GABY - TERMINAL DE CONSULTA DE PREÇO

Vitor Mateus Fank Teixeira[1]

Vanessa Lago Machado[2]

José Antônio Oliveira de Figueiredo[3]

RESUMO

Neste trabalho é apresentado o desenvolvido de um sistema para realizar consultas de preços por meio de dispositivos móveis. A principal motivação para a criação desse aplicativo foi atender uma demanda em potencial não explorada, tendo em vista que os Terminais de Consulta de Preço (TCP) disponíveis no mercado possuem um alto custo de aquisição, além de possuírem restrição devido ao fato de tratarem-se de modelos fixos. A fim de criar uma alternativa a esses modelos de TCP, foi desenvolvido um aplicativo para dispositivos móveis com tecnologia híbrida, totalmente interligado por um banco de dados em *cloud*, denominado Firebase. O aplicativo apresenta para o usuário uma tela que permite escanear o código de barras do produto que deseja saber o preço, com uso da câmera do smartphone. Com isso, o servidor recebe o código lido e retorna os dados do produto cadastrado, quando encontrado. Após implementação e realização dos testes com o aplicativo foi possível verificar sua potencialidade de uso, além de demonstrar seu potencial para novas funcionalidades.

Palavras-chave: TCP. Aplicativo. Tecnologia híbrida. Firebase.

1 INTRODUÇÃO

Cada vez mais as pessoas buscam agilidade e facilidade, segundo dados do Estadão, o Brasil conta com 168 milhões de dispositivos, gerando a média de 1.6 smartphones para cada habitante. Com essa realidade é previsto que alguns hábitos sejam alterados, como o fato de consultar preços de produtos por meio de outras formas, além do tradicional Terminal de Consulta de Preço (TCP).

Os TCP mais populares na região visam atender a demanda da falta de informação dos preços nas embalagens dos produtos, como por exemplo, os terminais da marca GERTEC. Ainda, destaca-se que, até o momento, não foram

[1] Aluno do curso de Tecnologia de Sistemas para Internet no Instituto Federal de Educação, Ciência e Tecnologia Sul-rio-grandense de Passo Fundo (IFSul). E-mail: v_m_ft@hotmail.com
[2] Orientadora, professora do Instituto Federal de Educação, Ciência e Tecnologia Sul-rio-grandense de Passo Fundo (IFSul). E-mail: vanessa.machado@passofundo.ifsul.edu.br
[3] Coorientador, professor do Instituto Federal de Educação, Ciência e Tecnologia Sul-rio-grandense de Passo Fundo (IFSul). E-mail: jose.figueiredo@passofundo.ifsul.edu.br

encontradas soluções móveis para consulta de preços. Visando atender essa demanda, o presente trabalho visa o desenvolvimento de um sistema de consulta de preços mobile, o qual conta com um banco de dados em cloud.

A divisão de seções deste artigo seguiu a seguinte ordem: na seção 2 é apresentado o referencial teórico, o qual refere-se aos TCP, código de barras adotado no Brasil, persistência de dados (banco de dados Firebase) e Ionic, tecnologia utilizada para desenvolvimento do aplicativo. A seção 3 apresenta os materiais e métodos que foram utilizados para o desenvolvimento. Na seção 4 são apresentados os resultados obtidos após o desenvolvimento, além dos resultados dos testes do aplicativo realizado com alguns usuários, e a seção 5 refere-se a conclusão sobre o trabalho desenvolvido.

2 REFERENCIAL TEÓRICO

Nesta seção serão apresentados os fundamentos sobre um aplicativo com tecnologia híbrida, além da tecnologia utilizada por terminais de consulta de preço atualmente.

2.1 TERMINAIS DE CONSULTA DE PREÇO

O Terminal de Consulta de Preço (TCP) pode ser definido como um “dispositivo eletrônico, dotado de visor e leitor de código de barras, em alguns casos com memória própria, que possibilita ao cliente, através da passagem do código de barras no leitor, a visualização do preço do produto no visor” (CLARO, 2013).

Conforme Gertec (2018), os TCP’s tratam-se de computadores que permitem consulta à base de dados de um sistema ou centro de informação, sendo que uma unidade de acesso distante permite o envio de informações de ordens e a recepção de mensagens.

