FACULDADE PITÁGORAS

CURSO CIÊNCIA DA COMPUTAÇÃO

**IGOR THADEU VELOZO BRITO**

**DESVENDANDO APLICATIVOS HÍBRIDOS: DESENVOLVENDO UM APLICATIVO MUILTI-PLATAFORMA UTILIZANDO APPGYVER**

SÃO LUÍS

2014

**IGOR THADEU VELOZO BRITO**

**DESVENDANDO APLICATIVOS HÍBRIDOS: DESENVOLVENDO UM APLICATIVO MUILTI-PLATAFORMA UTILIZANDO APPGYVER**

Monografia apresentada ao Curso de Ciência da Computação da Faculdade Pitágoras, para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Me. Thiago Nelson Faria do Reis

SÃO LUÍS

2014

**IGOR THADEU VELOZO BRITO**

**DESEVENDANDO APLICATIVOS HÍBRIDOS: DESENVOLVENDO UM APLICATIVO MUILTIPLATAFORMA**

Monografia apresentada ao Curso de Ciência da Computação da Faculdade Pitágoras, para obtenção do grau de Bacharel em Ciência da Computação.

Aprovado em: / /

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Me. Thiago Nelson Faria do Reis (orientador)

Faculdade Pitágoras

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1º EXAMINADOR

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2º EXAMINADOR

AGRADECIMENTOS

Primeiramente a Deus que permitiu que tudo isso acontecesse ao longo de minha vida, e não somente nestes anos como universitário, mas que em todos os momentos é o maior mestre que alguém pode conhecer. A esta universidade, seu corpo docente, direção e administração que oportunizaram a janela que hoje vislumbro um horizonte superior, eivando pela acendrada confiança no mérito e ética aqui presentes. Ao professor Thiago Nelson, pela orientação, apoio e confiança. Aos meus pais, irmão e noiva, pelo amor, incentivo e apoio incondicional. A todos que direta ou indiretamente fizeram parte da minha formação, o meu muito obrigado.

*“Para se ter sucesso, é necessário amar de verdade o que se faz. Caso contrário, levando em conta apenas o lado racional, você simplesmente desiste.*

*É o que acontece com a maioria das pessoas. “*

Steve Jobs

RESUMO

O presente trabalho teve como objetivo pesquisar, analisar e refletir acerca do desenvolvimento de aplicativos híbridos para dispositivos mobile. O estudo aborda a os valores importantes a serem considerados para o desenvolvimento de aplicativos mobile, levando em consideração a atual maneira desenvolvimento nativo que acaba consumindo muita mão-de-obra para obter um produto final que pode ser construído muito mais rápido e barato utilizando tecnologias de desenvolvimento híbrido. O objetivo desse trabalho não é afirmar que o desenvolvimento híbrido é melhor que o desenvolvimento nativo ou vise e versa.

Alguns paradigmas que estão sendo muito utilizados em novos frameworks e em metodologias de desenvolvimento com o DIY (Don’t Repeat Yourself), são defendidos fortemente nesse trabalho e será mostrado que é possível construir um aplicativo mobile com tecnologias web como HTML5, CSS3 e Javascript reaproveitando 100% do código e da lógica desenvolvida para no final termos um aplicativo multplataforma utilizando o AppGyver e suas dependências.

Palavras-Chave: aplicativos híbridos; desenvolvimento nativo; desenvolvimento híbrido

ABSTRACT

O presente trabalho teve como objetivo pesquisar, analisar e refletir acerca do desenvolvimento de aplicativos híbridos para dispositivos mobile. O estudo aborda a os valores importantes a serem considerados para o desenvolvimento de aplicativos mobile, levando em consideração a atual maneira desenvolvimento nativo que acaba consumindo muita mão-de-obra para obter um produto final que pode ser construído muito mais rápido e barato utilizando tecnologias de desenvolvimento híbrido. O objetivo desse trabalho não é afirmar que o desenvolvimento híbrido é melhor que o desenvolvimento nativo ou vise e versa.

Alguns paradigmas que estão sendo muito utilizados em novos frameworks e em metodologias de desenvolvimento com o DIY (Don’t Repeat Yourself), são defendidos fortemente nesse trabalho e será mostrado que é possível construir um aplicativo mobile com tecnologias web como HTML5, CSS3 e Javascript reaproveitando 100% do código e da lógica desenvolvida para no final termos um aplicativo multplataforma utilizando o AppGyver e suas dependências.

Palavras-Chave: aplicativos híbridos; desenvolvimento nativo; desenvolvimento híbrido

LISTA DE FIGURAS

Figura 01 – Camadas básica da internet1

Figura 02 – Exemplo básico de um documento HTML51

Figura 03 – Exemplo de código em CSS31

Figura 04 – Exemplo de código em Javascript1

Figura 05 – Telas do aplicativo Bíblia1

Figura 06 – Cenário de desenvolvimento nativo1

Figura 07 – Processo de compilação de um aplicativo híbrido1

Figura 08 – Estrutura de diretórios do AppGyver1

Figura 09 – Arquivo app/views/layouts/book.js 1

Figura 10 – Estrutura do diretório view1

Figura 11 – Arquivo app/controllers/book.js 1

Figura 12 – Arquivo app/views/book/show.html1

Figura 13 – Arquivo de configuração de preferencias para IOS localizado em www/config.ios.xml 1

LISTA DE SIGLAS

ARPANET – Advanced Research Projects Agency Network

DARPA – Agência de Projetos de Pesquisa Avançada dos Estados Unidos

SO – Sistema Operacional

MTA – Mobilie Telephony A

IDC – International Data Corporation

HTML – HyperText Transfer Protocol

IOS – Iphone Operating System

CSS – Cascading Style Sheets

SASS – Syntactically Awesome Style Sheets

MVVM – Model View View Model

RAD – Rapid Application Development

SDK – Software Development Kit

MVC – Model View Controller

SUMÁRIO

1 INTRODUÇÃO1

2 METODOLOGIA E FERRAMENTAS UTILIZADAS4

2.1 Plataformas Mobile5

2.1.1 IOS6

2.1.2 Android6

2.1.3 Windows Mobile6

2.1.4 BlackBerry6

2.2 Tecnologias Web5

2.2.1 HTML56

2.2.2 CSS36

2.2.3 Javascript6

2.3 Aplicativos Híbridos Mobile5

2.3.1 Frameworks Javascript6

2.3.2 AppGyver6

2.3.3 Angular JS6

**3 ESTUDO DE CASO4**

3.1 Escreva um e rode em qualquer lugar5

3.2 Arquitetura5

3.2.1 View6

3.2.2 Controller6

3.2.3 Model6

3.3 Configuração do Projeto5

3.4 Interface de Usuário5

3.5 Distribuição5

**4 CONCLUSÃO4**

**REFERÊNCIAS4**

**1 INTRODUÇÃO**

Em 1947 no laboratório de Bell nos Estados Unidos começou o desenvolvimento do primeiro sistema telefônico de alta capacidade interligado por várias antenas, onde era chamada cada antena de célula, no que acabou servindo de inspiração para o nome “celular”. Em 1956 a Ericsson desenvolveu o primeiro aparelho celular, ele se chamava MTA e pesava 40 quilos. Apenas 17 anos depois, em 1973 a Motorola apresentou seu primeiro modelo de telefone celular, o Dynatac 8000X que bem diferente dos projetos antecessores tinha 25cm de comprimento e 7cm de largura, além de pesar aproximadamente 1 quilo. Esse modelo foi utilizado por Martin Cooper, diretor de sistemas de operações da Motorola, para realizar a primeira ligação de um aparelho celular da história. Os primeiros países em que o uso comercial foi liberado foram o Japão e a Suécia em 1979 e só chegou nos Estado Unidos em 1983.

