ESTUDO COMPARATIVO ENTRE AS MODALIDADES DE DESENVOLVIMENTO MOBILE: WEB APP, HÍBRIDA E NATIVA.

Leonardo Amaral Sousa[1], Leandro Brito[2], Uedson Reis[3].

UNIME – União Metropolitana de educação e Cultura
42.700-00 – Lauro de Freitas – BA – Brasil

{leonardo.amaral.sousa[1], uedsonreis[3]}@gmail.com, leandro.brito@kroton.com.br[2]

RESUMO

O desenvolvimento de aplicações móveis tornou-se um ramo muito interessante nos últimos anos. Porém existia somente uma única modalidade de desenvolvimento, a nativa que ainda possuía pouca documentação. Assim surgiram diferentes modalidades de desenvolvimento com o objetivo de suprir a necessidade, que continua crescendo, de aplicativos móveis, porém o diferencial está nas características que cada modalidade possui. Esse artigo analisa, a partir de um estudo de caso, o uso de três destas modalidades: Web App, Híbrida e a Nativa. Ao final, foi possível identificar algumas vantagens e desvantagens de cada modalidade comparada às outras.

• 1. INTRODUÇÃO

No inicio do desenvolvimento para dispositivos móveis, existia somente uma modalidade de desenvolvimento, a nativa. Porém a mesma possuía pouca documentação e necessitava de uma curva de aprendizado extensa, tornando complicado e custoso a capacitação de novos profissionais. Assim os profissionais especialistas de desenvolvimento móvel criaram duas alternativas de desenvolvimento para dispositivos móveis, as modalidades web app e a hibrida.

Estas duas modalidades tem um intuito de transformar o desenvolvimento de aplicativos móveis acessível aos profissionais que não possuem um conhecimento especifico para tal atividade e ao mesmo tempo melhorar o processo de desenvolvimento, pois traz como característica a capacidade de um único código desenvolvido atender a diversas plataformas, sistemas operacionais, característica nomeada como multiplataforma. Porém cada modalidade possuem características determinantes, que limitam ou facilitam o desenvolvimento de um determinado app, por isso a discursão entre qual modalidade utilizar tornou-se frequente entre os profissionais que atuam no mercado móvel.

Este trabalho tem o objetivo de realizar um estudo comparativo entre as modalidades de desenvolvimento, baseado na necessidade de um estudo de caso, que apresenta a implementação de uma aplicação móvel nas modalidades *web app*, híbrida e nativa.

2. METODOLOGIA

Inicialmente será desenvolvida uma pesquisa teórica sobre as modalidades de desenvolvimento de aplicativos móveis, com intuito de analisar suas características e identificar as vantagens e desvantagens de uma modalidade quando comparada as outras. Logo após será desenvolvida um

estudo de caso onde teremos como contexto, o desenvolvimento de um projeto de aplicativo móvel, desenvolvido nas três modalidades abordadas por este trabalho, *web app*, híbrida e nativa. Após a execução do estudo de caso, serão analisados os resultados utilizando a métrica de complexidade ciclomática criada por Thomas McCabe, para definir a complexidade do projeto em cada modalidade. Para determinarmos o tamanho de cada projeto será utilizada a métrica LOC (*Lines of Code*), métrica utilizada para mensurar o tamanho de um software. Também será utilizada a métrica de pontos de função por hora, para determinarmos a produtividade de cada modalidade. Após analisarmos a complexidade, tamanho, tempo e produtividade, faremos também uma análise sobre a curva de aprendizado necessária para cada modalidade.

• 3. ESTADO DA ARTE DESENVOLVIMENTO MÓVEL

O desenvolvimento móvel utiliza tecnologias, linguagens e padrões de projetos já existentes no mercado, como podemos constatar nos guias para desenvolvedores da Google, Apple, BlackBerry e Microsoft. Mesmo utilizando as linguagens difundidas do mercado de software, é necessário possuir conhecimento específico para cada SDK(*Software Development Kit*) que é o kit de recursos necessários para o desenvolvimento de software para uma determinada linguagem de programação[1] e API (*Application Programming Interface*) que é conjunto de padrões estabelecidos para utilizar as funcionalidades de um software[2], conhecimento sobre a arquitetura de dispositivos móveis, controle de acesso utilizado pelos sistemas operacionais, permissão e distribuição da app. Diante da necessidade dos conhecimentos específicos sobre cada sistema operacional, surgiram modalidades de desenvolvimento de apps que não necessitam dos conhecimentos específicos e que reduzem o tempo de desenvolvimento, pois uma única implementação pode ser utilizada para vários sistemas operacionais, esta característica é conhecida como multiplataforma [3].

