CENTRO UNIVERSITÁRIO LUTERANO DE PALMAS

Recredenciado pela Portaria Ministerial nº 1.162, de 13/10/16, D.O.U. nº 198, de 14/10/2016
AELBRA EDUCAÇÃO SUPERIOR - GRADUAÇÃO E PÓS-GRADUAÇÃO S.A.

DESENVOLVIMENTO DE APLICATIVO MOBILE PARA CONSULTA DE PREÇOS DE COMBUSTÍVEIS NO ESTADO DO TOCANTINS

Palmas – TO

2019

Yuri Cordeiro Teixeira

DESENVOLVIMENTO DE APLICATIVO MOBILE PARA CONSULTA DE PREÇOS DE COMBUSTÍVEIS NO ESTADO DO TOCANTINS

Projeto de Pesquisa elaborado e apresentado como requisito parcial para aprovação na disciplina de Trabalho de Conclusão de Curso I (TCC I) do curso de bacharel em Ciência da Computação pelo Centro Universitário Luterano de Palmas (CEULP/ULBRA).

Orientadora: Profa. MSc. Heloise Acco Tives Leão.

Palmas – TO

2019

Yuri Cordeiro Teixeira

DESENVOLVIMENTO DE APLICATIVO MOBILE PARA CONSULTA DE PREÇOS DE COMBUSTÍVEIS NO ESTADO DO TOCANTINS

Projeto de Pesquisa elaborado e apresentado como requisito parcial para aprovação na disciplina de Trabalho de Conclusão de Curso I (TCC I) do curso de bacharel em Ciência da Computação pelo Centro Universitário Luterano de Palmas (CEULP/ULBRA).

Orientador: Profa. MSc. Heloise Acco Tives Leão.

Aprovado em: _____/_____/_____

BANCA EXAMINADORA

_____

Profª. MSc. Heloise Acco Tives

Orientadora

Centro Universitário Luterano de Palmas – CEULP

_____

Profª. Dra. Parcilene Fernandes de Brito

Centro Universitário Luterano de Palmas - CEULP

_____

Profª. MSc. Cristina D'Ornellas Filipakis

Centro Universitário Luterano de Palmas - CEULP

Palmas – TO

2019

RESUMO

Diante das grandes variações de preços de combustíveis nos postos revendedores, observa-se a importância de um aplicativo para consulta de preços em tempo real. Plataformas *mobile* podem ser vantajosas e utilizadas quando o aplicativo a ser desenvolvido necessita de mobilidade e que se encaixa com as características do problema desse trabalho. Neste, apresenta-se o desenvolvimento de um aplicativo *Mobile* com o objetivo de realizar consultas de preços de combustíveis dos postos revendedores do estado do Tocantins, sendo que o APP dá a possibilidade de acompanhar os preços em tempo real. A solução desenvolvida se enquadra como uma proposta de solução para o problema existente, que é disponibilizar um aplicativo que possibilite a consulta de preços de combustíveis. O projeto utiliza uma base de dados fictícia para auxiliar no desenvolvimento do APP, levando-se em consideração a Lei 19.888/17 do estado de Goiás que obriga a comunicação de alterações de preços dos combustíveis ao MP ainda não foi sancionada no Tocantins. O APP conta com as funcionalidades de consulta de postos através do *Google Maps* ou por cidade, listar preços dos combustíveis revendidos pelo posto selecionado, obtenção dos preços em tempo real, indicação dos postos com menor preço com base na localização e custo-benefício do veículo por quilômetro rodado e dentre outras.

Palavras-chave: Consulta de Preços de Combustíveis, Desenvolvimento Multiplataforma, Lei 19.888/17.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API – *Application Programming Interface*

APP – *Application*

BD – Banco de Dados

CEULP/ULBRA – Centro Universitário Luterano de Palmas

CSS – *Cascading Style Sheets*

FIPE - Fundação Instituto de Pesquisas Econômicas

GLP – Gás Liquefeito de Petróleo

GPS – *Global Positioning System*

HTML – *HyperText Markup Language*

IDE – *Integrated Development Environment*

JWT – *JSON Web Tokens*

MAC - *Macintosh Operating System*

MP/GO – Ministério Público do Estado do Goiás

MP/TO – Ministério Público do Estado do Tocantins

MVC – *Model-View-Controller*

MWA – *Mobile Web Application*

OHA – *Open Handset Alliance*

SDK – *Software Development Kits*

SO – Sistema Operacional

TCVM – *TotalCross Virtual Machine*

UFG – Universidade Federal de Goiás

XAML – *Extensible Application Markup Language*

SUMÁRIO

1 INTRODUÇÃO

A busca por produtos e serviços com preços acessíveis é natural para as pessoas, sendo que se percebe na rotina de muitos consumidores a execução de pesquisas em sites, aplicativos e comércios físicos antes da realização do negócio/compra.

O combustível é um dos produtos mais utilizados pelos brasileiros já que, conforme apontado por Trânsito (2019), existem 101.346.180 de veículos no Brasil dentre eles automóveis, bondes e caminhões. Esses veículos são utilizados pelos brasileiros como meios de transportes, seja para o trabalho ou o lazer. De acordo com Biocombustíveis (2019), no ano de 2017 foram consumidos 136.025 bilhões de litros de combustível no país.

A partir de pesquisas realizadas em lojas de aplicativos *Mobile*, como *Google Play* e *Apple Store*, para tentar identificar aplicativos que disponibilizassem a consulta em tempo real de preços dos combustíveis, um único aplicativo com esse propósito específico foi encontrado. Trata-se do APP "Olho na Bomba", que objetiva disponibilizar os preços praticados pelos postos revendedores de combustíveis do estado de Goiás. Um fato a se ressaltar sobre este Estado é que ele possui uma lei em vigor que normatiza e obriga os revendedores de combustíveis a informar de imediato ao MP/GO qualquer alteração referente ao valor praticado em seus combustíveis.

Na expectativa de que outros estados brasileiros sancionem uma lei similar que permita a criação de uma base de dados única e atualizada com os custos dos combustíveis para serem disponibilizados para os consumidores, este trabalho apresenta o desenvolvimento de uma aplicação para acompanhamento dos preços dos combustíveis em postos revendedores.

O intuito do aplicativo desenvolvido será de auxiliar o consumidor a comprar o combustível de sua preferência pelo menor preço possível, tendo funcionalidades como a possibilidade de denúncia contra os postos que revendem os combustíveis com preços divergentes com o da plataforma. O aplicativo utiliza ferramentas que possibilita a compilação para *smartphones iOS* e *Android*, dando praticidade às pessoas que utilizam o aplicativo de onde estiverem e que possui qualquer um dos SO.

Vale ressaltar que foi utilizada uma base de dados fictícia simulando os dados de descrição dos postos revendedores de combustível tocantinenses, contendo dados como

localização, tipos de combustíveis, preços, entre outros. Essa base foi criada no desenvolvimento desse projeto como forma de auxílio.

Esse trabalho se subdivide da seguinte forma: a seção 2 traz a revisão de literatura com os conceitos e as definições das tecnologias utilizadas nesse trabalho. A seção 3 traz os materiais e métodos utilizados no desenvolvimento deste trabalho. Na seção 4 são abordados os resultados e a discussão. Por fim, a conclusão na seção 5, que abordada de forma seguida as referências bibliográficas.

2 REFERENCIAL TEÓRICO

Nesta seção são apresentados os conceitos relevantes para a execução deste trabalho, sendo abordados itens como o desenvolvimento *Mobile*, as diferenças entre o uso de tecnologias nativas, desenvolvimento *Mobile Web Application* e tecnologias de desenvolvimento multiplataformas.

2.1 Desenvolvimento Mobile

Conforme Prezotto (2014), dispositivo móvel é todo aquele equipamento, que pode ser levado a qualquer lugar. Quanto menor dependente de equipamentos fixos de um lugar, maior será o grau de mobilidade. A mobilidade para o dispositivo móvel é considerada quando possui bateria de alta duração e curto tempo de recarga, tendo o aparelho por um longo tempo de uso.

De acordo com Da Silva (2014), a evolução da tecnologia dos aparelhos celulares oferece ao usuário recursos que vão muito além da realização de chamadas ou do envio de mensagens, sendo utilizados para operações como desbloqueio da tela via impressão digital à leitura de íris através da câmera frontal do aparelho, dentre várias outras.

Uma das funcionalidades necessárias para aplicativos *Mobile* é a facilidade de instalação, já que tais aplicativos são todos distribuídos por lojas oficiais de cada plataforma e o usuário é responsável pela instalação em seu aparelho. Para se conectar a essas fontes de distribuição, normalmente é cobrada uma taxa do desenvolvedor, por exemplo na *Google* essa taxa é única e custa $25 dólares e na *Apple* $99 dólares anual (MCILROY, 2016).

O desenvolvimento de softwares para dispositivos *Mobile* possui algumas limitações como por exemplo a internet que nem sempre tem a qualidade/velocidade necessária para o bom funcionamento de alguns aplicativos. Desta maneira, uma característica importante para os aplicativos é que estes consumam o mínimo dos dados móveis dos *smartphones*.

Outro ponto crítico no desenvolvimento de aplicativos móveis é a limitação de recursos físicos de alguns aparelhos, como por exemplo memória, armazenamento e processador. Esse fator tem que ser levado em consideração pelos desenvolvedores e projetistas, pois pode ser um limitador para a distribuição e uso do aplicativo.

Na próxima seção são apresentados os paradigmas de desenvolvimento de aplicativos para dispositivos celulares e a influência da escolha do paradigma para o resultado do desenvolvimento de um produto.

2.1.1 Desenvolvimento de Aplicativo Nativo para Dispositivos Celulares

O desenvolvimento de aplicativos para celulares requer atenção em relação à qual plataforma se deseja disponibilizar esse aplicativo, isso porque essa escolha influencia na determinação da IDE de desenvolvimento. As plataformas mais conhecidas são *Android* (*Google*), iOS (*Apple* Inc) e *Windows Mobile* (*Microsoft* Corp), Da Silva (2014). Uma pesquisa realizada no ano de 2015 no Brasil mostra que o sistema *Android* domina o mercado com 90,4% dos usuários e o iOS possui 4,7% dos usuários ativos Carvalho (2017).

De acordo com Da Silva (2014), as plataformas para *smartphones* são compostas de diversas tecnologias, tais como: sistema operacional, linguagens de programação e IDEs. O sistema operacional fica com a responsabilidade de comandar vários recursos do *smartphone*, as linguagens de programação são empregadas no desenvolvimento dos aplicativos e a IDE com a responsabilidade de disponibilizar ferramentas que auxiliem os desenvolvedores na criação dos APPs.

2.1.1.1 Android

A linguagem nativa e oficial a ser utilizada no desenvolvimento de aplicativo *Android* é o *Java*. De acordo com Gomes (2012), o *Java* atualmente é mantido pela OHA, um grupo constituído por aproximadamente 80 empresas que se unificaram para inovar e tornar o desenvolvimento de aplicativos e serviços mais ágil, com o objetivo de levar aos consumidores uma experiência mais rica em termos de recursos e financeiramente menos custoso para o mercado de telefonia móvel Da Silva (2012).