Desde o seu lançamento, os celulares vem passando por várias mudanças constantes inclusive a redução dos preços, o que ocasionou uma popularização em massa desses aparelhos. Também a concorrência entre os fabricantes acabaram trazendo novidades para chamar a atenção do consumidor. O que no início era a minimização desses aparelhos bem sucedidos no mercado, hoje o consumidor está bastante exigente com relação a performance e funcionalidades oferecidas.

Entre os principais fabricantes as que mais se destacam são Apple, Samsung, BlackBerry, Motorola e Nokia. O grande fator diferenciador entre essas cinco fabricantes é o SO (Sistema Operacional). Diferente da Samsung e da Motorola, todas utilizam SO que rodam até o momento exclusivamente em seus aparelhos e isso acaba estendendo a particularidade de casa plataforma para os aparelhos.

Os SO que também são conhecidos como plataformas, que mais são utilizados no mercado são o IOS, Android, Windows Phone e BlackBerry, segundo o IDC na pesquisa Smartphone OS Market Place, Q2 2014. O desenvolvimento para cada plataforma exige ambientes diferentes, isso porque suas características de programação, principalmente a linguagem são diferentes. Isso deixa praticamente impossível códigos que foram feitos para uma plataforma, serem reutilizados para outra.

Com o avanço da popularização do celular, e ampla cobertura de conexão a internet por meio desses dispositivos, tem aquecido o setor de desenvolvimento de aplicativos e isso tem chamado bastante atenção de empresas, investidores e estudantes. Os aplicativos para mobile se tornaram essenciais e muita das vezes mais utilizados do que a principal funcionalidade que deu origem ao celular, a chamada de voz. Esses aplicativos originalmente eram desenvolvidos com tecnologias nativa de cada plataforma, e com a possibilidade de desenvolver um aplicativo utilizando tecnologias web (HTML5, CSS3 e Javascript), e ele ser compatível com as outras plataformas tem se tornado bastante interessante.

Este trabalho tem como objetivo desvendar o desenvolvimento de aplicativos híbridos utilizando o AppGyver e serão apresentados os pontos positivos e negativos utilizando essa técnica de desenvolvimento. Um produto será desenvolvido com o conteúdo deste trabalho.

**2 METODOLOGIA E FERRAMENTAS UTILIZADAS**

Nesse capítulo iremos fundamentar toda a teoria utilizada para o desenvolvimento desse projeto.

**2.1 Plataformas Mobile**

Desde o início da era dos smartphones as vendas surpreendem a cada ano. O número de smartphones cresceu 25,3% no segundo trimestre de 2014, estabelecendo um novo recorde 301.3 Milhões de carregamentos segundo os dados do IDC, na pesquisa Worldwide Quaterly Mobile Phone Tracker. Com esse crescimento surpreendente de smartphones no mundo, as diferentes plataformas que fazem parte essencialmente desses dispositivos móveis, veem crescendo cada dia mais, e com seus grandes diferenciais elas conquistam e fidelizam seus usuários.

As plataformas são fundamentais para o funcionamento de um smartphone, elas são responsáveis por gerenciar aplicativos e componentes dos dispositivos de maneira que eles se comuniquem em uma linguagem binária. Isso significa que as plataformas conseguem gerenciar recursos como GPS, Câmera, sensores e muitos outros.

Não somente os smartphones não capazes de utilizar essas plataformas. Desde da chegada dos tabletes no mercado, a sua aceitação tem sido bastante entre os usuário pelo o fato de herdar a maioria das funcionalidades dos smartphones e contar com uma tela maior para interação, além de utilizarem as mesma plataformas dos smartphones tornando eles mais familiar aos usuários que vinham de um outro dispositivo.

Segundo o IDC na pesquisa Smartphone OS Market Place, Q2 2014, as plataformas que possuem maior parcela do mercado global são Android, IOS, Windows Phone e BlackBerry. Disparado na frente por conta do baixo custo de seus dispositivos, o Android aparece com 84,7% e se torna a plataforma mais utilizada no mundo como podemos ver na tabela abaixo:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Q2 2014** | **Q2 2013** | **Q2 2012** | **Q2 2011** |
| **Android** | 84.7% | 79.6% | 69.3% | 36.1% |
| **IOS** | 11.7% | 13.0% | 16.6% | 18.3% |
| **Windows Phone** | 2.5% | 3.4% | 3.1% | 1.2% |
| **BlackBerry** | 0.5% | 2.8% | 4.9% | 13.6% |
| **Outros** | 0.7% | 1.2% | 6.1% | 30.8% |

Em 2011 a parcela de outras plataformas eram de 30.8%, mas nos anos seguintes esse número foi diminuindo e migrando para o Android. Essas plataformas não conseguiram se firmar no mercado por conta da alta concorrência.

O perfil de preferência por uma plataforma muda de acordo com o tipo de pessoa que está utilizando. O usuário final geralmente prefere uma plataforma mais simples de se usar e com um custo menor mas ao mesmo tempo sem perder as mais conhecidas funcionalidades. Por outro lado os desenvolvedores buscam mais por rendimentos, e o que pode até parecer estranho mas não é, é que a plataforma mais utilizada é de longe a que retorna menos rendimento para seus desenvolvedores.

|  |  |  |  |
| --- | --- | --- | --- |
|  | IOS | Android | Windows Phone |
| Número de Apps | 1,2 Milhão | 1,3 Milhão | 300 Mil |
| Média de download por Apps | 40 Mil | 60 Mil | 4 Mil |
| Receita Média por download | U$$ 0,10 | U$$ 0,018 | U$$ 0,10 |
| Receita média por Apps | U$$ 21.276 | U$$ 1.125 | U$$ 2.222 |

O perfil de usuários do IOS é de um público mais ativo nas app stores, até mesmo pelo alto preço pago pelos dispositivos que trazem essa plataforma. Dessa forma os desenvolvedores acabam investindo mais tempo para desenvolver aplicativos para essa plataforma já que o retorno é quase sempre garantido.

2.1.1 IOS

Essa plataforma foi lançada pela Apple em 2007 na Macworld Conference & Expo e hoje ela pertence a todos os dispositivos mobile da companhia. No início não eram permitidas aplicações de terceiros e só em outubro do mesmo ano do lançamento, foi que a Apple anunciou o desenvolvimento da SDK para os futuros desenvolvedores.

O Iphone foi apresentado e grande parte do seu sucesso foi devido ao seu sistema operacional que oferecia novas funcionalidades e uma boa aparência para época. Isso desencadeou a venda desses aparelhos e a Apple chegou a vender mais de 1 milhão de Iphones em uma temporada de feriados de 2007 e logo em seguida despertou o interesse dos desenvolvedores por busca da SDK. A Apple inovou no conceito de App Store(loja de aplicativos), o que movimentou cada vez mais desenvolvedores e usuários querendo consumir esse tipo de produto. Com a chegada de Ipods e Ipads o mercado de dispositivos mobile da Apple expandiu cada vez mais e consequentemente o seu sistema operacional cresceu junto.

O IOS se tornou tão importante para o mercado que além de agregar qualidade em seus produtos, também beneficiou a máquina motora de criação de novos aplicativos, os desenvolvedores. O retorno financeiro para essas pessoas foram e ainda são considerados bons, tanto que estatisticamente a pesquisa pela IDC aponta que a receita média por aplicativos é de U$$ 21.276 quase dez vezes mais que o segundo colocado, o Android. O IOS se tornou tão bem conceituado no mercado que é muito difícil de não encontrar aplicativos bem sucedidos com disponibilidade na plataforma.

|  |  |
| --- | --- |
| Versão | Ano |
| Iphone OS 1 | 2008 |
| Iphone OS 2 | 2008 |
| Iphone OS 3 | 2009 |
| IOS 4 | 2010 |
| IOS 5 | 2011 |
| IOS 6 | 2012 |
| IOS 7 | 2013 |
| IOS 8 | 2014 |

Infelizmente para a Apple é que o mercado está mudando e a conectividade entres as pessoas está ainda maior, isso significa que elas querem trocar cada vez mais informações e o IOS é visto por muitos usuários de outras plataformas como um sistema “trancado“ onde não é possível compartilhar muitos tipos dados entre dispositivos.

