Análise comparativa entre ferramentas de desenvolvimento de aplicativos móveis multiplataforma utilizando características da ISO/IEC 25010 por meio da implementação de um cenário pré-definido

Gustavo Ribeiro Miranda[1], Daniela Marques[1]

[1]Instituto Federal de Ciência e Tecnologia de São Paulo (IFSP)
Hortolândia – SP – Brasil

gustavo.ribeiro@aluno.ifsp.edu.br, marquesdaniela@ifsp.edu.br

Abstract. *The purpose of this work is to compare two mobile application development tools, one hybrid-native (React Native) and one hybrid (Ionic) development. The analysis and criteria are based on ISO / IEC 25000 standards: Performance Efficiency, Functional Adequacy and Usability. For the comparison, it will be proposed a scenario of development of an application with the same functionalities in both tools. At the end, the user will answer some questions based on their own experience so that the analysis can be done and check the predefined criteria of the ISO standard.*

Resumo. *Este trabalho tem por objetivo comparar duas ferramentas de desenvolvimento de aplicativos móveis, sendo uma de desenvolvimento híbrido-nativo (React Native) e outra de desenvolvimento híbrido (Ionic). A análise e os critérios são baseados nas normas ISO/IEC 25000: Eficiência de Desempenho, Adequação Funcional e Usabilidade. Para a comparação, será proposto um cenário de desenvolvimento de um aplicativo com as mesmas funcionalidades nas duas ferramentas. Ao final, o usuário irá responder algumas questões baseado na sua experiência para que, posteriormente, possa ser feita a análise e verificar os critérios pré-definidos da norma ISO observados.*

1. Introdução

Aplicativos móveis estão ganhando cada vez mais notoriedade e participação no mercado de desenvolvimento de software devido à grande popularidade dos smartphones. Por este motivo, há uma alta busca por profissionais especializados nas ferramentas e linguagens de programação para criação de aplicativos [BURGER 2017]. No Brasil existem mais de 280 milhões de dispositivos conectados à internet, dentre *notebooks, tablets* e *smartphones*, o que representa 1,4 dispositivo portátil por habitante e os aplicativos capturam mais de metade do tempo global de mídia digital gasto [LOBO 2017].

Junto com este novo e grande mercado, novos desafios para os desenvolvedores de software começam a aparecer. Diferentes linguagens de programação, IDEs e padrões precisam ser rapidamente compreendidos para o desenvolvimento de aplicativos nativos para várias plataformas. Para que se atinja o público-alvo desejado pela empresa,

é necessário que o aplicativo esteja nas principais lojas das duas maiores plataformas de dispositivos móveis hoje: Android com a loja Google Play e iOS com a loja App Store.

Porém como manter um desenvolvimento de software acelerado dado que cada plataforma exige um novo ambiente completamente diferentes dos demais ? [HOLZINGER; TREITLER; SLANY 2012].

Segundo Corral et al, (2012), o desenvolvimento híbrido utilizando as tecnologias web (HTML, CSS e JavaScript) possui uma grande perda de performance pois, em sua arquitetura base, toda a aplicação é acoplada numa *webview*, que traz uma camada a mais de abstração para acesso a API nativa do celular, além dos problemas de usabilidade, dado que, não existe um padrão de interface, o que prejudica a experiência do usuário. Em contraponto, o processo de desenvolvimento é acelerado devido ao uso de uma plataforma única que utiliza as principais tecnologias já difundidas na *web*.

Desenvolver nativo ou multiplataforma possui vantagens e desvantagens. A partir deste ambiente é que nasce um novo modo de desenvolvimento que é o desenvolvimento híbrido-nativo, ou seja, uma plataforma que possibilita que o desenvolvedor possa desenvolver um aplicativo onde o mesmo código pode ser transformado em código nativo da plataforma desejada (Android ou IOS).

Neste contexto, este trabalho realizará uma análise comparativa entre duas ferramentas de desenvolvimento móvel, utilizando como parâmetro características da norma NBR ISO/IEC 25000, também conhecida como SQuaRE. O objetivo desta comparação é identificar quais ferramentas se destacam em quais características para resolução do mesmo problema: desenvolvimento híbrido com *webview* e desenvolvimento híbrido-nativo gerando um aplicativo nativo.

2. Referencial Teórico e Ferramentas

2.1. Desenvolvimento Híbrido

O desenvolvimento de aplicativos para plataformas móveis veio ganhando notoriedade ao longo do tempo, plataformas de desenvolvimento híbrido aumentaram o uso no mercado com a promessa de um desenvolvimento acelerado por meio de uma curva de aprendizado rápida. O desenvolvimento de aplicativo que funcione em várias plataformas consiste basicamente em apenas dois elementos: (i) um componente *web*, constituído por um código HTML5, CSS e Javascript e (ii) um *container*, conforme a Figura 1, integrando as funcionalidades que os dispositivos oferecem [Bezerra 2016].

De acordo com Corral et al (2012), no desenvolvimento híbrido existem três grandes pilares, que definem se a ferramenta oferece desenvolvimento híbrido ou não, as principais características são:

- *Online*: toda interação com o aplicativo é feita apenas pela *internet*, aliás, o aplicativo necessita de respostas de requisições HTTP.
- Responsividade: através deste recurso é possível que se crie um mesmo aplicativo adaptável a diferentes interfaces, de plataformas diferentes como: *desktop, tablet* e *smartphone*.
- Manutenção: é necessário realizar uma atualização do aplicativo e aguardar o usuário atualizar com a nova versão disponibilizada na loja, entretanto, em

um aplicativo *web*, será atualizado *on-the-fly*, ou seja, o usuário tem a nova versão do aplicativo sem necessidade de atualização.

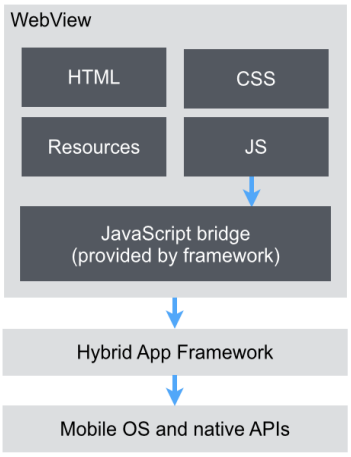


Figura 1. Arquitetura de um aplicativo híbrido

Fonte: [RIPKENS 2014]

2.1.1. Apache Cordova

O Apache Cordova foi criado pela Nitobi Software que foi comprada pela Adobe em 2011 onde teve seu nome mudado para "Phonegap". Logo após seu sucesso, teve seu código doado para a Apache Software Foundation a fim de garantir de que outras empresas e/ou pessoas pudessem contribuir com o projeto, tornando o projeto *open-source* [Sampaio; Rodrigues 2012].

