**UNIVERSIDADE REGIONAL DE BLUMENAU**

**CENTRO DE CIÊNCIAS EXATAS E NATURAIS**

**CURSO DE CIÊNCIA DA COMPUTAÇÃO – BACHARELADO**

# GOOD-BUY: UM PROTÓTIPO PARA COMPARTILHAMENTO DE OFERTAS OU PRODUTOS

**LEONARD WILLIAM DE AZEVEDO PEGLER**

**BLUMENAU**
**2017**

**LEONARD WILLIAM DE AZEVEDO PEGLER**

# GOOD-BUY: UM PROTÓTIPO PARA COMPARTILHAMENTO DE OFERTAS OU PRODUTOS

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**

# GOOD-BUY: UM PROTÓTIPO PARA COMPARTILHAMENTO DE OFERTAS OU PRODUTOS

Por

**LEONARD WILLIAM DE AZEVEDO PEGLER**

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. Simone Erbs da Costa, Especialista – FURB |

Blumenau, 11 de dezembro de 2017

Dedico este trabalho a minha família, especialmente a minha mãe Sônia Marinez por ter sempre me fornecido todo o suporte necessário ao meu desenvolvimento.

## AGRADECIMENTOS

A minha mãe Sônia Marinez e a minha irmã Rebecca Amanda, por ser meu alicerce e me auxiliar em todas as decisões de minha vida.

Aos meus amigos, por compreenderem o meu afastamento e minha ausência durante a elaboração deste trabalho.

Ao meu amigo Lucas Gomes por ter me ajudado na composição de elementos visuais deste trabalho.

Ao meu orientador Aurélio, por sempre me ajudar a observar um complexo problema sob uma outra perspectiva que me auxiliava na resolução dos empecilhos enfrentados no desenvolvimento deste trabalho.

A minha namorada Luana, por compreender minha ausência durante o período de desenvolvimento deste trabalho.

O sucesso nasce do querer, da determinação e persistência em se chegar a um objetivo. Mesmo não atingindo o alvo, quem busca e vence obstáculos, no mínimo fará coisas admiráveis.

José de Alencar

# RESUMO

Este trabalho descreve o desenvolvimento de um aplicativo que utiliza conceitos do consumo colaborativo para a resolução de um problema de consumo social com foco na disseminação de informações, tendo como objetivo apoiar o compartilhamento de ofertas, indicando ao usuário o melhor estabelecimento para realizar suas compras. O usuário poderá optar por autenticar-se através do Facebook ou via cadastro próprio. A partir disso, ele poderá cadastrar ofertas, monitorar preço de promoções, criar e participar de grupos com interesses em comum, elaborar uma lista de compras, simular o preço desta lista de produtos para obter o estabelecimento com o melhor custo benefício, consultar as últimas ofertas cadastradas por usuários e compartilhar uma oferta com um grupo específico ou diretamente com outro usuário da aplicação. O desenvolvimento do aplicativo foi realizado com o *framework* da Microsoft Xamarin.Forms, juntamente com bibliotecas nativas do Facebook e de persistência local através do SQLite com a bilioteca `Microsoft.Azure.Mobile.Client`. O Microsoft Azure foi utiliza como servidor pois provê serviços de autenticação, hub de notificações *push* e infraestrutura de comunicação. Os resultados dos testes com os usuários demonstraram que a aplicação possui uma boa confiança entre os usuários, com aceitação e desempenho satisfatórios, apesar de existir alguns pontos a serem melhorados em questão de aparência, usabilidade e população inicial de dados.

Palavras-chave: Ofertas. Lista de compras. Compartilhamento. Node.js. Azure. Xamarin.

# ABSTRACT

This work describes the development of an application that uses concepts of collaborative consumption to solve a problem of social consumption focused on the dissemination of information, aiming to support the sharing of offers, indicating to the user the best establishment to carry out their purchases. The user can choose to authenticate himself in the application through Facebook or via own register. From this, the user can register offers, monitor price promotions, create and participate in groups with interests in common, draw up a shopping list, simulate the price of this list of products to get the establishment with the best cost benefit, check the latest offers registered by users and share an offer with a specific group or directly with another application user. The application development was accomplished with Microsoft's framework Xamarin.Forms, along with native Facebook libraries and local persistence through SQLite with the `Microsoft.Azure.Mobile.Client` library. Microsoft Azure was used as the server, to which a Node.js. application was hosted. Azure was also used to provide authentication services, push notifications hub and communication infrastructure. The user test results demonstrated that the application has good confidence among users, with satisfactory acceptance and performance, although there are some points to be improved in terms of appearance, usability, and initial data population.

Key-words: Offers. Shopping List. Sharing. Node.js. Azure. Xamarin.

# LISTA DE FIGURAS

# LISTA DE QUADROS

# LISTA DE TABELAS

# LISTA DE ABREVIATURAS E SIGLAS

ABNT – Associação Brasileira de Normas Técnicas

API – Application Programming Interface

APNS – Apple Push Notification Service

FCM – Firebase Cloud Messaging

HTTP – Hypertext Transfer Protocol

IDE – Integrated Development Environment

JSON – JavaScript Object Notation

MER – Modelo de Entidade Relacionamento

REST – Representational State Transfer

SIM – Subscriber Identity Module

SQL – Structured Query Language

WNS – Windows Push Notification Services

# SUMÁRIO

# 1 INTRODUÇÃO

Segundo Rojo (1998), para criar entusiasmo ou conquistar clientes, o fornecedor precisa oferecer produtos de qualidade, bom atendimento e preço. Para Velluto (2012), a pesquisa de preço acaba sendo uma ferramenta fundamental para que o consumidor estabeleça este tripé comportamental e, que as empresas devem ficar atentas a estas relações de consumo.

De acordo com o PROCON-SC (2013) para a ceia de final de ano, o item que mais teve variação foi a maionese, com 118,72%. Entre as carnes, o valor do peru é o que mais varia, com 34,34% de diferença. Em outra pesquisa referente a material escolar realizada em seis estabelecimentos de Curitiba pelo PROCON-PR (2017), mostra que 35 itens (entre cadernos, canetas, colas, giz de cera, lápis, massa de modelar, tesouras e tinta guache) registraram diferença acima de 100%.

A partir deste cenário, pode-se perceber que a pesquisa de preços antes da compra é benéfica e que deveria ser um hábito do cidadão. Segundo Motta (2011), estas variações de preço entre estabelecimentos realmente acontecem na grande maioria dos produtos, mas o mesmo comportamento não acontece com a maioria das pessoas, que por impulso, por pressa, por preguiça, acabam comprando na primeira loja em que entram.

Segundo uma análise realizada pelo SPC e CNDL (2017), a divisão, a reciclagem, e reutilização de produtos passam a ser alternativas ao ato de acumular e consumir de forma excessiva. Com isso, o consumo colaborativo surge como uma poderosa força econômica e cultural capaz de reinventar o que consumimos e principalmente a forma de como consumimos. Na chamada economia compartilhada ou consumo colaborativo, ao invés de ter, sendo relevante aproveitar dos benefícios de produtos compartilhados pelos demais consumidores durante tempo que for necessário.

Martins e Martins (2015) indicam que hoje existem alguns aplicativos móveis que auxiliam os usuários em relação à pesquisa de preços e produtos. Eles normalmente permitem o armazenamento contínuo dos preços e ofertas dos produtos levantados pelos próprios usuários da aplicação, possibilitam a manutenção destas informações, garantindo a integridade e veracidade destes dados, promovem benefícios para o custo-benefício dos consumidores e também um possível aumento da credibilidade dos fornecedores.

A partir do levantamento realizado por Napo (2015, p. 128-129), foi identificado que as principais motivações para a concretização da compra são a conveniência, a variedade e

disponibilidade de informação e, que 31% dos usuários afirmam usar aplicativos diariamente para realizar buscas de ofertas.

Neste contexto, este trabalho apresenta o desenvolvimento de um aplicativo móvel colaborativo para auxiliar o usuário a escolher e realizar compras considerando o custo-benefício dos produtos.

## 1.1 OBJETIVOS

Este trabalho tem o objetivo de desenvolver um aplicativo colaborativo para apoiar o usuário na escolha de qual supermercado tem o melhor custo/benefício em relação aos seus desejos de compra.

Os objetivos específicos são:

a) utilização de uma base de dados na nuvem, que sincronizará de forma constante as ofertas, promoções e demais informações compartilhadas entre os usuários;
b) disponibilizar um mecanismo que valide e garanta a integridade dos preços cadastrados pelos usuários para que eles sejam utilizados como fonte de dados na aplicação.

## 1.2 ESTRUTURA

O presente trabalho é estruturado em quatro capítulos: introdução, fundamentação teórica, desenvolvimento e conclusões. O segundo capítulo disserta a respeito dos principais assuntos relacionados a este trabalho se embasando especificamente na motivação teórica para o desenvolvimento do trabalho. O terceiro capítulo apresenta os diagramas, modelagem dos dados a arquitetura da aplicação. Também são utilizados trechos de código algumas telas do aplicativo. Por fim, o quarto capítulo expõe as conclusões obtidas no trabalho a partir da análise dos resultados e sugestões de futuras extensões deste.

# 2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo descreve brevemente os assuntos que fundamentam o contexto do trabalho realizado, sendo úteis para melhor compreensão do que foi feito. A seção 2.1 demonstra conceitos sobre o comportamento de consumidor e o relacionamento entre as partes do consumo. A seção 2.2 descreve os principais conceitos de colaboração no âmbito social aplicado aos benefícios do consumo compartilhado. Por fim, na seção 2.3 são descritos os trabalhos correlatos.

## 2.1 COMPORTAMENTO DE CONSUMO

Miniard, Engel e Blackwell (2000), definem o comportamento de consumidor como aquele que compreende as atividades diretamente envolvidas na obtenção, consumo e descarte de produtos e serviços, incluindo o processo de decisão que precede e segue estas ações. Há uma relação, então, com um processo que envolve desde a compra de um produto ou serviço até o seu descarte (o consumidor se desfazer do produto), passando pelo consumo. Como segunda definição, Sheth, Mittal e Newman (1999, tradução nossa), definem o comportamento de consumido como atividades mentais, físicas e sociais realizadas pelos consumidores, que resultam em decisões de ações de pagar, comprar, e usar produtos/serviços, assim como descartá-los, conforme a situação.

Segundo Larentis (2012, p. 15), as atividades mentais exercidas pelo consumidor são aquelas relacionadas ao processamento de informações e de tomada de decisão do produto, sendo de caráter predominantemente psicológico. Entre elas, avaliar uma marca, inferir qualidade a partir de um anúncio, escolher um produto dentre diversas alternativas, avaliar experiências de consumo e decidir a melhor forma de descarte. Já as atividades físicas, são aquelas envolvidas nos deslocamentos, esforços físicos e o gasto de energia do consumidor relacionados à compra, consumo e descarte, sendo utilizados como exemplos o deslocamento até o ponto o ponto de venda, translado na loja, pagamento, armazenamento do produto, utilização e descarte. E, por fim, as sociais, se referem a interações entre as pessoas nos processos de compra, consumo e descarte, sendo de caráter predominante sociológico e antropológico do consumo. Como exemplo, o autor cita atividades sociais como visitar lojas, pagamentos, reclamações no atendimento e solicitação de informações ao vendedor.

Solomon (2016, p. 9-10) resume as principais variáveis demográficas para o comportamento de consumo como a idade (pessoas de mesma faixa etária tendem a compartilhar um conjunto de valores e experiências culturais comuns que mantêm ao longo da vida.), o gênero (produtos diferenciados por gênero ocorrem desde muito cedo com fraldas,

roupas e perfumes.), a estrutura familiar (jovens solteiros, e recém-casados são os que mais se exercitam, vão a shows, cinemas e consumem bebidas alcoólicas.), a classe social e renda (pessoas de mesma classe social são aproximadamente iguais em termos de renda, posição social, vestuário, arte e lazer. Além disso, elas tendem a se socializar e compartilhar ideias e valores de como devem viver a vida.), a raça e etnicidade (como a sociedade atual está cada vez mais multicultural, novas oportunidades surgem para oferecer produtos especializados a grupos étnicos e apresentar ofertas a esses grupos.), a geografia (muitas empresas nacionais adaptam suas ofertas para atrair consumidores que vivem em diferentes partes do país.) e o estilo de vida (o modo como nos sentimos a respeito de nós mesmos, as coisas que valorizamos e o que gostamos de fazer no tempo livre.).

Segundo Oliveira e Moreira Neto (2016, p. 39), a popularização da internet trouxe um novo modo de relacionamento online e de consumo. As mídias sociais são mais do que meios de comunicação, são espaços onde as pessoas podem expor e postar informações ou arquivos sem que exista um relacionamento propriamente direto entre elas. Solomon (2016, p. 25) relata que a internet é a espinha dorsal da sociedade atual e o amplo acesso a aparelhos como computadores e smartphones garante que os consumidores de quase qualquer idade e qualquer parte do mundo possam criar e compartilhar conteúdo. Ele ainda cita que estas informações não fluem simplesmente de grandes empresas ou dos governos para pessoas, mas também entre pessoas em si devido a possibilidade de comunicação com uma imensa quantidade de pessoas instantaneamente. O autor exemplifica baseando-se no vídeo do experimento de Coca-Cola e Mentos viralizado na internet que o maior fenômeno de marketing dessa década possivelmente seja o conteúdo gerado pelo consumidor, onde pessoas comuns expressam opiniões sobre produtos, marcas e empresas em blogs, podcasts e rede sociais. Reforçando assim a tendência de comportamento da era Web 2.0.

## 2.2 COLABORAÇÃO SOCIAL APLICADO AO CONSUMO COMPARTILHADO

Mercado (2006, p. 28) define que há uma clara distinção entre cooperação e colaboração. Para ele, o conceito da cooperação é uma visão de um ambiente distribuído onde cada membro de um grupo é responsável por parte da solução de uma tarefa, formando uma solução unificada ao término desta tarefa. O autor também define o conceito de colaboração como um ambiente em que o esforço mútuo é privilegiado, dividindo tarefas da qual cada um é responsável pela sua parte. Porém, cada membro deste grupo consegue visualizar e participar de forma ativa nas ações das tarefas do seu parceiro, onde há um compromisso global, responsabilizando todos para a resolução de forma conjunta de um problema maior.

Para Möhlmann (2015, tradução nossa), as ações destes indivíduos são baseadas em raciocínio moral, da qual buscam a maximização da utilidade e economia de custos. Diante disso, é natural e lógico que estes indivíduos procurem formas de consumir e colaborar uns com os outros.

Segundo Freitas, Petrini, Lisilene (2013, p. 2), o consumo através da colaboração está vinculado a um contexto social de movimento sustentável que os mais variados níveis sistemáticos da sociedade atual integram ou emergem como parte de uma economia compartilhada. A economia compartilhada surge como um manifesto ao consumo excessivo, elaborando novas práticas que quebram paradigmas. Freitas, Petrini e Lisilene (2016, p. 5) acrescentam que através desta recente importância que vem sendo atribuída ao paradigma do consumo excessivo, que está frequentemente exaltado em programas sustentáveis, possibilita que o consumo colaborativo ganhe mais espaço em diversas áreas. Ele é percebido além de uma alternativa sustentável também como uma opção alternativa e promissora aos negócios e economia.

Freitas (2016) explica que o consumo colaborativo trata-se de um modelo de prestação de serviços compartilhados de pessoa-para-pessoa, que possibilita ter acesso a bens e serviços sem que seja necessária a compra ou troca monetária entre as partes envolvidas. Compartilhar, emprestar, alugar e trocar substituem o verbo comprar no consumo colaborativo. Belk (2014, p. 1597, tradução nossa) caracteriza o consumo colaborativo como um ambiente que as próprias pessoas gerenciam a aquisição e distribuição de recursos através de troca ou alguma compensação. Estas compensações ocorrem na forma de sistemas organizados ou redes em que os próprios participantes conduzem o compartilhamento de atividades. Esta essência geralmente é localizada nas tradições das formas de partilha, por exemplo em um contexto familiar e de habituais atividades de câmbio no mercado.

Imprensa Mercado & Consumo (2017) analisa uma projeção a respeito de uma pesquisa realizada em todas as capitais do Brasil pelo Serviço de Proteção ao Crédito (SPC Brasil) e pela Confederação Nacional de Dirigentes Lojistas (CNDL) a respeito da modalidade de consumo colaborativo mais conhecidas pelos usuários. A pesquisa constata que 79% dos consumidores acreditam que a economia compartilhada torna a vida mais conveniente e 68% se imaginam no segmento colaborativo em menos de dois anos. Os entrevistados veem como os principais benefícios do serviço a contenção de um consumo em excesso e uma maior economia na renda. Entretanto, a maior preocupação dos usuários ainda é a falta de confiança nas pessoas envolvidas.

Laamanen, Wahlen e Campana (2015, tradução nossa) apresentam duas características importantes para o funcionamento do consumo colaborativo. A primeira delas é ser colaborativo, renunciando de forma parcial aos tradicionais mercados dos sistemas atuais e exercer ênfase na ação coletiva. A segunda é a coletividade, enquanto o consumo comum pode ser uma atividade solitária que conserva a privacidade de um indivíduo, o consumo colaborativo considera que atos coletivos promovem uma criação de conexão entre o individual-privado aos aspectos públicos-coletivos de consumo. Segundo Bardhi e Eckhardt (2012, p. 882), o consumo colaborativo é o consumo mantido através de um acesso, semelhante a uma partilha, que não envolve transferência de propriedade, mas sim ao senso de propriedade percebida. O consumo baseado em acesso pode ser adquirido através de uma associação a organizações onde vários produtos podem ser compartilhados, diferindo do tradicional processo de locação em virtude da possibilidade de mediação colaborativa através da internet e não necessariamente pelo mercado. O autor ainda diferencia diversas formas de acesso ao consumo e realiza um levantamento a respeito das dimensões específicas como o anonimato (acesso pode diferir no anonimato interpessoal de acordo com o uso do contexto público ou privado. Na utilização de carros e hotéis os consumidores possuem acesso exclusivo ao objeto de consumo, sem interação com outros consumidores. O acesso também pode ser naturalmente social, utilizado num contexto de consumo compartilhado como no caso de hospedagem em uma residência.), o envolvimento do consumidor (o nível da participação dos consumidores no consumo, podendo ser limitada como tradicionais serviços de aluguel de hotéis ou de maior envolvimento como no compartilhamento de bens de consumo.) e o tipo do objeto acessado (a natureza da qual o acesso ao consumo é realizado, podendo ela ser experimental como um quadro num museu ou funcional como uma bicicleta de uso compartilhado. O tipo de acesso pode também ser classificado como material ou imaterial, este no caso de músicas e arquivos digitas).

Belk (2014, p. 1597, tradução nossa) diferencia o consumo colaborativo em duas categorias distintas. O primeiro como o consumo colaborativo comum, exemplificado através de eventos que levam as pessoas a consumirem bens de serviço de forma conjunta através de atividades com uma ou mais pessoas. O segundo como consumo colaborativo transacional, que engloba ações como compartilhar, trocar, emprestar e presentear. Ao compartilhar, as partes envolvidas usufruem de benefícios inerentes ou dividem os custos do objeto compartilhado. Estes objetos compartilhados implicam na utilização de recursos de propriedade coletiva que podem ser físicos como uma casa ou carro, como objetos abstratos como conhecimento e relacionamentos.

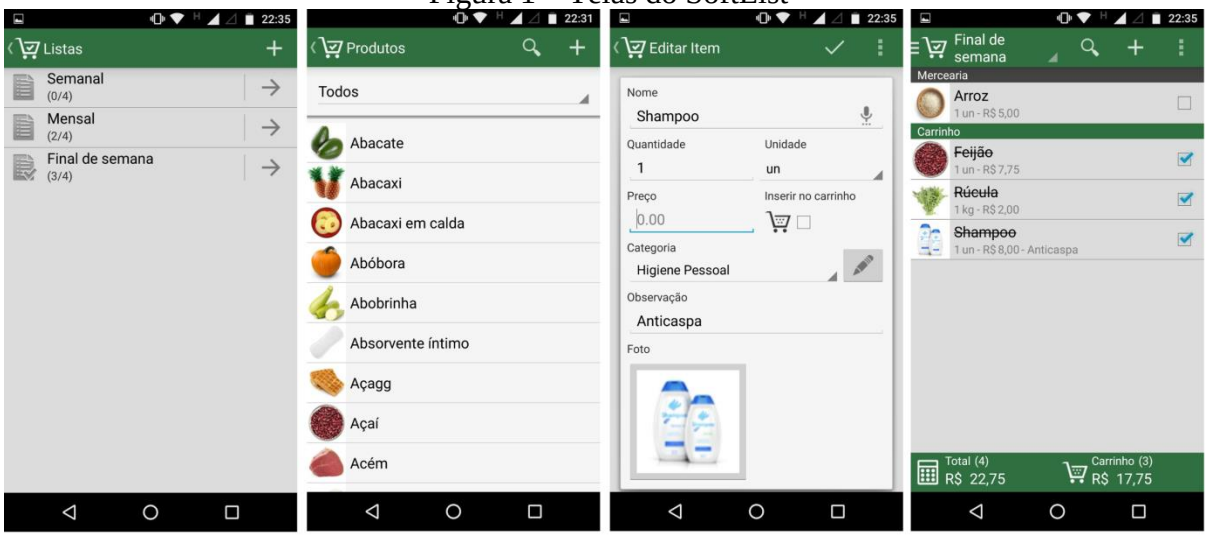
## 2.3 TRABALHOS CORRELATOS

Não foram encontradas aplicações que contemplassem todos os requisitos funcionais, tecnológicos ou conceituais propostos nesse trabalho. A seção 2.1 apresenta o aplicativo Lista de Compras - SoftList (MARTINS SOFTWARE, 2015). Suas principais características são o leitor de código barras, o compartilhamento de listas entre contas da aplicação e controle de preços dos produtos. Na seção 2.2 é descrito o aplicativo Shopping List - Buy Me a Pie (OOO KUPI BATON, 2012). Sua principal característica é a tecnologia de sincronização de listas em tempo real e dicionário de auto aprendizado. Por fim, a seção, apresenta o aplicativo Bring! Shopping List (BRING! LABS AG, 2012) que traz a ideia de notificar os usuários ao entrar em modo de compra e também é compatível com *smart watches*.

### 2.3.1 Listas de compras – Softlist

O Softlist (MARTINS SOFTWARE, 2015) é uma aplicação voltada essencialmente para a plataforma Android na qual possibilita controlar e gerenciar listas de compras de supermercados. Para o gerenciamento, o aplicativo dispõe da criação de diversas listas de compras. Quando selecionado uma lista de compra, é possível adicionar inúmeros produtos dentro dela. Estes produtos podem ser categorizados, quantificados e inseridos com algumas observações adicionais. Se o produto estiver cadastrado na base de dados, uma foto do produto é adicionada a este item juntamente com a categoria pré-cadastrada na base de dados do produto. Caso contrário, estas informações podem ser personalizadas pelo usuário em cada item. Outra forma de adicionar novos itens de compra é através da leitura do código de barras, utilizando auxílio do aplicativo externo Barcode Scanner (ZXING TEAM, 2008). Ainda na tela de listagem dos produtos é feita a totalização da compra de todos os produtos inseridos juntamente com a totalização dos itens adicionados no carrinho de compras. A Figura 1 demonstra algumas telas e fluxos do SoftList.