2.1.2 Android

Em 2005, a Google comprou a empresa Android Inc. que era uma empresa que desejava criar uma plataforma mobile baseada em Linux e que fosse bem flexível. Em 2007 foi lançado o sistema operacional Android, que se popularizou graças a facilidade que as empresas de telefones móveis tinham para utilizar essa tecnologia e isso tornou o Android segundo IDC, o sistema operacional mais utilizado no mundo. Hoje o Android não se encontra somente em smartphones, ele também pode ser encontrado em tabletes, relógios, televisores e outros dispositivos.

O Android entrou no mercado para ser comercializado em 22 de outubro de 2008, sendo utilizado pelo aparelho HTC Magic ou G1. Desde o momento do seu lançamento comercial, essa plataforma vem crescendo todo o ano no número de usuários. Hoje grandes companhias como Samsung e Motorola utilizam o Android como seu sistema operacional pela grande aceitação, estabilidade e custos.

A estabilidade do Android é dada por conta do seu kernel ser o mesmo do Linux, por esse motivo ele se torna bastante completo e estável possuindo vários drives, gerenciamento de processos, camada de segurança e gerenciamento de memória. Através do OpenGL ES, o Android é capaz de reproduzir excelentes gráficos 2D e 3D.

2.1.3 Windows Mobile

Em 15 de Fevereiro a Microsoft anunciou oficialmente o lançamento de sua nova plataforma mobile na Mobile World Congress 2010 na cidade de Barcelona na Espanha. O seu SDK foi disponibilizado em 16 de setembro de 2010. O sistema operacional ganhou muita força a partir do momento que os presidentes executivos da Microsoft, Steve Balmer, e da Nokia, Stephen Elop anunciaram a parceria entre as duas empresas, entusiasmados por uma futura competição com o IOS e o Android. Vários outros serviços da Microsft e da Nokia foram incorporado ao sistema operacional como o Bing que iria fazer as pesquisas nos dispositivos da Nokia, a integração da loja de aplicativos da Nokia com a da Microsoft e a junção do Ovi Maps e Bing Maps. Os primeiros dispositivos lançados pela Nokia utilizando o Windows Phone foram o Nokia Lumia 800 e o Nokia Lumia 710.

A mudança da Microsoft com seu design chamado de ‘metro’ alavancou a venda de dispositivos que utilizavam o Windows Mobile. A interface por ser muito fácil de usar foi o principal fator de aceitação dos usuários e tambem uma mudança nos designs da web e gráficos, surgindo um novo estilo chamado de ‘flat‘ e também comumente conhecido como ‘chapado‘. A tela inicial é viva, com mosaicos que se animam e mudam de acordo com a configuração do usuário onde é possível enxergar logo de primeira vista tudo o que está acontecendo no seu dispositivos incluindo inclusive status em redes sociais.

2.1.4 BlackBerry

Pertencente a RIM (Research in Motion), o sistema operacional BlackBerry foi uma das primeiras plataformas mobile e também uma das responsáveis pelo surgimento dos smartphones, estando sempre inovando e criando novas tecnologias inexistentes nas outras plataformas. Umas das suas características mais forte é o seu teclado QWERT que tem como o foco facilitar a utilização dos serviços de email, internet e mensagens de texto. Com isso a BlackBerry conquistou o mercado corporativo, se tornando o dispositivo móvel preferido entre as grandes companhias e seus executivos.

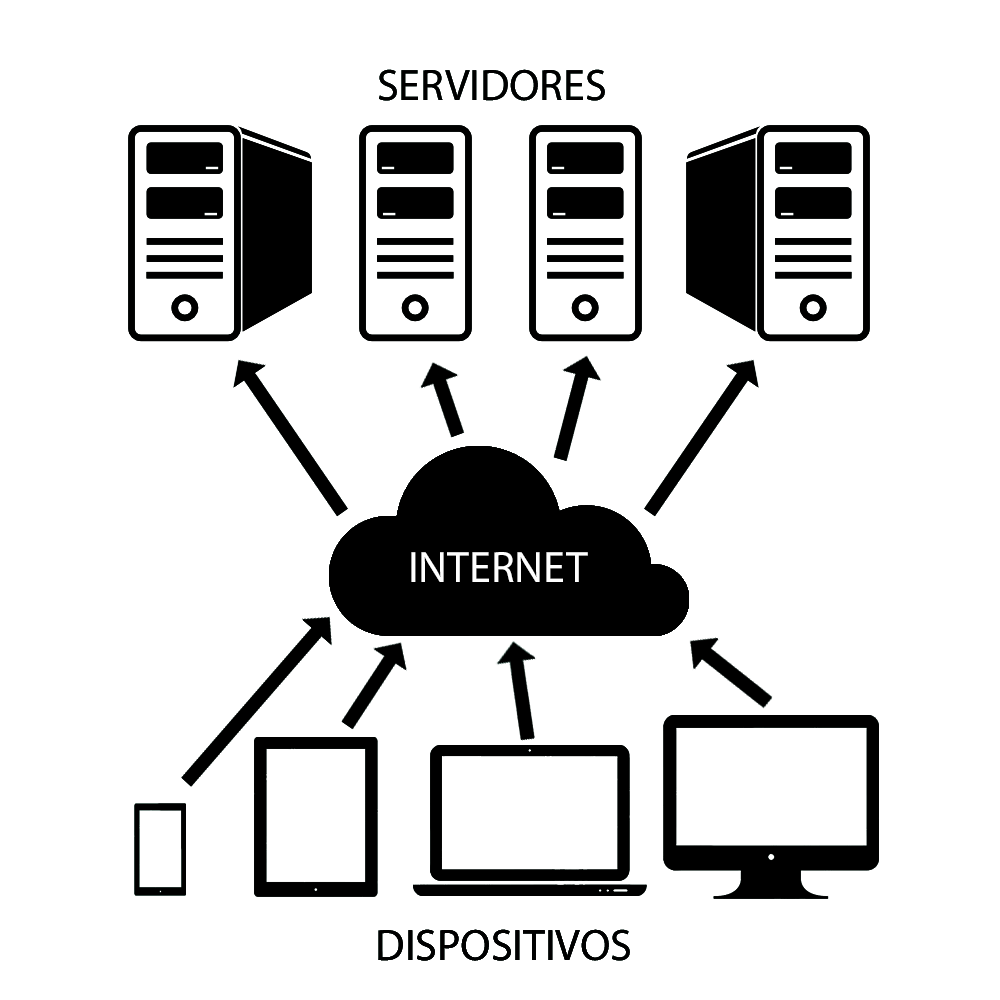
Uma característica interessante da RIM que se assemelha com a Apple, é que a própria RIM é responsável por desenvolver seus próprios aparelhos, sistema operacional e aplicativos que já vêm instalados na plataforma. Com isso é possível o sistema operacional consumir ainda mais e em alto nível seu hardware já que um foi desenvolvido em função do outro, o que em outras plataformas é um ponto fraco porque um sistema operacional como o Android deve se comportar em plataformas que são desconhecidas e ainda sim tentar manter uma boa performance. Mas mesmo assim a plataforma vem perdendo espaço no mercado para seus concorrentes, então a RIM resolveu comprar o sistema operacional QNX, e já está trabalhando em uma nova geração chamada de BBX.

**2.2 Tecnologias Web**

A internet se popularizou muito devido a velocidade no acesso de informações e para que isso ocorresse bem a facilidade de acessa-la por vários dispositivos era fundamental. Hoje se pode conectar a internet por uma gama de dispositivo desde de computadores, smartphones, carros, televisores e até mesmo eletrodomésticos comuns como geladeira. Para isso a acontecer o sistema operacional deve oferecer uma ferramenta para interpretar esse tipo de comunicação, e os mais conhecidos são os browsers (navegadores). Através deles e de sua versatilidade o conteúdo da web pode ser acessado por qualquer dispositivo que seu sistema operacional tenha suporte a um browser.