• 3.1. MODALIDADE WEB APP

Acessado geralmente por meio da *internet* e desenvolvido com linguagens suportadas por navegadores, tais como, *HTML(HyperText Markup Language)* que uma linguagem de marcação de texto para páginas web[4], *CSS (Cascading Style Sheets)* que é uma linguagem de folhas de estilo para definir o modelo de apresentação de um documento[5] e, *JavaScript*, que é um linguagem de programação interpretada utilizada em arquitetura de software cliente servidor em navegadores de web[6]. Este tipo de *software* é denominado aplicativo *web* [7]. W*eb app* são sistemas hospedados em servidores de aplicações *web*,onde suas páginas podem ser visualizadas por qualquer cliente, sejam *desktops*, *notebooks*, *tablets* ou *smartphones* mantendo a mesma experiência e identidade visual para ambos os clientes. Na figura 2 podemos entender como é a arquitetura de aplicações web apps.

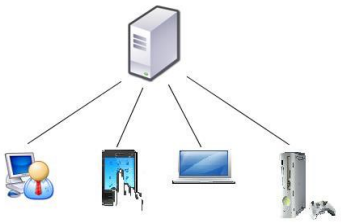

Figura 1 - Imagem ilustrando a arquitetura das web Apps.

A figura 2 demonstra a arquitetura das *web apps*, pois um único fonte hospedado em um servidor *web* pode ser executado em qualquer plataforma destino, tornando-se assim um sistema

multiplataforma. Para iniciar a desenvolver *web apps* é necessário conhecimento em *HTML* preferencialmente a versão 5 e bem como todas as suas novidades, pois são com elas que nossas *web apps* ficaram mais interessantes. É necessário também dominar o *CSS*, principalmente, a versão 3 que será muito utilizada quando for desenvolver suas *interfaces* (Meio físico ou lógico que possibilita com que dois elementos distintos se comuniquem dicionário Aurélio) com usuário. Junto com o *CSS* também é necessário conhecimento sobre WRD (*Web Responsive Design*), técnica que utiliza o HTML5 e o CSS3, para criar websites que funcionem bem em vários dispositivos e telas [8].

É necessário também conhecimento em *Javascript,* pois é a linguagem compatível com todos os navegadores, além de ter compatibilidade com *HTML* e *CSS*. O *Javascript* pode ser utilizado para criação das regras de negócios da aplicação, validações em formulários, formatação, máscaras e ajustes das páginas dinamicamente. Uma característica que deve ser levada em consideração ao se optar a desenvolver uma *web app* é a capacidade de processamento que esta modalidade oferece. Esta modalidade é acessado via *internet* então o desempenho do seu aplicativo terá relação direta com a banda utilizada pelo dispositivo. Além da velocidade de conexão com a *internet,* outro fator que limita as *web apps,* é o processamento, pois como são acessados via navegador de internet, utilizam recursos alocados por este e as regras são executadas pelo motor *Javascript* do navegador.

As *web apps* não possui acesso total ao *hardware* e funcionalidade do dispositivo, atualmente com o *HTML5*, as *web*s *apps* possuem acesso ao GPS, acelerômetro e ainda de forma limitada, [3]. A interface com usuário necessita de muita atenção durante o desenvolvimento, pois por ser multiplataforma não existe nenhuma restrição para os diversos dispositivos clientes que possuem variadas telas. Porém esta questão é facilmente resolvida com aplicação do modelo de formatação e de padronização de *design* apresentado pela WRD.

O *WRD* é uma nova tendência onde propõe aos desenvolvedores criarem *sites* e sistemas *web* preparados para se adaptarem a qualquer agente que vier a acessá-los. Essa proposta consiste na criação de um único documento que altere a apresentação do seu conteúdo para se adaptar ao dispositivo em que está sendo visualizado [8]. O *WRD* alinha conceitos e definições antigas da *web* com as novidades do *HTML5* e CSS3 para solucionar um problema novo. O WRD é utilizado também em outra modalidade de desenvolvimento móvel, como a modalidade híbrida, que veremos na próxima subseção.