O Cordova é um *framework* para desenvolvimento híbrido que possibilita acesso às funcionalidades nativas dos dispositivos conforme apresentado na Figura 2. Muitas ferramentas são criadas em cima deste *framework*, provendo bibliotecas para criação de um front-end que se aproxime de um aplicativo nativo padrão utilizando componentes visuais prontos como por exemplo: botões, modais, formulários ou *grid*. Alguns exemplos dessas ferramentas são: Materialize, Bootstrap, JQuery Mobile, o Ionic, entre outros [RIPKENS 2014].

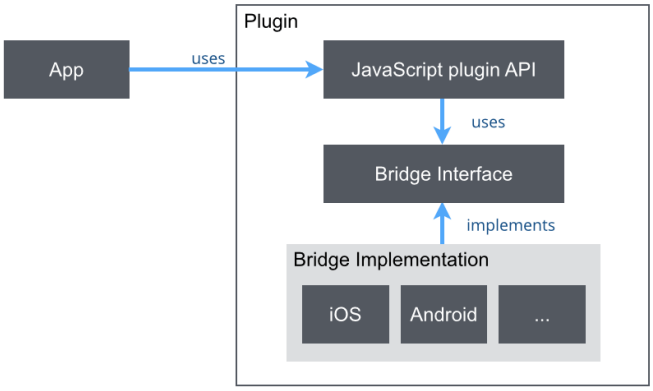


Figura 2. Implementação e acesso do Cordova

Fonte: [RIPKENS 2014]

2.1.2. Ionic

O Ionic é um *framework* para criação de aplicativos híbridos, utilizando as tecnologias *web* como: HTML, CSS e JavaScript. O Ionic assim como outras ferramentas, utiliza o Apache Cordova como interface para acesso aos dispositivos e sensores nativos do aparelho móvel. O Ionic utiliza o AngularJS para a criação do *front-end* possibilitando um melhor controle, qualidade de código e ferramentas que trarão novas possibilidades para a implementação de código na camada de visualização do aplicativo conforme a Figura 3 [BEZERRA; SCHIMIGUEL 2016].

2.1.3. Angular

O Angular é um *framework front-end* baseado em TypeScript e atualmente é mantido pela equipe técnica do Google. O Angular representa uma reescrita do projeto AngularJS feito pela mesma equipe.

2.1.4. TypeScript

O Angular utiliza o TypeScript como *transpiler* da linguagem Javascript. Uma das vantagens de se utilizar o typescript são: i) a habilidade de poder escrever o javascript em suas últimas revisões e com as novas propostas do ECMAScript, isto possibilita um maior número de funcionalidades para a linguagem e ii) a tipagem dos dados, que é inexistente no Javascript.

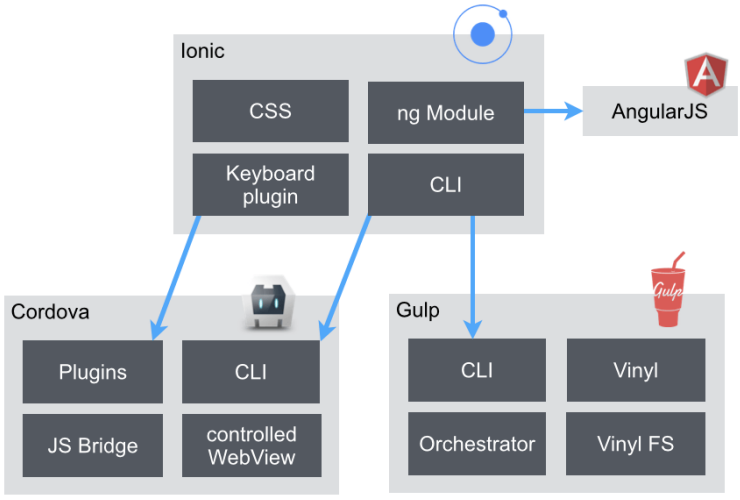


Figura 3. Arquitetura de uma aplicação Ionic

Fonte: [RIPKENS 2014]

2.2. Desenvolvimento Híbrido-Nativo

Outro meio possível para se construir um aplicativo é da forma híbrida-nativa, que consiste em duas partes: (i) a primeira parte é o código alto-nível criado em uma tecnologia com um código mais legível e de fácil entendimento e (ii) a segunda parte é um shell nativo que transpila de código alto-nível para baixo-nível, neste caso, o código nativo da plataforma.

No processo de transpilação, um transcompilador ou transpilador de fonte para fonte é um tipo de compilador que pega o código fonte de um programa escrito em uma

linguagem de programação como sua entrada e produz o código fonte equivalente em outra linguagem de programação [SAYED; TRAORÉ; ABDELHALIM 2018].

2.2.1. React Native

O React Native é uma ferramenta criada e mantida pelo Facebook para criação de aplicativos nativos, utilizando o React.js como base, para reuso de componentes e gerenciamento do estado de vida da aplicação.

2.2.1.1. Funcionamento Interno

Em sua arquitetura, o React Native, nos simuladores de iOS e Android, utiliza o JavaScriptCore que é a engine (motor) utilizada para interpretar JavaScript no navegador. Este código passa pela bridge (ponte) que posteriormente irá acessar os módulos nativos, conforme mostrado na Figura 4 [GABA; RAMACHANDRAN 2018].

