UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS QUIXADÁ

CURSO DE GRADUAÇÃO EM SISTEMAS DE INFORMAÇÃO

DANIEL WOLNEY MENDONÇA DE CASTRO

COMPARAÇÃO DE ARCABOUÇOS PARA DESENVOLVIMENTO DE APLICAÇÕES MÓVEIS MULTIPLATAFORMA NO AMBIENTE CORPORATIVO DE TI DO CEARÁ

QUIXADÁ

2016

DANIEL WOLNEY MENDONÇA DE CASTRO

COMPARAÇÃO DE ARCABOUÇOS PARA DESENVOLVIMENTO DE APLICAÇÕES MÓVEIS MULTIPLATAFORMA NO AMBIENTE CORPORATIVO DE TI DO CEARÁ

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Sistemas de Informação do Campus Quixadá da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de Bacharel em Sistemas de Informação.

Orientador: Prof. Dr. Jefferson de Carvalho Silva

QUIXADÁ

2016

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

C35c Castro, Daniel Wolney Mendonça de.
Comparação de arcabouços para desenvolvimento de aplicações móveis multiplataforma no ambiente de TI do Ceará. / Daniel Wolney Mendonça de Castro. – 2016.
48 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Quixadá, Curso de Sistemas de Informação, Quixadá, 2016.
Orientação: Prof. Dr. Jefferson de Carvalho Silva.

1. Dispositivos móveis.. 2. Framework (Programas de computadores). 3. Android (Programa de computador). 4. Interface de programas aplicativos (Software). I. Título.

CDD 005

DANIEL WOLNEY MENDONÇA DE CASTRO

COMPARAÇÃO DE ARCABOUÇOS PARA DESENVOLVIMENTO DE APLICAÇÕES MÓVEIS MULTIPLATAFORMA NO AMBIENTE CORPORATIVO DE TI DO CEARÁ

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Sistemas de Informação do Campus Quixadá da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de Bacharel em Sistemas de Informação.

Aprovada em:

BANCA EXAMINADORA

Prof. Dr. Jefferson de Carvalho Silva (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. Fábio Carlos Sousa Dias
Universidade Federal do Ceará (UFC)

Prof. Me. Jeferson Kenedy Morais Vieira
Universidade Federal do Ceará (UFC)

À toda Mundiça do Paulo.

AGRADECIMENTOS

Aos meus pais e à minha avó Marlene, pelo seu amor incondicional que cuidou de mim até aqui. Ao amor da minha vida, minha noiva Samylla, por apenas existir em minha vida. Ao meu grande amigo Marcelo Cabral, que sempre me deu apoio em tudo e que esteve comigo em todas as noites de estudo anteriores às provas. Ao Mardson, William, Paulo Filipe e Guylherme, amigos que também me ajudaram de maneira direta. À todos os amigos que fiz na graduação. À meu orientador, Jefferson Carvalho.

"Isto deve ser dito: há no mundo excesso de grandes homens. Há legisladores demais, organizadores, fundadores de sociedades, condutores de povos, pais de nações, etc. Gente demais se coloca acima da humanidade para regê-la, gente demais para se ocupar dela."

(Frédéric Bastiat)

RESUMO

O presente trabalho consiste em realizarmos uma análise comparativa dos dois frameworks para desenvolvimento multiplataforma mais utilizados por empresas do mercado de TI no Ceará. Foi desenvolvido uma aplicação utilizando cada framework como prova de conceito, e depois, realizado a comparação de acordo com métricas software discorridas. O levantamento desses frameworks mais populares foi realizado através de uma pesquisa com empresas que mantém projetos de software no estado do Ceará. Os resultados deste trabalho são destinados à essas empresas e à comunidade de desenvolvedores independentes.

Palavras-chave: Comparação. Aplicações Móveis. Multiplataforma. Android. Qt. Ionic.

ABSTRACT

The present research consists of performing a comparative analysis of the two multiplatform development frameworks most used by companies in the IT market in Ceará. An application was developed using each framework as proof of concept, and then we performed the comparison according to software metrics. The choose of these most popular frameworks was carried out through survey with companies that maintains software projects in the state of Ceará. The results of this work are intended for these companies and the independent community developers.

Keywords: Comparison. Mobile Applications. Cross-Platform. Android. Qt. Ionic.

LISTA DE ILUSTRAÇÕES

Figura 1 – Camadas de um ambiente com MOS. 17
Figura 2 – Camadas do iOS. 20
Figura 3 – Evolução do uso dos *Android's*. 21
Figura 4 – Infraestrutura nativa de desenvolvimento 23
Figura 5 – Infraestrutura de Web Apps. 24
Figura 6 – Estrutura de aplicação híbrida 26
Figura 7 – *Frameworks* utilizados no mercado de TI no Ceará. 35
Figura 8 – Tela de Login da Aplicação Qt. 36
Figura 9 – Tela de Notas existentes. 37
Figura 10 – Criação de Nota. 37
Figura 11 – Menu da Aplicação. 38
Figura 12 – Tela de Login da Aplicação Ionic. 38
Figura 13 – Tela de Notas existentes Ionic. 39
Figura 14 – Criação de Nota Ionic. 39
Figura 15 – Menu Aplicação Ionic. 40
Figura 16 – Tempo de inicialização das aplicações 42
Figura 17 – Consumo de memória RAM 43

LISTA DE TABELAS

Tabela 1 – Qt Essentials 29
Tabela 2 – Tamanhos das aplicações 41
Tabela 3 – Tempo de inicialização das aplicações 42
Tabela 4 – Consumo de memória RAM 43

SUMÁRIO

1 INTRODUÇÃO 14
2 SISTEMAS OPERACIONAIS MÓVEIS 17
2.1 Symbian 18
2.2 iOS 19
2.3 Android 20
2.4 Windows Phone 22
3 ABORDAGEM DE DESENVOLVIMENTO CROSS-PLATFORM . . . 23
3.1 Aplicações nativas 23
3.2 Web Apps 24
3.3 Aplicações híbridas 25
4 *FRAMEWORKS* DE DESENVOLVIMENTO *CROSS-PLATFORM* . . . 27
4.1 Qt 27
4.1.1 Histórico 27
4.1.2 Módulos 28
4.2 Ionic *Framework* 29
4.2.1 Ferramentas 30
5 MATERIAIS E MÉTODOS 31
5.1 Elaboração do questionário 31
5.2 Aplicação da pesquisa 31
5.3 Desenvolvimento da Prova de Conceito 32
5.3.1 Aplicação Server-side 32
5.3.2 Aplicação Client-side 32
5.4 Métricas para comparação 33
5.5 Avaliação de acordo com as métricas 33
5.6 Análise dos dados 34
6 RESULTADOS 35
6.0.1 Resultado da pesquisa com as empresas 35
6.0.2 Aplicação Qt 36
6.0.3 Aplicação Ionic 37
6.0.4 Comparativo das métricas 40

6.0.4.1 Tamanho da Aplicação 40
6.0.4.2 Performance da Aplicação 41
7 CONCLUSÕES E TRABALHOS FUTUROS 44
REFERÊNCIAS 45
APÊNDICES 47

1 INTRODUÇÃO

Não importa o lugar, nem o horário, existe uma grande probabilidade de que alguém ao seu lado esteja usando um *smartphone*. Há uma década atrás, falar algo desse tipo seria um absurdo, entretanto hoje absurdo seria imaginar o contrário. O uso dos dispositivos móveis se tornou comum no dia a dia das pessoas. Eles são usados para acessar redes sociais, verificar *e-mails*, fazer transações bancárias, dentre outros serviços. Atualmente no Brasil, temos aproximadamente 160 milhões de aparelhos *smartphones* sendo usados. Isso representa um crescimento de 9% em relação ao ano anterior, quando foram totalizados cerca de 151 milhões d e aparelhos (MEIRELLES, 2016). Uma pesquisa realizada por Pew Research Center (2015) apontou que 86% dos americanos entre 18 e 29 anos têm *smartphone*.

A demanda por aplicações móveis acompanha o crescimento das vendas de dispositivos móveis. No entanto, o desenvolvimento dessas aplicações não é trivial. Existe um número enorme de aparelhos diferentes que devem ser suportados. Cada dispositivo tem configurações distintas e isso torna o desenvolvimento, a testabilidade e a portabilidade de aplicações mais custosas e dispendiosas (SOUSA, 2014). Então, algumas dúvidas são frequentes. Questionamentos dos tipo: "qual plataforma de desenvolvimento utilizar?", "quais sistemas operacionais móveis são mais utilizados atualmente?", "Como dar suporte a tantos aparelhos e plataformas?"precisam ser levados em conta quando se decide desenvolver uma aplicação móvel.

Existem três estratégias para o desenvolvimento de aplicativos móveis: aplicativo nativo; *Cross-Platform* e site móvel (LUÍS, 2012). O primeiro utiliza uma abordagem mais simples e direta, uma versão do aplicativo é desenvolvida para cada plataforma. Um dos grandes problemas dessa estratégia é que a construção de *n* versões do mesmo aplicativo para várias plataformas, geralmente multiplica o custo do projeto na mesma taxa. Outra opção seria usar *frameworks* (arcabouços) de desenvolvimento de aplicações multiplataforma (*Cross-Platform*). A ideia é basicamente usar um desses *frameworks* para desenvolver uma única aplicação que seja suportada por vários sistemas operacionais móveis. Cada *framework* tem suas próprias peculiaridades no processo de desenvolvimento e limitações nas aplicações resultantes (SOUSA, 2014). As aplicações desenvolvidas nesses *frameworks* geralmente diminuem o custo do projeto, principalmente se for desenvolvido para várias plataformas. No entanto, por ser genérico, o acesso a recursos nativos se torna mais trabalhoso e frequentemente problemático. Por fim, os sites móveis vem ganhando popularização nos últimos anos. A estratégia de ter um site voltado para dispositivos móveis traz maior facilidade na hora de pensar sobre as capacidades de cada

plataforma (TOLEDO; DEUS, 2012).