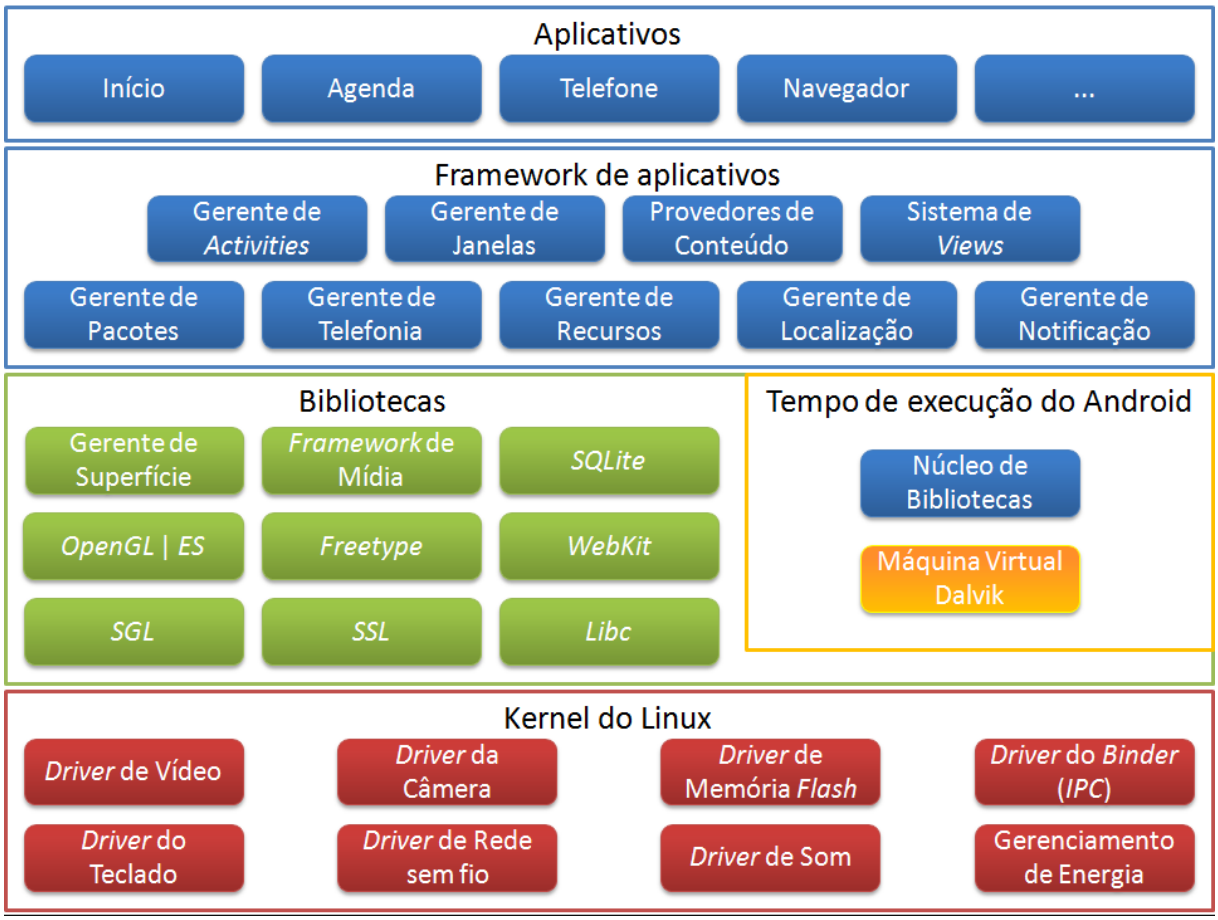
De acordo com Da Silva (2012), a *Google* realiza melhorias constantes no sistema operacional *Android*, que é baseado em *Linux*, com o objetivo de ser uma plataforma flexível, aberta e de fácil migração para as fabricantes de dispositivos celulares. O sistema operacional *Android* é incorporado em diversos dispositivos, não somente em celulares, mas em *tablets*, *netbooks*, TVs, relógios, entre outros.

O *Android* possui funcionalidades como gráficos 3D e suporte à Banco de Dados nativo, sendo que um diferencial entre os outros sistemas é o *Google Maps*, serviço de mapas do *Google* que é utilizado por diversos aplicativos. O *Google Maps* fornece diversos métodos que disponibilizam desde uma rota entre origem e vários destinos, como tempo que levará a viagem, quantos quilômetros, melhores percursos e dentre outros Gomes (2012).

O *Android* possui uma arquitetura composta por basicamente 4 camadas, ilustrada na Figura 1, e brevemente descrita a seguir:

- Aplicativos: a camada do topo da pirâmide que é composta por um conjunto de aplicações nativas do *Android* como por exemplo cliente *e-mail*, despertador, calendário, jogos, mapas, *browser* e internet, entre outros;
- *Framework* de Aplicativos: a camada de *framework* de aplicativos disponibiliza APIs utilizadas na criação de aplicativos do SO do *Android*, sendo reaproveitado pela camada de aplicações e abstraindo a complexidade e simplificando a reutilização dos procedimentos;
- Bibliotecas: nessa camada são é possível a manipulação do áudio, vídeo, gráficos, Banco de Dados e *browser* do SO. Exemplos de bibliotecas são a *Bionic*, OpenGL/ES para lidar com a interface gráfica e a SQLite para trabalhar com Banco de Dados;
- Tempo de Execução do Android: aqui cada *thread* é executado em sua própria instância da MV (Máquina Virtual). O desenvolvimento *Java* desktop e web, as aplicações são executadas em uma máquina virtual tradicional do *Java*, mas as aplicações para *Android*, utilizam uma MV diferente, chamada de *Dalvik*;
- *Kernel* do *Linux*: o *kernel* é baseado em um sistema do SO do *Linux* versão 2.6. É responsável pela abstração entre o hardware e os aplicativos e é encarregado pelos serviços principais do SO do *Android*, como o gerenciamento de memória e de processos.

Figura 1 - Estrutura do Sistema Android.



Fonte: http://docplayer.com.br/docs-images/18/753208/images/21-0.png

A arquitetura interna do *Android* possui um nível de complexidade elevada, mas as camadas superiores dessa estrutura auxiliam os desenvolvedores a deixar esse nível de complexidade menos complicado, tornando o desenvolvimento de aplicações nativas do *Android* algo comum atualmente.

2.1.1.2 iOS

De acordo com Silva (2011), o sistema operacional (SO) da *Apple* o iOS é um dos mais populares e robustos, detém de sistema de interface com os usuários, simples e intuitivo, com capacidade de reconhecimento de gestos e várias outras funcionalidades interessantes. Os mais recentes *iPhones*, como a versão X, possui um sistema de reconhecimento facial muito poderoso, capaz de desbloquear a tela do aparelho apenas com a face do proprietário do celular.

Sendo considerado um diferencial dos dispositivos produzidos pela *Apple*, o SO não é licenciado para uso em *hardware* de terceiros. No início do lançamento do iOS a *Apple* não permitia que aplicativos de terceiros rodassem nos seus dispositivos, somente em março de

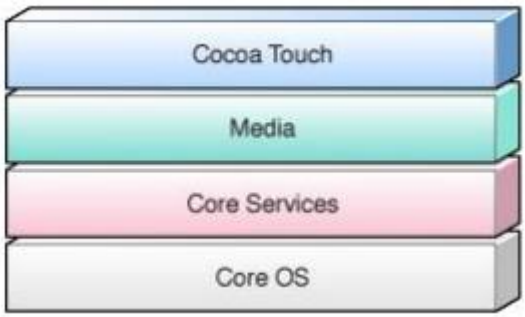
2008 quando foi liberado o *iPhone* SDK, que foi liberado para a comunidade o desenvolvimento de aplicativos para iOS. A *Apple* possui diversos aparelhos que rodam o só iOS, como *iPhone*, *iPad* e *iPod touch*, sendo o *iPhone* um dos mais utilizados atualmente.

A IDE oficial da *Apple* é o *XCode tools*, com essa ferramenta é possível editar e compilar os aplicativos gerados, possuindo um simulador de dispositivos como o *iPhone*, dando aos desenvolvedores a oportunidade de testar os aplicativos, sem um dispositivo físico. Um aspecto negativo em relação ao *Android* é que por ser um simulador, a *Google* fornece um emulador que é mais preciso no momento de emular a aplicação, deixando o comportamento dos aplicativos mais próximos dos dispositivos reais.

A arquitetura do sistema operacional iOS age como um *middleware*, intermediando entre a aplicação e o *hardware* do dispositivo. O acesso ao *hardware* do dispositivo se dá através de um conjunto de interfaces bem definidas. A *Apple* tem uma grande preocupação quando se trata da segurança do usuário, percebe-se pela adoção de um *sandbox* para os aplicativos executarem Silva (2011).

Cada aplicação somente pode ler e escrever em uma área específica, estabelecida pelo iOS, essa área é chamada de *sandbox*. Ainda sobre a estrutura do iOS, essa possui quatro camadas de abstração conforme apresentado na Figura 2 Silva (2011).

Figura 2 - Camadas de abstração do sistema operacional iOS.



Fonte: SILVA (2011).

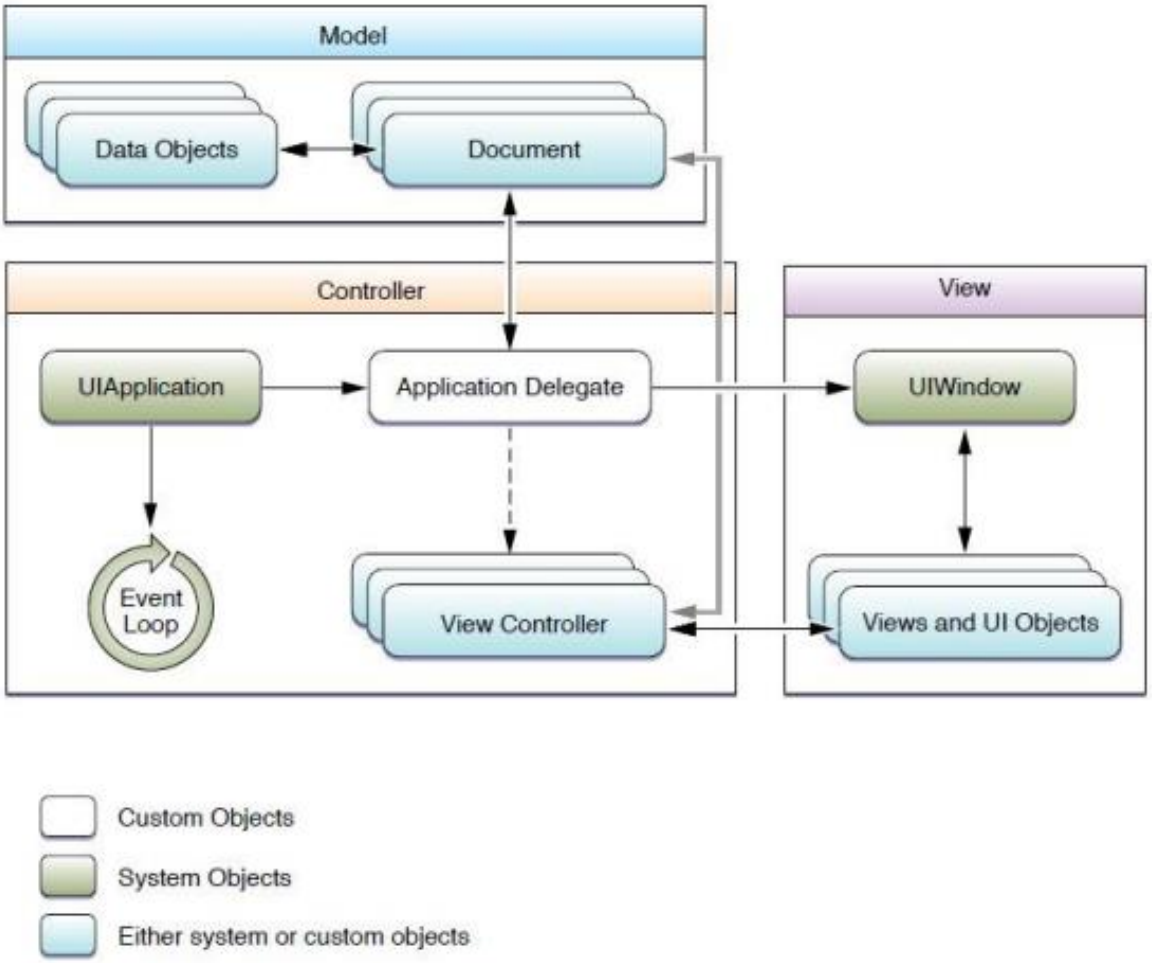
Como qualquer outro sistema operacional, o iOS possui algumas camadas baixo nível em termos de codificação, essas são as *Core Services* e *Core* OS, que são desenvolvidas em C, tendo uma certa complexidade para abstração. Já as camadas superiores utilizam tecnologias avançadas, geralmente são escritas em uma mistura de *Objective*-c e c, a explicação de cada camada é dada a seguir.