Por trás de toda essa facilidade existe uma infraestrutura que a nível de uma visão macro não parece ser tão complexa, mas que na verdade é. Composta basicamente por hardwares, protocolos e linguagens de programação, a rede mundial de computadores pode ser representada como uma camada que se encontra entre os clientes utilizando algum dispositivo e os servidores. Ela é responsável por fazer o intermédio entre os dois, permitindo o trafego de informações como representa a figura abaixo:

**Figura 01.** Camadas básica da internet



**Fonte:** BRITO, 2014.

Os dispositivos estão representando a camada do usuário final que através de algum dispositivo conectado a internet e na maioria das vezes utilizando um browser é capaz de fazer requisições HTTP (HyperText Transfer Protocol). A infraestrutura que compõe a internet vai fazer o redirecionamento para qual servidor é responsável por server uma resposta para a requisição feita.

Para se criar aplicações web são utilizadas várias linguagens de programação, variando do gosto do desenvolvedor e da necessidade a ser cumprida. Basicamente podemos dividir em duas categorias, server-side e client-side.

Server-side como o nome traduzido para o português já diz, é responsável pelo o lado do servidor. Essas linguagens geralmente não são interpretadas diretamente pelos navegadores, o principal foco dessa categoria é receber uma requisição, processa-la e depois responder um formato interpretável pelo navegador.

A client-side já é responsável por enviar requisições e também organizar a resposta obtida por uma requisição feita. Essas linguagens trabalhadas no lado do cliente são interpretadas pelo o navegador e são responsáveis por facilitar a usabilidade por parte do usuário final através da UI (User Interface).

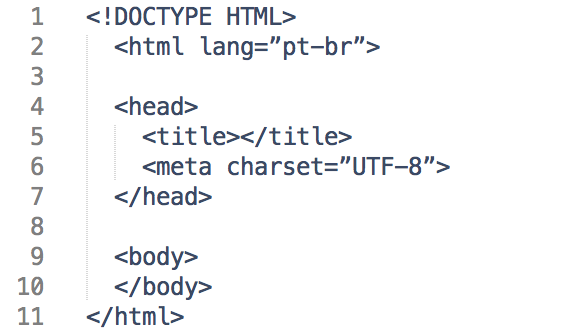
A UI é fundamental para a interação do usuário com a aplicação, é por meio desta que são fornecidos gráficos, formulários, textos, vídeos e elementos manipuláveis. Toda a responsabilidade de exibir isso é do browser, que basicamente recebe códigos HTML para estruturar, CSS para estilizar e as vezes Javascript para melhor interação que leva ao conceito de UX (User Experience), onde se mede a experiência do usuário em usar a aplicação.

2.2.1 HTML5

HTML é uma linguagem de marcação e que está presente em toda pagina na internet. Ela foi criada em 1990 com a ideia de possibilitar o acesso e troca de informações entre universidades. Essas páginas podem ser acessadas por qualquer dispositivo que tenha um navegador que interprete essa linguagem. Por ser relativamente básico, essa linguagem tem curva de aprendizado curta e é bem acessível para as pessoas que estão começando a programar para a web. Os padrões dessa linguagem são regidos pela W3C.

AS páginas em HTML são compostas por tags (etiquetas), responsáveis por separar e estruturar os conteúdos. Essas tags podem ser reconhecidas por estarem entre os parênteses angulares (<tag>). Por permitirem a organização de conteúdo na página, cada tag tem uma funcionalidade específica para cada situação e elas não aparecem nas páginas. Podemos ver um trecho de uma estrutura básica de uma página HTML:

**Figura 02.** Exemplo básico de um documento HTML5

****

**Fonte:** BRITO, 2014.

* **<!DOCTYPE HTML> :** Informa qual a versão do HTML que está sendo utilizado pela página.
* **<html lang=”pt-br”>...</html> :** É a abertura padrão de todo conteúdo em HTML. Todo código referente a isso deve vir entre essa tag.
* **<head>...</head> :** Inicia a parte do cabeçalho do documento. Tudo entre essa tag é usado principalmente para ajudar os motores de busca do navegador e classificar sua página.
* **<title>...</title> :** O conteúdo dentro dessa tag é exibido como título da página para o navegador e motores de busca.
* **<meta>...</meta> :** Esse tipo de tag da página informações especificamente para os motores de busca.
* **<body>...</body> :** Tudo visível para o usuário vai dentro dessa tag, é o corpo da página.

Existem dezenas de tags do HTML e hoje com algumas bibliotecas em Javascript e outras linguagens é possível estender esse numero, criando tag próprias e customizadas pelo o desenvolvedor. Essas tag são interpretadas pela própria biblioteca que se responsabiliza em informar ao navegador o que está acontecendo naquele trecho de código.

2.2.2 CSS3

CSS (Cascading Style Sheets) é uma linguagem de folha de estilo utilizada ara descrever a aparência de um documento escrito em uma linguagem de marcação, como por exemplo (HTML). Na maioria das vezes a utilização do CSS é para estilizar páginas em HTML, mas ele também pode ser utilizado para documentos em XML e outras linguagens de marcação. Com o HTML e Javascript, CSS é uma tecnologia que se encontra na maioria dos sites na internet sendo a maneira mais rápida e leve de se estilizar uma página.

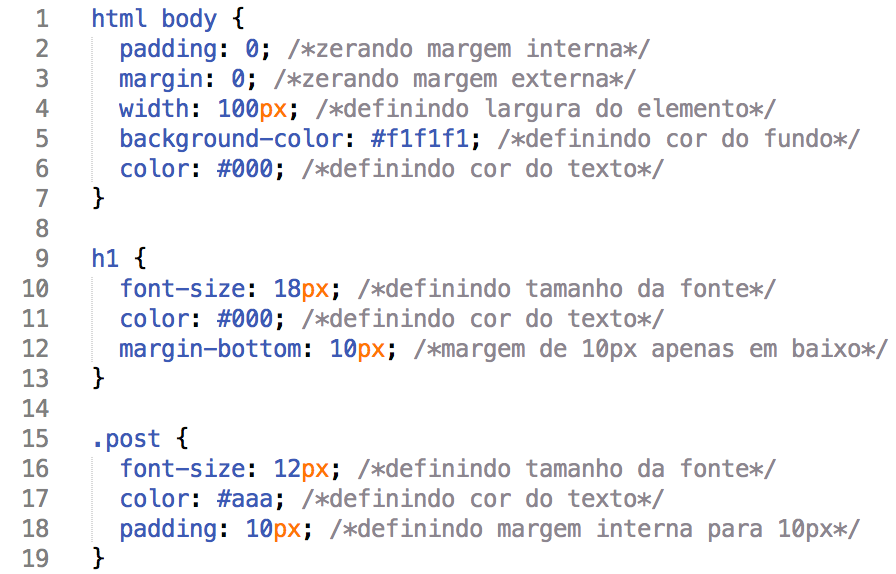
Um dos principais destaques do CSS é que ele foi projetado para separar o conteúdo do documento de sua estilização, podendo assim proporcionar mais flexibilidade em exibir o mesmo conteúdo com estilos diferentes para facilitar por exemplo a acessibilidade e também tornar possível compartilhar a mesma folha de estilo com outros documentos sem ter que estilizar um documento por vez, tornando a folha de estilo global para todos os arquivos que a importam. Uma grande vantagem disso é que se o layout sofrer uma alteração simples como uma mudança de cor, o desenvolvedor poderá modificar apenas um arquivo (folha de estilos) e surtirá efeito em todos os documentos que estiverem importando a mesma.