Este trabalho irá focar na estratégia de *Cross-Platform*. Hoje existem muitos *frameworks cross-plataform* no mercado e quase todos são baseados tecnologias como HTML, CSS e JavaScript. A escolha de um desses *frameworks* é difícil sem haver uma comparação criteriosa entre eles. Palmieri *et al.* (2012) apresentam uma comparação pragmática entre quatro *frameworks Cross-Platform*: *Rhodes*, *PhoneGap*, *DragonRad* e *MoSync*. Um dos principais focos da comparação é fornecer um *overview* sobre a disponibilidade das interfaces de programação, das linguagens e sistemas operacionais suportados, licenças de uso e IDEs disponíveis. Além disso, também apresentam pontos críticos como fator de extensibilidade nas ferramentas e os efeitos que eles podem trazer no mercado. A comparação tinha por objetivo servir como um guia para desenvolvedores fazerem suas escolhas baseadas em suas necessidades.

Dalmasso *et al.* (2013) realizaram uma *survey* para levantar tecnologias a serem comparadas. Embora não exponham detalhes de como a *survey* foi dirigida, eles escolheram as seguintes ferramentas: *PhoneGap*; *Sencha Touch*; *JQuery Mobile* e *Titanium*. Eles desenvolveram uma aplicação *Android* em cada uma dessas ferramentas e depois compararam a solução implementada de acordo com as seguintes métricas: consumo de memória RAM; uso de CPU e consumo de bateria. Charkaoui *et al.* (2014) fizeram um estudo com objetivo de diferenciar e comparar os diferentes tipos de abordagens utilizadas no desenvolvimento de aplicações *Cross-Platform*, de maneira a evidenciar os pontos fortes e fracos em cada abordagem. Os autores não se preocuparam em implementar uma prova de conceito, pois o intuito não era obter *benchmarks*, e sim apenas comparações conceituais.

Sousa (2014) analisou diferenças e limitações de três *frameworks* de desenvolvimento multiplataforma: *Phonegap*; *Sencha Touch* e *Titanium*. O autor comparou funcionalidades oferecidas por cada *framework*, os diferentes modelos de desenvolvimento utilizado por cada um deles e criou uma aplicação para prova de conceito em cada *framework* estudado. Por fim, selecionou medidas de desenvolvimento de software e a partir delas analisou o desempenho das aplicações resultantes do desenvolvimento utilizando os *frameworks* apresentados.

O objetivo deste trabalho é analisar e comparar as diferenças e limitações entre as soluções de desenvolvimento de aplicações móveis multiplataforma com foco no mercado do Ceará. O levantamento foi realizado através de uma pesquisa, utilizando-se de um questionário, encaminhado a empresas que mantém projetos de software no estado do Ceará, afim de obter quais os dois *frameworks* mais utilizados para desenvolvimento multiplataforma por empresas

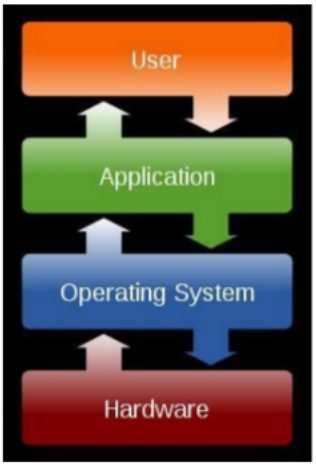
do mercado de TI no estado. Foi desenvolvido uma aplicação utilizando cada *framework* como prova de conceito, e após, realizado a comparação de acordo com métricas software discorridas. O resultado deste trabalho é destinado à essas empresas e à comunidade de desenvolvedores independentes e como resultado final, espera-se obter um documento que servirá de apoio à tomada de decisão no momento da escolha de uma dessas ferramentas.

2 SISTEMAS OPERACIONAIS MÓVEIS

Todos os dispositivos móveis que conhecemos atualmente, *smartphone*, *tablets*, etc, possuem um sistema operacional móvel (*Mobile Operation System* - MOS). Esses MOS's são construídos exclusivamente para esse tipo de dispositivo. Eles são bastante simples, leves e tem como princípio básico a identificação e definição das características e funcionalidades do dispositivo.

Sistemas operacionais móveis são conjuntos de programas cuja função é gerenciar os recursos de *hardware* e *software* para dispositivos móveis, fornecendo uma interface ao usuário final (SILBERSCHATZ *et al.*, 2004). O papel do MOS num dispositivo é análogo ao de um Sistema Operacional num computador, por exemplo. Ele é responsável por fornecer uma abstração que permita o acesso ao *hardware* pelas aplicações, que somente precisam conhecer a API (Application Programming Interface) fornecida para este propósito. Em um ambiente de interação com um MOS, o sistema operacional seria a 3ª camada, como pode ser observado na Figura 1.

Figura 1 – Camadas de um ambiente com MOS.



Fonte – Adaptado de Sousa (2014, p. 6)

Como um exemplo do funcionamento de um MOS num dispositivo, considere o processo de trocas de mensagens de texto entre você e um colega, através dos seus *smartphones*. Uma aplicação móvel presente no seu dispositivo permite que você escreva uma mensagem e envie para o outro dispositivo utilizando algum mecanismo de entrega, ondas de sinal de rádio, por exemplo. Depois que o dispositivo do seu colega recebe o sinal da mensagem, ele então notifica o MOS, que armazena e notifica o aplicativo de mensagens do seu colega.

2.1 Symbian

O *Symbian* foi lançado em 1997 pela *Nokia*. Sua versão mais recente é o *Symbian*3 que foi apresentado em 2010 no *Nokia* N8. Esse sistema foi bastante utilizado durante muito tempo. O *Symbian* foi fruto da parceria de empresas como *Ericsson*, *Motorola* e *Nokia*, algumas das quais tem licença até hoje sobre ele. Tinha como objetivos iniciais manter a integridade e segurança dos dados, trabalhar com recursos escassos e evitar o desperdício de tempo do usuário (SOUZA, 2014).

O *Symbian*3 teve duas atualizações importantes. A primeira em maio de 2011. A versão intitulada como *Symbian Anna*, trouxe algumas melhorias. Ícones de sistema, teclado, navegador de internet e melhorias no desempenho foram algumas das novidades. Além disso, novas versões da suíte de aplicativos Ovi, da *Nokia*, foram introduzidas. A segunda atualização importante foi o lançamento do *Symbian Belle*, em agosto do mesmo ano. Essa versão é considerada a mais significante, principalmente no aspecto visual. O sistema ganhou botões com ícones no lugar de textos, *widgets* nas *homescreens*, telas iniciais e barra de notificações. O *Belle* ainda teve duas atualizações, *Refresh* e FP2.

Pouco tempo após a *Nokia*[1] anunciar uma parceria com a *Microsoft*[2] em fevereiro de 2011, o *Symbian* passou a ser desenvolvido e receber suporte pela *Accenture*[3]. No entanto, em 2013 a *Nokia* decidiu interromper a utilização do sistema em seus aparelhos. O *Nokia* 808 *PureView* foi o último *smartphone* a utilizar o *Symbian*. Ele ganhou destaque no segmento de *cameraphones*, pois tinha como principais atrativos a tecnologia de imagens da *Nokia* e uma câmera potente de 41 *megapixel*. Contudo, acabou perdendo espaço depois do lançamento do *Lumia* 1020 em julho de 2013. Um estudo mais aprofundado sobre o sistema *Symbian* pode ser encontrado em (MORRIS, 2007).

A seguir destacamos algumas vantagens e desvantagens do *Symbian* apontadas por Costa e Filho (2013).

- Vantagens:
  - Rapidez e acessibilidade;
  - Sistema aberto e de baixo custo;
  - Portabilidade para dispositivos com processadores de 1,3Ghz;
  - Suporte às arquiteturas ARM e x86;

[1] http://www.nokia.com/
[2] https://www.microsoft.com
[3] https://www.accenture.com

  - Java Runtime 2.2 , Qt Mobility 1.1 e Qt4.7.
- Desvantagens:
  - Desenvolvimento principal descontinuado;
  - Sistema proprietário;
  - Portável somente para extensões .sis, .sisx e .jar.
  - Exclusivo para o *hardware Nokia*.

2.2 iOS

O *iPhone Operating System* é um sistema que roda nos dispositivos móveis da *Apple Inc*. Foi lançado inicialmente para o *iPhone* em outubro de 2012, e é um derivado do Mac OS X. Atualmente é usado em outros aparelhos da empresa, como *iPod Touch*, *iPad* e *Apple TV*. O iOS não pode ser executado em *hardware* de terceiros, pois ambos são de uso proprietário e exclusivo da *Apple*.

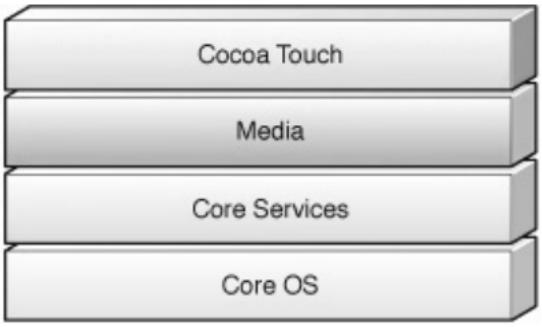
O iOS foi o primeiro sistema para *smartphones* a permitir partidas *multiplayers* por meio de uma central de jogos presente no aparelho. Ele também introduziu uma assistente pessoal, Siri. Este recurso, ativado por voz, permite ao usuário controlar várias funcionalidades do dispositivo, fazer perguntas e receber as respostas em áudio. O iOS permite também o acesso a arquivos, documentos, fotos e outros dados do usuário armazenados na Nuvem, por meio do serviço de *iCloud* (COUNTINHO, 2014).

A arquitetura do iOS é semelhante a do Mac OS (ROCHA; NETO, 2014). Ela é composta por quatro camadas, responsáveis por fornecer *frameworks* distintos para o desenvolvimento de aplicações. Essas camadas permitem que as aplicações se comuniquem com o sistema através de interfaces, protegendo-as assim de eventuais mudanças de *hardware*. Cada camada tem o papel de encapsular funcionalidades e prover abstrações para os desenvolvedores.