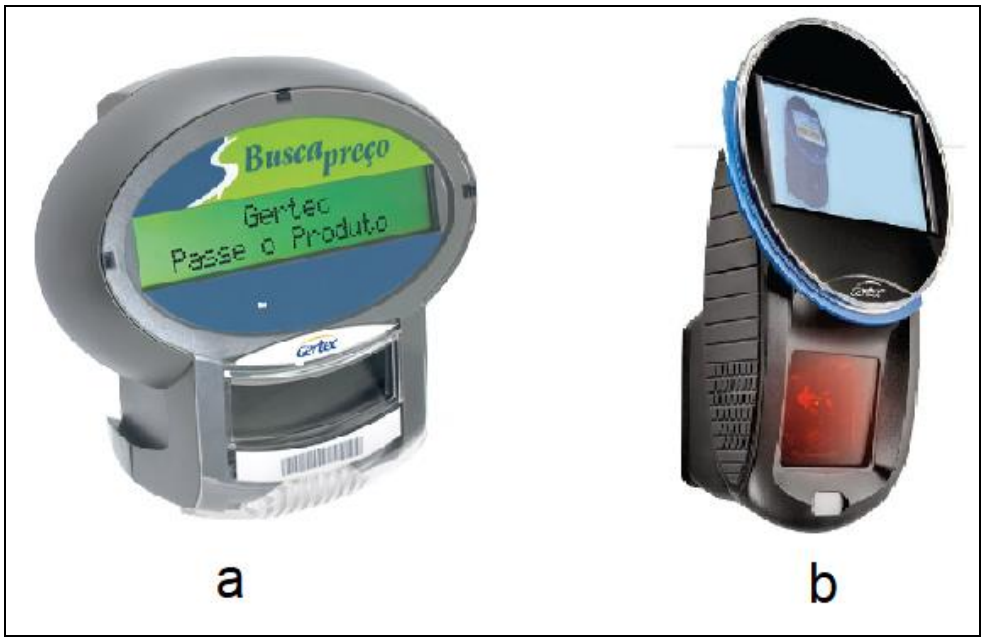
O Decreto nº. 5.903, de 20 de dezembro de 2006, que regulamenta a Lei nº 10.962, de 11 de outubro de 2004, estabelece a obrigatoriedade dos terminais de consulta de preços em estabelecimentos que possuem produtos com código de barras. A referida lei prevê que o cliente não deve se deslocar mais de 15 metros para ter acesso a um terminal. Além disso, de acordo com o seu artigo 7º, tem-se

que: “Os leitores óticos deverão ser dispostos na área de vendas, observada a distância máxima de quinze metros entre qualquer produto e a leitora ótica mais próxima”. (BRASIL, 2006) Assim, verifica-se a obrigatoriedade da implantação dos TCP’s no comércio varejista.

É possível identificar diferentes tipos e modelos de terminais, em que os modelos mais simples como exemplo ilustrado na Figura 1 (a) ajudam a suprir a necessidade do comércio, principalmente devido a seu baixo custo de aquisição, possibilitando somente visualizar o preço e uma descrição básica do produto, assim, esses modelos são indicados para locais que precisam de um número maior de terminais.

Há ainda os terminais com interface interativa, com a tecnologia de tela *touchscreen*, que pode transmitir vídeos ou imagens, proporcionando marketing dos produtos a cada consulta, possibilitando maior quantidade de funcionalidades ao usuário, como exemplo ilustrado na Figura 1 (b).

Figura 1 - Terminais de Consulta de Preços comercializados pela empresa Gertec.



Fonte: Do autor (Adaptada de Gertec, 2018).

2.2 CÓDIGO DE BARRAS LINEAR

Meados da década de 30, tecnologias com a mesma finalidade do código de barras linear começavam a ser estudadas, desenvolvidas e testadas. Conforme Milies (2008), testes com cartões perfurados começaram a ser realizados em um projeto na universidade de Harvard com o objetivo de identificar os produtos, o que influenciou na criação da máquina registradora. Com o surgimento da máquina registradora cada item passou a ganhar uma etiqueta adesiva com seu preço, possibilitando assim o registro dos valores das mercadorias e o cálculo do valor das transações de venda de determinado estabelecimento.

Em 20 de outubro de 1949 foi protocolado um pedido de patente para “Classificação de Aparatos e Métodos”, que descrevia os padrões de impressão. Após três anos, em 7 de outubro de 1952, foi emitida a patente americana #2.612.994 aos inventores Woodland e Silver para o primeiro sistema para codificação automática de produtos, que foi a primeira versão do código de barras (JUNIOR, 2010).

2.2.1 Características do padrão EAN-13, adotado no Brasil.

Os números que constam nos códigos de barras não têm uma importância tão relevante quanto às informações que comerciário e fornecedor irão inserir em seus sistemas. O que deve ser observado e respeitado é a autenticidade da ordem dos números que deve ser exclusiva. Existem vários tipos de códigos de barras, porém um dos mais utilizados é o EAN-13[4], padrão adotado no Brasil. Essa adoção de um padrão garante a uniformidade no sistema, o que evita confusões e divergências. Dessa forma, a Figura 2 demonstra como é montado um código de barras padrão EAN-13, Conforme Vanz e Figueiredo (2012).

[4] European Article Numbering, padrão que estabelece diretrizes referentes à implementação do sistema de código de barras no Mercado Europeu, e posteriormente no mercado mundial.

Figura 2 - Características do código de barras EAN-13

Fonte: Do autor

Segundo a GS1 Brasil, o prefixo do país é composto pelos três primeiros números do código, dessa forma esses dígitos indicam o país no qual o produto foi cadastrado, independente do país de fabricação. Cada país tem uma combinação própria, emitida pela entidade autorizada, no caso do Brasil o prefixo é o 789.

Todos os produtos produzidos pela mesma empresa têm o mesmo código da empresa, também chamado de código de fabricante. Assim, cada empresa possui um código, o qual trata-se de uma sequência única que identifica a indústria dona da marca do produto. Tal código pode variar de quatro a sete dígitos. Já em relação ao código do produto, refere-se a um código único de cada produto, composto geralmente de cinco dígitos dado pelo fabricante, podendo ter de dois a cinco dígitos de acordo com a quantidade de itens diferentes da empresa; Por fim, o dígito verificador, o qual comprova que a leitura foi feita corretamente, garantindo que o resultado lido seja realmente o correto.

