UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIA DA COMPUTAÇÃO – BACHARELADO

DITA OFERTAS: UMA APLICAÇÃO PARA REPRODUZIR OFERTAS DE PRODUTOS NO RÁDIO DO CARRO

LUCAS GOMES RAQUEL

BLUMENAU
2017

LUCAS GOMES RAQUEL

DITA OFERTAS: UMA APLICAÇÃO PARA REPRODUZIR OFERTAS DE PRODUTOS NO RÁDIO DO CARRO

Trabalho de Conclusão de Curso apresentado ao curso de graduação em Ciência da Computação do Centro de Ciências Exatas e Naturais da Universidade Regional de Blumenau como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação.

Prof. Aurélio Faustino Hoppe - Orientador

BLUMENAU
2017

DITA OFERTAS: UMA APLICAÇÃO PARA REPRODUZIR OFERTAS DE PRODUTOS NO RÁDIO DO CARRO

Por

LUCAS GOMES RAQUEL

Trabalho de Conclusão de Curso aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II pela banca examinadora formada por:

Presidente:	Prof. Aurélio Faustino Hoppe, Mestre – Orientador, FURB
Membro:	Prof. Alexander Roberto Valdameri, Mestre – FURB
Membro:	Profa. Joyce Martins, Mestre – FURB

Blumenau, 4 de julho de 2017

Dedico este trabalho a minha família e amigos que sempre me apoiaram.

AGRADECIMENTOS

A minha família por sempre me apoiar e me auxiliar nas decisões que tomei durante a minha vida.

Aos meus amigos que me apoiaram e compreenderam o meu afastamento principalmente na etapa final do desenvolvimento do trabalho.

Ao meu amigo Leonard William e a sua mãe Sônia, por terem me ajudado com os testes no rádio do carro dela.

Ao meu orientador Aurélio, por sempre ter me mantido calmo e com os pés no chão durante o desenvolvimento do trabalho, principalmente quando nos deparávamos com algum contratempo.

A todas as pessoas que nestes quatro anos e meio puderam me passar suas experiências e ensinamentos de alguma forma.

O dinheiro faz homens ricos, o conhecimento faz homens sábios e a humildade faz grandes homens.

Mahatma Gandhi

RESUMO

Este trabalho descreve o desenvolvimento de um aplicativo Android que tem como objetivo reproduzir no rádio do carro, via Bluetooth, ofertas de produtos extraídos de lojas virtuais. O usuário poderá escolher as categorias dos produtos e de quais lojas ele quer obter as ofertas. A partir disso foi desenvolvido um *web service* para fazer a extração dos dados, retornando as ofertas para serem reproduzidas via aplicativo Android. Para o desenvolvimento do aplicativo foi utilizado o *framework* React Native juntamente com alguns *plugins* que implementam o banco de dados SQLite e a API `android.speech.tts`, para a sintetização de fala. Para o desenvolvimento do *web service*, responsável pela extração dos dados, foi utilizada a plataforma Node.js juntamente com o *framework* Express e algumas bibliotecas utilizadas para recuperar o conteúdo das páginas web. A reprodução das ofertas no rádio do carro acontece por meio do pareamento entre o smartphone e o rádio do carro via Bluetooth. Os resultados dos testes realizados com usuários demonstraram que a aplicação tem um bom desempenho e que ela cumpre com seu objetivo, apesar de ter pontos a melhorar em questões de usabilidade e na extração dos dados.

Palavras-chave: Ofertas. Extração de conteúdo web. Rádio. React. React native. Node.js. Android.

ABSTRACT

This work describes the development of an Android application that aims to reproduce on the car radio, via Bluetooth, offers of products extracted from virtual stores. The user can choose the product categories and which stores he wants to get the offers. From this a web service was developed to make the data extraction, returning the offers to be reproduced via Android application. For the development of the application was used React Native framework along with some plugins that implement the SQLite database and the `android.speech.tts` API for speech synthesis. For the development of the web service, responsible for extracting the data, the Node.js platform was used along with the Express framework and some libraries used to retrieve the content of the web pages. The playback of the offers on the car radio takes place through the pairing between the smartphone and the car radio via Bluetooth. The results of the tests performed with users showed that the application performs well and that it accomplishes its objective, although it has points to improve in questions of usability and in the extraction of the data.

Key-words: Offers. Web content extraction. Radio. React. React native. Node.js. Android.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

Abramet - Associação Brasileira de Medicina de Tráfego

API - Application Programming Interface

CNI - Confederação Nacional da Indústria

CSS - Cascading Style Sheet

CTB - Código de Trânsito Brasileiro

DOM - Document Object Model

GPS - Global Positioning System

HTML - HyperText Markup Language

HTTP - Hypertext Transfer Protocol

IDE - Integrated Development Environment

JSON - JavaScript Object Notation

JSX - JavaScript XML

MER - Modelo de Entidade Relacionamento

NHTSA - National Highway Traffic Safety Administration

NPM - Node Package Manager

PDS - Processamento Digital de Sinais

PIN - Personal Identification Number

PLN - Processamento de Linguagem Natural

REST - Representational State Transfer

RF - Requisitos Funcionais

RNF - Requisitos Não Funcionais

SMS - Short Message Service

TTS - Text To Speech

URL - Uniform Resource Locators

W3C - World Wide Web Consortium

XML - eXtensible Markup Language

SUMÁRIO

1 INTRODUÇÃO

Segundo as estimativas da Pesquisa Nacional por Amostra de Domicílios (IBGE, 2013, p. 43), em 2013 a quantidade de pessoas com 10 anos de idade, ou mais, que possuíam celular para uso pessoal era de 130,2 milhões, o que correspondia a 75,2% da população do país nessa faixa de idade. Segundo a International Data Corporation Brasil (TEIXEIRA, 2015), em 2014 foram vendidos em torno de 54,2 milhões de smartphones, uma alta de 55% em relação ao ano anterior. Com a rápida expansão de redes móveis para diversas regiões do país e o aumento substancial em seu uso, os usuários brasileiros de internet vivem uma mobilidade conectada e não estão mais restritos a locais específicos para acessar a rede.

Por outro lado, segundo a Confederação Nacional da Indústria (CNI) (CNI, 2015, p. 04), em 2011, 26% dos brasileiros gastavam mais de uma hora por dia em seu deslocamento de carro para suas atividades rotineiras, como trabalho e estudo. Entre 2011 e 2014 esse percentual aumentou 5 pontos percentuais, chegando a 31%. Segundo Resende e Sousa (2009), dentre as principais atividades exercidas durante este período estão ouvir música, seguida de ouvir ou acessar informações em canais de notícias ou ofertas. Ainda, o meio de acesso mais utilizado é o smartphone ou o rádio do carro.

No entanto, é importante ressaltar que de acordo com o Código de Trânsito Brasileiro (CTB) (CTB, art. 252, parágrafo único), dirigir enquanto se está manuseando um telefone celular caracteriza-se como infração gravíssima. Segundo a National Highway Traffic Safety Administration (NHTSA), a autoridade de segurança de trânsito dos Estados Unidos, as distrações com o celular no trânsito são responsáveis por cerca de 80% dos acidentes causados. No Brasil, o Laboratório de Psicofísica e Percepção da Universidade de São Paulo afirma que dentre os 98% dos acidentes resultantes de fatores humanos, 72% são por falta de atenção. De acordo com Dirceu Rodrigues Alves, diretor da Associação Brasileira de Medicina de Tráfego (Abramet), as distrações mais comuns são o envio de mensagens de texto e falar ao celular (CARVALHO, 2016). Estes fatos fazem com que o condutor tenha que adotar outras medidas para ouvir músicas ou acessar informações em canais de notícias ou ofertas de produtos enquanto está dirigindo.

Diante desse cenário, talvez a solução mais viável seja tornar o rádio do carro, com auxílio do smartphone, um reprodutor de conteúdos de acordo com o interesse do usuário. Sendo assim, este trabalho apresenta o desenvolvimento de uma aplicação para dispositivos móveis que permitirá ao usuário selecionar *sites* para terem suas principais ofertas de produtos extraídas e reproduzidas na forma de áudio no rádio do carro.

1.1 OBJETIVOS

Este trabalho tem como objetivo o desenvolvimento de uma aplicação para dispositivos móveis que reproduza no rádio do carro as principais ofertas de *sites* de compras de acordo com o interesse do usuário.

Os objetivos específicos são:

a) disponibilizar um *web service* que faça a extração de ofertas em *sites* informados pelo usuário;
b) converter as ofertas para um formato de áudio afim de reproduzi-lo no rádio do carro via conexão Bluetooth.

1.2 ESTRUTURA

O presente trabalho compõe-se de quatro capítulos: introdução, fundamentação teórica, desenvolvimento e conclusões. O segundo capítulo discorre sobre os principais assuntos relacionados a este trabalho e se trata do embasamento teórico do desenvolvimento do trabalho. O terceiro capítulo apresenta os diagramas, a arquitetura da aplicação, assim como trechos de código e algumas telas do aplicativo. Por fim, o quarto capítulo expõe as conclusões obtidas a partir dos resultados do trabalho e sugestões de futuras extensões deste.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta os principais assuntos relacionados a este trabalho, sendo eles necessários para o seu entendimento. A seção 2.1 traz uma visão a respeito dos hábitos do consumidor. A seção 2.2 trata sobre a mineração web, seus enfoques e suas principais tarefas. A seção 2.3 traz uma abordagem sobre síntese de voz e algumas de suas aplicações. A seção 2.4 trata sobre o *framework* React Native e suas principais características. Por fim, a seção 2.5 apresenta os trabalhos correlatos.

2.1 COMPORTAMENTO DE CONSUMO

A revolução digital é uma das influências mais significativas sobre o comportamento do consumidor e o impacto da web continuará a se expandir à medida que as pessoas ao redor do mundo vão conectando-se umas às outras (SOLOMON, 2016, p. 23).

Além disso, não se trata somente de empresas que vendem aos consumidores (comércio eletrônico B2C - *business to consumer*). A explosão do ciberespaço criou uma revolução na atividade de consumidor para consumidor (comércio eletrônico C2C - *consumer to consumer*). Do mesmo modo que os consumidores virtuais não se limitam aos distribuidores varejistas locais para fazer compras, também não estão limitados às suas comunidades locais quando procuram pessoas que possuam interesses semelhantes aos seus. (SOLOMON, 2016, p. 24)

O amplo acesso a aparelhos como computadores pessoais, gravadores de vídeo e áudio digitais, *webcams* e smartphones garante que os consumidores de quase qualquer idade que vivem em praticamente qualquer parte do mundo possam criar e compartilhar conteúdo. Contudo, as informações não fluem simplesmente das grandes empresas ou dos governos para as pessoas. Hoje, é possível se comunicar com um número imenso de pessoas com um clique no teclado. Assim, as informações fluem também entre as pessoas (SOLOMON, 2016, p. 25).

Isso é definido como a revolução horizontal, onde ela é caracterizada pela prevalência das mídias sociais. As mídias sociais são meios online de comunicação, transmissão, colaboração e cultivo entre redes interconectadas e interdependentes de pessoas, comunidades e organizações aperfeiçoadas por recursos tecnológicos e mobilidade tecnológica. (SOLOMON, 2016, p. 25)

De acordo com a pesquisa do SPC Brasil e Meu Bolso Feliz, a influência da internet na experiência de compra em lojas físicas é uma realidade no comércio brasileiro, já que nove em cada dez consumidores virtuais (90%) fazem pesquisa virtual sobre o produto antes de comprá-lo numa loja física (SPC BRASIL, 2015).

De acordo com o estudo, o consumidor virtual brasileiro deseja empregar bem seu dinheiro, buscando saber se não está pagando mais caro ou adquirindo um produto que pode trazer problemas (SPC Brasil, 2015). Dentre as fontes de pesquisa dos consumidores, os sites de comparação de preço aparecem em primeiro lugar (62%). Em segundo estão os sites como o "Reclame Aqui", para verificar o índice de reclamação do produto (54%). Em terceiro estão as lojas virtuais (47%), para verificar preços, condições de pagamento e demais características. Além dessas três fontes, os consumidores também consultam as redes sociais (39%) e os blogs especializados (36%), mostrando que os consumidores virtuais estão atentos à opinião de outras pessoas, tanto especialistas quanto demais consumidores virtuais que possam compartilhar experiências positivas ou negativas a respeito da compra (SPC BRASIL, 2015).

2.2 MINERAÇÃO WEB

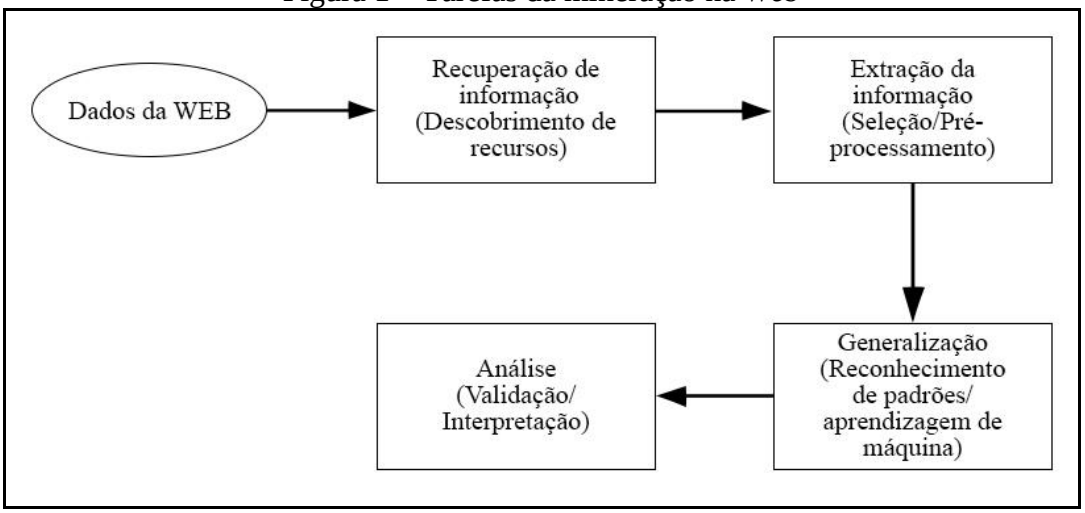
De acordo com Adaniya e Proença (2009, p.30), a *Web Mining* (Mineração *Web*) é o uso das técnicas de Mineração de Dados para descobrir e extrair automaticamente a informação de documentos na *web*.

De acordo com Santos (2009, p.17), a mineração *web* tem três enfoques principais:

a) mineração de conteúdo da *Web*: é o processo de extração de conhecimento do conteúdo de documentos e de seus metadados. Este enfoque abrange principalmente documentos textuais, como páginas em texto, HyperText Markup Language (HTML), blogs, etc;
b) mineração de Estruturas *Web*: que é o processo de descoberta de conhecimento a partir da ligação entre documentos na *web*;
c) mineração de Uso da *Web*: é voltado para a análise de dados coletados sobre o acesso a documentos *web*, como *logs*, com o intuito de descobrir padrões de acessos a *sites* ou conjuntos de documentos para melhorar a experiência dos usuários.

Segundo Adaniya e Proença (2009, p.30), a mineração *web* é decomposta em quatro tarefas: *Resource finding* (Coleta de Recursos), *Information selection and pre-processing* (Pré-processamento), *Generalization* (Extração de Padrões) e *Analysis* (Análise), conforme pode ser visto na Figura 1.