Figura 1 – Telas do SoftList




Fonte: Martins Software (2015).

O aplicativo Softlist também fornece algumas funcionalidades extras como a inserção de novas unidades de medida para quantidades unitárias, novas categorias para produtos, adicionar novos itens comandados por voz, compartilhamento das listas de compras por texto e também a extensão do aplicativo para os dispositivos *smart watches*.

Outro ponto a se destacar é que o SoftList (MARTINS SOFTWARE, 2015) requer a utilização de uma conta *premium* para funcionalidades de relatórios de histórico do produto, despesas e comparação de preços. Outro recurso restrito do aplicativo é a utilização de sincronização das listas com o servidor na nuvem e também o compartilhamento de suas listas com as contas de outros usuários *premium*.

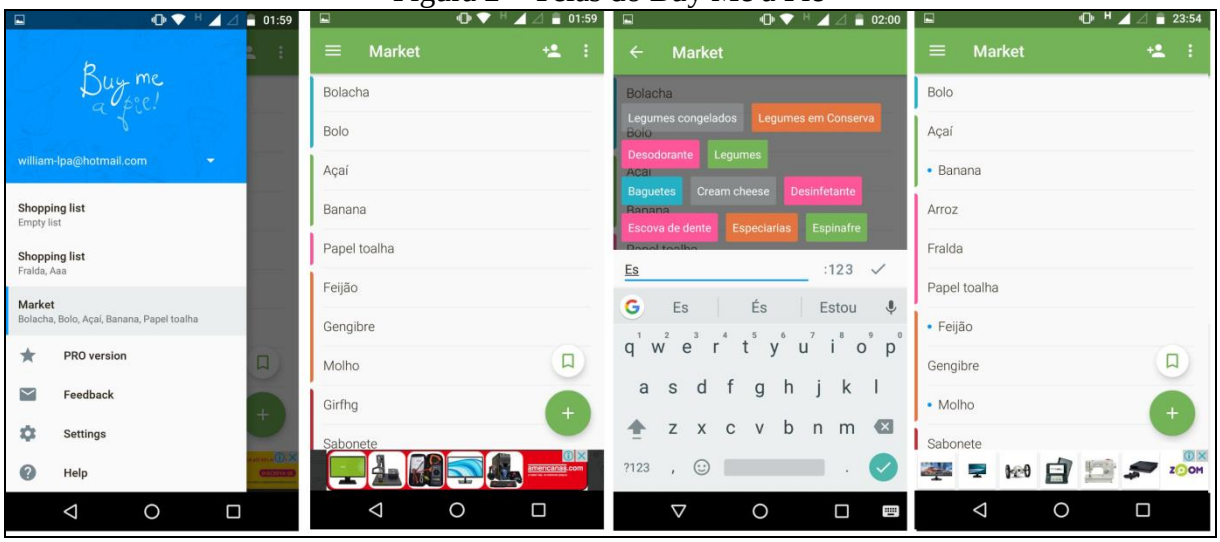
### 2.3.2 Shopping list – Buy Me A Pie

O Buy Me A Pie (OOO KUPI BATON, 2012), é uma aplicação móvel compatível nas plataformas IOS e Android, com suporte para web. A proposta do aplicativo é facilitar o gerenciamento de listas de compras de supermercados de seus usuários. Esta aplicação trabalha com a opção de criação e edição de multilistas de compras. Em cada uma destas listas é possível inserir produtos para sua compra. O Buy Me A Pie (OOO KUPI BATON, 2012) conta com um dicionário pré-configurado em sua base de dados, da qual realiza o agrupamento de produtos com base em sua categoria como mercearia, frutas, higiene, entre outros. Ao serem cadastrados, os produtos são exibidos e ordenados com base em sua categoria, permitindo uma melhor organização da lista de compras. As representações destas categorias são feitas na lista através de projeção de cores. Durante a inserção dos itens, o aplicativo conta com um sistema de autocompletar, que vai sugerindo produtos que se parecem com o conteúdo imputado pelo usuário. Caso o produto informado não conste no

dicionário, ele será adicionado automaticamente, juntamente com categoria definida. Isso permitirá a utilização do auto completar no novo produto apreendido. Outra característica do software é sua sincronização de produtos da lista em tempo real, podendo ser compartilhada por diversos usuários, tornando-a colaborativa. Quando uma alteração é realizada em uma lista colaborativa, os demais usuários são notificados com notificações *push* na tela do celular.

A Figura 2 demonstra algumas telas e fluxos do Buy Me a Pie. Nelas, é possível visualizar as listas de compras existentes, inserção de novos produtos na lista e produtos compartilhados por outro usuário.

Figura 2 – Telas do Buy Me a Pie




Fonte: OOO Kupi baton (2012).

O aplicativo Buy Me a Pie possui algumas funcionalidades diferenciadas para usuários *premium*. Atualmente, a aplicação restringe a criação máxima de três listas de compras e três usuários compartilhando a mesma lista. A função *premium* também traz a ausência de anúncios e novas cores para a categorização de produtos.

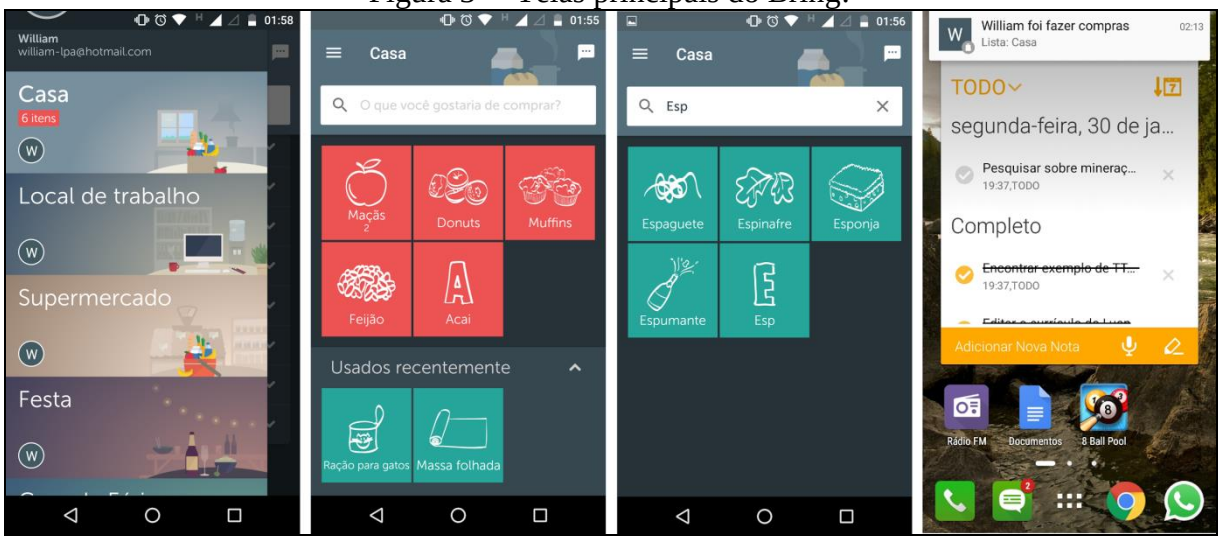
### 2.3.3 Bring! Shopping List

O Bring!, (BRING! LABS AG, 2012) foi desenvolvido para IOS e Android. A aplicação visa auxiliar o usuário no gerenciamento de suas compras de supermercado. Na tela inicial da aplicação, é permitido a criação de diversas listas com a possibilidade de direcioná-las a uma finalidade e também de renomeá-las. Dentro de uma lista de compras, a inserção de produtos é feita através de uma busca entre todos os itens cadastrados na base de dados. Caso o item não exista, ele será adicionado. Os itens possuem outras características para customização do usuário como a vinculação de imagens a produtos, campos de observação e realocação do produto no dicionário de dados. Bring! (BRING! LABS AG, 2012), permite o

compartilhamento de suas listas entre usuários que utilizam a mesma aplicação. A sincronização dos itens de compra compartilhados ocorre em tempo real. A qualquer momento o usuário pode determinar o estado atual da lista de compras, dentre as opções de “Vou às compras”, ”Lista Atualizada” e “Compras Feitas”.

A Figura 3 mostra algumas telas e as funcionalidades do sistema. Nelas, é possível visualizar as listas criadas pelo usuário, itens da lista de compra, inserção de novos produtos e a notificação enviada ao alterar o status para ida às compras.

Figura 3 – Telas principais do Bring!




Fonte: Bring! Labs AG (2012).

O aplicativo Bring! por sua vez, é um aplicativo totalmente gratuito nas plataformas Android e IOS. Ele ainda conta com recursos para personalização de ordenação de categorias, criação de novos templates e parametrizações de notificações *push*. Outra característica do aplicativo é o suporte a internacionalização de produtos. A composição e indexação do dicionário varia conforme idioma selecionado, podendo ser aplicado a nível de listas.

# 3 DESENVOLVIMENTO

Este capítulo demonstra as etapas de desenvolvimento deste trabalho. A seção 3.1 descreve os requisitos funcionais e não funcionais da aplicação. Na seção 3.2 são apresentados os diagramas que compõem a especificação da solução. Já na seção 3.3 encontra-se as etapas da implementação do aplicativo. Nela, encontram-se também imagens da aplicação, códigos e detalhes do desenvolvimento do aplicativo.

## 3.1 REQUISITOS

Nesta seção são demonstrados os Requisitos Funcionais (RF) e os Requisitos Não Funcionais (RNF) do aplicativo, no Quadro 1 e Quadro 2 respectivamente. Eles estão relacionados com os casos de uso apresentados na Figura 4.

Quadro 1 – Requisitos Funcionais

| **Requisitos funcionais** | **Casos de uso** |
|---|---|
| RF 01: permitir que o usuário possa cadastrar uma oferta. | UC03 |
| RF 02: permitir a criação de listas de compras. | UC08 |
| RF 03: permitir que o usuário informe uma lista de produtos para obter o melhor custo/benefício de compra a partir das ofertas compartilhadas. | UC09 |
| RF 04: permitir que o usuário acesse o aplicativo de forma prática e autenticada no servidor. | UC02 |
| RF 05: utilizar o número do telefone para garantir a segurança e autenticidade dos usuários e os dados compartilhados. | UC01 |
| RF 06: permitir que o usuário possa analisar e avaliar a confiabilidade das ofertas. | UC04 e UC05 |
| RF 07: permitir a criação de grupos de compartilhamento públicos e privados. | UC11 |
| RF 08: permitir que o usuário possa compartilhar ofertas com grupos da qual esteja participando ou diretamente com outro usuário. | UC04 e UC05 |
| RF 09: permitir ao usuário monitorar o preço das ofertas até o preço ideal. | UC06 |
| RF 10: permitir que o usuário possa receber notificações *push* quando novas ofertas forem compartilhadas. | UC04, UC05 e UC07 |
| RF 11: permitir que o usuário adicione participantes aos seus grupos de compartilhamento e a suas listas de compras. | UC10 |

Fonte: elaborado pelo autor.

Quadro 2 – Requisitos Não Funcionais

| **Requisitos funcionais** |
|---|
| RNF 01: garantir a sincronização das ofertas entre todos os usuários do aplicativo. |
| RNF 02: utilizar o banco de dados Azure SQL Server para persistência e sincronização dos dados. |
| RNF 03: permitir a personalização das listas de compras de forma rápida e intuitiva. |
| RNF 04: utilizar o *framework* da Microsoft, Xamarin para o desenvolvimento da aplicação móvel em Android. |
| RNF 05: o servidor da aplicação deve ser desenvolvido em Node.js. |
| RNF 06: enviar os dados ao servidor em intervalos de tempos pré-definidos para evitar requisições redundantes a aplicação. |
| RNF 07: retornar os dados solicitados em formato JavaScript Object Notation (JSON). |
| RNF 08: permitir a seleção da lista de contatos do próprio sistema operacional para localizar usuários. |
| RNF 09: utilizar gráficos para melhor visualização da variação de preço das ofertas. |

Fonte: elaborado pelo autor.

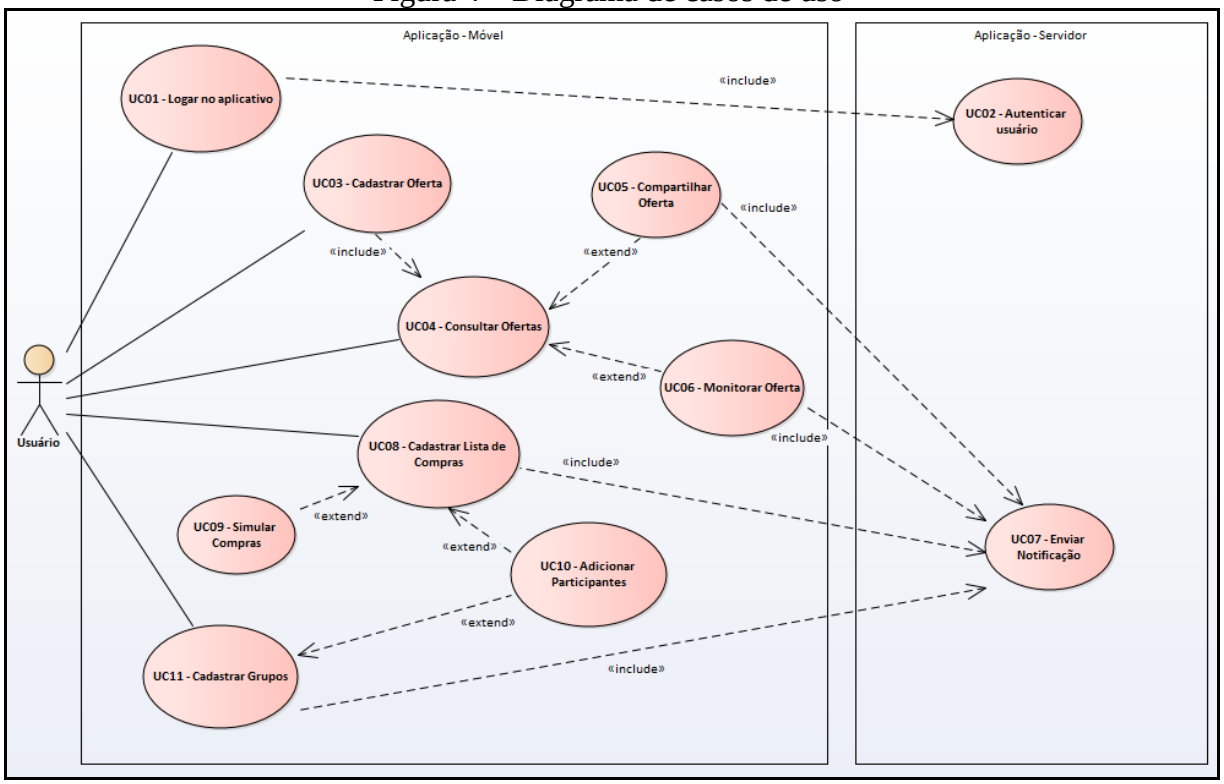
## 3.2 ESPECIFICAÇÃO

Na especificação deste trabalho foi utilizada a ferramenta Microsoft SQL Server Management Studio para a elaboração do Modelo de Entidade Relacionamento (MER). Para melhor compreensão do MER, o modelo relacional das entidades foi adaptado através da ferramenta Enterprise Architect para um diagrama de maior cunho visual. Na seção 3.2.1 encontra-se o diagrama de caso de usos do qual também foi desenvolvido através da ferramenta Enterprise Architect.

### 3.2.1 Diagrama de casos de uso

Nesta seção é apresentado o diagrama de casos de uso, ilustrado na Figura 4. Nele, estão representadas as funcionalidades na perspectiva do ator `Usuário`, sendo responsável por utilizar o dispositivo móvel e enviar dados ao servidor.

Figura 4 – Diagrama de casos de uso




Fonte: elaborado pelo autor.

Ao utilizar a aplicação pela primeira vez, o `Usuário` precisa autenticar-se, conforme caso de uso `UC01 – Logar no aplicativo`. A autenticação poderá ser realizada utilizando as credenciais do Facebook ou através da conta criada no próprio dispositivo. Escolhida a forma de autenticação, o servidor Azure realiza as chamadas e validações para que o usuário possa entrar e utilizar o aplicativo.

Ao logar, o `Usuário` poderá cadastrar uma nova oferta a partir do caso de uso `UC03 – Cadastrar Oferta`. As ofertas poderão ser consultadas a partir do caso de uso `UC04 – Consultar Ofertas`. O usuário poderá compartilhar as ofertas com determinados ou todos usuários da aplicação, utilizando o `UC05 – Compartilhar Oferta`. Para isso, ele poderá vincular-se a grupos de compartilhamento de ofertas através do `UC11 – Cadastrar Grupos`. Através do caso de uso `UC06 – Monitorar Oferta`, o `Usuário` poderá monitorar o preço da oferta e a qualquer alteração no preço ou status, ele será notificado a partir do caso de uso `UC07 – Enviar Notificação`. A partir do caso de uso `UC08 – Cadastrar Lista de Compra`, o `usuário` poderá elaborar sua lista de necessidades nos diversos estabelecimentos cadastrados. Tanto na elaboração das listas de compras como na criação de grupos é possível que o usuário adicione participantes para a realização do compartilhamento de informações, demonstrado no caso de uso `UC10 – Adicionar Participantes`. Para consultar o melhor custo-benefício de seus produtos cadastrados na lista de compras, ele poderá realizar a

simulação da compra utilizando o caso de uso `UC09 – Simular Compras`. É importante destacar que a sincronização dos dados com o servidor não acontece de forma constante. A persistência resultante dos casos de usos `UC05`, `UC08` e `UC11` são sempre salvos primeiramente em uma base SQLite local, sendo enviadas para o servidor após um período de tempo pré-definido, podendo gerar novas notificações.

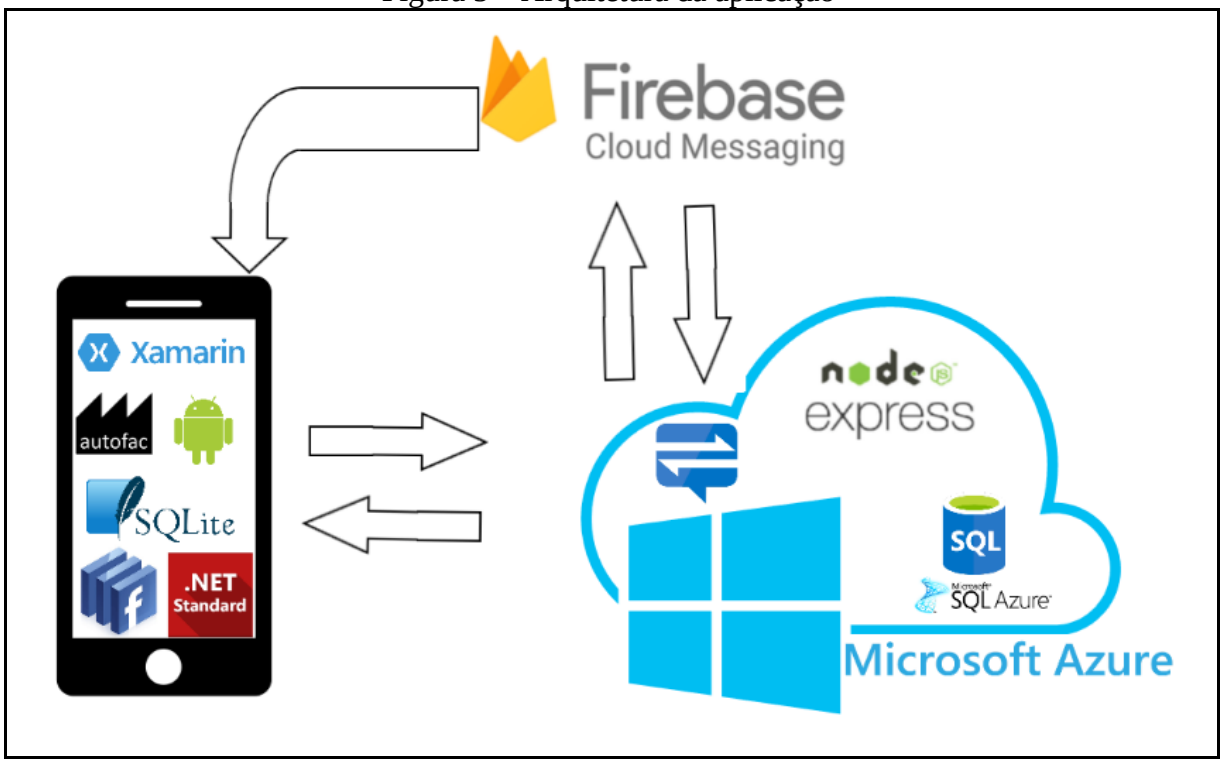
## 3.3 IMPLEMENTAÇÃO

A seguir são mostradas as técnicas e ferramentas utilizadas no desenvolvimento da aplicação. Para armazenamento dos registros foi construída uma base de dados representada por um diagrama adaptado do Modelo de Entidade de Relacionamento (MER), disponível no Apêndice A.

### 3.3.1 Técnicas e ferramentas utilizadas

Para o desenvolvimento do servidor foi utilizado o Microsoft Azure para hospedar, gerenciar dados e serviços da aplicação. O servidor foi desenvolvido em JavaScript, com a plataforma Node.js, versão 6.9.1, utilizando a Integrated Development Environment (IDE) Visual Studio Code e uma IDE web semelhante Visual Studio Code, acoplada dentro do próprio portal do Azure. O aplicativo móvel foi desenvolvido em C#, utilizando o *framework* da Microsoft Xamarin.Forms, versão 2.4.0.282 através da IDE Microsoft Visual Studio 2017. A Figura 5 representa a arquitetura utilizada na implementação do aplicativo e suas principais ferramentas.