3.2. MODALIDADE HÍBRIDA

Aplicações híbridas estão exatamente entre as *web apps* e as aplicações nativas, elas são parcialmente nativas e parcialmente *web apps* [9]. Aplicações híbridas possuem uma "casca" escrita em linguagem nativa e seu interior escrito em HTM5, CSS3 e *JavaScript*. Esta modalidade é muito utilizada para migrar de *web apps,* para aplicativos que sejam executados no dispositivo móvel com a finalidade de utilizar o hardware do mesmo. Na figura 3, poderemos compreender melhor a composição de um aplicativo desenvolvido na modalidade híbrida.

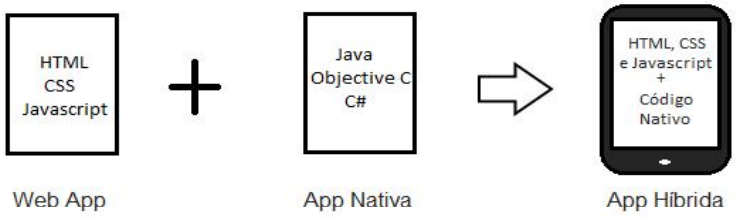

Figura 2 - Composição de uma aplicação desenvolvida na modalidade híbrida.

Para desenvolver *apps* híbridas, é necessário conhecimento em HTML5 e CSS para criar as telas da app, estiliza-las e definir a identidade visual da sua aplicação [3]. Faz-se necessário também conhecer as indicações e definições passadas pelo WRD, pois nessa modalidade, assim como no desenvolvimento de web *apps*, também é necessário o ajuste da sua aplicação de acordo com o dispositivo que esteja executando-a. Nesta modalidade o conhecimento em *Javascript* torna-se ainda mais necessário, pois o *Javascript* será utilizado tanto para as regras da aplicação, como conexão entre o código nativo do framework escolhido com a sua aplicação HTML, CSS e *Javascript* [3]. Os *frameworks* para as aplicações híbridas, possuem implementações em linguagem nativa de cada sistema operacional, tratando do acesso ao *hardware* do dispositivo [3].

Esta modalidade, assim como na modalidade nativa, necessita que possua o ambiente de desenvolvimento, composto por uma IDE (I*ntegrated Development Environment*), SDK e API destinada para o sistema operacional que deseja desenvolver. Outro fator que pode ou não aumentar o custo de um projeto de uma aplicação híbrida é a escolha do *framework*, pois existem *frameworks* gratuitos ou pagos, é interessante não só levar em consideração o custo, mas também a equipe que o elabora e o desenvolve, deve ser levado em consideração também a solidez da empresa ou grupo responsável por manter o *framework.*

As aplicações híbridas possuem uma vantagem sobre as *web apps*, elas podem ser executadas *off-line* (estado onde os dispositivos não possuem conexão com a internet) sem necessidade de pré-configuração, assim como as aplicações nativas. Essa vantagem caracteriza-se como uma grande desvantagem da modalidade web app, pois como suas páginas ficam hospedadas em um servidor na internet, se o dispositivo não possuir conexão não terá acesso as web apps. A modalidade nativa também possibilita o acesso off-line à aplicação como poderemos visualizar na próxima subseção.

- **3.3. MODALIDADE NATIVA**

Nessa modalidade, são desenvolvidas aplicações para serem executadas em uma plataforma específica. Elas são escritas na linguagem nativa da plataforma destino, como por exemplo: *Java* para *Android* ou *Objective C* [8]que a linguagem de programação utilizada no desenvolvimento de aplicações para iOS que é o sistema operacional do Iphone[9]. Para esta modalidade também é necessário utilizar o SDK e a API disponibilizada pela organização responsável por desenvolvimento e manutenção do SO, esta modalidade não é multiplataforma, pois cada só necessita de uma implementação diferente.

Para iniciarmos o desenvolvimento de aplicativos móveis é necessário conhecimento na linguagem utilizada pela plataforma (sistema operacional) para qual deseja desenvolver, além da linguagem, deve-se conhecer o SDK e a API da plataforma, pois neles estarão todas as bibliotecas e pacotes utilizados para acessar as funcionalidades, *hardware* e serviços do dispositivo. Este conjunto de conhecimentos específicos torna o desenvolvimento nativo complexo, necessitando de um profissional específico para esta tarefa.