Figura 1 – Tarefas da mineração na *Web*

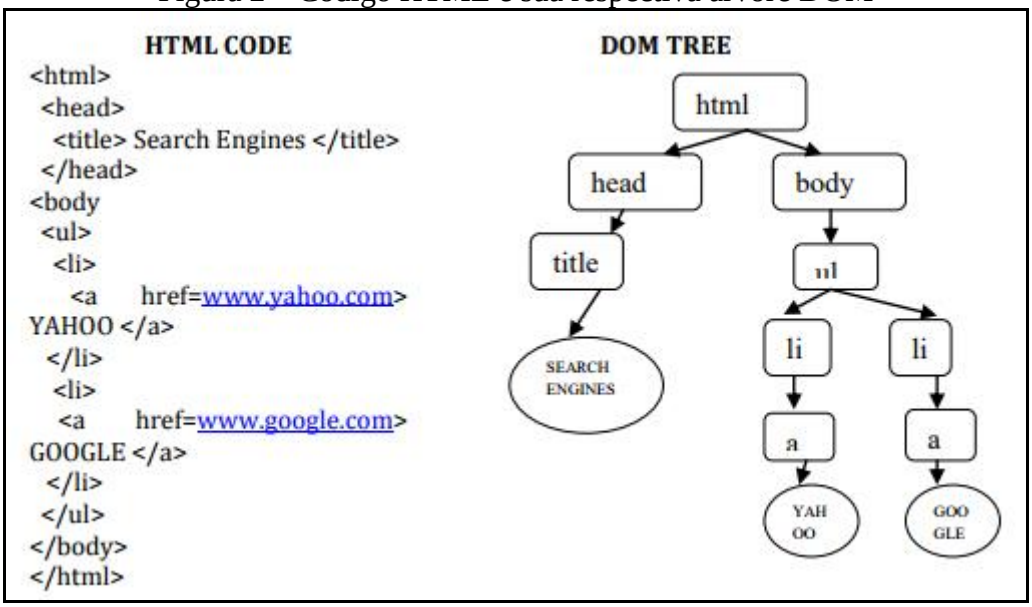


Fonte: Marinho e Girardi (2003, p. 2).

De acordo com Andaniya e Proença (2009, p.28), a Recuperação de Informação é a tarefa de encontrar documentos relevantes a partir de um corpus ou conjunto de textos em resposta a uma necessidade de informação de um usuário. Na tarefa de Extração da informação é onde ocorre a identificação de fragmentos específicos que constituem o núcleo semântico de um documento em particular e a construção de modelos de representação da informação a partir dele. Na tarefa de Generalização são aplicadas técnicas de mineração de dados e aprendizagem de máquina para descobrir novo conhecimento a partir do que já existe. A última tarefa, a de Análise é onde os analistas utilizam técnicas e ferramentas apropriadas para entender, visualizar, interpretar e validar os padrões descobertos nas tarefas anteriores.

Uma estratégia muito utilizada para a extração de conteúdo *web* é através da árvore Document Object Model (DOM), no qual ao invés de utilizar o documento HTML em si, a árvore DOM deste documento é construída para que desta forma seja possível navegar por entre os nodos da árvore e fazer a extração dos elementos desejados (MANJULA; CHILAMBUCHELVAN, 2016, p. 1772). Na Figura 2 é possível visualizar o código HTML e a sua respectiva árvore DOM, onde fica evidente que cada elemento HTML que é renderizado pelo *browser* possui um nó correspondente na árvore DOM. É através dele que é possível obter o conteúdo e as propriedades dos elementos HTML correspondentes.

Figura 2 – Código HTML e sua respectiva árvore DOM



Fonte: Mamjula e Chilambuchelvan (2016, p. 1771-1772).

De acordo com a especificação da World Wide Web Consortium (W3C), o DOM é uma Application Programming Interface (API) que define a estrutura lógica de documentos HTML e a forma com que eles são acessados e manipulados. O DOM é utilizado para construir o documento, navegar entre a sua a estrutura e realizar operações como adicionar, alterar e deletar as propriedades dos elementos. (MANJULA, CHILAMBUCHELVAN, 2016, p. 1772).

2.3 SÍNTESE DE VOZ

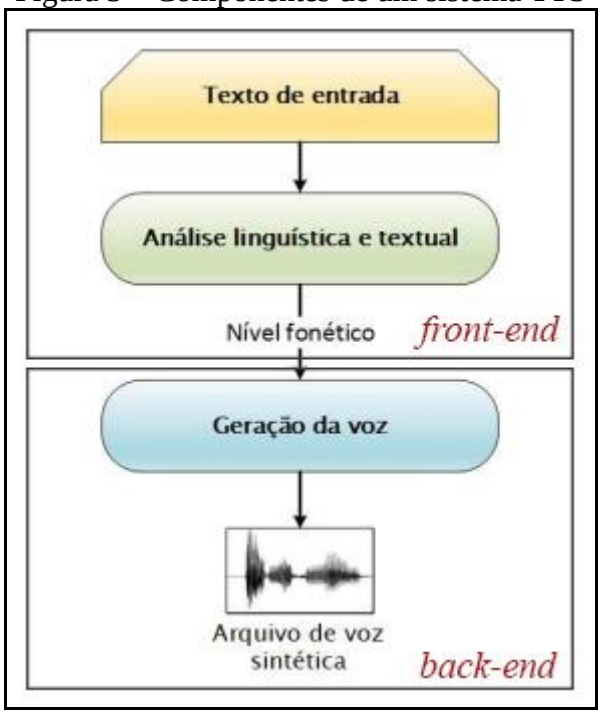
Segundo Zuffo e Pistori (2004, p.1), a síntese de voz consiste na geração de sinais sonoros que reproduzem as palavras equivalentes em uma determinada linguagem natural. Ainda segundo eles, o crescimento da utilização de sistemas computadorizados pelas pessoas aumenta a necessidade de comunicação ágil entre máquinas e seres humanos. Com o aumento na capacidade de processamento e da disponibilidade de dispositivos para reprodução de som digital, a síntese de voz tem se tornado viável em uma grande variedade de aplicações.

Hoje, a síntese de voz pode ser aplicada em sistemas de orientação e navegação (sistemas de voz aplicados a navegação por Global Positioning System (GPS)), no ensino *e-learning* com interfaces de voz, controle de sistema por voz e em telecomunicação (leitura Short Message Service (SMS) por voz, útil a cegos) (RIBEIRO, 2010, p. 20).

Uma das aplicações da síntese de voz pode ser vista em sistemas Text To Speech (TTS), no qual tem como entrada um texto que será convertido em uma voz artificial (ARAÚJO, 2015, p.12).

Existem algumas APIs que são disponibilizadas como serviços na nuvem, que facilitam a criação de aplicações TTS, como é o caso da API de Text to Speech fornecida pela IBM Watson, que é uma plataforma de Inteligência Artificial. Além dela, há também a API Speech oferecida pela Microsoft Azure. Outra opção na nuvem, oferecida pela Amazon, é o serviço Amazon Polly que transforma texto em fala. Nativamente, uma das opções disponíveis para o desenvolvimento de aplicações TTS é o uso da biblioteca `android.speech.tts`, que está disponível no Android desde a sua versão 1.6 (ANDROID, 2017). A classe `TextToSpeech`, disponível nessa biblioteca, é responsável por transformar texto em voz para a reprodução imediata ou gerar um arquivo de áudio com o texto sintetizado (ANDROID, 2017). De acordo com Araújo (2015, p.12), os sistemas TTS são compostos por dois componentes principais, o *front-end* e o *back-end*, ilustrados pela Figura 3.

Figura 3 – Componentes de um sistema TTS



Fonte: adaptado de Araújo (2015, p.12).

O *front-end* é responsável por receber o texto de entrada, converter símbolos em palavras escritas equivalentes, além de realizar a transcrição fonética de cada palavra e dividir o texto em unidades menores. Essa análise textual e linguística é feita por um módulo de Processamento de Linguagem Natural (PLN). Ainda segundo a autora, o *back-end* realiza a geração da voz, convertendo a representação linguística em voz através de um componente de Processamento Digital de Sinais (PDS).

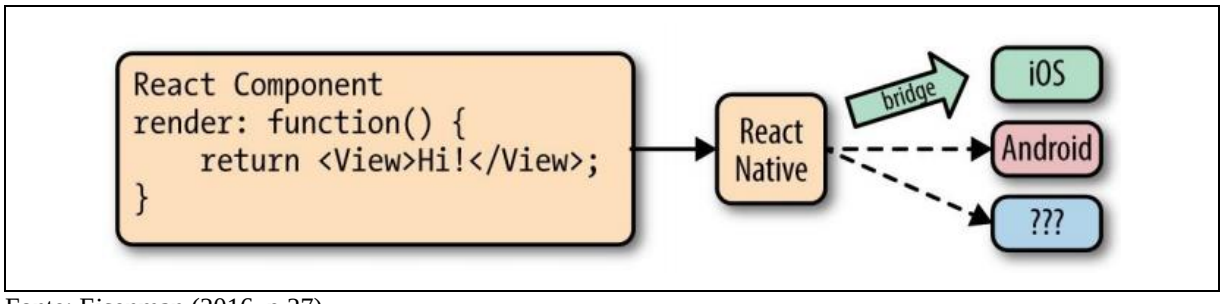
Segundo Araújo (2015, p.13), a qualidade de um sistema TTS possui duas medidas, a segmental e a suprasegmental. A qualidade segmental representa a eficiência da máquina em produzir fala soando próxima a uma voz natural e a qualidade suprasegmental representa a riqueza de conteúdo prosódico (entonação da voz) que a máquina é capaz de explorar.

2.4 REACT NATIVE

O React Native é um *framework* em JavaScript baseado na biblioteca React, que foi desenvolvida pelo Facebook. Seu propósito é permitir o desenvolvimento de aplicações *mobile* para as plataformas Android e iOS. As aplicações em React Native são desenvolvidas utilizando Cascading Style Sheet (CSS) e JavaScript XML (JSX), que é uma mistura de JavaScript e eXtensible Markup Language (XML), portanto qualquer desenvolvedor familiarizado com o React não encontrará muitas dificuldades para desenvolver aplicações utilizando o *framework.*

Diferente de outras alternativas para aplicações *mobile* que utilizam *webviews* para renderizar a aplicação, o React Native invoca as APIs de renderização nativas em Objective-C, para iOS, ou Java para Android. Portanto, a aplicação será renderizada utilizando componentes nativos da plataforma (EISENMAN, 2016). Isso acontece por causa da "*bridge"* que fornece os componentes React com uma interface para a plataforma alvo, com isso componentes como uma `<View>` são renderizados como uma `UiView` nativa da plataforma iOS ou uma `View` nativa do Android. A atuação da *bridge* pode ser vista na Figura 4.

Figura 4 – Renderização de componentes React para diversas plataformas



Fonte: Eisenman (2016, p.27).

Caso seja preciso usar alguma API ou componente além dos que já são disponibilizados pelo *framework,* é possível instalar *plugins* que permitem que estes recursos sejam utilizados na aplicação, como *Text to Speech* ou SQLite, por exemplo. Uma das vantagens do React Native em relação a outros *frameworks,* é que caso seja preciso utilizar algum recurso que não está disponível pelo próprio *framework* ou não tenha nenhum *plugin* que atenda ao que se deseja, é possível implementar módulos nativos com Java, para Android, ou Objective-C, para iOS.

Além disso, com o React Native não é necessário refazer o *build* da aplicação para poder visualizar as alterações feitas, a menos que tenha sido adicionada alguma imagem ao projeto (REACT NATIVE, 2017). Ele permite ao desenvolvedor atualizar a aplicação através da opção "*Reload*" disponível para o desenvolvedor. Para atualizar a aplicação o React Native conta com mais duas opções além do "*Reload*", uma delas é o *Live Reload* que recarrega a aplicação sempre que o desenvolvedor salva as alterações no código. A outra opção é o *Hot Reloading* que ao contrário do *Live Reload* não recarrega toda a aplicação, mas sim apenas o que foi alterado pelo desenvolvedor, ou seja, mantém-se o estado atual da aplicação e apenas o que for modificado ou for novo, é atualizado na tela (REACT NATIVE, 2017).

2.5 TRABALHOS CORRELATOS

A seguir são apresentados três trabalhos correlatos voltados para a mineração na *web*. A seção 2.4.1 aborda o trabalho de Singh (2012), no qual é apresentada uma aplicação para a extração de conteúdos não necessários de páginas *web*. A seção 2.4.2 descreve o trabalho de Gondse e Raut (2014), no qual foi desenvolvido um método para extração de conteúdos em páginas *web*. Por fim, a seção 2.4.3 apresenta o trabalho de Vieira et al. (2015).

2.5.1 Web content extraction to facilitate web mining

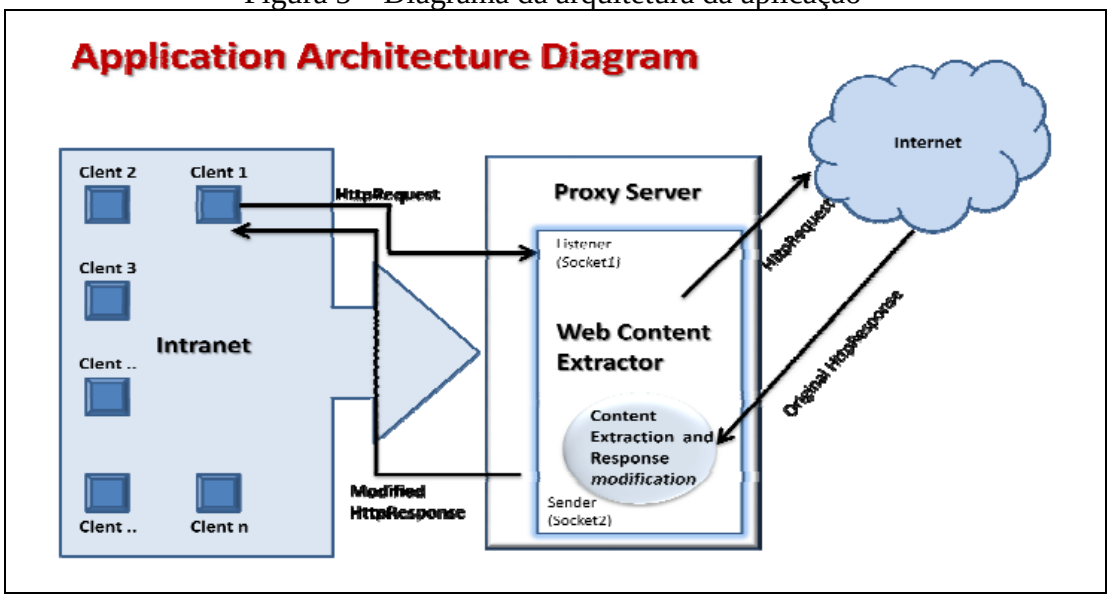
Singh (2012) utilizou a estrutura DOM das páginas *web* para implementar uma técnica para remover dados irrelevantes e assim otimizar o processo de mineração *web*. Seu foco principal foi desenvolver uma técnica que mantenha a estrutura da navegação, porém, removendo imagens, publicidade e melhorando a eficiência da navegação.

O extrator de conteúdo das páginas *web* de Singh (2012) navega pela árvore DOM recursivamente usando uma série de diferentes técnicas de filtragem para remover e modificar nós específicos e deixar apenas o conteúdo. Ou seja, as filtragens servem para remover elementos pesados da página, como imagens e publicidades, antes de renderizar a saída para o usuário, reduzindo o consumo de banda.

Depois que a árvore DOM é analisada e modificada, ela pode ser retornada em formato HTML ou em texto. Quando retornada em formato texto, todas as *tags* são removidas, deixando apenas o conteúdo do *site*, tornando-a ideal para sumarização, renderização em voz ou armazenamento.

Para atender o que foi proposto, Singh (2012) implementou um *proxy* de aplicativo *web*, com uma interface gráfica que permite ao administrador da rede ter controle sobre os dados enviados para os clientes dentro da intranet, conforme pode ser visto na Figura 5.

Figura 5 – Diagrama da arquitetura da aplicação



Fonte: Singh (2012, p. 05).