- *Cocoa Touch*: é a camada mais utilizada no desenvolvimento de aplicativos para iOS, contendo os principais *frameworks* necessários para a criação dos aplicativos.

- Media (Mídia): é responsável por gerir recursos tais como áudio, vídeo e computação gráfica. Essa camada possui diversas tecnologias para desenhar na tela do dispositivo, sendo as principais tecnologias;
    - *Core Graphics*: que suporta vetores 2D e renderização baseadas em imagens;
    - *Core Animation*: Que provê suporte avançado a animação das *views*;
    - *Core Image*: Detém de métodos para manipulação de fotos e vídeos;
    - OpenGL ES e GLKIT: Contendo suporte a renderização em 2D e 3D contando com a aceleração de hardware;
    - *Core Text*: Dando suporte a layout e renderização de textos;
    - *Image* I/O: suporte a leitura e escrita em diversos formatos de imagens.
- *Core Services*: contém diversos serviços de sistema, que não é utilizada de forma direta pelos desenvolvedores, as aplicações fazem uso destas pois diversas partes do sistema são construídas com base nestes serviços. Os principais serviços desta camada são: armazenamento em nuvem (iCloud), suporte a XML, SQLite, In-APP *Purchase* (Sistema de compras utilizado pelas aplicações), ARC (Serviço que simplifica a gerência de memória dos aplicativos) e Bloco de objetos (citada em outras linguagens como *closure* ou *lambda*);
- *Core* OS: possui abstração baixo nível, com serviços disponibilizados que são utilizados para criar as demais camadas. Essa camada só é utilizada diretamente quando se é necessário realizar a comunicação com acessórios externos ou com questões referentes à segurança.

Os desenvolvedores de aplicativos para iOS seguem o padrão MVC, que tem como principal objetivo separar a lógica de negócio, da lógica de apresentação. Na Figura 3, é possível verificar a utilização dessa arquitetura com os principais elementos do iOS.

Figura 3 - Ligação da arquitetura MVC com os principais elementos do iOS.



Fonte: SILVA (2011).

O sistema operacional iOS mostra-se eficiente e seguro, possuindo uma interface agradável e de fácil utilização, isso pode ser visto com a imensa comunidade que utiliza produtos da *Apple*. Contudo, com a grande preocupação com a segurança que a *Apple* possui, pode dificultar o trabalho dos desenvolvedores, tornando tarefas que seriam cruciais em outras plataformas, bastante difícil de serem implementadas no iOS Silva (2011).

2.1.2 Desenvolvimento Mobile Web Application

De acordo com Tavares (2016), *Web* APPs na realidade são sites que de diversas formas possuem características responsivas para *smartphones,* mas não são considerados aplicativos nativos. Esses aplicativos executam por intermédio de um navegador e são tipicamente escritos em *HTML5*. Suas execuções se iniciam como uma aplicação *web*, quando os usuários abrem um navegador no dispositivo, inserem uma *url* e tem-se um *website* responsivo.

Esse tipo de aplicação consome menos memória que um aplicativo propriamente da plataforma, pois seus serviços não estão executando diretamente no dispositivo e sim hospedado em um serviço *web*. Assim diminuindo o consumo de recursos dos dispositivos celulares, porém, as funcionalidades deste tipo de aplicação são limitadas em comparação a APPs nativos.

Aplicações *Webs* e *mobile* têm suas diferenças possuindo seus pontos positivos e negativos, a escolha da plataforma é um dos requisitos no momento do desenvolvimento de uma aplicação sendo bastante possível que a escolha acabam sendo as duas coisas. Conforme Lopes (2016), *web* APPs favorece a descoberta de conteúdo despretensiosamente: não exige instalação e não exige compromisso dos usuários. Os APPs ao mesmo tempo geram fidelização com o público e uma experiência mais incorporada à plataforma por estar disponível em diferentes dispositivos sejam em *smartphones*, computadores ou *tablets*.

Embora os aplicativos da *web* possam replicar uma grande quantidade de comportamentos de aplicativos nativos, existem algumas áreas em que tecnologias baseadas na *web* não podem competir com funcionalidades nativas. Como utilizar recursos avançados do dispositivo, por exemplo, em manipular a câmera ou o giroscópio do *iPhone*, por isso o desenvolvimento híbrido ou nativo tende a ser mais utilizado nesse contexto Sin (2012).

2.1.3 Desenvolvimento de Aplicativo Multiplataforma Para Dispositivos Celulares

As empresas ao desenvolverem aplicativos móveis devem identificar a plataforma na qual será disponibilizada a aplicação, para com isso diminuir o menor esforço e tempo necessário durante o desenvolvimento da aplicação, bem como manter a geração do APP mais transparente possível para os desenvolvedores Schmitz (2016).

Segundo Schmitz *apud* Petzold (2015), o desenvolvimento de aplicativos móveis para as principais plataformas do mercado enfrenta pelo menos gera quatro problemas, sendo:

1. Diferença de interfaces dos usuários, seja pela variação de botões físicos e capacitivos ou posicionamento de menus e opções para usuários.
2. Diferentes IDEs utilizadas no desenvolvimento.
3. Diferentes interfaces de programação, onde cada sistema possui uma determinada API, gerando nomes distintos para um mesmo tipo de objeto da interface do usuário.
4. Cada sistema possui sua própria linguagem de programação.

Com a alta demanda de aplicações *Mobile*, surgiram diversas ferramentas que permitem aos desenvolvedores criar uma aplicação e portá-la rapidamente para os diferentes sistemas operacionais *mobile*. Dentre as ferramentas em destaque se encontram o *Xamarin*, *TotalCross, Cordova, PhoneGap* e *Ionic* Schmitz (2016).

Essas tecnologias possuem algumas diferenças em suas execuções de aplicativos, pois enquanto as aplicações geradas com o *Xamarin*, *Cordova*, *PhoneGap* e *Ionic* executam diretamente nos dispositivos, os aplicativos gerados com a *TotalCross* executam na máquina virtual TCVM, que através dela é realizado a comunicação direta com o SO do dispositivo. Outras características importantes dessas tecnologias são apresentadas na Tabela 1.

Tabela 1 - Diferença entre as ferramentas de desenvolvimento mobile multiplataforma.

Características	**Xamarin**	**TotalCross**	**Cordova, PhoneGap, Ionic**
Criação da interface	Linguagem própria chamada de *Xamarin.Forms*.	Linguagem Java.	Linguagens HTML5 e CSS.
Recursos nativos	APIs nativas das plataformas.	TCVM.	*Plugins* Cordova.
Execução	Quase nativa.	Emulada na TCVM.	Nativa, encapsulada em *web* APP.
Método de acesso às APIs nativas	Mapeamentos internos, gerenciados pela própria ferramenta.	SDK do Java SE.	*Cordova* e *PhoneGap* realiza comunicação com as APIs. *Ionic* é usado para *Layout.*
Vantagem	Proporciona um ambiente transparente para o desenvolvedor;	Desenvolvimento para iOS sem necessidade MAC OS.	Utilização de linguagem *web* para desenvolvimento de APPs.

Fonte: SCHMITZ (2016).

Por ser uma solução viável e de grande impacto para o desenvolvimento *Mobile*, os *frameworks* multiplataforma vieram para auxiliar o desenvolvimento, tornando essencial, sendo que o mercado respondeu rapidamente com múltiplas soluções Freire (2013). Entre as vantagens de utilizar soluções multiplataforma no desenvolvimento *mobile* têm-se Bernardes (2016):

- Redução da necessidade de conhecimento específico;

- Aprendizado de linguagens como *HTML* e *JavaScript* no desenvolvimento *Mobile* é menos desafiador do que obter conhecimentos em *Objective*-C e Java no desenvolvimento nativo;
- Redução no número de linhas de código, sendo que o código é escrito apenas uma vez para as diferentes plataformas;
- Delimitação no tempo de desenvolvimento e custo de manutenção;
- Redução da necessidade de conhecimento de APIs, pois com o uso de ferramentas multiplataforma não há necessidade de conhecimento das APIs de cada plataforma, e sim de apenas a da ferramenta em questão.

Uma decisão que deve ser tomada pelos desenvolvedores diz respeito ao tipo de desenvolvimento, podendo ser nativo ou híbrido que de acordo com Lopes (2016), essa decisão é puramente técnica já que do ponto de vista do usuário não há diferença. Onde o aplicativo requer alto processamento, nesse caso aplicações nativas são recomendadas, caso contrário híbridas terão um diferencial em tempo de desenvolvimento e dentre outras.

Os aplicativos híbridos são parcialmente nativos e parcialmente MWA, igualmente como os nativos, eles devem ser instalados através de um aplicativo de loja, ficam armazenados na tela principal do dispositivo e utilizam todos os recursos e funcionalidades do dispositivo (câmera, GPS, acelerômetro, gestos etc.), Tavares (2016).

2.2 TRABALHOS RELACIONADOS

Nesta seção são abordados os trabalhos relacionados com o desenvolvimento *Mobile* multiplataforma, levando em consideração as tecnologias utilizadas, metodologias de desenvolvimento, pontos positivos e pontos negativos da solução final.

A parceria entre MP/GO e UFG para sanar o problema da transparência dos preços de combustíveis para os consumidores, resultou no desenvolvimento de aplicação *Mobile*, que deu a possibilidade da realização de consultas de preços de combustíveis em tempo real pelos consumidores. A solução desenvolvida está presente apenas para os consumidores do estado do Goiás que com a ajuda da Lei 19.888/17 que obriga os postos de gasolina a se registrarem na plataforma e notificar as alterações dos preços dos combustíveis, permitiu o sucesso do APP Goiás (2017).

O aplicativo “Olho na Bomba”, entre outubro e dezembro de 2018 a ferramenta obteve 230 mil downloads e tendo a média de 20 mil usuários ativos por dia e 506 denúncias.

Com isso percebe-se como uma ferramenta similar pode contribuir para apoiar o consumidor a obter informações, com a credibilidade necessária para embasar sua decisão de compra em relação aos combustíveis.

O *Xamarin Forms* é um *framework* utilizado no desenvolvimento de aplicações multiplataforma, que diversas empresas deixam de utilizá-la por falta de conhecimento ou até mesmo de suas vantagens Radi (2016).

O trabalho de Radi (2016) apresenta o desenvolvimento de uma aplicação móvel simples com o uso do *framework Xamarin Forms.* Neste trabalho são demonstradas as características comuns vistas em muitas aplicações para dispositivos móveis. Além disso o projeto explorou recursos de *hardware* através do *Xamarin*, como acessar o GPS, com o objetivo de realizar uma avaliação imparcial sobre o *framework*.

Ainda de acordo com o trabalho de Radi (2016), é identificado que o *Xamarin* é uma excelente ferramenta para desenvolvimento *mobile*, já que possibilita o fornecimento de apenas um código e linguagem única para as diferentes plataformas destino. Como problemas identificados, está a implementação de recursos específicos de *hardware* como GPS para plataformas destino, como Android e iOS. Por fim, a pesquisa de Radi (2016), aponto que embora a comunidade do *Xamarin* esteja se expandindo, não se compara com as comunidades de tecnologias nativas, e ainda é perceptível a falta de algumas bibliotecas e recursos no *Xamarin* que são importantes para sua disseminação, embora já ofereça vantagens como desenvolvimento rápido, eficiente e fácil.

Na pesquisa de Da Silva (2014), são apresentadas informações sobre os tipos de paradigmas de desenvolvimento de aplicativos para aparelhos celulares, apontando a importância da escolha do paradigma como fator para a garantia da qualidade dos apps a serem desenvolvidos.

Esta mesma pesquisa identificou que as ferramentas de desenvolvimento permitem a redução de custo, tempo e complexidade, mas deve ser levado em consideração também a existência de limitações por parte da ferramenta multiplataforma selecionada para o desenvolvimento. O cenário dos aplicativos para *smartphones* pode ser claramente associado a experiência recente com os *desktops*, onde softwares nativos foram superados pelos softwares *Web* Da Silva (2014).