Aplicações nativas costumam possuir um desempenho superior as outras, pelo acesso ilimitado ao hardware [4]. Por exemplo, aplicações nativas podem trabalhar com várias tarefas junto ao processador, estabelecendo comunicação entre elas e distribuindo o processamento. Além disso,

o uso de memória pode ser realizado de forma mais eficiente, podendo controlar a alocação e deslocação da memória, não possuindo limite para uso da memória do dispositivo.

4. ESTUDO DE CASO

Esse estudo de caso visa investigar, na prática, as vantagens e desvantagens de cada modalidade de desenvolvimento de aplicativos para dispositivos móveis. O método utilizado para a realização desse estudo foi baseado na experiência adquirida através do desenvolvimento de um aplicativo móvel. Esse aplicativo será desenvolvido nas três modalidades e, após o desenvolvimento do mesmo, serão utilizadas as experiências obtidas em cada modalidade para realizarmos uma análise comparativa.

Esse estudo de caso será utilizado para validar a proposta do estudo comparativo que tem como objetivo analisar as modalidades de desenvolvimento de aplicativos móveis, sendo elas: Nativa, Híbrida e Web App utilizando somente a plataforma *Android* como base. Esperasse avaliar neste estudo de caso, as modalidades de desenvolvimento móveis, demostrando as características de cada uma e compara-las para encontrar vantagens e desvantagens entre elas.

O contexto desse estudo de caso será o desenvolvimento de um aplicativo móvel, a partir da necessidade de uma associação de carpinteiros, do Canadá, a qual não foi possível utilizar o nome, devido ao tempo necessário para que a equipe jurídica da mesma avaliasse e liberasse o documento para ser assinado pelo presidente da Associação não coincidir com o prazo para submissão deste trabalho a banca avaliadora. Devido a este fato, a mesma será referenciada neste trabalho como associação de carpinteiros.

A associação tem a necessidade de manter o carpinteiro informado sobre a matriz, mas próxima dentro da região que ele esteja no momento. Assim foi elaborada a especificação de um aplicativo móvel, restrito a dispositivos *Android*, contendo as informações para contato as matrizes da associação, possuindo as seguintes funcionalidades especificadas em RFs (Requisitos Funcionais): **RF 01** - A aplicação permitirá ao usuário visitar o site da matriz selecionada, **RF 02** - A aplicação permitirá ao usuário iniciar o aplicativo telefone com o número da matriz já preenchido para efetuar uma ligação, **RF 03** - A aplicação permitirá ao usuário iniciar o aplicativo e-mail com o campo destinatário preenchido com o e-mail da matriz, **RF 04** - A aplicação permitirá ao usuário visualizar no *Google Maps* a localização da matriz, **RF 05** - A aplicação permitirá ao usuário acessar aos álbuns de fotos no *Instagram* da matriz, **RF 06** - A aplicação permitirá ao usuário acessar ao canal de vídeos da matriz no *Youtube*, **RNF 01** - A aplicação será executada exclusivamente no sistema operacional *Android*, **RNF 02** - Os requisitos RF 04, 05, 06 serão exclusivamente utilizáveis quando conectado a internet.

O sistema deve disponibilizar aos carpinteiros, os contatos das matrizes da associação, para que os mesmos possa escolher qual ação executar, seja ligar, enviar e-mail, ver localização, fotos ou vídeos da matriz. O aplicativo proposto por este trabalho possui um total de 29 pontos de função, foram contabilizados pelas funcionalidades e tabelas que a aplicação irá possuir.

5. ANÁLISE DOS RESULTADOS

Após a conclusão do estudo de caso, foi possível identificar o nível de complexidade do projeto em cada modalidade, utilizando a técnica de Thomas McCabe sobre complexidade ciclomática. Para a análise de complexidade foram utilizadas ferramentas apropriadas, para as modalidades

web app e híbrida que possuem a suas regras em *Javascript* foi utilizado a ferramenta online de analise de complexidade ciclomática o *JSComplexity*. Para a análise do código gerado pela modalidade nativa foi utilizada a ferramenta *Gmetrics v0.6* dedicada a analise de complexidade para códigos Java.

