CONSTRUÇÃO DE APP COM REACT NATIVE

APP BUILDING WITH REACT NATIVE

Denys Alves da Silva,
Caio Frias de Sousa

RESUMO

A realidade brasileira de desemprego da população economicamente ativa atingiu o patamar de 13,4 milhões de pessoas no primeiro trimestre de 2019 (GONÇALVES, 2019), impelindo parte dessa população para o mercado informal, principalmente o de comerciante ambulante. Uma boa parte desses ambulantes entraram nesse empreendimento sem o preparo básico em controle do negócio, dessa forma, não tem a visibilidade se o empreendimento está dando retorno ou não. Este artigo relata esse cenário observado e experimentado por um dos autores em um local da região do Distrito Federal, Brasil. Após observado o processo dos comerciantes ambulantes, surgiu a questão em identificar o quanto os comerciantes ambulantes movimentavam em montantes financeiros, e se eles também gerenciam tem controle do seu negócio, com a finalidade de propor solução que poderia sensibilizar esses ambulantes a praticar no mínimo o Fluxo de Caixa para avaliar financeiramente seu empreendimento, ou até mesmo, estimular apoios de organizações governamentais, educacionais ou sociais na capacitação em gestão negócios para esses ambulantes. Decidiu-se pela construção de um app Fluxo de Caixa, pesando na decisão que "o Brasil registrou um número de 231.827.959 linhas móveis no mês de dezembro de 2018." (ANATEL, 2019), deduzindo que seria uma tecnologia acessível pelos comerciantes ambulantes. Este artigo também relata a experiência no desenvolvimento do app, e apresenta o estudo sobre as tecnologias para *mobile* que apresentem benéficos de baixos custos, continuidade, qualidade, produtividade, flexibilidade e facilidade, concluindo-se pela adoção do uso da biblioteca React, do *framework* React Native e do banco de dados Firebase para a construção do app Fluxo de Caixa.

Palavras chaves: App; React Native; Fluxo de Caixa.

ABSTRACT

The Brazilian unemployment rate of the economically active population reached 13.4 million people in the first quarter of 2019 (GONÇALVES, 2019), impelling some of this population to the informal market, especially the street vendor. A good part of these itinerants entered this enterprise without the basic preparation in control of the business, in this way, it does not have the visibility if the enterprise is giving return or not. This article reports on this scenario observed and experienced by one of the authors in a local area of the Federal District, Brazil. After observing the process of the street vendors, the question arose in identifying how much the street traders moved in financial amounts, and if they also manage has control of their business, with the purpose of proposing solution that could sensitize these itinerants to practice at least the Cash Flow to financially evaluate your venture, or even, stimulate support

from governmental, educational, or social organizations in training business management for these street vendors. It was decided to build a Cash Flow app, weighing in the decision that "Brazil registered a number of 231,827,959 mobile lines in the month of December 2018." (ANATEL, 2019), deducing that it would be a technology accessible by traders ambulantes This article also reports the experience in the development of the app, and presents the study on the technologies for mobile that present benefits of low costs, continuity, quality, productivity, flexibility and ease, concluding by the adoption of the React library, of the framework React Native and the Firebase database for building the Cash Flow app.

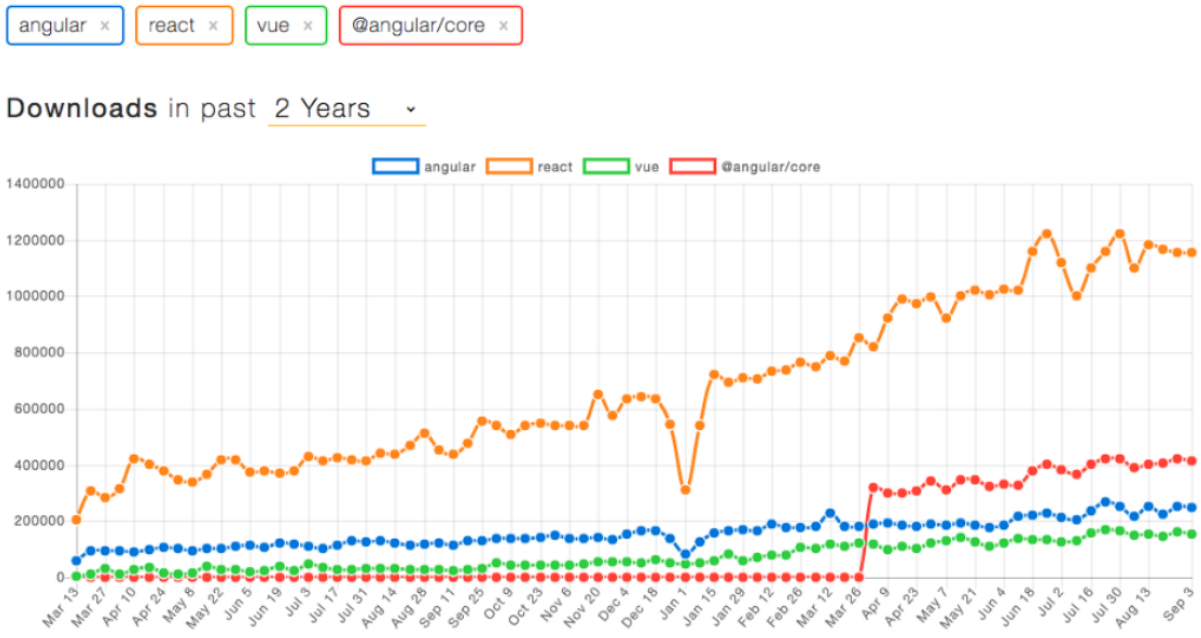
***Keyword**: App; React Native; Cash Flow*.