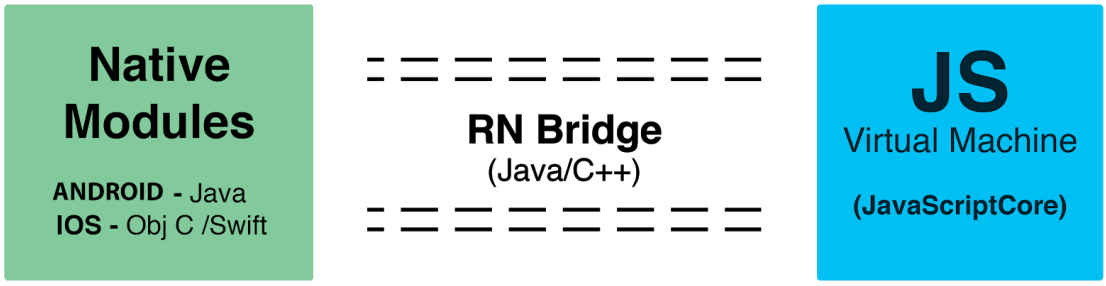


Figura 4. Módulos Nativos e Bridge no React Native

Fonte: [GABA; RAMACHANDRAN 2018]

2.2.1.2. Bridge – Nativo para Híbrido e Híbrido para Nativo

Em alguns casos, o aplicativo necessita de acesso a uma API nativa pois o React Native ainda não possui um módulo correspondente. O *framework* possibilita a reutilização dos códigos existentes em: Objective-C, Swift ou C++ sem ter que implementá-lo em JavaScript, conforme observado na Figura 5. Esta funcionalidade é útil para casos específicos em que é necessário que se crie um código de alto desempenho para processamento de imagens ou banco de dados, por exemplo [GABA; RAMACHANDRAN 2018].

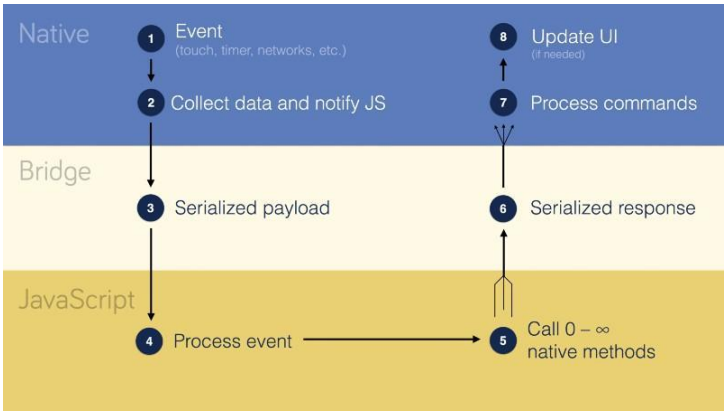


Figura 5. Funcionamento das Bridges

Fonte: [GABA; RAMACHANDRAN 2018]

Assim como a comunicação tradicional fica em Javascript comunicando com o Java, Swift/Objective-C a comunicação pode ser feita vice-versa, ou seja, Java, Switch/Objective-C comunicando com o Javascript.

2.2.1.3. Plataformas Suportadas

Oficialmente, com o React Native é possível gerar código nas plataformas: Android, iOS e WebOS (Sistema utilizado em SmartTVs). Entretanto, a Microsoft já lançou um módulo para construir aplicativos em React Native para as plataformas: Windows 10, Windows 10 Mobile e Xbox One (UWP) [MICROSOFT 2018].

2.2.2. React.js

Também denominado como React, trata-se de uma biblioteca Javascript para criação de interfaces de usuário. Criado e mantido pelo Facebook, o React segundo a Similartech (2018) possui uma alta adoção no mercado e hoje é utilizado por grandes sites como: Yahoo, Bing, Microsoft, Imdb, BBC, Paypal, Mercado Livre, entre outros, devido a características específicas da qual o *framework* efetua o controle de componentização e gerenciamento do ciclo de vida de um componente.

2.2.2.1. Virtual DOM

O React utiliza o conceito de Virtual DOM, que é uma representação virtual de uma interface de usuário é mantida em memória e sincronizada no DOM "Real".

Esta abordagem habilita a API declarativa do React que determina em qual estado a interface de usuário deve estar e garante que o DOM corresponda a esse estado. Esta abstração possibilita a manipulação de atributos, eventos e atualização do DOM manual, conforme apresentado na Figura 6 [Naukri Engineering 2018].

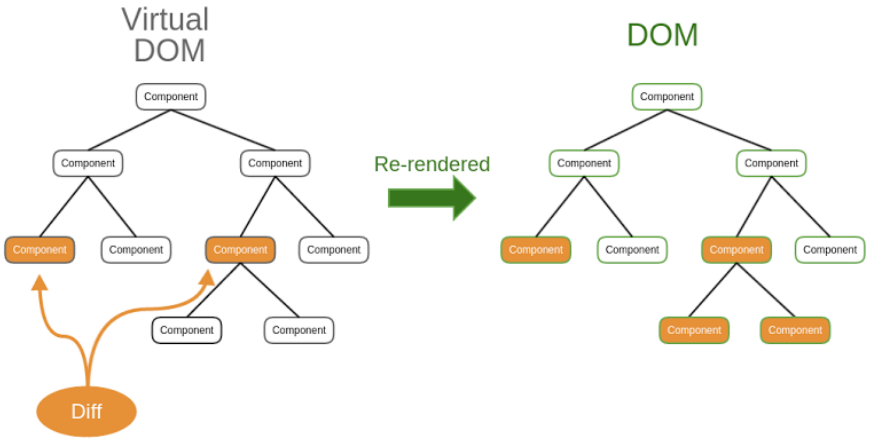


Figura 6. Virtual DOM vs Real DOM

Fonte: [Naukri Engineering 2018]

2.3. Desenvolvimento Híbrido-Nativo vs Desenvolvimento Híbrido

Embora haja semelhanças entre as ferramentas Ionic e React Native, ambas possuem abordagens diferentes para a construção final do aplicativo.

No Ionic, o aplicativo fica acoplado numa camada *webview* do dispositivo que faz sua simulação, entretanto é apenas um *browser* acessando sensores nativos do

celular através do Apache Cordova. Já no React Native, após a criação do aplicativo, o código passa por uma fase de transpilação de código híbrido (*web*) para código nativo do alvo selecionado, seja ele: Android ou iOS.

2.4. ISO/IEC 25000 SQuaRE

Na engenharia de software, a qualidade de software é a área que visa garantir bons produtos através de processos, visando a melhoria contínua da qualidade [SOMMERVILLE 2011].

A norma técnica SQuaRE consiste de cinco divisões de qualidade, sendo:

- ISO/IEC 2500n – Divisão gestão de qualidade;
- ISO/IEC 2501n – Divisão modelo de qualidade;
- ISO/IEC 2502n – Divisão medição da qualidade;
- ISO/IEC 2503n – Divisão requisitos de qualidade;
- ISO/IEC 2504n – Divisão avaliação da qualidade.

