



**UFOP**

Universidade Federal  
de Ouro Preto

**Universidade Federal de Ouro Preto  
Instituto de Ciências Exatas e Aplicadas  
Departamento de Computação e Sistemas**

# **SocialMarket: Aplicativo Colaborativo Para Auxiliar nas Compras em Supermercados**

**Amanda Fagundes de Paula**

## **TRABALHO DE CONCLUSÃO DE CURSO**

**ORIENTAÇÃO:**

Vicente José Peixoto de Amorim

**COORIENTAÇÃO:**

Leonardo Vieira dos Santos Reis

**Fevereiro, 2018  
João Monlevade–MG**

**Amanda Fagundes de Paula**

# **SocialMarket: Aplicativo Colaborativo Para Auxiliar nas Compras em Supermercados**

Orientador: Vicente José Peixoto de Amorim

Coorientador: Leonardo Vieira dos Santos Reis

Monografia apresentada ao curso de Sistemas de Informação do Instituto de Ciências Exatas e Aplicadas, da Universidade Federal de Ouro Preto, como requisito parcial para aprovação na Disciplina “Trabalho de Conclusão de Curso II”.

**Universidade Federal de Ouro Preto**

**João Monlevade**

**Fevereiro de 2018**

# FOLHA DE APROVAÇÃO DA BANCA EXAMINADORA

## SocialMarket: Aplicativo Colaborativo Para Auxiliar nas Compras em Supermercados

Amanda Fagundes de Paula

Monografia apresentada ao Instituto de Ciências Exatas e Aplicadas da Universidade Federal de Ouro Preto como requisito parcial da disciplina CSI499 – Trabalho de Conclusão de Curso II do curso de Bacharelado em Sistemas de Informação e aprovada pela Banca Examinadora abaixo assinada:



---

Vicente José Peixoto de Amorim  
Mestre em Ciência da Computação  
DECSI – UFOP



---

Leonardo Vieira dos Santos Reis  
Doutor em Ciência da Computação  
DCC – UFJF



---

Igor Mizetti Pereira  
Mestre em Ciência da Computação  
Examinador  
DECSI – UFOP



---

George Henrique Godim da Fonseca  
Doutor em Engenharia Elétrica  
Examinador  
DECSI – UFOP


João Monlevade, 16 de fevereiro de 2018

## ATA DE DEFESA

No dia 16 do mês de Fevereiro de 2018, às 14:30 horas, na sala C304 do Instituto de Ciências Exatas e Aplicadas, foi realizada a defesa de Monografia pelo(a) aluno(a) **Amanda Fagundes de Paula**, sendo a Comissão Examinadora constituída pelos professores: Vicente José Peixoto de Amorim, Leonardo Vieira dos Santos Reis, Igor Muzetti Pereira, George Henrique Godim da Fonseca. O(a) candidato(a) apresentou a monografia intitulada: "**SocialMarket: Aplicativo Colaborativo Para Auxiliar nas Compras em Supermercados**". A comissão examinadora deliberou, por unanimidade, pela aprovação do candidato, com nota 9,5 (nove vírgula cinco), concedendo-lhe o prazo de 15 dias para incorporação das alterações sugeridas ao texto final. Na forma regulamentar, foi lavrada a presente ata que é assinada pelos membros da Comissão Examinadora e pelo(a) graduando(a).



Vicente José Peixoto de Amorim  
Mestre em Ciência da Computação  
DECSI – UFOP



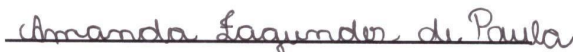
Leonardo Vieira dos Santos Reis  
Doutor em Ciência da Computação  
DCC – UFJF



Igor Muzetti Pereira  
Mestre em Ciência da Computação  
Examinador(a)  
DECSI – UFOP



George Henrique Godim da Fonseca  
Doutor em Engenharia Elétrica  
Examinador(a)  
DECSI – UFOP



Amanda Fagundes de Paula

## TERMO DE RESPONSABILIDADE

Eu, **Amanda Fagundes de Paula** declaro que o texto do trabalho de conclusão de curso intitulado “***SocialMarket: Aplicativo Colaborativo Para Auxiliar nas Compras em Supermercados***” é de minha inteira responsabilidade e que não há utilização de texto, material fotográfico, código fonte de programa ou qualquer outro material pertencente a terceiros sem as devidas referências ou consentimento dos respectivos autores.