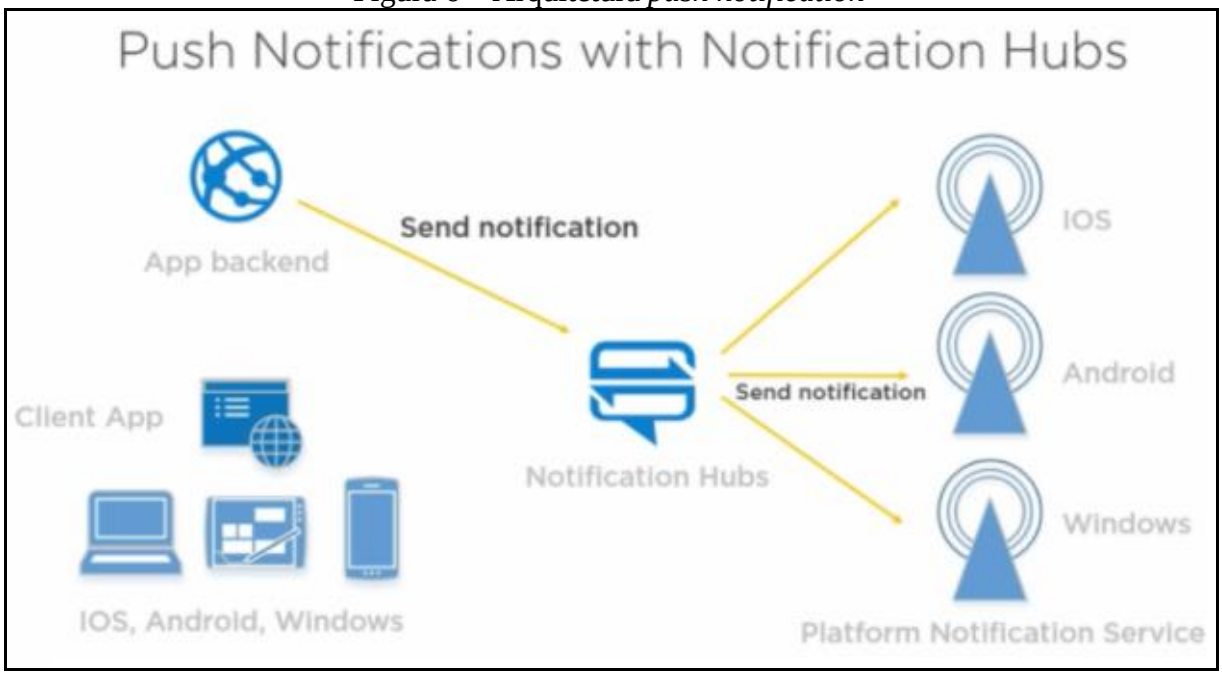
Figura 5 – Arquitetura da aplicação




Fonte: elaborado pelo autor.

Para a interceptação e manipulação de chamadas via Application Programming Interface (API) no servidor, foi utilizado o *framework* Express. A principal dependência do servidor é o pacote `azure-mobile-apps`, da qual é possível manipular dados de uma tabela Structured Query Language (SQL) com JavaScript, autenticar-se de acordo com o provedor especificado pelo aplicativo, utilizar o *hub* do Azure para enviar notificações em broadcast para plataformas específicas e demais serviços. A aplicação está hospedada em uma máquina virtual com infraestrutura de camada gratuita nos servidores do Azure. Os registros enviados pelos dispositivos móveis são armazenados em uma base de dados SQL Server, disponível em uma outra máquina virtual. O hub de notificação é outro serviço isolado da aplicação. Ele é o responsável por fazer o redirecionamento e abstração de toda a infraestrutura necessária para o envio de *push notification*, específica em cada plataforma. A Figura 6 representa a arquitetura conceitual desta infraestrutura utilizada pelo *hub*.

Figura 6 – Arquitetura *push notification*




Fonte: Vargis (2017).

No desenvolvimento do aplicativo móvel, foi utilizado a biblioteca `Microsoft.Azure.Mobile.Client`, responsável por realizar a conexão e utilização de serviços fornecidos pelo servidor no Azure. Outra biblioteca também fornecida pelo Azure é a `Microsoft.Azure.Mobile.Client.SQLiteStore`, que possibilita a utilização de uma base de dados SQLite no dispositivo local e posteriormente, a sincronização das informações com a base de dados do servidor SQL Server via chamadas a APIs de arquitetura *Representational State Transfer* (REST). Para auxiliar na abstração da utilização do padrão de projeto injeção de dependência, foi utilizado a biblioteca `AutoFac`, responsável por gerenciar o container de dependências e instanciar abstrações em classes dependentes. Para a integração com a rede social Facebook, foi utilizado a biblioteca `Xamarin.Facebook.Android`, desenvolvida pelo próprio Xamarin utilizando a SDK nativa do Facebook, apresentando uma *webview* para autenticação do usuário ou utilizando as credencias do próprio aplicativo do Facebook quando instalado previamente no dispositivo. O *plugin* `SkiaSharp`, foi utilizado para a exibição de gráficos em 2D. Como o aplicativo é desenvolvido em na tecnologia .NET, o projeto compartilhado foi desenvolvido utilizando o `.NET Standard 1.6`.. Este *framework* visa padronizar as APIs e bibliotecas de todas as tecnologias .NET, da qual todas compartilham e implementam a mesma interface.

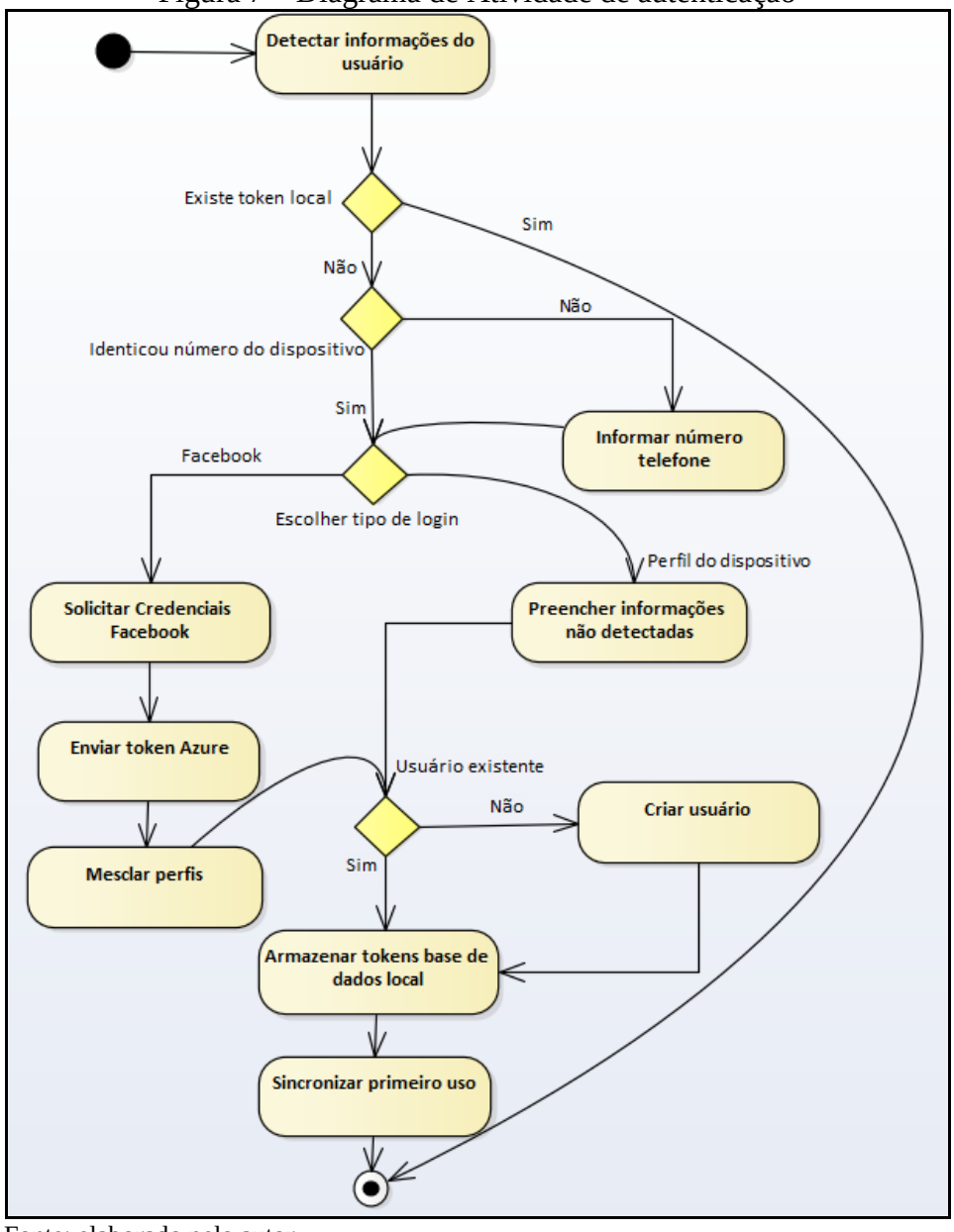
### 3.3.2 Etapas do desenvolvimento

Nesta seção encontram-se os trechos de códigos mais relevantes da aplicação juntamente com seus respectivos detalhamentos. A seção 3.3.2.1 aborda o processo de autenticação no desenvolvimento do aplicativo. Na seção 3.3.2.2 é relatado de forma detalhada o desenvolvimento do aplicativo móvel. Por fim, a seção 3.3.2.3 demonstra como é realizada a notificação de mensagens entre o aplicativo, o hub e plataforma designada.

#### 3.3.2.1 Autenticação e integridade

Para melhor compreensão do fluxo de autenticação, o diagrama da Figura 7 será utilizado para demonstrar os passos e fluxos necessários para utilizar o aplicativo.

Figura 7 – Diagrama de Atividade de autenticação




Fonte: elaborado pelo autor.

A primeira etapa do desenvolvimento foi a integração do aplicativo com a rede social Facebook com objetivo de garantir a identidade do `usuário` do aplicativo. É importante ressaltar que o Facebook não é pré-requisito obrigatório para utilizar a aplicação, portanto existe um fluxo alternativo utilizando as informações do próprio dispositivo como credenciais. Quando o aplicativo é inicializado, verifica-se se o usuário já realizou o *login* anteriormente para não solicitar as credenciais sempre que o aplicativo for utilizado, conforme pode ser visto no Quadro 3.

Quadro 3 – Login Automático ao iniciar o aplicativo

```
 1  public async void DoSSOLoginAsync()
 2   {
 3       LoginIn = true;
 4       Client.CurrentUser = RetrieveAzureTokenFromSecureStore("azure");
 5       if (Client.CurrentUser!= null &&
 6           !IsTokenExpired(Client.CurrentUser.MobileServiceAuthenticationToken))
 7        {
 8          await Client.SyncContext.InitializeAsync(Store, new MobileServiceSyncHandler());
 9          CurrentUser = await RetrieveUserFromSecureStoreAsync("facebook");
10        }
11       else
12        {
13          await Client.SyncContext.InitializeAsync(Store, new MobileServiceSyncHandler());
14          CurrentUser = await RetrieveUserFromSecureStoreAsync("localUser");
15        }
16       if (CurrentUser != null)
17        {
18          LoginUser(CurrentUser.User);
19          CreateOrRefreshPushRegistration();
20        }
21      LoginIn = false;
22   }
23  public async Task<LoginResultContent> RetrieveUserFromSecureStoreAsync(string key)
24   {
25      if (storeAccount == null)
26          storeAccount = AccountStore.Create();
27      var account = storeAccount.FindAccountsForService(key).FirstOrDefault();
28      string token = null;
29      if ((account?.Properties?.TryGetValue("token", out token)).GetValueOrDefault())
30      {
31         var user = await GetTable<User>().LookupAsync(account.Username);
32         return new LoginResultContent(user)
33         {
34           Token = token,
35           Result = Result.OK,
36           Message = "Retrieved from Account Store"
37         };
38      }
39      return null;
40  }
```

Fonte: elaborado pelo autor.

Na linha 4, tenta-se recuperar o *token*, armazenado no dispositivo local, que contém as informações do usuário autenticado previamente no Azure. Se o *token* for encontrado, ele é retornado como uma instância de *login* na linha 32. Então, um novo *token* é carregado para instanciar o perfil do usuário registrado no Facebook, conforme demonstrado na linha 9. Quando não existir um o *token* vinculado ao Azure, tenta-se recuperar o perfil do usuário registrado localmente. Quando nenhum for encontrado, a aplicação interpreta que é a primeira

vez que o usuário está efetuando o acesso ao aplicativo, conforme exemplificado na linha 21. Existirá a necessidade de autenticar-se na aplicação através de seu número telefônico. A Figura 8 ilustra a tela de acesso do aplicativo.

Figura 8 – Tela de *Login*


(a) (b)

Fonte: elaborado pelo autor.

O aplicativo tenta obter o número do aparelho automaticamente de duas formas distintas. Na Figura 8 item (a) é exibida a mensagem de confirmação quando a aplicação conseguiu acessar o número telefônico. Do contrário, será apresentada a mensagem conforme demonstrado na Figura 8 item (b). Um exemplo de como o número do dispositivo é obtido pode ser visto através do Quadro 4.

Quadro 4 – Obtenção número do telefone

```
 1  public string TryGetPhoneNumber
 2  {
 3   get
 4      {
 5        try {
 6          var manager =
 7          (TelephonyManager)Forms.Context.GetSystemService(Context.TelephonyService);
 8          string numberManager = manager.Line1Number;
 9          if (!string.IsNullOrEmpty(numberManager))
10              return numberManager;
11
12           string[] PROJECTION =
13           { ContactsContract.Contacts.InterfaceConsts.Id,
14             ContactsContract.CommonDataKinds.Phone.Number,
15             ContactsContract.Contacts.Data.InterfaceConsts.Mimetype
16           };
17          ContentResolver content = MainActivity.CurrentActivity.ContentResolver;
18          var cursor = content.Query(Android.Net.Uri.WithAppendedPath(
19                                      ContactsContract.Profile.ContentUri,
20                                      ContactsContract.Contacts.Data.ContentDirectory),
21                                      PROJECTION,
22                                  ContactsContract.Contacts.Data.InterfaceConsts.Mimetype + "= ?",
23                                      new string[] {
24                                  ContactsContract.CommonDataKinds.Phone.ContentItemType },null);
25
26         if (cursor.MoveToFirst())
27         {
28           do
29              {
30                 var index =
31                 cursor.GetColumnIndex(ContactsContract.Contacts.Data.InterfaceConsts.Mimetype);
32                 string mime_type = cursor.GetString(index);
33                  if (mime_type == (ContactsContract.CommonDataKinds.Phone.ContentItemType))
34                        return cursor.GetString(1);
35
36               } while (cursor.MoveToNext());
37
38         }
39        } catch (Exception err){
40             Log.Instance.AddLog(err);
41           }
42     return "";
43     }
44
45  }
```

Fonte: elaborado pelo autor.

Na linha 7 é utilizado a classe `TelephonyManager` do Android que possibilita o acesso e o gerenciamento a serviços de telefonia do aparelho. Na linha 8 é feito a tentativa de obter o número do dispositivo gravado no chip do aparelho. Como é uma informação gravada no cartão Subscriber Identity Module (SIM) do celular, o acesso a este registro depende da disponibilização da operadora e do SIM do telefone. Se o número não for obtido desta forma, o aplicativo tenta localizar a informação realizando uma consulta no perfil da lista de contato do usuário atual. Depois destas verificações, o `Usuário` deverá optar por uma das formas de autenticação. Quando utilizado a autenticação via perfil do dispositivo, uma tela de cadastro adicional será exibida ao qual o `Usuário` deverá preencher para prosseguir. A Figura 9 ilustra a tela de cadastro e as informações obtidas de forma automatizada.

Figura 9 – Detecção de informações do dispositivo no cadastro do usuário


Fonte: elaborado pelo autor.

Quando o usuário optar pelo acesso via Facebook, a biblioteca nativa da rede social se encarregará de construir a infraestrutura necessária para a obtenção das credenciais, sendo realizada através do aplicativo instalado no dispositivo ou através de um *webview* padrão, conforme demonstrado na Figura 10 (a) e (b) respectivamente.

Figura 10 – Autenticação com o Facebook


(a) (b)

Fonte: elaborado pelo autor.

Ao autenticar-se com a rede social, uma função call-back é invocada para registrar um usuário na aplicação, conforme pode ser visto no Quadro 5.

Quadro 5 – Criando um usuário a partir do Facebook

```
 1  public void OnSuccess(Java.Lang.Object result)
 2  {
 3    var login = result as LoginResult;
 4    var token = login.AccessToken;
 5    var request = GraphRequest.NewMeRequest(token, null);
 6    Bundle parameters = new Bundle();
 7    parameters.PutString("fields", "id,name,gender,locale,birthday, location,email,
 8                                    width(180),installed");
 9    request.Parameters = parameters;
10    var userInfo = Task.Run(() => (request.ExecuteAndWait().JSONObject.ToString())).Result;
11    var definition = new { Id = 0L, Name = "", Gender = "", Locale = "", Birthday = "",
12                           Location = new { Name = "" }, Email = "",
13                           Picture = new { Data = new { Url = "" } } };
14    var userConverted = JsonConvert.DeserializeAnonymousType(userInfo, definition);
15    var user = new User()
16              {
17                  FacebookId = token.UserId,
18                  FullName = userConverted?.Name,
19                  Birthday = userConverted?.Birthday != null ? DateTime.ParseExact(
20                             userConverted.Birthday, "MM/dd/yyyy", CultureInfo.InvariantCulture)
21                             : default(DateTime),
22                  Email = userConverted?.Email,
23                  Male = userConverted?.Gender == "male",
24                  Avatar = userConverted?.Picture?.Data?.Url,
25                  Locale = userConverted?.Locale,
26                  Location = userConverted?.Location?.Name
27              };
28    using (var scope = App.Container.BeginLifetimeScope())
29            scope.Resolve<IAuthentication>().LoginResult =
30             new LoginResultContent(user) { Token = token.Token };
31  }
```

Fonte: elaborado pelo autor.

Na linha 4, é armazenado o *token* de acesso retornado pelo objeto de autenticação do Facebook. Na linha 7 é construído uma cadeia de caracteres que será enviado a API do Facebook para obter as informações do usuário registrado. Como retorno tem-se um objeto anônimo, utilizando a biblioteca `Newtonsoft.Json` para desserialização dos registros retornados.

Após a realização do processo de autenticação no aplicativo, o *token* gerado pelo Facebook é enviado ao Azure para que o servidor tenha conhecimento da identidade do usuário, tempo de expiração do *token* e permissões dentro do aplicativo. No final deste processo é feito uma mescla dos dados obtidos via Facebook e do aparelho. Em ambos os fluxos é verificado a necessidade de criar um novo registro no banco de dados quando o número ainda não foi previamente cadastrado para ou como um usuário. No menu principal da aplicação, será apresentado um alerta para o usuário informando que os dados serão sincronizados, conforme demonstrado na Figura 11.

Figura 11 – Primeiro acesso aplicativo


Fonte: Elaborado pelo autor.

Após a confirmação da mensagem, será exibida uma tela de carregamento ao qual o processo de sincronização será inicializado, conforme pode ser visto no Quadro 6.

Quadro 6 – Sincronização primeiro acesso

```
 1  private async Task SyncronizeData(DateTime? date = null)
 2  {
 3     await Task.WhenAll(new Task[]
 4     {
 5      Task.Run(() => OfertaRepository.SyncDataBaseAsync(date)),
 6      Task.Run(() => CarteiraProdutoRepository.SyncDataBaseAsync(date)),
 7      Task.Run(() => ProdutoRepository.SyncDataBaseAsync(date)),
 8      Task.Run(() => TipoRepository.SyncDataBaseAsync(date)),
 9      Task.Run(() => MarcaRepository.SyncDataBaseAsync(date)),
10      Task.Run(() => EstabelecimentoRepository.SyncDataBaseAsync(date)),
11      Task.Run(() => UnidadeMedidaRepository.SyncDataBaseAsync(date)),
12      Task.Run(() => GrupoOfertaRepository.SyncDataBaseAsync(date)),
13      Task.Run(() => HistoricoOfertaRepository.SyncDataBaseAsync(date)),
14      Task.Run(() => ParticipanteGrupoRepository.SyncDataBaseAsync(date)),
15      Task.Run(() => UserRepository.SyncDataBaseAsync(date)),
16      Task.Run(() => CategoriaRepository.SyncDataBaseAsync(date)),
17      Task.Run(() => MonitoramentoOfertaRepository.SyncDataBaseAsync(date)),
18      Task.Run(() => ListaCompraRepository.SyncDataBaseAsync(date)),
19      Task.Run(() => ParticipanteListaCompraRepository.SyncDataBaseAsync(date)),
20      Task.Run(() => ParticipanteListaCompraRepository.SyncDataBaseAsync(date)),
21      Task.Run(() => ProdutoListaCompraRepository.SyncDataBaseAsync(date)),
22      });
23  }
```

Fonte: elaborado pelo autor.

Na linha 3, o fluxo de execução é suspenso, retornando para a instrução que a invocou anteriormente até que todas as *Tasks*, que são classes auxiliares para execução de código multi-processado e assíncrono, concluam a sincronização dos dados.

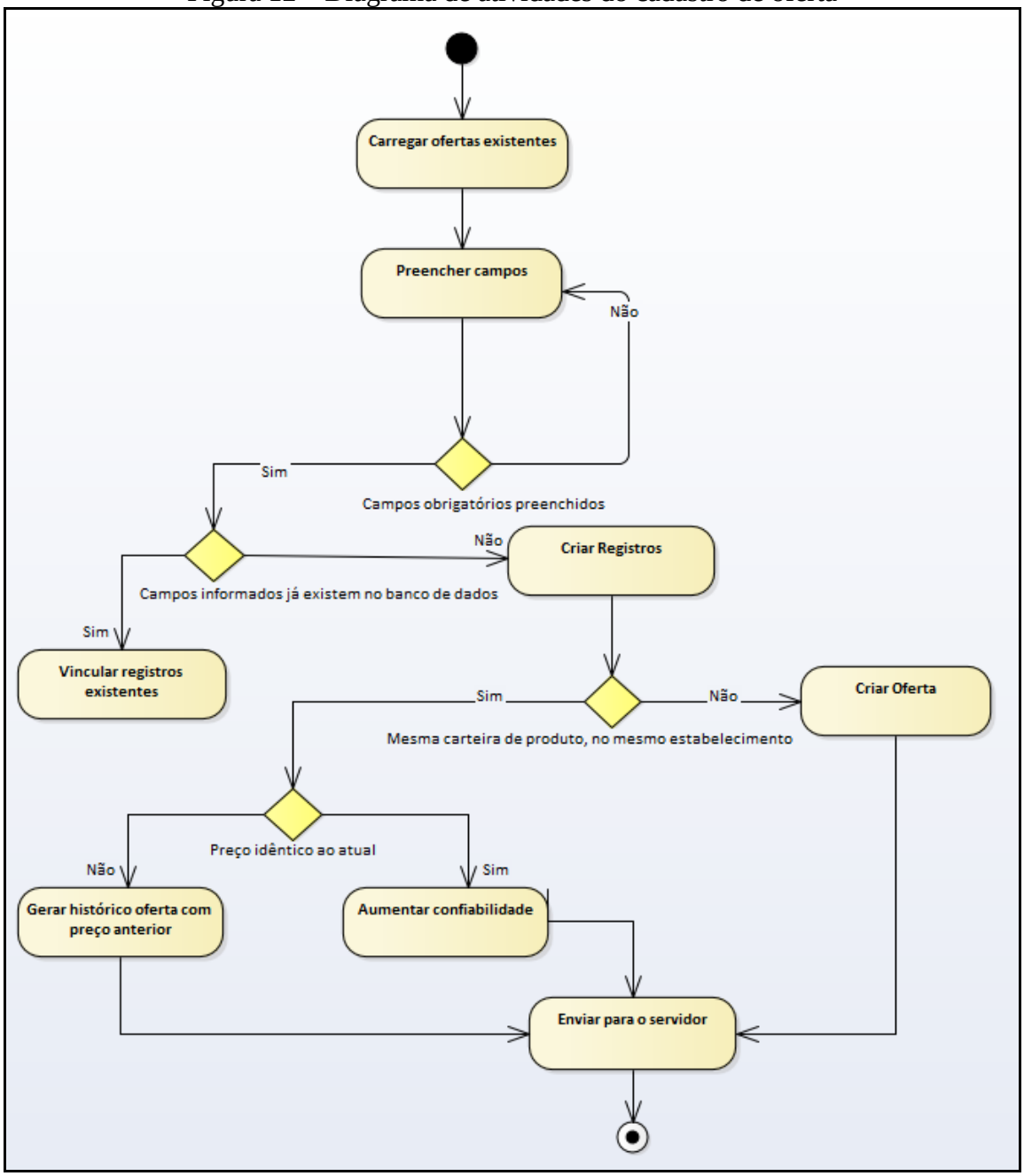
#### 3.3.2.2 Aplicação Móvel

Na aplicação móvel estão as funcionalidades acessíveis para o usuário. Ela possui diversos fluxos de utilização que serão descritos nas próximas seções.