O CSS possui seletores que estão associados às tags do HTML. Os seletores são usados ​​para declarar que parte da marcação receberá o estilo que esta sendo declarado por meio de tags correspondentes e atributos na própria marcação. Seletores podem ser aplicadas para:

* Todos os elementos de um tipo específico, por exemplo, todo conteúdo que estiver dentro da tag <h1>.
* Elementos especificados pelo atributo , em particular:
  + ID : um identificador único para o documento, identificado no CSS pelo o uso de “#“.
  + Classe : um identificador de vários elementos em um documento que estejam utilizando esse atributo identificado no CSS como “.“.
  + Elementos, dependendo de como eles são colocados em relação a outros na árvore do documento .

Exemplo de um trecho de código CSS3 abaixo:

**Figura 03.** Exemplo de código em CSS3



**Fonte:** BRITO, 2014.

O CSS3 é a mais nova versão da linguagem e trazem um poder maior de baixo e alto nível, principalmente quando integrada ao HTML5. As principais mudanças que aconteceram foram a introdução de módulos e novas funcionalidades para os seletores.

2.2.3 Javascript

O que muitos pensam é que há relação entre a linguagem Java e o Javascript, mas elas são infinitamente diferentes e só se assemelhar por serem linguagens de programação. Javascript foi desenvolvido originalmente por Brendan Eich , enquanto trabalhava para a Netscape Communications Corporation.

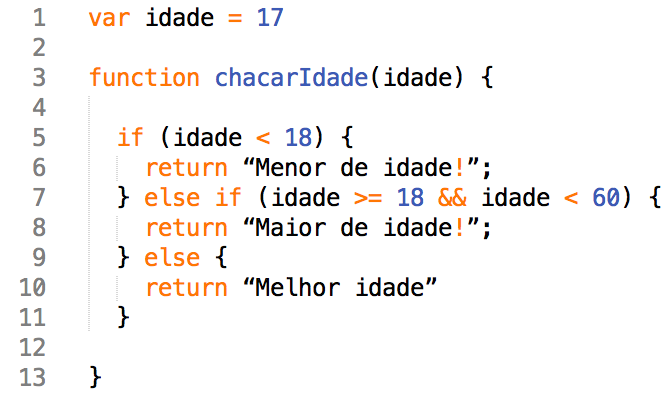
Javascript é uma linguagem de programação dinâmica e é bastante utilizada pelos os navegadores onde permitem interpretação de scripts para interação com o usuário final podendo ser executado de forma assíncrona, controlar o navegador, alterar o conteúdo apresentado sem necessidade de recarregar a página, e muitas outras funcionalidades.

Durante muito tempo o Javascript vem sendo utilizado para criar efeitos, alertas e um pouco mais de dinamismo para as páginas, mas isso é muito pouco para o poder real dessa ferramenta. O Github, a maior rede de desenvolvedores do mundo, é monitorada pelo o site http://githut.info/ que é responsável por colher estatísticas do Github. No final do ano de 2014 o site apontou que a linguagem que mais tem repositórios no Github é o Javascript.

When Java™ applets failed, JavaScript became the “Language of the Web” by default. JavaScript’s popularity is almost completely independent of its qualities as a programming language. (CROCKFORD, 2008, p. 1)

Exemplo de um trecho de código em Javascript:

**Figura 04.** Exemplo de código em Javascript



**Fonte:** BRITO, 2014.

O código acima faz uma checagem de uma idade que é passada por parâmetro para a função checarIdade() e retorna um texto diferente para cada situação.

O Javascript que era utilizado para front-end (quem desenvolve client-side ou seja, para o lado do cliente) agora vem sendo bastante utilizada para backend (server-side, para o lado do servidor). De acordo com HUGHES-CROUCHER, WILSON (2012, p. 34) Por trabalhar muito bem com threads e utilizar javascript, o Nodejs vem crescendo bastante no cenário de desenvolvimento.

Many people use the JavaScript programming language extensively for programming the interfaces of websites. Node.js allows this popular programming language to be applied in many more contexts, in particular on web servers. There are several notable features about Node.js that make it worthy of interest. (HUGHES-CROUCHER, WILSON, 2012, p. 3)

**2.3 Tecnologias para Aplicativos Híbridos Mobile**

Na sessão anterior foi visto sobre as tecnologias web e como elas são muito utilizadas para seu fim principal. Será apresentado agora sobre as tecnologias que serão utilizadas para o desenvolvimento de aplicativos híbridos mobile, que junto com as tecnologias web serão capazes que no final entregar um aplicativo multiplataforma.

2.3.1 Frameworks Javascript

Frameworks são bibliotecas que o principal objetivo é auxiliar no desenvolvimento, abstraindo a complexidade que seria desenvolver sem ele, aumentando a velocidade e a organização do projeto.

No mercado de hoje já existem vários frameworks que podem auxiliar no desenvolvimento desses aplicativos, justamente pelo o crescimento do numero de empresas e desenvolvedores utilizando esses serviços. A maioria dos frameworks são gratuitos para uso e alguns cobram por um serviço a mais.

Esses tipos de frameworks suportam o desenvolvimento webapp, onde ele é capaz de rodar um website dentro de um navegador dentro de aplicativo exibindo o conteúdo como se fosse um aplicativo nativo. Ele pode ser capaz de rodar sem precisar de nenhuma conexão com a internet, já que o próprio webapp pode está hospedado dentro do próprio aplicativo. Grande maioria dos frameworks funcionam dessa maneira, alguns do mais interessantes estão listados abaixo:

* **Cordova:** Apache Cordova é um conjunto de APIs de dispositivos que permitem o acesso a funções de dispositivos nativo, como a câmera ou o acelerômetro utilizando Javascript. Isso permite que um aplicativo de smartphone seja desenvolvido apenas com HTML, CSS e Javascript. Ele é muito utilizado como base para outros frameworks.
* **PhoneGap:** O PhoneGap não é exatamente um framework, na verdade ele é baseado no código aberto do Cordova que lhe fornece uma API para compilar os arquivos de tecnologias web para nativos. Dessa maneira ele é capaz de gerar aplicativos para iOS, Android, Blackberry, Symbian, Bada e Windows Phone. Essa é a principal escolha dos desenvolvedores pela sua popularidade e quantidade de material disponível.
* **Sencha Touch:** É propriedade da empresa Sencha e é baseado na biblioteca ExtJS, que é uma biblioteca da Sencha para desenvolver aplicações com interfaces ricas. É muito interessante para os desenvolvedores que ja vem utilizando essa biblioteca para desenvolvimento web.
* **Appgyver:** Assim como o PhoneGap, o Appgyver tem como estrutura base a API do Cordova, mas o seu principal diferencial é o Steroids.js. O Steroids.js é como o próprio nome sugere, ele tem algumas implementações que melhoram a API do Cordova.
* **Ionic:** IONIC é um dos mais promissores HTML5 frameworks de aplicação móvel. Construído com SASS, fornece muitos componentes de interface do usuário para ajudar a desenvolver aplicativos ricos e interativos. Ele usa o framework Javascript MVVM, AngularJS para aplicações.

2.3.2 AppGyver

O AppGyver foi escolhido para o estudo de caso de trabalho por conta de utilizar tecnologias modernas e de ótimo desempenho. O Steroids.js fornecido pelo AppGyver é capaz de acessar novos recursos nativos dos dispositivos, que não são acessados pelo Cordova, causando melhor desempenho pelo fato de estar deixando de utilizar algo híbrido para acessar recursos nativos.

2.3.3 Angular JS

Conhecido também como angular, é um framework em Javascript open source mantido pela Google, comunidade de desenvolvedores individuais e por empresas que utilizam o framework para desenvolver aplicações web. Seu objetivo é simplificar tanto o desenvolvimento quanto o teste de tais aplicações, fornecendo um quadro para client-side de arquitetura MVC (Model View Controller), juntamente com os componentes comumente usados ​​em aplicações ricas para internet.

Por sua arquitetura sem uma das mais utilizadas por frameworks nos dias de hoje, o angular ganhou bastante força com isso e por ser utilizado em projetos internos e externos da Google.