REACT

Segundo o tutorial de introdução ao React (TUTORIAL, 2019), o “React é uma biblioteca JavaScript declarativa, eficiente e flexível para criar interfaces com o usuário. Ele permite compor interfaces de usuário complexas a partir de pequenos e isolados códigos chamados componentes”.

O Facebook©, criador do React, em 2011 utilizou essa biblioteca pela primeira vez na interface do mural de notícias da rede social. Em 2012 foi implementado no *Instagram*, além de outras soluções da empresa. A partir de 2013, o Facebook© abriu o código do React para a comunidade, essa estratégia proporcionou uma grande popularização do React, conforme demonstrado na Figura 01 a grande evolução de número de *downloads* do React em relação a outras bibliotecas (NEUHAUS, 2018).

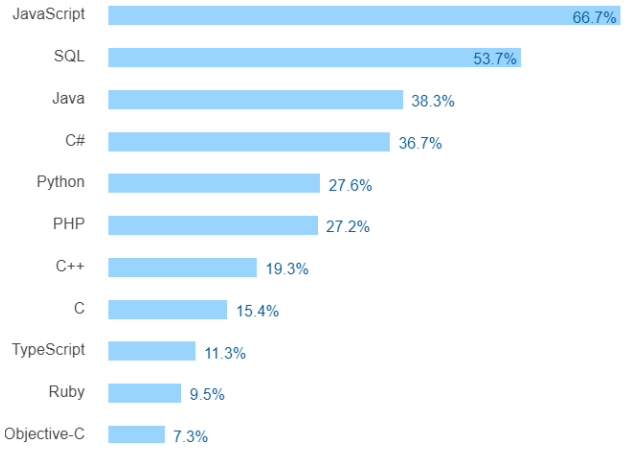
Figura 01: Número de downloads Angular, React, Vue e @Angular/Core.



FONTE: Neuhaus, 2018.

Conforme o *Rank site React* (LIBSCORE, 2019), importantes sites utilizam o React, tais como, Uber, Netflix, Twitter, Pinterest, Reddit, Udemy, Wix, Paypal, Imgur, Feedly, Stripe, Tumblr, Walmart. Outro fator que corrobora com o sucesso da utilização do React, segundo a *Developer Survey Results* 2017 (STACK OVERFLOW, 2018), é que a linguagem de programação do React é o JavaScript, e essa é a linguagem mais utilizada pelos profissionais de desenvolvimento (Figura 2).

Figura 02: Linguagem de programação mais utilizadas por profissionais de desenvolvimento.



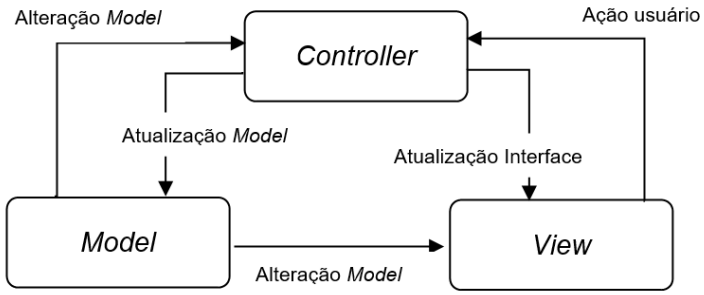
FONTE: Stack Overflow, 2018.

FUNCIONAMENTO DO REACT

Antes de entender o funcionamento do React, é importante saber sobre a *Model-View-Controller* (MVC), que é uma arquitetura de desenvolvimento em camadas, ou seja, onde cada tipo de lógica deve estar localizado na aplicação. A camada *Model* gerencia os dados por meio de instruções de acesso a base de dados e responde às instruções para alterar o estado. A camada *View* gerencia as informações que serão exibidas em tela. A camada *Controller* controla e interpreta as informações recebidas e controlando se será a *View* ou *Model* que será utilizada (BURBECK, 2012).

