INTERFACE TECNOLÓGICA

A LINGUAGEM JAVASCRIPT COMO ALTERNATIVA PARA O DESENVOLVIMENTO DE APLICAÇÕES MULTIPLATAFORMA

THE JAVASCRIPT LANGUAGE AS AN ALTERNATIVE FOR THE DEVELOPMENT OF MULTIPLATAFORM APPLICATIONS

Vitor da Silva Cruz – vitor.dasilvacruz@gmail.com

Erick Eduardo Petrucelli – erick.petrucelli@fatectq.edu.br

Eder Carlos Salazar Sotto – eder.sotto@fatectq.edu.br

Faculdade de Tecnologia de Taquaritinga (FATEC) – SP – Brasil

DOI: 10.31510/infa.v15i2.476

RESUMO

O mercado de desenvolvimento aplicações em geral vem crescendo ano após ano, juntamente com o uso crescente de dispositivos, sejam eles *desktops*, *smartphones*, *laptops*, *smartvs*, entre outros. Estes aparelhos usam sistemas operacionais que os gerenciam e permitem a instalação e o uso de aplicações, bem como conexão com a internet. Estes aparelhos, ainda que do mesmo tipo, divergem entre diversos sistemas operacionais, o que acaba ocasionando o surgimento de um novo problema, que é o de como desenvolver para todas as plataformas de forma mais ágil, a ponto de não necessitar de códigos muito diferentes para plataformas e sistemas operacionais diferentes. O presente trabalho é baseado em artigos, documentações, e livros, e tem como objetivo apresentar o JavaScript como uma maneira de se minimizar as dificuldades dos programadores em aprender e desenvolver em diferentes linguagens para cada plataforma e sistema, como também das empresas, em precisar possuir times específicos para cada plataforma, além do desafio de integrar estes times, de forma que, ainda que trabalhem em plataformas distintas, mantenham-se integrados como um projeto único. Ao final, encontram-se as considerações finais sobre os conteúdos que serão apresentados.

Palavras-chave: Aplicações; Desenvolvimento; JavaScript; Plataformas; Programação;

ABSTRACT

The market for application development in general has been growing year after year, along with the increasing use of devices, be they dektops, smartphones, laptops, smartvs, among others. These devices use operating systems that manage them and allow the installation and use of applications as well as connection to the internet. These devices, although of the same type, diverge between different operating systems, which ends up giving rise to a new problem, which is how to develop for all the platforms in a more agile way, to the point of not needing very different codes for different platforms and operating systems. The present work is based on articles, documentation, and books, and aims to present JavaScript as a way to minimize the difficulties of programmers in learning and developing in different languages

for each platform and system, as well as of companies in need have specific teams for each platform, and the challenge of integrating these teams, so that even if they work on different platforms, they remain integrated as a single project. At the end, you will find the final considerations about the contents that will be presented.

Keywords: Applications; Development; JavaScript; Platforms; Programation;

1 INTRODUÇÃO

A utilização de computadores e dispositivos móveis são crescentes em todas as áreas, seja nas áreas industriais, agrícolas, comercial, aeronáutica entre outros. Este crescimento não é somente de computadores comuns, mas também aparelhos computacionais e microcomputadores como: *smartphones, laptops, smartvs*, etc.

"O estudo estima ainda que o mercado doméstico de *software* e serviços de TI deve crescer entre 2017 e 2021. [...] as vendas de software e serviços de TI devem alcançar USD 25,8 bilhões com um aumento de 5,7% por ano." (COMPUTERWORLD, 2018). Com o crescente mercado de *software* cresce também a demanda por mão de obra, o que implica na quantia de plataformas ao qual irá atingir os ambientes de desenvolvimento, as linguagens e a quantidade de pessoas ao qual irão compor a equipe de desenvolvimento.

Javascript é usado por uma grande quantidade de desenvolvedores e está presente em quase todos, senão todos os *websites*, e com a revolução da indústria 4.0, deverá ganhar maior relevância (EXAME, 2018).