João Monlevade, 16 de fevereiro de 2018

Amanda Fagundes de Paula  
Amanda Fagundes de Paula

*Este trabalho é dedicado aos meus pais, por serem essenciais em minha vida, me dando  
todo o apoio e incentivo necessários.*

# Agradecimentos

Agradeço primeiramente à minha família, meus pais, Rita e José Carlos, e meu irmão Rafael que sempre estiveram comigo e nunca permitiram que eu fraquejasse e me incentivam diariamente a progredir.

Aos meus amigos, incluindo aqueles que, mesmo distantes (Ana Rita e Luiz Fernando), me deram todo o apoio e carinho que precisava para seguir em frente e não desistir dos meus sonhos.

Agradeço também aos meus amigos da salinha que estiveram comigo desde o início e nunca hesitaram em me ajudar quando precisei.

Agradeço aos meus orientadores, Vicente Amorim e Leonardo Reis, pela paciência, suporte e conhecimento compartilhado.

*“Tudo o que temos de decidir é o que fazer com o tempo que nos é dado.”*

— Gandalf,  
*em: O Senhor dos Anéis: A Sociedade do Anel.*



# Resumo

A Tecnologia da Informação tem desempenhado um papel cada vez mais importante na sociedade, transformando a forma como os indivíduos e organizações interagem com o meio e tomam decisões, principalmente no que diz respeito às escolhas relacionadas ao consumo. Um aspecto importante que se relaciona a esse fato são as compras realizadas em supermercados, que fazem parte da rotina dos cidadãos. Uma característica inerente à realidade do comércio é que, para um mesmo produto, podem existir diversas variações de preço em diferentes estabelecimentos. Dessa forma, um indivíduo que deseja adquirir o produto pelo menor preço deve visitar todas as lojas de uma determinada região para verificar onde é mais vantajoso realizar a aquisição do produto. No entanto, essa prática se torna inviável para a maioria da população, frente aos custos que ela ocasiona, em termos de deslocamento e do tempo consumido. Portanto, a concepção de ferramentas que auxiliem os consumidores no momento de cotação de preços em estabelecimentos físicos se mostra relevante. Neste contexto, foi proposto o SocialMarket, um aplicativo colaborativo para auxiliar nas compras em supermercados. O aplicativo possibilita que seus usuários consultem preços de produtos nos estabelecimentos próximos, além de permitir a criação de listas de compras nas quais poderão ser adicionados produtos, de forma a informar ao usuário em qual estabelecimento é mais vantajoso realizar a compra dos itens.

**Palavras-chaves:** aplicativo colaborativo. comércio eletrônico. cotação de preços. consumo.

# Abstract

Information Technology has been playing an increasingly important role in society, changing the way individuals and organizations interact with the environment and make decisions, mainly regarding choices related to consumption. An important aspect, related to this fact are purchases carried out at supermarkets, that are part of citizens' routine. An inherent feature of commerce reality is that for the same product there might be several price variations in different places. Therefore, an individual who wishes to acquire a product for the lowest price has to visit all stores of a particular region to determine where it is more advantageous to purchase the product. However, this practice becomes impossible for the majority of the population, given the costs which it generates in terms of displacement and time consumed. Consequently, the design of tools that help consumers in their shopping time is relevant. In this context, SocialMarket was proposed as a collaborative application to help with supermarkets purchases. The application allows its users to consult products prices in nearby establishments, it also allows the creation of shopping lists in which products can be added, in order to inform in which stores are more beneficial to buy the items.