Tabela 1 - Nível de complexidade ciclomática por modalidade

	Complexidade (nc)
Web App	29 nc
Híbrida	33 nc
Nativa	45 nc

Através da tabela 1 identificamos que os dados apontam que web app possui o menor nível de complexidade, pois o próprio navegador possui abstrações sobre acesso, mesmo que limitado, a serviços e hardware do dispositivo, por exemplo, na funcionalidade de localização o navegador possui um elemento pronto para obtenção das coordenadas e quando a aplicação necessitava abrir o aplicativo telefone com o número preenchido, foi necessário somente colocar um atributo href a uma tag âncora com o valor tel:<número de telefone>, reduzindo o tamanho do código e possíveis caminhos.

Logo em seguida temos a modalidade híbrida que possui seus códigos *Javascript*, quase idênticos ao da modalidade web app, porém como houve a inclusão de chamadas ao SDK do framework, *PhoneGap*, e o mesmo possui uma linha de execução assíncrona, obrigando assim existir regras como retorno para uma determinada função, seja de sucesso ou erro, assim logicamente aumentando a complexidade para interpretação do código. Porém a variação é pouca, pois continuamos a utilizar as abstrações do navegador e agora também do próprio framework. Por fim podemos identificar que os resultados obtidos através do estudo de caso, indicam que o desenvolvimento na modalidade nativa possui o maior nível de complexidade [10], devido a não possuir nenhuma abstração, de framework ou de navegador, sendo necessário desenvolver toda a regra para acesso ao hardware ou serviços do dispositivo. Outro fator que aumenta a complexidade é a necessidade de uma classe Java para cada tela e o Java por ser linguagem comportamental deve ser levado em consideração na contabilização da complexidade.

Uma das características utilizadas para medir complexidade ciclomática de um software é o seu tamanho e para definir o tamanho do software podem ser utilizadas duas métricas, pontos por função ou LOC. Porém pontos de função contabiliza funcionalidade não levando em consideração a modalidade de desenvolvimento ou a plataforma utilizada, já a LOC contabiliza a quantidade de linhas de código funcional do software, por isso, está será a métrica utilizada para comparar o tamanho do projeto em cada modalidade.

Tabela 2 - Linhas de códigos por modalidade

	LOC
Web App	545 loc
Híbrida	637 loc
Nativa	894loc

Os resultados apresentados na tabela 2 seguem a mesma linha da tabela 1, apontando a aplicação desenvolvida na modalidade *web app* como menor aplicação. Enquanto a modalidade híbrida vem logo em segundo lugar com uma diferença muito pouca para a modalidade web app. A modalidade híbrida esta em segundo lugar, devido a algumas funcionalidades precisar ser escritas utilizando o framework *PhoneGap*, assim aumentando a quantidade de linhas em

relação a web apps. Já a modalidade nativa além dos dados apontarem como a que possui o maior nível de complexidade, também aponta a modalidade nativa como a que mais necessita de linhas de códigos para obtenção do comportamento desejado.

Uma característica muito importante quando tratamos de desenvolvimento de software, é a produtividade alcançada por determinada linguagem, ferramenta ou metodologia. Neste trabalho podemos avaliar a produtividade de desenvolvimento em cada modalidade, baseando-se no total de pontos de função da aplicação e horas gastas por cada modalidade durante o desenvolvimento do aplicativo. Percebemos que a modalidade *Web App*, por usar conhecimentos para desenvolvimento web, como o HTML, CSS, *Javascript* e por também possuir seu nível de complexidade inferior, neste estudo de caso possuiu o tempo de desenvolvimento reduzido quando comparado às outras duas modalidades, como podemos verificar na tabela 3 e assim possuindo uma produtividade mais elevada que as demais modalidades abordadas neste trabalho.