2.3 PERSISTÊNCIA DE DADOS

A persistência de dados consiste no armazenamento confiável e coerente das informações em um Sistema de Gerenciamento de Banco de Dados (SGBD). Uma das soluções para fazer a persistência dos dados é a utilização do banco de dados Firebase.

2.3.1 Firebase

Desenvolvido pela empresa Google, o Firebase é um conjunto de produtos distribuído gratuitamente, com um limite de utilização. Entre esses produtos, existem serviços de hospedagem, armazenamento em nuvem e o banco de dados. (FIREBASE, 2017)

Segundo a documentação disponível no site do fabricante:

> O Firebase Realtime Database é um banco de dados hospedado na nuvem. Os dados são armazenados como JSON e sincronizados em tempo real com todos os clientes conectados. Quando você cria apps em plataformas cruzadas com nossos SDKs para iOS, Android e JavaScript, todos os clientes compartilham uma instância do Realtime Database e recebem automaticamente atualizações com os dados mais recentes (Firebase, 2017).

A utilização gratuita permite até 100 acessos simultâneos. A ferramenta permite que, com poucas linhas de código, o banco de dados seja adicionado a diferentes plataformas, como aplicações web, Android e iOS, permitindo que todas as plataformas se conectem ao mesmo banco de dados, sem requerer conhecimentos sobre a infraestrutura do sistema.

2.4 IONIC

O Ionic foi criado por Max Lynch, Ben Sperry e Adam Bradley da Drifty Co. em 2013. Os produtos anteriores da Drifty Co. incluem ferramentas de construção de interfaces, como Codiqa e Jetstrap, os quais tratam-se de ferramentas *drag-and-drop*[5] usando jQuery Mobile e Bootstrap (GRILLO, 2015). Buscando atender a mais clientes, a equipe decidiu construir sua própria estrutura que se concentraria no desempenho e seria modelada com padrões da web modernos.

Hoje o Ionic é um completo SDK (*framework*) de código aberto para o desenvolvimento de aplicativos móveis híbridos. Construído no topo do AngularJS e Apache Cordova[6], o Ionic fornece ferramentas e serviços para desenvolver

[5] Ação de clicar em um objeto virtual e "arrastá-lo"

[6] Framework que permite desenvolvimento de aplicações multiplataforma.

aplicativos móveis híbridos usando tecnologias da Web como CSS, HTML5 e Sass[7]. Os projetos podem ser criados com essas tecnologias web e distribuídos por lojas de aplicativos nativos, tais como Play Store e Google Play, o que garante a disponibilidade nos principais sistemas operacionais móveis existentes (IONIC FRAMEWORK).

Ionic nada mais é do que um conjunto de componentes e outros *frameworks*, sendo esses componentes: Cordova, para fazer a Integração com recursos nativos dos dispositivos; AngularJS[8], responsável pela criação da parte Web do App; Ionic Module e o Ionic CLI, ferramentas e componentes disponibilizados pelo *framework*.

3 METODOLOGIA - MATERIAIS E MÉTODOS

Neste estudo foi desenvolvido um aplicativo mobile com foco no comércio varejista; A aplicação foi desenvolvida com o intuito de ser uma alternativa aos terminais fixos de consulta de preço, devido ao fato de possuírem um alto custo para aquisição, além desse modelo de consulta dificultar, ou em muitas vezes inviabilizar, a consulta do valor de um produto de grande porte, visto que o mesmo deverá ser deslocado até o leitor.

O aplicativo GABY é multiplataforma e utiliza tecnologia híbrida, o qual foi desenvolvido usando as tecnologias HTML5, CSS e Javascript, por meio do *framework* Ionic e o Apache Cordova, pacotes do NodeJS[9], o qual foi utilizado a versão node-v6.11.2-x64 e NPM [10]3.10.10. Em relação ao Ionic trabalhado, foi utilizado a versão 3.19.1, e em relação ao cordova foi utilizado a versão 8.0.0. Além disso, foi utilizado o Android SDK 25.5.5 e angularfire2 5.0.0-rc.6. Como ambiente de desenvolvimento foi utilizado a IDE Visual Studio.

O aplicativo utiliza um banco de dados em *cloud*, denominado Firebase, possibilitando que as consultas sejam realizadas de qualquer lugar e em qualquer horário.

---

[7] Linguagem de extensão CSS.