**Key-words:** collaborative application. price quotation. e-commerce. consumer

# Lista de ilustrações

Figura 1 – Arquitetura da Aplicação . . . . .	26
Figura 2 – Fluxo de comunicação entre componentes do sistema . . . . .	27
Figura 3 – Diagrama entidade-relacionamento . . . . .	28
Figura 4 – Cadastro . . . . .	31
Figura 5 – Tela de <i>Login</i> . . . . .	31
Figura 6 – Listas do usuário . . . . .	33
Figura 7 – Produtos da Lista . . . . .	33
Figura 8 – Menu de configurações gerais . . . . .	33
Figura 9 – Menu de configurações específicas . . . . .	33
Figura 10 – Tela de busca . . . . .	35
Figura 11 – Lista de Ofertas . . . . .	36
Figura 12 – <i>Dialog</i> para adição produto à lista . . . . .	36
Figura 13 – Seleção do tipo de produto a ser cadastrado . . . . .	37
Figura 14 – Cadastro de Produto com Código de Barras . . . . .	37
Figura 15 – Cadastro de preço para Produto sem Código de Barras . . . . .	37
Figura 16 – Momento das compras . . . . .	39
Figura 17 – Produtos que não possuem preço cadastrado . . . . .	39

# Lista de tabelas

Tabela 1 – Requisitos funcionais do aplicativo . . . . .	21
Tabela 2 – Requisitos não funcionais do aplicativo . . . . .	22

# Lista de abreviaturas e siglas

**IDE** *Integrated Development Environment*

**MVC** Modelo-Visão-Controlador

**SGBD** Sistema de Gerenciamento de Banco de Dados

**REST** *Representational State Transfer*

**HTTP** *Hypertext Transfer Protocol*

**CRUD** *Create-Read-Update-Delete*

**JSON** *JavaScript Object Notation*

**XML** *eXtensible Markup Language*

**ER** Entidade-Relacionamento

# Sumário

<b>1</b>	<b>INTRODUÇÃO</b>	<b>14</b>
<b>2</b>	<b>FERRAMENTAS COLABORATIVAS PARA COMPRAS EM SUPERMERCADOS</b>	<b>16</b>
2.1	SoftList	16
2.2	Meus Preços	17
2.3	eMercado	17
2.4	Outros	18
<b>3</b>	<b>IMPLEMENTAÇÃO</b>	<b>20</b>
<b>4</b>	<b>RESULTADOS</b>	<b>25</b>
<b>4.1</b>	<b>Sistema</b>	<b>25</b>
4.1.1	Visão Geral	25
4.1.2	Aplicação Móvel	29
4.1.3	Memória <i>cache</i>	29
4.1.4	Servidor	30
<b>4.2</b>	<b>O Aplicativo SocialMarket</b>	<b>31</b>
4.2.1	Cadastro e Autenticação	31
4.2.2	Listas de Compras	32
4.2.3	Configurações de Busca	33
4.2.4	Mecanismos de Busca	34
4.2.5	Ofertas	35
4.2.6	Cadastro de Produtos e Preços	36
4.2.7	Compras	38
<b>4.3</b>	<b>Considerações</b>	<b>39</b>
<b>5</b>	<b>CONCLUSÃO</b>	<b>40</b>
<b>5.1</b>	<b>Trabalhos Futuros</b>	<b>41</b>
	<b>REFERÊNCIAS</b>	<b>42</b>

# 1 Introdução

O ato de consumir faz parte da rotina humana, seja para satisfazer necessidades básicas ou de realização pessoal. É fato que sempre estamos em busca daquilo que satisfaça a nossa vontade. Atualmente, o desejo de adquirir bens é impulsionado pela tecnologia da informação, que permite a disponibilização de grande quantidade de dados aos quais temos acesso e, diariamente, temos de analisar esses dados e fazer escolhas de consumo.