O trabalho de Kronbauer (2017), visou demonstrar a avaliação da usabilidade de aplicativos móveis com foco principal para aplicativos que são executados em *smartphones*,

tendo como escopo de interesse abordagens e técnicas para as avaliações da usabilidade de aplicativos em plataformas móveis. Algumas informações são levadas em consideração como as tarefas a serem executadas, dados do contexto, características dos usuários, recursos dos dispositivos, novos paradigmas de interação e a satisfação dos usuários ao interagirem com os aplicativos.

Ainda de acordo com Kronbauer (2017), a coleta de dados de usabilidade da aplicação após o desenvolvimento é importante, principalmente referente a interação do usuário final com a aplicação, para apoiar na identificação perfil dos usuários. Estes diversos tipos de dados, possibilitam realizar análises alusivas à usabilidade correlacionando diversos aspectos e possibilitando a detecção de problemas com as interfaces.

3 METODOLOGIA

Nesta seção serão apresentadas a lista de materiais e as etapas que compõem a metodologia utilizada para desenvolvimento do aplicativo proposto.

3.1 MATERIAIS

Para execução deste aplicativo foram levados em consideração aspectos sobre tempo de desenvolvimento, tecnologias multiplataforma e custo de manutenção do software. As ferramentas selecionadas para atender essas necessidades são apresentadas a seguir.

3.1.1 Visual Studio

O *Microsoft Visual Studio* é considerado uma ferramenta adequada para o desenvolvimento de softwares, proporcionando uma IDE para linguagens como o C#, J#, *Visual Basic* (VB), C, C++, *Xamarin*, dentre outras.

O Visual *Studio* será utilizado neste trabalho como IDE principal de desenvolvimento, integrando as necessidades da linguagem de programação C# e o *framework Xamarin Forms* para criação do APP.

3.1.2 Xamarin Forms

Framework multiplataforma como o *Xamarin Forms* possibilita aos desenvolvedores criar aplicativos para várias plataformas usando o mesmo código base. Neste trabalho o uso do *Xamarin Forms* será dado em função da necessidade de desenvolvimento do APP para as plataformas *Android* e iOS.

3.1.3 PHP

Linguagem de programação que está no mercado desde o ano 1995, sendo uma tecnologia bem amadurecida dando credibilidade ao ser escolhida como linguagem *Back-End*, seja para o desenvolvimento de sites, sistemas *Web*, APIs, entre outros. A aplicação do PHP para o tratamento de dados na API objetiva a integração com o *Composer*, *Silex* e *Mysql* como proporciona uma sintaxe de simples entendimento e transparência na manutenção do código.

3.1.3 Silex

Trata-se de um micro-*framework* que provê um sistema de rotas e condições de resolvê-las através dos *Services* e *Providers*, Kstro (2015). As rotas da API deste trabalho foram implementadas utilizando o *silex*, devido sua integração com o *jwt* e o PHP.

3.1.4 Composer

O Composer é uma ferramenta que faz o gerenciamento de dependências do PHP, realizando o gerenciamento dos pacotes instalados no projeto, como por exemplo o sílex Zemel (2012).

O projeto da API do *Back-End* que será abordado na seção de desenvolvimento utiliza esse *framework* para gerenciar as dependências do projeto, de modo que ao ser implantado o analista não necessite realizar a instalação dos pacotes individualmente.

3.1.5 Mysql

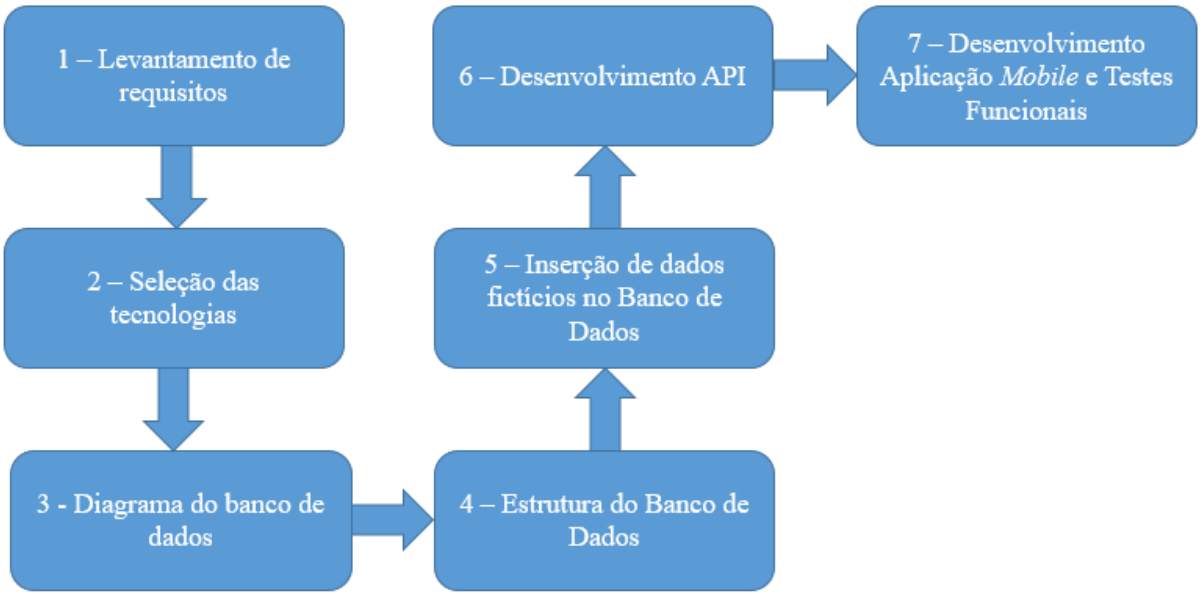
O *Mysql*, além de ser *open-source*, na sua versão *Community Edition*, possui as vantagens de ter compatibilidade com diversos sistemas operacionais (*Linux*, *Windows* e *Mac*), desempenho elevado, uso gratuito, possibilidade de ser multiusuário e robustez Santos (2014).

O *Mysql* utiliza pouco espaço em disco e pode ser instalado adequada e rapidamente, foi utilizado neste trabalho para armazenar dados fictícios para testes do aplicativo *mobile* durante o desenvolvimento.

3.2 PROCEDIMENTOS

Para atingir os objetivos propostos neste trabalho as fases executadas são apresentadas na Figura 4 e descritas a seguir.

Figura 4 - Procedimentos para elaboração do projeto.



Os fluxos da Figura 4 são discriminadas logo a seguir.

1. Levantamento de requisitos: estudo sobre as funcionalidades que o aplicativo contém.
2. Seleção das tecnologias: definição das tecnologias utilizadas no desenvolvimento do aplicativo mobile e *Back-End* (API e Banco de Dados).
3. Diagrama do Banco de Dados: criação do diagrama com as tabelas e seus respectivos relacionamentos das entidades que contém, visando o entendimento dos requisitos coletados.
4. Estrutura do Banco de Dados: execução do diagrama no Banco de Dados, criando as tabelas, relacionamentos, *stored procedures* e *views*, utilizando a ferramenta Mysql.
5. Inserção de dados fictícios no Banco de Dados: inserção de dados nas tabelas do banco para simular os dados dos postos de combustíveis, como descrição, preços, formas de pagamentos dos combustíveis, entre outros.
6. Desenvolvimento API: criação das rotas que irá intermediar a comunicação do APP e o Banco de Dados, onde toda e qualquer comunicação com o BD será realizada pela API utilizando em conjunto as tecnologias PHP, Composer e Mysql. Sendo a última etapa do desenvolvimento do aplicativo.
7. Desenvolvimento Aplicação *Mobile* e Testes de Unidade: criação da a aplicação para os *smartphones* que possuírem as plataformas *Android* e iOS, testes de unidade que consiste na aplicação de testes nas assinaturas de entrada e saída do aplicativo e validação dos dados, que será utilizado os materiais Visual Studio e Xamarin Forms.

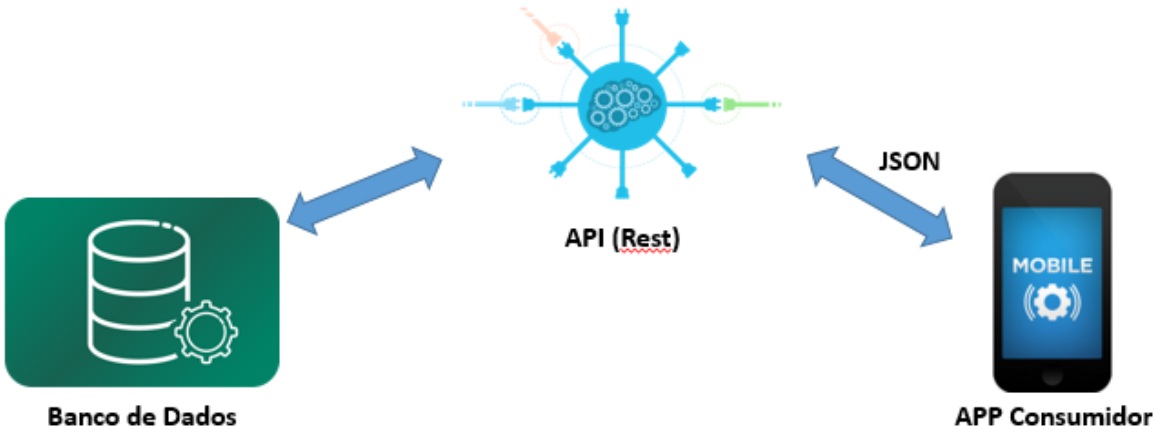
4 RESULTADOS E DISCUSSÕES

Nesta seção são apresentados os resultados obtidos no desenvolvimento da aplicação mobile, *API* e banco de dados, bem como alguns detalhes das funcionalidades que foram propostas neste trabalho. As subseções estão organizadas da seguinte forma: estrutura da aplicação, banco de dados, *API* e aplicativo.

4.2 ESTRUTURA DA APLICAÇÃO

Para melhor entendimento do funcionamento do projeto, a Figura 5 apresenta a estrutura da aplicação.

Figura 5 - Estrutura de comunicação do projeto.



O elemento banco de dados da Figura 5 representa o gerenciamento dos dados fornecidos pelos postos revendedores de combustíveis como: preços; descrição dos combustíveis; formas de pagamento; endereço e o que mais for informado. As requisições a esses dados, como por exemplo, as inserções, são manipuladas entre o banco de dados e a *API* (*Rest*) sendo que o *App mobile* não tem nenhum tipo de acesso diretamente com o BD.

O elemento *API* da Figura 5 mantém a lógica de negócio da aplicação onde os dados recebidos da aplicação *mobilem* são validados, manipulados e se necessário inseridos no BD. Esses dados são recebidos ou requisitados através de rotas que compõem a *API*. As informações trocadas entre a *API* e o *APP* estão em formatação *JSON* sendo possível ser manipulado em qualquer das tecnologias utilizadas nesse projeto.