Durbeck (2012), destaca que a camada *Views* deve gerenciar o espaço na tela e exibir nas formas de textos ou gráficas, já a camada *Controllers* deve garantir que seja interpretada a entrada de comandos, validando os dados antes deles serem exibidos ou manipulados, servindo como intermédio entre a *View* e o *Model*. Dessa forma, quando se muda os dados na *Model*, a *View* tem que ser atualizada, gerenciada e exibida a interface do usuário (textos e gráficos), sendo o reflexo do estado da *Model*, conforme representado na Figura 03.

Figura 03: Objetos utilizados no MVC e suas interações.



FONTE: Elaborado pelos Autores.

Conforme exposto anteriormente, a camada *View* será responsável pela parte de fornecer uma Interface de Usuário (IU) mais responsiva. A responsabilidade adicional do controlador é fundamental devido à necessidade de gerenciar o fluxo de controle inerente à Experiência do Usuário (UX). O React se enquadra na parte da *View*, pois é uma biblioteca de *front-end* utilizada para construir interfaces. Ela fará requisições para *Controllers* de *WebServices* recebendo e enviando dados via *JavaScript Object Notation* (JSON).

Uma vez que a lógica do componente é escrita em JavaScript, e não em *templates*, pode-se passar diversos tipos de dados ao longo da aplicação e ainda manter o estado fora do *Document Object Model* (DOM), melhorando performance e tempo para fazer a transferência de dados do JSON em relação ao XML (Quadro 1).

Quadro 1: Comparação entre tempo de transmissão JSON x XML.

	JSON	XML
Número de objetos	1.000.000	1.000.000
Tempo total (ms)	78.257,90	4.546.694,78
Tempo médio (ms)	0,08	4,55

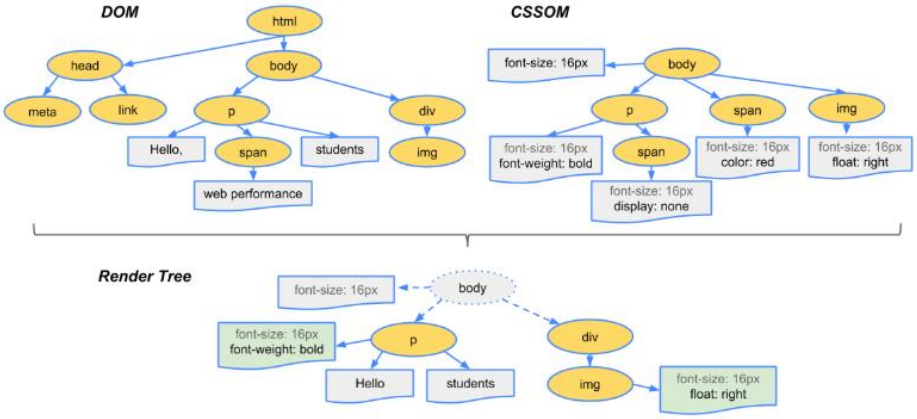
Fonte: Nurseitov at al, 2009, adaptado pelos Autores.

O React é renderizado no *client-side*, o que significa que todo o processamento da página Web é feito no momento do acesso dessa. Segundo Grigorik (2019), engenheiro de desenvolvimento Web na Google, a renderização de uma página Web ocorre da seguinte maneira: “Primeiro, o navegador combina o DOM e o CSSOM em uma árvore de renderização que captura todo o conteúdo visível do DOM e todas as informações de estilo do CSSOM de cada nó”.

Grigorik (2019), explica que na construção da árvore de renderização (Figura 04), o navegador Web realiza o seguinte:

1. Começando na raiz da árvore DOM, percorra cada nó visível;
2. Para cada nó visível, encontre as regras correspondentes do CSSOM correspondentes e aplique-as;
3. Emitem nós visíveis com conteúdo e seus estilos computados;
4. A saída final é uma renderização que contém as informações de conteúdo e estilo de todo o conteúdo visível na tela. (Grigorik, 2019).

Figura 04: Objetos utilizados no MVC e suas interações.



Fonte: Grigorik, 2019.