Todas as solicitações dos clientes assim como as respostas para estas solicitações, vindas da internet, passam pelo *proxy* onde é feito a extração do conteúdo no qual é modificado e enviado para o cliente. O conteúdo será o mesmo, porém sem elementos como imagens e anúncios.

Segundo Singh (2012), sua abordagem para a extração de conteúdo das páginas *web* obteve bons resultados quando se refere a um menor uso de banda da intranet. Porém, sugere a utilização de abordagens mais eficientes para realizar a extração de conteúdo *web*, pois a estrutura das páginas *web* mais modernas dificultam o processo de extração do conteúdo.

2.5.2 Main content extraction from web page using dom

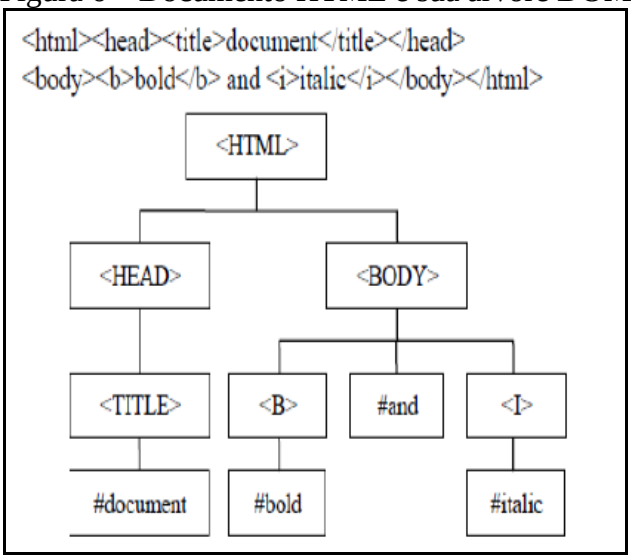
Tendo em vista o aumento do conteúdo na internet e junto dele conteúdos que não podem ser classificados como conteúdo informativo, como blocos de navegação, barras de pesquisa e anúncios, Gondse e Raut (2014) propuseram o desenvolvimento de um método que consiga extrair o conteúdo principal das páginas *web* de forma rápida e precisa.

A abordagem proposta concentra-se nas páginas onde as informações subjacentes são textos não estruturados.

> A técnica utilizada para a extração das informações é aplicada a todas as páginas web, enquanto eles atualmente buscam informações apenas a partir de blocos de conteúdo primário das páginas web. O usuário especifica a informação necessária ao sistema. Os web *crawlers* baixam as páginas web a partir de uma ou mais listas de Uniform Resource Locators (URLs), baixando cada uma das páginas associadas, extraindo os *hyperlinks* contidos nelas e recursivamente baixando essas páginas (GONDSE; RAUT, 2014, p. 2, tradução nossa).

Após a extração das páginas relevantes ter sido feita pelos *web crawlers*, elas são representadas em forma de uma árvore DOM. A Figura 6 ilustra um documento HTML e sua respectiva árvore DOM.

Figura 6 – Documento HTML e sua árvore DOM



Fonte: Gondse e Raut (2014, p. 2).

A partir dessa estrutura, o algoritmo de segmentação particiona a página *web* baseado na primeira *tag* para identificar os blocos, subpartições são feitas baseado nas *tags* seguintes e assim por diante. Essa identificação dos blocos continua até que haja uma *tag* na lista de *tags.*

Para a extração do conteúdo, Gondse e Raut (2014) realizaram 5 passos. O primeiro passo foi de limpeza da página HTML. Nele, é feita a validação léxica e sintática das *tags* HTML. O segundo passo foi o de pré-processamento das *tags*, onde todas as *tags* inválidas ou que não possuíam conteúdo foram removidas. No terceiro passo foi feita a seleção dos nós ótimos que possuem conteúdo, se um nó não atendia essa condição, o texto abaixo dele não é identificado. O quarto passo é para a extração do conteúdo através de ferramentas que analisam o HTML. Se o nó não satisfizesse as condições, deveria retornar ao passo 3 para poder encontrar nós que possuem texto como sendo seu filho. O quinto e último passo é para ajustar os resultados da extração do passo 4. Mesmo o terceiro passo tendo o objetivo de selecionar apenas os nós que possuem o conteúdo parecido com o desejado, se a estrutura de uma página *web* for descentralizada, uma sessão ou um parágrafo de todo o conteúdo pode ser extraído. O texto também deve ser extraído de nós adjacentes que atendam às condições do passo 3. Com isso, todo o texto será extraído de nós qualificados do mesmo nível.

Gondse e Raut (2014) mostraram que utilizando a manipulação da árvore DOM é possível fazer a extração de conteúdo das páginas *web* filtrando os conteúdos não

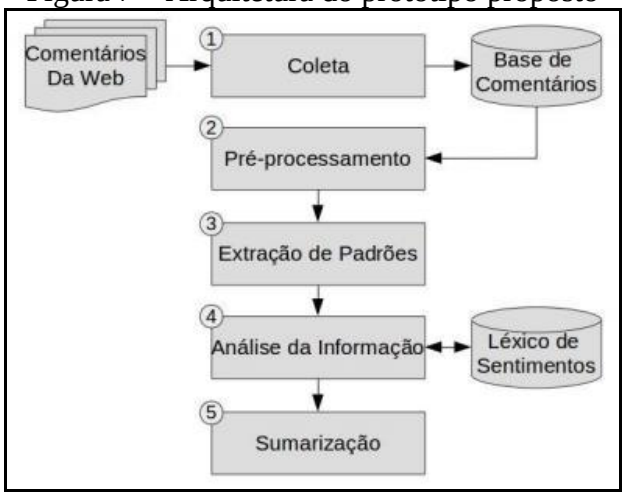
informativos, além de poder navegar pela estrutura, adicionar, modificar ou remover elementos e o conteúdo das páginas *web*.

2.5.3 Análise de web reviews sobre produtos ou serviços

Vieira et al. (2015) propuseram um protótipo para coletar automaticamente comentários sobre determinado produto e analisá-los para identificar as principais características e o nível de aceitação do produto. A coleta destes comentários é feita em *sites* de comparação de preços. No caso deles, ficaram restritos ao *site* Buscapé.

O protótipo Vieira et al. (2015) tem como entrada a URL de um produto contido no *site* Buscapé e a partir desta entrada os comentários são extraídos. Estes comentários passam por algumas etapas, desde o pré-processamento, extração de padrões até a sumarização, conforme ilustra a Figura 7.

Figura 7 – Arquitetura do protótipo proposto



Fonte: Vieira et al. (2015, p. 03).

A etapa de coleta consiste em capturar os dados na *web* para compor a base de dados textuais. Nesta atividade, utilizou-se "rastreadores da *Web*" (*Web crawlers*) para minerar o conteúdo HTML de uma dada URL que possui determinado produto ou serviço (VIEIRA et al., 2015, p. 03). Nesta etapa, foi utilizado a biblioteca JSoup que converte o conteúdo HTML em dados indexáveis. Para realizar esta conversão, uma URL é passada como entrada e a partir de seletores CSS um vetor é retornado contendo os elementos que se enquadram na busca feita através dos seletores definidos. Após a etapa de coleta, os dados são pré-processados utilizando técnicas de Processamento de Linguagem Natural e, posteriormente, na etapa de extração de padrões foram identificados os padrões sintáticos que sugerem os pares <característica, palavra opinativa>.

Na etapa de extração de padrões, os dados são submetidos à etapa de análise da informação, onde a polaridade das palavras opinativas de cada tupla <característica, palavra opinativa> foram definidas através do uso de um léxico de sentimentos. Cada palavra do léxico de sentimentos possui uma polaridade associada que pode ser: positiva (1), negativa (-1) ou neutra (0) (VIEIRA et al., 2015, p. 04). Desta forma, o somatório das palavras indicará se a polaridade do comentário será positiva, negativa ou neutra. Na última etapa, a sumarização, um documento com os resultados encontrados é gerado contendo o comentário, os padrões linguísticos e a polaridade do sentimento dos padrões e do comentário.

Segundo Vieira et al. (2015, p. 06), os resultados obtidos foram satisfatórios. A partir de testes realizados em uma base de dados com 2000 comentários (1000 positivos e 1000 negativos) o melhor resultado obteve uma precisão de 74,30% em comparação com a avaliação dos comentários realizada pelo *site* Buscapé.

3 DESENVOLVIMENTO

Este capítulo demonstra as etapas de desenvolvimento da aplicação. A seção 3.1 apresenta os requisitos funcionais e não funcionais. Na seção 3.2 está a especificação da solução. A seção 3.3 apresenta as etapas da implementação da aplicação. Nela, encontram-se imagens das telas da aplicação *mobile*, trechos de código e detalhes do desenvolvimento do *web service*.

3.1 REQUISITOS

Nesta seção os Requisitos Funcionais (RF) e os Requisitos Não Funcionais (RNF) do *web service* estão sendo representados pelo Quadro 1 e Quadro 2 respectivamente, já os Requisitos Funcionais e os Requisitos Não Funcionais da aplicação *mobile* são representados no Quadro 3 e Quadro 4.

Quadro 1 – Requisitos Funcionais do *web service*

Requisitos funcionais	**Casos de uso**
RF 01: permitir que o administrador armazene algumas categorias padrões que irão aparecer como opções de escolha para o usuário no aplicativo.	UC07
RF 02: permitir que o administrador armazene as lojas que terão as categorias e ofertas extraídas.	UC04
RF 03: o *web service* deve extrair todas as categorias das lojas armazenadas.	UC09
RF 04: o *web service* deve extrair as ofertas correspondentes às categorias padrões.	UC08

Fonte: elaborado pelo autor.

Quadro 2 – Requisitos Não Funcionais do *web service*

Requisitos não funcionais
RNF 01: utilizar a arquitetura Representational State Transfer (REST).
RNF 02: retornar os dados solicitados em formato JavaScript Object Notation (JSON).
RNF 03: extrair os dados das páginas *web* através de seletores CSS.
RNF 04: o *web service* deve ser desenvolvido em Node.js.
RNF 05: o *web service* deve utilizar o banco de dados MySQL.
RNF 06: as ofertas devem ser extraídas no mesmo período todo dia para mantê-las sempre atualizadas na base de dados.
RNF 07: deve ser utilizada a versão ECMAScript 6 do JavaScript para o desenvolvimento do *web service*.

Fonte: elaborado pelo autor.

Quadro 3 – Requisitos Funcionais da aplicação *mobile*

Requisitos funcionais	**Casos de uso**
RF 01: permitir que o usuário consiga selecionar as categorias ao qual deseja receber as ofertas.	UC01
RF 02: permitir que o usuário consiga selecionar as lojas ao qual as ofertas serão extraídas.	UC01
RF 03: permitir que o usuário possa criar perfis com diferentes categorias e lojas.	UC01
RF 04: permitir que o usuário possa buscar as ofertas de acordo com o perfil selecionado, sintetizando-as.	UC02
RF 05: permitir que o usuário possa ouvir as ofertas que já foram consultadas.	UC03
RF 06: permitir que o usuário possa acessar a página de detalhes da oferta que está sendo sintetizada.	UC02 e UC03

Fonte: elaborado pelo autor.

Quadro 4 – Requisitos Não Funcionais da aplicação *mobile*

Requisitos não funcionais
RNF 01: a aplicação *mobile* deve ser desenvolvida para a plataforma Android.
RNF 02: a aplicação *mobile* deve ser desenvolvida com React Native.
RNF 03: a aplicação *mobile* deve utilizar a biblioteca Redux.
RNF 04: a navegação da aplicação *mobile* deve ser feita utilizando a biblioteca React Navigation.
RNF 05: a aplicação *mobile* deve utilizar o banco de dados SQLite para armazenamento local.
RNF 06: consumir pouca memória do smartphone e banda de internet.
RNF 07: possibilitar que o usuário consiga fazer conexão bluetooth com o rádio do carro para ouvir as ofertas nele.

Fonte: elaborado pelo autor.

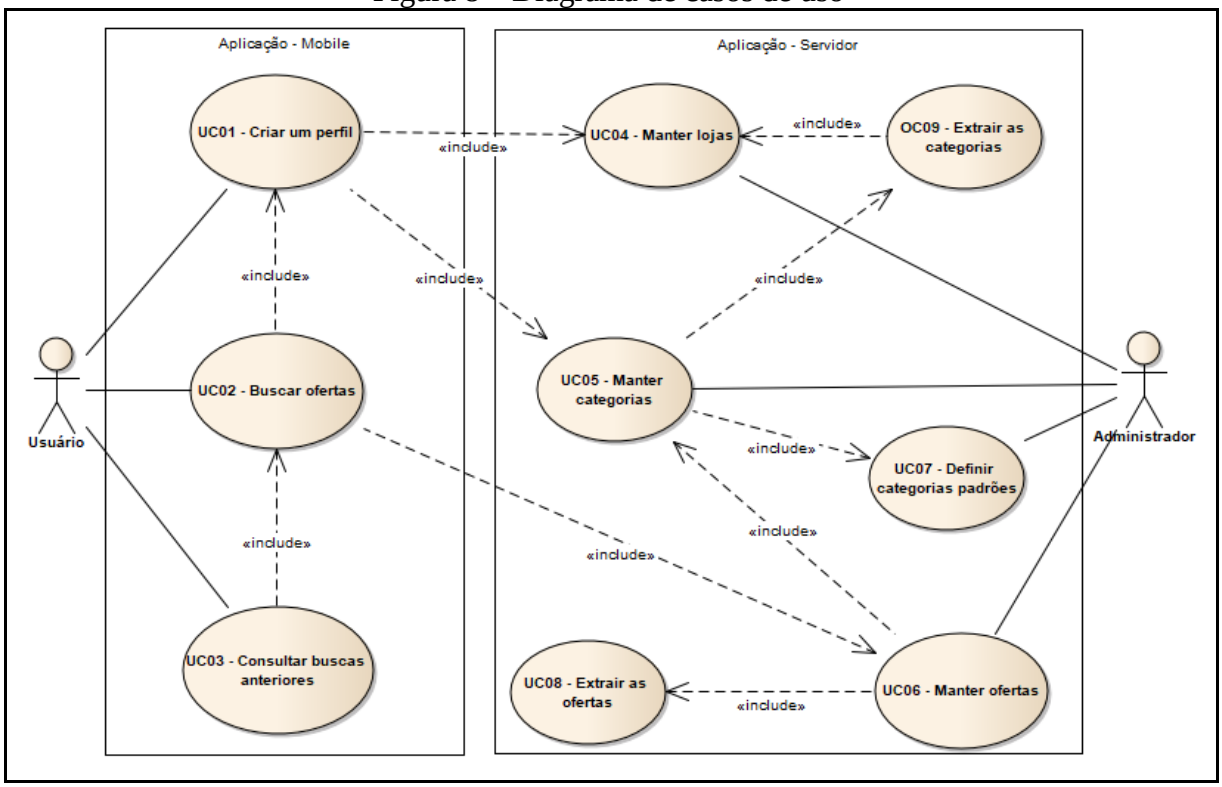
3.2 ESPECIFICAÇÃO

A especificação deste trabalho foi desenvolvida utilizando o diagrama de casos de uso, desenvolvido com a ferramenta Enterprise Architect e o Modelo de Entidade Relacionamento (MER), disponível no Apêndice B, que foi desenvolvido através da ferramenta MySQL Workbench.

3.2.1 Diagrama de casos de uso

Nesta seção é apresentado o diagrama de casos de uso ilustrado na Figura 8, que representa as funcionalidades dos atores `Usuário` e `Administrador`. O ator `Usuário` é responsável pelas ações disponíveis no aplicativo utilizado no smartphone enquanto que o `Administrador` é responsável por operar as ações requisitadas pelo aplicativo ao servidor. O detalhamento do diagrama de casos de uso encontra-se no apêndice A

Figura 8 – Diagrama de casos de uso



Fonte: elaborado pelo autor.