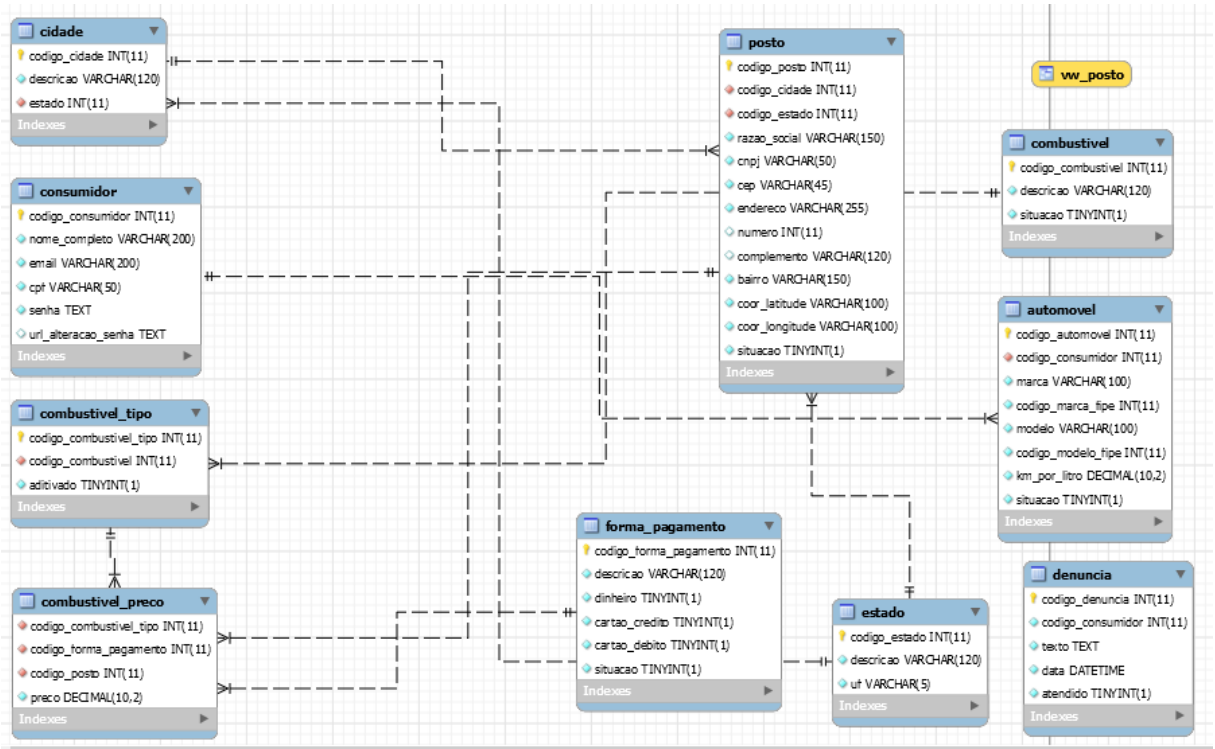
O elemento *APP* consumidor da Figura 5 mantém a comunicação apenas com a *API*, como por exemplo no caso das solicitações dos preços dos combustíveis, na localização dos

postos, nas formas de pagamento, entre outros. Esses dados são manipulados pela aplicação e apresentados ao consumidor.

4.2 BANCO DE DADOS

O banco de dados utilizado no desenvolvimento contém dados fictícios para auxiliar nos testes da aplicação sendo que o escopo desta monografia é a aplicação *mobile*, a base de dados criada simula a possível integração com o MP/TO que seria responsável pela coleta dos dados reais dos pontos revendedores. A Figura 6, que contém o diagrama da estrutura do banco de dados, demonstra as tabelas e *views* criadas juntamente com seus respectivos relacionamentos.

Figura 6 - Diagrama do banco de dados.



Durante o levantamento de requisitos foi planejada a estrutura mais adequada para o banco de dados, conforme Figura 6. A qualidade desse diagrama influência nos outros módulo como *API* e *APP*. Além das tabelas e *views* foram criadas as *stored procedures* que gerenciam a lógica de negócio do BD.

4.2.1 VIEWS

Views funcionam de forma similar às tabelas, sendo a diferença que as *views* podem ser consideradas tabelas virtuais onde pode-se criar consultas *sql's* do tipo *select* com vários

relacionamentos e disponibilizar os campos necessários para posterior consulta. A Figura 7, possui o relacionamento das tabelas posto, cidade e estado.

Figura 7 - View de dados dos postos revendedores.

```
CREATE
    ALGORITHM = UNDEFINED
    DEFINER = `root`@`localhost`
    SQL SECURITY DEFINER
VIEW `consumidor_atualizado`.`vw_posto` AS
    SELECT
        `p`.`codigo_posto` AS `codigo_posto`,
        `p`.`razao_social` AS `razao_social`,
        `p`.`cnpj` AS `cnpj`,
        `p`.`endereco` AS `endereco`,
        `p`.`numero` AS `numero`,
        `p`.`complemento` AS `complemento`,
        `p`.`bairro` AS `bairro`,
        `p`.`cep` AS `cep`,
        `p`.`coor_latitude` AS `coor_latitude`,
        `p`.`coor_longitude` AS `coor_longitude`,
        `p`.`situacao` AS `situacao_posto`,
        `e`.`codigo_estado` AS `codigo_estado`,
        `e`.`descricao` AS `descricao_estado`,
        `e`.`uf` AS `uf`,
        `c`.`codigo_cidade` AS `codigo_cidade`,
        `c`.`descricao` AS `descricao_cidade`
    FROM
        ((`consumidor_atualizado`.`posto` `p`
        JOIN `consumidor_atualizado`.`cidade` `c` ON ((`p`.`codigo_cidade` = `c`.`codigo_cidade`)))
        JOIN `consumidor_atualizado`.`estado` `e` ON ((`c`.`estado` = `e`.`codigo_estado`)))
```

A união das tabelas apresentadas na Figura 7 visa melhorar o desempenho das consultas, tendo como por exemplo a obtenção dos dados necessários para exibir as informações dos postos para o consumidor como razão social, endereço e preços dos combustíveis. A *view* pode ser consultada por qualquer *sql* como uma tabela denominada como “vw_posto” que contém relacionamento das tabelas Posto, Cidade e Estado.

Como um posto possui um endereço então é vinculado o código de uma cidade, e como uma cidade possui um estado é possível realizar a união dessas três tabelas e quando consultado um posto trazer os dados da cidade e do estado.

4.2.2 STORED PROCEDURES

Foi criada uma lógica de negócio dentro das *stored procedures* para que seja possível armazenar todas as *sql’s* de gerenciamento dos dados das tabelas do BD mantendo uma separação dos módulos. Com as *SP* criadas é possível receber valores de parâmetros da *API* e filtrar as consultas conforme necessário, como também retornar valores como por exemplo o *ID* de um registro recém inserido. A

Figura 8 apresenta a *stored procedure* “sp_combustivel”.

Figura 8 - Stored Procedure de consultas dos combustíveis.

```
CREATE DEFINER=`smartsisonline`@`localhost` PROCEDURE `sp_combustivel`(in p_codigo_combustivel int(11),
in p_aditivado tinyint(1), in p_codigo_forma_pagamento int(11), in p_codigo_posto int(11), in p_codigo_cidade int(11),
in p_ordenacao varchar(100), in p_acao varchar(50), out retorno int(11))
BEGIN
if (p_acao = 'buscarCombustiveisPorCidade') then
    SELECT p.codigo_posto, p.razao_social, p.cnpj, p.endereco, p.numero, p.complemento, p.bairro, p.codigo_cidade,
    p.descricao_cidade, p.codigo_estado, p.descricao_estado, p.uf, p.coor_latitude, p.coor_longitude, p.cep,
    (select JSON_ARRAYAGG(JSON_OBJECT("codigo_combustivel", codigo_combustivel, "preco", preco,
    "descricao_combustivel", descricao_combustivel, "aditivado", aditivado,
    "codigo_forma_pagamento", codigo_forma_pagamento, "descricao_forma_pagamento", descricao_forma_pagamento,
    "dinheiro", dinheiro, "cartao_credito", cartao_credito,
    "cartao_debito", cartao_debito))
    from vw_combustivel where codigo_posto = p.codigo_posto and codigo_combustivel = p_codigo_combustivel
    and aditivado = p_aditivado order by preco asc) AS precos
    FROM vw_posto p where p.codigo_cidade = p_codigo_cidade GROUP BY p.codigo_posto order by razao_social asc;
```

Na

Figura 8 representa-se o armazenamento de todas as *sql’s* para consultas de combustíveis dos postos revendedores, condicionais e outros métodos são utilizados para que a lógica de negócio funcione. O condicional “if” determina que tipo de busca a chamada da *API* quer realizar, buscar combustível por cidade ou por posto.

Foi necessário utilizar na *sql* que busca os dados dos combustíveis os métodos *JSON_OBJECT* e *JSON_ARRAYAGG*, o primeiro para criar um objeto dos dados retornados do *select* mais interno e o segundo é um *array* utilizado para guardar os objetos criados. O *array* criado pelo *JSON_ARRAYAGG* é agrupado com os dados do primeiro *select* e então esses dados filtrados são retornados da *API*.

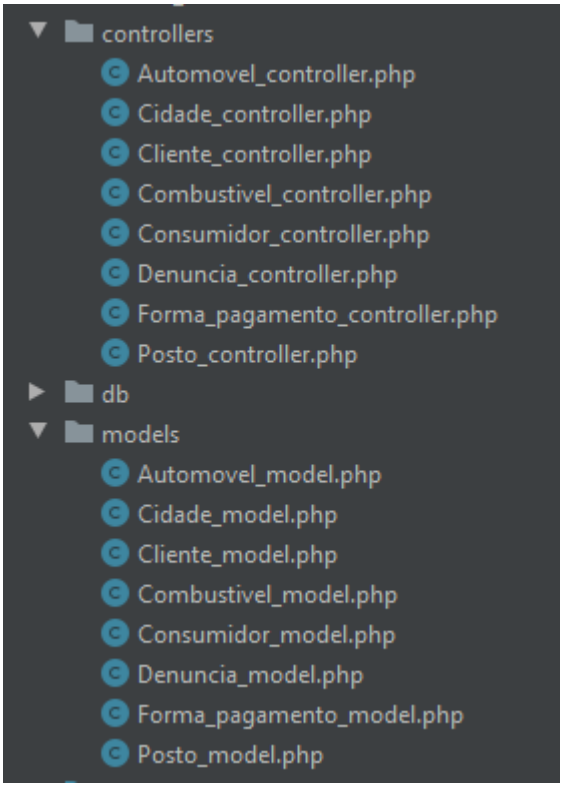
Vale ressaltar que cada posto possui vários combustíveis e cada combustível pode conter várias formas de pagamento e cada forma de pagamento pode haver apenas um preço. Para organizar esses dados foi necessário trazer em uma linha os dados do posto e dos combustíveis com seus respectivos preços. Com isso percebe-se a necessidade dos métodos mencionados acima para realizar a função de agregação para os dados.

4.3 API

A *API* mantém a intermediação entre o *APP* e o BD ao realizar o tratamento de dados recebidos e enviados ao *APP*, sendo que essa transição na comunicação utiliza codificação *JSON* na formatação dos dados. No desenvolvimento da *API* foi utilizado a arquitetura *MVC*

sem utilizar a camada *View* já que o sistema não utiliza exibições, mas apenas o tratamento de dados. Na Figura 9 é apresentada a estrutura das classes *Model* e *Controllers*. Vale ressaltar que a *API* é formada por rotas que servem o *Front-End* da aplicação.

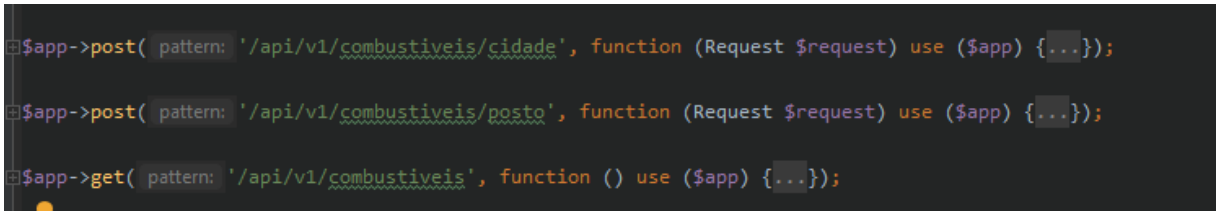
Figura 9 - Organização das classes Model e Controllers da API.



A Figura 9 apresenta as classes dos *controllers* e dos *models*, os *controllers* são utilizadas para tratamento de dados recebidos e enviado ao *front-end* como validação dos dados recebidos e não possui nenhuma comunicação direta com o BD. As classes em *models* são utilizadas apenas para comunicação com BD mas não possui nenhum tipo de *sql* no código e contém métodos de comunicação com as *stored procedures*.

As rotas utilizam os verbos do *HTTP* como o *Post*, *Get*, *Put* e *Delete* para diferenciar os tipos de procedimento que o *front-end* irá realizar, ao chamar uma rota nenhum procedimento é realizado no seu escopo mas sim criado uma instância do *controller*. É possível chamar o método correspondente para realizar o processamento dos dados, a Figura 10 demonstra as rotas de busca de combustíveis.