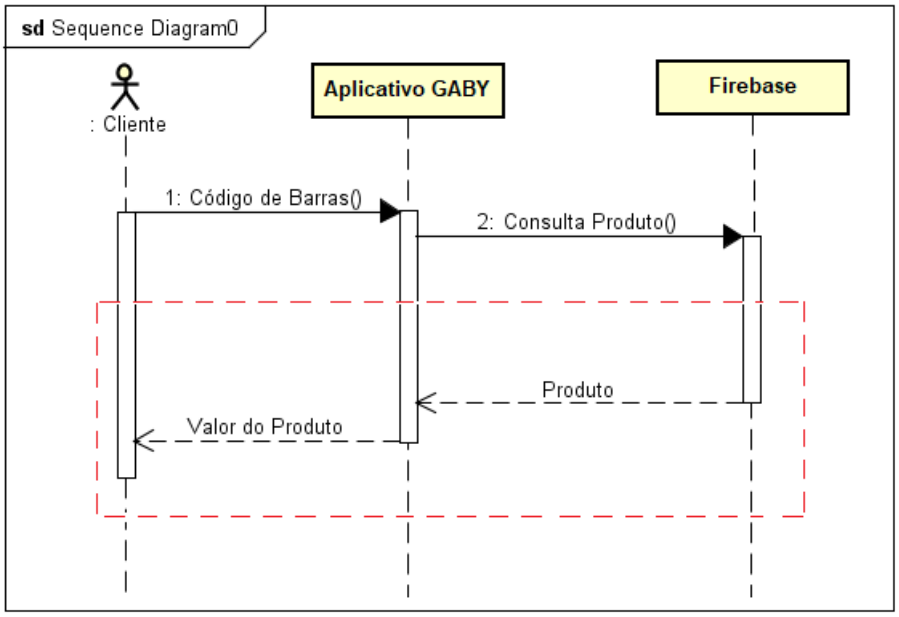
[8] Framework em javascript

[9] Interpretador de código JavaScript

[10] Node Package Manager (Gerenciador de Pacotes do NodeJS).

Assim, foi previsto o fluxo da consulta de preço via aplicativo, o qual encontra-se ilustrado na Figura 3, por meio do diagrama de sequência.

Figura 3 - Diagrama de sequência: fluxo da consulta de preço realizada pelo usuário no Aplicativo.



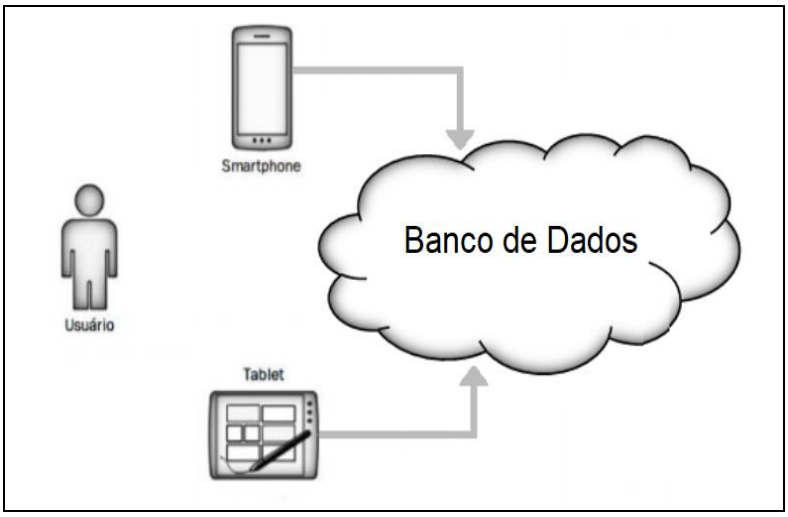
Fonte: Do autor.

O diagrama de sequência (Figura 3) demonstra o fluxo do sistema e o processo do sincronismo dos dados. Assim, o aplicativo faz a comunicação com o SGBD, o qual realiza a comunicação de retorno à aplicação. Detalhadamente é possível verificar que para utilização do aplicativo o usuário realiza a leitura do código de barras do produto, utilizando a câmera do dispositivo, como um TCP, assim o código lido será enviado para o banco de dados Firebase, a fim de verificar a existência do registro do mesmo. Se o produto for encontrado o sistema retornará os dados do produto, e exibirá na tela os respectivos dados previamente cadastrados. Concomitante a esse fluxo, além dos dados do produto serem apresentados ao usuário, são armazenados em um banco de dados local, denominado *storge*, para manter um histórico das últimas consultas realizadas. Por outro lado, caso não seja encontrado o registro do produto, a aplicação está programada para exibir uma mensagem *default*.

4 RESULTADOS OBTIDOS

O aplicativo desenvolvido pode ser instalado em smartphone ou tablet, como visto na Figura 4.

Figura 4 - Modelo de funcionamento do sistema desenvolvido



Fonte: Do autor.