**3 ESTUDO DE CASO**

Esse estudo de caso tem como objetivo, desvendar como é possível desenvolver aplicações multiplataformas utilizando somente um código, reutilizando o mesmo para gerar um outro aplicativo. Isso utilizando tecnologias web em conjunto com o AppGyver.

O resultado desse estudo é um aplicativo Bíblia que será desenvolvido com o conteúdo desse trabalho e no final estará disponível para ser utilizado no IOS e no Android. O aplicativo utilizará o texto em versão da Almeida Recebida (Bíblia em Português) terceira revisão, que listará todos os livros e ao escolher, o usuário terá acesso ao texto.

**Figura 05.** Telas do aplicativo Bíblia



**Fonte:** BRITO, 2014.

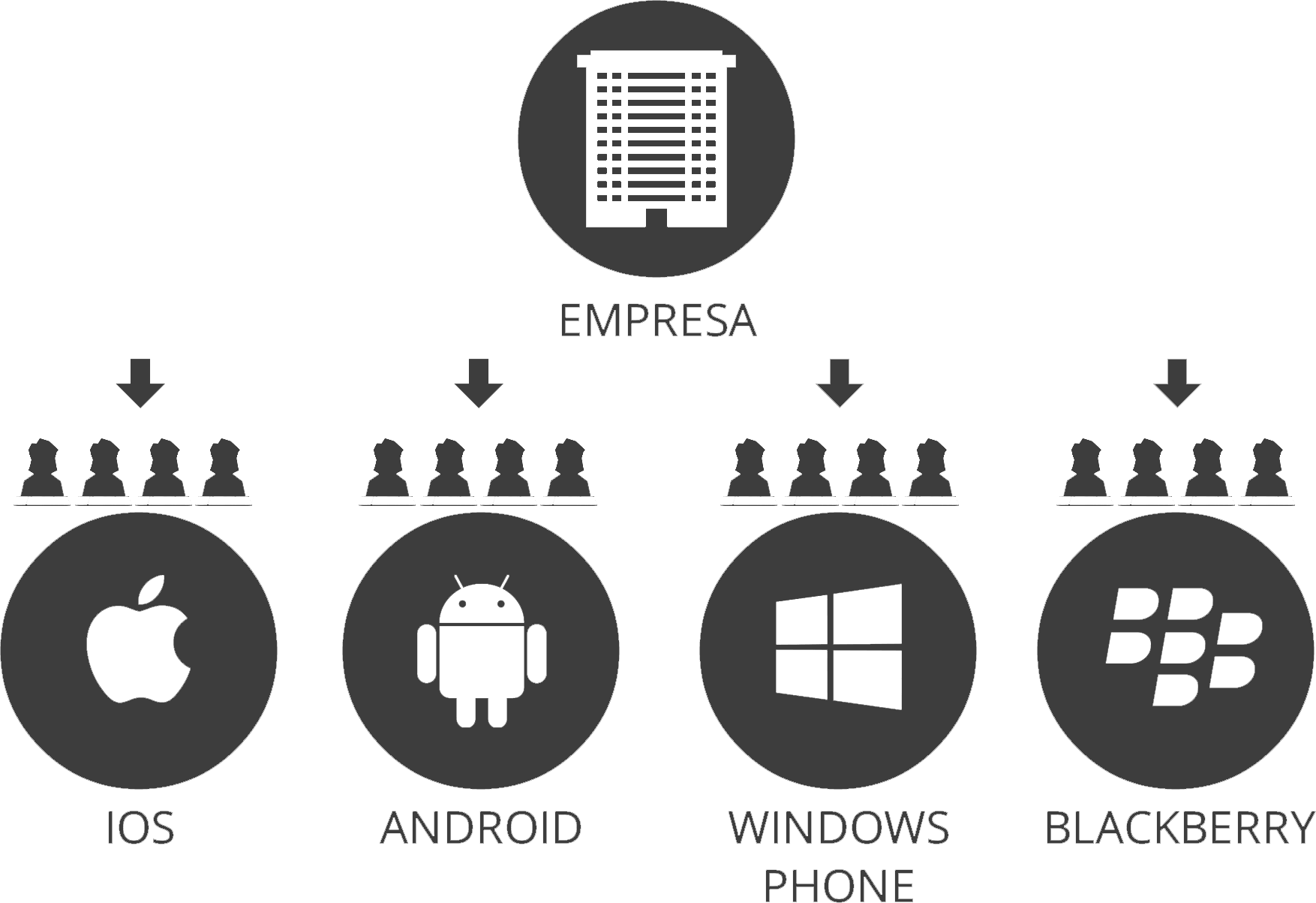
**3.1 Escreva um e rode em qualquer lugar**

Segundo Allen et al. (2010, p. 5), estes novos frameworks são influenciados pelo RAD (Rapid Application Development). As demandas são muitas, o mercado vem crescendo muito rápido e já estava na hora de encontrar uma solução para se desenvolver aplicações suportáveis por várias plataformas.

O desenvolvimento nativo sempre foi utilizado como a principal maneira de se desenvolver especificamente para uma plataforma. Seguramente é o que tem melhor desempenho e estabilidade, pelo fato de se utilizar tecnologias recomendadas pela empresa responsável pela plataforma. Esse conjunto de ferramentas é chamado de SDK (Software Development Kit). O grande problema de soluções nativas é que elas estão qualificadas apenas para executarem em sua plataforma de origem e limitando o alcance da aplicação, quando uma das maiores prioridades é atingir o máximo de usuários final possível.

Analisando um cenário muito comum com o surgimento das startups (empresas com grande potencial que estão começando):

**Figura 06.** Cenário de desenvolvimento nativo

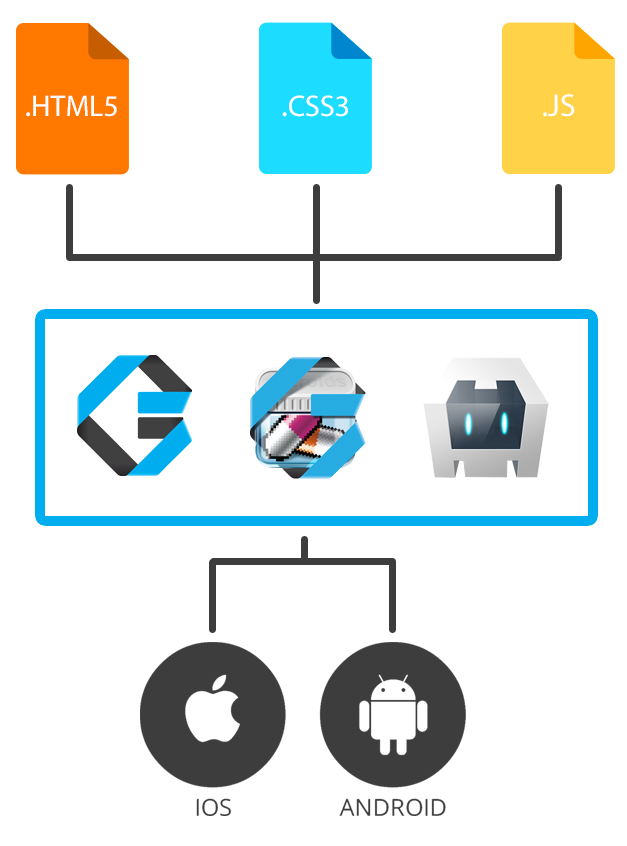


**Fonte:** BRITO, 2014.

Uma empresa tem um projeto e quer desenvolver ele para quatro plataformas totalmente diferentes (com exceção da BlackBerry que recentemente anunciou a compatibilidade com aplicativos do Android, mas ainda não é estável), mas sem ter muito gasto porque a empresa está começando e deseja ter um feedback do usuário para melhorar o seu produto antes de investir mais ainda. O que normalmente é feito quando a empresa tem um suporte financeiro bom, é contratar desenvolvedores nativos para cada plataforma, formando quatro equipes de desenvolvimento, gerando um custo alto e se não tiver um bom gerenciamento, a empresa também terá essas equipes trabalhando sem sincronia o que pode resultar um alguns padrões diferentes. Quando a empresa não pode contratar tudo isso, ela escolhe uma plataforma e investe nela, limitando o lançamento do seu aplicativo. O desenvolvimento nativo é infinitamente melhor, mas sem sempre o mais viável.