O *JavaScript* não está mais apenas no cenário *Web*, mas começou a ser uma das alternativas para equipes de programação, bem como empresas e *startups* de desenvolvimento para ser utilizado na a criação de *softwares desktop*, *Web*, aplicativos para dispositivos móveis, microcomputadores, e assim por diante.

2 METODOLOGIA

Os procedimentos metodológicos para produção do artigo foi através de análise de livros, estudo sobre artigos, entre outros. Pela carência de conteúdos acadêmicos sobre diversos dos assuntos abordados, também foi utilizado como base de estudo as documentações oficiais de tais tecnologias e ferramentas, disponibilizado pelas equipes de desenvolvimento destas. A escolha de tais conteúdos se deu por levantamento de dados,

verificação da relevância entre os usuários destas ferramentas, bem por ser livros ou artigo aceito e publicado, ou mesmo documentações oficiais das mesmas.

2.1 Estrutura

Na introdução, encontra-se a problematização do desenvolvimento de aplicações multiplataformas, bem como uma introdutória a linguagem de programação JavaScript, ao qual se torna o foco deste artigo.

No Capítulo 2 é apresentada a metodologia do presente trabalho.

No Capítulo 3 é apresentado um pouco da história da linguagem JavaScript, e cresceu ao ponto de atingir diversas outras plataformas além daquela ao qual era seu foco.

No Capítulo 4, apresenta-se uma discutição sobre as vantagens do uso da linguagem JavaScript para criação de aplicações diversas.

No Capítulo 5, conclui-se o que está proposto como estudo do artigo apresentando alguns pontos chaves descobertos ao longo de toda a pesquisa.

3 JAVASCRIPT DOS NAVEGADORES AO MUNDO

Segundo Rauschmayer (2014, cp. 4), no ano de 1994 uma empresa com o nome de *Netscape* (Atualmente Mozilla) com o potencial crescente da internet criou o seu navegador, percebendo que a *Web* necessitava ser mais dinâmica, contratou um dos desenvolvedores responsáveis pela criação da linguagem *Java*, então pertencente à empresa *Sun Microsystem* (Atualmente *Oracle*), ao qual deu início ao desenvolvimento da linguagem ao qual conhecemos como *JavaScript* (*EcmaScript*). O *JavaScript* nasceu com a finalidade de programar e dinamizar as paginas *web*, ao qual até então eram estáticas, e partir disto começou seu crescimento como linguagem de programação.

3.1 A Plataforma Web

Kurose e Ross (2013, p. 2), apontam a *internet* como o que conecta centenas de milhões de dispositivos computacionais ao redor do mundo (*WEB*), ao qual através de estações de trabalhos de serviços (servidores) transmitem informações e servem funcionalidades, tais como *websites*, *emails*, videochamadas dentre outros. A *internet* hoje

conecta não só os usuários entre si como oferecem serviços aos clientes. A *web* é divida em dois, o lado cliente, que é a parte ao qual se pode ver através de navegadores ou outras aplicações conectadas á *internet*, e o lado servidor, que é responsável por diversos serviços aos usuários e também á outros serviços.

3.2 JavaScript Front-End

> Algumas páginas apresentam informações estáticas e podem ser chamadas de documentos. (A apresentação dessa informação estática pode ser bastante dinâmica - por causa do JavaScript, mas a informação em si é estática). Outras páginas da web parecem mais com aplicativos do que com documentos. Essas páginas podem carregar dinamicamente novas informações conforme necessário, elas podem ser gráficas em vez de textuais e podem operar off-line e salvar dados localmente para que possam restaurar seu estado quando você visitá-los novamente. (FLANAGAN, 2011, p. 307)

Toda esta interação que torna as páginas *web* de apenas um documento estático para uma aplicação dinâmica é graças ao *JavaScript* no lado cliente (navegador).

3.3 NodeJS e o Início do JavaScript para além dos Navegadores

Google (2018), caracteriza o V8 como o mecanismo de código aberto escrito em C++ de execução de *JavaScript* de alto desempenho, ao qual sua implementação se dá no seu próprio navegador (*Google Chrome*) e no *Node.js* podendo ser executado em sistemas *Windows*, *MacOS*, e *Linux* ou até mesmo incorporado em aplicações C++. O que possibilita o uso na maioria dos aparelhos de hoje ao quais alguns são programados em C++.