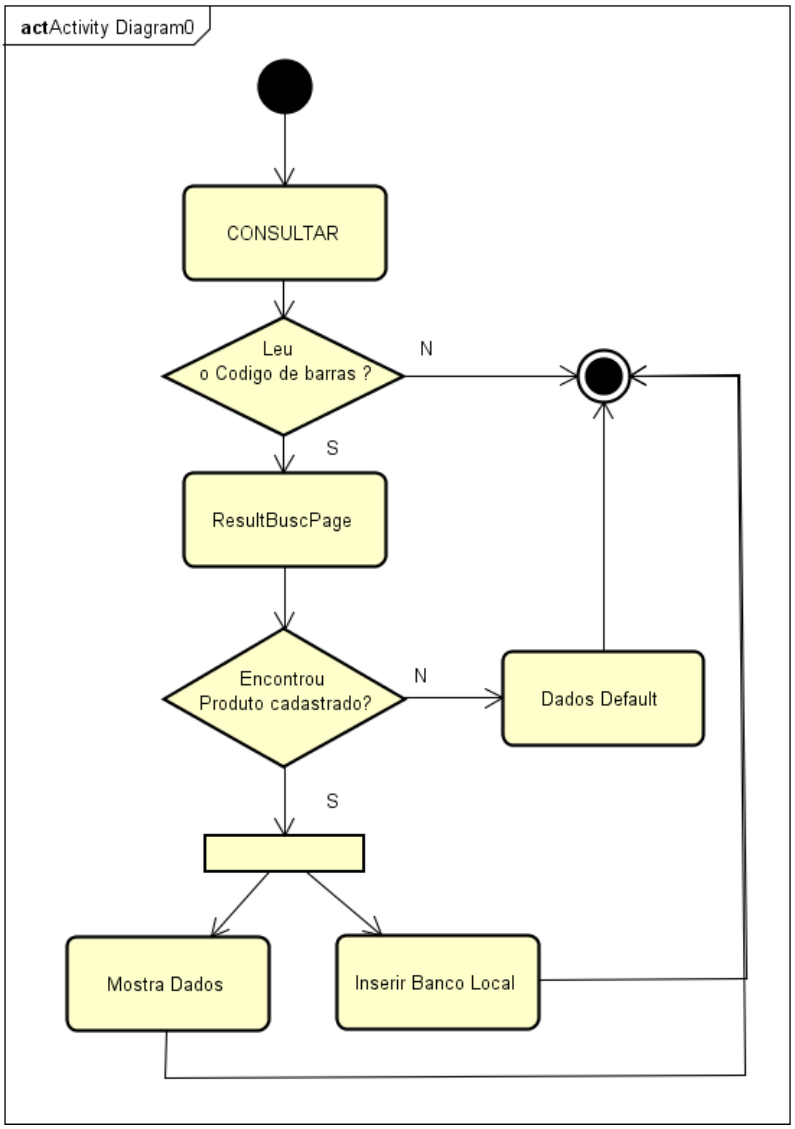
Para realizar a consulta de preço dos produtos, o aplicativo deverá estar com permissão de acesso à câmera, possibilitando tais consultas. O aplicativo é composto ao todo por três telas, em que é possível observar na Figura 5 a tela da página inicial do aplicativo (Home).

Figura 5 - Gaby - Fragmento da Tela Principal

Fonte: Do autor.

A tela principal do aplicativo (Home) apresenta uma interface simples, contendo um botão [CONSULTAR] para realizar a leitura do código de barras do produto; Ao ser acionado o botão é disparada uma função (onGetBarcode), a qual utiliza o *plugin* BarcodeScanner, em que uma visualização da câmera é aberta e permite escanear automaticamente um código de barras, tal função é melhor detalhada na Figura 6.

Figura 6 - Gaby - Diagrama de Atividades representando o fluxo de uma consulta.



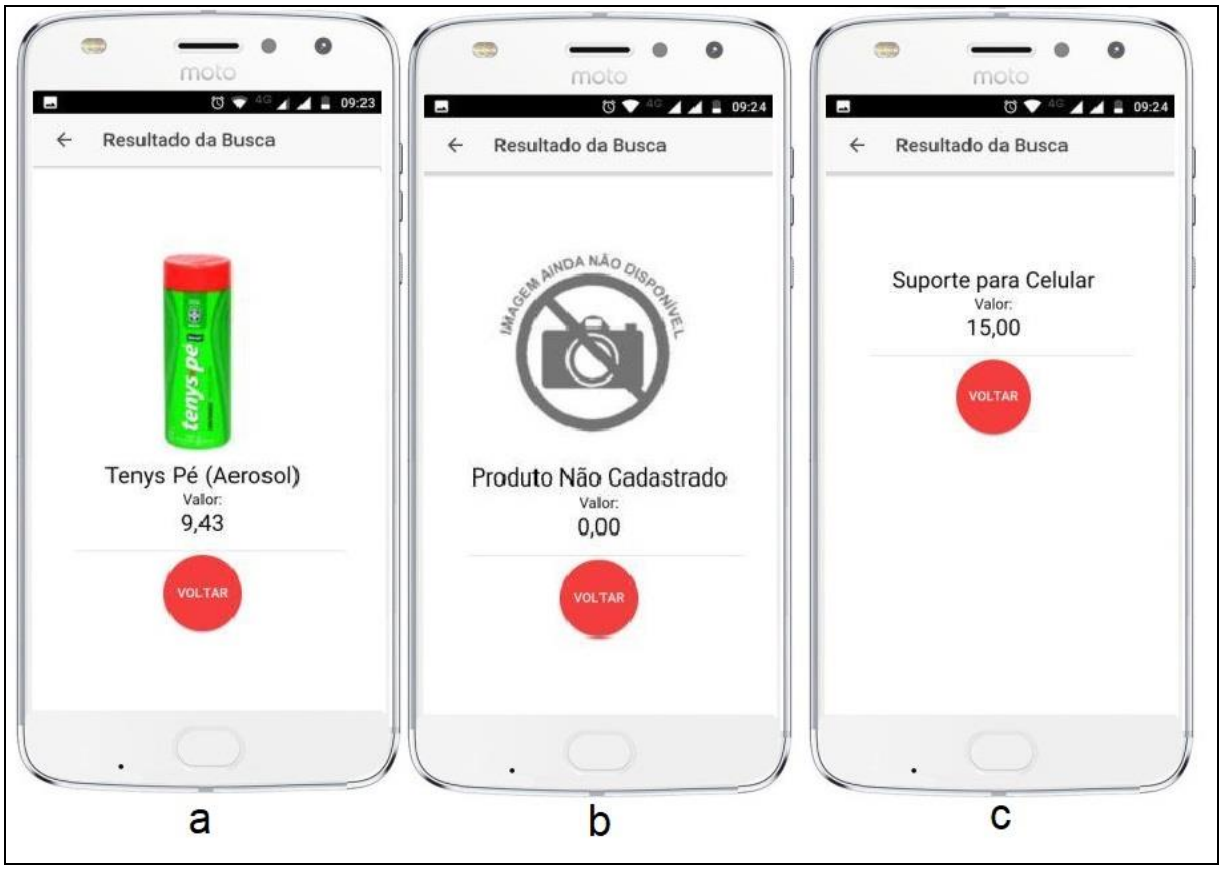
Fonte: Do autor.