No exemplo apresentado na Figura 04, a *tag body* da árvore DOM é estilizada com o elemento css *font size* tamanho 16 na árvore CSSDOM, isso implica que cada nó filho da *tag body* herdará essa característica de estilo. No caso de um nó filho já ter aquele mesmo elemento css definido, o que prevalecerá é o valor atribuído a ele mesmo, não o do seu nó pai. Pode-se definir que o navegador Web, interface do usuário, realiza o seguinte: (i) processa toda a marcação HTML, (ii) monta uma árvore DOM, (iii) depois processa toda a marcação CSS, montando uma árvore gêmea ao DOM, o CSSDOM. Combinados todos os elementos homogêneos entre as árvores, o resultado visível é renderizado na tela. Porém, o React utiliza um DOM virtual, o VDOM. No processo chamado "*Reconciliation*", o VDOM é mantido na memória e depois sincronizado com DOM real (GRIGORIK, 2019).

> Essa abordagem habilita a API declarativa de React: Você diz ao React em qual estado deseja que a UI esteja, e garante que o DOM corresponda a esse estado. Isso abstrai a manipulação de atributos, a manipulação de eventos e a atualização manual de DOM que, caso contrário, você teria que usar para criar seu aplicativo. (DOM,2013)

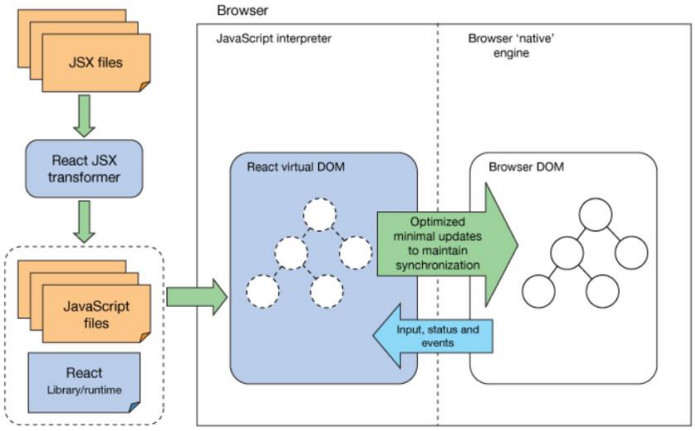
Conclui-se que ocorre pequenos "*refreshes*", e a informação é renderizada na tela, mas somente no nó onde ela é atualizada, o que reflete uma troca de estado de aplicação. Quando essa troca ocorre, a página Web não precisa ser atualizada por

completo, uma vez que a mudança é feita somente naquele nó específico, ignorando o resto do DOM e otimizando o processamento.

“O React também pode ser renderizado no *server-side*, resultando em uma comunicação interna, operável entre os dois lados” (DANIELSSON, 2016, p. 8), sendo auxiliado pelo *Server-Side Rendering* (SSR), que monta e disponibiliza um componente para resposta antes mesmo da requisição do *client-sid*e.

O React opera no paradigma de linguagem declarativa, o que possibilita maior legibilidade e facilidade em dar manutenção no código. Isso porque o React faz uso do compilador JSX, que é essencialmente Javascript (Figura 5), porém voltado para uma linguagem de componentes, o que facilita não só a organização, mas também a visualização da montagem dos componentes na hora de desenvolver.

Figura 05: Funcionamento do React.



Fonte: Li (2016).

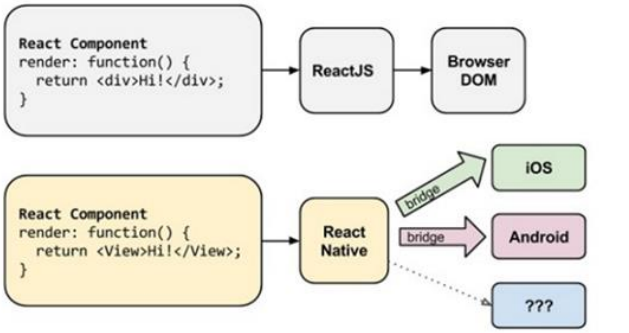
A figura 5 demonstra resumidamente o funcionamento do React, conforme exposto neste tópico, principalmente a utilização da biblioteca React para geração de JavaScript interpretáveis na camada *React Virtual DOM*, e comunicação entre essa camada e a camada *Browser DOM*.

REACT NATIVE

Segundo Occhino (2015), engenheiro diretor do grupo React do Facebook©, o lema: “Aprenda uma vez, escreva em qualquer lugar”, significa que o desejo de fazer com que a experiência que os desenvolvedores têm para desenvolver para Web com o *ReactJS*, seja a mesma para desenvolver mobile (Figura 06). Atualmente, há uma infinidade de opções para se desenvolver o mesmo projeto para diferentes plataformas, porém, cada aplicação tem suas peculiaridades e capacidades.

Seguindo o contexto acima, em 2015 o Facebook© lançou o React Native, *framework* para desenvolvimento mobile, inicialmente com suporte para IOS, e depois para Android. Destaca-se que basta desenvolver um app, que o mesmo poderá ser executado nas duas plataformas, Android ou IOS (Figura 06).