A *Core OS* é a camada de mais baixo nível do sistema. Nela, estão presentes aspectos do sistema operacional, tais como gerenciamento de memória, sistemas de arquivos e comunicação de processos. O iOS fornece acesso a essas informações através de bibliotecas escritas em C/C++. Na próxima camada, *Core Services*, estão presentes recursos fundamentais do sistema, comum a quase todos os aplicativos. Nela estão disponíveis tecnologias como *SQLite*, que possibilita incorporar um banco de dados ao seu aplicativo, sem recorrer a um servidor de dados remoto. A camada seguinte, conhecida como *Media*, contém as tecnologias de gráfico, áudio e vídeo. Ela fornece ferramentas para a criação de aplicativos que necessitam de

mais recursos de multimídia. E também é utilizada em aplicações mais robustas e complexas. Por fim, a *Cocoa Touch*, camada mais externa do iOS, define a infraestrutura para tecnologias fundamentais como serviços de notificações, proteção de dados, etc. Mais detalhes sobre a arquitetura do iOS podem ser encontrados em (COSTA; FILHO, 2013; ROCHA; NETO, 2014; SOUSA, 2014).

Figura 2 – Camadas do iOS.



Fonte – Adaptado de Rocha e Neto (2014, p. 2)

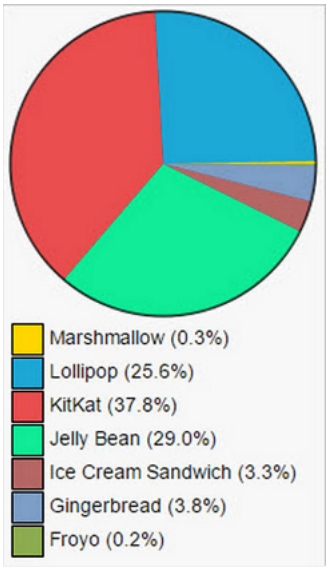
Segundo Costa e Filho (2013), a grande vantagem do iOS é que existe um alto desempenho dos produtos que usam esse sistema operacional móvel, pois tanto *hardware* como *software*, são projetados um para o outro na mesma plataforma. No entanto, algumas desvantagens são observadas quando o sistema tem que interagir com outra aplicação ou dispositivo externo ao ambiente.

2.3 Android

O *Android* é um sistema baseado no *kernel* do *Linux*. Foi oficialmente lançado em 2008 no extinto *Android Market*. A primeira versão disponibilizada, *Astro* 1.0, permitia acesso à Internet, integração a aplicações do google, reprodução de mídias, etc. Atualmente está na versão 6.0, que recebeu o nome de *Marshmallow*. Essa versão conta com vários recursos como otimização do uso da bateria quando o dispositivo está em *stand-by*, suporte nativo para leitores de impressão digital, *backup* e restauração automática no *drive* para dados e aplicativos, etc (MEYER, 2015). Esse SO caracteriza-se por ser uma plataforma livre e de código aberto. Apesar de ser escrito em C e C++, a maioria das aplicações para *Android* são desenvolvidas utilizando *Java*. No entanto, é possível utilizar C ou C++ no desenvolvimento de aplicações com auxilio da tecnologia *Java Native Interface* (JNI) e da ferramenta *Android* NDK (*Native Development Kit*) (SOUSA, 2014).

Android é o sistema mais utilizado do mundo. De 2009 a 2014, mais de 3 milhões de dispositivos com suporte ao *Android* foram comercializados no mundo, um crescimento de quase 78% no número de aparelhos *Android* vendidos mundialmente. A popularidade do *Android* fica ainda mais evidente quando comparamos por exemplo com o *Windows Phone*, que detém 2,7% do mercado mundial de dispositivos móveis (MEYER, 2015). A Figura 3 a seguir apresenta a evolução dos uso do *Android* desde a versão *Froyo* (2009) até a *Marshmallow* (2015).

Figura 3 – Evolução do uso dos *Android's*.



Fonte – Adaptado de Meyer (2015).

A seguir apresentamos algumas vantagens e desvantagens do *Android* destacadas por Costa e Filho (2013).

- Vantagens:
  - Máquina Virtual *Dalvik*: É o atual *Android Runtime* (ART). Permite que programas escritos em *Java* possam ser compilados e executados em qualquer dispositivo *Android*, independentemente do processador utilizado;
  - Ambiente de desenvolvimento (SDK): contém um emulador, ferramentas de *debbuging*, memória e análise de desempenho.
- Desvantagens:
  - Aplicativos pagos mais caros;
  - Carência e problemas de atualização do sistema operacional para o aparelho;
  - Incompatibilidade de aplicativos e suas versões entre aparelhos diferentes.

2.4 Windows Phone

O *Windows Phone* é o sistema operacional móvel pertencente a *Microsoft*. Lançado em 2010, substituindo a antiga plataforma *Windows Mobile*, o novo SO, apesar de utilizar o mesmo *Kernel* da versão antiga, não é compatível com o anterior. O *Windows Phone* difere bastante dos seus concorrentes: iOS e *Android*. Umas das principais diferenças está na interface, mais intuitiva e de fácil navegação. Inicialmente suportava aplicações desenvolvidas em *Visual Basic* e C# (UEDA; SILVA, 2013).

Assim como fizemos para os sistemas anteriores, apresentamos a seguir algumas vantagens e desvantagens do *Windows Phone* apontadas por Costa e Filho (2013).

- Vantagens:
  - *Accelerometer*: o acelerômetro é responsável por detectar a aceleração do telefone, o que representa uma mudança de velocidade física. Pode ser utilizado para detectar a orientação do telefone, ou movimentos realizados com ele;
  - *Location*: utilizado para detectar a localização geográfica do telefone, GPS embutido;
  - Tecnologia NFC (*Near Field Communication*): permite transferir arquivos de maneira rápida e interações de mídias (*online*) com *skype*.
- Desvantagens:
  - Multitarefa: só poderá abrir um aplicativo por vez com exceções. Se levarmos em consideração que outros realizam esta função;
  - Sem suporte a *Adobe Flash*;
  - Sem e-mail unificado: ao contrário do fluxo, o aplicativo de e-mail do WP8 não unifica as suas contas e exige mais “cliques” do que seria necessário;

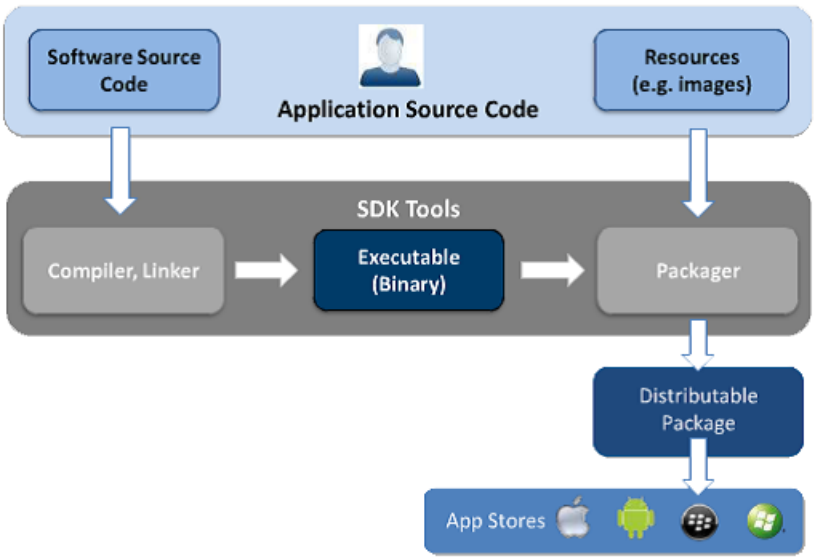
3 ABORDAGEM DE DESENVOLVIMENTO CROSS-PLATFORM

3.1 Aplicações nativas

Segundo Toledo e Deus (2012), “uma aplicação nativa/embarcada é um software desenvolvido para executar em uma plataforma específica. Os arquivos resultantes da compilação do aplicativo devem ser instalados diretamente no sistema operacional, tais como apresentação, processamento e armazenamento de dados”. Um aplicativo nativo é uma aplicação desenvolvida com uma linguagem de programação dedicada a uma plataforma específica e usufruem de todos os recursos do dispositivo móvel (CHARKAOUI *et al.*, 2014). Apesar da necessidade de plataforma e conhecimentos específicos, esta estratégia proporciona maior qualidade e experiência de usuário do que outros métodos de desenvolvimento de aplicativos móveis (REDDA, 2012).

Aplicações nativas permitem a manipulação de dados *off-line*, ou seja, armazenados em um banco de dados no próprio aparelho, permitindo ao software nativo funcionar mesmo em locais sem acesso à internet, bem como acesso ao hardware presente no dispositivo, como o telefone, câmera, microfone, bluetooth e acelerômetro (TOLEDO; DEUS, 2012). O desenvolvimento de aplicações nativas permite ao desenvolvedor um arcabouço de possiblidades de utilização dos recursos disponíveis no hardware, contribuindo para um maior aproveitamento dos recursos e maior desempenho da aplicação.

Figura 4 – Infraestrutura nativa de desenvolvimento

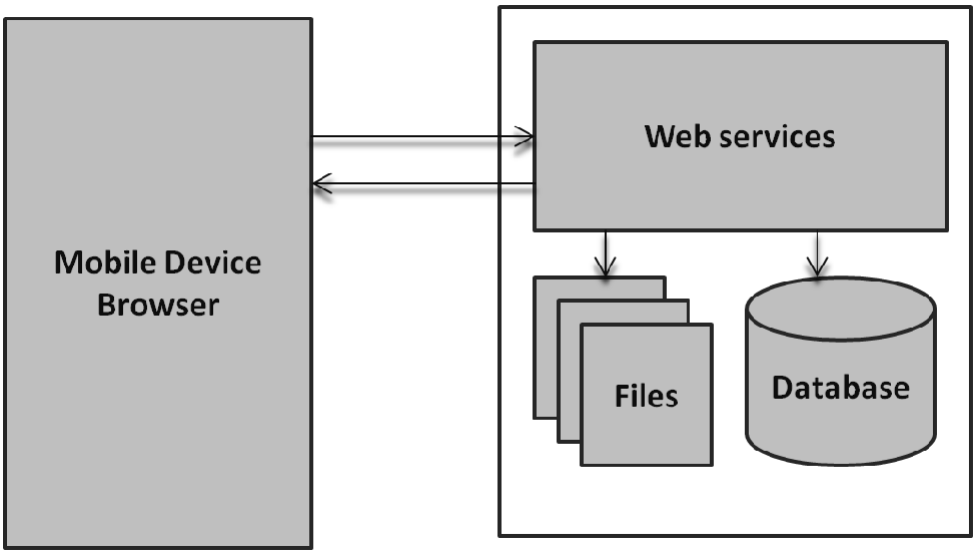


Fonte – (REDDA, 2012).