##### 3.3.2.2.1 Cadastrar oferta

A oferta é a principal entrada de dado na aplicação. Ela é utilizada como fonte de informação em diversos cenários. Para que o cadastro de uma oferta seja realizado, ele deve seguir os passos apresentados no diagrama de atividades da Figura 12;

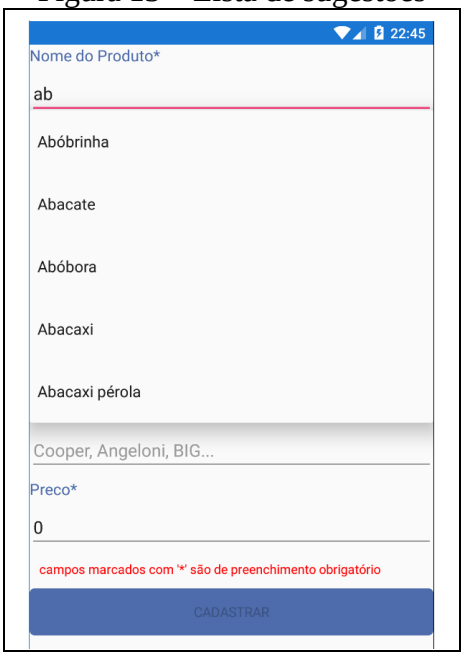
Figura 12 – Diagrama de atividades do cadastro de oferta




Fonte: elaborado pelo autor.

Quando o `Usuário` cadastra uma oferta na aplicação, a primeira ação realizada pelo aplicativo é carregar uma lista de sugestões com as informações já cadastradas anteriormente. Esta abordagem foi utilizada para melhorar a usabilidade da aplicação, permitindo mais agilidade e assertividade para o usuário. A Figura 13 ilustra esta lista de sugestões para o nome do produto de uma oferta.

Figura 13 – Lista de sugestões




Fonte: elaborado pelo autor.

O carregamento destas listas é realizado em *background* de forma assíncrona no momento do carregamento da tela de registro da oferta. No Quadro 7 é possível analisar os detalhes do código desenvolvido.

Quadro 7 – Carregamento das listas de sugestões

```
1  public static event Action<string, List<string>> OnCollectionLoaded;
2
3  internal async Task LoadAutoCompleteAsync()
4  {
5     var Produtos = new List<string>((await produtoRepository.GetEntitiesAsync())
6                         .Select(x => x.Nome).Distinct());
7     OnCollectionLoaded(nameof(Produtos), Produtos);
8     var Tipos = new List<string>((await tipoRepository.GetEntitiesAsync())
9                     .Select(x => x.Nome).Distinct());
10    OnCollectionLoaded(nameof(Tipos), Tipos);
11    var UnidadesMedidas = new List<string>((await unidadeMedidaRepository.GetEntitiesAsync())
12                              .Select(x => x.Nome).Distinct());
13    OnCollectionLoaded(nameof(UnidadesMedidas), UnidadesMedidas);
14    var Marcas = new List<string>((await marcaRepository.GetEntitiesAsync())
15                      .Select(x => x.Nome).Distinct());
16    OnCollectionLoaded(nameof(Marcas), Marcas);
17    var Categorias = new List<string>((await categoriaRepository.GetEntitiesAsync())
18                         .Select(x => x.Nome).Distinct());
19    OnCollectionLoaded(nameof(Categorias), Categorias);
20    var Estabelecimentos = new List<string>((await estabelecimentoRepository.GetEntitiesAsync()
21                                              ).Select(x => x.Nome).Distinct());
22    OnCollectionLoaded(nameof(Estabelecimentos), Estabelecimentos);
23 }
```

Fonte: elaborado pelo autor.

Na linha 1 é definido um evento que notificará os observadores registrados. Para disparar o evento, é necessário enviar dois parâmetros. O primeiro é o identificador da coleção a ser exibido na lista de sugestões e o segundo são os dados a serem carregados. O observador da lista receberá estes eventos é uma função da classe `AutoCompleteTextView`, nativa do Android para manipulação deste tipo de componente. O `Usuário` deverá preencher todos os campos obrigatórios para poder prosseguir. Conforme ilustrado na Figura 13, o botão ficará desabilitado até que a oferta esteja consistente para ser enviada ao servidor. Quando a oferta estiver íntegra, inicia-se o processo de persistência de dados no dispositivo local, conforme pode ser observado no Quadro 8;

Quadro 8 – Criação registros dependentes de uma oferta

```
 1  public async void CriarNovaOfertaAsync(Oferta oferta)
 2  {
 3     var idEstabelecimento =  await CreateOrRetrieveEntityByNameAsync(oferta?.Estabelecimento);
 4     var idTipo =
 5                 await CreateOrRetrieveEntityByNameAsync(oferta?.CarteiraProduto?.Produto?.Tipo);
 6     var idUnidadeMedida = await CreateOrRetrieveEntityByNameAsync(oferta?.CarteiraProduto?
 7                                                                   .Produto?.UnidadeMedida);
 8     var idCategoria = await CreateOrRetrieveEntityByNameAsync(oferta?.CarteiraProduto?
 9                                                               .Produto?.Categoria);
10     var idMarca = await CreateOrRetrieveEntityByNameAsync(oferta?.CarteiraProduto?.Marca) ??
11                   (await marcaRepository.SyncTableModel.Where(x =>
12                    x.Nome.ToLower() == "sem marca").ToListAsync()).First().Id;
13     var idProduto = await CriarProdutoAsync(oferta.CarteiraProduto.Produto.Nome,
14                                             idUnidadeMedida, idTipo, idCategoria,
15                                             oferta.CarteiraProduto.Produto.QuantidadeMensuravel);
16     var idCarteira = await CriarCarteiraProdutoAsync(idProduto, idMarca);
17     await CriarOfertaAsync(idCarteira, idEstabelecimento, oferta.PrecoAtual);
18     await SyncronizeBaseDeOfertasAsync();
19  }
20
21  public async Task<string> CreateOrRetrieveEntityByNameAsync<TEntity>(TEntity entity)
22                                           where TEntity : class, IEntity, IName, new()
23  {
24    var repository = await GetEntityService<TEntity>();
25    if (string.IsNullOrEmpty(entity?.Nome)) return null;
26
27    return ((await repository?.SyncTableModel
28             .Where(x => x.Nome.ToLower() == entity.Nome.ToLower())
29             .Select(x => x.Id).ToEnumerableAsync()).FirstOrDefault()) ??
30             await repository?.CreateEntityAsync(new TEntity() { Nome = entity.Nome });
31  }
```

Fonte: elaborado pelo autor.

Antes de inserir uma oferta, é necessário persistir os registros dependentes, ou seja, o relacionamento de chaves estrangeiras. Na linha 21 está uma função que tem a responsabilidade de verificar se o registro dependente já existe, retornando a chave primária correspondente ou senão, retornando a nova chave. Na linha 18, todas as tabelas envolvidas na inserção serão enviadas ao servidor no Azure para que a oferta seja armazenada no banco de dados. É importante destacar que a oferta ainda possui algumas regras e fluxos adicionais a serem seguidas durante sua criação, sendo demonstradas no Quadro 9.

Quadro 9 – Criando uma oferta

```
 1  private async Task CriarOfertaAsync(string idCarteira, string idEstabelecimento,decimal preco)
 2  {
 3    if (idCarteira == null || preco < 1) return;
 4
 5    if (idEstabelecimento != null)
 6    {
 7       var ofertaExistente = (await ofertaRepository.SyncTableModel
 8                              .Where(x => x.IdCarteiraProduto == idCarteira
 9                              && x.IdEstabelecimento == idEstabelecimento)
10                              .OrderByDescending(x => x.UpdatedAt)
11                              .Select(x => x).ToEnumerableAsync())
12                              .FirstOrDefault();
13       if (ofertaExistente != null)
14       {
15          if (ofertaExistente.PrecoAtual == preco)
16          {
17            await ApplyLikeAsync(ofertaExistente.Id);
18          }
19          else
20          {
21            var historico = CriarHistoricoAPartirDeOfertaAsync(ofertaExistente, preco);
22          }
23       }
24        else
25             await ofertaRepository?
26                   .CreateEntityAsync(new Oferta(idEstabelecimento, idCarteira, preco));
27    }
28  }
```

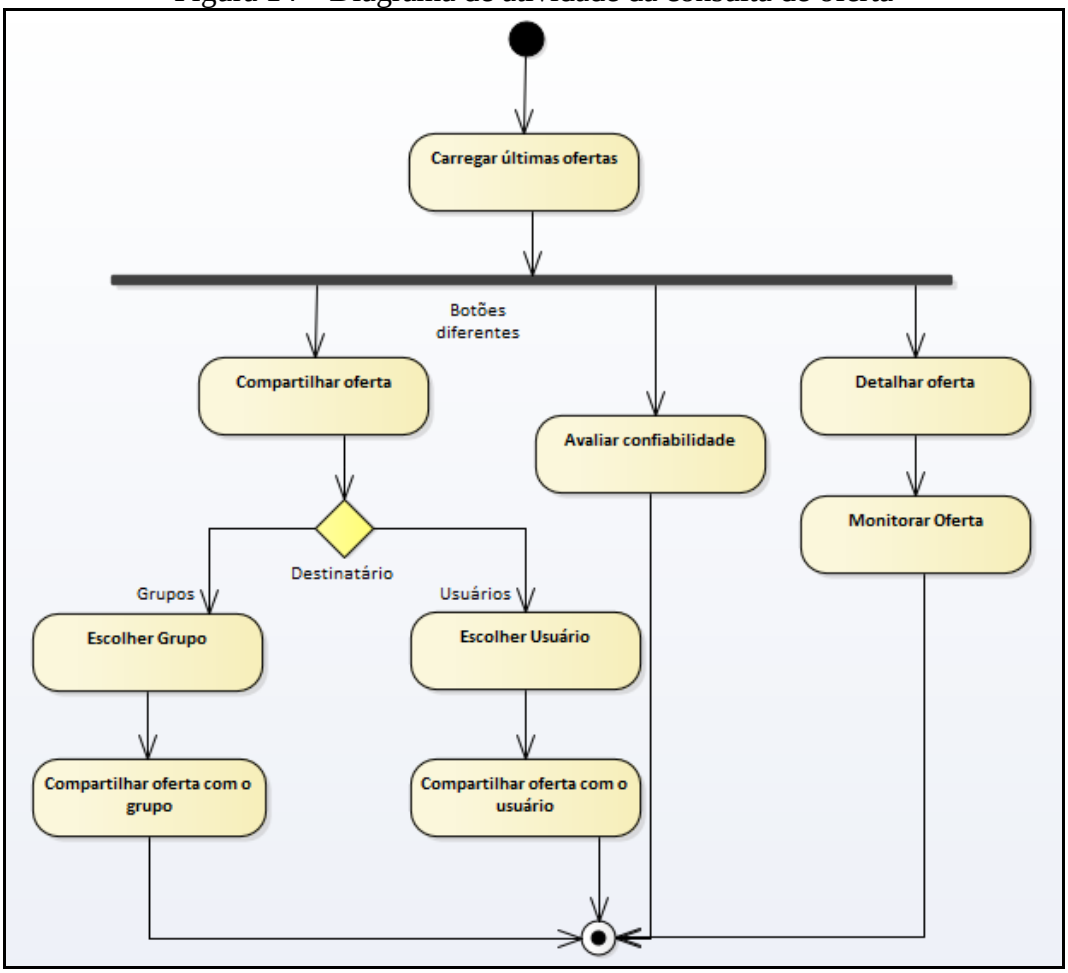
Fonte: elaborado pelo autor.

Na linha 7 é realizada uma consulta na base de dados SQL do dispositivo para verificar se a oferta a ser inserida já existe na base de dados com o mesmo produto, marca e estabelecimento. Se o registro for encontrado, é interpretado que a oferta já existe no estabelecimento correspondente. Se o preço da oferta atual coincidir com a oferta existente, o aplicativo entende que a oferta é confiável, aumentando sua confiabilidade. Se o preço for divergente, é provável que o preço do produto tenha sido alterado no estabelecimento, portanto o registro da oferta existente é transferido para uma tabela de histórico, conforme demonstrado na linha 21. Se a oferta não for localizada, entende-se que é uma nova oferta. Dessa forma, cadastrando-a conforme indicado na linha 25.

#### 3.3.2.2.2 Consultar oferta

Toda oferta cadastrada será exibida na tela de consultas. Nesta tela também estão disponíveis outras funcionalidades, conforme representado no diagrama de atividades da Figura 14.

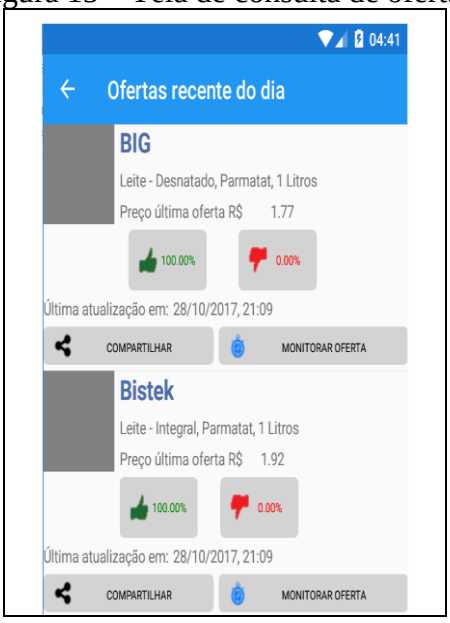
Figura 14 – Diagrama de atividade da consulta de oferta




Fonte: elaborado pelo autor.

Ao consultar as ofertas, o `Usuário` visualizará as últimas ofertas informadas compartilhadas pelos usuários, conforme ilustrado na Figura 15.

Figura 15 – Tela de consulta de ofertas




Fonte: elaborado pelo autor.

Inicialmente é realizado o carregamento das últimas ofertas registradas no aplicativo. Cada item da lista de exibição corresponde a uma oferta previamente cadastrada, exibindo apenas as informações mais relevantes. Para cada oferta, o `Usuário` pode avaliar a confiabilidade da oferta, fornecendo aos demais usuários uma forma de analisar e disseminar a veracidade da oferta informada. O código responsável por esta funcionalidade pode ser visto no Quadro 10.

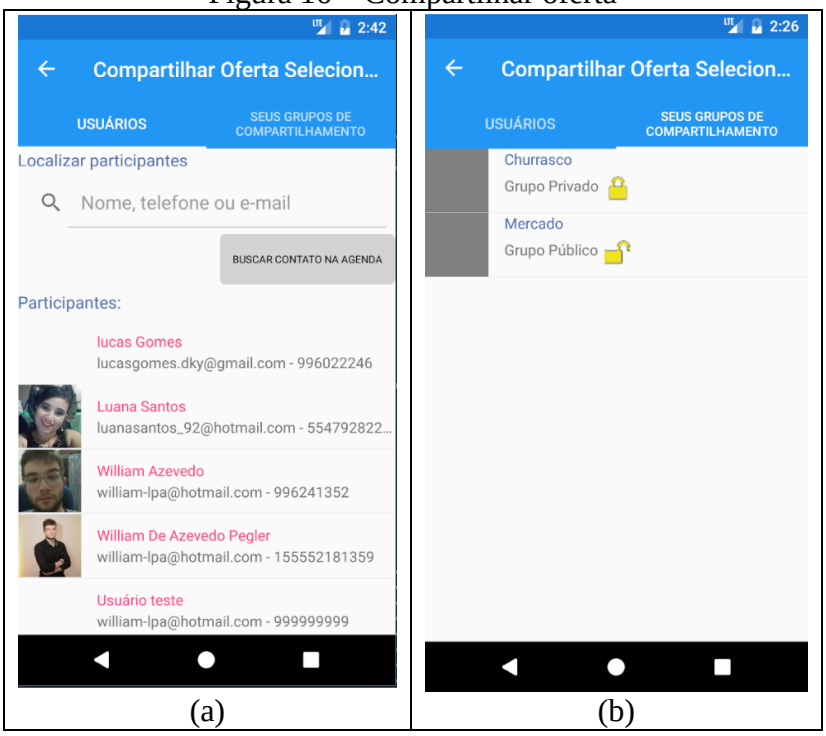
Quadro 10 – Confiabilidade de ofertas

```
 1  internal async Task<float> ApplyLikeAsync(string idOFerta)
 2  {
 3    var oferta = await ofertaRepository.GetByIdAsync(idOFerta);
 4    oferta.Avaliacoes++;
 5    oferta.Likes++;
 6    await ofertaRepository.UpdateEntityAsync(oferta);
 7    return CalculateConfiabilidade(oferta);
 8  }
 9
10  internal async Task<float> ApplyDislikeAsync(string idOFerta)
11  {
12    var oferta = await ofertaRepository.GetByIdAsync(idOFerta);
13    oferta.Avaliacoes++;
14    await ofertaRepository.UpdateEntityAsync(oferta);
15    return CalculateConfiabilidade(oferta);
16  }
17
18  internal float CalculateConfiabilidade(Oferta oferta)
19  {
20     if (oferta.Likes == 0)
21        return 0;
22      return (oferta.Likes * BASE_PERCENTUAL) / oferta.Avaliacoes;
23  }
24
25  internal async Task<float> RevertConfiabilidadeAsync(string idOFerta, bool revertLike)
26  {
27    var oferta = await ofertaRepository.GetByIdAsync(idOFerta);
28    oferta.Avaliacoes--;
29    if (revertLike)
30    oferta.Likes--;
31    await ofertaRepository.UpdateEntityAsync(oferta);
32    return CalculateConfiabilidade(oferta);
33  }
```

Fonte: elaborado pelo autor.

Na linha 1 é especificado a função que realiza a avaliação positiva da oferta. Toda oferta é criada de forma positiva. Nas avaliações positivas, o número de avaliações e o número de aprovações são aumentados conforme ilustrado nas linhas 4 e 5 respectivamente. Na avaliação negativa, definida na linha 10, apenas o número de avaliações é aumentado, reduzindo o percentual de aprovação por serem inversamente proporcionais. O cálculo da confiabilidade é realizado na linha 18. Sempre que o usuário *logado* reavaliar uma oferta será feito a reversão da ação anterior a fim de garantir unicidade da avaliação. Ainda na tela de consulta é possível realizar o compartilhamento da oferta e suas informações. A qualquer momento o `Usuário` poderá enviar a oferta que está sendo visualizada para um de seus grupos ou para algum usuário especifico do aplicativo, conforme mostra a Figura 16.

Figura 16 – Compartilhar oferta




Fonte: elaborado pelo autor.

Na Figura 16 item (a) é possível escolher os usuários que serão notificados sobre a existência da oferta. No item (b), o usuário poderá escolher os grupos. Os dados serão enviados para o servidor ao qual será responsável por disparar as notificação, conforme pode ser visto no Quadro 11.

Quadro 11 – Enviando oferta compartilhada a um grupo de ofertas

```
1  private async void ExecuteEditarGroupoOfertasAsync(GrupoOferta grupoOferta)
2  {
3     if (NotSharing)
4     {
5       var parameters = new Dictionary<string, string>();
6       parameters.Add("ID", grupoOferta.Id);
7       await PushAsync<NovoGrupoOfertaPageViewModel>(false, parameters);
8     }
9     else
10    {
11       if (await MessageDisplayer.Instance
12           .ShowAskAsync("Compartilhar Oferta", $"Você tem certeza que deseja compartilhar a
13                          oferta selecionada com o grupo {grupoOferta.Name} ?", "Sim",
14                          "Não")
15       {
16          var body = new CompartilhamentoOfertaGrupo()
17          {
18            Title = $"Nova oferta Compartilhada no grupo {grupoOferta.Name}",
19            Description = $"{azureService.CurrentUser.User.FullName} compartilhou uma nova
20                             oferta, clique para mais detalhes",
21            IdGrupo = grupoOferta.Id,
22            IdOferta = CompartilharOfertasPageViewModel.IdSharingOferta,
23          };
24          await azureService.Client.InvokeApiAsync<CompartilhamentoOfertaGrupo,
25          CompartilhamentoOfertaGrupo>("compartilharOfertaGrupo", body);
26          await PopAsync<OfertasPageViewModel>();
27       }
28    }
   }
```

Fonte: elaborado pelo autor.

O trecho de código da linha 11 é responsável por exibir ao usuário uma *modal* de confirmação de compartilhamento. Quando a mensagem for confirmada, será enviado ao servidor o identificador da oferta a ser compartilhada e o identificador do grupo, que serão utilizados para obter a relação de participantes. Estas informações são encapsuladas dentro de um objeto que será enviado ao servidor através de uma requisição HTTP. Na tela de consulta também há a possibilidade de o usuário acompanhar o histórico de variação de preços, assim como, monitorar determinada oferta até um preço ideal. Esta funcionalidade está disponível dentro do detalhamento da oferta, conforme mostra a Figura 17.

Figura 17 – Monitorar Oferta


Fonte: elaborado pelo autor.

Quando um novo monitoramento de oferta for criado, o usuário receberá um *push notification* com a confirmação do alerta criado. Se o alerta for atualizado ou removido, uma notificação de confirmação também será enviada. Quando uma oferta é inserida no servidor e o preço corresponder ao que se deseja monitorar, uma notificação será enviada, conforme pode ser visto no Quadro 12.