Para que as ofertas dos produtos possam ser extraídas o `Administrador` precisa armazenar as lojas de onde serão feitas a extração. Isso é feito através do caso de uso `UC04 – Manter lojas`. O caso de uso `UC07 – Definir categorias padrões` é utilizado pelo `Administrador` para definir as categorias padrões que irão aparecer como opções de escolha no aplicativo. Essas categorias serão utilizadas como filtro no caso de uso `UC09 – Extrair as categorias`. Tendo as categorias extraídas, o `Administrador` pode salvá-las no banco de dados através do caso de uso `UC05 – Manter categorias`. A partir delas, a aplicação conseguirá extrair as ofertas de todas as URLs vinculadas à categoria escolhida pelo `Usuário`. Este procedimento é feito através do caso de uso `UC08 – Extrair as ofertas`. Através do caso de uso `UC06 – Manter ofertas`, as ofertas extraídas são salvas no banco de dados. É importante ressaltar que a rotina de extração das ofertas é executada em *background* para que as ofertas estejam sempre atualizadas antes de serem consumidas pelo aplicativo.

A interação do `Usuário` começa com a criação de um perfil através do caso de uso `UC01 – Criar um perfil`, onde ele define o nome do perfil, as categorias que deseja obter as ofertas e as lojas. Tendo seu perfil criado, o `Usuário` pode ouvir as ofertas através do caso de uso `UC02 – Buscar ofertas`. É a partir deste caso que o aplicativo passará para o *web service* (servidor) as categorias e lojas vinculadas ao perfil selecionado. Por sua vez, o *web service* retornará a lista de ofertas para que sejam sintetizadas pelo aplicativo. Após realizar a

consulta das ofertas, o `Usuário` poderá visualizá-las novamente através do caso de uso `UC03 – Consultar buscas anteriores`.

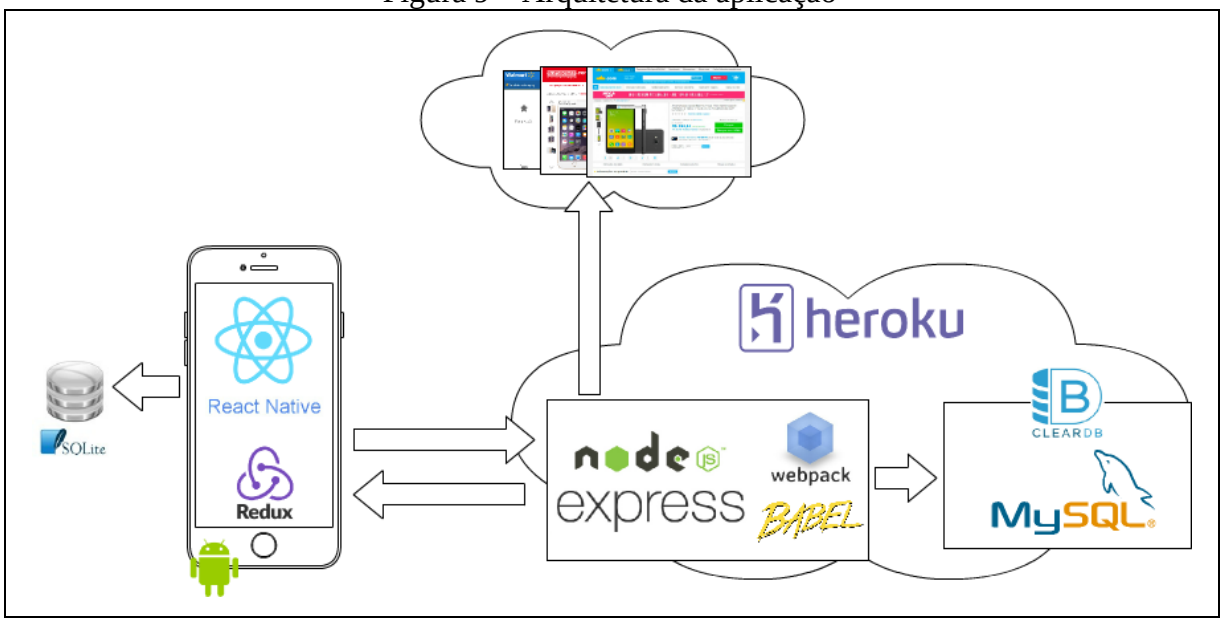
3.3 IMPLEMENTAÇÃO

Nesta seção são mostradas as técnicas e ferramentas utilizadas para o desenvolvimento do aplicativo utilizado pelo usuário e do *web service* responsável pela extração das ofertas. Para o armazenamento dos dados da aplicação foi desenvolvida uma base de dados representada pelo Modelo de Entidade Relacionamento (MER) do Apêndice B.

3.3.1 Técnicas e ferramentas utilizadas

O desenvolvimento deste trabalho é dividido em duas etapas, a primeira é o desenvolvimento do *web service* e a segunda é o desenvolvimento do aplicativo para smartphone. Tanto o *web service* quanto o aplicativo foram desenvolvidos em JavaScript utilizando a Integrated Development Environment (IDE) Visual Studio Code e o editor de texto Sublime Text. A Figura 9 representa a arquitetura de toda a aplicação, juntamente com todas as ferramentas utilizadas.

Figura 9 – Arquitetura da aplicação



Fonte: elaborado pelo autor.

Para o desenvolvimento do *web service* foi utilizada a plataforma Node.js juntamente com o *framework* Express. Como foram utilizadas algumas características do JavaScript suportadas apenas na versão ES6 (EcmaScript 6) e a versão utilizada do Node.js, versão 7.3.0, não possui suporte, foi necessário utilizar a ferramenta Babel para transformar o código escrito em ES6 para um código compatível com a versão EcmaScript 5. Para a instalação das

dependências utilizadas para a extração dos dados foi utilizado o Node Package Manager (NPM).

As dependências instaladas para a extração dos dados das lojas são o PhantomJS, Request e Cheerio. A biblioteca Cheerio é utilizada para poder extrair elementos HTML de uma página web. O PhantomJS e o Request são utilizados para o mesmo propósito, que é o de recuperar o conteúdo HTML da página de uma URL específica. Porém, a principal diferença entre eles é que o PhantomJS interpreta JavaScript, ou seja, ele consegue recuperar o conteúdo de páginas web renderizadas no lado do cliente. Esse tipo de página é bastante comum hoje em dia devido ao uso de bibliotecas como o React, Vue e *frameworks* como o AngularJS, que renderizam elementos HTML através do JavaScript.

Para hospedar o *web service* foi utilizado o Heroku, que é uma plataforma como um serviço na nuvem que permite a integração com o GitHub para realizar o *deploy* da aplicação de forma simples. O conteúdo extraído dos sites é salvo em uma base de dados MySQL criada através do *Add-on* ClearDB MySQL instalado no Heroku. Para unir todos os arquivos JavaScript que compõem o *web service* e mandar para o Heroku através do *deploy*, foi utilizado o Webpack, que é um empacotador de módulos para aplicações JavaScript, ou seja, ele gera um *bundle* com todos os arquivos utilizados para o funcionamento da aplicação.

Para o desenvolvimento do aplicativo para smartphone foi utilizado o React Native, que é um *framework* multiplataforma desenvolvido pela equipe do Facebook onde utiliza-se a biblioteca React, também desenvolvido pela equipe do Facebook, para desenvolver seus componentes. Para a sintetização das ofertas foi instalado o plugin `react-native-tts` e para salvá-las no banco local do smartphone foi instalado o plugin `react-native-sqlite-storage`. Esses plugins também foram instalados utilizando o gerenciador de dependências NPM.

O estado da aplicação é controlado pela biblioteca Redux, que mantém o estado em uma *store*, onde as alterações no estado são feitas por meio de ações disparadas e tratadas por funções puras denominadas de *reducers*. Para controlar a navegação entre as telas do aplicativo foi utilizado a biblioteca React Navigation.

Com o React Native a aplicação é desenvolvida utilizando o React. Porém, quando é feito o *build*, o *framework* gera código nativo da plataforma escolhida, para isso basta acessar o diretório da aplicação e executar no terminal o comando `react-native run-android` e com isso é gerado o .apk do aplicativo.

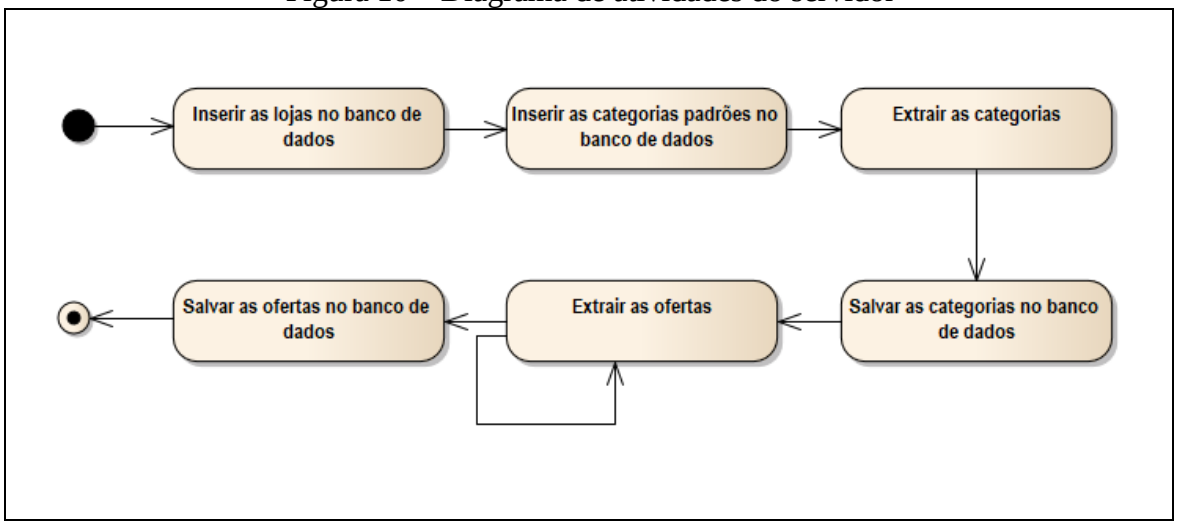
3.3.2 Etapas do desenvolvimento

Nesta seção encontram-se os trechos de códigos mais relevantes da aplicação, com seus devidos detalhamentos. A seção 3.3.2.1 apresenta o desenvolvimento do *web-service*, responsável pela extração das ofertas. A seção 3.3.2.2 aborda o desenvolvimento da aplicação mobile. Por fim, a seção 3.3.2.3 apresenta como foi feita implementação da comunicação entre o aplicativo e o rádio do carro.

3.3.2.1 Web service

Para melhor compreensão do fluxo do *web service*, o diagrama de atividades da Figura 10 demonstra os passos necessários para realizar a extração das ofertas em sites escolhidos pelo usuário.

Figura 10 – Diagrama de atividades do servidor

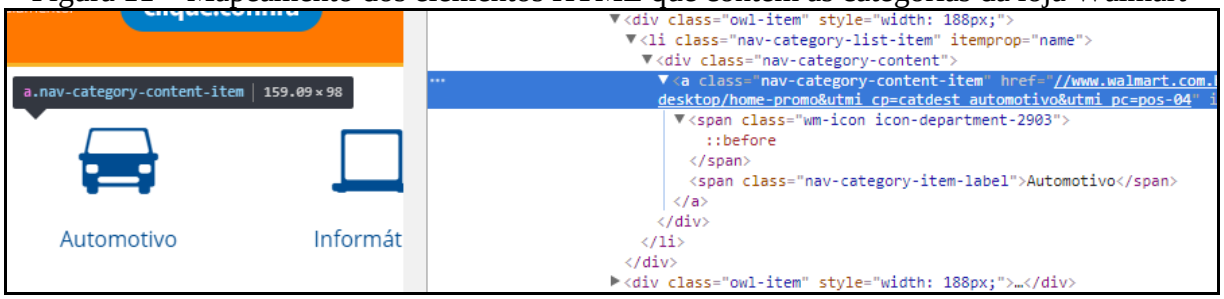


Fonte: elaborado pelo autor.

A primeira etapa realizada no desenvolvimento do *web service* foi a inserção, no banco de dados, das lojas de onde serão extraídas as ofertas. Após incluir as lojas, é feita a inserção das categorias que irão aparecer para o usuário no aplicativo. Estas inserções são feitas pelo `Administrador` do sistema no momento da criação da base de dados.

Para fazer a extração das categorias é necessário fazer o mapeamento dos elementos HTML dos sites escolhidos. Neste caso, procura-se todas as *tags* `<a />` que contém a classe `.nav-category-content-item`. Para obter a URL da categoria será preciso pegar o conteúdo do atributo `href` e para obter o nome dela é preciso pegar o conteúdo do elemento `<span />` que contém a classe `.nav-category-item-label`, conforme pode ser visto na Figura 11.

Figura 11 – Mapeamento dos elementos HTML que contém as categorias da loja Walmart



Fonte: elaborado pelo autor.

Após ter mapeado as categorias, foi utilizado a biblioteca `Request` juntamente com a biblioteca `Cheerio`, que faz uso dos seletores `CSS` para poder extrair as informações dos elementos `HTML`, conforme pode ser visto no Quadro 5.

Quadro 5 – Função para extrair as informações das categorias da loja Walmart

```
1  getCategories = () => new Promise((resolve, reject) => {
2      const categories = [];
3      request(this.options, (error, response, body) => {
4        if (error) reject(error);
5
6        const $ = cheerio.load(body);
7        const $listItems = $('.nav-category-bar .nav-category-list .nav-category-content-item');
8        $listItems.each((index, item) => {
9          const categoryName = $(item).find('.nav-category-item-label').text();
10         const categoryURL = $(item).attr('href');
11
12         categories.push({
13           name: removeAccentuation(categoryName),
14           url: formatURL(categoryURL),
15           store: getStores.WALMART,
16         });
17       });
18       resolve(categories);
19     });
20   });
```

Fonte: elaborado pelo autor.

Na linha 3 pode-se perceber que a função `request` recebe 2 parâmetros, o primeiro é um objeto contendo as configurações (URL da página) e o segundo parâmetro é uma função que será chamada após a requisição Hypertext Transfer Protocol (HTTP). O conteúdo da página requisitada é recuperado através do parâmetro `body`. De posse do conteúdo da página, é utilizada a biblioteca `Cheerio` para extrair os elementos HTML utilizando os seletores CSS. Na linha 7 todos os elementos que possuem a classe `.nav-category-content-item` e que estão dentro dos elementos com as classes `.nav-category-bar` e `.nav-category-list` são selecionados e salvos na variável `$listItems`.

A partir da linha 8 o nome e a URL de cada categoria contida na lista `$listItems` são recuperados e inseridos na lista `categories` que será salva no banco de dados.