A tecnologia da informação tem desempenhado um papel cada vez mais importante na sociedade, transformando a forma como os indivíduos e organizações interagem com o meio e tomam decisões. A qualquer instante e lugar é possível buscar e acessar um grande volume de informações via dispositivos móveis, possibilitando fazer uma análise mais precisa e tomar decisões mais acuradas.

Em decorrência disso, o setor supermercadista tem passado por diversas transformações, ocasionadas principalmente no que diz respeito ao comportamento do consumidor, que busca mais conveniência e dispõe de menos tempo (COSTA et al., 2007). Atualmente, uma das estratégias utilizadas pelos consumidores é a pesquisa de preço, processo no qual são analisadas várias ofertas e, ao final, é selecionada aquela cujo custo-benefício seja mais vantajoso para o consumidor.

Pedrozo (2016) aponta que a pesquisa de preço pode gerar uma economia significativa para aqueles que a praticam. No entanto, essa prática envolve um grande dispêndio de tempo, uma vez que para realizá-la em estabelecimentos físicos o consumidor deve se deslocar entre as lojas e registrar as variações de preço – para, somente após visitar todas, poder analisar em qual estabelecimento é mais vantajoso comprar um determinado produto. Este método pode ser facilitado com o uso de tecnologias, como mecanismos de buscas, que auxiliem no processo de cotação de preço.

No entanto, apesar da vasta quantidade de informações disponíveis na internet, nem sempre aquelas necessárias para uma situação específica estão concentradas em um lugar de fácil acesso ou estão, até mesmo, indisponíveis. Em outras palavras, existe uma vasta possibilidade de integração dos objetos da realidade objetiva com a internet. Portanto, existe demanda e oportunidade para a concepção e desenvolvimento de aplicativos que obtenham, filtrem, processem e forneçam informações seguras e de qualidade para auxiliar na tomada de decisão dos indivíduos e organizações.

O objetivo deste trabalho é finalizar o desenvolvimento do SocialMarket, um aplicativo colaborativo para dispositivos móveis cuja finalidade é auxiliar os usuários em suas compras nos supermercados, fornecendo informações dos preços dos produtos nos diversos estabelecimentos de uma região. O aplicativo é colaborativo no sentido de que as

informações de preços e atualizações serão obtidas e atualizadas a partir das compras e das informações fornecidas pelos próprios usuários do aplicativo.

Deste modo, de forma a cumprir com o objetivo proposto, foram realizados ajustes no aplicativo, juntamente com a implementação do servidor responsável por gerenciar e manter os dados gerados pelos usuários. A partir disso, o aplicativo foi finalizado, sendo capaz de informar ao usuário em qual estabelecimento é possível comprar um determinado produto (ou uma lista de produtos) pelo menor preço. Para tanto, é permitido ao usuário cadastrar informações relacionadas aos produtos e seus respectivos valores, que podem ser atualizados pelas outras pessoas que utilizam o aplicativo. O SocialMarket conta com diversas funcionalidades, como a busca de produtos a partir de um leitor de código de barras, a possibilidade de adicionar produtos e estabelecimentos aos favoritos, de forma a facilitar o acesso às informações relacionadas a esses itens, dentre outras que serão descritas neste trabalho.

O trabalho está organizado em cinco capítulos. O Capítulo 2 apresenta alguns aplicativos que possuem proposta semelhante a do SocialMarket e descreve suas principais funcionalidades. No Capítulo 3 são descritas as atividades realizadas no desenvolvimento do aplicativo e do servidor, bem como o processo de integração entre os dois. No Capítulo 4 são apresentados os resultados do trabalho. No Capítulo 5 são relatadas as considerações finais e recomendações para trabalhos futuros.