Figura 06: React JS e React Native.



Fonte: Eisenman (2016).

O React Native tem uma sintaxe semelhante à do ReactJS, pois também utiliza o JSX. Porém, seus elementos não são renderizados da mesma maneira. No React Native, os elementos são emulados de forma nativa, utilizando o *JavaScript Core*, como uma ponte entre o JSX e as linguagens. Essa ponte abstrai uma camada de aplicação que possibilita executar API de renderização do Java e do Objective-C.

A *framework* possui três *threads* principais nas quais os diferentes eventos dentro das aplicações são acionados. A *Shadow Queue*, onde o *layout* é manipulado, *thread* principal, onde ocorre todo o processo de renderização da interface e a *thread* Javascript onde os *scripts* serão executados (DANIELSSON, 2016).

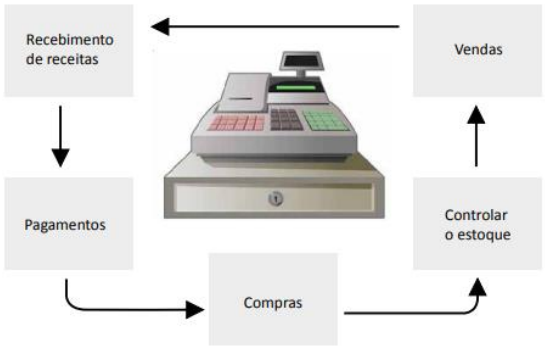
A licença do React Native é do tipo MIT, ou seja, licença permissiva que faz referência ao uso, redistribuição e modificação do software tanto para softwares livres como para software proprietário. A única condição implícita a essa licença é que ela deve ser aplicada a todas as cópias e componentes do software que serão posteriormente utilizadas e/ou vendidas.

O React e o React Native estão disponíveis para livre acesso no site do GitHub©. Também contam com site React, aonde se pode obter gratuitamente o tutorial principal, documentação de apoio, suporte, blog e comunidades, que facilitam o uso dessas soluções.

FLUXO DE CAIXA

Segundo Andrade (2018), *Controller* da ContaAzul, “Fluxo de Caixa é o movimento de entradas e saídas de dinheiro do caixa da empresa, ou seja, o que você recebe e o que paga em seu negócio”, observe a Figura 07. Entende-se que tudo que entra e sai do caixa faz parte do fluxo de caixa, esse histórico pode revelar como está a saúde financeira de uma empresa, gerando dados para análise e apoio a gestão e tomada de decisões de um negócio.

Figura 07: Fluxo de Caixa.



Fonte: SEBRAE-SP (2016, p. 10).

Na movimentação de caixa para formação do Fluxo de Caixa, todas as operações devem ser registradas, desde notas fiscais saída (vendas), recebimento (dinheiro, cheque, cartão), pagamentos diversos, compras (entradas) e movimentação de estoque (Figura 07). A partir do momento que o registro é feito corretamente, um cenário sobre toda a movimentação financeira da empresa é estabelecido. O principal objetivo do fluxo de caixa é assegurar que o negócio da empresa esteja em equilíbrio financeiro, onde o quanto sobra em recursos próprios tende a aumentar e o quanto é descontado do saldo total, tende a diminuir, resultando em um cenário positivo ou negativo (SEBRAE-SP, 2016).

CONSTRUÇÃO DO APP FLUXO DE CAIXA

SOBRE A RELEVÂNCIA

Segundo Gonçalves (2019), o IBGE constatou que 13,4 milhões de pessoas estavam em busca de trabalho no primeiro trimestre de 2019, que resultou na taxa de desocupação de 12,7% (Figura 08). Observa-se que a taxa de desocupação no Brasil corresponde à taxa de desemprego da população economicamente ativa.

Figura 08: Taxa de desocupação Brasil.

Fonte: Gonçalves (2019)

Frente a esse cenário, brasileiros na situação de desocupados procuram saídas para se sustentarem, sendo comum essas pessoas adentrarem no comércio ambulante como uma alternativa para as dificuldades de conseguir uma ocupação, em especial no segmento de refeição e lanches.

SOBRE A IDEALIZAÇÃO

O autor Caio Frias, ao assistir o vídeo do mochileiro Eliézer Tymniak, que apresenta a venda de brigadeiros como uma forma de gerar renda no “COMO FAZER 1000 REAIS EM 1 SEMANA!” (TYMNIAK, 2019), em que é apresentado como é possível juntar dinheiro para diversos fins, teve a ideia de como realizar na prática o que foi apresentado no vídeo. O autor e um colega montaram um negócio informal para fabricação e vendas de brigadeiros caseiros em eventos e parques. No primeiro evento realizado, todos os custos foram recuperados com as vendas dos brigadeiros, obtendo margem de lucro e capital para investimento no próximo evento. Porém, o resultado financeiro, devidamente controlado, não foi o único ponto importante dessa experiência, o relacionamento com os demais comerciantes ambulantes também teve destaque.