Tabela 3 – Calculando produtividade utilizando ponto de função por horas

	Ponto de Função (PF)	Horas (h)	PF / Horas
Web App	29 PF	40h	0.725 PF/h
Híbrida	29 PF	42h	0.69 PF/h
Nativa	29 PF	168h	0.172 PF/h

A tabela 3 indica a aplicação desenvolvida na modalidade híbrida, como segunda na comparação de produtividade logo após a modalidade *web app*, possuindo uma produtividade de aproximadamente 0.69 pontos de função por horas. A diferença entre a modalidade híbrida e a modalidade *web app* se dar pela necessidade da utilização do framework, fazendo-se necessário recorrer a documentação do mesmo. E em terceiro lugar temos a modalidade nativa, pois a mesma possuiu o maior tempo de desenvolvimento da aplicação. Um dos motivos por ser a modalidade que mais demandou tempo, foi à dificuldade encontrada na utilização do editor de interface gráfica para aplicativos móveis que existe na IDE, *Eclipse*, IDE mais utilizada no desenvolvimento para *Android*, o mesmo é confuso, ineficiente e possui uma forma de alinhamento dos componentes um tanto quanto complicada. Outro fator que impactou diretamente no desenvolvimento móvel nativo foi à necessidade de conhecer a API e SDK do sistema *Android*. Assim está modalidade demandou muitas horas de pesquisa no site oficial do *Google Android Developer*.

Os resultados deste estudo de caso apontam também a curva de aprendizado necessária para a modalidade nativa como a maior comparada as outras duas modalidades. Pois esta modalidade exige dos desenvolvedores conhecimentos específicos da arquitetura do sistema operacional, fazendo assim com que o desenvolvedor seja obrigado a buscar ajuda no guia para desenvolvedores *Android*. A tabela 4 mostra os conhecimentos básicos e iniciais necessários para cada modalidade.

Tabela 4 - Curva de aprendizado por modalidade

	Web App	Híbrida	Nativa
HTML5	•	•	
CSS3	•	•	
Javascript	•	•	
WRD	•	•	
Framework para apps híbridas. (*PhoneGap*)		•	
Arquitetura do Sistema Operacional			•

SDK *Android*			•
API *Android*			•
SDK Java			•
Eclipse + ADT			•

A tabela 4 aponta a modalidade nativa como a que possui a maior curva de aprendizado, não só pela quantidade de conhecimentos, mas também pela extensão de cada, o SDK e a API *Android* são extremamente extensos e ainda exige que os desenvolvedores relembrem os conhecimentos de Java que aprenderam na faculdade. As modalidades *web apps* e híbridas possuem uma curva de aprendizado menor devido à simplicidade do HTML, CSS e até mesmo do *Javascript*, linguagem de script comportamental fracamente tipada e sintaxe muito flexível.

Outra característica que esse trabalho julgou interessante comparar é os recursos de hardware e serviços do dispositivo que cada modalidade proporciona aos desenvolvedores, para serem utilizados nos seus aplicativos. Ao se planejar um aplicativo móvel logo se imagina a utilização dos recursos comuns a estes dispositivos para tornar o aplicativo o mais útil possível para os usuários.
A modalidade nativa não possui limites de acesso ao hardware dos dispositivos, tornando possível desenvolver qualquer tipo de aplicativo e com desempenho sem igual. A modalidade web app é a que mais possui limitação de acesso ao hardware, à mesma só possui acesso a GPS do navegador, Web SQL do navegador, acelerômetro e acesso off-line limitado [10]. Porém a modalidade híbrida possui acesso ao acelerômetro, câmera, contatos, armazenamento interno e externo, GPS, mídia, notificação push, serviços de conexões de rede e acesso off-line. [3].

6. CONCLUSÃO

Este trabalho se propôs a realizar uma analise comparativa entre as modalidades de desenvolvimento móvel, *web app*, híbrida e nativa. Para tornar possível a comparação e validá-la, foi desenvolvido um estudo de caso sobre o desenvolvimento de um projeto móvel nas três modalidades de desenvolvimento de aplicativos móveis, com a finalidade de possibilitar a realização de uma análise comparativa entre as modalidades *web apps*, híbrida e nativa. Para viabilizar a elaboração do estudo de caso, este trabalho trouxe a definição e principais características das três modalidades.

Após a elaboração da aplicação nas três modalidades, identificou-se que, de fato, a modalidade w*eb app*, possuiu um menor tempo de execução do projeto e o produto gerado nessa modalidade é compatível com qualquer plataforma e qualquer resolução. Já o projeto desenvolvido na modalidade híbrida possuiu um tempo maior que a w*eb app* e o conhecimento a mais do SDK do *framework* escolhido, nesse trabalho o *PhoneGap*. Porém a aplicação gerada possui acesso ao *hardware*, diferentemente da modalidade *web app*, possui acesso off-line por padrão, já que as páginas HTML do aplicativo estão armazenadas localmente e seu desempenho não é afetado pela velocidade da conexão com a internet. A mesma ainda possui a vantagem de ser multiplataforma assim como as *web apps*, sendo necessário gerar um projeto para cada plataforma, mas utilizando os mesmos fontes HTML, CSS e *Javascript* para todos eles.