## 2 Ferramentas Colaborativas Para Compras em Supermercados

Com o avanço da Tecnologia da Informação e o aumento da quantidade de informações disponíveis, os consumidores têm de lidar com uma grande quantidade de dados nos quais devem se basear para fazer escolhas de consumo. Essas escolhas fazem parte do processo de decisão de compra – conceito que descreve as etapas realizadas por um indivíduo na busca por um produto ou serviço. Segundo Schiffman and Kanuk (2009), um consumidor está em posição de tomar uma decisão quando se depara com situações em que existam alternativas a serem analisadas e, a partir disso, deve ser feita uma escolha – como optar por uma marca ou efetuar ou não uma compra.

Nesse contexto, ferramentas que gerenciem informações relacionadas a oferta de bens ou serviços se colocam como uma grande alternativa para aqueles que buscam auxílio no momento de compra. No entanto, apesar de a pesquisa de preços na internet se encontrar bastante desenvolvida, ao considerarmos o cenário de compra em estabelecimentos físicos, a aquisição dessas informações se coloca como sendo um desafio, visto que não são disponibilizadas na *web*.

Um meio de contornar esse problema é a utilização dos conceitos de Inteligência Coletiva, que diz respeito ao conhecimento que pode ser adquirido a partir da colaboração entre indivíduos que compartilham informações em um meio. “É uma inteligência distribuída por toda parte, na qual todo o saber está na humanidade, já que, ninguém sabe tudo, porém todos sabem alguma coisa” (LéVY, 2007, p. 212).

Baseando-se nisso, foram desenvolvidos diversos aplicativos que buscam ajudar o consumidor no processo de compra em estabelecimentos físicos. Tais aplicativos são descritos nas seções subsequentes.

### 2.1 SoftList

O SoftList <sup>1</sup> é um aplicativo com filosofia semelhante a do SocialMarket e, apesar de terem sido lançados outros aplicativos mais completos, é indispensável mencionar suas principais características, uma vez que foi o aplicativo que mais se assemelhou ao SocialMarket no início de seu desenvolvimento.

O SoftList é um aplicativo colaborativo, em que seus usuários podem consultar e cadastrar informações de preço de produtos e, a partir dos produtos cadastrados é possível

---

<sup>1</sup> <[www.softlist.aptoide.pt](http://www.softlist.aptoide.pt)>

criar listas de compras e consultar seus preços nos estabelecimentos próximos. Nas listas criadas pelo usuário, ao lado de cada produto, é informado seu preço médio juntamente com o maior e menor valor para aquele produto. O aplicativo dispõe de um leitor de código de barras que, a partir do código lido, permite que o usuário cadastre um produto ou consulte seus preços.

Além da busca pelo *scanner*, o aplicativo permite que as buscas sejam realizadas pelo nome do produto, exibindo as ofertas disponíveis na região com base na localização do usuário. O SoftList conta também com a funcionalidade de “Compras”, em que o usuário pode adicionar produtos que pretende comprar e consultar o valor total da compra.

Para uma melhor precisão, o aplicativo permite que o usuário configure o raio máximo de alcance das buscas, além de oferecer a opção de adicionar estabelecimentos aos favoritos de maneira a priorizá-los no momento em que as ofertas são exibidas.

No entanto, ao atualizar ou cadastrar um novo preço, este não é disponibilizado aos outros usuários em tempo hábil, o que faz com que o aplicativo deixe de fornecer informações acuradas. Com isso, apesar de ser bem visto pela maioria de seus usuários, o conceito de colaboratividade do aplicativo se perde um pouco.

## 2.2 Meus Preços

O Meus Preços <sup>2</sup> é outro aplicativo colaborativo que auxilia seus usuários nas decisões de compras. Diferente do SoftList, é sincronizado com a Nota Fiscal Paulista, de modo que as informações exibidas estejam sempre atualizadas. No entanto, algumas de suas principais funcionalidades ficam restritas às pessoas que possuem cadastro no programa da Nota Fiscal Paulista e as informações exibidas são referentes apenas aos estabelecimentos do estado de São Paulo. De forma geral, o aplicativo permite que seus usuários consultem informações acerca de preços de produtos, exibindo seu preço médio e uma lista com produtos semelhantes. Para visualizar o valor do produto em diferentes estabelecimentos, o Meus Preços conta com um mapa, no qual são exibidos os preços do produto na região. É possível ainda acompanhar a evolução do preço de um determinado produto a partir do histórico