3.2 Web Apps

Um web app móvel é uma aplicação web que é projetada para executar no navegador do dispositivo móvel (CHARKAOUI *et al.*, 2014). Aplicações web móveis, geralmente, são desenvolvidas utilizando HTML, CSS e JavaScript. Nesta abordagem, o dispositivo móvel não terá quaisquer componentes específicos do aplicativo instalado. As aplicações serão baseadas em navegador e os dados do aplicativo são impulsionados por servidor. Uma vez que o aplicativo é baseado em navegador, o aplicativo será independente de plataforma (RAJ; TOLETY, 2012). Nesta abordagem, algumas operações são realizadas no lado do cliente, enquanto que as

Figura 5 – Infraestrutura de Web Apps.



Fonte – (RAJ; TOLETY, 2012)

operações críticas são realizadas pelo servidor. O cliente recebe a lógica de validação de dados de interface de usuário e usuário do aplicativo, enquanto o servidor implementa as lógicas de negócio (RAJ; TOLETY, 2012). Ao utilizar essa abordagem, os desenvolvedores implementam a aplicação como um site otimizado para dispositivos móveis, e se comportam, parcialmente, como páginas web comuns (HEITKÖTTER *et al.*, 2012). Raj e Tolety (2012) apresentam as vantagens e desafios da abordagem de *Web Apps*, são elas:

- Vantagens:
  - Um *Web App* não requer qualquer instalação no dispositivo móvel. O aplicativo pode ser acessado usando uma URL por meio do navegador do dispositivo.
  - Como os dados e aplicação são armazenados no servidor, qualquer atualização da aplicação é livre de manutenção do dispositivo. Inexiste a necessidade de atualização

do dispositivo para utilização do aplicativo.
  - As interfaces do usuário Web podem ser reutilizadas em diferentes plataformas, visto que os navegadores são bastante padronizados.
- Desafios:
  - *Web Apps* não são distribuídos por lojas de aplicativos. Dessa forma, a ausência em lojas de aplicativos, pode impactar negativamente na popularidade do aplicativo.
  - Podem perder desempenho devido instabilidade de conexão e rede.
  - Não podem acessar o hardware e o software do dispositivo, como: GPS, câmera etc.
  - *Web Apps* devem suportar diferentes resoluções de tela. Este é um fator fundamental que deve ser considerado durante o desenvolvimento de aplicações móveis.
  - O desenvolvedor tem menos controle sobre a forma como diferentes navegadores processarão o conteúdo do aplicativo.
  - *Web Apps* possuem limitações na utilização de gestos oferecidos em plataformas.
  - Grande demanda de testes do aplicativo em diferentes resoluções de tela.
  - A monetização de um Web App não é trivial como de um aplicativo nativo, visto que não está disponível em uma loja de aplicativos.

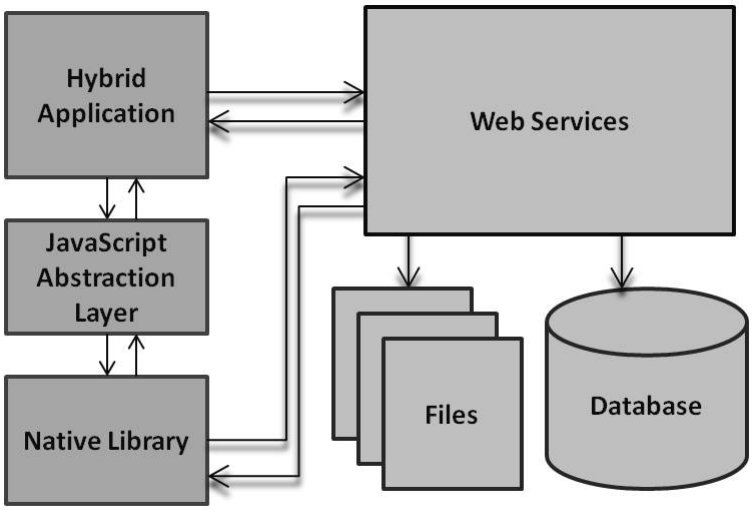
Para utilizar os recursos dos dispositivos, uma abordagem híbrida foi proposta e será abordada a seguir.

3.3 Aplicações híbridas

Uma abordagem híbrida está entre a abordagem web e a nativa. A aplicação híbrida é desenvolvida utilizando tecnologias web e é executada dentro do container nativo do dispositivo móvel (CHARKAOUI *et al.*, 2014). De acordo com Lopes (2014), as aplicações híbridas são desenvolvidas utilizando HTML, CSS e JavaScript e empacotadas em *packages apps* e publicadas em *apps stores* como se fossem nativas, ou seja, em linhas gerais é um meio termo das abordagens nativas e web. Uma abordagem híbrida utiliza o mecanismo do navegador do dispositivo que exibe o conteúdo HTML em tela cheia. A camada de abstração expõe as capacidades do dispositivo como APIs JavaScript. Ao contrário das aplicações Web, as aplicações híbridas devem ser baixadas e instaladas no dispositivo (RAJ; TOLETY, 2012). Raj e Tolety (2012) apresentam as vantagens e desafios da abordagem híbrida, são elas:

- Vantagens:
  - Aplicações híbridas são distribuídas pelas lojas de aplicativos

Figura 6 – Estrutura de aplicação híbrida



Fonte – (RAJ; TOLETY, 2012)

  - A principal vantagem da abordagem híbrida é que a interface de usuário pode ser reutilizada em diferentes plataformas utilizando recursos da plataforma nativa.
  - Através da camada de abstração de hardware, a aplicação pode fazer uso de recursos do dispositivo.
  - Em aplicações híbridas, a aplicação web é alimentada por capacidades de computação do dispositivo.
- Desafios:
  - Aplicações híbridas tem desempenho inferior em comparação com aplicativos nativos, devido a execução no mecanismo do navegador.
  - Uma vez que uma aplicação híbrida utiliza JavaScript na camada de abstração, é submetida ao espaço transversal de vulnerabilidades de computação. Aplicações híbridas também sofrem de comportamento específico e de incompatibilidades do modelo de segmentação da plataforma JavaScript.
  - Mesmo com a reutilização da interface de usuário em diferentes plataformas, não terá a sensação de aparência de aplicativos nativos. Para atingir o nativo, o estilo específico da plataforma deve ser levado em consideração durante o desenvolvimento.

4 *FRAMEWORKS* DE DESENVOLVIMENTO *CROSS-PLATFORM*

Nesta seção serão apresentados os *frameworks* estudados e utilizados no trabalho.

4.1 Qt

Qt é um *framework* de desenvolvimento de software completo, com ferramentas projetadas para simplificar a criação de aplicações com interfaces de usuário de alta interatividade para *desktops*, sistemas embarcados e sistemas operacionais móveis (COMPANY, 2016) utilizando C++ como principal linguagem de programação.

O *framework* possibilita a escrita de aplicações multiplataforma, dando suporte desde à sistemas operacionais mais *mainstreans* como Windows, Linux, Mac OS X, Android e IOS, à outras plataformas menos convencionais como Windows CE [1], Windows Runtime (WinRT) [2], QNX [3] e Linux embarcados.

4.1.1 Histórico

O trabalho de havia se iniciado em 1991, quando Haavard Nord começou a escrever o código que viria fazer parte do *framework* e, em 1993, Eirik Chambe-Eng se junta ao amigo para desenvolverem os primeiros componentes gráficos do projeto (BLANCHETTE, 2008). Juntos, em 1995 fundaram a Quasar Technologies que posteriormente passa a se chamar Trolltech. A versão 0.9 do *framework* foi disponibilizada em maio de 1995, com suporte à ambientes X11/Linux. No mesmo ano, a empresa foi contratada pela Metis, uma empresa norueguesa, para desenvolver um software em Qt, surgindo a partir daí seu primeiro cliente. No ano seguinte, 1996, a Agência Espacial Europeia tornou-se seu segundo cliente, adquirindo dez licenças comerciais. Ainda no mesmo ano, as versões 1.0 e 1.1 foram lançadas (BLANCHETTE, 2008).

Em junho do ano de 1999, o Qt 2.0 foi lançado juntamente com uma nova licença de código aberto, a Q Public License (QPL). A versão 3.0 foi lançada em 200, dando suporte para Windows, Mac OS X, Unix e Linux. Em relação ao Qt 2.0, a versão 3.0 trouxe 42 novas classes. Em 2005, o Qt 4.0 foi lançado e apresentado uma mudança mais significativa, contando com de 500 classes e mais de 9000 funções. Nesta última versão, houve um grande avanço em relação às versões anteriores, com melhorias que incluem entre uma avançada e eficiente arquitetura

[1] https://msdn.microsoft.com/en-us/library/ee504813(v=winembedded.70).aspx
[2] https://msdn.microsoft.com/en-us/windows/uwp/winrt-components/index
[3] http://www.qnx.com/

model/view (modelo que separa a lógica de programação da interface de usuário) e uma rápida e flexível estrutura de desenho bidimensional. Qt 4 foi a primeira edição a disponibilizar tanto a licença *open source* quanto comercial em todas as plataformas suportadas (BLANCHETTE, 2008).

No ano de 2008, a Nokia compra a Trolltech, passando a ter os direitos de licenciamento do Qt. A partir daí, o desenvolvimento de aplicações em Qt foi voltado para os dispositivos Nokia, que tinha como plataforma móvel o Symbian. Contudo, logo em 2011, a Nokia anunciou sua mudança de estratégia para *smartphones*, trocando o projeto Symbian por tecnologias Microsoft e, consequentemente, abrindo mão do projeto Qt juntamente com licenciamento comercial e serviços de suporte para a empresa finlandesa Digia (COMPANY, 2016).