O autor Caio percebeu que nesse ambiente é comum as conversas entre as pessoas, principalmente para contar os diversos motivos de estarem como ambulante e como operavam seu negócio. O autor também percebeu a falta de organização e conhecimento de gestão desses ambulantes para com o próprio negócio frente a variedade de produtos oferecidos. Por percepção, o autor identificou que o nível de instrução da maioria é de baixa escolaridade, entendendo assim, o motivo de não praticarem nenhum controle matemático do que era gasto e do era recebido, essencial para ter uma informação se o negócio gera lucro ou prejuízo. O autor Caio percebeu que uma grande parte desses ambulantes empreenderam sem ter a mínima noção de gestão, apenas foram levados pela força da necessidade de se sustentar.

Diante do cenário narrado, o autor Caio deparou-se com a questão em identificar o quanto os comerciantes ambulantes movimentavam em montantes financeiros, e se eles também gerenciam seus negócios da melhor maneira possível, dessa forma, a resposta poderia sensibilizar esses ambulantes a praticar no mínimo o Fluxo de Caixa para avaliar financeiramente seu negócio, ou até mesmo, estimular apoios de organizações governamentais, educacionais ou sociais na capacitação em gestão negócios para esses ambulantes.

No intuito de responder à questão acima, o autor teve a ideia em construir um aplicativo móvel com a finalidade de identificar o perfil de quem exerce esse tipo de atividade e entender como essa ambulante trabalha com a questão do gerenciamento do próprio negócio, mensurando a sua saúde financeira. Os dados a serem captados terão utilidade para determinar qual seria a renda média de quem exerce esse tipo de atividade, quanto dinheiro é movimentado, os produtos que são mais e menos vendidos, picos de vendas, rentabilidade, e os principais fatores que podem ser abordados para determinar se o negócio é próspero ou não.

O “*perfilamento*” do comerciante ambulante, que contribuir com a pesquisa, será com base no preenchimento de questionário contendo o seguinte:

- Sexo;
- Idade;
- Número de filhos;
- Possui veículo próprio?
- Possui imóvel próprio?
- Com qual regularidade você faz vendas? (Dias da semana e horas por dia)
- Você tem algum planejamento para fazer suas vendas? Se sim, como ele é feito?
- Escolaridade;
- Já trabalhou anteriormente?
- Motivação principal para engajar nessa atividade.

Espera-se que com o resultado da análise dos dados coletados obter o nível de compreensão sobre quem faz parte do grupo de comerciante ambulante, suas motivações e necessidades. Em contrapartida, e por motivação, pela participação da pesquisa, será disponibilizado um app de controle de Fluxo de Caixa para o comerciante ambulante, que tem a finalidade de iniciar o ambulante na gestão do Fluxo de Caixa e coletar dados quantitativos para analisar o comportamento financeiro do grupo, possibilitando ações de orientação e melhorias.

DECISÃO PELO DESENVOLVIMENTO DO APP

Percebeu-se que os comerciantes ambulantes, observado neste artigo, não utilizam controles de suas vendas, apenas uma pequena parte deles realizam o controle de estoque, porém de forma manual. A justificativa dada pelos ambulantes seria o pouco tempo no ato da transação comercial (atendimento) com o cliente.

Sistemas de gestão são mais comuns em sites e programas pagos, geralmente feitos por encomenda de acordo com as necessidades do empreendimento. O Conta Azul, por exemplo, é um software de gestão financeira *online* que tem funcionalidades para praticamente todas necessidades de um negócio, porém, o preço não é tão acessível e rentável para esses ambulantes, e nem todos tem condições de comprar e aprender a manusear um computador.

Entende-se, então, que o app seja a solução mais viável a partir do momento que o uso de *SmartPhone* tornou-se comum e acessível, uma vez que, segundo a Anatel (2019) “o Brasil registrou um número de 231.827.959 linhas móveis no mês de dezembro de 2018”. Com base no exposto, infere-se que grande parte dos comerciantes ambulantes utiliza essa tecnologia de comunicação.