O desenvolvimento híbrido utiliza de tecnologias web conhecidas por vários desenvolvedores que com ajuda de frameworks é possível criar um aplicativo multiplataforma. A curva de aprendizado é muito menor e é possível treinar ou encontrar profissionais capacitados em pouco tempo e com um investimento muito menor do que a maneira nativa. A o aplicativo é escrito em HTML, CSS e Javascript e é submetido para a compilação de um framework (no nosso caso é o AppGyver), após algumas configurações é possível tem o aplicativo executável no IOS e no Android.

**Figura 07.** Processo de compilação de um aplicativo híbrido



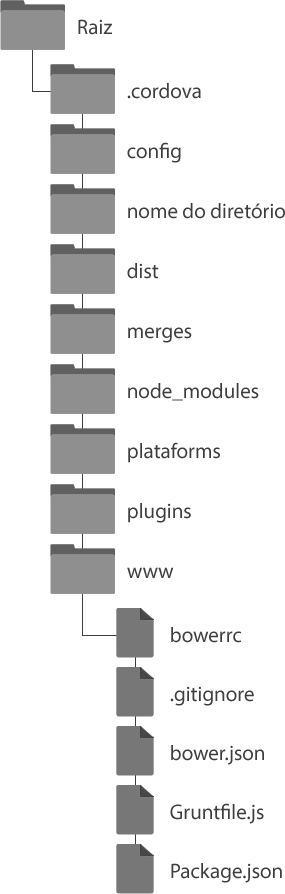
**Fonte:** BRITO, 2014.

Como é ilustrado acima, um único conjunto de arquivos com linguagens populares é possível com a ajuda do AppGyver, que é responsável por gerenciar suas dependências (Steroids.js e Cordova), gerar aplicativos para IOS e Android. Como foi apresentado no desenvolvimento nativo seria necessário uma equipe de desenvolvimento para cada plataforma, e no desenvolvimento híbrido é possível reutilizar o código para outras plataformas.

**3.2 Arquitetura**

O Steroids possui um construtor “make” que é um script responsável por executar várias sub tarefas do Grunt definidas no plugin “grunt-steroids”. O resultado dessas tarefas é o diretório “/dist” na raiz do projeto. Dentro do diretório “/dist“ está toda a estrutura necessária para aplicação executar no dispositivo e na medida que a aplicação for sofrendo mudanças o diretório “/dist” vai sendo atualizado. Abaixo a estrutura de diretórios definidos por padrão.

**Figura 08.** Estrutura de diretórios do AppGyver



**Fonte:** BRITO, 2014.

Todos os arquivos da aplicação podem ficar no diretório “/www”, eles serão copiados para o diretório “/dist“ assim que a tarefa “make” for executada. Mas existe um problema dependendo do tamanho da aplicação, que é a organização dos arquivos e seus códigos que se forem feitos assim ficarão muito desorganizado, causando confusão.

O AppGyver suporta o padrão MVC (Model View Controller) que é um padrão de arquitetura muito utilizado para melhorar a organização de projetos. Para melhor proveito dos frameworks e qualidade da aplicação o AppGyver recomenda criar o diretório “/app” no diretório raiz do projeto e dentro arquitetar a estrutura MVC.

3.2.1 View

Para as Views é necessário criar um diretório com esse nome dentro do diretório “/app” e todos os arquivos que ficarão responsáveis por exibir conteúdos irão dentro desse diretório criado. Como para cada página do aplicativo que for carregada ela precisará que todas as dependências sejam carregadas também. Isso iria resultar em muita repetição de código e muito trabalho para fazer a manutenção, porque se fosse preciso adicionar uma dependência a mais, teria que ser feito isso em todos os arquivos. Para evitar esse problema é possível criar um diretório dentro de “/app/views/layout” e depois criar um arquivo HTML5 que terá todo o conteúdo das views renderizado nele.

**Figura 09.** Arquivo app/views/layouts/book.js

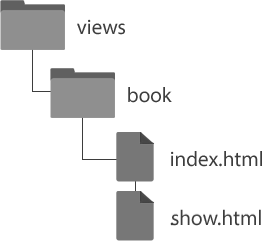


**Fonte:** BRITO, 2014.

Esse exemplo acima é o arquivo “/app/views/layouts/book.html“ utilizado no desenvolvimento do aplicativo Bíblia e pode ser visto que da linha 7 até a linha 21 estão sendo carregadas dependências que toda página vai precisar, dessa forma na linha 25 é renderizado o conteúdo referente a uma página da aplicação, aproveitando todo o código já feito.

Os arquivos com as parciais que são renderizadas, ficam um dentro do diretório “/app/views“ agrupados por classe para ficar mais organizado, mas dependendo do desenvolvedor ele tem liberdade para distribuir os arquivos da maneira que achar melhor. A estrutura utilizada no aplicativo Bíblia foi a seguinte:

**Figura 10.** Estrutura do diretório view



**Fonte:** BRITO, 2014.

Nessa modelagem os arquivos e os métodos “index” são utilizados para exibir ou requisitar uma coleção de algum objeto, que nesse caso são os livros da Bíblia. Já os arquivos e métodos “show” são utilizados para exibir ou requisitar apenas um item de alguma coleção que nesse caso é um livro. Dentro desses arquivos estão somente o conteúdo especifico de casa um, assim como seus estilos. Esses arquivos serão exibidos na próxima sessão.

3.2.2 Controller

Assim como na camada das views, os arquivos das controllers são melhores distribuídos utilizando o nome das classes, mas só que dessa vez o nome é atribuído ao arquivo. Esses arquivos ficam dentro diretório “app/controllers/” e serão importados na linha 21 da figura 09 onde o roteamento vai dizer qual o objeto que está sendo requisitado, fazendo desnecessário carregar arquivos de outras controllers.

Cada controller tem métodos referente ao objeto e onde é feita a logica de negócio das aplicações e para cada view há um controlador que nesse caso é utilizando o framework AngularJs.

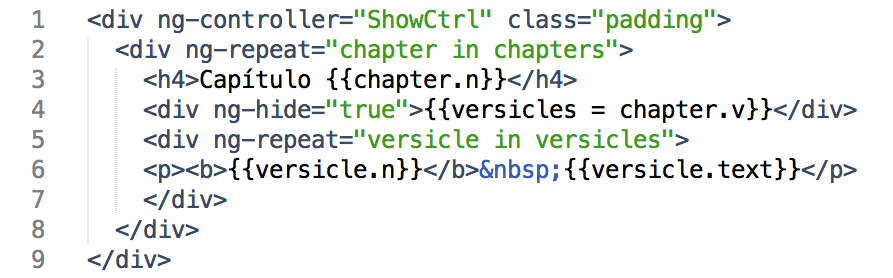
**Figura 11.** Arquivo app/controllers/book.js



**Fonte:** BRITO, 2014.

Nas linhas 4 e 25 são instanciadas as controllers “IndexCtrl” e “Show“ e cada uma com os métodos referentes a sua controller. É possível notar que os nome dos métodos contém explicito o motivo de sua criação e pelo o que ela é responsável e seus nomes são similares aos arquivos das views que estão no diretório “app/views/”. Para que o Angular entenda que parte da view é gerenciada por alguma controller é preciso atribuir a tag HTML uma diretiva especificas para o uso das controllers assim como mostra no código abaixo:

**Figura 12.** Arquivo app/views/book/show.html



**Fonte:** BRITO, 2014.

A diretiva que se encontra na primeira DIV na linha 1 nomeada de “ng-controller“ é a responsável por informar ao Angular que todo o conteúdo interno dela é submetido a controller “ShowCtrl“ que se contra no arquivo “app/controllers/book.js” conforme demonstrado na figura 11.