O framework atualmente está em sua quinta versão (Qt 5), lançada em dezembro de 2012, tendo como principais recursos: bibliotecas OpenGL e OpenGL ES (OpenGL for Embedded Systems). Essas bibliotecas dão suporte a criação de aplicações que utilizam gráficos em 3D e, no caso do OpenGL ES, os recursos gráficos são oferecidos para sistemas embarcados utilizados em dispositivos como *smartphones*, vídeo games, sistemas automotivos interativos, etc. Esta versão passou a oferecer suporte oficial ao desenvolvimento de aplicações para a sistemas Android, iOS, Blackberry (COMPANY, 2016)

4.1.2 Módulos

Desde a versão 4, o Qt é organizado em módulos, que podem ser adicionados ou removidos do projeto, de acordo com a necessidade da aplicação. Esses módulos dão suporte à mais diversas necessidades do desenvolvedor, fazendo que não se faça necessário o uso de uma outras bibliotecas para *features* comuns ao desenvolvimento de sistemas como acesso à banco de dados, comunicação pela rede, *threads*, suporte à impressão de documentos, testes, dentre outras. Os módulos são divididos em dois grupos: *Qt Essentials*, que é a base do *framework*, e *Qt Add-Ons* que são módulos que agregam valor para propósitos específicos, podendo estar com uso limitado pela plataforma alvo ou pela licença de uso. A Tabela 1 enunera os principais módulos do Qt.

Tabela 1 – Qt Essentials

Módulo	Descrição
Qt Core	Classe comuns utilizadas por outro módulos
Qt GUI	Classes base para componentes de interface gráfica de usuário (GUI)
Qt Multimedia	Classes para funcionalidades de audio, video, radio e camera.
Qt Multimedia Widgets	Componentes gráficos para implementação de funcionalidades multimídia.
Qt Network	Classes para fazer programação de comunicação de redes de maneira mais fácil e portável.
Qt QML	Classes para QML e JavaScript.
Qt Quick	Um *framework* para construção de aplicações como interfaces altamente customizáveis.
Qt Quick Controls	Controles baseados em Qt Quick para criação de interfaces de usúario em estilos clássicos.
Qt Quick Dialogs	Tipos para criar e interagir com o sistema de diálogos em uma aplicação Qt Quick.
Qt Quick Layouts	*Layouts* são itens que são usados para posicionar outros itens do Qt Quick 2 em aplicações de interfaces gráficas.
Qt SQL	Classes para integração com banco de dados relacionais.
Qt Test	Classes para teses unitários.
Qt Widgets	Classes quem extendem Qt GUI com elementos gráfico em C++ e XML.

Fonte: Adaptado de Company (2016)

4.2 Ionic *Framework*

O Ionic é um *framework* para o desevolvimento de aplicações mobile híbridas através de um *SDK* - Software Development Kit HTML5 que permite que as aplicações sejam construídas através da utilização de tecnologias utilizadas no desenvolvimento para a *WEB*, HTML, CSS e Javascript (IONIC, 2015).

O Ionic é construído em cima do AngularJS [4] e utiliza o Apache Cordova [5] para construção da aplicação de um *Webview* que de maneira simplificada seria a execução em um navegador que não possui barra de endereço e que é capaz de renderizar as tecnologias previamente citadas (KHANNA; HARTINGTON, 2016). Devido a esta integração com o AngularJS o Ionic herdou várias diretivas, que é uma das *features* do ecossistema do AngularJS que permite extender o HTML de modo a fornecer reuso, componentização e lógica para a marcação.

O Ionic tem seu grande foco o *Look and Feel*, ou seja na interação do usuário com a interface da aplicação, através da construção de interfaces poderosas e de simples desenvolvimento. Embora exista uma dependência do AngularJS para desenvolvimento para consumir todo o recurso que o Ionic pode oferecer, pode-se utilizar apenas o CSS mas perdendo assim componentes de interação, animação dentre outros já fornecidos na plataforma (IONIC, 2015).

O SDK do Ionic é de código aberto hospedado no Github [6] o que permite que toda a comunidade possa ter acesso ao código e trabalhar em melhorias que beneficiem a todos, está sob a licença MIT o que permite o desenvolvimento de aplicações pessoais e comerciais sem

[4] https://angularjs.org/
[5] https://cordova.apache.org/
[6] https://github.com/driftyco/ionic

custo.

4.2.1 Ferramentas

O Ionic possui um conjunto de ferramentas pertecentes ao *framework*, tais ferramentas auxiliam o trabalho do desenvolvedor na execução de suas atividades, tais como: *Ionic Lab*, que trata-se de uma aplicação desktop que permite a realização de criação, visualização prévia emulação, teste das aplicações em suas respectivas plataformas. *Live Reload*, que fica observando qualquer mudança no fonte da aplicação e automaticamente recarrega a aplicação quando necessário, permitindo assim que seja menos necessário a reconstrução da aplicação a cada pequena alteração. Como dependência para uso da ferramenta a máquina de desenvolvimento e o dispositivo devem estar na mesma rede local e o dispositivo possuir suporte a *web sockets*. CLI - *Command Line Interface* que facilita inicialização, o *build* e emulação das aplicações. Deve ser utilizado de uma *shell* para usuários Mac do terminal e usuários Windows do Prompt de Comando (KHANNA; HARTINGTON, 2016) (IONIC, 2015).

5 MATERIAIS E MÉTODOS

Este trabalho tem o caráter de uma Pesquisa Comparativa, que segundo Heidenheimer (1983), é o ato de comparar duas ou mais coisas, tento como objetivo descobrir algo sobre uma ou todas as coisas as quais estão sendo comparadas. Os objetos desta comparação foram dois *frameworks* mais usado pelas empresas do mercado de T.I de Fortaleza e Aquiraz. O *frameworks* foram elencados através de questionário aplicados à essas empresas.

5.1 Elaboração do questionário

Nesta etapa foi elaborado um questionário simples a ser respondido por diretores, gerentes ou profissionais responsáveis por projetos nas empresas de TI em Fortaleza e Aquiraz, com o objetivo de realizar um levantamento dos *frameworks* de desenvolvimento para aplicações móveis multiplataforma mais utilizados no mercado de cearense de TI. O formulário da pesquisa feita foi disponibilizado *on-line*, através da ferramenta *Google Forms* [1].

O objetvo do formulário era quantificar o número de projetos que estão sendo desenvolvidos utilizando cada *framework* de desenvolvimento multiplataforma. Foram levados em consideração apenas projetos que já estavam em produção, fase de manutenção ou que ainda estavam em desenvolvimento, visto que estes são os únicos que agregavam algum valor à empresa e ao mercado, deixando de lado projetos descontinuados.

A escolha dos *frameworks* listados no formulário da pesquisa foi feita de forma empírica, tendo como base pesquisas em fóruns e sites da internet, mas ainda deixando o máximo de opções à quem iria responder. Ainda contendo um espaço para apontar outro *framework* que não estivesse na lista.

5.2 Aplicação da pesquisa

Com o questionário desenvolvido, o mesmo foi encaminhado as empresas selecionadas, que são relevantes no mercado de TI no Ceará. Ao final desta etapa, obteve-se o número de projetos em cada *framework* de desenvolvimento multiplataforma. A pesquisa foi realizada entre os dias 13 de Abril de 2016 e 16 de Maio do mesmo ano, e contou com a participação de 11 empresas de TI. Sendo 10 situadas em Fortaleza-CE e uma em Aquiraz-CE devido ao prévio conhecimento e vínculo empregatício que o autor já possuiu com a mesma. Todas as respostas

[1] https://apps.google.com/intx/pt-BR/products/forms/

foram fornecidas por diretores, gerentes ou analistas líderes de projetos.

5.3 Desenvolvimento da Prova de Conceito

A prova de conceito trata-se de um aplicativo de Notas, onde o usuário pode realizar o login na aplicação, salvar notas de texto localmente e sincronizar com um serviço RESTful [2]. Essas características na aplicação foram importantes para poder avaliar a disponibilidade e facilidades em cada *framework*:

- Acesso e troca de dados através da internet;
- Acesso à um *storage* local, com um banco SQLite;

5.3.1 Aplicação Server-side

Como já citado neste trabalho, faz-se uso do padrão de arquitetura de aplicações REST, o desenvolvimento da aplicação de modo a utilizar este padrão é destacado a seguir: A *Application Programming Interface* (API) foi desenvolvida em NodeJS [3]que trata-se de um interpretador Javascript no lado do servidor, ExpressJS [4] para mapeamentos das rotas que são consumidas pelas aplicações e MongoDB [5] como banco de dados. Todo o código fonte da aplicação está aberto e encontra-se hospedado no Github [6] em link especificado no rodapé.

A aplicação foi hospedada no Heroku [7] e o banco de dados MongoDB no mLab [8].

5.3.2 Aplicação Client-side

A aplicação cliente foi desenvolvida usando duas tecnologias Cross-platform e com diferentes abordagens: Qt e Ionic.

A aplicação Qt foi desenvolvida com as seguintes tecnologias:

Versão do Qt 5.7, com linguagem visual QML usando banco de dados local SQLite.

[2] http://www.jopera.org/files/www2008-restws-pautasso-zimmermann-leymann.pdf
[3] https://nodejs.org/en/
[4] http://expressjs.com/pt-br/
[5] https://www.mongodb.com/
[6] https://github.com/danielwolney/api-notas-tcc
[7] https://keepnotes-api.herokuapp.com/v1/api
[8] https://mlab.com/

5.4 Métricas para comparação

Para avaliar os *frameworks*, foram definidos alguns critérios como: Tamanho do arquivo da aplicação, tamanho da aplicação instalada, consumo de memória, tempo de carregamento, linhas de código e número de arquivos gerados, que foram aplicados nas provas de conceitos desenvolvidas em cada ferramenta.