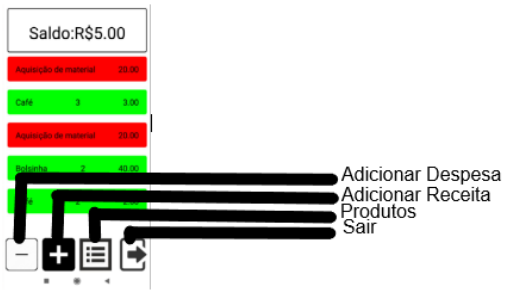
PROTÓTIPO DO APP DE FLUXO DE CAIXA

Na construção do aplicativo (app) de Fluxo de Caixa, versão protótipo, foram aplicados o *React Native* e *Firebase*, banco de dados em tempo real disponibilizado pelo Google. As imagens e textos referentes as suas respectivas funcionalidades estão em estágio inicial e podem ser tanto alteradas de acordo com o andamento da pesquisa quanto trocadas por outras. Esse app está na fase de preparação para os testes, e pode ser obtido gratuitamente no Github, acessando o endereço https://github.com/Shouganaii/fluxo-de-caixa.

TELA PRINCIPAL

A tela principal do app Fluxo de Caixa contém dois *cards* importantes, o primeiro mostra o balanço atual do saldo total do negócio, que muda conforme vendas e compras são feitas. O segundo *card* lista todas as transações realizadas, sendo que cada transação é diferenciada pela cor de fundo, Vermelho significa uma saída, enquanto Verde significa uma entrada. No extremo inferior da tela, há uma barra de acesso que nos direciona para outras *Views* do *app*, conforme Figura 09.

Figura 09: Tela principal do app Fluxo de Caixa.



ADICIONAR DESPESA

Na *View* Adicionar despesas (Figura 10), pode-se realizar as saídas do Fluxo de Caixa. O campo “Valor” indica o quanto foi gasto e o campo “Descrição” indica no que o valor foi gasto. Dentro do app, o valor que foi gasto será subtraído do saldo total, tanto a descrição quanto o valor serão adicionados ao histórico.

Figura 10: Tela Adicionar despesas do app Fluxo de Caixa.

ADICIONAR RECEITA

Na *View* Adicionar receita (Figura 11), pode-se realizar as saídas do Fluxo de Caixa. O *Picker* onde está escrito “Café”, na representação da Figura 11, comporta a lista de produtos cadastrados pelo colaborador e seus respectivos valores. No campo Quantidade é possível informar quantos produtos estão sendo vendidos. Essa quantidade é multiplicada pelo valor do produto selecionado, e caso a venda seja consumada, o resultado dessa multiplicação será somado ao montante total do saldo.

Figura 11: Tela Adicionar receita do app Fluxo de Caixa.

SEUS PRODUTOS

O cadastro de produtos é realizado na *View* Seus produtos (Figura 12). Os produtos já existentes são apresentados nessa *view* e clicando neles, há a possibilidade de alterar o valor e/ou o nome do que foi selecionado. Também é possível adicionar novos produtos. Os produtos aqui adicionados são listados no *Picker* da Adicionar receita.

Figura 12: Tela Seus produtos do app Fluxo de Caixa.

SAIR

Ao clicar em Sair, o usuário será desconectado e redirecionado para a tela apresentada na Figura 13.

Figura 13: Tela Sair do app Fluxo de Caixa.

LOGIN E CADASTRO DE LOGIN

A *View* Login tem a finalidade de realizar a conexão de segurança do usuário. A *View* de Cadastro de Login tem a finalidade de cadastrar o usuário do app. Essas *Views* têm o mesmo *layout* e campos, conforme Figura 14. O acesso é pela autenticação do *Firebase*. É necessário que o e-mail informado seja real e válido para que haja uma posterior recuperação de senha. A recuperação de senha, login e cadastro estão desenvolvidas.

Figura 14: Tela Login e Cadastro de Login do app Fluxo de Caixa.

CONSIDERAÇÕES FINAIS

Desde a identificação do problema até a finalização da versão protótipo do app Fluxo de Caixa foram diversos desafios. O primeiro desafio foi em observar e entender o processo de trabalho dos comerciantes ambulantes, sua realidade e seus problemas. O segundo desafio foi a identificação de proposta de solução para o problema de controle financeiro do negócio do ambulante. O terceiro desafio foi em buscar o conhecimento em Fluxo de Caixa para ajudar na proposta de solução. O quarto desafio foi em pesquisar soluções de tecnologia de informação para prover o desenvolvimento de app de Fluxo de Caixa para ambulante.

Nesse contexto de desafios, este artigo apresentou o relato sobre a primeira fase da construção de uma solução em tecnologia *mobile* - app Fluxo de Caixa - e a proposta de pesquisa para avaliar o público alvo para esse app. Conclui-se que para o cenário analisado, a solução tecnológica aplicável por questões de baixos custos, continuidade, qualidade, produtividade, flexibilidade e facilidade do projeto de desenvolvimento de software, seriam o uso da biblioteca React, do *framework* React Native e do banco de dados Firebase para a construção do app.