Ao realizar a leitura do código de barras, utilizando o *plugin* nativo barcodeScanner, é retornado um valor, o qual será automaticamente utilizado para realizar a comparação no banco de dados, logo uma nova página com o nome ResultbuscaPage será carregada, sobrepondo a home. Na página ResultbuscaPage constará os dados do produto encontrado, ou os dados *default* para o produto não encontrado. Ainda, posterior ao retorno dos dados, quando o produto é encontrado, uma ação é acionada, na qual os dados são armazenados no banco de dados localStorage, visando a recuperação desses pela página de consultas.

Após a leitura do código de barras há duas possibilidades de *layouts* retornados ao usuário: a primeira ocorre quando a busca realizada no banco de

dados encontrou o registro, nesse caso mostrará os dados do produto encontrado, nessa tela poderá (Figura 7(a)) ou não (Figura 7(c)) conter imagem ilustrativa do produto, sendo que essa possui caráter opcional em seu cadastro; A segunda tela refere-se ao caso da busca não encontrar o produto, nesse caso é apresentada uma mensagem pré-programada, conforme pode-se observar na Figura 7 (b).

Figura 7 - Gaby - Telas que podem ser apresentadas após o retorno de uma consulta de um produto.



Fonte: Do autor.

Para realizar a busca do produto pelo código de barras no banco de dados é utilizada uma Querying do angularfire2, seguindo o padrão da linguagem, no qual é necessário primeiro ordenar os dados e em seguida fazer a comparação do código lido com os dados gravados. Caso seja encontrado o produto, entre os dados persistidos, os dados serão armazenados para posteriormente serem exibidos na tela, conforme Figuras 7 (a,c).

O aplicativo permite que o usuário visualize as últimas consultas realizadas (Figura 8), em que toda consulta que for realizada, e o produto for localizado, será

armazenada, facilitando assim futuras comparações entre valores dos diferentes produtos escaneados.

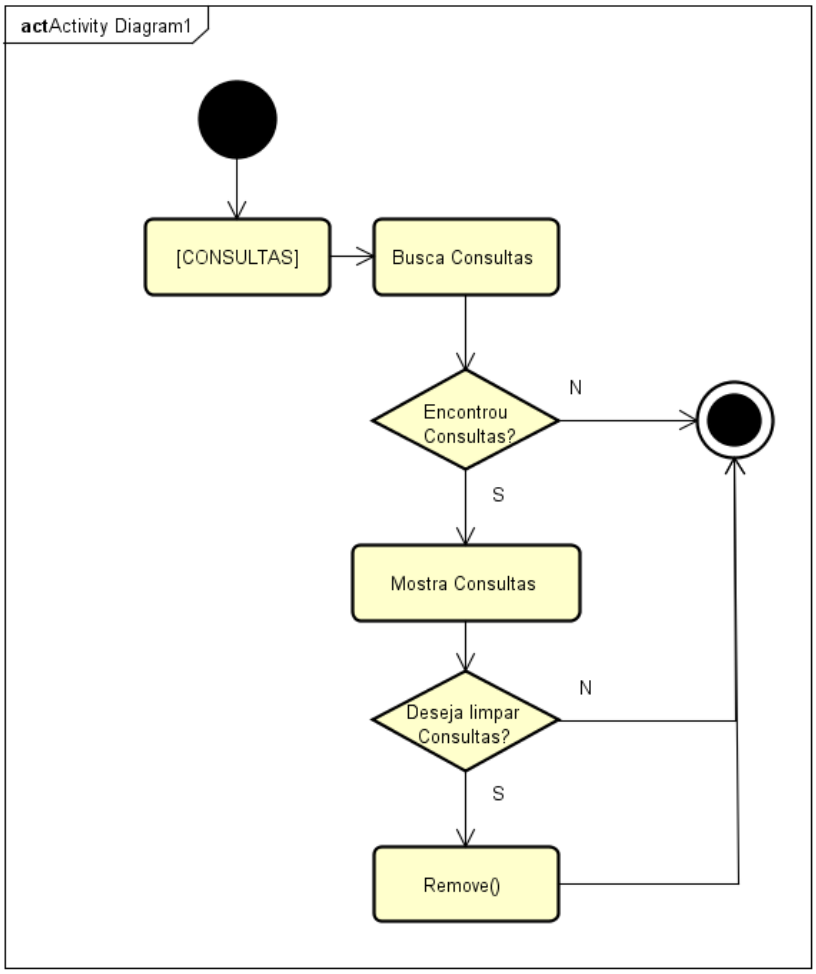
Figura 8 - Gaby - Tela de Consultas - apresenta histórico das consultas realizadas com produtos encontrados.