Quadro 12 – Monitorando ofertas inseridas

```
 1  table.update(function (context) {
 2      logger.info('Running Oferta.update');
 3
 4      var query = {
 5          sql: `select distinct MonitoramentoOferta.IdUser from MonitoramentoOferta
 6                    WHERE MonitoramentoOferta.IdOferta =@IdOferta
 7                    and MonitoramentoOferta.PrecoAlvo >= @preco`,
 8          parameters: [
 9              { name: 'IdOferta', value: context.item.id },
10              { name: 'preco', value: context.item.precoAtual },
11          ],
12      };
13
14      context.data.execute(query).then(function (result) {
15          if (context.push) {
16              sendNotification(context.item.id, result);
17          }
18      });
19      return context.execute()
20          .then(function (results) {
21          return results;
22      }).catch(function (error) {
23          logger.error('Error while 2 running context.execute: ', error);
24      });
25  });
```

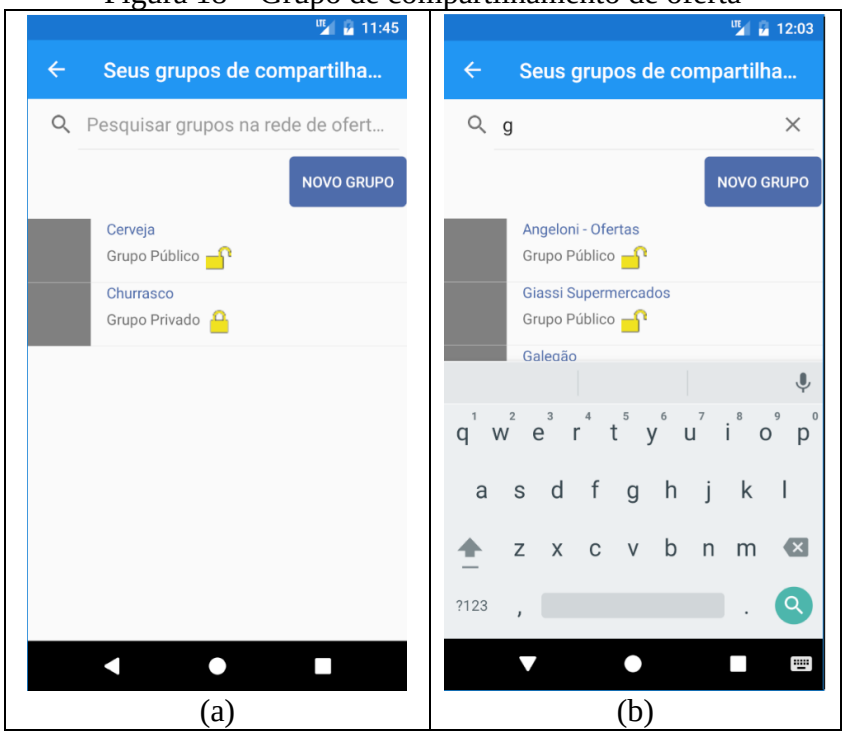
Fonte: elaborado pelo autor.

Na linha 1 encontra-se a função, da biblioteca `express`, responsável pela atualização da tabela de oferta. O controle de monitoração é feito apenas na atualização da oferta pois o `Usuário` irá observar somente alterações na oferta já existente. Durante a atualização, é realizada uma consulta no banco de dados para obter todos os potenciais usuários que estariam monitorando a oferta. Todos os usuários identificados receberão a notificação indicando que o preço desejado foi atingido, conforme realizado na linha 16.

#### 3.3.2.2.3 Cadastrar grupos

Na aplicação, o conceito de grupos serve para que usuários com interesses em comum possam compartilhar ofertas entre si. Existe um controle de privacidade nos grupos, podendo ser público ou privado. A Figura 18 ilustra esta diferenciação de privacidade.

Figura 18 – Grupo de compartilhamento de oferta




Fonte: elaborado pelo autor.

Na Figura 18 item (a) tem-se a listagem dos grupos ao qual o usuário participa. Quando a tela de grupos é carregada, realiza-se uma consulta na base de dados para obter os grupos aos qual o usuário está vinculado, conforme implementado no Quadro 13.

Quadro 13 – Grupos que o usuário logado está participando

```
1  public async Task<IEnumerable<GrupoOferta>> CarregarGrupoDeOfertasUsuarioLogadoAsync()
2  {
3     Dictionary<string, GrupoOferta> grupos =
4                                        (await grupoRepository.GetEntitiesAsync())
5                                        .ToDictionary(key => key.Id, value => value);
6     var retorno = (await participantesRepository.GetEntitiesAsync())
7                   .Where(x => x.IdUser == azureService.CurrentUser.User.Id)
8                   .Select(x => grupos[x.IdGrupoOferta]).OrderBy(x => x.Name);
9     return retorno.ToArray();
10 }
11
12 public async Task<IEnumerable<GrupoOferta>>
13 CarregarGrupoDeOfertasUsuarioLogadoSync(IEnumerable<string> localResult = null)
14 {
15   var date = azureService.CurrentUser.User.UpdatedAt;
16   await participantesRepository.SyncDataBaseAsync(date);
17   await grupoRepository.SyncDataBaseAsync(date);
18   var newResult = await CarregarGrupoDeOfertasUsuarioLogadoAsync();
19   if (newResult.Count() > localResult.Count())
20   {
21      return newResult.Where(x => !localResult.Contains(x.Id)).ToList();
22   }
23   return null;
24 }
```

Fonte: elaborado pelo autor.

Inicialmente são carregados os grupos, sendo armazenados em uma estrutura de árvore. Na linha 6 são obtidos os grupos do usuário através do relacionamento com a tabela de participantes, que retorna o nome dos grupos em ordem alfabética. Os grupos de compartilhamento tendem a estarem em constantes alterações com a inclusão de novos

participantes. Por isso é importante mantê-los sempre sincronizados com o banco de dados existente no servidor. Quando os grupos da base local são retornados na tela, é chamada uma função em *background* que realiza a sincronização com o servidor, conforme declaração da linha 12. A partir da linha 15 é realizado uma requisição, ao servidor, para que os dados sejam atualizados. A Figura 18 item (b) ilustra uma pesquisa por grupos públicos. O usuário poderá participar de qualquer grupo público desde que seja de seu interesse e, ao qual, receberá notificações de ofertas. A qualquer momento um participante também poderá sair do grupo. Estas ações são ilustradas no Quadro 14.

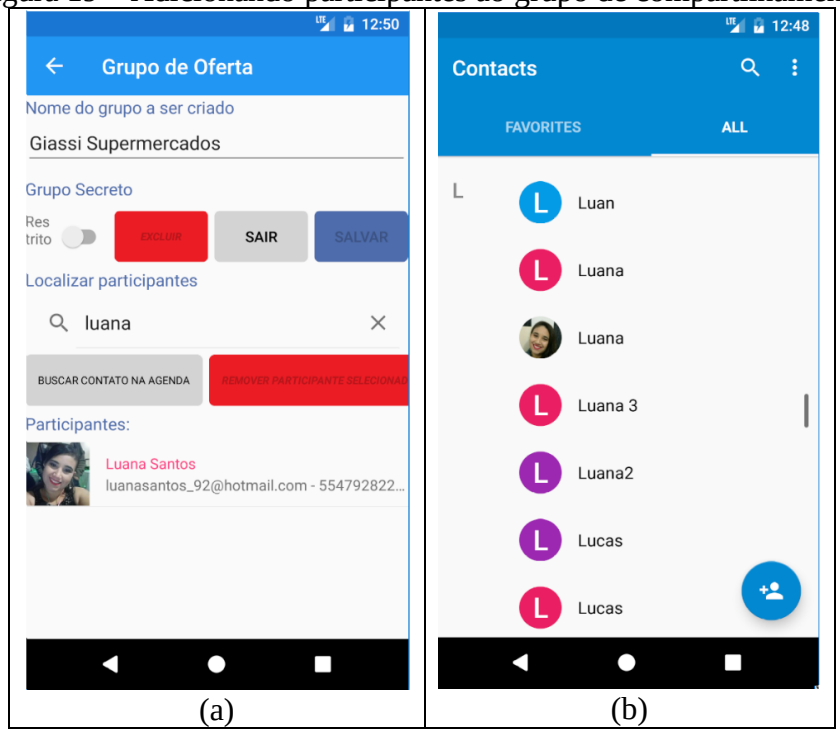
Quadro 14 – Permanência no grupo de ofertas

```
 1  private async void ExecutePermanenciaGrupoAsync()
 2  {
 3     if (UsuarioLogadoPertenceAoGrupo)
 4     {
 5        if (await MessageDisplayer.Instance.ShowAskAsync("Deixar o grupo de oferta",
 6                                                         $"Você tem certeza que deseja sair do
 7                                                          grupo {editGrupoOferta.Name} ?",
 8                                                          "Sim", "Não"))
 9         {
10             var membro = Members.First(x => x.IdUser == azureService.CurrentUser.User.Id);
11             Members.Remove(membro);
12             await grupoOfertaService.ExcluirParticipanteGrupoOfertaAsync(membro);
13             await PopAsync<GruposOfertasPageViewModel>();
14         }
15     }
 6     else
17     {
18        if (await MessageDisplayer.Instance.ShowAskAsync("Participar do grupo de oferta",
19                                                        $"Você tem certeza que deseja participar
20                                                         do grupo {editGrupoOferta.Name} ?",
21                                                         "Sim", "Não"))
22        {
23          var user = azureService.CurrentUser.User;
24          AdicionarParticipante(new ParticipanteGrupo(user.Id) { User = user });
25          await grupoOfertaService.ParticiparGrupoAsync(editGrupoOferta, Members.Last());
26          AtualizarStatus();
27        }
28     }
29  }
```

Fonte: elaborado pelo autor.

Através da linha 5 o aplicativo apresenta uma mensagem para que o usuário confirme a sua intenção de sair do grupo. Quando a mensagem for confirmada, o usuário é removido da lista de participantes daquele grupo. O fluxo da linha 18 ocorrerá quando um participante solicitar a participação em um grupo público da qual ele ainda não está participando. Quando a mensagem for confirmada, o usuário é adicionado na lista de participantes do grupo. A partir da linha 25, os dados do participante são armazenados localmente e no servidor. Tanto em grupos privados como em públicos é possível adicionar vários participantes. O participante poderá ser adicionado através de uma busca realizada no aplicativo ou utilizando a lista de contatos do Android, conforme demonstrado na Figura 19.

Figura 19 – Adicionando participantes ao grupo de compartilhamento

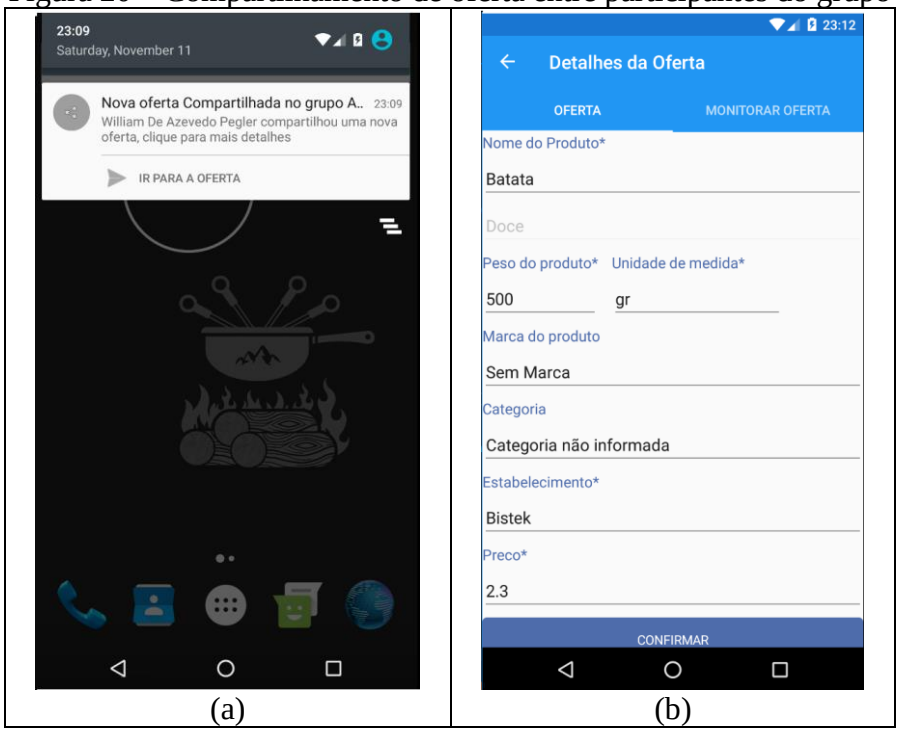



(a) (b)

Fonte: elaborado pelo autor.

A Figura 19 (a) ilustra como o usuário pode adicionar um participante. Este processo pode ser realizado por nome ou e-mails ou telefones dos usuários já existentes. A Figura 19 item (b) demonstra a lista de contatos do próprio Android que pode ser utilizada para encontrar possíveis participantes. Somente contatos que já utilizam a aplicação podem ser adicionados a um grupo. Caso um participante seja selecionado na lista de contatos e não esteja utilizando o aplicativo, uma mensagem será apresentada alertando sobre esta inconsistência. Quaisquer alterações realizadas na estrutura do grupo serão enviadas notificações a todos os integrantes através de *push notification*. O principal objetivo do grupo de ofertas é garantir que todos os membros recebam uma oferta quando ela for compartilhada, conforme mostra a Figura 20.

Figura 20 – Compartilhamento de oferta entre participantes do grupo




(a) (b)

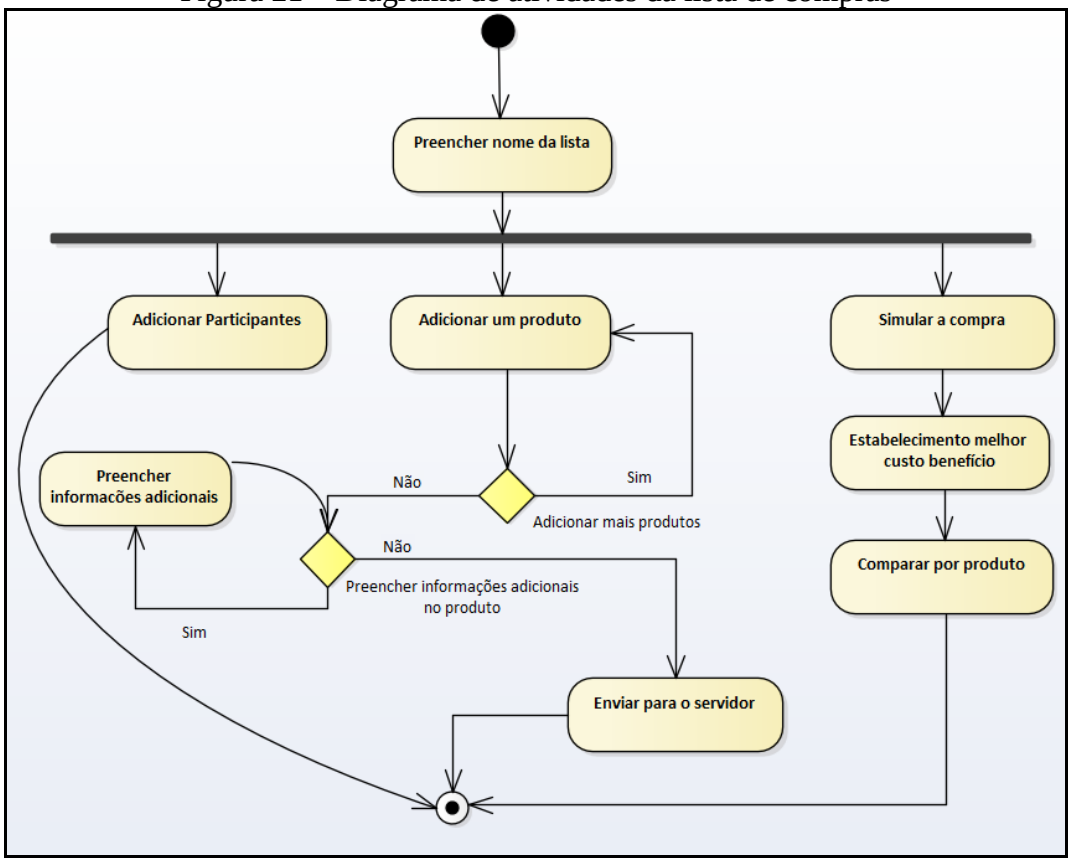
Fonte: elaborado pelo autor.

Na Figura 20 item (a) pode-se observar a mensagem recebida do *hub* de notificações. A Figura 20 item (b) apresenta os detalhes da oferta que foi compartilhada com o usuário. Esta ação de redirecionamento é exercida no aplicativo quando o usuário clica na notificação.

#### 3.3.2.2.4 Cadastrar lista de compras

A lista de compras permite ao usuário elaborar uma lista de produtos da qual necessita descobrir o estabelecimento com o melhor custo-benefício. Os produtos da lista de compras utilizam as ofertas cadastradas na aplicação como fonte de dados para decisão de preços e estabelecimentos. Para isso, segue-se os passos apresentados no diagrama de atividades da Figura 21.

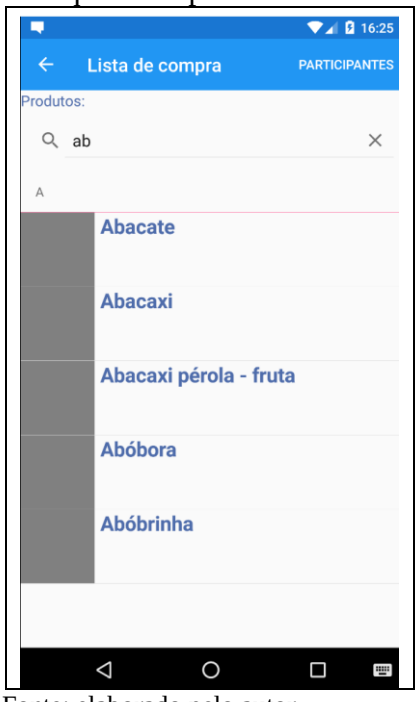
Figura 21 – Diagrama de atividades da lista de compras




Fonte: elaborado pelo autor.

Inicialmente, o `Usuário` deverá indicar o nome de identificação da lista de compras. A lista de produtos é agrupada alfabeticamente para facilitar a seleção do usuário, conforme ilustrado na Figura 22.

Figura 22 – Pesquisando produtos na lista de compras




Fonte: elaborado pelo autor.

Na Figura 22 estão os produtos correspondentes a uma busca feita pelo usuário. Os produtos estarão sempre agrupados e ordenados em ordem alfabética. O algoritmo utilizado para a busca dos produtos pode ser visto no Quadro 15.

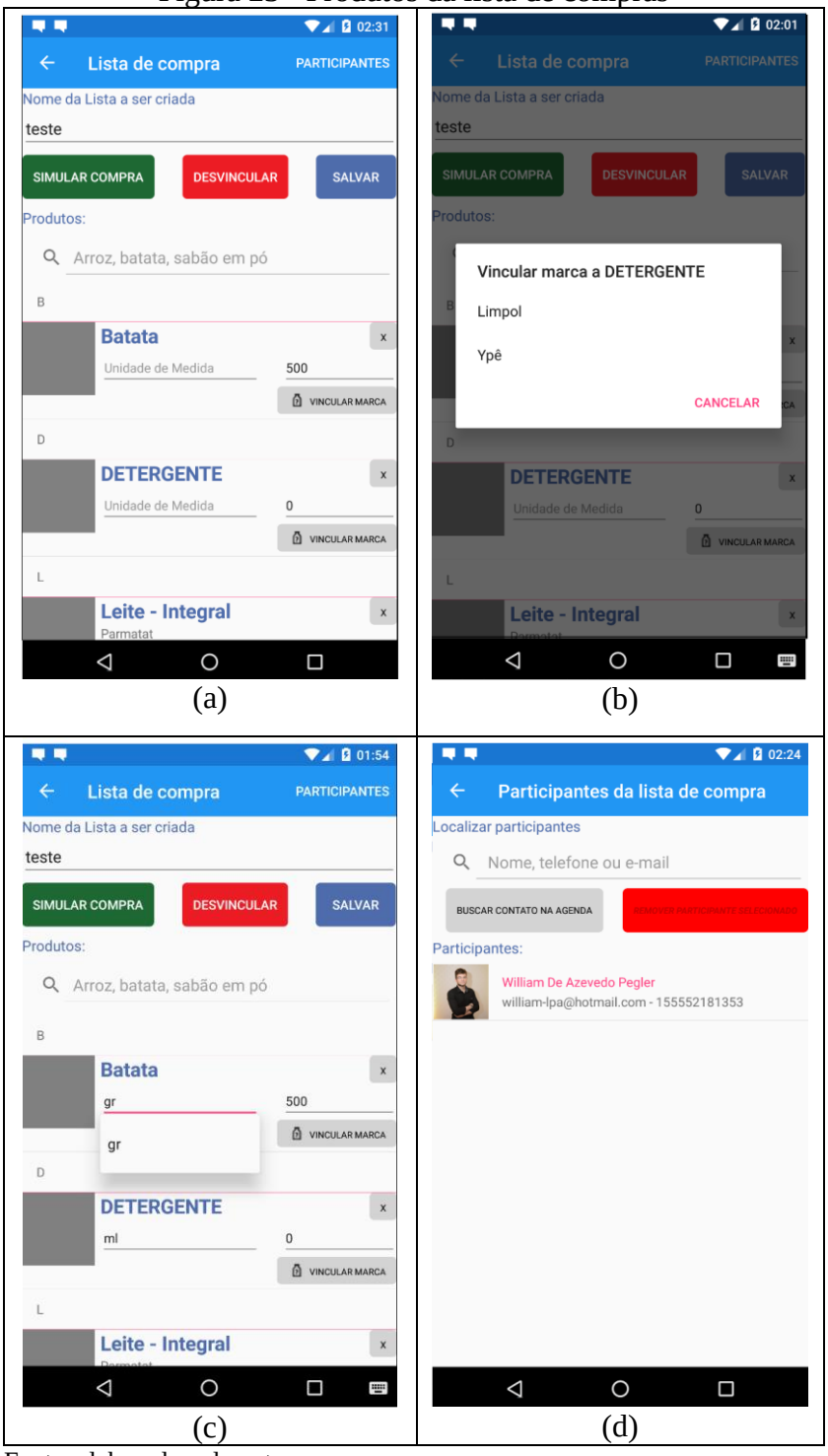
Quadro 15 – Localizando produtos na lista de compras

```
 1  private async void ExecuteSearchProductAsync(string expression)
 2  {
 3    if (expression == null) return;
 4
 5    if (expression?.Length == 1 && CachedList.Count == 0)
 6    {
 7      IsNotSearching = false;
 8      OnPropertyChanged(nameof(IsNotSearching));
 9      CachedList = new ObservableCollection<GroupCollection<char, ProdutoListaCompraViewModel>>
10                                                                          (ProdutosListaCompra);
11    }
12     ProdutosListaCompra.Clear();
13
14     if (string.IsNullOrEmpty(expression))
15     {
16       IsNotSearching = true;
17       OnPropertyChanged(nameof(IsNotSearching));
18       foreach (var item in CachedList)
19       {
20         ProdutosListaCompra.Add(item);
21       }
22       await listaCompraService
23             .LoadAutoCompleteAsync(TransformGroupedCollectionToModel(ProdutosListaCompra)
24             .Select(x => x.Produto?.Id).ToArray());
25      CachedList.Clear();
26      return;
27     }
28     var produtos = await listaCompraService.ObterProdutoPorNome(expression);
29     if (produtos != null)
30     GroupCollection(produtos);
31  }
```

Fonte: elaborado pelo autor.