A segunda fase seria a realização dos testes para validação do app, e realização da pesquisa com os comerciantes ambulantes. Na terceira fase, dar-se-á com a operação do app Fluxo de Caixa pelo público alvo, quando será realizada uma coleta de dados sobre as movimentações do público alvo. Essas segunda e terceira fases, quando concluídas, serão temas para a elaboração de artigo.

REFERÊNCIAS

ANDRADE, Marcio Roberto. O que é fluxo de caixa e como ele pode ajudar sua empresa, 2018. Disponível em: <https://blog.contaazul.com/o-que-e-fluxo-de-caixa/>. Acesso em: 01 mai. 2019.

ANATEL. Brasil registra 231,8 milhões de linhas móveis em novembro. Disponível em: <http://www.anatel.gov.br/institucional/noticias-destaque/2175-brasil-registra-231-8-milhoes-de-linhas-moveis-em-novembro >. Acesso em: 02 mai. 2019.

BURBECK, Steve. *Applications Programming in Smalltalk*-80 (*TM*): *How to use Model-View-Controller* (*MVC*), 2012. Disponível em: <http://www.dgp.toronto.edu/~dwigdor/teaching/csc2524/2012_F/papers/mvc.pdf>. Acesso em: 05 mai. 2019.

DANIELSSON, William. *React Native application development*. **Linköpings universitet**, Swedia, 2016.

DOM. *Virtual DOM and Internals,* ***Facebook React Development Team***, 2013. Disponível em: <https://reactjs.org/docs/faq-internals.html>. Acesso em: 30 abr. 2019.

EISENMAN, *Bonnie. Writing Cross-Platform Apps with React Native*, 2016. Disponível em: <https://www.infoq.com/articles/react-native-introduction >. Acesso em: 01 mai. 2019.

GONÇALVES, Adriana; PERET, Luiz Eduardo. Desemprego sobe para 12,7% com 13,4 milhões de pessoas em busca de trabalho, 2019. Disponível em: <https://agenciadenoticias.ibge.gov.br/agencia-noticias/2012-agencia-de-noticias/noticias/24283-desemprego-sobe-para-12-7-com-13-4-milhoes-de-pessoas-em-busca-de-trabalho>. Acesso em: 04 mai. 2019

GRIGORIK, Ilya. *Render-tree Construction, Layout and Paint*. Disponível em: <https://developers.google.com/web/fundamentals/performance/critical-rendering-path/render-tree-construction?hl=en >. Acesso em: 30 abr. 2019.

LI, Sing. React: crie componentes de UI de alto desempenho, **IBM**, 2016. Disponível em: <https://www.ibm.com/developerworks/br/library/wa-react-intro/index.html>. Acesso em: 30 abr. 2019.

LIBSCORE. *Rank site React*, 2019. Disponível em: <http://libscore.com/#React>. Acesso em: 05 mai. 2019.

NEUHAUS, Jens. *Angular vs. React vs. Vue*: A 2017 comparison, **unicorn.supplies**, 2018. Disponível em: <https://reactjs.org/tutorial/tutorial.html#what-is-react>. Acesso em: 05 mai. 2019.

NURSEITOV, Nurzhan et al. *Comparison of JSON and XML data interchange formats: a case study*. **Caine**, v. 9, p. 157-162, 2009.

OCCHINO, Tom. *React Native: Bringing modern web techniques to mobile*, 2015. Disponível em: <https://code.fb.com/android/react-native-bringing-modern-web-techniques-to-mobile/>. Acesso em: 22 abr. 2019.

SEBRAE-SP. Curso Fluxo de Caixa, **Sebrae**, SP. Disponível em: <https://bibliotecas.sebrae.com.br/chronus/ARQUIVOS_CHRONUS/bds/bds.nsf/a54c29c120d08789e9369c2da15aa9e1/$File/9880.pdf >. Acesso em: 01 mai. 2019.

STACK OVERFLOW. *Developer Survey Results* 2017, 2018. Disponível em: <https://insights.stackoverflow.com/survey/2017#technology-_-frameworks-libraries-and-other-technologies>. Acesso em: 05 mai. 2019.

TYMNIAK, Eliézer. COMO FAZER 1000 REAIS EM 1 SEMANA! Dinheiro rápido e fácil sozinho - 4 MIL REAIS POR MÊS. Disponível em: <https://www.youtube.com/watch?v=HFQ6m2tKoVg >. Acesso em: 04 mai. 2019.

TUTORIAL. Tutorial: ***Intro to React***, 2019. Disponível em: < https://reactjs.org/tutorial/tutorial.html#what-is-react>. Acesso em: 05 mai. 2019.