Fonte: Do autor.

Para permitir esse histórico, os dados das últimas consultas são armazenados em um banco de dados local, para isso foi utilizado o armazenamento nativo, Cordova plugin NativeStorage. Esse *plugin* fornece armazenamento e persistência de dados nativos no Android, iOS e Windows. Ele é criado por causa da propriedade não persistente do LocalStorage no WebView do Android e do iOS. As consultas armazenadas no LocalStorage podem ser removidas ao ser acionado o botão [APAGAR CONSULTAS].

A Figura 9 demonstra como são manipulados os dados do LocalStorage, o qual encontra-se organizado em funções (inserir, buscarConsultas e remove).

Figura 9 - Gaby - Diagrama de Atividades representando o fluxo da Tela de consultas armazenadas.



Fonte: Do autor.

A persistência dos dados é ilustrada na Figura 6 quando a consulta ao banco de dados Firebase é realizada, na qual além dos dados do produto é necessário inserir um código, chave primária da tabela. Assim, visando a resolução da regra de unicidade foi setado como código do produto a data, hora e minutos atuais, juntamente com o objeto produto, que contém todos os dados que foram retornados da consulta. Dessa forma, os registros das consultas realizadas são armazenadas e futuramente apresentadas ao usuário, conforme Figura 9. A função remove faz a limpeza de todo o localStorage, ou seja, os dados uma vez deletados não podem mais ser recuperados.

4.1 AVALIAÇÃO DO APLICATIVO COM USUÁRIOS

Uma versão de teste do aplicativo Gaby foi disponibilizada para um grupo de sete pessoas voluntárias a avaliar o aplicativo. Para esse teste foi disponibilizado a versão para Android. Os dados foram coletados no dia 29 de maio de 2018.

Após os usuários instalarem o aplicativo, foi disponibilizado um questionário, para avaliação, juntamente com alguns códigos de barras para simular produtos a serem consultados. No questionário foram apresentadas as seguintes questões:

- 01 - O que você achou do aplicativo em geral?
- 02 - Foi fácil de usar o aplicativo?
- 03 - Você considera importante o uso de aplicativos como este?
- 04 - Você utilizaria um aplicativo para adicionar suas compras (Carrinho de Compras) e pagar suas compras?

Para as questões acima, o usuário poderia classificar a resposta em “Sim” e “Não”, exceto na questão 01, a qual poderia ser avaliada em “Bom”, “Razoável” e “Ruim”.

Após a análise das respostas recebidas, obteve-se a avaliação do aplicativo em geral (questão 01), em que 100% dos usuários avaliaram de maneira positiva o aplicativo. Com relação à usabilidade do aplicativo (questão 02), 100% dos usuários avaliaram positivamente. No quesito de importância do uso de aplicativos como este no processo de automação comercial (questão 03), 71% dos participantes informaram que consideram importante. Referente ao questionamento sobre implementações futuras do aplicativo (questão 04), 71% dos participantes afirmaram que usariam o aplicativo para adicionar produtos no carrinho de compras e realizar o pagamento, demonstrando o potencial do aplicativo para novas funcionalidades.

4.2 REQUISITOS PARA FUNCIONAMENTO NO SMARTPHONE

Após testes em diferentes versões de Android determinou-se os requisitos necessários em relação ao *hardware* e *software* para execução do aplicativo. Nesse sentido verificou-se a necessidade de um smartphone contendo uma câmera, Sistema Operacional Android na versão 4.0.1, ou superior, e espaço em disco liberado de, no mínimo, 500 MB, para a instalação do aplicativo.

4.3 FORMA DE IMPLEMENTAÇÃO

Para implementação do aplicativo em um comércio existe a necessidade de importação dos dados dos produtos do mesmo para o banco de dados Firebase, para isso os dados podem ser importado para o mesmo com um arquivo .Json, no qual o comércio deve realizar a exportação do seu banco de dados atual, ou seja, realizar um backup dos dados no formato de arquivo de texto JSON, o qual será importado ao banco de dados Firebase. Dessa forma a sincronização dos dados entre o banco de dados local e o banco de dados em *cloud* deverá ocorrer com determinada frequência (sempre que houver alterações dos produtos).

Contudo, o aplicativo desenvolvido realizará a busca dos dados em um banco de dados em *cloud*, assim, o acesso aos dados não fica restrito somente ao funcionamento interno no comércio.

5 CONSIDERAÇÕES FINAIS

O aplicativo desenvolvido neste trabalho visa atender a uma demanda, que não é atendida de maneira adequada pelos sistemas disponíveis. O aplicativo desenvolvido apresenta alguns diferenciais em relação à maneira atual de disponibilizar aos clientes um terminal de consultas preços.

Dessa forma foi desenvolvido um aplicativo de fácil usabilidade para realizar consultas de preços em determinado estabelecimento por meio da leitura do código de barras do produto, para isso o aplicativo foi desenvolvido usando o *framework* Ionic. Além disso, para armazenamento dos dados foi utilizado o banco de dados local *Storage* e o banco de dados em *cloud* Firebase.