> Node é um rápido interpretador JavaScript baseado em C ++ com ligações às APIs de baixo nível do Unix para trabalhar com processos, arquivos, soquetes de rede e também para APIs de clientes e servidores HTTP. Com exceção de alguns métodos síncronos especialmente nomeados, os vínculos do Node são todos assíncronos e, por padrão, os programas Node nunca são bloqueados, o que significa que eles normalmente são dimensionados adequadamente e lidam com cargas altas de maneira eficaz. (FLANAGAN, 2011, p. 296)

Node.js utiliza a *engine* (mecanismo) de execução de *JavaScript* criado pela equipe de desenvolvimento da *Google*, o qual possibilitou a criação e execução de códigos *JS* (*JavaScript*) fora dos navegadores, podendo acessar *APIs* (Interface de Programação de

Aplicações) nativas, o que antes não era possível através do navegador, como a *API* de manipulação de arquivos do sistema.

3.4 JavaScript *Back-End*

"Node.js é um ambiente de execução multi-plataforma em JavaScript que permite aos desenvolvedores produzirem aplicações *server-side* para a rede usando o JavaScript como linguagem." (MDN, 2018). O *Node* permite ao desenvolvedor criar um servidor *web* com a utilização de *JavaScript*, com acesso á *APIs* nativas do sistema assim como outras linguagens de servidores o fazem.

Diferente de outras linguagens de servidor, *Node* é orientado a eventos, é *mono-thread* (utiliza um processo, o principal) e é assíncrono, evitando assim o bloqueio de seu processo principal, porém sendo possível utilizar-se do multi-processamento ao se desenvolver utilizando-o (NODE, 2018).

3.5 JavaScript em banco de dados

Segundo RethinkDB (2018), o banco de dados *RethinkDB* é um banco de dados em *JSON* (*JavaScript Object Notation* ou Notação de Objeto *JavaScript*) de código aberto para aplicações em tempo real, ao qual se utiliza de expressões *JavaScript* para a manipulação de várias funções.

O banco de dados *MongoDB* utiliza-se do formato *BSON* (*JSON* Binário) para armazenagem de dados, e de execução de *JavaScript* como comando para execução de funções do banco em seu terminal de comando (MONGO, 2018).

Os bancos apresentados são exemplos de bancos baseados em notação de objetos *JavaScript*, de uma forma ou outra, e execução de *scripts* da mesma linguagem para execução de funções dentro deles.

3.6 Desenvolvimento *Desktop* com Electron

> Electron é uma biblioteca de código aberto desenvolvida pelo GitHub para o desenvolvimento de aplicativos desktop multiplataforma usando HTML, CSS e JavaScript. O Electron consegue isso combinando Chromium e Node.js em um único ambiente de execução. Aplicativos Electron podem ser lançados para Mac, Windows e Linux. (GITHUB, 2018)

O *Electron* possibilita a criação de aplicações *desktop* multiplataforma, ou seja, sistemas Windows, Linux e MacOS com um mesmo código-fonte, tendo o *Node* como base. Esta ferramenta permite acessar *APIs* nativas como manipulação de arquivos, e *APIs* Web por estar sendo renderizado em um navegador, o que facilita desenvolver aplicações para acesso a alguns periféricos como *webcam*, microfone entre outros. Os aplicativos desenvolvidos nele utilizam *HTML*, *CSS* e *JavaScript*. O desenvolvedor pode utilizar a mesma base de código para desenvolver um *WebSite* e uma aplicação *desktop*, tendo ambas a mesma interface, porém aplicação desktop pode ter funções extras como acesso a pastas do sistema.

3.7 Desenvolvimento Desktop com Proton-Native

Diferentemente do *Electron*, o *proton-native* não utiliza o *Chromium*, portanto o *HTML* e *CSS* não são utilizados para a criação de interface, mas sim uma *lib* (biblioteca) que é compilada em uma *interface* nativa dos sistemas operacionais, *MacOS*, *Windows* e *Linux*, utilizando o *React* e o *Node* para criação da interação e programação da lógica (PROTON-NATIVE, 2018).