A Divisão Modelo de Qualidade é referente a avaliação de um produto de software. É nesta divisão que são citadas as características a serem considerados na avaliação das propriedades de um produto de software. A Figura 7 apresenta as 8 características de qualidade definidos na ISO e suas subcaracterísticas. A Tabela 1 descreve resumidamente cada subcaracterística da ISO.

Figura 7. Qualidade de um produto de software ISO/IEC 25000

Fonte: [ISO/IEC 25010 2011]

3. Metodologia de Desenvolvimento

A primeira etapa foi realizar um estudo na literatura de desenvolvimento multiplataforma e norma técnica SQuaRE. Verificou-se quais as características da norma poderiam ser aplicadas no trabalho considerando a análise das ferramentas.

Após definido os critérios a serem avaliados foi necessário estipular um roteiro de desenvolvimento do aplicativo para que fosse enviado às pessoas que participariam da pesquisa. Estudou-se as funcionalidades mais comuns e requisitadas nos aplicativos para que fossem utilizadas no trabalho.

Cada pessoa realizaria o cenário proposto nas duas ferramentas e responderia às perguntas elaboradas para que ao final levassem dados para a análise. Foi utilizada a abordagem GQM *(Goal-Question-Metrics)* para a elaboração do questionário.

A abordagem GQM baseia-se na teoria de que todas as medições devem ser [1] orientadas para objetivos, ou seja, tem que haver alguma lógica e necessidade de coletar

medições, ao invés de coletar por uma questão de coleta. Cada medida coletada é declarada em termos dos principais objetivos. [2] As perguntas são derivadas das metas e ajudam a refinar, articular e determinar se as metas podem ser alcançadas. [3] As métricas ou medidas que são coletadas são então usadas para responder às perguntas de uma maneira quantificável, conforme a Figura 8.

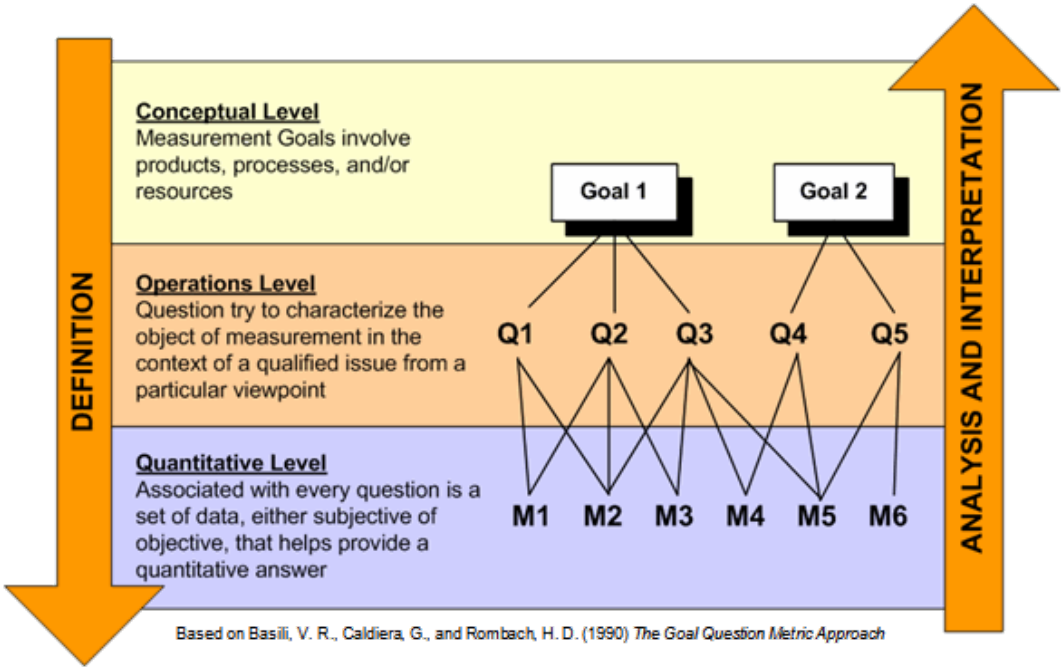


Figura 8. Go Question Metric

Fonte: [HUETHER 2019]

4. Desenvolvimento do Trabalho

A primeira atividade desenvolvida neste trabalho foi o estudo das ferramentas para desenvolvimento de aplicativos móveis multiplataforma. O resultado deste estudo foi a seleção das ferramentas React Native e Ionic para o trabalho.

O estudo da ISO/IEC 25000 (SquaRE) resultou na seleção das características e subcaracterísticas que poderiam ser aplicadas na análise comparativa das ferramentas. Esses são os critérios selecionados para uso neste trabalho: Eficiência e Desempenho, Adequação Funcional e Usabilidade, conforme a Tabela 1.

Após selecionado os critérios, a próxima atividade era definir o cenário a ser desenvolvido na pesquisa. Neste caso, considerou-se como premissa que o estudo deveria durar de 30 a 40 minutos.

O cenário proposto foi criar um aplicativo com três abas (Figura 9). São eles:

1. Componentes: esta aba tem como objetivo mostrar componentes com acesso a API nativa do dispositivo, tais como: acesso a câmera, giroscópio e armazenamento de arquivos do dispositivo.
2. Geolocalização: demonstrar o uso da geolocalização utilizando a API do Google Maps para inserir um marcador no local atual do dispositivo.
3. OAuth: esta aba irá demonstrar o uso de APIs do Google e do Facebook, a fim de demonstrar o suporte as integrações OAuth na plataforma testada.