- Tamanho do arquivo da aplicação: Pretendia-se verificar o tamanho do arquivo gerado, o executável final. Esta medida se faz importante pois o número impacta no diretamente no consumo de dados e tempo de download da aplicação;
- Tamanho da aplicação instalada: Verificar o tamanho da aplicação depois de estar devidamente instalada no dispositivo. Sua importância se dá pelo fato de consumir mais ou menos memória do dispositivo.
- Consumo de memória: Verificar através da ferramenta *Android Debug Bridge* (ADB) os picos de consumo de memória durante o ciclo de vida da aplicação. Esta verificação torna-se importante para usuários que utilizam muitas aplicações simultaneamente;
- Tempo de carregamento: Tempo levado desde clique no ícone da aplicação até renderização da tela inicial. Um número elevado influencia negativamente na experiência do usuário;
- Linhas de código: Verificar o número de linhas de código escrito, não sendo levados em conta códigos de arquivos gerados como *Meta Object Code* (MOC), arquivos de *resources* ou outros tipos de arquivos. Esta verificação não é relevante para o usuário final, mas para o programador por influenciar na legibilidade e manutenção do projeto;
- Número de arquivos gerados: Verificar o número de arquivos finais do projeto. Assim como o número de linhas de código, esta verificação também é irrelevante para o usuário final, importando apenas para o programador por também influenciar na legibilidade e manutenção do projeto;

5.5 Avaliação de acordo com as métricas

Concluída a implementação das provas de conceito, iniciou-se a fase onde foi realizada a comparação e avaliação da duas aplicações de acordo com as métricas definidas no tópico 5.4. Foram utilizadas ferramentas disponibilizadas pelo Android SDK para obtenção dos dados.

5.6 Análise dos dados

Este passo tinha como objetivo consolidar as informações obtidas ao final do passo 5.5, que constitui o objetivo geral proposto neste trabalho. Foram geradas tabelas e gráficos contendo os valores obtidos em cada aplicação de acordo com cada métrica definida.

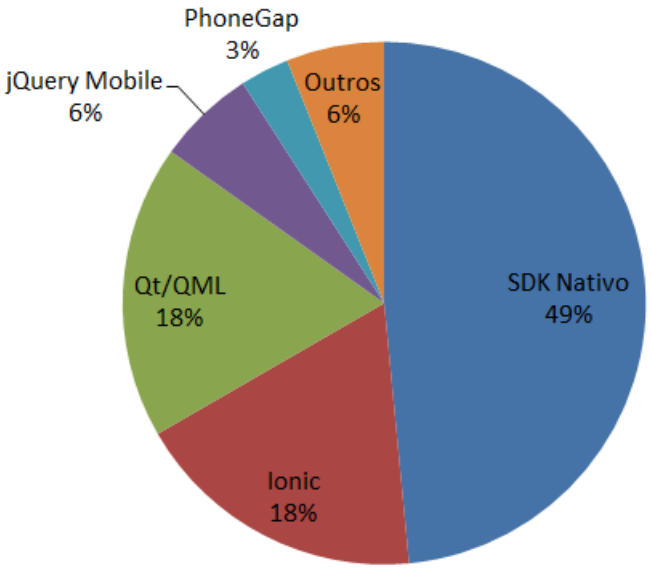
6 RESULTADOS

Nesta seção serão apresentados os resultados deste trabalho de acordo com as definições dispostas na seção 5.

6.0.1 Resultado da pesquisa com as empresas

Os dados levantados pelas pesquisa apontam que 51% dos projetos das empresas consultadas são desenvolvidos utilizando alguma tecnologia *cross-platform*, restando o outros 49% para o SDK Nativo de cada plataforma. A pesquisa mostra que os *frameworks* para desenvolvimento multiplataforma mais utilizados são o Qt/QML [1] e Ionic[2], ambos com uma fatia de 18%, como mostra a Figura 7.

Figura 7 – *Frameworks* utilizados no mercado de TI no Ceará.



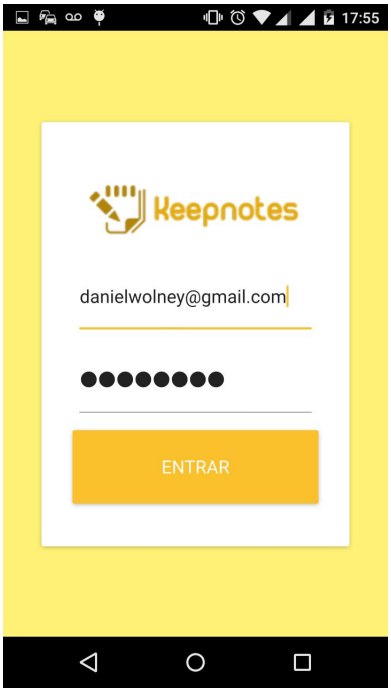
Fonte – Elaborado pelo autor

Como descrito no passo 5.3, foi implementada uma aplicação utilizando-se da duas tecnologias anteriormente citadas, *Qt* e *Ionic*. A aplicação *server-side* não terá ênfase nas discussões devido ao seu impacto na aplicação haja vista tratar-se da mesma aplicação para ambas as tecnologias. A seguir será melhor detalhada cada aplicação em sua respectiva tecnologia.

[1] http://www.qt.io/
[2] http://ionicframework.com/

6.0.2 Aplicação Qt

Figura 8 – Tela de Login da Aplicação Qt.



Fonte – Elaborado pelo autor

A tela de login permite que o usuário se autentique na aplicação consumindo a API informando suas credenciais: Email e senha, a fim de obter acesso às suas notas.

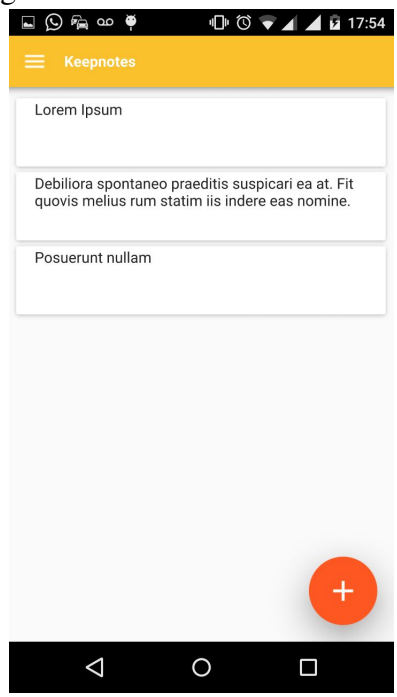
Ao executar a ação de entrar a aplicação recupera os dados informados nos campos e envia para a API, esta faz uma verificação das credencias e se tudo for validado o acesso é garantido ao usuário.

A tela de notas permite que o usuário visualize as notas que ele já criou através da exibição do conteúdo das notas em forma de uma listagem bem definida. O cadastro de uma nova nota é possível através do clique no botão vermelho com sinal de + (mais).

A tela de cadastro de nova é um campo de texto livre onde o usuário irá informar o conteúdo da nota sem restrição de caracteres.

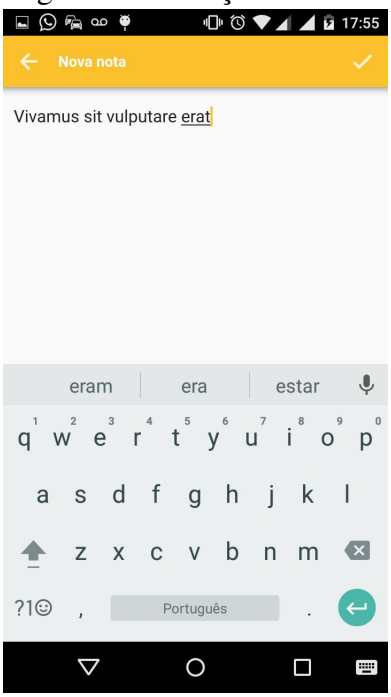
O menu da aplicação permite que o usuário volte a visualizar suas notas existentes bem como realizar a ação de sair, quando ele deseja encerrar sua autenticação da aplicação e assim ser redirecionado a tela de login.

Figura 9 – Tela de Notas existentes.



Fonte – Elaborado pelo autor

Figura 10 – Criação de Nota.



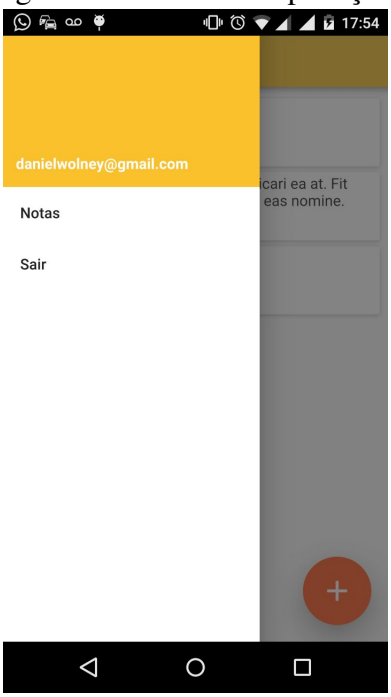
Fonte – Elaborado pelo autor

6.0.3 Aplicação Ionic

Afim de garantir a melhor visualização do comparativo, as mesmas telas da aplicação Ionic também serão detalhadas.

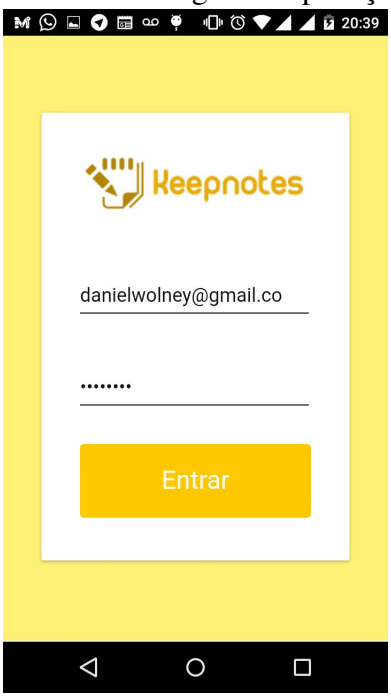
A forma de autenticação na aplicação Ionic também ocorre da mesma forma, uma

Figura 11 – Menu da Aplicação.