A função de busca, definida na linha 1, é chamada a cada vez que o usuário alterar algum caractere no campo de busca. Na linha 9 todos os produtos adicionados serão transferidos para uma estrutura auxiliar para que não sejam sobrescritos com os dados da pesquisa. Todos os produtos localizados de acordo com a digitação feita pelo usuário serão exibidos como resultado. Quando a busca for finalizada, o trecho de código da linha 14 será executado e os antigos produtos da estrutura de dados serão transferidos para a coleção principal. Os novos produtos terão campos adicionais/opcionais conforme pode ser observado na Figura 23.

Figura 23 - Produtos da lista de compras




Fonte: elaborado pelo autor.

Conforme ilustrado na Figura 23 item (a), para cada produto da lista de compras poderá ser definido opcionalmente a marca do produto e quais são as dimensões dele. Estas informações serão utilizadas para definir qual estabelecimento tem melhor custo-benefício. A marca pode ser selecionada dentre as marcas já existentes para aquele produto, expostos em uma modal ilustrada na Figura 23 item (b). A unidade de medida do produto é um campo livre de digitação, exemplificado na Figura 23 item (c) que irá sugerir ao usuário as opções

disponíveis para o determinado produto. Conforme Figura 23 item (d), os participantes de uma lista de compra funcionam de forma semelhante aos grupos de compartilhamento de oferta. Todas as listas de compras são privadas, mas permitem ao usuário adicionar mais participantes para que ambos compartilhem e interajam na lista de compras. O principal objetivo da lista de compras além de compor uma lista de necessidades do usuário é simular o custo dela, apresentando para o usuário o estabelecimento com melhor custo benefício em relação aos seus itens de compra. O algoritmo que realiza a simulação de compras pode ser visto no Quadro 16.

Quadro 16 – Algoritmo de simulação de compras

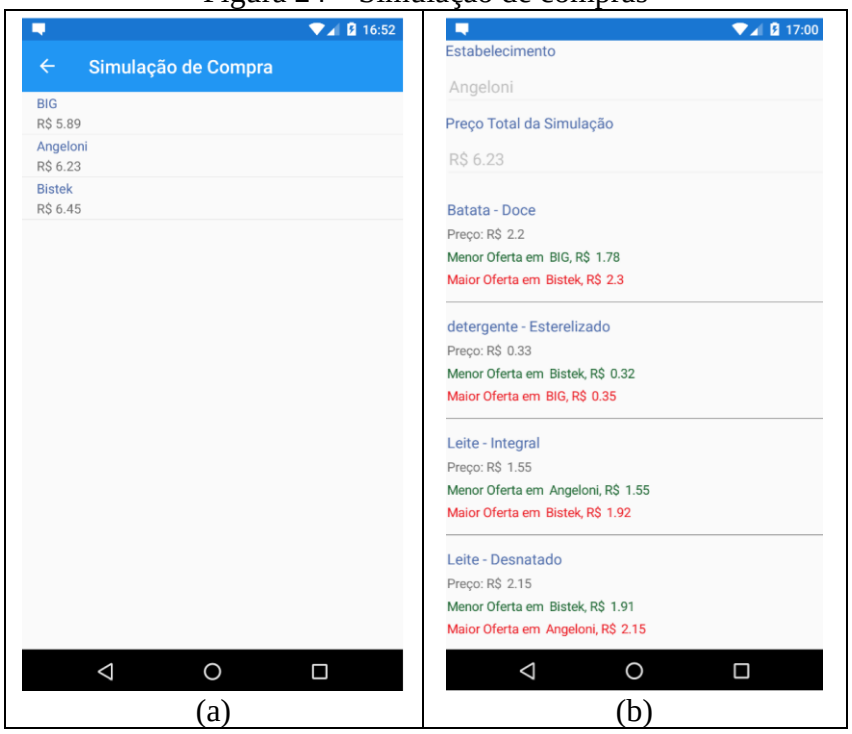
```
 1  internal async Task<IEnumerable<ProdutoDto>> ObterMelhoresEstabelecimentos
 2                        (string idEstabelecimento, IEnumerable<ProdutoListaCompra> initialValue)
 3  {
 4     var ids = await ofertaRepository.SyncTableModel
 5                     .Select(x => new { x.Id, x.IdEstabelecimento })
 6                     .ToListAsync().ConfigureAwait(false);
 7     var idsOfertas = ids.Where(x => x.IdEstabelecimento == idEstabelecimento).ToArray();
 8     List<Oferta> ofertas = new List<Oferta>();
 9     foreach (var idOferta in idsOfertas)
10     {
11        var retorno = await ObterOfertaCompletaAsync(idOferta.Id);
12        ofertas.Add(retorno);
13     }
14     List<ProdutoDto> produtos = new List<ProdutoDto>();
15
16     foreach (var produtoLista in initialValue)
17     {
18       var melhorOferta =
19       ofertas.Where(x => x.CarteiraProduto.Produto.Nome.ToLower() ==
20                      produtoLista.Produto?.Nome?.ToLower() &&
21                     (produtoLista.Produto.IdTipo == null || x.CarteiraProduto.Produto.IdTipo ==
22                      produtoLista.Produto.IdTipo) && (produtoLista.IdMarca == null ||
23                      x.CarteiraProduto.IdMarca == produtoLista.IdMarca) &&
24                      produtoLista.IdUnidadeMedida == null ||
25                      x.CarteiraProduto.Produto?.UnidadeMedida?.Nome?.ToLower() ==
26                      produtoLista.UnidadeMedida?.Nome?.ToLower()) &&
27                      (produtoLista.QuantidadeMensuravel == 0 ||
28                      x.CarteiraProduto.Produto.QuantidadeMensuravel ==
29                      produtoLista.QuantidadeMensuravel))
30              .OrderBy(x => x.PrecoAtual).ToArray().FirstOrDefault();
31        if (melhorOferta != null)
32             produtos.Add(new ProdutoDto(melhorOferta));
33     };
34    return produtos;
35  }
```

Fonte: elaborado pelo autor.

Inicialmente o algoritmo obtém todos os estabelecimentos que atendam aos critérios dos produtos estabelecidos na lista de compras juntamente com todos os seus preenchimentos opcionais como marcas e unidade de medidas. Quando os estabelecimentos forem identificados é inicializada a segunda parte do algoritmo que é a obtenção dos estabelecimentos com menores preços. A função da linha 1 será chamada para cada estabelecimento que atender aos critérios da lista de compras. A partir da linha 16, o algoritmo tentará localizar cada produto que o usuário inseriu na lista de compras dentro das melhores ofertas que foram obtidas e que respeitem as condições de busca. Os produtos serão

adicionados a uma coleção que será enviada para classe responsável por gerenciar o algoritmo de simulação de compras. Na Figura 24 são apresentados os resultados da simulação.

Figura 24 – Simulação de compras




(a) (b)

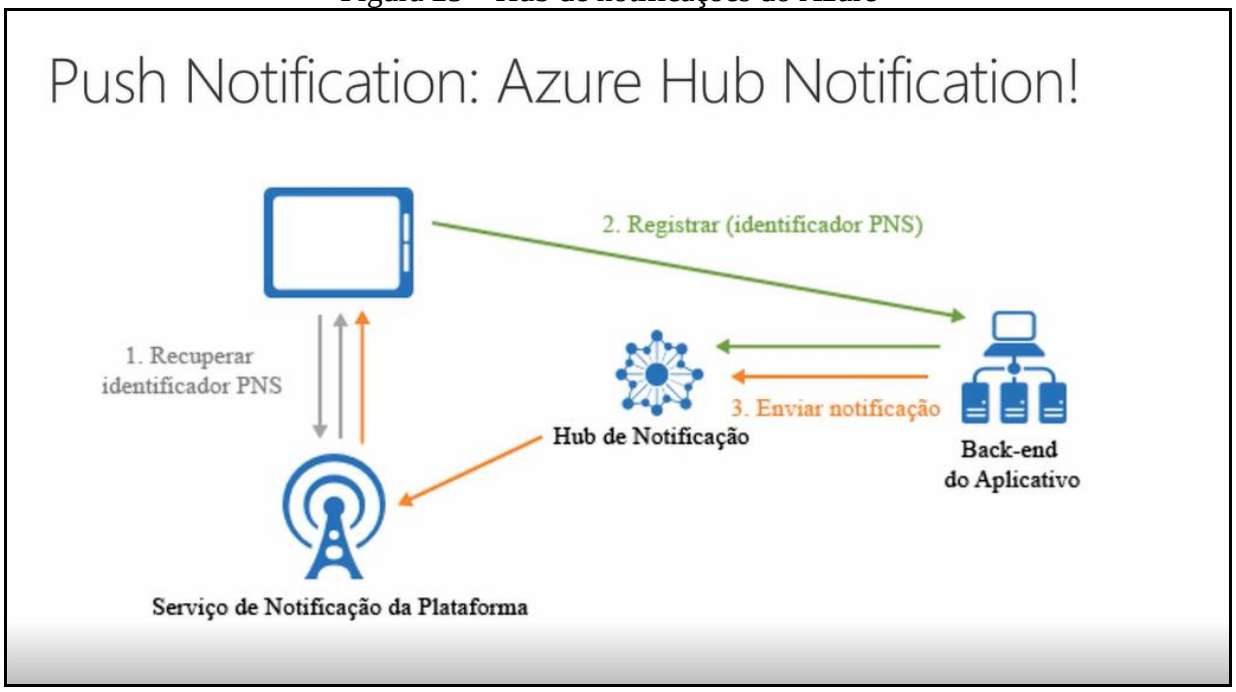
Fonte: elaborado pelo autor.

No item (a) da Figura 24 pode-se observar a lista dos estabelecimentos que possuem os produtos indicados na lista de compras. Os estabelecimentos serão exibidos de acordo com o custo total, de forma crescente. Além do custo total da compra, a aplicação permite ao usuário a visualização detalhada do preço de cada produto assim como, a variação de preço entre estabelecimentos (maior e o menor custo por produto), conforme mostra a Figura 24 item (b).

### 3.3.2.3 Envio de mensagens de notificação para os dispositivos

A notificação entre dispositivos é realizada sempre que existir a necessidade de comunicar um ou mais usuários sobre quaisquer alterações significantes no aplicativo. Esta notificação é realizada através da tecnologia de *push notification*, que possibilita o envio de notificações aos usuários sem que ele esteja executando o aplicativo. Cada plataforma é responsável pelo gerenciamento de infraestrutura de rede a fim de garantir que a mensagem seja entregue ao usuário do aplicativo correto. O processo utilizado para registrar a aplicação na plataforma de notificação do Android está disponível no APÊNDICE B. A Figura 25 demonstra o fluxo realizado para registrar o aplicativo na rede de notificações.

Figura 25 – Hub de notificações do Azure




Fonte: adaptado de MSDN (2015).

Ao iniciar o aplicativo é gerado, pelo Azure, um *token* de identificação única do dispositivo no hub de notificações. Durante a inicialização, informações adicionais do usuário como *tags* e *templates* também serão geradas, conforme demonstrado no Quadro 17.

Quadro 17 – Registrando o aplicativo no *hub* de notificação

```
 1  public async void RegisterAsync(MobileServices.Push push, IEnumerable<string> tags)
 2  {
 3    var installation = new DeviceInstallation
 4    {
 5      InstallationId = Client.InstallationId,
 6      Platform = "gcm",
 7      PushChannel = RegistrationID
 8    };
 9    using (var scope = App.Container?.BeginLifetimeScope())
10    {
11      (await scope?.Resolve<GrupoOfertaService>()?
12             .CarregarGrupoDeOfertasUsuarioLogadoAsync()).Select(x => {
13              installation.Tags.Add(x.Id); return x; }).ToArray();
14      (await scope?.Resolve<ListaCompraService>()?
15             .CarregarListasDeComprasUsuarioLogadoAsync()).Select(x => {
16              installation.Tags.Add(x.Id); return x; }).ToArray();
17      installation.Tags.Add("user:" + scope?.Resolve<AzureService>()?.CurrentUser.User.Id);
18    }
19    PushTemplate genericTemplate = new PushTemplate
20    {
21      Body = @"{""data"":{""key"":""$(keyParam)"",""message"":""$(messageParam)""}}"
22    };
23    PushTemplate ofertaTemplate = new PushTemplate
24    {
25      Body = @"{""data"":{""key"":""$(keyParam)"",""ofertaTitle"":""$(ofertaTitleParam)"",
26                          ""ofertaDescription"":""$(ofertaDescriptionParam)"",
27                          ""idOFerta"":""$(ofertaIdParam)""}}"
28    };
29    installation.Templates.Add("genericTemplate", genericTemplate);
30    installation.Templates.Add("ofertaTemplate", ofertaTemplate);
31    await Client.InvokeApiAsync<string[], object>("refreshPushRegistration",
32                                                  installation.Tags.ToArray());
33  }
```

Fonte: elaborado pelo autor.

Na linha 3 é instanciado um objeto da qual será transformado em um JSON, sendo enviado para o Azure. Este objeto é composto pelo código de instalação do dispositivo, que será utilizado como identificador no Azure. Ele também contém as informações sobre a plataforma e o identificador da aplicação disponibilizado no Firebase. O aplicativo utiliza a chave primária de cada grupo de compartilhamento de oferta que o usuário está participando para adiciona-lo ao objeto como uma *tag*. Esta *tag* será utilizada para notificar todas as instalações do hub. O mesmo procedimento é realizado com as listas de compras ao qual o usuário está vinculado, conforme demonstrado na linha 14. Na instalação são adicionados *templates* responsáveis por informar ao hub quais modelos de mensagens a aplicação consegue receber. No servidor a aplicação se conecta com o hub através de bibliotecas gerenciadas pelo Azure para manipular a instalação de cada dispositivo, conforme demonstrado no Quadro 18.

Quadro 18 – Criação de uma instalação de um novo dispositivo

```
 1  var azure = require('azure-sb');
 2  var notificationHubService = azure.createNotificationHubService('goodbuy-push',
 3                               'Endpoint=sb://goodbuy.servicebus.windows.net/;
 4                                SharedAccessKeyName=DefaultFullSharedAccessSignature;
 5                                SharedAccessKey=6KUDiCOPb5sNJBlAjvdaOjyc4/f//2yasT8+o2vTxFc=');
 6  module.exports = {
 7      post: function (request, response, next) {
 8          var tags = request.body;
 9          var installationId = request.headers['x-zumo-installation-id'];
10          notificationHubService.getInstallation(installationId, (error, installation) => {
11              tags.forEach(function (tag) {
12
13                  if (!installation.tags) {
14                      installation['tags'] = [];
15                  }
16                  var index = installation.tags.indexOf(tag);
17                  if (index === -1) {
18                      installation.tags.push(tag);
19                  }
20              }, this);
21             notificationHubService.createOrUpdateInstallation(installation,(error, result) => {
22                                                                                          });
23          });
24          response.json({ sent: 'OK' });
25      }
26  };
```

Fonte: elaborado pelo autor.

Na linha 2 é realizado a importação do pacote de conexão do Service Bus do Azure para a manipulação da instância do hub de notificações. Na linha 7 é definido uma API REST para registrar ou atualizar *tags* da instalação do dispositivo. Sempre que o aplicativo for inicializado, ele atualizará a instalação de notificação pertencente ao usuário *logado* para que não ocorra expiração do *token*. O identificador da instalação do usuário é recuperado através do cabeçalho do protocolo Hypertext Transfer Protocol (HTTP), conforme demonstrado na linha 9. O segundo parâmetro desta chamada é uma função que será executada após obtenção da instalação. Todas as *tags* enviadas ao servidor serão inclusas na instalação atual, conforme

indicado nas linhas 18 e 21. Quando algum `Usuário` desejar compartilhar alguma oferta com outro usuário, a notificação será enviada do servidor ao hub de notificações, conforme mostra o Quadro 19.

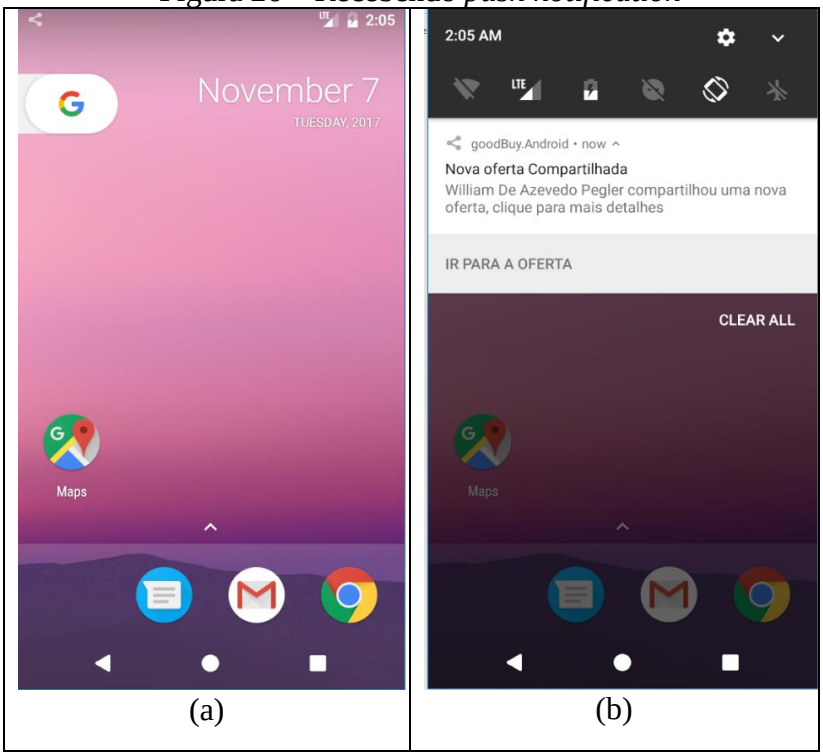
Quadro 19 – Compartilhando oferta com usuário no servidor

```
 1  var azure = require('azure-sb');
 2  var notificationHubService = azure.createNotificationHubService('goodbuy-push',
 3                               'Endpoint=sb://goodbuy.servicebus.windows.net/;
 4                                SharedAccessKeyName=DefaultFullSharedAccessSignature;
 5                                SharedAccessKey=6KUDiCOPb5sNJBlAjvdaOjyc4/f//2yasT8+o2vTxFc=');
 6  module.exports = {
 7      post: function (request, response) {
 8          var title = request.body.title;
 9          var description = request.body.description;
10          var tag = request.body.idUser;
11          var idOferta = request.body.idOferta;
12          var payload = {
13              data: {
14                  key: '705',
15                  ofertaTitle: title,
16                  ofertaDescription: description,
17                  idOferta: idOferta
18              }
19          };
20          notificationHubService.gcm.send(`user:${tag}`, payload, function (error) {
21              if (!error) {
22                console.log(error')
23              }
24      });
25          response.json({ sent: 'OK' });
26  }
```

Fonte: elaborado pelo autor.

Na linha 7 é definida a API que realizará o envio da oferta compartilhada. Ela precisa de dois parâmetros. O primeiro é a requisição HTTP enviada pelo aplicativo para o servidor e o segundo será a resposta da requisição após a execução da função. A partir da linha 12 é gerado o objeto que será enviado ao dispositivo móvel de destino. Na linha 14 é atribuído o código da notificação, informação que será necessária para que o aplicativo decida qual *template* utilizar na notificação. O envio da mensagem é realizado na linha 20. A notificação é enviada ao hub de notificação do Azure que direciona o conteúdo da mensagem para a rede de notificações da plataforma correta. No caso do Android, o Firebase é o responsável por entregar a notificação ao dispositivo. O celular destinatário receberá e exibirá a mensagem, conforme mostra a Figura 26.

Figura 26 – Recebendo *push notification*




(a) (b)

Fonte: elaborado pelo autor.

Quando o *smartphone* receber a mensagem, a notificação será exibida no aparelho mesmo que a aplicação não esteja executando em primeiro plano, conforme exibe a Figura 26 item (a). O `Usuário` ainda poderá visualizar detalhes da mensagem de notificação, executando determinadas ações conforme demonstra a Figura 26 item (b). Para interceptar a mensagem, o aplicativo utiliza a biblioteca `GCM.Client`, que possibilita a manipulação de notificações no Android, como pode ser visto no Quadro 20.

Quadro 20 – Construindo um *push notification*

```
1  private void CreateNotification(string title, string desc, string contentText,
2                                  string actionName, string parameter = null)
3  {
4    var notificationManager = GetSystemService(Context.NotificationService) as
5                              NotificationManager;
6
7    var startupIntent = new Intent(this, typeof(MainActivity));
8    startupIntent.PutExtra("param", parameter);
9
10    var builder = new Notification.Builder(this)
11                  .SetContentTitle(title)
12                  .SetContentText(contentText)
13                  .SetSmallIcon(Android.Resource.Drawable.IcMenuShare)
14                  .SetContentIntent(PendingIntent.GetActivity(this, 0, startupIntent,
15                                    PendingIntentFlags.UpdateCurrent))
16                  .AddAction(Android.Resource.Drawable.IcMenuSend, parameter == "grupos" ?
17                  "Ver grupos" : (parameter.Contains("+") ? "Ver Lista" : "Ir para a oferta"),
18                   PendingIntent.GetActivity(this, 0, startupIntent,
19                                             PendingIntentFlags.UpdateCurrent));
20
21    Notification notification = new Notification.BigTextStyle(builder).BigText(desc).Build();
22    notification.Flags |= NotificationFlags.AutoCancel;
23    notificationManager.Notify(0, notification);
24 }
```

Fonte: elaborado pelo autor.

O serviço `NotificationManager` do Android é responsável por informar ao usuário quando algum evento é realizado em *background*. A partir da linha 10 é instanciado um objeto de notificação utilizando o padrão de projeto *builder* para construir e personalizar a mensagem exibida de acordo com os parâmetros passados para o método. Na linha 21 são aplicados estilos para que a notificação possua suporte a botões de ações e textos com mais de uma linha. A partir disso, ela é exibida na tela.

## 3.4 ANÁLISE DOS RESULTADOS

Ao longo desta seção serão apresentados os resultados obtidos a partir dos questionários do APÊNCIDE C. Nielsen (1993), explica que para se obter resultados satisfatórios em testes de usabilidade, são necessários apenas cinco usuários. Ele alega que acima desta quantidade de usuários, o tempo de avaliação começa a ser desperdiçado pois praticamente os mesmos problemas de usabilidade já detectados começarão a se repetir, sem a identificação de novos erros. Todavia, este experimento foi realizado com 15 usuários para uma maior abrangência de perfis de usuário. A seção 3.4.1 relata os resultados que foram obtidos com o questionário de análise do perfil dos usuários. Na seção 3.4.2 são demonstrados os resultados do questionário de atividades dos usuários. Na seção 3.4.3 estão os resultados obtidos no questionário de usabilidade. Por fim, é realizada uma comparação entre as características dos trabalhos correlatos e o trabalho desenvolvido.

### 3.4.1 Análise do perfil dos usuários

No Quadro 21 estão as respostas dos 15 usuários que responderam o questionário do perfil dos usuários.