Tabela 1. Característica e subcaracterísticas da ISO

Característica	**Subcaracterística**	**Descrição**
Adequação Funcional	Adequação Funcional	Capacidade de realizar tarefas e certos objetivos de maneira fácil
	Integridade Funcional	Capacidade de atender às tarefas e aos objetivos específicos do usuário a que foi destinado
	Correção Funcional	Capacidade de fornecer resultados corretos e com precisão
Eficiência de desempenho	Comportamento em relação ao uso do tempo	Capacidade de fornecer tempos de resposta e processamento apropriados quando o software executa suas funções
	Utilização de recursos	Grau com que os tipos e as quantidades de recursos usados atendem aos requisitos
	Capacidade	Grau com que a capacidade máxima de parâmetros do software atende aos requisitos
Usabilidade	Reconhecibilidade	Grau com que o usuário é capaz de reconhecer se o produto é adequado às suas necessidades
	Apreensibilidade	Capacidade que o software possui de ser usado para alcançar objetivos específicos de aprendizagem, de forma eficiente, eficaz e sem riscos, garantindo que o usuário se sinta satisfeito no contexto em questão
	Operacionalidade	Capacidade do software de permitir ao usuário operá-lo e controlá-lo de forma fácil.
	Proteção de erro	Capacidade de proteger os usuários de cometer falhas
	Acessibilidade	Capacidade de ser usado por usuários com diferentes características e habilidades para alcançar objetivos especificados

Fonte: [ISO/IEC 25010 2011]

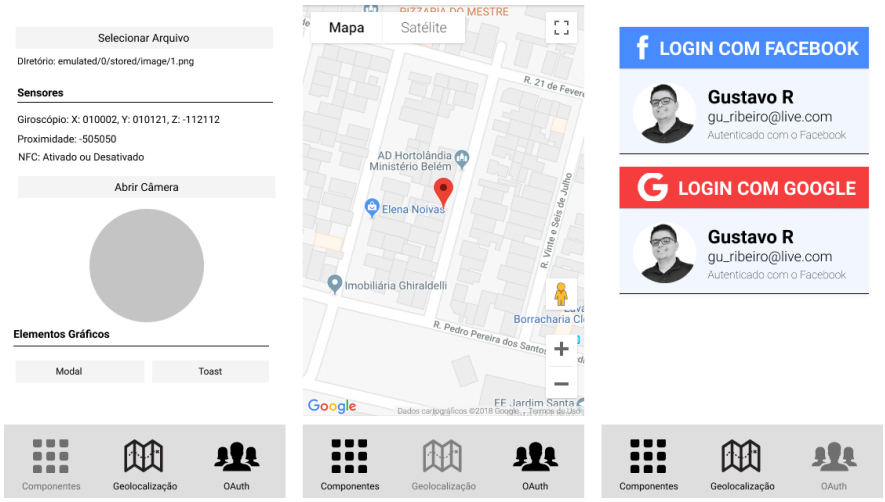


Figura 9. Protótipo criado para criação do desafio a ser proposto ao aluno

Definido o cenário a ser realizado pelos participantes, foi elaborado o questionário a ser respondido. Utilizou-se a abordagem GQM para criação do questionário, que possui uma abordagem *top-down* para medição de software [PRESSMAN 2011]. Neste trabalho, esta abordagem foi usada para derivação das questões a serem aplicadas, as Metas *(Goal)* são as subcaracterísticas selecionadas da norma ISO, a partir das metas foram elaboradas as Questões *(Questions)* e por fim, o resultado do questionário traria as Métricas para se fazer a análise dos dados.

Cada participante recebeu o cenário de criação do aplicativo e após realizá-lo nas ferramentas propostas tinham que responder o questionário (Tabela 2). As questões relativas aos critérios da ISO utilizam a escala Likert, onde a escala varia de 1 (muito insatisfatório) a 5 (muito satisfatório).

Tabela 2. Questionário x Características ISO/IEC 25000

Número da Questão	Questionário	Característica ISO
1	Qual ferramenta foi utilizada? React Native ou Ionic	Não aplicado.
2	Possui alguma experiência com desenvolvimento web (HTML, CSS, JS)? Sim ou Não	Não aplicado.
3	Se a resposta anterior foi sim, há quanto tempo? 0-1 anos, 2-3 anos, 4-5 anos e Acima de 5 anos	Não aplicado.
4	Qual foi a capacidade da ferramenta de fornecer tempos de resposta e processamento apropriados?	Eficiência e Desempenho
5	O quanto a ferramenta foi capaz de resolver o problema proposto?	Adequação Funcional
6	O quanto a ferramenta foi capaz de realizar tarefas de maneira fácil?	Adequação Funcional
7	O quanto a ferramenta foi fácil de operar/controlá-lo?	Usabilidade
8	O quanto a ferramenta foi fácil de aprender a ser utilizada?	Usabilidade

Considerando as características e subcaracterísticas ISO/IEC, este trabalho propõe o seguinte uso:

- Característica Eficiência e Desempenho: avaliar as subcaracterística de Comportamento em Relação ao Uso do Tempo;
- Característica Adequação Funcional: avaliar as subcaracterísticas de Adequação Funcional e Integridade Funcional;
- Característica Usabilidade: avaliar as subcaracterísticas Operacionalidade e Apreensibilidade.

O cenário e o questionário foram enviados aos profissionais da tecnologia da informação. O formulário foi preenchido no *Google Forms*.

5. Resultados Obtidos

O questionário foi respondido por cinco (5) pessoas, todos são alunos não concluintes do curso de Análise e Desenvolvimento de Sistemas (ADS) do Instituto Federal de Educação, Ciência e Tecnologia de São Paulo.

As três primeiras questões não eram relacionadas aos critérios a serem avaliados, mas serviram para identificar a ferramenta que estava sendo analisada e o nível de conhecimento dos respondentes sobre web. A primeira questão se referia a ferramenta utilizada.

As próximas duas questões se referiam a experiência de cada respondente. Todos (100%) afirmaram conhecer as principais tecnologias utilizadas na web (Figura 10) e a maioria dos respondentes (66,7%) possuíam de 2 a 3 anos de experiência (Figura 11).

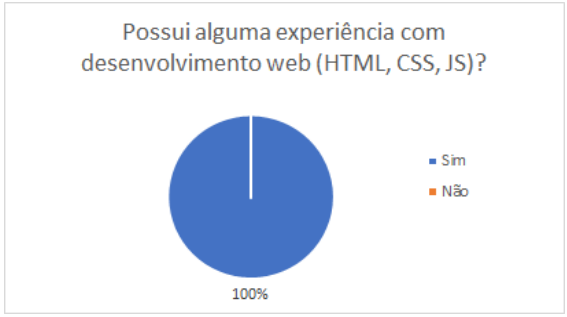


Figura 10. Resultado dos dados referente ao conhecimento de tecnologias web (Pergunta 2)

Figura 11. Análise dos dados referentes aos anos de experiência com desenvolvimento web (Pergunta 3)

5.1. Eficiência de Desempenho

A quarta questão é relacionada aos critérios a serem analisados foi referente a característica Eficiência de Desempenho e a subcaracterística Comportamento em Relação ao Uso do Tempo. Como explicado anteriormente, o Ionic utiliza a webview como renderizador de tela para ter um *preview* no desenvolvimento. Logo, para ter um *preview* do aplicativo é utilizado o navegador, ou seja, o *feedback* ao desenvolvedor quando muda um código é quase instantâneo.