Com as categorias já extraídas e salvas, as próximas etapas são a de extrair e salvar as ofertas. Porém, um dos problemas encontrados ao fazer a extração dos dados nas lojas é que

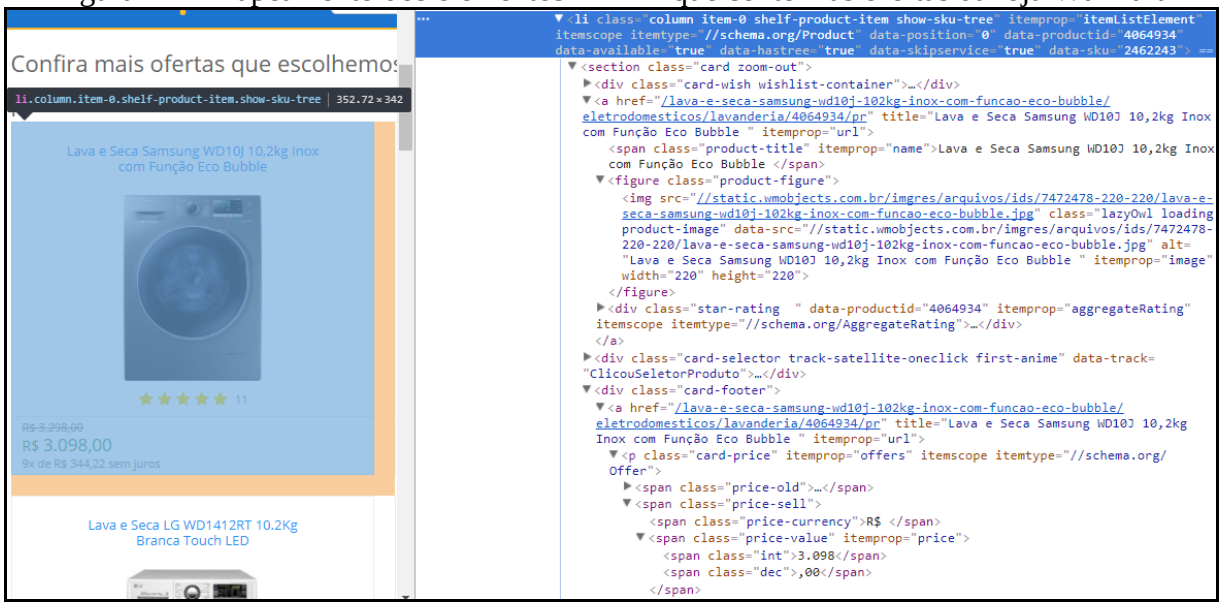
seus sites são desenvolvidos utilizando a biblioteca `React`. Portanto, a renderização do conteúdo HTML é feita no lado do cliente (*client-side rendering*).

Inicialmente foi utilizada a biblioteca `Cheerio` para recuperar o HTML das páginas das lojas. Porém, como ela não interpreta o JavaScript da página, todos os elementos HTML renderizados através do JavaScript não podiam ser extraídos. Por conta disso, optou-se por utilizar o `phantom` que é um módulo que integra a biblioteca `PhantomJS` com o Node.js. Com isso é possível interpretar o JavaScript das páginas e por sua vez, recuperar todo o conteúdo HTML das páginas.

Para se obter as ofertas, realizou-se o mesmo procedimento feito para extração das categorias, ou seja, inicialmente elas foram mapeadas e posteriormente todos os elementos com a classe `.shelf-product-item` foram selecionados, pois dentro destes elementos estão as informações das ofertas.

Para obter o nome de cada produto, foi preciso recuperar o conteúdo do elemento que possui a classe `.product-title`. O preço dos produtos foi obtido através dos elementos que possuem as classes `.int` e `.dec`. Para obter a URL de cada produto foi pego o conteúdo do atributo `href` da *tag* `<a />` que engloba a imagem da oferta, como pode ser visto pela Figura 12.

Figura 12 – Mapeamento dos elementos HTML que contém as ofertas da loja Walmart



Fonte: elaborado pelo autor.

Através de uma função recursiva, o conteúdo HTML correspondente à URL de cada categoria é recuperado através do módulo `phantom` e, a partir dele, as informações das ofertas são extraídas, como pode ser visto no Quadro 6.

Quadro 6 – Função para extrair as informações das ofertas da loja Walmart

```
1  getProductsList = (html, category) => new Promise((resolve, reject) => {
2      const $ = cheerio.load(html);
3      const productsContainers = $('.product-list');
4      const productsList = [];
5
6      try {
7        productsContainers.each((index, container) => {
8          $(container).find('.shelf-product-item').filter((index, product) => {
9            return $(product).find('.product-title').text().length > 0;
10         }).slice(0, 10).each((index, product) => {
11           const name = $(product).find('.product-title').text();
12           const priceInt = $(product).find('.price-value .int').text();
13           const priceDec = $(product).find('.price-value .dec').text();
14           const url = $(product).find('a:first-of-type').attr('href');
15
16           if (name.length > 0 && priceInt.length > 0) {
17             productsList.push({
18               name: scapeQuot(name),
19               url: formatURL(url),
20               price: `R$ ${priceInt}${priceDec}`,
21               category: category.categoryAppId,
22               store: category.idStore
23             });
24           }
25         });
26       });
27
28       resolve(productsList);
29     } catch (err) {
30       reject(err);
31     }
32   });
```

Fonte: elaborado pelo autor.

Na linha 1 a função `getProductList` recebe dois parâmetros, onde o primeiro é o conteúdo HTML da página correspondente à categoria e o segundo é um objeto contendo as informações da categoria cujas ofertas serão extraídas. Como na página pode haver mais de uma seção de produtos, por exemplo, “Mais Vendidos”, “os mais populares na categoria”, “destaques da categoria”, entre outros, as seções são identificadas na linha 3 pela classe `.product-list`.

A partir da linha 6 todos os elementos que possuem a classe `.shelf-product-item` são percorridos e as informações como nome, URL e preço são extraídas através dos seletores CSS. Na linha 17 cada oferta é inserida na lista `productList` que depois será retornada e salva na linha 28.

A rotina que extrai as ofertas das categorias é executada todo dia. Desta forma, a base sempre terá as ofertas das lojas atualizadas para serem consultadas pelos usuários da aplicação no smartphone. Para isso foi utilizado o módulo `node-cron` que permite executar uma tarefa programada em um período específico. Neste caso, todo dia à meia noite, como pode ser visto no Quadro 7.

Quadro 7 – Configuração do período de execução da rotina de extração das ofertas

```
1  import CronJob from 'node-cron';
2  import ApplicationController from './../controllers/Application';
3
4  CronJob.schedule('0 0 * * *', function(){
5    ApplicationController.saveOffers();
6  });
```

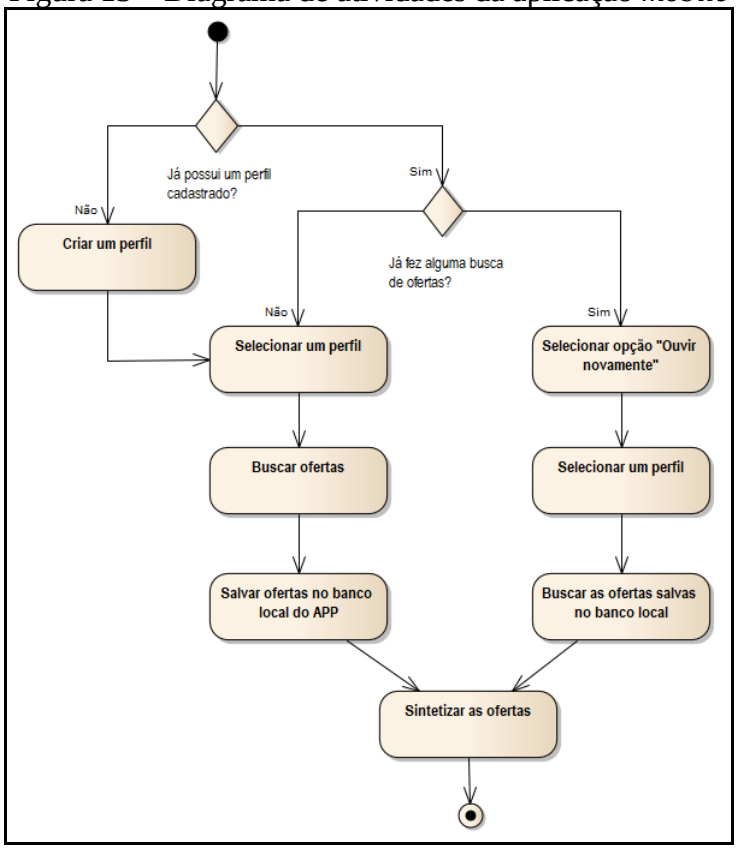
Fonte: elaborado pelo autor.

Na linha 4 é feita a configuração do período de execução da tarefa, os dois primeiros valores indicam os minutos e as horas respectivamente e os demais valores são os dias do mês o mês e o dia da semana.

3.3.2.2 Aplicação Mobile

A aplicação mobile tem como principal objetivo sintetizar as ofertas extraídas para o usuário de acordo com as categorias e lojas que ele escolheu. Para isso, ele deve seguir alguns passos, conforme o diagrama de atividades ilustrado pela Figura 13.

Figura 13 – Diagrama de atividades da aplicação *mobile*

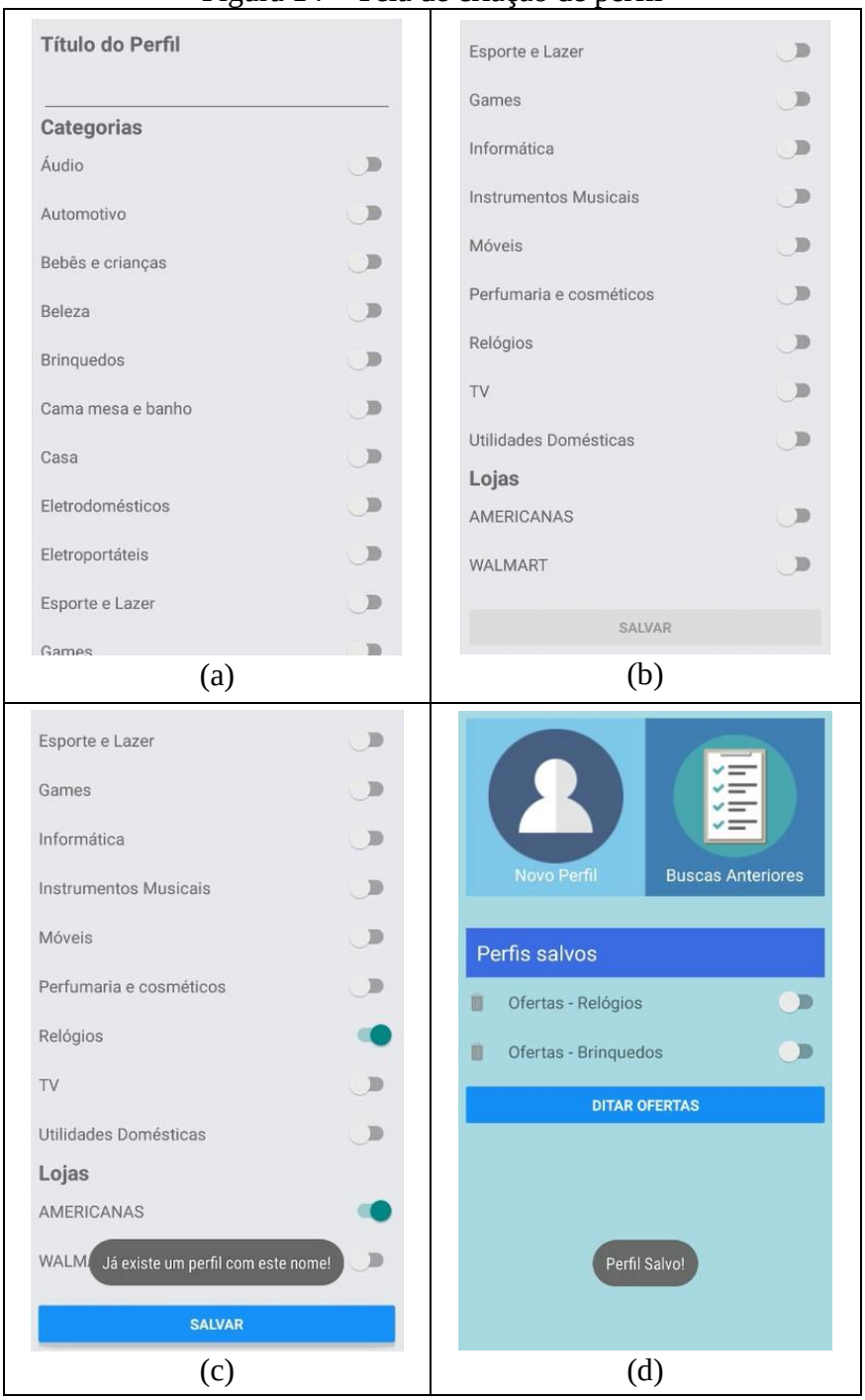


Fonte: elaborado pelo autor.

Na primeira execução da aplicação, as tabelas da base de dados local do dispositivo são criadas e as categorias/lojas cadastradas pelo administrador na criação do *web service* são recuperadas e listadas para o usuário quando ele definir o seu perfil. Para a criação da base local no smartphone foi utilizado o módulo `react-native-sqlite-storage`. Contudo,

primeiramente o usuário deve criar um perfil, no qual ele definirá um nome para o perfil, as categorias e de quais lojas ele deseja selecionar as ofertas, como pode ser visto na Figura 14(a) e (b).

Figura 14 – Tela de criação de perfil



Fonte: elaborado pelo autor.

O usuário não conseguirá salvar um perfil com um nome que já existe, como pode ser visto na Figura 14 item (c). Após salvar um perfil válido ele será redirecionado para a tela inicial da aplicação, como pode ser visto na Figura 14 item (d), onde poderá ter as ofertas relacionadas ao perfil escolhido, sintetizadas.

Quando o usuário clicar no botão `ditar ofertas`, uma requisição GET é feita para o *web service* enviando dois parâmetros, `categories` e `stores`, que correspondem às listas de categorias e lojas vinculadas ao perfil selecionado. Com base nisso, o *web service* retorna um JSON contendo uma lista com as ofertas que foram extraídas e salvas no banco de dados. A lista retornada pelo *web service* é mantida em memória através da biblioteca `Redux`, que mantém o estado da aplicação em um único objeto denominado *store*. Nela, ficam todas as informações da aplicação. Um exemplo da estrutura utilizada pelo `Redux` pode ser visto no Quadro 8.

Quadro 8 - Estruturas utilizadas pelo Redux

Constants

```
1  export const FETCH_OFFERS = 'FETCH_OFFERS';
2  export const FETCH_OFFERS_SUCCESS =
   'FETCH_OFFERS_SUCCESS';
3  export const FETCH_OFFERS_ERROR =
   'FETCH_OFFERS_ERROR';
4  export const SAVE_OFFERS_SEARCHED =
   'SAVE_OFFERS_SEARCHED';
5  export const SAVE_OFFERS_SEARCHED_SUCCESS =
   'SAVE_OFFERS_SEARCHED_SUCCESS';
6  export const SAVE_OFFERS_SEARCHED_ERROR =
   'SAVE_OFFERS_SEARCHED_ERROR';
```

Action Creators

```
1  export const fetchOffersError = err => ({
2    type: types.FETCH_OFFERS_ERROR,
3    payload: err,
4  });
5
6  export const fetchOffersSuccess = offers =>
   ({
7    type: types.FETCH_OFFERS_SUCCESS,
8    payload: offers.data
9  });
10
11 export const fetchOffers = params =>
12 (dispatch) => {
13   dispatch({ type: types.FETCH_OFFERS });
14   return api.getOffers(params)
15     .then((response) => {
16 dispatch(fetchOffersSuccess(response));
17       return response;
18     })
19     .catch((err) => {
20       dispatch(fetchOffersError(err));
21       throw err;
22     });
23 };
```

Reducer

```
1  export default (state = initialState, { type,
   payload }) => {
2    switch (type) {
3      case types.APP_IS_ON_BACKGROUND:
4        return initialState;
5
6      case types.FETCH_OFFERS:
7        return {
8          ...state,
9          fetchingOffers: true,
10       };
11
12     case types.FETCH_OFFERS_SUCCESS: {
13       if (!payload.data.length) {
14         return {
15           ...state,
16           fetchingOffers: false,
17         };
18       }
19       return state;
20     }
21
22     case types.SAVE_OFFERS_SEARCHED_SUCCESS:{
23       const offers = payload.data;
24       return {
25         ...state,
26         offers: offers.map(offer => ({
27           ...offer,
28           played: false,
29         })),
30         productToPlay: {
31           ...offers[state.indexOffer],
32         },
33         indexOffer: state.indexOffer + 1,
34         fetchingOffers: false,
35         offersFetched: true,
36       };
37     }
38
39     default:
40       return state;
41   }
42 };
```

Fonte: elaborado pelo autor.