Quadro 21 – Questionário de análise do perfil dos usuários

| | |
|---|---|
| Qual o seu sexo? | 53,3% masculino<br>46,7% feminino |
| Qual sua idade? | 86,7% entre 18 e 30 anos<br>6,6% entre 30 e 45 anos<br>6,7% Acima de 45 anos |
| Você costuma fazer pesquisas de preço online antes de efetuar a compra? | 66,7% sim<br>33.3% não |
| Você costuma pesquisar antes de realizar compras no supermercado? | 20.1% sim, sempre pesquiso<br>53,3% pesquiso quando há tempo<br>13,3% não, costumo ir sempre no mesmo estabelecimento de minha preferência.<br>13,3% não, costumo ir sempre na loja mais próxima de casa |
| Quais são os meios que você geralmente utiliza para ficar ciente de promoções nos supermercados locais? Você pode selecionar várias opções | 26,7% Rádio<br>13,3% Televisão<br>6,7%Jornal<br>53,3% Internet<br>80% Panfletos |
| Você possui o hábito de solicitar ajuda na elaboração da lista de compras de sua residência? | 20% sim<br>80% não |
| No supermercado durante as compras, você tem o hábito de avisar seus amigos ou parentes a respeito de uma possível promoção do estabelecimento? | 53,3% sim<br>46,7% não |
| Você é membro de algum grupo de compartilhamento de promoções e vendas em alguma rede social como o Facebook ou o WhatsApp? | 20% sim<br>80% não |

Fonte: elaborado pelo autor.

Com base no quadro acima, percebesse que a maioria dos usuários é do sexo masculino e tem idade entre 18 a 30 anos. É possível afirmar que 66,7% dos usuários tem o hábito de realizar pesquisas online antes de efetuar alguma compra. Outro dado observado é que 13,3% dos usuários tem a preferência em dirigirem-se sempre ao mesmo estabelecimento, enquanto, 13,3% dos usuários preferem o estabelecimento mais próximo de sua residência e, que necessariamente não tem o melhor custo benefício. Segundo os usuários, a grande maioria (80%) não solicita nenhum tipo de auxílio na elaboração da lista de compras de sua residência e 53.3% dos entrevistados possuem o hábito de compartilhar de alguma forma as promoções existentes em estabelecimentos com seus parentes e amigos.

### 3.4.2 Análise das atividades do usuário

Após a análise do perfil dos voluntários, foi realizada a avaliação dos dados obtidos a partir da lista de tarefas executada pelos usuários. Foi solicitado a instalação do aplicativo e a realização de algumas atividades a fim de avaliar o fluxo das funcionalidades desenvolvidas e

disponibilizadas pela aplicação. A Tabela 1 apresenta os resultados das atividades que estão relacionadas a esse objetivo.

Tabela 1 – Questionário de atividades dos usuários

| Atividades | Percentual (%) de conclusão |
|---|---|
| Instalação do aplicativo | 86,7% - Sim<br>6,6% - Não<br>6,7% - Não responderam |
| Login no aplicativo | 53,3% - Sim, com o Facebook<br>33,3% - Sim, com autenticação local<br>13,4% - Não responderam |
| Cadastrar uma oferta | 86,7% - Sim<br>13,3% - Não responderam |
| Dúvidas no preenchimento da oferta | 6,6% - Sim<br>80% - Não<br>13,4% - Não responderam |
| Cadastrar grupos | 80% - Sim<br>6,6% - Não<br>13,4% - Não responderam |
| Grupos públicos | 60% - Sim<br>20% - Não<br>20% - Não responderam |
| Consultar oferta | 80% - Sim<br>20% - Não responderam |
| Confiabilidade da oferta | 86,7% - Sim<br>13,3% - Não responderam |
| Compartilhar Oferta | 60% - Sim<br>6,7% - Não<br>33,3% - Não responderam |
| Cadastrar lista de compras | 53,3% - Sim<br>26,7% - Não<br>20% - Não responderam |
| Simular compras | 46,7% - Sim<br>33,3% - Não<br>20% - Não responderam |

Fonte: elaborado pelo autor.

Com base na Tabela 1, pode-se perceber que apenas a lista de compras teve índice de aprovação abaixo de 50%. Do contrário, pode-se observar que na maioria das questões o índice de aprovação foi superior ao de reprovação ou abstenção. Algumas observações feitas pelos usuários indicam que o aplicativo deveria apresentar uma mensagem de confirmação ao realizar o cadastro de uma oferta para não gerar incertezas se a oferta foi ou não adicionada. Um usuário comentou que ele não conseguiu instalar o aplicativo pois o dispositivo tinha espaço insuficiente. Outro ponto mencionado por alguns usuários foi em relação ao componente de busca, que não apareceu ao localizar produtos e grupos públicos. Isso pode ter acontecido por causa do modelo do aparelho ou versão do Android.

### 3.4.3 Análise da usabilidade do aplicativo

Após analisar os resultados da lista de tarefas e perfil dos usuários, foram analisados os resultados obtidos do questionário de usabilidade aplicativo. A primeira parte do questionário visa avaliar a objetividade, desempenho, experiência com o usuário e o funcionamento do aplicativo. Os resultados obtidos com 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 | 40% | 40% | 13,3% | 6,7% | |
| A velocidade da sincronização de ofertas é rápida | 66,7% | 20% | 13,3% | | |
| É fácil navegar entre as telas do aplicativo | 53,3% | 33,3% | 6,7% | 6,7% | |
| O aplicativo em algum momento parou inesperadamente | 33,3% | 6,7% | 6,7% | 6,7% | 46,6% |
| A aparência do aplicativo é boa (cores e ícones) | 33,3% | 46,7% | 6,7% | | 13,3% |
| O tempo de notificação de compartilhamento de ofertas ocorreu dentro do prazo esperado | 66,7% | 20% | 13,3% | | |

Fonte: elaborado pelo autor.

A partir da Tabela 2, pode-se considerar que o aplicativo é simples e objetivo. De acordo com os dados acima, a aplicação ainda pode ser melhorar o tempo de resposta da interação feita pelo usuário e na interceptação de erros na comunicação entre o aplicativo e o servidor visto que 33,3% dos usuários relataram interrupções inesperadas durante a realização dos testes. Outro ponto que pode ser melhorado é a aparência do aplicativo pois apenas 33,3% dos usuários concordaram plenamente sobre a aparência da aplicação. Também pode-se afirmar que o desempenho da sincronização de ofertas e do tempo de envio de notificações foi satisfatório, sendo que cerca de 66,7% dos usuários concordaram plenamente com a eficácia desses recursos.

Ainda neste formulário foram realizadas algumas perguntas a respeito do entendimento e do propósito da aplicação na perspectiva do usuário. Os resultados deste último formulário estão ilustrados 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 existir grupos com interesses de ofertas em comuns para a divulgação de promoções? | 93,3% - Sim<br>6,7% - Não |
| Você enfrentou problemas de lentidão durante a utilização do aplicativo? | 6,7% - Sim<br>93,3% - Não |
| Você conseguiu visualizar sua oferta? | 93,3% - Sim<br>6,7% - Não |
| Você acha interessante a funcionalidade de monitorar ofertas até um determinado preço disponível no aplicativo? | 100% - Sim |
| Você utilizaria o aplicativo para pesquisa de preços? | 86,7% - Sim<br>13,3% - Não |
| Você recomendaria o aplicativo para outras pessoas? | 93,3% - Sim<br>6,7% - Não |
| Você confiaria nessa simulação de compras realizada a ponto de se dirigir até o estabelecimento sugerido, mesmo que ele não fosse o estabelecimento que você compra com frequência ou o mais próximo de sua residência? | 93,3% - Sim<br>6,7% - Não |
| Você acha interessante a funcionalidade de elaborar lista de compras com a participação de conhecidos ou familiares? | 93,3% - Sim<br>6,7% - Não |

Fonte: elaborado pelo autor.

De acordo com a Tabela 3, os usuários da aplicação conseguiram compreender bem o objetivo e o propósito do aplicativo que é compartilhar ofertas com grupos de interesses em comum ou entre si. Mais uma vez o desempenho do aplicativo se mostrou satisfatório, apresentando lentidão em um percentual mínimo de usuários. Além disso, a maior parte dos usuários utilizaria o aplicativo para realizar pesquisas de preços e inclusive o indicaria para outras pessoas. A possibilidade de elaborar uma lista de compras com familiares e conhecidos e a confiança na simulação de preços nos produtos da lista de compras também se mostraram bastante positivas. Além das perguntas do questionário, foi disponibilizado um espaço para que o usuário pudesse fazer críticas e sugestões. Alguns usuários sugeriram que a lista de compras deveria permitir a inserção de novos produtos que não estivessem pré-cadastrados na base de ofertas. Um usuário sugeriu que o compartilhamento de ofertas tivesse uma opção que permitisse o compartilhamento externo da oferta através de SMS, e-mail e outras redes sociais para divulgação do aplicativo e maior abrangência de informação.

### 3.4.4 Comparação com trabalhos correlatos Comparação com trabalhos correlatos

O Quadro 22 apresenta uma comparação entre as principais características dos trabalhos correlatos e do trabalho desenvolvido.

Quadro 22 – Comparativo entre os trabalhos correlatos

| **Características** / **Trabalhos** | **Trabalho desenvolvido** | **Martins Software (2015)** | **OOO "Kupi baton" (2012)** | **Bring! Labs AG (2012)** |
|---|---|---|---|---|
| Plataforma | Android | Android | Android/iOS | Android/iOS |
| Lista de compras colaborativa | Sim | Não | Sim | Sim |
| Sincronização de dados em tempo real | Sim | Não | Sim | Sim |
| Notificações Automáticas | Sim | Não | Sim | Parcialmente |
| Armazena histórico de preços | Sim | Sim | Não | Não |
| Produtos pré-cadastrados | Não | Sim | Sim | Sim |
| Grupos de compartilhamento de ofertas | Sim | Não | Não | Não |
| Base de dados de ofertas colaborativa | Sim | Não | Não | Não |

Fonte: elaborado pelo autor.

A partir do Quadro 22, observa-se que as aplicações Buy Me a Pie – Lista de Compras (OOO KUPI BATON, 2012) e Bring! Shopping List (BRING! LABS AG, 2012) e o trabalho desenvolvido são os aplicativos que realizam a sincronização dos dados da aplicação em tempo real. Todavia, o aplicativo da Bring! Labs AG (2012), o Buy Me a Pie – Lista de Compras (OOO KUPI BATON, 2012) e o trabalho desenvolvido são os únicos aplicativos que possibilitam a interação com mais de um usuário em uma mesma lista de compra. O aplicativo SoftList (MARTINS SOFTWARE, 2015) e o trabalho desenvolvido são os únicos entre os trabalhos que possibilitam realizar a contabilização e o controle de preços na lista de compra através de relatórios e totalizações. Além do trabalho desenvolvido, observa-se também que o Buy Me a Pie - Lista de Compras é o único aplicativo dos correlatos que realiza notificações *push* automaticamente para o usuário. Ele também permite que a forma de configuração de notificação seja alterada, pois ela vem desabilitada por padrão. O trabalho desenvolvido é o único dos demonstrados acima que não possui registros pré-cadastrados na base de dados para uma melhor conveniência do usuário.

Diferente dos demais trabalhos, o trabalho desenvolvido é o único que possui grupos de compartilhamento de oferta para que usuários com interesse em comum possam disseminar ofertas específicas, alertando os demais participantes a respeito de quais estabelecimentos possuem estes produtos. Outro ponto a observar é que o trabalho desenvolvido utiliza uma base de dados pública e colaborativa entre os usuários, da qual possibilita que ele mesmo possa fazer o cadastro e o anúncio de uma oferta ou promoção em determinado local.

# 4 CONCLUSÕES

Este trabalho apresentou o desenvolvimento de uma aplicação móvel colaborativa que visa auxiliar o usuário na elaboração de listas de compras e na disseminação de promoções em estabelecimentos através do compartilhamento de ofertas utilizando produtos cadastrados pelos próprios usuários do aplicativo. Para este trabalho haviam dois objetivos específicos, o primeiro era utilizar uma base de dados hospedada na nuvem para realizar a sincronização de forma constante dos dados mais recentes compartilhados entre os usuários. Já o segundo objeto era disponibilizar um mecanismo para validação da integridade dos preços atribuídos pelos usuários da aplicação.

O primeiro objetivo que era hospedar e utilizar serviços por uma plataforma na nuvem foi atendido através do Microsoft Windows Azure. O Azure forneceu toda a infraestrutura necessária para o desenvolvimento da aplicação na parte servidor. Foi disponibilizado o SQL Server para utilização do banco de dados e um serviço de aplicação móvel em Node.js pré-configurado com o *framework* Express, Babel e outras bibliotecas para comunicação com serviços da própria plataforma, como o acesso autenticado ao servidor e o hub de notificações. Uma das dificuldades encontradas durante o desenvolvimento do servidor foi realizar o *broadcast* de uma oferta para todos os participantes de um grupo. Para que todos os participantes de um determinado grupo recebam uma notificação, cada um dos participantes deveria possuir uma *tag* com o identificador deste grupo vinculado ao seu usuário de instalação no hub de notificações. As *tags* desta instalação expiravam em um determinado tempo, fazendo que os usuários parassem de receber notificações do respectivo grupo. Para resolver este problema, cada vez que o usuário abrir a aplicação será feita uma requisição ao servidor para que estas *tags* sejam atualizadas.

A sincronização de dados no servidor de forma constante ainda no primeiro objetivo específico também foi atendido. Desenvolvendo o aplicativo móvel com o Xamarin Forms da Microsoft e utilizando as bibliotecas do Azure para a parte cliente da aplicação, a sincronização dos dados foram realizadas sem muitos problemas. A persistência é realizada primeiramente no dispositivo local, através do banco de dados relacional SQLite. Após a persistência local os dados são enviados ao servidor em um determinado tempo ou ao concluir determinadas operações.

O segundo objetivo também foi atendido. Alguns mecanismos de integridade e confiabilidade foram utilizados. Para autenticar-se na aplicação, é identificado o número do dispositivo para que seja utilizado como identificador da tabela de usuário. Ainda no processo

de *login*, o usuário pode optar por autenticar-se com o Facebook ou com as informações do seu dispositivo local. Independentemente da opção selecionada, será obtido um perfil do usuário *logado* com informações básicas como número do telefone, nome completo, e-mail e município. Dentro da aplicação, para controlar a integridade da oferta cadastrada por estes usuários é possível avaliar cada oferta informada, alertando os demais usuários a respeito da confiabilidade das ofertas.

O desenvolvimento da aplicação móvel foi feito utilizado o *framework* Xamarin Forms, desenvolvido pela Microsoft. Em geral, o uso do *framework* se mostrou bastante satisfatório, pois com esta tecnologia é possível escrever de forma isolada a regra de negócio e lógica da aplicação, sendo possível compilar o aplicativo de forma nativa no Android, Windows Phone, iOS e Tizen. O *framework* conta ainda com um recurso de padronização de codificação de interfaces gráficas, escrevendo o código apenas uma vez e renderizando de forma nativa em cada plataforma. Outro recurso utilizado no desenvolvimento do trabalho foi o Xamarin.Forms Previewer, uma ferramenta de visualização gráfica da qual é possível manipular as telas do aplicativo sem a necessidade de compilar a aplicação novamente. O *framework* conta também com um recurso para a conversão de bibliotecas nativas do iOS em Objetive C (.a) e o Java Archive (.jar) do Android para utilização em uma aplicação .NET através de uma Dynamic Link Library (.dll).

A aplicação apresentou um comportamento satisfatório visto que seus principais objetivos foram atingidos. O usuário consegue acessar o aplicativo de forma autenticada, cadastrar ofertas, avaliar as últimas ofertas, interagir com diversos grupos, obter o melhor custo-benefício de acordo com uma lista de compras e compartilhar ofertas entre os demais usuários do aplicativo. Apesar de atender o que foi proposto, a aplicação possui alguns pontos que precisam ser melhorados. A aparência das telas do dispositivo precisa ser elaborada de acordo com a avaliação de alguns usuários. Para uma melhor interação com os usuários, também se faz necessário uma população inicial na base de dados com diversas marcas, produtos e estabelecimentos. Também pode-se melhorar a inteligência da simulação de compras, fazendo otimizações nas buscas e a realização de um mapeamento de equivalências entre unidade de medidas, pois atualmente a busca é feita através de uma comparação exata.

De modo geral o trabalho desenvolvido torna-se relevante principalmente no âmbito acadêmico e tecnológico pois aborda o uso das mais recentes tecnologias de codificação nativa em multiplataformas que podem ainda ser mais exploradas no desenvolvimento de aplicativos móveis tanto no meio comercial quanto na área científica. O Xamarin.Forms torna-se uma alternativa ao uso de outros *frameworks* de desenvolvimento híbrido como o

caso do React Native, o NativeScript ou o Phonegap. Além disso, o compartilhamento de ofertas e a elaboração de listas de compras não se limita apenas a supermercados, a ideia pode ser estendida e melhor explorada em diversas outras áreas.

## 4.1 EXTENSÕES

Sugerem-se as seguintes extensões para trabalhos futuros:

a) disponibilizar uma base de dados com pré-cadastramento de marcas, produtos e estabelecimentos para melhor a usabilidade da aplicação;
b) permitir a inserção de produtos não informados como ofertas durante a elaboração de uma lista de compras;
c) melhorar a inteligência do algoritmo de simulação de compras para obtenção do melhor custo benefício;
d) adicionar meios de segurança para garantir que o número detectado ou informado pelo usuário corresponde ao número do dispositivo;
e) permitir o compartilhamento de uma oferta com fontes externas como e-mail e outras redes sociais;
f) adicionar um controle de gerenciamento do tráfego de dados de acordo com a localização do usuário para que a sincronização seja feita de acordo com a necessidade e demanda.

## REFERÊNCIAS

BARDHI, Fleura; ECKHARDT, Giana M. Access-based consumption: The case of car sharing. **Journal of consumer research**, Chicago, v. 39, n. 4, p. 881-898, 2012. Disponível em: < https://www.cass.city.ac.uk/__data/assets/pdf_file/0011/203789/Access-Based-Consumption.pdf/>. Acesso em: 26 nov. 2017.

BELK, Russell. You are what you can access: Sharing and collaborative consumption online. **Journal of Business Research**, Toronto, v. 67, n. 8, p. 1595-1600, 2014. Disponível em: < http://collaborativeeconomy.com/wp/wp-content/uploads/2015/05/Belk-R.2014.-You-are-what-you-can-access-Sharing-and-collaborative-consumption-online.Journal-of-Business-Research.pdf/>. Acesso em: 26 nov. 2017.

BRING! LABS AG. **Bring! Shopping List.** 2012. Disponível em: <https://www.getbring.com/#!/app>. Acesso em: 18 mar. 2017.

FREITAS, Andréa. **Consumo colaborativo ou economia compartilhada. Uma chance de transformarmos nosso comportamento.** São Paulo, 2016. Disponível em: < https://www.startupbrasil.org.br/2016/09/26/consumo-colaborativo-ou-economia-compartilhada-uma-chance-de-transformarmos-nosso-comportamento//>. Acesso em: 26 nov. 2017.

FREITAS, Cássio Stedetn de; PETRINI, Maira De Cássia; SILVEIRA, Lisilene Mello da. Desvendando o consumo colaborativo: uma proposta de tipologia. In: Congresso Latino-Americano no Varejo (CLAV 2016): Varejo na economia de colaboração, 9., 2016, São Paulo. **Anais**... São Paulo: FGV-EAESP, 2016. 16 p. Disponível em: < http://repositorio.pucrs.br/dspace/bitstream/10923/10138/2/DESVENDANDO_O_CONSUMO_COLABORATIVO_UMA_PROPOSTA_DE_TIPOLOGIA.pdf>. Acesso em: 22 nov. 2017.

IMPRENSA MERCADO & CONSUMO. **Economia compartilhada: 40% dos brasileiros já trocaram hotel por residência de terceiros.** [S. l.], 2017. Disponível em: < http://www.mercadoeconsumo.com.br/2017/08/10/economia-compartilhada-40-dos-brasileiros-ja-trocaram-hotel-por-residencia-de-terceiros/>. Acesso em: 20 nov. 2017.

LAAMANEN, Mikko; WAHLEN, Stefan; CAMPANA, Mario. Mobilising collaborative consumption lifestyles: a comparative frame analysis of time banking. **International Journal of Consumer Studies**, Helsinki, v. 39, n. 5, p. 459-467, 2015. Disponível em: < http://onlinelibrary.wiley.com/doi/10.1111/ijcs.12190/full?wol1URL=/doi/10.1111/ijcs.12190/full&regionCode=BR-RJ&identityKey=3144251c-3ae6-4660-a5e7-49be5bba4fff/>. Acesso em: 26 nov. 2017.

LARENTIS, Fabiano. **Comportamento do consumidor**. Curitiba: IESDE Brasil SA, 2012. Disponível em: < https://goo.gl/v8WkNK/>. Acesso em: 27 mai. 2017.

MARTINS SOFTWARE. **SoftList.** 2015. Disponível em: <https://play.google.com/store/apps/details?id=br.com.ridsoftware.shoppinglist&hl=pt_BR>. Acesso em: 16 mar. 2017.

MARTINS, Tallys Gustavo; MARTINS, Winstein Caldeira. **Sistema para Comparação de Preços de Lojas Físicas**. 2015, 47 f. TCC (Graduação) – Curso de Engenharia de Software, Universidade de Brasília, Brasília. Disponível em: <https://fga.unb.br/articles/0001/0296/TCC_1_Tallys_e_Winstein.pdf>. Acesso em: 01 abr. 2017.

MERCADO, Luís Paulo Leopoldo. **Experiências com Tecnologias de Informação e Comunicação na Educação.** Alagoas: UFAL, 2006.Disponível em: < https://goo.gl/NZoQoH >. Acesso em: 22 nov. 2017.

MINIARD, Pauli W.; ENGEL, James; BLACKWELL, Roger. **Comportamento do consumidor**. Rio de Janeiro: LTC–Livros Técnicos e Científicos Editora, 2000. Disponível em: <https://goo.gl/H0qqCR/>. Acesso em: 26 mai. 2017.

MÖHLMANN, Mareike. **Collaborative consumption:** determinants of satisfaction and the likelihood of using a sharing economy option again. 2015, 68 f. TCC (Graduação) – Curso de Economia, University of Hamburg, Hamburgo. Disponível em: < https://goo.gl/DXS8Np>. Acesso em: 23 nov. 2017.

MOTTA, Marcio. **Você pesquisa antes de comprar?.** [S. l.], 2011. Disponível em: <https://www.palavrafiel.com.br/voce-pesquisa-antes-de-comprar/>. Acesso em: 18 mar. 2017.

MSDN**. Notification Hubs Overview.** 2017. Disponível em: < https://msdn.microsoft.com/en-us/library/azure/jj927170.aspx/>. Acesso em: 11 nov. 2017.

NAPO, Paula Rodrigues. **Influências da interface gráfica em m-commerces sobre as motivações de compra online em smartphones**. 2015, 223 f. Tese (Mestrado) – Curso de Design, Universidade Federal do Paraná, Curitiba. Disponível em: < http://acervodigital.ufpr.br/bitstream/handle/1884/38772/R%20-%20D%20-%20PAULA%20RODRIGUES%20NAPO.pdf?sequence=1&isAllowed=y>. Acesso em: 27 nov. 2017.