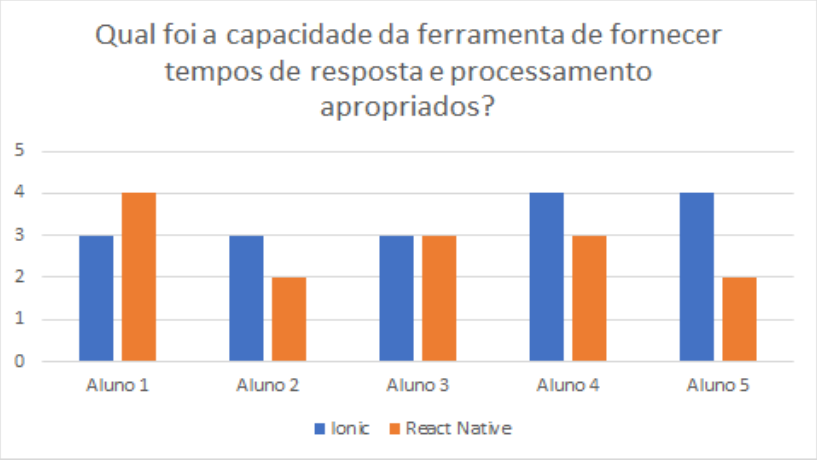


Figura 12. Resultado dos dados referente a característica Eficiência e Desempenho (Pergunta 4)

Analisando os dados obtidos, pode-se considerar que um fator contribuinte para este resultado (Figura 12) é a obrigatoriedade da utilização do emulador de Android no Windows no React Native. Isto pode gerar problemas arquiteturais pois os dispositivos Android utilizam a arquitetura de processadores ARM e o Windows por padrão, em sua maioria são processadores Intel ou AMD, que possuem a arquitetura x86. [COSTA 2018].

Esse problema é abstraído por uma ferramenta criada pelo Facebook chamada Expo, entretanto, o tempo ainda é maior do que o navegador devolve ao Ionic e, isto implica diretamente no tempo de *feedback* ao usuário quando uma alteração é feita no código.

5.2. Adequação Funcional

As duas questões referentes a característica Adequação Funcional avaliaram as subcaracterísticas sobre Integridade Funcional e Adequação Funcional. O resultado obtido para Integridade Funcional (Figura 13) mostra que a ferramenta React Native foi considerada um pouco melhor na opinião dos respondentes (4 respondentes consideraram a ferramenta Muito Satisfatória contra 3 da Ionic), neste caso foi utilizado a quinta pergunta.

Em relação a subcaracterística Adequação Funcional relacionado à sexta pergunta do formulário, o Ionic mostrou-se melhor em relação a React Native, como é possível perceber pelos resultados obtidos que são apresentados na Figura 14.

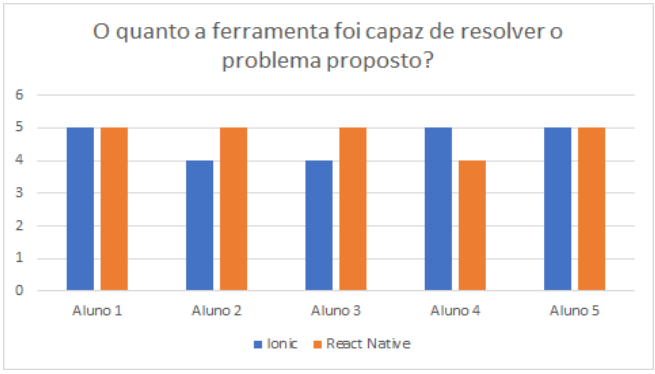


Figura 13. Resultado dos dados referente a subcaracterística Integridade Funcional (Pergunta 5)

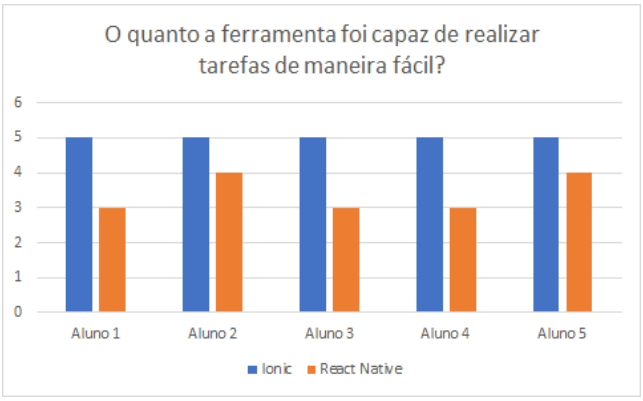


Figura 14. Resultado dos dados referente a subcaracterística Adequação Funcional (Pergunta 6)

5.3. Usabilidade

Na característica Usabilidade foram avaliadas duas subcaracterísticas: Operacionabilidade e Apreensibilidade. Nota-se que nesta característica, o Ionic se mostrou melhor na opinião dos respondentes. A Figura 15 mostra o resultado em relação a subcaracterística Operacionabilidade e a Figura 16 refere-se a Apreensibilidade, relacionado à sétima pergunta do formulário.

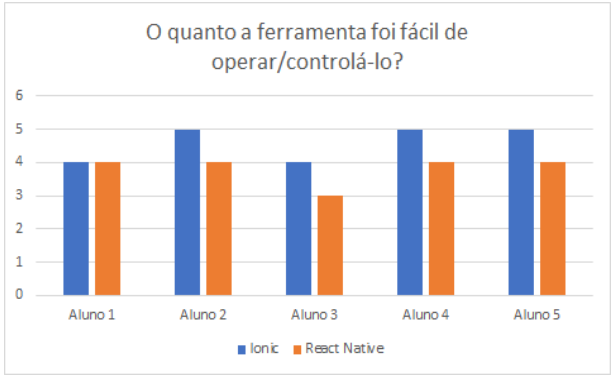


Figura 15. Resultado dos dados referente a subcaracterística Operacionabilidade (Pergunta 7)

Observa-se que na opinião dos respondentes, na subcaracterística Apreensibilidade, a ferramenta Ionic tem mais vantagem do que a React Native. Esta subcaracterística foi a que mostrou mais opiniões divergentes de toda a pesquisa, relacionado à oitava pergunta do formulário.