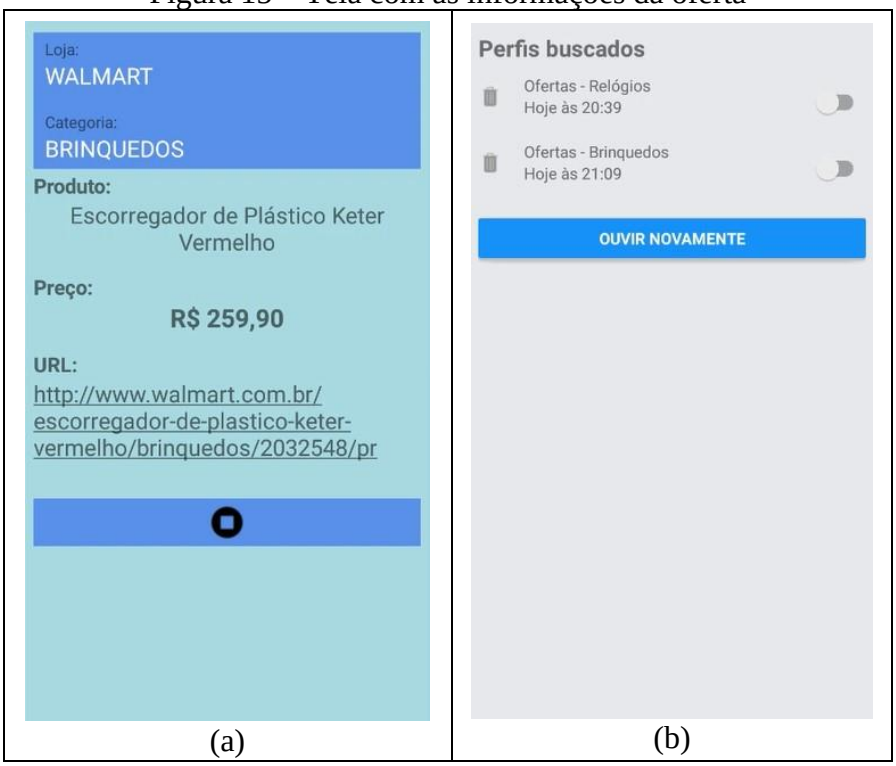
Como pode ser visto no Quadro 8, na seção *constants* são definidas as constantes que são utilizadas como o tipo das ações. Todas as ações criadas precisam de um tipo, definido como *type*, que serve para identificar que tipo de ação será executada. Por padrão, o *type* deve

ser definido como uma constante do tipo `string`. Na linha 1 a constante `FETCH_OFFERS` é definida para indicar a ação emitida quando o usuário clicar em `ditar ofertas`.

As ações obrigatoriamente precisam do *type* que define qual é o seu tipo e se elas podem ter ou não um *payload* que será o dado que ela levará para a *store*. Na seção *action creators* estão as funções que fazem a emissão das ações. Na linha 11 é definida a função `fetchOffers` que recebe como parâmetro um objeto contendo as listas de categorias e lojas escolhidas pelo usuário. Na linha 14 a função `fetchOffers` retorna a resposta da requisição GET feita para o *web service* e as ofertas retornadas são passadas como parâmetro para a função `fetchOffersSuccess` que dispara a ação `FETCH_OFFERS_SUCCESS` passando as ofertas como *payload*.

Todas as ações emitidas são tratadas pelo *reducer*, que é uma função que recebe como parâmetro o estado atual da aplicação e a ação que está sendo emitida. Dentro do *reducer* é implementado um `switch` que verifica o *type* das ações e de acordo com cada *type* ele retorna um novo estado. Quando as ofertas retornadas pelo *web service* são salvas na aplicação, é emitida a ação `SAVE_OFFERS_SEARCHED_SUCCESS` que está sendo verificada no `case` da linha 22 da seção *reducer*. Quando essa ação é emitida, um novo estado é retornado para a aplicação contendo algumas informações, dentre elas as ofertas e qual o produto que será reproduzido na sintetização. Após a aplicação receber as ofertas do *web service*, inicia-se a sintetização delas, conforme ilustra a Figura 15.

Figura 15 – Tela com as informações da oferta



Fonte: elaborado pelo autor.

A Figura 15 item (a) mostra a tela que fica visível para o usuário enquanto o título e o preço das ofertas estão sendo sintetizadas em voz. Ela permite que o usuário possa visitar a página que contém os detalhes do produto clicando no *link* da URL.

A sintetização das ofertas é feita através do módulo `react-native-tts` que é uma implementação da biblioteca `android.speech.tts`, nativa do Android. Na inicialização da tela principal, são definidas algumas configurações iniciais para utilizar este módulo, como definir a linguagem padrão, a velocidade da fala e as funções que serão chamadas ao iniciar e terminar a síntese, como demonstra o Quadro 9.

Quadro 9 – Configurações do módulo de Text To Speech

```
1  componentDidMount() {
2    if (!this.props.TTSConfigured) {
3      TTS.addEventListener('tts-start', this.voiceStarted);
4      TTS.addEventListener('tts-finish', this.voiceFinished);
5      TTS.setDefaultLanguage('pt-BR');
6      TTS.setDefaultRate(0.4);
7      this.props.actions.TTSConfigured();
9    }
10 }
```

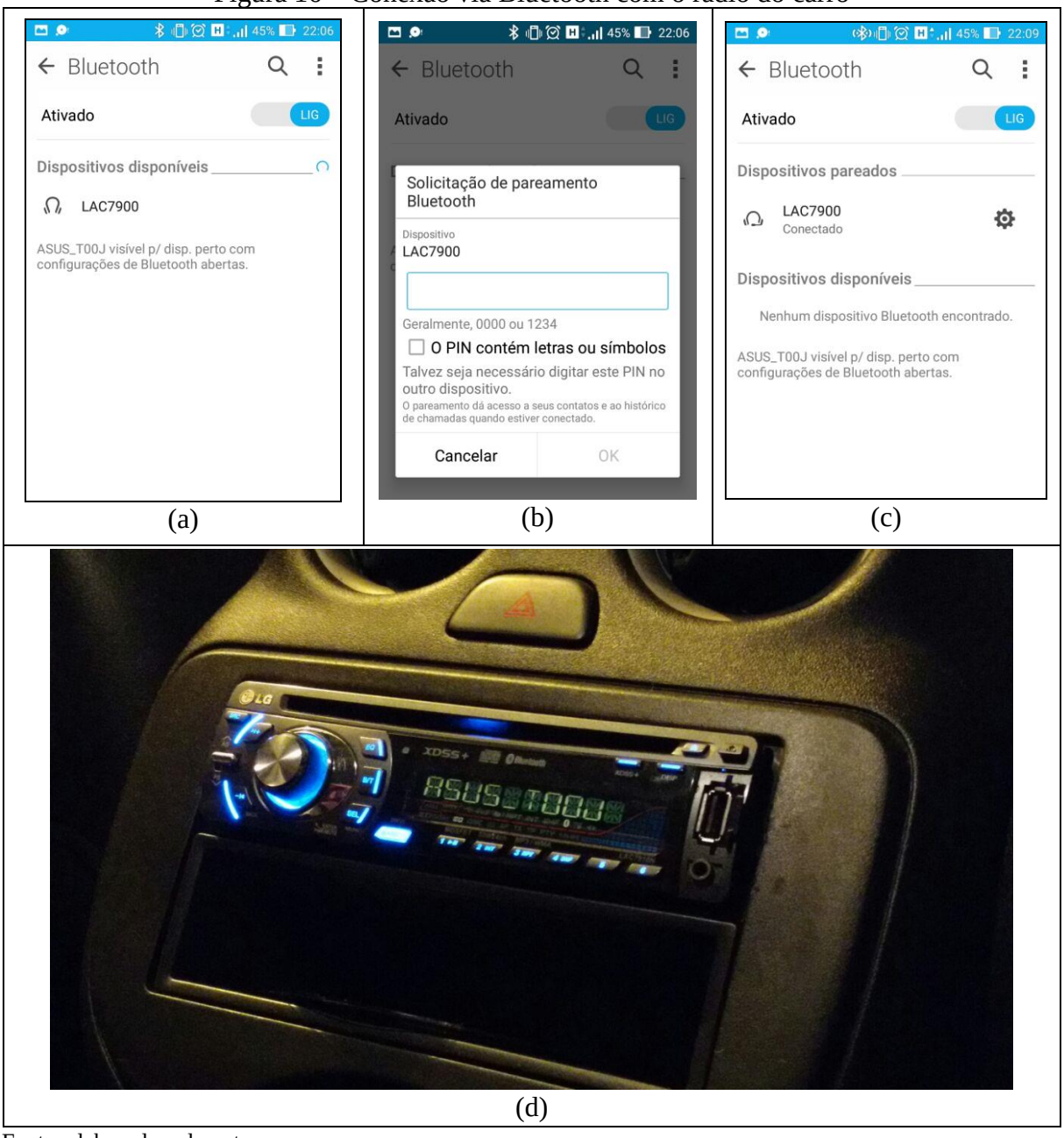
Fonte: elaborado pelo autor.

Quando a sintetização das ofertas terminar ou o usuário finalizá-la, o perfil utilizado fica salvo na tela de `Buscas Anteriores`, conforme ilustra a Figura 15 item (b). Com isso, ele poderá consultar novamente as ofertas vinculadas ao perfil selecionado sem ter que fazer uma nova requisição para o *web service*.

3.3.2.3 Comunicação com o rádio do carro

A reprodução das ofertas no rádio do carro acontecerá a partir do momento em que o smartphone estiver conectado via Bluetooth, através do pareamento com o rádio. O usuário deverá ativar o Bluetooth do smartphone para fazer o pareamento, com isso ele deverá encontrar o rádio do carro na listagem de dispositivos disponíveis e iniciar o pareamento, conforme mostra a Figura 16 item (a). Ao selecionar o rádio, o usuário deverá informar o Personal Identification Number (PIN) para parear com o rádio, esse PIN é informado pelo próprio smartphone, conforme a Figura 16 item (b) ou é informado pelo rádio.

Figura 16 – Conexão via Bluetooth com o rádio do carro



Fonte: elaborado pelo autor.

Quando os dispositivos estiverem pareados, ambos irão representar o pareamento, conforme pode ser visto na Figura 16 nos itens (c) e (d). Com isso, a partir do momento que o usuário clicar na opção `ditar ofertas` da aplicação, as ofertas já serão reproduzidas no rádio do carro.

3.4 ANÁLISE DOS RESULTADOS

Para fazer a coleta de informações a respeito dos usuários e da usabilidade da aplicação, 10 usuários testaram a aplicação e responderam os três questionários que podem ser vistos no Apêndice C. Ao longo desta seção são apresentados os resultados obtidos a partir destes questionários. Na seção 3.4.1 estão os resultados obtidos com o questionário de análise do perfil dos usuários. Na seção 3.4.2 estão os resultados a partir do questionário de atividades dos usuários. Na seção 3.4.3 estão os resultados obtidos com o questionário de usabilidade da aplicação. Por fim, a seção 3.4.4 faz uma comparação entre as características do trabalho desenvolvido e os trabalhos correlatos.

3.4.1 Análise do perfil dos usuários

No Quadro 10 estão as respostas dos usuários que responderam o questionário do perfil dos usuários.

Quadro 10 – Questionário para a análise do perfil dos usuários

Qual o seu sexo?	60% masculino 40% feminino
Qual sua idade?	80% entre 18 e 30 anos 20% entre 30 e 45 anos
O que costuma fazer quando fica parado no congestionamento?	80% ouve música 60% ouve notícias no rádio 50% acessa as redes sociais 10% visita sites de compra
Você costuma ouvir rádio enquanto dirige ou está de carona?	100% sim
Você já recebeu alguma multa por utilizar o celular enquanto dirige?	100% não
Você tem o costume de fazer compras online?	90% sim 10% não
Você costuma fazer pesquisas de preço online antes de efetuar a compra?	90% sim 10% não
Quais categorias de produto você mais compra?	10% não compra 40% eletrônicos 30% roupas 10% informática 40% livros

Fonte: elaborado pelo autor.

Com base no Quadro 10, percebe-se que a maioria dos usuários tem idade entre 18 e 30 anos e que 60% deles utilizam o rádio do carro para ouvir notícias e 80% ouvem músicas

enquanto estão no congestionamento. Ainda é possível afirmar que todos têm costume de ouvir o rádio do carro e que 90% fazem pesquisas de preço online quando vão fazer compras online. Segundo os usuários, as categorias de produtos mais compradas são eletrônicos (40%), roupas (30%) e livros (40%).

3.4.2 Análise das atividades do usuário

Após os usuários terem respondido o questionário de análise dos seus perfis, eles tiveram que responder o questionário de atividades, onde eles tinham que baixar o aplicativo e executar algumas atividades. A partir disso a Tabela 1 mostra os resultados obtidos com as respostas daqueles que testaram a aplicação.

Tabela 1 – Questionário de atividades dos usuários

Atividades	Percentual (%) de conclusão
Instalação do aplicativo	70% - Sim 20% - Não 10% - Não responderam
Criar um perfil	70% - Sim 10% - Não 20% - Não responderam
Buscar as ofertas	70% - Sim 10% - Não 20% - Não responderam
Acessar a página de detalhe do produto	70% - Sim 10% - Não 20% - Não responderam
Consultar buscas efetuadas	70% - Sim 10% - Não 20% - Não responderam
Exclusão dos registros	70% - Sim 10% - Não 20% - Não responderam
Criar um perfil	70% - Sim 10% - Não 20% - Não responderam

Fonte: elaborado pelo autor.

A partir da Tabela 1, é possível perceber que dos usuários que testaram a aplicação, pelo menos 70% deles conseguiram realizar todas as atividades com sucesso. Porém, alguns usuários fizeram observações quanto às funcionalidades do aplicativo. Um dos pontos levantados é que ao criar um perfil com alguma informação errada, o usuário não conseguia editar as informações do perfil. Ele teria que excluir o perfil errado ou criar um novo, pois atualmente a aplicação não permite editar um perfil já criado. Outro ponto mencionado foi que algumas ofertas traziam informações desnecessárias para o usuário, como o modelo do

produto, código, entre outras, isto ocorre pois não foi realizado nenhum tratamento para remover esse tipo de informação.

3.4.3 Análise da usabilidade do aplicativo

Após terem utilizado o aplicativo, os usuários responderam um questionário com perguntas avaliando a usabilidade da aplicação. A primeira parte deste questionário tinha por objetivo avaliar a navegabilidade, aparência e funcionamento do aplicativo. Os resultados obtidos com o questionário podem ser vistos na Tabela 2.

Tabela 2 – Questionário de usabilidade – funcionamento do aplicativo

Perguntas / Critérios de avaliação	**Concordo plenamente**	**Concordo parcialmente**	**Não concordo nem discordo**	**Discordo parcialmente**	**Discordo totalmente**
O aplicativo é simples e objetivo	50%	30%	20%		
A qualidade do áudio das ofertas é boa	30%	50%	20%		
É fácil navegar entre as telas do aplicativo	80%	20%			
O aplicativo em algum momento parou inesperadamente	10%		20%		70%
A aparência do aplicativo é boa (cores e ícones)	60%	10%	30%		

Fonte: elaborado pelo autor.