## 2.3 eMercado

Atualmente, o aplicativo que mais se assemelha ao SocialMarket é o eMercado <sup>3</sup> que, de forma semelhante aos outros, apresenta informações relacionadas aos produtos vendidos nos estabelecimentos próximos aos usuários. No entanto, além de contar com as

---

<sup>2</sup> <<http://meusprecos.com.br>>

<sup>3</sup> <<http://emercadoapp.com>>

informações cedidas pelos próprios usuários, o eMercado possui uma equipe que visita os estabelecimentos e atualiza o preço dos produtos cadastrados no aplicativo.

O cadastro e atualização das informações dos produtos pode ser feito manualmente ou através da leitura do *QRCode* presente em notas fiscais. O aplicativo permite também que seus usuários criem e gerenciem listas de compras além de contar com a função de controle, onde os usuários podem acessar as listas já criadas, com o histórico de preços, data e local no qual a compra foi realizada.

## 2.4 Outros

São diversas as aplicações com o mesmo objetivo do SocialMarket, no entanto, a maioria não possui o conceito de colaboratividade implementado. São exemplos disso aplicativos como Bring!<sup>4</sup> que permite a criação e compartilhamento de listas de compras e adição de produtos às mesmas. Nesse aplicativo, todas as informações dos produtos são inseridas pelo próprio usuário e não são compartilhadas, de modo que ao cadastrar um novo produto, somente tem acesso a esses dados aquele que realizou o cadastro. Alguns outros aplicativos como Out of Milk<sup>5</sup> e Our Groceries<sup>6</sup> permitem ainda que o usuário crie, além das listas de compras, listas de despensa e de receitas, permitindo que se tenha controle dos produtos que possui em casa e execute receitas com mais facilidade. O Our Groceries conta ainda com uma versão paga, na qual os usuários podem adicionar fotos aos itens e adicioná-los à lista a partir de um leitor de código de barras.

No geral, pode-se observar que os aplicativos analisados não são capazes de informar qual é o melhor estabelecimento para realizar a compra a partir de uma lista de produtos. Com isso, o usuário fica encarregado de analisar os preços e tomar decisões acerca de onde comprar sem uma ferramenta que o auxilie no processo. Um outro aspecto observado nos aplicativos colaborativos, com exceção do “Meus Preços”, consiste na dificuldade em cadastrar um produto ou preço e disponibilizá-los aos outros usuários. Acredita-se que tenha sido desenvolvida uma política de segurança de dados a fim de barrar informações fraudulentas e usuários mal intencionados. No entanto, a partir do momento em que os usuários ficam impossibilitados de compartilhar informações, a eficiência dessas políticas acaba se perdendo juntamente com o conceito de colaboratividade.

O SocialMarket se destaca justamente pelo fato de informar ao usuário qual é o melhor estabelecimento para se comprar um determinado produto, além de permitir a criação de listas de compras nas quais podem ser adicionados diversos itens e informar em qual/quais estabelecimentos a compra pode ser efetuada pelo menor preço. O aplicativo permite também que o usuário visualize o valor total de sua compra, exibindo-o indivi-

---

<sup>4</sup> <<https://www.getbring.com>>

<sup>5</sup> <<https://www.outofmilk.com/>>

<sup>6</sup> <<https://www.ourgroceries.com>>

dualmente para os estabelecimento que satisfaçam seus critérios de busca, juntamente com o valor total caso o usuário opte por visitar vários estabelecimentos. Neste caso, é considerado, para cada produto, o menor preço cadastrado para os supermercados que satisfaçam os parâmetros definidos pelo usuário no menu de configuração de buscas.