Figura 10 - Rotas para buscar informações de combustíveis.



A Figura 10 demonstra algumas rotas do tipo *Post* para que seja possível passar alguns parâmetros de filtros e outra do verbo *Get* onde não é necessário passar nenhum dado para essa rota. O final das rotas que contém "/cidade" e "/posto" essas retornam informações completas dos combustíveis com suas relações com os postos como informações dos postos, combustível e preços e a "/combustiveis" apenas informações dos combustível como código, descrição e situação.

4.4 APLICATIVO

O *front-end* do *APP* foi desenvolvido utilizando *XAML* enquanto o *back-end* utilizou *C#* sendo que a combinação de ambos vem do *framework Xamarin Forms*, antes de iniciar a criação das telas do *APP* foi estruturado o banco de dados e a *API*. Essa estratégia proporcionou maior agilidade na etapa de criação do *layout* das telas.

O desenvolvimento do *layout* das telas iniciou-se pelo *splash*, seguindo com o menu de opções, a tela de listagem de postos, combustíveis e preços e um *pop up* com algumas opções para filtrar essas informações e entre outras funcionalidades complementares. A seguir será apresentado um conjunto de figuras que demonstram o aplicativo instalado e em funcionamento em um dispositivo *Motorola* Moto G6 que utiliza o SO *Android*.

A parte de testes durante o desenvolvimento para o *iOS* foi realizado com o auxílio de um emulador rodando em uma máquina virtual com sistema operacional *Mac*, o dispositivo emulado foi o *iPhone* 7 com *iOS* 12.2. O comportamento da aplicação como do *layout* são exatamente iguais para os dois sistemas operacionais, sendo a única diferença os controles abordados nas Figuras Figura 11Figura 12
Figura 13, com isso o restante das explicações abordará as telas do *Android*.

Figura 11 – Postos com iOA, com seus respectivos combustíveis da cidade de Palmas.

A Figura 11, aborda a listagem dos postos com seus respectivos combustíveis da cidade de Palmas, o *App* está implantado em um emulador que está simulando o *iPhone* 7 com *iOS* 12.2.

Figura 12 - Pop Up dos filtros disponíveis para consulta de postos por cidade, iOS.

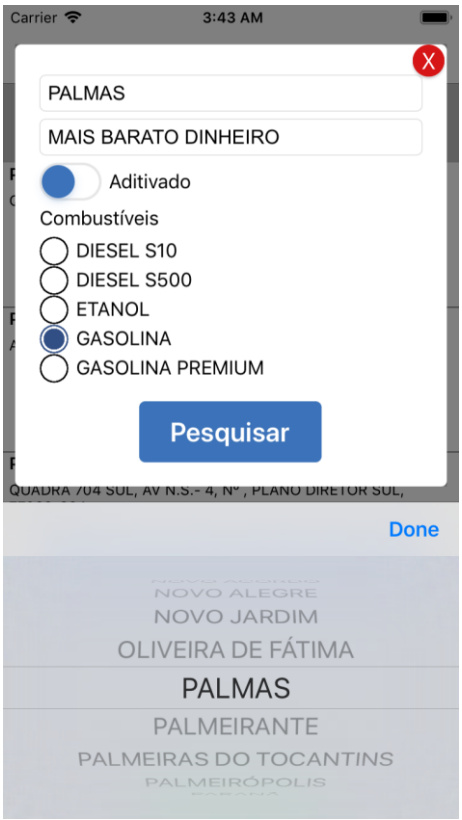
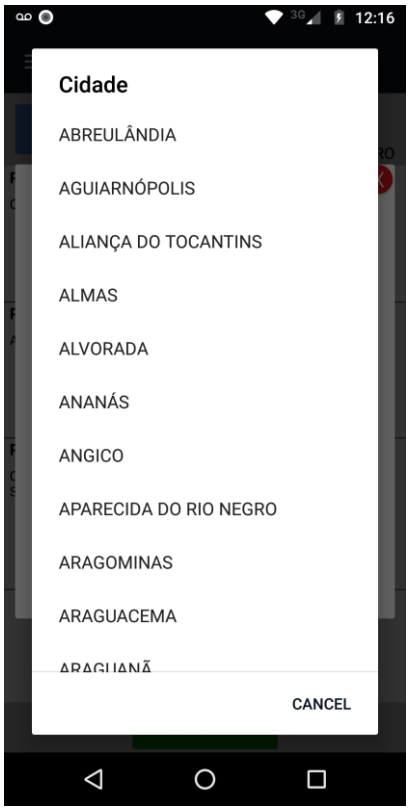


Figura 13 - Listagem de cidades em dispositivo *Android*.



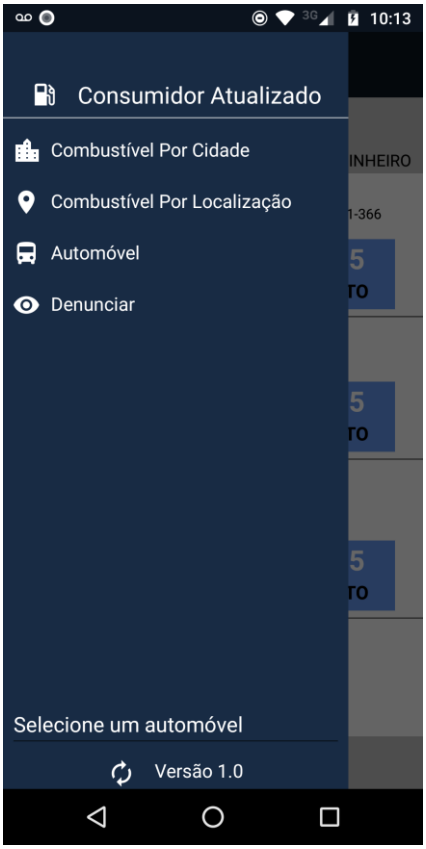
Na Figura 12, a diferença para o *Android* se dá pelo controle *Picker* que dá a possibilidade de listar várias descrições e que o usuário pode estar selecionando apenas uma opção, o comportamento de exibição deste controle no *iOS* fica na parte inferior em forma de uma lista. O *Android* exibe esse mesmo controle em forma de um *pop up* com a listagem das descrições que nesse caso são os nomes das cidades, conforme

Figura 13. A seguir as explicações das telas do aplicativo se dá pelo sistema operacional *Android*.

Figura 14 - Splash ao iniciar o APP.

A Figura 14 demonstra a tela de *splash* com uma imagem de duas bomba de abastecimento, pois a figura se relaciona com o foco deste projeto que é disponibilizar um aplicativo para consulta de preços de combustíveis. Essa tela tem como objetivo chamar a atenção do usuário quando ele abre o aplicativo, sendo que o tempo que essa tela fica aberta refere-se ao período de carregamento das dependências do aplicativo.

Figura 15 - Menu com opções de navegação e informações sobre o APP.



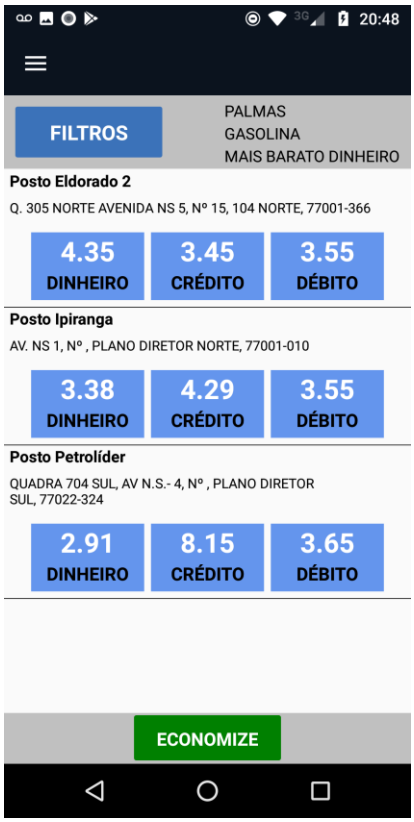
A Figura 15, apresenta o menu de navegação para o consumidor que pode selecionar a opção que deseja utilizar, sendo:

- Combustível por cidade: lista os postos revendedores de combustíveis por cidade. O aplicativo captura de início a localização do usuário e filtra a cidade para que sejam pré-definidos os filtros de consulta da listagem, como por exemplo a cidade, o combustível e a ordenação.
- Combustível por localização: realiza a mesma funcionalidade do item anterior, sendo o diferencial o fato do filtro não ser por cidade, mas sim, por uma distância em quilômetros entre o consumidor e os postos. Essa distância pode ser ajustada pelo usuário na faixa de 1 km a 100 km.
- Automóvel: apresenta a lista dos automóveis do consumidor, e para cada registro permite realizar a exclusão, nessa mesma tela há a possibilidade de realizar o cadastro do veículo com as opções de marca, modelo e consumo de litros por quilômetros rodados. O cadastro dos automóveis é necessário para auxiliar no cálculo de sugestão

dos postos de melhor custo benefício quando utilizado as opções “Combustível Por Cidade e Localização”.

- Denunciar: possibilita o registro de denúncias de revendedores que divergem os preços entre a plataforma e os postos, como também permite listar todas essas denúncias com opção exclusão e acompanhamento do status das denúncias, que pode ser concluído ou em atendimento.
- Selecionar um automóvel: lista os automóveis cadastrados pelo consumidor permitindo a seleção do veículo utilizado no momento da utilização do *App*. O veículo selecionado irá impactar na listagem dos postos de melhor custo benefício com base em seu consumo de combustível.

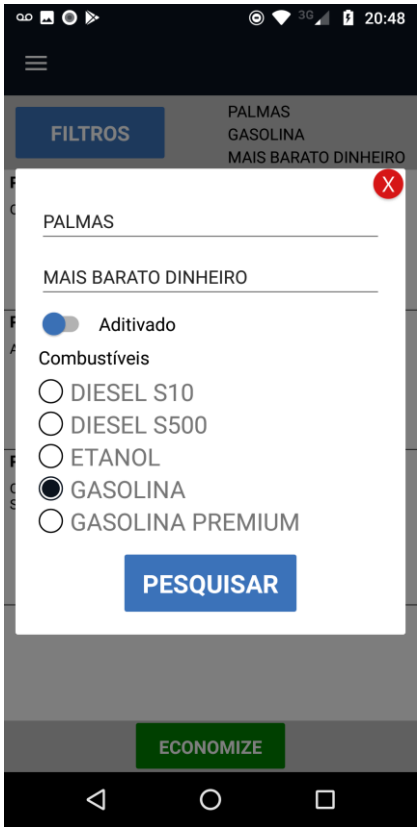
Figura 16 - Listagem dos postos com seus respectivos combustíveis da cidade de Palmas.



A Figura 16 apresenta o carregamento da listagem com os filtros pré-definidos que são: a cidade, o combustível e a ordenação. O *APP* ao ser iniciado solicita permissão para utilizar o *GPS* do dispositivo, pois o filtro padrão da cidade é selecionada de acordo com a localização do consumidor. A localização atual do dispositivo é capturada e gera um par de coordenadas que então é consultada através da *API* do *Google* (*Geocoding API*) que retorna um endereço da coordenada e através disso filtrado a cidade.

O botão FILTROS ao ser clicado abre um Pop Up que disponibiliza os filtros para que o consumidor possa selecionar a cidade, a ordenação, o tipo do combustível (aditivado ou não), conforme pode ser verificado na Figura 17.

Figura 17 - Pop Up dos filtros disponíveis para consulta de postos por cidade.



A Figura 17 lista as cidades do estado do Tocantins, as possibilidades de ordenação são mais barato em dinheiro; crédito; débito e mais caro em dinheiro; crédito; débito. Não é possível selecionar mais de uma ordenação, pois como o combustível possui mais de uma forma de pagamento, sendo elas, dinheiro, crédito e débito isso traria problemas na *Sql* do banco de dados.

O botão ECONOMIZE demonstrado na Figura 16, quando acionado abre uma tela de listagem de postos de melhor custo benefício com os mesmos filtros aplicados na tela inicial da opção Combustível Por Cidade. O cálculo para filtrar esses postos é demonstrado na Figura 18.