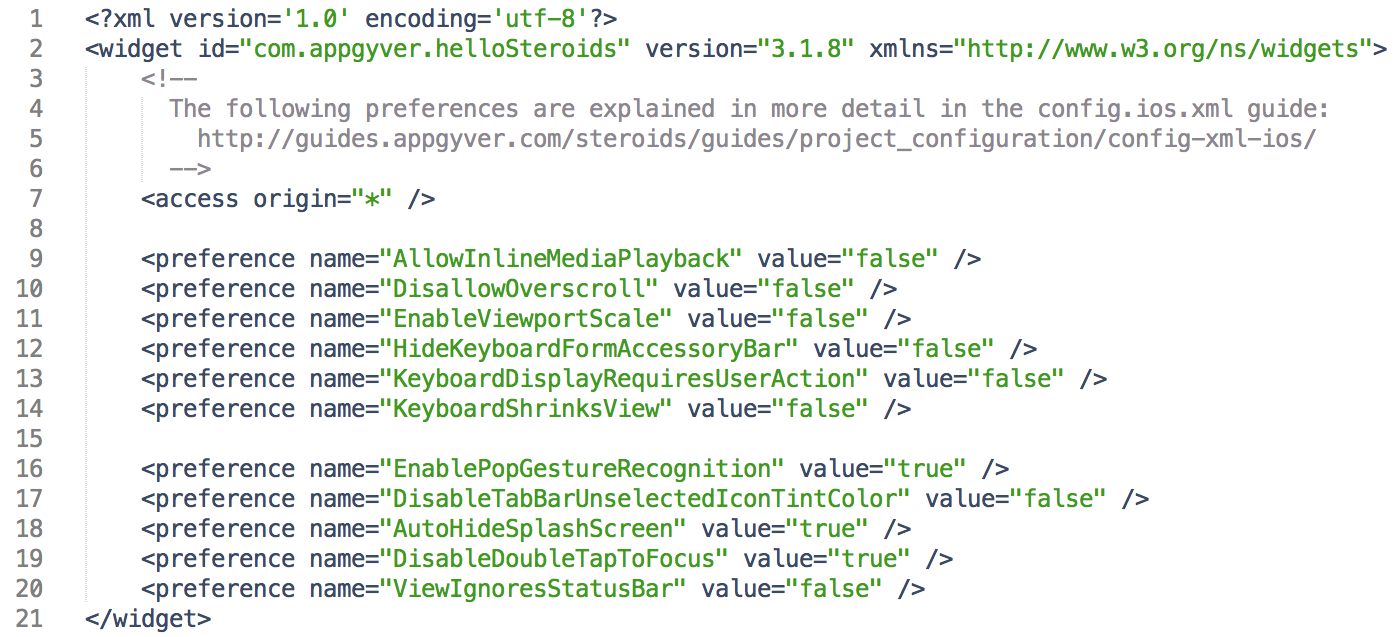
3.2.3 Model

A importação e a organização das dependências dos models funcionam exatamente como para as controllers e pode ser visto na linha 19 da figura 9. A grande diferença nos models é quando a task make executa e manda os arquivos para a pasta dist, os arquivos do models são enviados separadamente e também é criado um arquivos models.js que contém todos os arquivos models concatenados podendo carregar todos de uma vez só que seja preciso.

**3.3 Configuração do Projeto**

Apesar da excelente interpretação do Cordova e do Steroids sobre o HTML5 CSS3 e Javascript, alguns pontos ainda causam incompatibilidade entre as duas plataformas que o AppGyver suporta(IOS e Android). Por isso o Cordova e o Steroids utilizam arquivos no formato .xml para descreverem as preferências universais para WebViews do aplicativo, distintas de cada plataforma causando nesse momento uma pequena separação entre as plataformas para configurações individuais. O código dentro do diretório “/app” permanecerá salvo, compatíveis e sem interferências diretas das configurações dos arquivos xml.

**Figura 13.** Arquivo de configuração de preferencias para IOS localizado em www/config.ios.xml



**Fonte:** BRITO, 2014.

**3.4 Acessando recursos nativos**

Um grande diferencial para se utilizar o AppGyver é pela integração com o Steroids.js. Com ele é possível acessar recursos nativos de um dispositivo tornando as aplicações híbridas cada vez mais com a aparência e desempenho iguais as nativas. O Steroids conta com um código modulado em Javascript e é possível através de um linguagem utilizada na web acessar esses recursos com os comandos mostrado na lista abaixo.

**3.5 Distribuição**

O AppGyver e suas dependências são altamente poderosos e capazes de facilitar o deploy de um aplicativo, isso devido a uma estrutura de um servidor online que permite receber essas demandas de forma gratuita, mas também inclui uma versão paga com mais benefícios. Tudo começa quando um simples comando no terminal, com o comando “steroids deploy” é possível disparar uma cadeia de eventos que são responsáveis principalmente por rodar internamente os comandos “steroids make” e “steroids package”. O “steroids make” atualiza todo o conteúdo do diretório “/dist“ e os arquivos são submetidos previamente as tarefas descritas no grunt onde as principais são a minificação e concatenação de arquivos.

Para ter acesso aos repositório no servidor que fica disponível do AppGyver, é preciso ter uma conta criada, e a partir desses dados é possível se autenticar pelo terminal com o comando “steroids login”. Depois de se estar autenticado e executar o comando “steroids deploy” pela primeira vez em uma aplicação ele irá criar arquivo no diretório “/config” com o nome “cloud.json” e dentro desse arquivo serão armazenados os dados “id” e “identification\_hash” e esses dados serão fundamentais para a identificação de quem está fazendo deploy e de que aplicativo.

Algumas configurações só são realizadas após o deploy ser completado, elas estão disponíveis exclusivamente no site do AppGyver nas opções de gerenciamento de aplicativos. Essas configurações devem ser feitas separadamente para cada plataforma que se deseja instalar o aplicativo, entre essas opções estão as escolhas dos ícones e splashscreens em vários tamanhos e tipo acessos ao dispositivo serão necessário para seu pleno funcionamento e entres essas opções estão, acelerômetro, giroscópio e outras. Após isso o aplicativo é submetido para a compilação e ficará disponível para download ao desenvolvedor geralmente em poucos minutos.

**4 CONCLUSÃO**

O mercado para desenvolvimento mobile vem crescendo e se modificando bastante e com a existência de novas plataformas e as já consolidadas, estão exigindo muito mais das pessoas e empresas envolvidas nesse processo. A concorrência tem sido grande e investimentos altos, a cada dia são publicados milhares de aplicativos distribuídos em diversas categorias e estilos de usuários e o tempo ainda tem sido o maior inimigo para se entregar os aplicativos a tempo de seu lançamento colocando todo um projeto de meses em risco.

Diversas metodologias já existem e são bastante utilizadas, mas foi apresentado os benefícios em escolher o desenvolvimento híbrido e o reaproveitamento de códigos entre diferentes plataformas. Várias ferramentas estão habilitas para auxiliar nesse processo utilizando tecnologias web como HTML5, CSS3 e Javascript. Dentre essas ferramentas foi abordado mais profundamente sobre o AppGyver que com suas dependências permitem a criação de aplicativos multiplataforma em pouquíssimo tempo e código.

Algumas configurações só são realizadas após o deploy ser completado, elas estão disponíveis exclusivamente no site do AppGyver nas opções de gerenciamento de aplicativos. Essas configurações devem ser feitas separadamente para cada plataforma que se deseja instalar o aplicativo, entre essas opções estão as escolhas dos ícones e splashscreens em vários tamanhos e tipo acessos ao dispositivo serão necessário para seu pleno funcionamento e entres essas opções estão, acelerômetro, giroscópio e outras. Após isso o aplicativo é submetido para a compilação e ficará disponível para download ao desenvolvedor geralmente em poucos minutos.

**REFERENCIAS**