## 3 Implementação

Neste capítulo serão apresentados os resultados obtidos e decisões de projeto mais relevantes, englobando as tarefas realizadas desde o projeto de iniciação científica, que foram divididos em quatro etapas:

1. Requisitos funcionais e não funcionais;
2. Desenvolvimento do aplicativo;
3. Implementação do servidor;
4. Comunicação do aplicativo com o servidor.

O início de toda atividade de desenvolvimento de *software* se dá a partir da análise e levantamento de requisitos. Segundo Pfleeger (2004), um requisito é uma característica do sistema ou a descrição de algo que o sistema é capaz de realizar para atingir os seus objetivos. Os requisitos podem ser divididos em dois grupos: (i) os requisitos funcionais, que definem o que o sistema fará através da descrição das funcionalidades que serão implementadas; e (ii) os requisitos não funcionais, que tratam de questões relacionadas às restrições e desempenho do *software*. Dessa forma, no início do desenvolvimento deste trabalho foram listados os seguintes requisitos funcionais, separados por categorias:

CATEGORIAS	REQUISITOS
Autenticação e cadastro	<ul style="list-style-type: none"><li>• Cadastrar usuários</li><li>• Autenticar usuários</li></ul>
Gerenciamento de listas	<ul style="list-style-type: none"><li>• Criar listas</li><li>• Editar listas</li><li>• Excluir listas</li><li>• Acessar listas</li><li>• Duplicar lista</li><li>• Buscar produto na lista de compras</li></ul>

Gerenciamento de produtos	<ul style="list-style-type: none"><li>• Cadastrar produtos</li><li>• Editar informações dos produtos</li><li>• Buscar produto (por nome ou código de barras)</li><li>• Adicionar produtos às listas</li><li>• Excluir produtos das listas</li><li>• Visualizar o preço médio do produto</li><li>• Consultar histórico de pesquisa</li></ul>
Gerenciamento de estabelecimentos	<ul style="list-style-type: none"><li>• Cadastrar estabelecimentos</li><li>• Editar informações dos estabelecimentos</li><li>• Adicionar estabelecimento aos favoritos</li></ul>
Suporte ao usuário	<ul style="list-style-type: none"><li>• Recuperar de senha</li><li>• Sugerir melhor local para compras: toda lista de compras em um único estabelecimento ou em vários estabelecimentos</li><li>• Alterar o alcance da busca por estabelecimentos</li><li>• Sugerir o aplicativo para amigos através de e-mail e redes sociais</li><li>• Selecionar itens que já foram comprados (<i>checkbox</i>)</li><li>• Visualizar valor total da compra a partir dos itens selecionados</li><li>• Ordenar resultado da busca por ofertas (pelo preço, proximidade e atualização de preço mais recente)</li><li>• Pontuar usuários de acordo com a confiabilidade das informações enviadas e pelo <i>feedback</i> de outros usuários</li><li>• Acessar tutorial explicando como usar o aplicativo</li><li>• Avaliar o aplicativo na Play Store</li><li>• Entrar em contato com a equipe de desenvolvedores</li><li>• Visualizar informações sobre o aplicativo</li></ul>

Tabela 1 – Requisitos funcionais do aplicativo

Pelo fato de os requisitos não funcionais abrangerem o sistema como um todo, não

foi necessário separá-los em grupos:

REQUISITOS
<ul style="list-style-type: none"> <li>• O sistema deve ser escalável</li> <li>• O aplicativo deve ser fácil de utilizar</li> <li>• O aplicativo deve possuir uma <i>interface</i> amigável</li> <li>• Apenas usuários cadastrados poderão acessar o aplicativo</li> </ul>

Tabela 2 – Requisitos não funcionais do aplicativo

A partir da definição dos requisitos descritos nas Tabelas 1 e 2, foram definidas todas as funcionalidades que o aplicativo deveria possuir. Tais funcionalidades serviram como ponto de partida para o desenvolvimento do projeto e, a partir disso, foi criado um protótipo do aplicativo como forma de validar a relevância das funcionalidades definidas na etapa anterior e fazer modificações, se necessário. Um esboço do que viria a ser a aplicação foi criado a partir do *Fluid UI* <sup>1</sup>, uma ferramenta de prototipagem *online* que permite a construção de *layouts* para diversas plataformas.