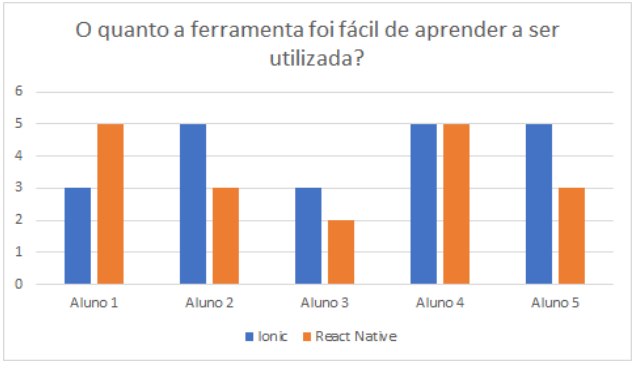


Figura 16. Resultado dos dados referente a subcaracterística Apreensibilidade (Pergunta 8)

6. Conclusão

Vários aprendizados foram obtidos durante o decorrer do trabalho. O primeiro deles foi adquirido ao preparar o cenário para a pesquisa. Notou-se algumas diferenças nas ferramentas e boas práticas adotadas.

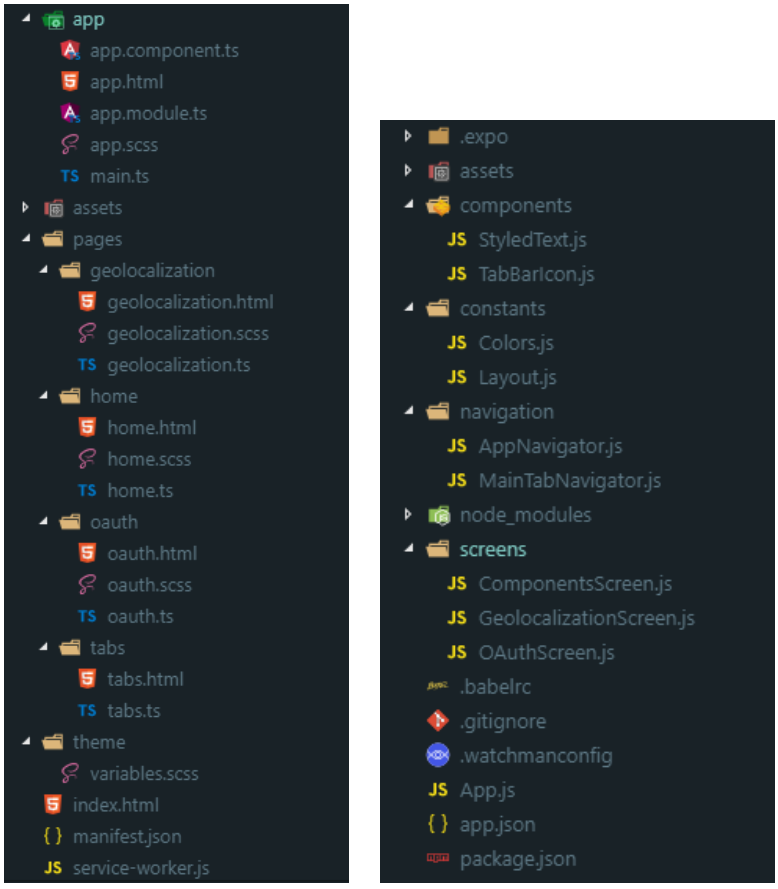
Com base na Figura 17, ignorando a pasta node_modules (contém as dependências do projeto) que estão em ambos os projetos e a pasta assets que possui o conteúdo estático do aplicativo, pode-se observar que o Ionic utiliza uma maior segregação de responsabilidade dos componentes, separando a regra de negócio no .ts (TypeScript) com tipagem forte, o .html (HTML) que faz o *two-way data binding* com o .ts e o .scss (SASS) que estiliza o componente, onde é pré-processado e posteriormente é compilado como um .css (CSS). Essa organização vem por padrão nos projetos com *template* e é uma boa prática recomendada pela comunidade técnica do Angular.

Por outro lado, o React Native dá uma maior liberdade ao desenvolvimento de forma que o desenvolvedor ou a equipe possa criar um padrão ou guia de boas práticas para organização do código de maneira que se adapte aos padrões de codificação utilizados.

Em relação aos aprendizados durante o desenvolvimento, destacam-se (i) o Expo é uma ferramenta para React Native que possibilita gerar *bundles* e obter um *hot reload* de código e emulador do aplicativo que está sendo desenvolvido no celular, entretanto, ela possui algumas limitações com GPS e *Loader*, que só podem ser resolvidos fazendo o "*eject*" da aplicação e (ii) para o Ionic, na versão quatro, o *framework* possibilita a utilização de *web components*, ou seja, não necessariamente devemos utilizar o *Angular* como *framework* principal da ferramenta. Entretanto, assim como toda versão recém lançada, ainda existem poucas implementações com outros *frameworks* conhecidos no mercado.

Em relação ao objetivo do projeto, a Tabela 3 mostra um resumo de cada característica e subcaracterística da ISO/IEC 25000 e a ferramenta que teve melhor avaliação em cada quesito de acordo com a opinião dos respondentes. O Ionic

demonstrou ser capaz de resolver melhor os desafios propostos para este trabalho. O resultado pode ser influenciado pelo que já foi citado anteriormente, a habilidade de se utilizar o navegador enquanto se desenvolve e a facilidade relacionada às tecnologias utilizadas na ferramenta como HTML, CSS e JS com Angular.