Já a modalidade nativa mostrou que não é multiplataforma [4], necessita de mão de obra especializada, ambiente de desenvolvimento específico e complexidade de desenvolvimento maior que as outras duas modalidades [6]. Por ter um nível de complexidade maior também

necessita de um tempo maior para a elaboração da aplicação nessa modalidade. Contudo a modalidade nativa cria aplicações com desempenho superior às outras modalidades, possui acesso ilimitado ao *hardware* e serviços do *SO*, possibilidade de trabalhar com bibliotecas gráficas duas dimensões e três dimensões, mesmo que não tenha sido utilizado nesse estudo de caso.

Nesta comparação foi identificada que a modalidade híbrida é a mais vantajosa quando se trata de aplicativos que não necessitam de alto processamento de dados ou de processamentos gráficos avançados, como por exemplo, os aplicativos da categoria de jogos que normalmente utiliza a modalidade nativa. As aplicações desenvolvidas pela modalidade híbrida possuem acesso ao hardware dos dispositivos e serviços do sistema operacional. Aplicações híbridas são multiplataforma, reduzindo assim o custo do projeto e o tempo, pois uma única implementação servirá para várias plataformas, este fator agrega muito valor a esta modalidade como também a web app, pois a aplicação consegue atingir um numero maior de usuário por atender a vários mercados, como por exemplo, o mercado iOS, Android, Windows Phone, Blackberry.

Como possíveis trabalhos futuros, pode-se apontar a realização desta mesma análise comparativa e estudo de caso, com a participação de diversos desenvolvedores, para aumentar a quantidade de resultados obtidos e assim encontrar uma estatística mais precisa na comparação das três modalidades. Pode-se apontar também, uma análise comparativa de custo para cada modalidade de desenvolvimento de aplicações móveis, também uma análise comparativa entre a modalidade de desenvolvimento móvel nativa para *Android* e *iOS*, esclarecendo os pré-requisitos, a curva de aprendizado e as vantagens e desvantagens encontradas em cada uma das plataformas. Elaborar uma pesquisa exploratória sobre o e-commerce mobile, definição, empresas que já utilizam e números sobre.

-
-
- **REFERÊNCIAS**

[1]SDK. Disponível em: <http://pt.wikipedia.org/wiki/Software_Development_Kit>. Acesso em 29 novembro 2014.

[2] API. Disponível em: <http://pt.wikipedia.org/wiki/API>. Acesso em 29 novembro 2014.

[3]PhoneGap. Disponível em: <http://docs.phonegap.com/en/2.9.0/index.html>. Acesso em: 11 maio 2014.

[4]HTML. Disponível em: HTML em:<http://tableless.com.br/o-que-html-basico/>. Acesso em 29 novembro 2014.

[5]CSS. Disponível em: <http://www.w3schools.com/css/>. Acesso em 29 novembro 2014.

[6]JS. Disponível em: <htp://www.w3schools.com/js/> Acesso em 28 novembro 2014.

[7]Miszura, J. T. e Divino, G. Desenvolvimento em Smartphones: Aplicativos Nativos e Web, Universidade Católica de Goiás, Goiás, Brasil, **2013**.

[8]Frain, B. Responsive Web Design with HTML5 and CSS3: Learn responsive design usin HTML5 and CSS3 to adapt websites to any browser or screen size. 1 ed. Editora PACKT, Massachusetts, EUA, **2012**.

[8]Object-C. Disponível em: <http://pt.wikipedia.org/wiki/Objective-C>. Acesso em 28 novembro 2014.

[9]iOS Developer. Disponível em: <https://developer.apple.com/devcenter/ios/index.action>. Acesso em: 11 maio 2014.

[10]Taurion, Cezar / IBM developerWorks. Disponível em: <https://www.ibm.com/developerworks/community/blogs/ctaurion/entry/apps_nativos_versus_html5?lang=en>. Acesso em: 11 maio 2014.2014.