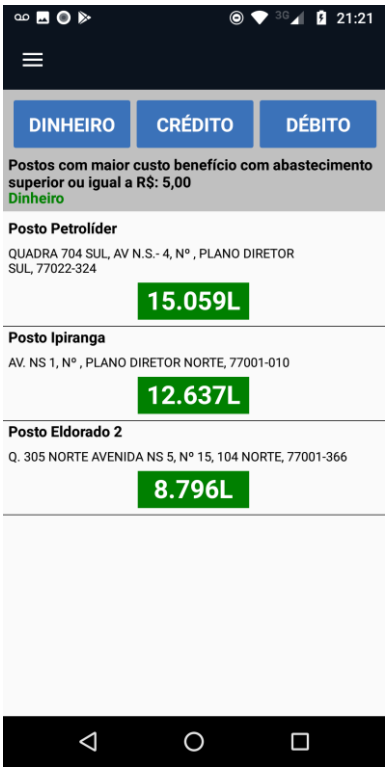
Figura 18 - Planilha com simulação dos cálculos para dois postos.

	Distância Até o Posto KM	Posto	Vlr. Combustível R$	Consumo Veículo	Qtd. Litros Gasto P/ Chegar	Vlr. Abastecer R$	Qtd. Litros	Qtd. Litros - Qtd. Gastar P/ Chegar
Consumidor	10.00	Posto 1	1.00	8.00	1.25	5.00	5.00	3.75
Consumidor	5.00	Posto 2	3.00	8.00	0.63	5.00	1.67	1.04
Consumidor	10.00	Posto 1	1.00	8.00	1.25	2.00	2.00	0.75
Consumidor	5.00	Posto 2	3.00	8.00	0.63	2.00	0.67	0.04
Consumidor	10.00	Posto 1	1.00	8.00	1.25	150.00	150.00	148.75
Consumidor	5.00	Posto 2	3.00	8.00	0.63	150.00	50.00	49.38
Consumidor	10.00	Posto 1	1.00	8.00	1.25	1.00	1.00	-0.25
Consumidor	5.00	Posto 2	3.00	8.00	0.63	3.00	1.00	0.38
					B/E		G/D	H-F

A Figura 18 apresenta o cálculo da distância entre o consumidor e os postos de combustíveis com base em suas coordenadas geográficas e a quantidade de litros gastos para o consumidor chegar no posto de abastecimento. O cálculo é composto pela divisão da distância e o consumo do veículo por quilômetros rodados, logo na sequência, o aplicativo calcula a quantidade de litros total que o valor de abastecimento irá proporcionar adquirido pela divisão do valor de abastecimento e o valor do combustível.

Essa quantidade de litros total deve ser subtraída da quantidade de litros gastos para chegar no posto de combustível, e então é gerado o resultado sobre a quantidade de litros final que o consumidor terá em seu tanque ao término do abastecimento que se dá pela última coluna da planilha. Quando esse valor for negativo, o posto é retirado da lista de melhores custos benefício, pois para o consumidor não será uma boa escolha, o valor de abastecimento predefinido no algoritmo é de 5 reais utilizado como base para o cálculo esse valor é informado ao usuário conforme Figura 19.

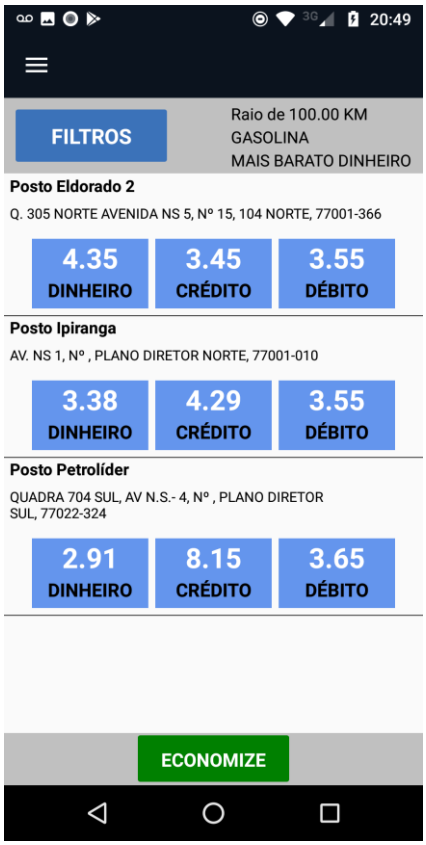
Figura 19 - Listagem dos postos revendedores de combustíveis de melhores custos benefício.



A Figura 19 possui a listagem por forma de pagamento como dinheiro, crédito e débito, sendo que o valor do combustível pode ser diferente para cada uma, o posto de melhor custo benefício considera a ordenação dos postos de combustíveis de cima para baixo e do total de litros adquiridos que está na cor verde. O consumidor pode selecionar entre as formas de pagamento que melhor lhe convier.

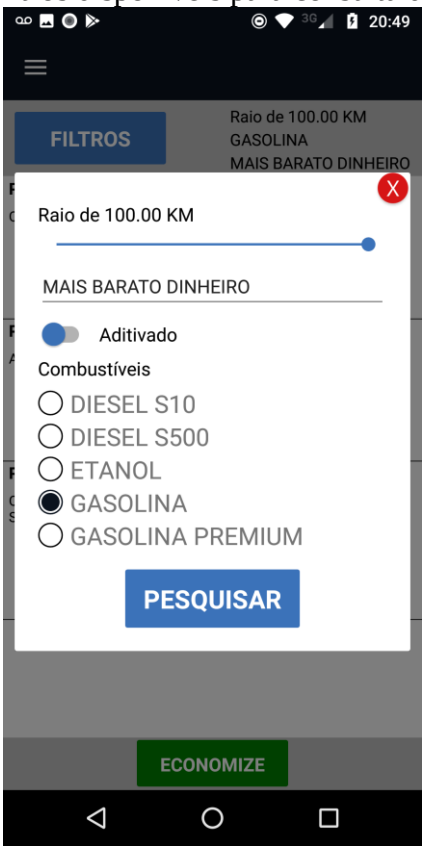
A opção “Combustível por localização” herda as características da tela de Combustível por cidade sendo que o diferencial é o modo de busca que ao invés de ser por cidade é por distância entre o consumidor e os postos de combustíveis. A alteração no filtro pode ser observada conforme Figura 20.

Figura 20 - Listagem dos postos com seus respectivos combustíveis em uma distância de 100 km do consumidor.



A Figura 20 demonstra os controles disponíveis, que ao ser acionada o botão de FILTROS possui o mesmo comportamento da tela da Figura 14, sendo, excluída a listagem de cidades e adicionada um controle deslizante onde o consumidor pode ajustar o raio de quilômetros conforme apresentado na Figura 21.

Figura 21 - Pop Up dos filtros disponíveis para consulta de postos por localização.



De acordo que o controle deslizante é arrastado o valor no Texto “Raio de 100.00 KM” é alterado e demonstra ao usuário qual o valor que está sendo selecionado, o raio pode ser ajustado no mínimo em 1 e no máximo em 100 quilômetros. O botão ECONOMIZE aciona a mesma funcionalidade da Figura 19 com a diferença que a busca dos postos que é por distância, mas continua o mesmo algoritmo de cálculo de melhor custo benefício como também o *layout*.

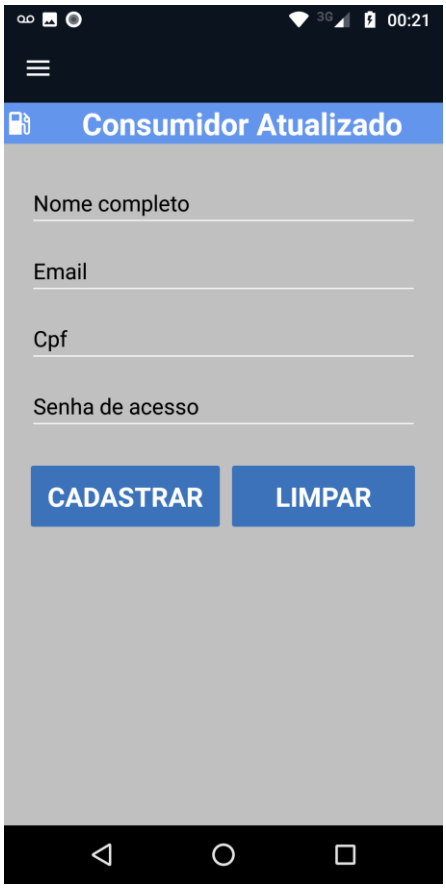
As opções Automóvel e Denunciar do menu requerem autenticação de usuário composto por CPF e Senha, o consumidor ao se deparar com a tela de login se não houver um usuário cadastrado. Na mesma tela pode se solicitar o cadastro de um usuário conforme Figura 22.

Figura 22 - Tela de login de usuário.

A Figura 22 demonstra que o aplicativo requer autenticação do consumidor para listar, cadastrar e excluir um automóvel ou denúncias. O usuário pode estar selecionando a opção “Lembrar senha?” para que não seja necessário digitar novamente os dados em um login futuro esses dados ficam armazenado no próprio dispositivo do usuário. Se o consumidor não houver usuário pode acionar o botão CADASTRAR para realizar o cadastro, ou ENTRAR para se autenticar no aplicativo.

A tela de cadastro requer alguns dados do consumidor como Nome completo, E-mail, CPF e uma Senha de acesso, a Figura 23 demonstra a tela de cadastro.

Figura 23 - Tela de cadastro de usuário.



A Figura 23 apresenta os campos necessários para cadastrar um usuário, a requisição desses dados são essenciais para manter um histórico de quem registrou uma denúncia e manter o vínculo dos veículos cadastrados com o consumidor, os dados são validados na submissão pela *API* como se o e-mail está composto corretamente, CPF é válido e nenhum campo se encontra vazio.

A opção Automóvel lista todos os veículos cadastrados pelo consumidor o maior objetivo dessa funcionalidade é capturar o consumo do veículo que está sendo utilizado pelo usuário no momento da utilização do *App*. Com o consumo do veículo é possível realizar com mais precisão o algoritmo de cálculo de melhor custo benefício dos postos revendedores de combustíveis, a

Figura 24 ilustra a listagem dos automóveis.

Figura 24 - Listagem de automóveis cadastrados pelo consumidor.

Figura 25 - Opção de excluir um automóvel cadastrado.

A

Figura 24 ilustra a tela de listagem de automóveis que possibilita o cadastramento do veículo acionando o botão CADASTRAR. Quando um usuário pressiona sobre um veículo cadastrado o aplicativo disponibiliza a opção de excluir este, conforme Figura 25 enquanto a Figura 26 contém a tela de cadastro de automóveis.

Figura 26 - Tela de cadastro de automóveis.

A Figura 26 apresenta os controles que são possíveis selecionar a marca, modelo e digitar o consumo atual do veículo que está sendo cadastrado, as marcas e modelos dos veículos estão sendo adquiridos de uma *API* da Fipe. Ao selecionar uma marca automaticamente é carregado no segundo *select* todos os modelos vinculados a esta, e o consumo pode ser digitado como um número decimal de 2 casas após o ponto.

O botão CADASTRAR da Figura 26 demonstra a realização do cadastro do veículo no sistema armazenando os dados do veículo para o usuário logado no aplicativo. O botão LIMPAR, limpa todos os campos da tela.

A opção Denunciar traz a listagem de todas as denúncias realizadas pelo usuário logado no aplicativo, e pode gerenciar funcionalidades como excluir ou cadastrar uma nova solicitação de denúncia. A tela de listagem e exclusão está definida conforme apresentado nas Figuras Figura 27

Figura 28 respectivamente.

Figura 27 - Listagem de denúncias de usuário logado no aplicativo.

