado por cada *framework*:

1. aplicações de processamento intensivo;
2. aplicações de reprodução de vídeos;
3. aplicações de reprodução de áudios.

A Figura 3 resume os resultados obtidos durante os testes realizados em aplicações com um elevado processamento, organizados do mais eficiente para o menos eficiente. A coluna "Energy" mostra a média e o desvio padrão do consumo de energia. As restantes colunas, "CPU load" e "Duration" dizem respeito, respetivamente, à percentagem de CPU utilizada pela aplicação e ao seu tempo de execução.

Framework	Energy (mWh.)		CPU load (%)		Duration (s.)	
	$\overline{E}$	S_E	$\overline{C}$	S_C	$\overline{T}$	S_T
Cordova	1.597	0.136	35.924	2.571	8.467	0.679
Titanium	1.692	0.096	37.480	2.395	8.355	0.643
Android NDK	1.789	0.092	32.434	1.876	9.745	0.366
NativeScript	1.792	0.176	33.357	2.217	9.109	1.789
Xamarin	3.036	0.185	32.072	1.768	17.891	0.973
Android SDK	3.463	0.149	32.468	1.332	18.568	2.938
Corona	7.304	0.189	34.315	1.102	38.877	1.492

Figura 3: Aplicação de elevado processamento.

De acordo com os resultados obtidos, podemos verificar a existência de 3 grupos de *frameworks*: O primeiro grupo, e mais energeticamente eficiente, é composto pelas *frameworks* Cordova, Titanium, Native Android NDK e NativeScript. Por sua vez, o Xamarim encontra-se no segundo grupo, com uma eficiência regular, juntamente com o Native Android SDK. Por fim, o grupo menos energeticamente eficiente é composto pela *framework* Corona.

Desta forma, podemos verificar que o Xamarim, relativamente ao Cordova tem um consumo de energia e tempo de execução 2.11 vezes superior, mas uma percentagem de ocupação da CPU 1.12 vezes inferior. Relativamente à *framework* Corona, a menos eficiente, o Xamarim tem um consumo de energia e tempo de execução 2 vezes inferior e uma percentagem de ocupação do CPU 1.06 vezes inferior.

A seguinte figura expões os resultados obtidos durante os testes de aplicações capazes de reproduzir vídeos, mais uma vez, organizados por ordem crescente de consumo de energia.

Framework	Energy (mWh.)		CPU load (%)		Duration (s.)	
	$\overline{E}$	S_E	$\overline{C}$	S_C	$\overline{T}$	S_T
Android SDK	4.776	0.287	14.540	0.862	61.600	0.814
Corona	4.992	0.235	14.704	0.711	62.733	0.907
Xamarin	5.119	0.473	15.465	1.608	62.333	0.959
Titanium	5.262	0.502	15.204	1.643	63.633	1.033
NativeScript	11.112	1.590	17.839	2.210	63.333	1.295
Cordova	13.866	0.536	22.358	0.903	62.833	0.834

Figura 4: Aplicação de reprodução de vídeos.

Assim, é possível verificar a existência de dois grupos, sendo que a *framework* Xamarin pertence àquele que apresenta uma eficiência energética maior. Comparativamente à *framework* Android SDK (nativa), o Xamarin é cerca de 1.1 vezes menos eficiente e, relativamente à *framework* Cordova (híbrida), o Xamarin consegue ser 2.7 vezes mais eficiente.

A Figura 5, por sua vez, mostra os resultados obtidos durante os testes realizados a aplicações de reprodução de áudios. Aqui, podemos verificar que não diferenças muito significativas entre as diferentes *frameworks*, pelo que o Xamarin é apenas 1.02 vezes menos eficiente do que a *framework* Android SDK e 1.3 vezes mais eficiente do que a *framework* Corona.