Com base nas respostas acima e nas sugestões dadas pelos usuários, o aplicativo pode ser considerado simples e objetivo. Porém, segundo os usuários, as opções podem ser mais sugestivas, assim como a qualidade do áudio das ofertas precisa ser melhorado, tendo em vista que 50% dos usuários concordaram parcialmente com a qualidade do áudio. Outro ponto que ainda pode ser melhorado é a aparência do aplicativo, levando em consideração que 60% dos usuários concordou plenamente, sendo que os demais ou não concordaram nem discordaram ou concordaram parcialmente.

Ainda neste formulário foram feitas outras perguntas mais voltadas para o entendimento do usuário quanto à proposta do aplicativo e que podem ser vistas na Tabela 3.

Tabela 3 – Questionário de usabilidade – entendimento da proposta do aplicativo

Perguntas	Percentual (%) de respostas
Você conseguiu compreender o objetivo do aplicativo?	100% - Sim
Você considera o aplicativo útil?	100% - Sim
Você achou importante salvar as ofertas que já foram vistas anteriormente?	80% - Sim 20% - Não
Você conseguiu acessar a página de detalhes do produto clicando na URL disponível na tela das informações da oferta?	80% - Sim 20% - Não
Você conseguiu interromper a reprodução das ofertas?	80% - Sim 20% - Não
Você utilizaria o aplicativo para pesquisa de preços?	90% - Sim 10% - Não
Você recomendaria o aplicativo para outras pessoas?	100% - Sim
Você acha interessante a funcionalidade de reprodução das ofertas no som do carro?	90% - Sim 10% - Não

Fonte: elaborado pelo autor.

De acordo com a Tabela 3, os usuários compreenderam bem o propósito do aplicativo além de concordarem que a funcionalidade de reprodução das ofertas no rádio do carro é importante. Além disso, eles indicaram que utilizariam o aplicativo para realizar consultas de preços e recomendariam para outras pessoas. Além destas perguntas, foi aberto um espaço para sugestões e críticas. Alguns usuários sugeriram salvar no histórico as ofertas ao invés de toda busca, pois desta forma não precisariam ouvir tudo novamente até chegar em um produto que chamou a atenção, ou senão, deveria ter alguma forma do usuário poder avançar entre as ofertas. Uma crítica foi feita em relação ao conteúdo das ofertas, pois tem informações que não são relevantes para o usuário, como por exemplo, o código do produto que vem na descrição dele.

3.4.4 Comparação com trabalhos correlatos

O Quadro 11 mostra uma comparação entre as principais características dos trabalhos correlatos e o trabalho desenvolvido.

Quadro 11 – Características dos trabalhos correlatos e do trabalho desenvolvido

Características \ Trabalhos	Trabalho desenvolvido	Singh (2012)	Gondse e Raut (2014)	Vieira et al. (2015)
Forma de extração do conteúdo	Através de seletores CSS	Através da árvore DOM	Através da árvore DOM	Através de seletores CSS
Alteração na estrutura do site interfere na extração do conteúdo	Sim	Não	Não	Sim
Salva o conteúdo extraído em documento texto	Não	Sim	Não	Não
Permite a seleção do conteúdo de interesse	Sim	Não	Sim	Não
Aplicação mobile	Sim	Não	Não	Não
Utiliza síntese de voz	Sim	Não	Não	Não

Fonte: elaborado pelo autor.

Os trabalhos de Gondse e Raut (2014) e de Singh (2012), fazem a extração do conteúdo das páginas *web* através da árvore DOM e, por isso, não sofrem interferência com a alteração da estrutura do *site*. Ao contrário do trabalho de Vieira et al. (2015) e do trabalho desenvolvido, onde ambos extraem o conteúdo utilizando os seletores CSS, porém assim como o trabalho de Gondse e Raut (2014), com o trabalho desenvolvido é possível extrair das páginas web apenas o conteúdo de interesse. O trabalho desenvolvido por Singh (2012) é o único que salva o conteúdo extraído em documento de texto.

Diferente dos demais trabalhos, o trabalho desenvolvido possui uma aplicação mobile onde o usuário tem acesso ao conteúdo extraído sintetizado em voz. Além disso, também foi desenvolvido um *web service* responsável pela extração dos dados de acordo com o perfil / site de oferta escolhido pelo usuário.

4 CONCLUSÕES

Este trabalho apresentou o desenvolvimento de uma aplicação que faz a busca de ofertas de produtos em lojas virtuais e reproduz diretamente no rádio do carro através de conexão via Bluetooth ou no próprio smartphone. Para este trabalho haviam dois objetivos específicos, dentre eles era o de disponibilizar um *web service* que seria responsável por fazer a extração das ofertas em sites informados pelo usuário e o segundo era o de converter as ofertas extraídas para um formato de áudio afim de serem reproduzidas no rádio do carro via conexão Bluetooth.

O primeiro objetivo que era o desenvolvimento do *web service* foi atendido utilizando a plataforma Node.js juntamente com o *framework* Express e outras tecnologias como o Webpack, Babel e ES6. Uma das maiores dificuldades encontradas no desenvolvimento do *web service* foi na extração das ofertas de sites onde o conteúdo é renderizado do lado do cliente (*client-side rendereing)*. A biblioteca `Cheerio` que foi inicialmente utilizada para a extração do conteúdo das páginas, não conseguia recuperar o conteúdo que é renderizado através de bibliotecas em `JavaScript`, como `React` e `AngularJS`. Para resolver este problema foi utilizada a biblioteca `PhantomJS` que é mais robusta neste sentido e consegue fazer a extração desse tipo de conteúdo.

O segundo objetivo também foi atendido, embora tenha sido utilizada uma abordagem diferente da que havia sido proposta para fazer a reprodução das ofertas no rádio do carro. Inicialmente, a ideia era de gerar um arquivo no formato de áudio com todas as ofertas extraídas, pois pensava-se que apenas desta forma o rádio do carro faria a reprodução de áudio via Bluetooth. Porém, através de testes foi visto que utilizando a biblioteca `android.speech.tts` que é nativa do Android, o rádio também fazia a reprodução quando estava pareado com o smartphone via Bluetooth. Por questões de performance optou-se por utilizar a biblioteca nativa do Android, pois assim a aplicação mobile teria apenas que passar para a biblioteca o texto que seria sintetizado em voz, sem precisar realizar nenhum tipo de processamento a mais para criar arquivos de áudio.

O desenvolvimento da aplicação mobile foi feito utilizando o *framework* React Native, onde utiliza-se a biblioteca React, desenvolvida pelo Facebook. O uso do *framework* foi muito satisfatório, tendo em vista que ele oferece alguns recursos que facilitam muito o desenvolvimento de aplicações, dentre elas é a opção de "*live reloading"*, onde não é preciso ter que gerar o "*build"* da aplicação novamente, ao identificar que houve alguma alteração ele automaticamente atualiza a aplicação para poder visualizar as alterações feitas. Além disso, o

framework é multiplataforma então caso seja necessário disponibilizar a aplicação para outras plataformas, como o iOS por exemplo, teriam que ser feitas apenas algumas alterações mais específicas da plataforma, mas o código ainda seria o mesmo.

A aplicação apresenta um comportamento satisfatório tendo em vista que o seu propósito é atendido, o usuário consegue criar perfis onde ele define as categorias e lojas no qual deseja obter as ofertas. Mesmo atendendo o que foi proposto, a aplicação possui alguns pontos em que é preciso melhorar, principalmente na extração das ofertas. Como a extração é feita através de seletores CSS mapeados a partir da estrutura dos sites, qualquer alteração feita por terceiros nesta estrutura, pode impactar na integridade da extração do conteúdo pois terá que ser feito um novo mapeamento para a obtenção das ofertas continuar funcionando. Além disso, outro ponto que é preciso melhorar na aplicação é no pré-processamento das informações, atualmente as ofertas trazem algumas informações que não são relevantes para o usuário e isso pode causar um descontentamento no usuário já que ele terá que ouvir algumas informações que não são necessárias.

De modo geral este trabalho torna-se relevante principalmente no âmbito acadêmico e tecnológico pois aborda o uso de tecnologias que podem ser mais exploradas para o desenvolvimento de aplicações mobile, tanto comerciais quanto de cunho científico, que é o caso do React Native que se torna uma alternativa ao uso do Ionic ou do Phonegap. Além disso, a extração de dados contidos na web não se limita apenas às ofertas, a ideia pode ser explorada e aplicada em diversas áreas.

4.1 EXTENSÕES

Sugerem-se as seguintes extensões para trabalhos futuros:

a) adicionar mais opções de lojas;
b) adicionar um histórico de preços das ofertas: desta forma o usuário conseguirá acompanhar as mudanças nos produtos nas diferentes lojas disponíveis;
c) permitir que as ofertas buscadas sejam enviadas por e-mail através de comando por voz: com isso o usuário não precisaria se preocupar com a exclusão de algum perfil que possui as ofertas de seu interesse, além disso o usuário não precisaria mexer no smartphone para enviar as ofertas pois ele poderá utilizar um comando por voz para fazer isto;
d) permitir o compartilhamento de ofertas específicas através de redes sociais: com isso o usuário poderia compartilhar com outras pessoas alguma oferta que fosse do seu interesse ou da outra pessoa com quem ele está compartilhando;

e) notificar o usuário caso ele esteja sem conexão com internet: desta forma o usuário não corre o risco de tentar buscar as ofertas e ficar esperando sem ter um retorno da aplicação;
f) adicionar inteligência artificial para a extração das ofertas feita pelo *web service*: desta forma não seria preciso implementar um *crawler* para cada loja.

REFERÊNCIAS

ANDANIYA, Mário; PROENÇA, Mário. **Extração de informação na web.** 2009. Disponível em: <http://seer.ufrgs.br/index.php/cadernosdeinformatica/article/view/v4n2p27-34/10926>. Acesso em: 29 out. 2016.

ARAÚJO, Fabíola Pantoja Oliveira. **Imitação da voz humana através do processo de análise-por-síntese utilizando algoritmo genético e sintetizador de voz por formantes.** 2015. 123 f. Tese (Doutorado em Engenharia Elétrica) – Programa de Pós-Graduação em Engenharia Elétrica, Universidade Federal do Pará, Belém, 2015. Disponível em: < http://www.ppgee.ufpa.br/ARQUIVOS/teses/Fabiola%20Pantoja%20Oliveira%20Araujo.pdf >. Acesso em: 31 out. 2016.

BRASIL. Lei n. 9.503, de 23 de setembro de 1997. Institui o Código de Trânsito Brasileiro. **Presidência da República**. Casa Civil. Subchefia para Assuntos Jurídicos. Brasília: 2007. Disponível em: <http://www.planalto.gov.br/ccivil_03/leis/L9503.htm>. Acesso em: 9 nov. 2016.

CARVALHO, Isadora. **Distração ao volante**. out, 2016. Disponível em: < http://quatrorodas.abril.com.br/noticias/distracao-ao-volante/>. Acesso em: 17 jun. 2017.

CETIC.br - Centro Regional de Estudos para o Desenvolvimento da Sociedade da Informação. **Uso da Internet pelo celular cresce entre os brasileiros, revela Cetic.br**. 2015. Disponível em: < http://www.cetic.br/noticia/uso-da-internet-pelo-celular-cresce-entre-os-brasileiros-revela-cetic-br/ >. Acesso em: 21 set. 2016.

CNI. **Retratos da sociedade brasileira**: mobilidade urbana**.** [S.I.]: Indicadores CNI, 2015. Disponível em: <http://arquivos.portaldaindustria.com.br/app/cni_estatistica_2/2015/10/14/195/RetratosDaSociedadeBrasileira_27_MobilidadeUrbana.pdf>. Acesso em: 18 set. 2016.

EISENMAN, Bonnie. **Learning React Native**: Building Mobile Applications with JavaScript. California: O'Reilly Media, 2016.

GONDSE, Ms. Pranjali G.; RAUT, Professor Anjali B.. Main Content Extraction From Web Page Using Dom. **International journal of advanced research in computer and communication engineering.** Bulandshahr, p. 5302-5304. mar. 2014. Disponível em: <http://www.ijarcce.com/upload/2014/march/IJARCCE5H a pranjali MAIN CONTENT EXTRACTION.pdf>. Acesso em: 18 set. 2016.

GOOGLE. **Android.speech.tts.** [S.I.], 2017. Disponível em: < https://developer.android.com/reference/android/speech/tts/package-summary.html>. Acesso em: 22 jun. 2017.

_____. **Manifesto do aplicativo**: <uses-sdk>**.** [S.I.], 2017. Disponível em: < https://developer.android.com/guide/topics/manifest/uses-sdk-element.html>. Acesso em: 22 jun. 2017.

IBGE. **Acesso à Internet e à televisão e posse de telefone móvel celular para uso pessoal.** Rio de Janeiro, 2013. Disponível em: <http://biblioteca.ibge.gov.br/visualizacao/livros/liv93373.pdf>. Acesso em: 18 set. 2016.

MANJULA R.; CHILAMBUCHELVAN A. An effective approach to extract information from web pages. **International journal of advanced research in computer and communication engineering.** p. 1770-1776. abr. 2016. Disponível em: < https://www.irjet.net/archives/V3/i4/IRJET-V3I4352.pdf>. Acesso em: 10 nov. 2016.

MARINHO, Leandro; GIRARDI, Rosario. **Mineração na Web.** São Luís: UFMA, 2003. 8 p. Disponível em: <http://paginas.fe.up.pt/~mgi03006/ARI/MineracaoNaWeb.pdf>. Acesso em: 29 out. 2016.

RESENDE, Paulo Tarso Vilela de; SOUSA, Paulo Renato de. Mobilidade urbana nas grandes cidades brasileiras: um estudo sobre os impactos do congestionamento. In: SIMPÓSIO DE ADMINISTRAÇÃO DA PRODUÇÃO, LOGÍSTICA E OPERAÇÕES INTERNACIONAIS, 2009, São Paulo. **Anais**... São Paulo: FGV-EAESP, 2009. 16 p. Disponível em: <http://www.simpoi.fgvsp.br/arquivo/2009/artigos/e2009_t00138_pcn41516.pdf>. Acesso em: 19 set. 2016.

RIBEIRO, Jussara. **Síntese computacional de voz**. 2010. 62 f. TCC (Graduação) - Curso de Engenharia Elétrica Com ênfase em Eletrônica, Escola de Engenharia Elétrica de São Carlos, da Universidade de São Paulo, São Paulo, 2010. Disponível em: < http://www.tcc.sc.usp.br/tce/disponiveis/18/180450/tce-18112011-091852/?&lang=br >. Acesso em: 31 out. 2016.

SANTOS, Rafael. **Conceitos de mineração de dados na web.** Instituto Nacional de Pesquisas Espaciais - INPE. 2009, 40 p. Disponível em: <http://www.lac.inpe.br/~rafael.santos/Docs/WebMedia/2009/webmedia2009.pdf>. Acesso em: 29 out. 2016.

SOLOMON, Michael R. **O comportamento do consumidor**: Comprando, Possuindo e Sendo. 11. ed. São Paulo: Bookman Editora LTDA, 2016. 608 p.

SPC Brasil. **Comparativo do consumo em lojas físicas x lojas virtuais**. 2015. Disponível em: <https://www.spcbrasil.org.br/uploads/st_imprensa/spc_brasil_analise_compras_on_off_maio_20151.pdf>. Acesso em: 18 jun. 2017.

SINGH, Anjali. Web Content Extraction to Facilitate Web Mining. **International journal of electronics and computer science engineering.** Bulandshahr, p. 1292-1299, 2012. Disponível em: <http://www.ijecse.org/wp-content/uploads/2012/06/Volume-1Number-3PP-1292-1299.pdf>. Acesso em: 18 set. 2016.

TEIXEIRA, Marcelo. **Uso da Internet no Brasil.** 2015. Disponível em: <https://techinbrazil.com.br/uso-da-internet-no-brasil>. Acesso em: 18 set. 2016.