3.8 Desenvolvimento Desktop com Vuido

> O Vuido é uma estrutura para criar aplicativos de desktop nativos baseados no Vue.js. Os aplicativos que usam o Vuido podem ser executados no Windows, OS X e Linux, usando componentes GUI nativos [...]. Os aplicativos que usam o Vuido têm aparência e comportamento nativos em cada plataforma, sem precisar escrever código separado ou usar linguagens de programação específicas da plataforma. (VUIDO, 2018)

O *Vuido* utiliza-se de um *framework JavaScript*, o *Vue.js*, para a codificação da interface e programação da aplicação, assim como *proton-native* possibilita a compilação para as três plataformas *desktop* mais conhecidas e utilizadas, *Windows*, *MacOS* e *Linux*, usando apenas um código-fonte.

3.9 Desenvolvimento Móvel com NativeScript

NativeScript possibilita a criação de aplicações para *Android* e *iOS* com desempenho nativo, utilizando o renderizador nativo de cada plataforma para a criação da aplicação. Pode

ser utilizado o *Vue.js*, o *Angular* com o *TypeScript* ou *JavaScript* puro, além do *XML* para os elementos visuais e o *CSS* para estilização (ŠMITALOVÁ, 2017, p. 11).

3.10 Desenvolvimento Móvel com Weex

Segundo a empresa Alibaba (2018), "O Weex permite que os desenvolvedores usem a experiência moderna de desenvolvimento da Web para criar aplicativos para Android, iOS e Web com uma única base de código."

3.11 Desenvolvimento Móvel com React-Native

> Como um aplicativo no React Native não usa WebViews, é possível criar aplicativos com uma capacidade de resposta nativa. Além disso, em vez de empurrar uma base de código para todas as plataformas, uma base de código compartilhada é gravada e implantada nas plataformas e apenas algumas seções, como elementos gráficos e componentes específicos da plataforma, são gravadas separadamente para cada plataforma de destino. [...] o React Native é capaz de renderizar os componentes React Native para visualizações nativas reais para Android ou UI Views para iOS. Isso é possível devido à camada de abstração conhecida como "ponte", que permite que o React Native invoque as APIs de renderização no Java para Android ou no Objective-C para iOS. Além disso, o framework revela a interface do JavaScript, permitindo que o aplicativo acesse recursos específicos da plataforma como a bateria ou o local.(DANIELSSON, 2016, p.11)

O *React-native* renderiza o componente nativo da plataforma ao qual está a rodar, e através de uma ponte, criada entre ele e o aparelho, faz o acesso as ferramentas nativas, como o GPS por exemplo.

Nestas subseções de desenvolvimento móvel observou-se que o *JavaScript* é uma opção viável para o desenvolvimento de aplicações móveis, com performance nativa e com maior aproveitamento de código entre plataformas.

3.12 Desenvolvimento Físico

Segundo a empresa Tessel (2018), a placa ao qual leva o mesmo nome da empresa, *Tessel*, é uma plataforma de desenvolvimento *IoT* (*Internet of Things* – Internet das Coisas) e robótica, ao qual é possível interagir com o mundo físico através de sensores, câmeras e tantos

outros dispositivos utilizando o *JavaScript* e módulos do *Node*, sendo seu suporte a *JavaScript* nativo.

Com o *JavaScript* é possível controlar dispositivos e placas IoT, que podem ser utilizados em diversas situações no mundo físico, como programar o acender de luzes, sensores entre outros, (BOCOUP, 2018).

4 DISCUSSÃO SOBRE USOS DO JAVASCRIPT

> Os benefícios em utilizar uma aplicação baseada em JavaScript abrange dois fatores principais: (i) Escalabilidade; que se define pela disponibilidade de serviços requisitados a proporções crescentes de novos usuários; e (ii) Performance; o desempenho obtido levando em conta a escalabilidade e recursos consumidos. Outro benefício que justifica a utilização de JavaScript é por ser uma linguagem orientada a eventos, fazendo com que suas operações sejam executadas somente quando as mesmas são requisitadas em alguma parte da aplicação. (BERA, MINE e LOPES, 2015, p. 2)

Tal informação indica que algumas das vantagens da utilização do *JavaScript* para desenvolvimento de aplicações é a fácil escalabilidade, desempenho, e a forma de se programar, a qual é orientada a eventos.