Framework	Energy (mWh.)		CPU load (%)		Duration (s.)	
	$\overline{E}$	S_E	$\overline{C}$	S_C	$\overline{T}$	S_T
Android SDK	3.920	0.291	10.497	0.882	64.033	0.999
Xamarin	4.010	0.201	10.592	0.613	64.967	1.098
Titanium	4.189	0.277	11.865	0.835	64.767	1.104
NativeScript	4.224	0.229	11.233	0.644	65.867	1.042
Cordova	4.288	0.191	11.473	0.487	65.733	1.388
Corona	5.194	0.387	14.680	1.080	64.800	1.031

Figura 5: Aplicação de reprodução de áudios.

É de notar que, em todos estes resultados, os valores obtidos pela multiplicação de cada valor C pelo respetivo valor de T, mantém praticamente a mesma ordem de resultados que o consumo de energia, pelo que podem ser bons estimadores para o consumo de energia da aplicação, o que acaba por ser mais conveniente dado que ambos os parâmetros são mais facilmente obtidos do que o valor real do consumo de energia.

Desta forma, o Xamarin, uma *framework cross-compilation*, é considerada uma *framework* energeticamente eficiente em dois dos três tipos de aplicações testadas, nomeadamente, nas que reproduzem vídeo e nas que reproduzem áudio. A única *framework* que é considerada eficiente nos três tipos de aplicações é a *interpreted framework* Titanium, pelo que a sua utilização é, de forma geral, mais apropriada em todos os casos de estudo do que o Xamarin.

Para além disso, podemos verificar que, apesar de ser uma *framework* nativa e teoricamente mais eficiente, o *Android SDK* não tem uma vantagem muito relevante relativamente ao Xamarin no que toca a aplicações de reprodução de áudio e de vídeo.

Por fim, os dados analisados sugerem que determinar qual a *framework* mais eficiente no desenvolvimento de aplicações não é tão importante quanto determinar quais as menos eficientes, ou seja, as *frameworks* Cordova (híbrida), Corona (cross-compilation) e NativeScript (interpreted). Isto deve-se ao facto de não haver uma diferença significativa entre a melhor *framework* e as intermédias para cada tipo de aplicação, pelo que poderá ser uma boa opção utilizar a *framework* Xamarin.

3 Conclusão

Concluindo, ferramentas de *cross-platform* como o Xamarin tornaram-se cada vez mais populares devido à sua capacidade de criar aplicações móveis que podem ser executadas em várias plataformas utilizando uma única base de código. O Xamarin oferece uma gama de benefícios em relação a outros *frameworks* de *cross-platform*, incluindo a sua manutenção simplificada, o suporte técnico da Microsoft e o seu desempenho semelhante ao nativo.

Ao usar Xamarin, programadores podem economizar tempo e recursos enquanto fornecem aplicações móveis de alta qualidade que oferecem uma experiência de utilizador perfeita em diferentes dispositivos. O Xamarin pode ainda ser aplicado a uma variedade de casos de uso em vários setores, incluindo serviço de transporte de carga e encomendas, restauração e informação.

No entanto, é importante observar que ferramentas de *cross-platform* como o Xamarim nem sempre são a melhor solução. Dependendo dos requisitos do projeto, o desenvolvimento nativo de aplicações ou até outras *frameworks* de *cross-platform* podem ser mais apropriados. Portanto, é importante que os programadores avaliem cuidadosamente as suas opções e escolham a ferramenta que melhor se adapta às suas necessidades e objetivos específicos.

4 Referências

[1] O que é o Xamarin?, https://learn.microsoft.com/pt-br/xamarin/get-started/what-is-xamarin#how-xamarin-works

[2] Exemplos de código, https://learn.microsoft.com/pt-br/samples/browse/?products=xamarin

[3] Aplicabilidade, https://softjourn.com/insights/xamarin-app-development-advantages-and-disadvantages

[4] Estudo, https://dl.acm.org/doi/abs/10.1145/3197231.3197242

[5] Aplicações que utilizam Xamarin, https://www.redbytes.in/apps-using-xamarin/#13

[6] Vantagens e Limitações, https://blog.back4app.com/pt/xamarin-vs-react-native-segredos-desvendados/