Fonte – Elaborado pelo autor

Figura 12 – Tela de Login da Aplicação Ionic.

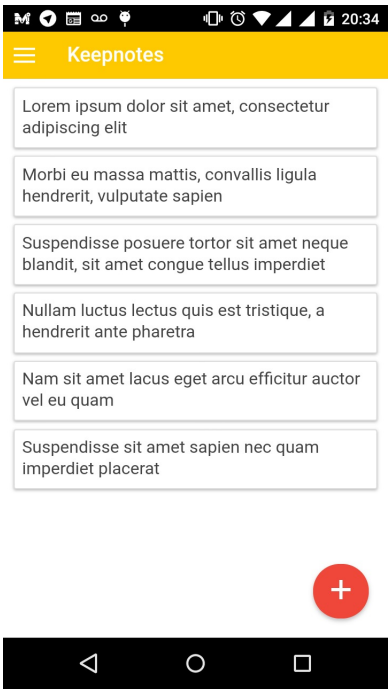


Fonte – Elaborado pelo autor

das vantagens da utilização da API é a uniformidade na forma de autenticação, onde a responsabilidade de garantir o acesso do usuário é da API e não das aplicações cliente.

De maneira idêntica a aplicação *Qt*, após a autenticação, o usuário é direcionado a tela de notas existentes, onde a estrutura fora mantida e para a criação de uma nova nota, o

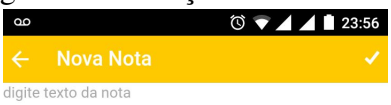
Figura 13 – Tela de Notas existentes Ionic.



Fonte – Elaborado pelo autor

usuário deve clicar no botão vermelho com o símbolo + (mais) dentro.

Figura 14 – Criação de Nota Ionic.

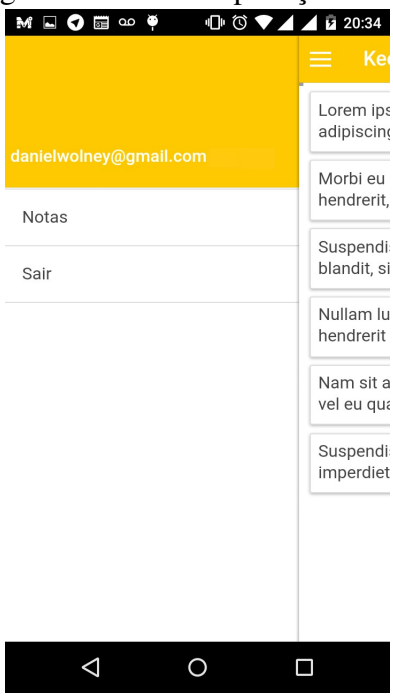


Fonte – Elaborado pelo autor

Similarmente, a tela de cadastro de nova é um campo de texto livre onde o usuário irá informar o conteúdo da nota sem restrição de caracteres.

Seguindo o mesmo padrão da aplicação *Qt* o menu da aplicação *Ionic* também é

Figura 15 – Menu Aplicação Ionic.



Fonte – Elaborado pelo autor

baseado em um menu *off-canvas*.

6.0.4 Comparativo das métricas

Conforme explanado na seção 5.5, um comparativo com as métricas já definido foi realizado a fim de obter-se um comparativo coerente entre as duas tecnologias, os dados do comparativo estão demonstrados abaixo:

6.0.4.1 Tamanho da Aplicação

Essa medida tem como objetivo avaliar o tamanho que a aplicação ocupa no dispositivo do usuário em termos de armazenamento.

O tamanho do APK (*Applicattion Package*) gerado: A aplicação **Qt:** gerou um APK de 10,5 MB (Megabytes) e a aplicação **Ionic:** gerou um APK de 7,15 MB(Megabytes).

Neste ponto observa-se que a aplicação Ionic gerou um apk menor, o que é um bom indicador, cerca de 28 % menor.

A medida seguinte colhida foi o quanto o APK devidamente instalado no dispositivo do usuários iria ocupar de armazenamento em definitivo, o que se colheu foi que a aplicação **Qt:** após a instalação ocupou do armazenamento do dispositivo 32,86 MB em contrapartida a aplicação **Ionic** ocupou 8,05 MB, uma diferença de 75 % além da observância de que a

aplicação instalada em um comparativo com o APK gerado teve seu tamanho triplicado para o *Qt* e aumentado em pouco mais de um décimo para a aplicação *Ionic*.

Em termos de arquivos criados, para a aplicação *Qt* vinte e três arquivos foram criados, para dez criados para a aplicação *Ionic*. Entretanto em termos de arquivos gerados, a aplicação *Qt* gerou um total de 33 arquivos cerca de 50 % a mais do que os originalmente criados, para 216 gerados pela aplicação *Ionic*, cerca de 2000 % a mais em comparativo com os arquivos criados.

A aplicação *Qt* demandou 1631 linhas de códigos escritos em comparativo com 774 pela aplicação *Ionic*.

A tabela 2 mostras os dados de maneira consolidada.

Tabela 2 – Tamanhos das aplicações

	Qt	**Ionic**
Tamanho do arquivo (.apk)	10,5 MB	7,15 MB
Aplicativo Instalado	32,86 MB	8,05 MB
Número de arquivos criados	23	10
Arquivos gerados	33	216
Linhas escritas	1631	774

Fonte – Elaborado pelo autor

6.0.4.2 Performance da Aplicação

Foram analisados dois aspectos das aplicação que dizem respeito ao desempenho das aplicações finais: tempo de inicialização da aplicação e consumo de memória RAM. Essas informação são importantes pois influenciam diretamente na experiencia do usuário. O dispositivo utilizado para execução dos testes foi um *smartphone* modelo Moto G de primeira geração com processador Quadcore de 1.2 GHz com 1GB de memória RAM e GPU Adreno 305.

Os números referentes ao tempo de inicialização da aplicação foram contabilizados com um cronômetro disparado manualmente a partir do clique no ícone do *app* e pausado quando a primeira tela fosse corretamente renderizada e passível de interação com o usuário. Os resultados obtidos neste passo são demonstrados detalhadamente na tabela 3.

Os dados nos informam que a primeira execução foi mais lenta comparadas as subsequentes em ambas as aplicações, possuindo tempos similares no decorrer das execuções. A aplicação Qt teve uma média de 3,71 segundos para inicializar, uma peformance mais significante nas execuções do que a aplicação Ionic que teve média de 4,848 segundos. Os dados podem ser

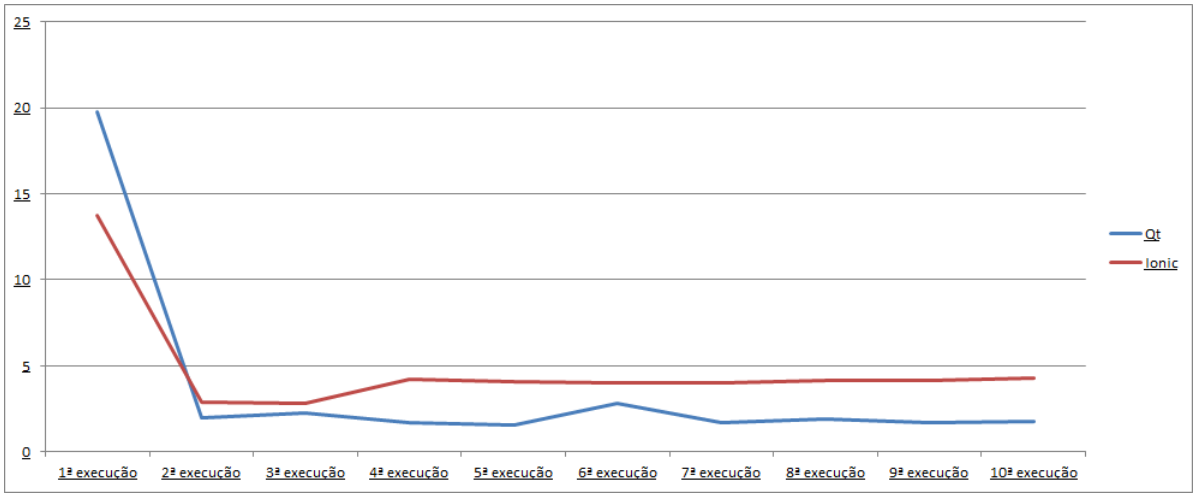
Tabela 3 – Tempo de inicialização das aplicações

	Qt	**Ionic**
1ª execução	19,75s	13,78s
2ª execução	1,95s	2,9s
3ª execução	2,25s	2,85s
4ª execução	1,67s	4,23s
5ª execução	1,55s	4,1s
6ª execução	2,81s	4s
7ª execução	1,71s	4,04s
8ª execução	1,9s	4,15s
9ª execução	1,72s	4,17s
10ª execução	1,79s	4,26s
Média	3,71s	4,848s

Fonte – Elaborado pelo autor

visualizados de maneira consolidados no gráfico da imagem 16

Figura 16 – Tempo de inicialização das aplicações



Fonte – Elaborado pelo autor

Para a coleta dos dados relativos ao consumo de memória RAM, foi utilizado a ferramenta disponível no Android Studio. Esses dados apontam que a aplicação Qt teve mais variância no consumo de memória, em momentos consumia pouca comparado a sua média em outro bem mais novamente em relação a média, já aplicação Ionic ficou em um intervalo de variância menor em termos de consumo, entretanto este dados possam variar em diferentes dispositivos, pois no caso do Ionic, irá depender da versão do WebView do sistema operacional. A tabela 4 nos mostra nos números obtidos nesta análise.