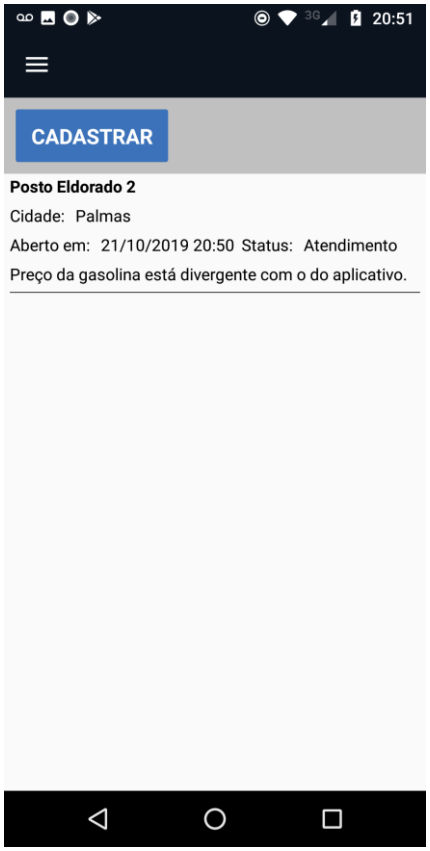


Figura 28 - Opção de excluir uma denúncia cadastrada.

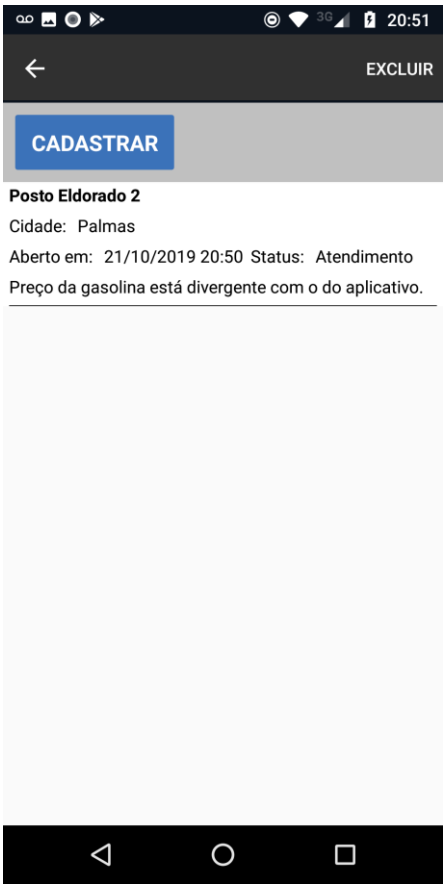


Figura 29 - Tela de cadastro de denúncia.

A Figura 29 demonstra a tela de cadastro de denúncia onde solicita algumas informações ao usuário como a cidade em que está localizada o revendedor, posto de combustível e detalhes sobre o motivo da denúncia. Ao selecionar uma cidade automaticamente o aplicativo realiza uma busca de todos os postos daquela cidade e insere-os na listagem de postos (Postos), esse filtro proporciona uma facilidade ao usuário no momento de selecionar um posto de interesse.

Acionando o botão CADASTRAR então o aplicativo realiza o cadastro da solicitação no banco de dados vinculando-a ao usuário logado, logo se pressionando o botão LIMPAR os campos são todos restaurados para os valores padrão.

5 CONSIDERAÇÕES FINAIS

Neste trabalho foi proposto e desenvolvido uma aplicação *mobile* para *Android* e *iOS* que permite aos usuários consultar os preços dos combustíveis em tempo real dos postos revendedores do estado do Tocantins. O desenvolvimento da aplicação foi motivado com base no aplicativo Olho na Bomba e pelo fato de não haver nenhum aplicativo que realizasse essa funcionalidade em tempo real no Tocantins.

Outro ponto observado para buscar esta implementação deve-se a consulta constante de economia dos consumidores no momento de abastecimento dos veículos, a dificuldade em encontrar o posto ideal ou pelo menos o posto de melhor custo benefício. Com o uso do aplicativo os consumidores poderão encontrar o posto de melhor custo benefício com base no consumo do veículo, distância entre os postos e os atuais preços dos combustíveis.

O desenvolvimento foi dividido em três camadas, sendo, banco de dados: foi criado um BD utilizando o *Mysql* para simular os possíveis dados coletados pelo MP/TO; *API*: armazena toda a lógica de negócio da aplicação como tratamento de dados e comunicação direta com o BD e o *App*, umas das tecnologias utilizadas foi o PHP e o *Silex*; *Front-End*: com o *framework Xamarin Forms* da *Microsoft* foi possível disponibilizar o *App* para *Android* e *iOS* em um mesmo código.

No desenvolvimento da aplicação muita atenção e tempo de desenvolvimento foi destinada ao balanceamento de carga das requisições e do BD, por se tratar de uma aplicação *mobile* é necessário se pensar no consumo do pacote de internet do dispositivo. O App ao requerer dados da *API* sempre se mantém no mínimo possível e no necessário não requisitando dados excedentes, todas as consultas *Sql's* foram otimizadas utilizando os recursos que o *Mysql* disponibiliza como *stored procedures*, *views*, *json_arrayagg*, *json_object* e dentre outras.

Ao final do desenvolvimento foram realizados testes de funcionalidades com o *App* implantado em um dispositivo físico *Android* e em um emulador com sistema operacional *iOS*. A aplicação foi submetida a testes de validação para verificar e analisar se as informações enviadas e retornadas pela *API* estavam corretas, como resultado tem-se uma aplicação finalizada e funcional.

Para trabalhos futuros, novas funcionalidades podem ser adicionadas à aplicação para assim melhorar a experiência do usuário, como por exemplo, módulo de notificação: quando houver algum posto de melhor custo benefício com base na localização atual do usuário;

módulo de anúncios: inserir anúncios de empresas para custear gastos de serviços como servidores, taxas das lojas da Google e *Apple Store*; portal de gerenciamento dos postos: módulo *web* de gerenciamento dos postos de combustíveis possibilitando o revendedor está cadastrando e gerenciando os preços dos combustíveis e como informações do próprio posto.

REFERÊNCIAS

ALMEIDA, EDMAR LUIS FAGUNDES DE; OLIVEIRA, PATRICIA VARGAS DE; LOSEKANN, LUCIANO. **Impactos da contenção dos preços de combustíveis no Brasil e opções de mecanismos de precificação**. Brazilian Journal of Political Economy, v. 35, n. 3, p. 531-556, 2015.

BERNARDES, TATIANA FREITAS; MIYAKE, MARIO YOSHIKAZU. **Cross-platform mobile development approaches: A systematic review**. IEEE Latin America Transactions, v. 14, n. 4, p. 1892-1898, 2016.

BIOCOMBUSTÍVEIS, AGÊNCIA NACIONAL DE PETRÓLEO GÁS NATURAL E. **Consumo de combustíveis no Brasil subiu 0,4% na comparação entre 2017 e 2016**. 2019. Disponível em: <http://www.anp.gov.br/noticias/4334-consumo-de-combustiveis-no-brasil-subiu-0-4-na-comparacao-entre-2017-e-2016>. Acesso em: 06 mar. 2019.

CARVALHO, LUCAS. **Android cresce no Brasil e aumenta distância para iOS e Windows Phone**. 2017. Disponível em: <https://olhardigital.com.br/noticia/android-cresce-no-brasil-e-aumenta-distancia-para-ios-e-windows-phone/68023>. Acesso em: 19 maio 2019.

CARVALHO, VERSANNA. **Olho na Bomba adquire relevância social, dizem MP e UFG**. 2018. Disponível em: <https://www.ufg.br/n/112291-olho-na-bomba-adquire-relevancia-social-dizem-mp-e-ufg>. Acesso em: 06 mar. 2019.

DA SILVA, MARCELO MORO; SANTOS, MARILDE TEREZINHA PRADO. **Os paradigmas de desenvolvimento de aplicativos para aparelhos celulares**. Revista TIS, v. 3, n. 2, 2014.

FREIRE, PEDRO J.; RIBEIRO, RUI. **Revisão de Literatura de Frameworks de Desenvolvimento Móvel Multiplataforma CAPSI/2013**. 2013.

GOIÁS, MINISTÉRIO PÚBLICO DO ESTADO DE; GOIÁS, Universidade Federal de. **Aplicativo Olho na Bomba**. 2017. Disponível em: <http://www.mpgo.mp.br/portal/conteudo/aplicativo-olho-na-bomba#.XPKw3IhKjIV>. Acesso em: 25 maio 2019.

GOMES, RAFAEL CAVEARI, JEAN ALVES R. FERNANDES, VINICIUS CORRÊA FERREIRA. **Sistema Operacional Android**. Universidade Federal Fluminense, 2012.

KRONBAUER, ARTUR HENRIQUE. **Um modelo de avaliação de usabilidade de aplicativos para smartphones baseado na captura automática de interações com o usuário**. 2017.

KSTRO, NANDO. **Conhecendo e instalando o Silex**. 2015. Disponível em: <https://tableless.com.br/conhecendo-e-instalando-o-silex/>. Acesso em: 14 maio 2019.

LOPES, SÉRGIO. **Aplicações mobile híbridas com Cordova e PhoneGap**. Editora Casa do Código, 2016.

LUCENA, WENNER GLAUCIO LOPES, JACQUELINE ARAGÃO DE MEDEIROS, ALANE LIMA DE OLIVEIRA, SIMONE PEREIRA DA SILVA, GILENO FERNANDES MARCELINO. **Finanças comportamentais: fatores que influenciam os consumidores na hora da compra**. Estudos do CEPE, n. 33, p. 93-126, 2011.

MARJOTTA-MAISTRO, MARTA CRISTINA; BARROS, GS DE C. **Relações comerciais e de preços no mercado nacional de combustíveis**. In: CONGRESSO BRASILEIRO DE ECONOMIA E SOCIOLOGIA RURAL. 2002.

MCILROY, STUART; ALI, NASIR; HASSAN, AHMED E. **Fresh apps: an empirical study of frequently-updated mobile apps in the Google play store**. Empirical Software Engineering, v. 21, n. 3, p. 1346-1370, 2016.

NAVARRO, CAMILA; FRESSATTI, WYLLIAN. **Ferramentas para o Desenvolvimento em C#**. Paranavaí, Unipar, 2014.

PRACIANO, ELIAS. **Os benefícios e as vantagens do PHP**. 2014. Disponível em: <https://elias.praciano.com/2014/02/15-beneficios-e-vantagens-do-php/>. Acesso em: 13 maio 2019.

PREZOTTO, EZEQUIEL DOUGLAS; BONIATI, BRUNO BATISTA. **Estudo de frameworks multiplataforma para desenvolvimento de aplicações mobile híbridas**. Anais do EATI, ano, v. 4, p. 72-79, 2014.

RADI, AMER A. **Evaluation of Xamarin Forms for MultiPlatform Mobile Application Development**. 2016.

SANTOS, ADRIANO. **MySQL: Quem é você?**. 2014. Disponível em: <https://www.devmedia.com.br/mysql-quem-e-voce/1752>. Acesso em: 14 abril 2018.

SCHMITZ, LEONARDO. **Análise de ferramentas de desenvolvimento multiplataforma para criação de aplicativos móveis**. 2016.

SILVA, TOMAZ ROCHA DA. **Um estudo de interação com displays grandes usando dispositivos iOS**. 2011.

SIN, DAVID; LAWSON, ERIN; KANNOORPATTI, KRISHNAN. **Mobile web apps-the non-programmer's alternative to native applications**. In: 2012 5th International Conference on Human System Interactions. IEEE, 2012. p. 8-15.

TAVARES, HENRIQUE LEAL. **Introdução a Desenvolvimento de Aplicações Híbridas**. Revista Eletrônica eF@ tec, v. 6, n. 1, p. 11-11, 2016.

TR NSITO, DEPARTAMENTO NACIONAL DE. **Frota de Veículos - 2019**. 2019. Disponível em: <http://www.denatran.gov.br/estatistica/639-frota-2019>. Acesso em: 11 maio 2019.

ZEMEL, TÁRCIO. **Composer: a evolução PHP**. 2012. Disponível em: <http://desenvolvimentoparaweb.com/php/composer-a-evolucao-php/>. Acesso em: 13 maio 2019.

APÊNDICES

ANEXOS