VIEIRA, João Paulo A. et al. **Análise de web *reviews* sobre produtos ou serviços usando um léxico de Sentimentos**. Teresina, **[**2015]. 6 p. Disponível em: <http://www.eripi.ufpi.br/images/anais/141276.pdf>. Acesso em: 18 set. 2016.

ZUFFO, Felipe Augusto; PISTORI, Hemerson. **Tecnologia adaptativa e síntese de voz:** Primeiros Experimentos. Campo Grande, 2004, 4 p. Disponível em: < http://www.gpec.ucdb.br/pistori/publicacoes/zuffo_wsl2004.pdf >. Acesso em: 31 out. 2016.

APÊNDICE A – Detalhamento dos casos de uso

Neste apêndice encontram-se os quadros com o detalhamento de todos os casos de uso da aplicação *mobile* e do *web service*.

Quadro 12 – `UC01 Criar um perfil`

Número	001
Caso de Uso	Criar um perfil.
Ator	Usuário
Descrição	Descreve o fluxo para a criação de um perfil que é utilizado para fazer a busca das ofertas.
Pré-condições	`UC05 – Manter categorias` `UC04 – Manter lojas`
Cenário principal	1. O usuário seleciona a opção de criar um perfil. 2. O usuário é redirecionado para a tela de criação do perfil. 3. O usuário digita um título para o seu perfil. 4. O usuário seleciona as categorias que ele quer adicionar ao seu perfil. 5. O usuário finaliza o cadastro do seu perfil.

Fonte: elaborado pelo autor.

Quadro 13 – `UC02 Buscar ofertas`

Número	002
Caso de Uso	Buscar ofertas.
Ator	Usuário
Descrição	Descreve o fluxo da busca das ofertas de acordo com o perfil escolhido.
Pré-condições	`UC01 – Criar um perfil` `UC06 – Manter ofertas`
Cenário principal	1. O usuário seleciona um dos perfis criados. 2. Com o perfil selecionado o usuário pressiona o botão de ditar as ofertas. 3. A aplicação buscará no banco de dados as ofertas que correspondem à(s) categoria(s) e loja(s) vinculada(s) ao perfil selecionado. 4. Após feita a busca, a aplicação iniciará a reprodução das ofertas em sequência. 5. Enquanto uma oferta está sendo reproduzida, suas informações estarão sendo mostradas na tela para o usuário. 6. Após o usuário ou a aplicação finalizar a reprodução das ofertas o usuário será redirecionado para a tela principal.

Fonte: elaborado pelo autor.

Quadro 14 – UC03 Consultar buscas anteriores

Número	003
Caso de Uso	Consultar buscas anteriores.
Ator	Usuário
Descrição	Descreve o fluxo para o usuário ouvir novamente ofertas que já foram buscadas.
Pré-condições	UC02 – Buscar ofertas
Cenário principal	1. O usuário seleciona a opção de buscar ofertas anteriores. 2. O usuário é redirecionado para a tela de buscas anteriores contendo as buscas que já foram feitas. 3. O usuário seleciona uma das buscas já feitas e pressiona o botão ouvir novamente. 4. A aplicação buscará as ofertas armazenadas no banco local do smartphone e iniciará a reprodução delas. 5. Enquanto a oferta está sendo reproduzida, suas informações estarão sendo mostradas na tela para o usuário. 6. Após o usuário ou a aplicação finalizar a reprodução das ofertas o usuário será redirecionado para a tela principal.

Fonte: elaborado pelo autor.

Quadro 15 – UC04 Manter lojas

Número	004
Caso de Uso	Manter lojas.
Ator	Administrador
Descrição	Descreve o fluxo para salvar no banco de dados as lojas disponíveis para fazer a extração das ofertas.
Pré-condições	A estrutura HTML do site deve estar de acordo com o que foi mapeado na criação dos *crawlers.*
Cenário principal	1. O administrador insere no banco de dados as lojas disponíveis para a extração das ofertas.

Fonte: elaborado pelo autor.

Quadro 16 – UC05 Manter categorias

Número	005
Caso de Uso	Manter categorias.
Ator	Administrador
Descrição	Descreve o fluxo para salvar no banco de dados as categorias extraídas das lojas e que estão de acordo com as categorias definidas como padrões.
Pré-condições	UC07 – Definir categorias padrões UC09 – Extrair as categorias
Cenário principal	1. A aplicação acessa a URL de cada loja e faz a extração das categorias através do caso de uso UC09. 2. O caso de uso UC09 retorna uma lista contendo as categorias extraídas. 3. A aplicação salva no banco de dados apenas as categorias que estão de acordo com as que foram definidas no caso de uso UC07.

Fonte: elaborado pelo autor.

Quadro 17 – UC06 Manter ofertas

Número	006
Caso de Uso	Manter ofertas
Ator	Administrador
Descrição	Descreve o fluxo para salvar no banco de dados as ofertas extraídas de cada categoria das lojas definidas.
Pré-condições	UC05 - Manter categorias UC08 – Extrair as ofertas
Cenário principal	1. A aplicação acessa URL de cada categoria salva pelo caso de uso UC05. 2. A aplicação faz a extração das ofertas de cada categoria através do caso de uso UC08. 3. A aplicação salva no banco de dados as ofertas retornadas pelo caso de uso UC08. 4. A aplicação executa essa rotina todo dia à meia noite para manter sempre as ofertas atualizadas no banco de dados.

Fonte: elaborado pelo autor.

Quadro 18 – UC07 Definir categorias padrões

Número	007
Caso de Uso	Definir categorias padrões.
Ator	Administrador
Descrição	Descreve o fluxo para salvar no banco de dados as categorias definidas como padrão e que aparecerão como opções de escolha para o usuário no aplicativo.
Pré-condições	As categorias escolhidas devem ser comuns às lojas que terão as ofertas extraídas.
Cenário principal	1. O administrador define categorias que são comuns às lojas que terão as ofertas extraídas. 2. As categorias escolhidas aparecem no aplicativo para serem escolhidas pelo usuário.

Fonte: elaborado pelo autor.

Quadro 19 – UC08 Extrair as ofertas

Número	008
Caso de Uso	Extrair as ofertas.
Ator	Administrador
Descrição	Descreve o fluxo para a extração das ofertas na URL da categoria específica.
Pré-condições	A estrutura HTML do site deve estar de acordo com o que foi mapeado na criação dos *crawlers.*
Cenário principal	1. A aplicação acessa a URL de cada categoria salva no banco de dados através do caso de uso UC05. 2. A aplicação faz a extração das ofertas contidas na URL específica. 3. A aplicação retorna uma lista contendo as ofertas extraídas.

Fonte: elaborado pelo autor.

Quadro 20 – `UC09 Extrair as categorias`

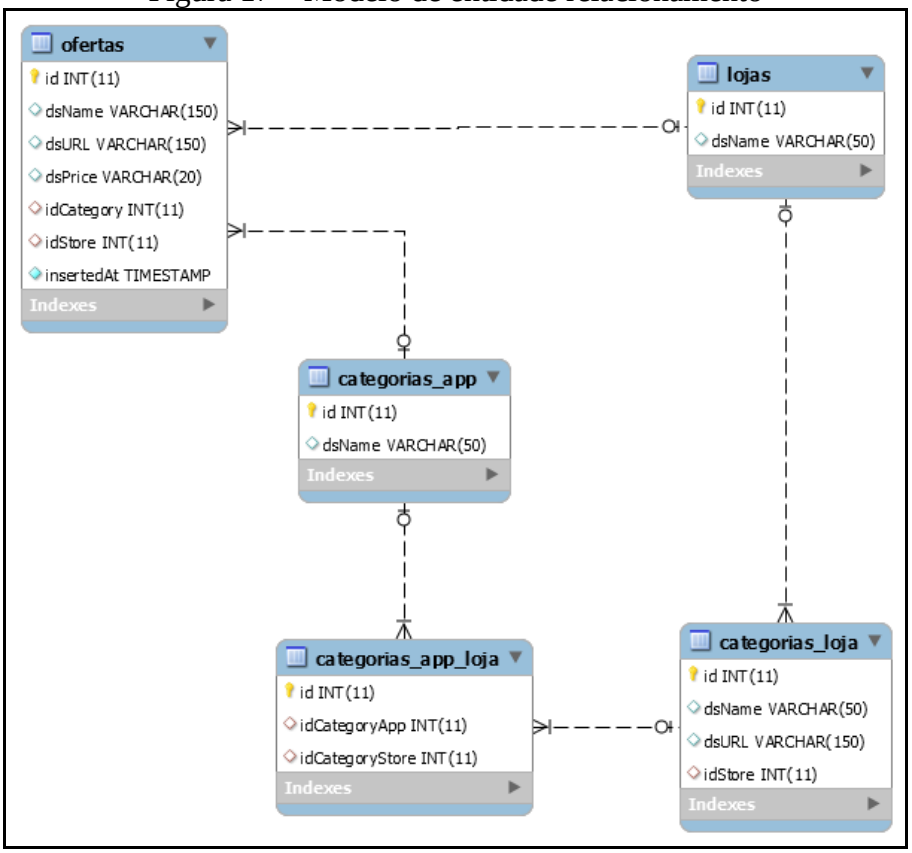
Número	009
Caso de Uso	Extrair as categorias.
Ator	Administrador
Descrição	Descreve o fluxo para a extração das categorias na URL da loja específica.
Pré-condições	A estrutura HTML do site deve estar de acordo com o que foi mapeado na criação dos *crawlers*
Cenário principal	1. A aplicação acessa o site de cada loja definida no caso de uso `UC02` e faz a extração das categorias contidas nele. 2. A aplicação retorna uma lista contendo as categorias extraídas.

Fonte: elaborado pelo autor.

APÊNDICE B – Modelo de Entidade Relacionamento

O banco de dados do *web service* é constituído por 5 tabelas, como pode ser visto no MER ilustrado pela Figura 17.

Figura 17 – Modelo de entidade relacionamento



Fonte: elaborado pelo autor.

Todas as ofertas extraídas das lojas são armazenadas na tabela `ofertas` em que a identificação de qual loja elas pertencem é feita através do seu relacionamento com a tabela `lojas`.

As categorias definidas como padrões e que irão aparecer como opções de escolha para o usuário no smartphone são armazenadas na tabela `categorias_app`. Estas categorias são vinculadas às ofertas através do seu relacionamento com a tabela `ofertas`. Todas as categorias extraídas das lojas são armazenadas na tabela `categorias_loja`, que possui um relacionamento `N:N` com a tabela `categorias_app`, que é representado pela tabela de ligação `categorias_app_loja`.

APÊNDICE C – Questionário de avaliação de usabilidade e funcionalidade da aplicação

Neste apêndice consta o questionário utilizado para a avaliação da usabilidade e funcionalidade da aplicação.

Quadro 21 – Questionário de perfil do usuário

Dita Ofertas – uma aplicação para reproduzir ofertas de produtos no rádio do carro

O Dita Ofertas é um aplicativo com o propósito de facilitar a consulta de ofertas de lojas online enquanto o usuário está no trânsito. Com o aplicativo o usuário pode criar perfis onde ele pode selecionar categorias de lojas virtuais que desejar e o aplicativo reproduzirá as ofertas buscadas dessas lojas no rádio do carro através de conexão Bluetooth.

Gostaríamos da sua ajuda para avaliar o aplicativo desenvolvido a fim de mapear possíveis melhorias e deixar a experiência do usuário melhor. Para isto, gostaríamos que você respondesse o formulário de perfil de usuário e em seguida o formulário de avaliação do aplicativo.

PERFIL DE USUÁRIO

Sexo: ( ) Feminino ( ) Masculino

Idade:

( ) Entre 18 e 30 anos ( ) Entre 30 e 45 anos ( ) Acima de 45 anos

O que você costuma fazer quando fica parado no congestionamento?

( ) Ouve música ( ) Ouve notícias no rádio ( ) Acessa as redes sociais
( ) Visita sites de compras
Outros:_____

Você costuma ouvir rádio enquanto dirige ou está de carona?

( ) Sim ( ) Não

Você já recebeu alguma multa por utilizar o celular enquanto dirige?

( ) Sim ( ) Não

Você tem o costume de fazer compras online?

( ) Sim ( ) Não

Você costuma fazer pesquisas de preço online antes de efetuar a compra?

( ) Sim ( ) Não

Quais categorias de produto você mais compra?

( ) Não compro
Categorias:

Quadro 22 – Questionário de atividades do usuário

LISTA DE ATIVIDADES DO USUÁRIO

A seguir são apresentadas algumas atividades com o objetivo de avaliar a usabilidade do aplicativo. Por gentileza, realize o que é solicitado em cada atividade, indicando se a mesma foi concluída ou não.

1) Instalação do aplicativo
   Baixe o aplicativo e faça sua instalação.
   Você conseguiu fazer a instalação? Sim, não? Por quê?

   _____

   Criar um perfil

   É necessário estar conectado à internet para realizar o cadastro do perfil.
   Você conseguiu criar um perfil? Sim, não? Por quê?

   _____

   _____

2) Buscar as ofertas
   Após efetuar o cadastro do perfil, selecione um perfil e clique na opção “Ditar ofertas” para ouvir as ofertas relacionadas ao perfil.
   Você conseguiu ouvir as ofertas buscadas? Sim, não? Por quê?

   _____

   _____

3) Acessar a página de detalhe do produto
   Quando as ofertas estiverem sendo reproduzidas clique na URL do produto.
   Você conseguiu acessar a página com mais detalhes do produto no browser? Sim, não? Por quê?

   _____

   _____

4) Consultar buscas efetuadas
   Após realizar alguma busca, clique no botão “Buscas Anteriores”, selecione algum dos perfis que foram buscados e clique em “Ouvir Novamente”.
   Você conseguiu ouvir as ofertas buscadas novamente? Sim, não? Por quê?

   _____

   _____

5) Exclusão dos registros
   Exclua alguns dos perfis criados.
   Você conseguiu excluir os perfis? Sim, não? Por quê?

   _____

   _____

Quadro 23 – Questionário de usabilidade

QUESTIONÁRIO DE AVALIAÇÃO DO APLICATIVO

Após a utilização do aplicativo, você está convidado a responder um questionário para avalia-lo. As respostas deverão ser feitas na tabela abaixo observando às impressões obtidas com a utilização do aplicativo. Você deve responder preenchendo uma das alternativas. Após o questionário com perguntas objetivas, é apresentado um espaço para comentários gerais sobre a ferramenta e sugestões de melhorias.

Perguntas / Critérios de avaliação	**Concordo totalmente**	**Concordo parcialmente**	**Não concordo nem discordo**	**Discordo parcialmente**	**Discordo totalmente**
O aplicativo é simples e objetivo					
A qualidade do áudio das ofertas é boa					
É fácil navegar entre as telas do aplicativo					
O aplicativo em algum momento parou inesperadamente					
A aparência do aplicativo é boa (cores e ícones)					

Você conseguiu compreender o objetivo do aplicativo?
( ) Sim ( ) Não

Você considera o aplicativo útil?
( ) Sim ( ) Não

Você achou importante salvar as ofertas que já foram vistas anteriormente?
( ) Sim ( ) Não

Você conseguiu acessar a página de detalhes do produto clicando na URL disponível na tela das informações da oferta?
( ) Sim ( ) Não

Você conseguiu interromper a reprodução das ofertas?
( ) Sim ( ) Não

Você utilizaria o aplicativo para pesquisa de preços?
( ) Sim ( ) Não

Você recomendaria o aplicativo para outras pessoas?
( ) Sim ( ) Não

Você acha interessante a funcionalidade de reprodução das ofertas no som do carro?
( ) Sim ( ) Não

Qual é a sua opinião sobre o aplicativo quanto ao seu uso e funcionalidades? Fique à vontade para fazer críticas e sugestões.

_____
_____
_____