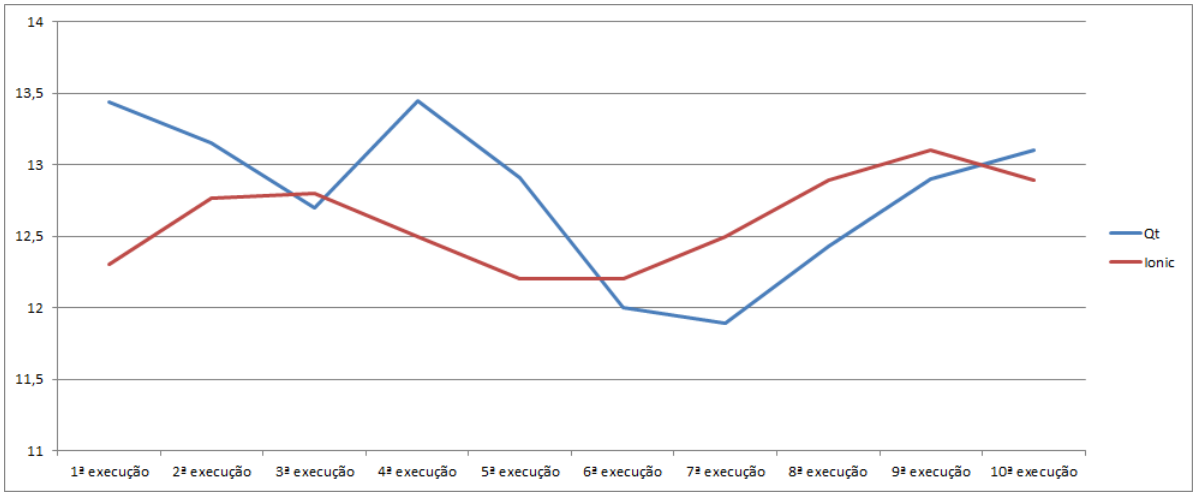
A imagem 17 nos mostra com mais clareza a variação do consumo de memória de cada uma das aplicações.

Neste aspecto, é importante observar que o consumo de memória RAM em aplicações

Tabela 4 – Consumo de memória RAM

	Qt	**Ionic**
1ª execução	13,44 MB	12,3 MB
2ª execução	13,15 MB	12,77 MB
3ª execução	12,7 MB	12,8 MB
4ª execução	13,45 MB	12,5 MB
5ª execução	12,91 MB	12,2 MB
6ª execução	12 MB	12,2 MB
7ª execução	11,89 MB	12,5 MB
8ª execução	12,43 MB	12,89 MB
9ª execução	12,9 MB	13,1 MB
10ª execução	13,1 MB	12,89 MB
Média	12,79 MB	12,615 MB

Figura 17 – Consumo de memória RAM



Fonte – Elaborado pelo autor

Qt, de maneira geral também é variável, visto que parte do código será escrita em C++ e alocação de desalocação de memória será de total responsabilidade do programador, já que a linguagem não dispõe de um *Garbage Collector*.

7 CONCLUSÕES E TRABALHOS FUTUROS

A partir dos resultados verificou-se que o Qt obteve melhor desempenho que o Ionic apenas no critério de tempo de inicialização da aplicação, embora tenha levado um tempo maior na primeira execução, a versão Qt conseguiu uma média menor de tempo nas execuções subsequentes. O framework Ionic leva vantagem em questões de esforço de desenvolvimento e tamanho final da aplicação: os dados mostram que a aplicação Qt tanto demanda mais linhas de códigos escritas, quanto o tamanho final da aplicação é bastante superior ao Ionic, chegando a ser 4 vezes maior. Essa diferença no tamanho final do aplicativo é explicada pelas *libs* compiladas do Qt que são embarcadas junto com a aplicação final, causando esse aumento significativo de tamanho. Enquanto que na aplicação Ionic, as *libs* são mantidas em texto puro e ainda minificadas, como permite a linguagem Javascript.

Esse trabalho tinha como principal objetivo analisar e comparar as diferenças e limitações entre as soluções de desenvolvimento de aplicações móveis multiplataforma com foco no mercado do Ceará, fornecendo critérios relevantes para escolha para quem pretende construir uma aplicação multiplataforma. Ao final conseguimos atingir esse objetivo, embora que, por limitações de ferramentas e recursos, não foi possível realizar a análise em outras plataformas.

Um trabalho futuro decorrente desta pesquisa poderia ser a comparação da mesma aplicação em outras plataformas, como iOS, por exemplo. A pesquisa realizada no mercado para determinar os *frameworks* ainda poderia ser elevada à âmbito nacional através de parcerias com sindicatos para que a pesquisa seja viabilizada.

REFERÊNCIAS

BLANCHETTE, M. S. J. **C++ GUI Programming with Qt 4**. 2. ed. Massachusetts: Prentice Hall, 2008.

CHARKAOUI, S.; ADRAOUI, Z.; BENLAHMAR, E. H. Cross-platform mobile development approaches. In: IEEE. **Information Science and Technology (CIST), 2014 Third IEEE International Colloquium in**. [S.l.], 2014. p. 188–191.

COMPANY, Q. 2016. Disponível em: <https://www.qt.io/>.

COSTA, N. P. O.; FILHO, N. F. D. Análise e avaliação funcional de sistemas operacionais móveis: vantagens e desvantagens. **Revista de Sistemas e Computação-RSC**, v. 3, n. 1, 2013.

COUNTINHO, D. **O iOS é o sistema da Apple para os seus dispositivos móveis**. 2014. Disponível em: <http://www.techtudo.com.br/tudo-sobre/ios.html/>.

DALMASSO, I.; DATTA, S. K.; BONNET, C.; NIKAEIN, N. Survey, comparison and evaluation of cross platform mobile application development tools. In: IEEE. **Wireless Communications and Mobile Computing Conference (IWCMC), 2013 9th International**. [S.l.], 2013. p. 323–328.

HEIDENHEIMER, C. T. A. H. H. A. J. **Comparative Public Policy**. [S.l.]: St. Martin's Press, 1983.

HEITKÖTTER, H.; HANSCHKE, S.; MAJCHRZAK, T. A. Evaluating cross-platform development approaches for mobile applications. In: SPRINGER. **International Conference on Web Information Systems and Technologies**. [S.l.], 2012. p. 120–138.

IONIC. 2015. Disponível em: <http://ionicframework.com/docs/overview/>.

KHANNA, R.; HARTINGTON, M. **Getting Started with Ionic**. 1. ed. [S.l.]: Packt publishing, 2016.

LOPES, S. **A web mobile: programe para um mundo de muitos dispositivos**. [S.l.]: Editora Casa do Código, 2014.

LUÍS, J. **Desenvolvimento para dispositivos móveis: Nativo x Cross Platform**. 2012. Disponível em: <http://www.teclogica.com.br/blog/desenvolvimento-para-dispositivos-moveis-nativo-x-cross-platform/>.

MEIRELLES, F. S. **Pesquisa: Administração e Uso da TI -Tecnologia de Informação nas Empresas**. 27. ed. FGV-EAESP: GVcia, 2016.

MEYER, M. **A história do Android**. 2015. Disponível em: <https://www.oficinadanet.com.br/post/13939-a-historia-do-android>.

MORRIS, B. **The symbian OS architecture sourcebook: design and solution of a mobile phone OS.** England: John Wiley & Sons, 2007.

PALMIERI, M.; SINGH, I.; CICCHETTI, A. Comparison of cross-platform mobile development tools. In: IEEE. **Intelligence in Next Generation Networks (ICIN), 2012 16th International Conference on**. [S.l.], 2012. p. 179–186.

PEW RESEARCH CENTER. **Technology Device Ownership: 2015**. 2015. Disponível em: <http://www.pewinternet.org/2015/10/29/technology-device-ownership-2015/>.

RAJ, C. R.; TOLETY, S. B. A study on approaches to build cross-platform mobile applications and criteria to select appropriate approach. In: IEEE. **2012 Annual IEEE India Conference (INDICON)**. [S.l.], 2012. p. 625–629.

REDDA, Y. A. Cross platform mobile applications development: Mobile apps mobility. Institutt for datateknikk og informasjonsvitenskap, 2012.

ROCHA, A. M.; NETO, R. M. F. Introdução à arquitetura apple ios. Departamento de Ciência da Computação – Universidade Federal de Goiás (UFG) –Campus Catalão, 2014.

SILBERSCHATZ, A.; GALVIN, P. B.; GAGNE, G. **Sistemas operacionais com Java**. edição. Rio de Janeiro: Campus, 2004.

SOUSA Édipo da S. **Uma análise comparativa de ferramentas de desenvolvimento multiplataforma para dispositivos móveis**. Universidade Federal do Ceará - Instituto UFC Virtual, Fortaleza: [s.n.], 2014.

SOUZA, E. de. **Symbian é um sistema operacional da Nokia Lançado em 1997**. 2014. Disponível em: <http://www.techtudo.com.br/tudo-sobre/symbian.html/>.

TOLEDO, J. M.; DEUS, G. D. de. **Desenvolvimento em Smartphones-Aplicativos Nativos e Web**. 2012.

UEDA, L.; SILVA, V. M. d. Maas: sistema para integração de compromissos acadêmicos empregando tecnologia móvel. Curitiba, 2013.

Frameworks para desenvolvimento de aplicações móveis multiplataforma no Ceará

Esta pesquisa é um levantamento dos frameworks de desenvolvimento para aplicações móveis multiplataforma mais utilizados no mercado de TI do Ceará. Servirá como parte do Trabalho de Conclusão de Curso do aluno Daniel Wolney com o objetivo de obtenção do título de Bacharel em Sistemas de Informação na Universidade Federal do Ceará.

*Obrigatório

Função:

Sua resposta

Empresa: *

Sua resposta

Área de atuação da empresa:

Sua resposta

Quantos projetos mobile ativos existem na sua empresa? *

Sua resposta

Quantos projetos existem na sua empresa em cada framework abaixo?

	1	2	3	4 ou mais
	○	○	○	○

Appcelerator Titanium Mobile				
Famous	○	○	○	○
Framework 7	○	○	○	○
Ionic	○	○	○	○
Intel XDK	○	○	○	○
JQ Touch	○	○	○	○
jQuery Mobile	○	○	○	○
Marmalade	○	○	○	○
PhoneGap	○	○	○	○
Qt/QML	○	○	○	○
Sencha Touch	○	○	○	○
Xamarin	○	○	○	○
SDK Nativo	○	○	○	○
Outro(s)	○	○	○	○

Se marcou a opção "Outro(s)", indique-o(s).

Sua resposta

ENVIAR

Nunca envie senhas pelo Formulários Google.