NIELSEN, Jakob**. Usability Engineering**. San Franciso: Morgan Kaufmann Publishing, 1993.Disponível em: < https://www.nngroup.com/books/usability-engineering/>. Acesso em: 15 dez. 2017.

OLIVEIRA, Ingrid Caroline; MOREIRA NETO, Alfredo Lopes da Costa. **Comportamento do consumidor:** a influência das mídias sociais na decisão de compra de produtos gamers. 2016, 52 f. Dissertação (MBA) - Gestão de Marketing, Faculdade Cidade Verde, São Paulo. Disponível em: < https://goo.gl/oEn8l4>. Acesso em: 30 mai. 2017.

OOO "KUPI BATON". **Buy Me a Pie! - Lista de Compras.** 2012. Disponível em: <https://buymeapie.com/pt>. Acesso em: 17 mar. 2017.

PROCON- PR. **Material Escolar 2017.** Curitiba, 2017. Disponível em: <http://www.procon.pr.gov.br/modules/conteudo/conteudo.php?conteudo=677>. Acesso em: 19 mar. 2017.

PROCON-SC. **Preços de alimentos de festas de fim de ano variam mais de 100% em Florianópolis.** Florianópolis, 2013. Disponível em: <http://dc.clicrbs.com.br/sc/noticias/noticia/2013/12/precos-de-alimentos-de-festas-de-fim-de-ano-variam-mais-de-100-em-florianopolis-4361407.html>. Acesso em: 17 mar. 2017.

ROJO, Fernando José Grandis. **Qualidade total**: uma nova era para os supermercados**.** 1998, 89 f. TCC (Graduação) – Administração Mercadológica, Fundação Getúlio Vargas, São Paulo. Disponível em: <http://rae.fgv.br/sites/rae.fgv.br/files/artigos/10.1590_S0034-75901998000400004.pdf>. Acesso em: 18 mar. 2017.

SHETH, Jagdish N.; MITTAL, Banwari; NEWMAN, Bruce I. **Consumer behavior and beyond**. Nova York: Harcourt Brace, 1999. Disponível em: < https://goo.gl/Mh4DFk>. Acesso em: 27 mai. 2017.

SOLOMON, Michael R.. **O Comportamento do Consumidor-11ª Edição: Comprando, Possuindo e Sendo.** Porto Alegre: Bookman Editora, 2016. Disponível em: <https://goo.gl/6fnwLV>. Acesso em: 29 mar. 2017.

SPC e CNDL. **Consumo colaborativo:** 40% dos brasileiros já trocaram hotel por residência de terceiros**.** [S. l.], 2017. Disponível em: < https://goo.gl/oxy7Gy/>. Acesso em: 16 dez. 2017.

VARGIS, Leroy. **Enabling Push Notifications for Mobile Devices with Azure Notification Hubs.** [S. l.], 2017. Disponível em: < http://blog.spektrasystems.com/2017/05/10/enabling-push-notifications-for-mobile-devices-with-azure-notification-hubs//>. Acesso em: 15 nov. 2017.

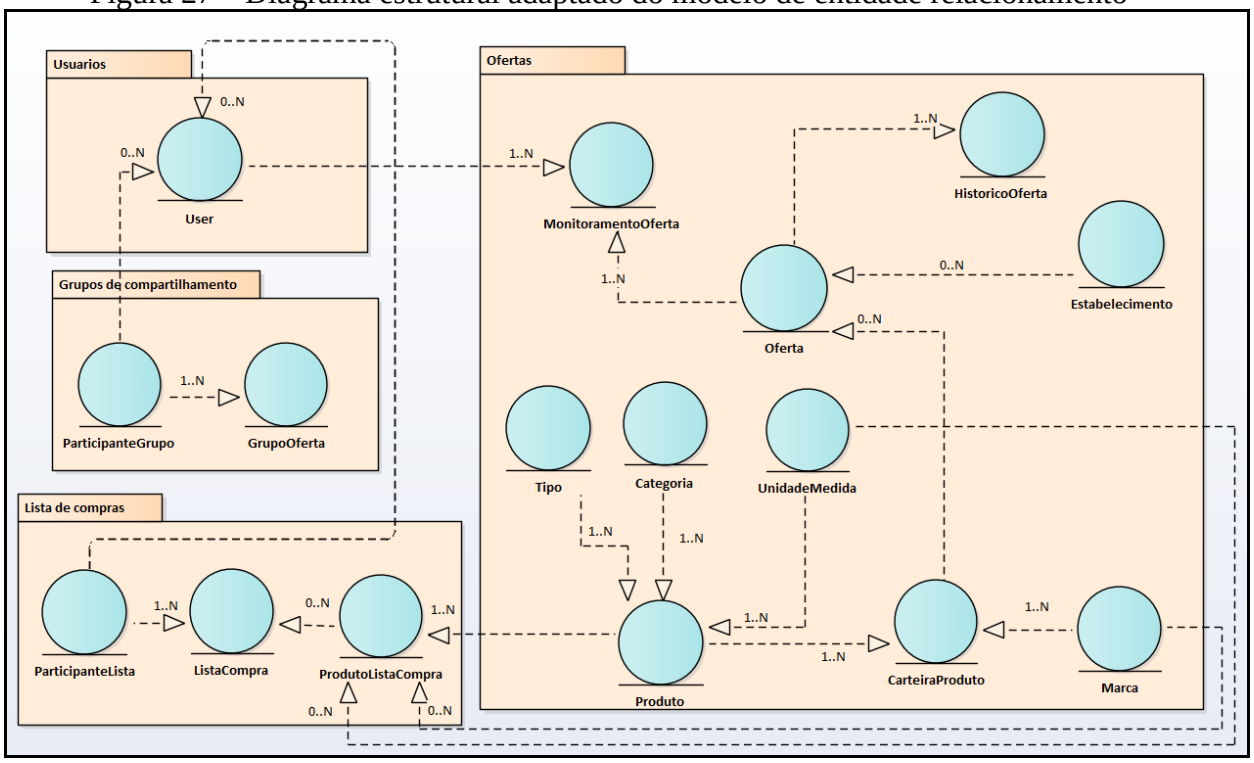
VELLUTO, Luciele. **Variação de preço de itens de inverno chega a 368%.** São Paulo, 2012. Disponível em: <http://www.estadao.com.br/blogs/jt-seu-bolso/2012/07/21/variacao-de-preco-de-itens-de-inverno-chega-a-368/>. Acesso em: 17 mar. 2017.

ZXING TEAM. **ZXing Decoder.** 2008. Disponível em: < https://zxing.org/w/decode.jspx >. Acesso em: 12 mai. 2017.

## APÊNDICE A – Modelo de Entidade Relacionamento

A base de dados do aplicativo é composto por 16 tabelas, onde podem ser analisadas em um diagrama estrutural elaborado a partir do MER da aplicação, conforme mostra a Figura 27.

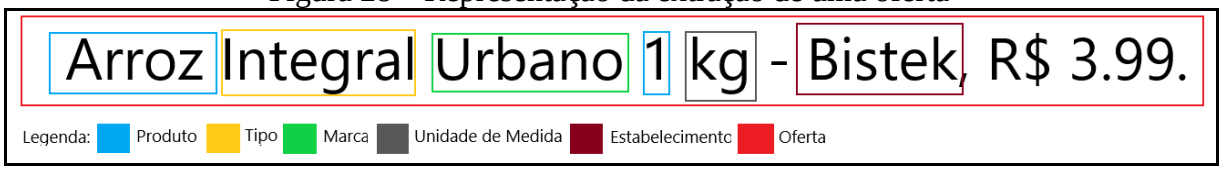
Figura 27 – Diagrama estrutural adaptado do modelo de entidade relacionamento




Fonte: elaborado pelo autor.

Todas as ofertas cadastradas pelos usuários são mantidas na tabela `Oferta` e a identificação do estabelecimento é feita através do relacionamento com a tabela `Estabelecimento`. Uma oferta está relacionada com uma carteira de produtos, através da tabela `CarteiraProduto`. A carteira de produto representa todos os produtos que uma marca pode produzir, portanto ela realiza o relacionamento *N:N* entre as tabelas `Marca` e `Produto`. O produto representa o que é comercializado nos estabelecimentos sem a vinculação de uma marca. Na tabela `Produto` é possível informar a categoria, unidade de medida e o tipo do produto, que são relacionados com as tabelas `Categoria`, `UnidadeMedida` e `Tipo` respectivamente. Ao cadastrar novas ofertas para um determinado estabelecimento as ofertas mais antigas passam a ser gravadas na tabela `HistoricoOferta`, sendo utilizadas como fonte de observação da variação de preço. A Figura 28 demonstra como é transferida a representação de uma oferta real para o nível da modelagem dos dados.

Figura 28 – Representação da extração de uma oferta



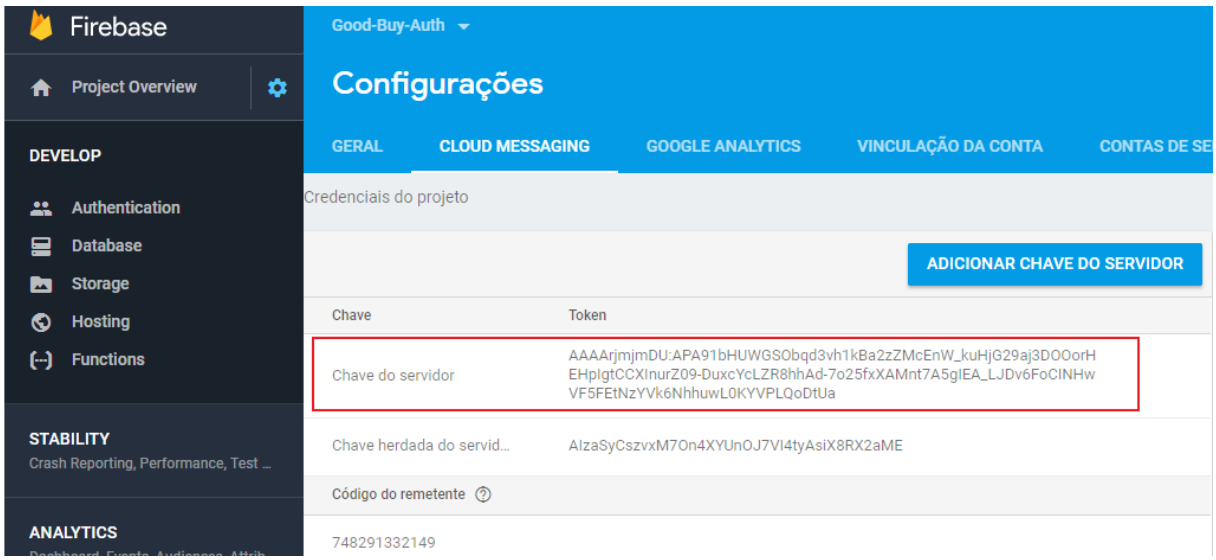

Fonte: elaborado pelo autor.

O compartilhamento da oferta pode ser realizado através de um usuário autenticado na aplicação, na qual é identificado na tabela `User` ou utilizando um dos grupos deste usuário, representado pelas tabelas `GrupoOferta` e `ParticipanteGrupo`. Sempre que for de interesse do usuário monitorar uma oferta até atingir o preço julgado ideal, a persistência deste monitoramento ficará na tabela `MonitoramentoOferta`. As listas de compras elaboradas pelos usuários serão relacionadas na tabela `ListaCompra`. Uma lista de compras poderá ser composta por muitos produtos e muitos participantes, sendo representados relacionalmente nas respectivas tabelas `ProdutoListaCompra` e `ParticipanteLista`.

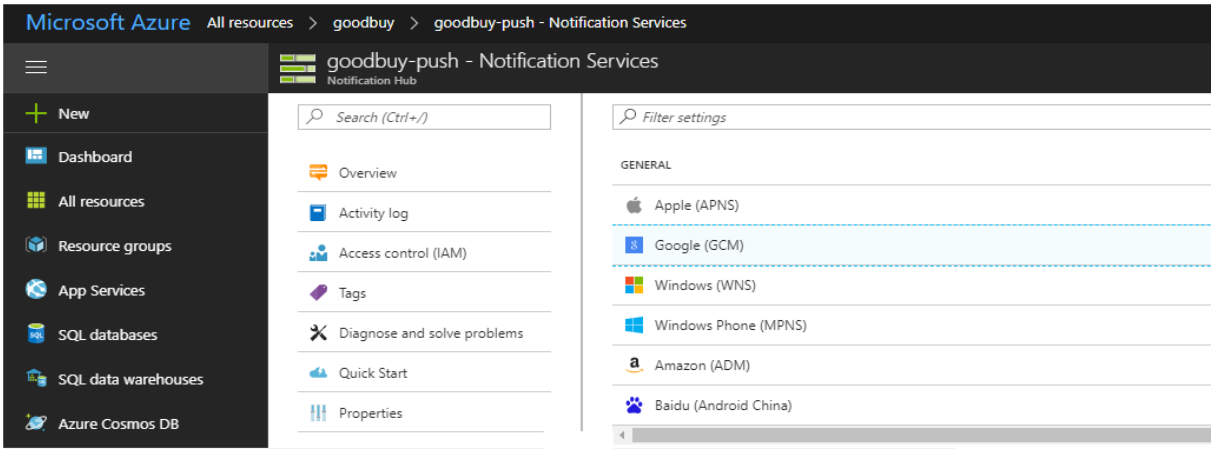
## APÊNDICE B – Registrando o aplicativo para utilização de *push notification*

A Figura 29 item (a) demonstra o *token* fornecido pelo Firebase, da Google ao registrar a aplicação para receber notificações.

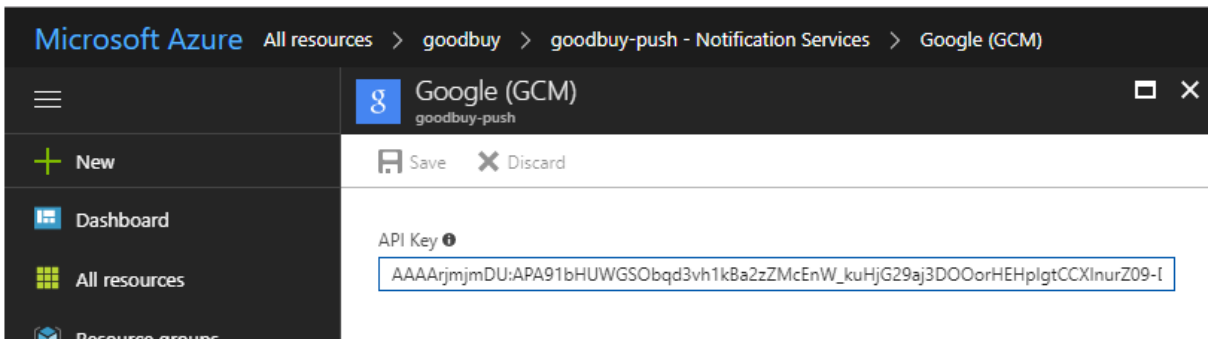
Figura 29 – Registrar aplicativo para *push notification*




(a)




(b)




(c)

Fonte: elaborado pelo autor

Após registrar o aplicativo na rede do Firebase do Google, o *token* gerado é configurado no hub de notificações do Azure, conforme demonstrado na Figura 29 item (b) e (c). O hub de notificações do Azure foi utilizado por auxiliar na abstração de envio de mensagens entre as diversas infraestruturas específicas de notificação de cada plataforma como a Apple Push Notification Service (APNS), o Windows Push Notification Services (WNS) e o Firebase Cloud Messaging (FCM). A partir da utilização dele, não foi preciso implementar rotinas de comunicação no servidor para a utilização da plataforma Android. Cada aplicativo recebe um *token* de identificação única dentro da aplicação móvel do Azure, que será utilizado posteriormente no envio da notificação.

## APÊNCIDE C – Questionário de avaliação de usabilidade e funcionalidade da aplicação

Este apêndice contém o questionário da aplicação apresentado com o objetivo de avaliar a usabilidade, funcionalidade e assertividade do aplicativo.

Quadro 23 – Questionário de perfil do usuário

**GOOD-BUY: UM PROTÓTIPO PARA COMPARTILHAMENTO DE OFERTAS OU PRODUTOS**

O Good-Buy é um aplicativo colaborativo entre os usuários com o objetivo de compartilhar e disseminar ofertas e promoções dos estabelecimentos mais próximos. Com o aplicativo você pode informar uma oferta, consultar as últimas ofertas dos estabelecimentos locais e compartilhá-las através de grupos de ofertas ou diretamente com um outro usuário. O aplicativo também permite a elaboração de listas de compras que lhe mostrará o melhor custo-benefício para a realização da compra dos seus produtos.

Gostaríamos da sua colaboração para avaliar e opinar a respeito das funcionalidades, possíveis melhorias e usabilidade em geral no aplicativo. Para isto, é necessário a avaliação do aplicativo desenvolvido. Isso nos ajudará no levantamento de possíveis melhorias tanto na experiência com o usuário como no desenvolvimento de novas funcionalidades. O formulário é composto pelo perfil do usuário e 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

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

( ) Sim ( ) Não

Você costuma pesquisar antes de realizar compras no supermercado?

( ) Sim, sempre pesquiso ( ) Pesquiso quando há tempo

( ) Não, costumo ir sempre no mesmo estabelecimento de minha preferência.

( ) Não, costumo ir sempre na loja mais próxima de casa.

Quais são os meios que você geralmente utiliza para ficar ciente de promoções nos supermercados locais?

( )Rádio ( ) Televisão ( ) Jornal ( ) Internet ( ) Panfletos

Outros:_____

Você possui o hábito de solicitar ajuda na elaboração da lista de compras de sua residência?

( ) Sim ( ) Não

No supermercado durante as compras, você tem o hábito de avisar seus amigos ou parentes a respeito de uma possível promoção do estabelecimento?

( )Sim ( )Sim

Você é membro de algum grupo de compartilhamento de promoções e vendas em alguma rede social como o Facebook ou o Whatsapp?

( ) Sim ( ) Não

Quadro 24 – Questionário de atividades do usuário

**LISTA DE ATIVIDADES DO USUÁRIO**

A seguir são apresentadas algumas atividades com o objetivo de avaliar a experiência com o aplicativo. Por gentileza, realizar o que é descrito em cada uma das questões abaixo e indicar se a atividade foi concluída com sucesso ou não.

1) Instalação do aplicativo
   Baixe o aplicativo e faça sua instalação.
   Você conseguiu fazer a instalação? Sim, não? Por quê?
   _____
   Login no aplicativo
   É necessário estar conectado à internet para realizar o login no aplicativo. Você teve algum problema para conseguiu logar no aplicativo? A escolha do tipo de login foi o Facebook ou autenticação local?
   _____
   _____

2) Cadastrar uma oferta
   pós realizar o login no aplicativo, no menu principal clique em “Informar Oferta” e simule o preenchimento de uma oferta em um determinado estabelecimento.
   Você conseguiu cadastrar uma oferta com sucesso? Sim, não? Por quê?
   _____
   _____
   Você teve alguma dúvida no preenchimento da oferta? Sim, não? Por quê?
   _____

3) Cadastrar grupos
   No menu principal do aplicativo, clique em “Grupos de ofertas” e tente criar um novo grupo clicando no botão “Novo Grupo” e preencha as informações necessárias.
   Você conseguiu cadastrar um grupo com sucesso? Sim, não? Por quê?
   _____
   _____
   Grupos Públicos
   Ainda na listagem de grupos, também existem grupos públicos, que são abertos para participação de quaisquer usuários. Digite “Questionário Good-Buy” e clique sobre o grupo para solicitar a participação através do botão “Participar”.
   Você conseguiu participar do grupo público com sucesso? Sim, não? Por quê?
   _____
   _____

4) Consultar ofertas
   No menu principal, clique em “Ofertas do dia”. A oferta previamente cadastrada deverá ser visualizada na listagem de ofertas.
   Você conseguiu visualizar a listagem de ofertas e a sua oferta cadastrada anteriormente? Sim, não? Por quê?
   _____
   _____

Confiabilidade da oferta
Para alertar sobre a autenticidade de uma oferta aos demais usuários, avalie positivamente e negativamente algumas ofertas de seu interesse na listagem de ofertas.
Você conseguiu avaliar as ofertas selecionadas? Sim, não? Por quê?

_____
_____

Compartilhar Oferta
Na tela de consultas, selecione a opção "Compartilhar" de alguma oferta. Compartilhe a oferta com o seu usuário ou algum conhecido ou então compartilhe a oferta com um de seus grupos. Caso preferir, crie um grupo e me adicione como participante. Assim que eu entrar no grupo, compartilharei uma oferta para você receber a notificação. Você conseguiu compartilhar a oferta e enviar/receber a notificação com sucesso? Qual foi o tipo de compartilhamento e o destinatário da oferta?

_____
_____

5) Cadastrar lista de compras
No menu principal do aplicativo, clique em "Suas Listas" e tente criar uma nova lista de compras clicando no botão "Nova Lista". Cadastre alguns produtos disponíveis que seja de seu interesse e clique em "Salvar"
Você conseguiu cadastrar uma lista de compras com sucesso? Sim, não? Por quê?

_____
_____

Simular compras
Selecione a lista de compras recém cadastrada e selecione a opção "Simular compra".
Você conseguiu simular o preço total da lista de compra com sucesso? Sim, não? Por quê?

_____
_____

Quadro 25 – Questionário de usabilidade

## QUESTIONÁRIO DE AVALIAÇÃO DO APLICATIVO

Após a utilização do aplicativo, você está convidado a responder um questionário para avaliá-lo. As questões devem ser respondidas na tabela abaixo dando o seu parecer a respeito do aplicativo com base na utilização e realização dos testes. Você deve responder preenchendo somente 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 velocidade da sincronização de ofertas é rápida | | | | | |
| É fácil navegar entre as telas do aplicativo | | | | | |
| O aplicativo em algum momento parou inesperadamente | | | | | |
| A aparência do aplicativo é boa (cores e ícones) | | | | | |
| O tempo de notificação de compartilhamento de ofertas ocorreu dentro do prazo esperado | | | | | |

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

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

Você achou importante existir grupos com interesses de ofertas em comuns para a divulgação de promoções?
( ) Sim ( ) Não

Você enfrentou problemas de lentidão durante a utilização do aplicativo?
( ) Sim ( ) Não

Você conseguiu visualizar sua oferta?
( ) Sim ( ) Não

Você acha interessante a funcionalidade de monitorar ofertas até um determinado preço disponível no aplicativo?
( ) Sim ( ) Não

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

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

Você confiaria nessa simulação de compras realizada a ponto de se dirigir até o estabelecimento sugerido, mesmo que ele não fosse o estabelecimento que você compra com frequência ou o mais próximo de sua residência?
( ) Sim ( ) Não

Você acha interessante a funcionalidade de elaborar listas de compras com a participação de conhecidos ou familiares?
( ) Sim ( ) Não

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

_____
_____
_____