Segundo Mardan (2015, p. xx), algumas das razões de se programar *full stack* com *JavaScript* são a reutilização de códigos, a não mudança de contexto ao se mudar de plataforma alvo, a não necessidade de compilação por conta da interpretação da linguagem na hora de executar, um bom desempenho através de uma estrutura não bloqueante, entre outros.

4.1 Ferramentas de Testes e Geração de relatórios

De acordo com Facebook (2018), o *Jest* é uma ferramenta para testar código *JavaScript*, mostrando os resultados no terminal ou gerando relatórios de forma facilitada, podendo coletar as informações de códigos mesmo os não testados, entre tantas outras funcionalidades.

> O Mocha é um framework de teste JavaScript rico em recursos, rodando em Node.js e no navegador, tornando os testes assíncronos simples e divertidos. Os testes Mocha são executados em série, permitindo relatórios flexíveis e precisos, enquanto mapeiam exceções não detectadas para os casos de teste corretos. (MOCHA, 2018)

O *JavaScript* além de trazer várias vantagens na questão do desenvolvimento do código, também conta com bibliotecas que facilitam os testes de código, ajudando assim a descobrir erros antes de entrar em produção sem contar que tal ferramenta facilita na hora de documentar.

5 CONSIDERAÇÕES FINAIS

A *Web* cresceu muito nos últimos anos e uma parte disto é graças as interatividades que só foram possíveis por conta da criação do *JavaScript* que após certo tempo acabou não sendo exclusividade apenas dos navegadores de internet, podendo hoje ser utilizado em outras plataformas.

A criação de aplicações para a *internet* utilizando uma mesma linguagem de programação de *front-end* à *back-end* (*full stack*) torna a integração dos dois lados muito mais fácil.

Criar aplicações *desktop* para um desenvolvedor que até então só desenvolveu aplicações para a W*eb* antes era uma tarefa mais árdua necessitando despender de tempo para aprender outras linguagens e arquiteturas, geralmente mais de uma por conta da existência de vários sistemas operacionais (SO), ou no caso de empresas, até mesmo contratar equipes que desenvolvam para os SOs desejados. No entanto, *frameworks* como *Electron*, *Vuido* e *Proton-native* se tornaram uma alternativa, possibilitando a criação de uma mesma base de código em *JavaScript*, no qual pode ser gerado um executável para os três principais SOs do mercado, *Windows*, *Linux* e *MacOs*.

Aplicativos para *smartphones* e algumas plataformas que se utilizam de sistemas móveis, como TVs, as ferramentas *JavaScripts* como *Weex*, *React-native* e *NativeScript* possibilitam criar através do uso de código *JavaScript* aplicações para sistemas *Android* e sistemas da *Apple*.

Uma das coisas mais importantes em tudo isso, principalmente em projetos grandes, é que uma vez que a base de código é única, facilita a fase de testes evitando assim colocar em produção aplicações cheias de falhas e facilitando também na geração da documentação da aplicação.

Vários dos *frameworks* apresentados são criados por grandes empresas como Facebook, Alibaba, Github entre outros. É possível constatar com isso que investir no *JavaScript* pode servir como forma de sucesso, tanto da aplicação, como das empresas que

desenvolvem estes *frameworks*, pois muitas delas acabam utilizando estes *frameworks* em suas próprias aplicações.

O *JavaScript* evoluiu muito nos últimos anos como uma linguagem de programação podendo se encaixar em diversas plataformas e situações, mas nem sempre pode ser a opção mais viável. Sua viabilidade depende muito do tipo de aplicação, e principalmente do conhecimento dos desenvolvedores, que são cruciais para identificar qual a melhor escolha para o desenvolvimento de uma nova aplicação.

REFERÊNCIAS

ALIBABA. **Weex Documentation**. 2018. Disponível em: <https://weex.apache.org/>. Acesso em: 15 set. 2018.