No decorrer do desenvolvimento foi enfrentado alguns conflitos em relação ao versionamento do Android cordova com o *plugin* nativo barcode Scanner, sendo a mesma resolvida, utilizando uma versão anterior do mesmo.

Como trabalhos futuros verifica-se a possibilidade de aperfeiçoamento da usabilidade do sistema e também para aumentar a sua funcionalidade. Em relação à usabilidade é possível realizar melhorias relacionadas ao design do aplicativo. Já em relação às novas funcionalidades verifica-se a possibilidade de: adicionar o produto

ao carrinho de compras após realizar a consulta do valor; adicionar um lembrete caso o usuário deseja receber um aviso quando o carrinho chegar a um valor pré-determinado; realizar pagamento das compras via cartão de crédito/débito; e, restrição do acesso aos dados do banco de dados em *cloud* para funcionamento do aplicativo somente no ambiente interno do comércio.

ABSTRACT

Was developed this work, a system to perform price queries through mobile devices. The main motivation for the development of this application was to meet the potential unexplored demand, considering that the Price Consultation Terminals (PCT) available in the market have a high cost of acquisition, besides having a restriction due to the fact that they are fixed models. To create an alternative to these PCT models, a hybrid tecnology based mobile application was developed, fully interconnected by a cloud database, called Firebase. The application presents the user with a screen that allows scanning the barcode of the product that wants to know the price, using the camera of the smartphone. With this, the server receives the code read and returns the data of the registered product, when found. After the implementation and realization of the tests performed with the application, it was possible to verify its potentiality of use, besides demonstrating its potential for new functionalities.
Keywords: PCT; App; Hybrid technology; Firebase.

6

REFERÊNCIAS

BARBOSA, Alvaro Cesar Pereira. Middleware para Integração de Dados Heterogêneos Baseado em Composição de Frameworks. 2001. 167 f. Tese (Doutor em Informática: Ciência da Computação). Departamento de Informática da PUC-Rio, Rio de Janeiro, 2001

BRASIL, Decreto Nº 5.903, DE 20 DE SET. DE 2006. Regulamenta a Lei no 10.962, de 11 de outubro de 2004, e a Lei no 8.078, de 11 de setembro de 1990, Brasilia, DF, set 2006. Disponível em: <http://www.planalto.gov.br/ccivil_03/_ato2004-2006/2006/decreto/d5903.htm>. Acesso em: 25 fev. 2018.

Callebaut, Gilles. What is NativeStorage. 2016. Disponível em: <https://github.com/TheCocoaProject/cordova-plugin-nativestorage/wiki>. Acesso em 23 de abr. 2018.

CLARO, Alberto. Sistemas de informações gerenciais: 1º Edição: setembro de 2013. São Paulo/SP, ISBN: 978-85-8065-266-0.

COBAN. Código EAN-13 <http://codigodebarrasnacional.com.br/codigo-de-barras-ean-13>. Acesso em: 27 mar. 2018.

DEVMEDIA. Introdução ao Ionic. Disponível em: <https://www.devmedia.com.br/guia/ionic/38372>. Acesso em: 23 de abr. 2018.

ESTADÃO. Brasil tem 168 milhões de smartphones. Disponível em <https://epocanegocios.globo.com/Tecnologia/noticia/2016/04/epoca-negocios-brasil-tem-168-milhoes-de-smartphones.html>. Acesso em 12 de mar. 2018.

FIREBASE. 2017. Documentação do Firebase. Disponível em: <https://firebase.google.com/docs>. Acesso em: 16 mar. 2018.

GERTEC. Busca preço. Disponível em: <https://www.gertec.com.br/produtos/busca-preco/ >. Acesso em: 21 mar. 2018.

GERTEC. TC506 Mídia. Disponível em: <https://www.gertec.com.br/produtos/tc-506-midia/>. Acesso em: 21 mar. 2018.

GRILLO, Rafael. Introdução ao Ionic Framework, Tabless, 2015. Disponível em: <http://tableless.com.br/introducao-ao-ionic-framework/>. Acesso em: 23 de abr. 2018.

GS1 Brasil. Código de Barras: EAN/UPC. Disponível em: <https://www.gs1br.org/codigos-e-padroes/codigo-de-barras/ean-upc>. Acesso em 22 de abr. 2018.

Ionic Framework. Installing Ionic. Disponível em: <https://ionicframework.com/docs/intro/installation/>. Acesso em: 23 abr. 2018.

JUNIOR, Luciano Tavares. A Origem do Código de Barras. Linha base Softwares Ltda, 2010.

MILIES, C.P. A matemática dos códigos de barras. Boletim da Sociedade Brasileira de Matemática, v. 65, p. 46-53, 2008.

VANZ, Nórton Mattiello; FIGUEIREDO, José Antônio Oliveira de. UM ESTUDO SOBRE A EVOLUÇÃO DO CÓDIGO DE BARRAS LINEAR ATÉ O QR CODE E SUA APLICAÇÃO EM UM ESTUDO DE CASO. 2012. 65 f. TCC (Graduação) - Curso de Tecnólogo em Sistemas Para Internet, Instituto Federal Sul-rio-grandense, Passo Fundo, 2012. Disponível em:

<http://painel.passofundo.ifsul.edu.br/uploads/arq/20160330191112156236124 8.pdf>. Acesso em: 25 dez. 2017.