Ionic **React Native**

Figura 17. Comparação entre a organização dos projetos

Tabela 3. Característica e Subcaracterísticas da ISO/IEC x Ferramenta

Característica	Subcaracterística	Ferramenta
Eficiência e Desempenho	Comportamento em relação ao uso do tempo	Ionic
Adequação Funcional	Integridade Funcional	React Native
	Adequação Funcional	Ionic
Usabilidade	Operacionalidade	Ionic
	Apreensibilidade	Ionic

A principal utilidade do React Native no mercado é a utilização de um *framework* que se aproveita bem da reutilização de código utilizando componentes em

aplicações grandes e que requerem uma maior estrutura (arquitetura). Essa característica acabou não sendo bem visualizada nos cenários propostos.

Ressalta-se novamente que os respondentes são alunos do curso ADS e, portanto, não se pode concluir que este resultado representa a opinião de uma maioria no mercado. O escopo abordado na pesquisa foram alunos que pudessem executar o cenário proposto, trabalhando com componentes para acessar APIs nativas do dispositivo, geolocalização e uso de APIs de terceiros (tais como Google ou Facebook) para demonstrar as integrações na plataforma.

Salienta-se a fragilidade desta pesquisa possuir apenas cinco respondentes, ou seja, pontos fora da curva de pessoas que já possuem facilidade com as ferramentas e que pode afetar positivamente ou negativamente para uma ferramenta preferida, impactando assim, diretamente no resultado exposto. Como proposta de trabalhos futuros, sugere-se expandir essa pesquisa para um número maior de pessoas e envolvendo outras ferramentas diferentes.

Durante este trabalho pude articular conhecimentos aprendidos no curso, tais como: Desenvolvimento de Sistemas Web (SWI6), Qualidade de Software (SWI5), Engenharia de Software (ESW) e Arquitetura de Software (ASWI4). Além disso, aprendi as principais evoluções das tecnologias que rodam na *web* atualmente, arquitetura de alta complexidade para aplicações multiplataforma, características da ISO/IEC 25000 e o estudo de métricas através de questões, através do método GQM.

6. Referências Bibliográficas

BEZERRA, P. T., SCHIMIGUEL, J. Desenvolvimento de aplicações mobile crossplatform utilizando phonegap. 2016. Disponível em: <http://eumed.net/cursecon/ecolat/br/16/phonegap.html>. Acesso em: 13 jun. 2018.

BLOM S., BOOK M., GRUHN V., HRUSHCHAKK R., KOHLER A. Write once, run anywhere: A survey of mobile runtime environments. Proceedings of the 3rd International Conference on Grid and Pervasive Computing Workshops, pp. 132-137. 2008. Disponível em: <https://ieeexplore.ieee.org/document/4539337>. Acesso em: 27 set. 2018.

BURGER, Luciana. Perspectivas do Cenário Digital Brasil 2017. ComScore. Disponível em: <https://www.comscore.com/por/Insights/Apresentacoes-e-documentos/2017/Perspectivas-do-Cenario-Digital-Brasil-2017>. Acesso em: 28 mar. 2018.

CORRAL, L., JANES, A., REMENCIUS, T. Potential Advantages and Disadvantages of Multiplatform Development Frameworks - A Vision on Mobile Environments. Procedia Computer Science, v. 10, p. 1202–1207, jan 2012. ISSN 1877-0509. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1877050912005303>. Acesso em: 10 jun. 2018.

COSTA, Ricardo. ARM x Intel: o que isso significa para quem está comprando um Android hoje?. Disponível em <https://www.tecmundo.com.br/android/65202-arm-x-intel-significa-comprando-android.htm>. Acesso em: 30 mar. 2019.

GABA, Raul; RAMACHANDRAN, Atul. React Made Native Easy. Disponível em <https://www.reactnative.guide/3-react-native-internals/3.1-react-native-internals.html>. Acesso em: 11 set. 2018.

HOLZINGER A., TREITLER P., SLANY W. (2012) Making Apps Useable on Multiple Different Mobile Platforms: On Interoperability for Business Application Development on Smartphones. In: Quirchmayr G., Basl J., You I., Xu L., Weippl E. (eds) Multidisciplinary Research and Practice for Information Systems. CD-ARES 2012. Lecture Notes in Computer Science, vol 7465. Springer, Berlin, Heidelberg Disponível em: <http://link.springer.com/chapter/10.1007/978-3-642-32498-7_14> Acesso em: 10 jun. 2018.

HUETHER, Derek. GQM: HOW DO YOU KNOW YOUR METRICS ARE ANY GOOD. Disponível em: < https://www.leadingagile.com/2013/07/gqm-how-do-you-know-your-metrics-are-any-good/>. Acesso em: 4 abr. 2019.

ISO/IEC 25010:2011. International Organization for Standardization. Disponível em: <https://www.iso.org/standard/35733.html>. Acesso em: 28 mar. 2018.

LOBO, Ana Paula. Brasil possui 280 milhões de dispositivos móveis conectáveis à Internet. Abranet. Disponível em: <http://www.abranet.org.br/Noticias/Brasil-possui-280-milhoes-de-dispositivos-moveis-conectaveis-a-Internet-1419.html>. Acesso em: 11 set. 2018.

Market Share & Web Usage Statistics. Similar Tech. Disponível em <https://www.similartech.com/technologies/react-js>. Acesso em: 12 set. 2018.

MICROSOFT. Github: React Native plugin for Universal Windows Platform (UWP). Disponível em <https://github.com/Microsoft/react-native-windows>. Acesso em: 11 set. 2018.

Naukri Engineering.Virtual Dom: Lets you write HTML as a function of state. Disponível em <https://medium.com/naukri-engineering/naukriengineering-virtual-dom-fa8019c626b>. Acesso em: 30 set. 2018.

PRESSMAN, Roger et al. (2011) Engenharia de software. São Paulo, Editora Bookman.7. Ed.

RIPKENS, Ben. Ionic: An AngularJS based framework on the rise. Disponível em <https://blog.codecentric.de/en/2014/11/ionic-angularjs-framework-on-the-rise>. Acesso em: 11 set. 2018.

SAMPAIO, C., RODRIGUES, F. (2012) Mobile Game Jam. Editora Brasport. 1. ed.

SAYED, B., TRAORÉ, I., ABDELHALIM, A. If-transpiler: Inlining of hybrid flow-sensitive security monitor for JavaScript. Computers & Security, v. 75, p. 92 - 117, jan 2018. ISSN 0167-4048. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0167404818300397>. Acesso em: 18 mar. 2019.

SOMMERVILLE, Ian. (2011), Engenharia de Software, São Paulo: Pearson Addison Wesley, 9. ed.