BERA, M. H. G; MINE, A. F; LOPES, L. F. B. **MEAN Stack:** Desenvolvendo Aplicações Web Utilizando Tecnologias Baseadas em JavaScript. out. 2015. Maringá: Faculdade Cidade Verde. Disponível em: <http://fcv.edu.br/admin/assets/repositorio_arquivo/413a333c0cc0872f3ee8f9d22f7cc166.pdf >. Acesso em: 15 set. 2018.

BOCOUP. **Johny-five Documentation.** 2018. Disponível em: <http://johnny-five.io/>. Acesso em: 15 set. 2018.

COMPUTERWORLD. **Mercado de Softwares e serviços de TI deve manter crescimento até 2021, prevê estudo**. abr. 2018. Disponível em: <https://computerworld.com.br/2018/04/23/mercado-de-software-e-servicos-de-ti-deve-manter-crescimento-ate-2021-preve-estudo/>. Acesso em: 15 set. 2018.

DANIELSSON, W. **React Native application development:** A comparison between native Android and React Native. 2016. Sweden: Linköpings universitet. Disponível em: <http://www.diva-portal.org/smash/get/diva2:998793/FULLTEXT02.pdf>. Acesso em: 15 set. 2018.

GITHUB, **ELECTRON**. 2018. Disponível em: <https://electronjs.org>. Acesso em: 15 set. 2018.

EXAME. **Estas são as linguagens de programação para ficar de olho em 2018**. dez. 2017. Disponível em: <https://exame.abril.com.br/tecnologia/estas-sao-as-linguagens-de-programacao-para-ficar-de-olho-em-2018/>. Acesso em: 15 set. 2018.

FACEBOOK. **Jest Documentation.** Disponível em: <https://jestjs.io/pt-BR/>. Acesso em: 15 set. 2018.

FLANAGAN, D. **JavaScript:** The Definitive Guide. 6. ed. California: O'Reilly Media, 2011.

GOOGLE. **V8 Wiki**. 2018. Disponível em: <https://github.com/v8/v8/wiki>. Acesso em: 15 set. 2018.

KUROSE, J; ROSS, K. **Computer Networking:** A Top-down Approach. 6. ed. Londres: Pearson PLC, 2013.

MARDAN, A. **Full Stack JavaScript.** 2. ed. New York: Apress Media, 2015.

MDN. **MDN Documentation**. 2018. Disponível em: <https://developer.mozilla.org>. Acesso em: 15 set. 2018.

MOCHA. **Mocha Documentation.** 2018. Disponível em: <https://mochajs.org/>. Acesso em: 15 set. 2018.

MONGO. **MongoDB Documentation**. 2018. Disponível em: <https://docs.mongodb.com>. Acesso em: 15 set. 2018.

NODEJS. **NODE**. 2018. Disponível em: <https://nodejs.org>. Acesso em: 15 set. 2018.

PROTON-NATIVE. **Proton-native Documentation**. 2018. Disponível em: <https://proton-native.js.org>. Acesso em: 15 set. 2018.

RAUSCHMAYER, A. **Speaking JavaScript:** An In-Depth Guide for Programmers. 1. ed. Califórnia: O'Reilly Media, 2014.

RETHINKDB. **RethinkDB Documentation**. 2018. Disponível em: <https://www.rethinkdb.com/>. Acesso em: 15 set. 2018.

ŠMITALOVÁ,L. **NativeScript Application for Continuous Improvement of Software Engineer's Skills**.2017. Czech Republic: Faculty of Informatics,Masaryk University. Disponível em: <https://is.muni.cz/th/410198/fi_b/thesis.pdf>. Acesso em: 15 set. 2018.

TELERIK. **NativeScript Documentation.** 2018. Disponível em: <https://www.nativescript.org/>. Acesso em: 15 set. 2018.

TESSEL. **Tessel.** 2018. Disponível em: <https://tessel.io/>. Acesso em: 15 set. 2018.

VUIDO, **Vuido Documentation**. 2018. Disponível em: <https://vuido.mimec.org>. Acesso em: 15 set. 2018.