Baseando-se nos requisitos definidos e no protótipo criado, o desenvolvimento do *SocialMarket* foi de fato iniciado. O aplicativo foi desenvolvido para dispositivos *Android* e toda sua construção foi realizada no *Android Studio*, o Ambiente de Desenvolvimento Integrado – do inglês, *Integrated Development Environment (IDE)* – da Google.

Primeiramente, foi desenvolvida uma versão *offline* do aplicativo como forma de validar suas funcionalidades. Essa versão incluía um banco de dados interno que foi desenvolvido a partir do *SQLite*, um Sistema de Gerenciamento de Banco de Dados (SGBD) escrito em Linguagem C, que permite que o banco de dados seja entregue junto com a aplicação. Dessa forma, o aplicativo eliminava a necessidade de ter um servidor externo para que pudesse ser utilizado. No entanto, o conceito de colaboratividade não estava presente nessa versão, visto que, os dados cadastrados e atualizados por um usuário não eram compartilhados com outros.

Por se tratar da continuação de um projeto de iniciação científica, o desenvolvimento deste trabalho se inicia na etapa de finalização da versão *offline* do aplicativo. Nesse momento, foram feitas algumas melhorias no *layout* do aplicativo, de forma a proporcionar uma melhor experiência para o usuário. Posteriormente, na etapa de testes e correções, foi possível identificar algumas falhas através da realização de testes e, a partir disso, foram aplicadas as devidas correções.

A partir disso, foi iniciado desenvolvimento do servidor, que ficaria responsável por

<sup>1</sup> <[www.fluidui.com](http://www.fluidui.com)>

gerenciar o banco de dados e interpretar as requisições dos usuários. O banco de dados interno passou a ser utilizado como *cache* da aplicação e é responsável por manter dados das consultas realizadas pelo usuário, evitando requisições desnecessárias ao servidor. Para a implementação do servidor, foram consideradas duas abordagens:

- A criação de um servidor Java *RESTful*, que faz uso do modelo arquitetural *Representational State Transfer* (**REST**).
- A utilização da linguagem Ruby e do *framework* Ruby on Rails para construção do servidor.

O servidor Java faria uso do estilo arquitetural **REST**, uma abstração da arquitetura da *World Wide Web*, que segue o modelo cliente-servidor. **REST** se baseia no *Hypertext Transfer Protocol*, o **HTTP**. Este funciona como um protocolo de requisição-resposta e permite que o cliente faça solicitações ao servidor através do envio de mensagens padronizadas. Este protocolo define oito métodos de solicitação (*GET*, *HEAD*, *POST*, *PUT*, *DELETE*, *TRACE*, *OPTIONS* e *CONNECT*) que devem ser especificados no corpo da mensagem, permitindo que o servidor seja capaz de identificar qual ação deverá realizar. A principal vantagem dessa abordagem é a escalabilidade e o desempenho que a linguagem traz ao lidar com operações complexas.

O servidor Rails seria desenvolvido utilizando a linguagem Ruby e o *framework* Ruby on Rails (RoR), também chamado de Rails. Este *framework* segue o padrão Modelo-Visão-Controlador (**MVC**), portanto cada entidade da aplicação possui modelo, visão e controle bem definidos. Nessa abordagem, as requisições são feitas através de rotas, que consistem em *URLs* que especificam a ação que dever ser tomada por um controlador específico. Essas rotas podem seguir quatro modelos, inclusive o *RESTful* e fica a cargo do desenvolvedor escolher as que melhor se adequam ao seu contexto. Uma das principais vantagens na utilização do *framework* é a agilidade que este traz para o desenvolvimento e a *syntax sugar* (açúcar sintático) presente no Ruby, que é uma característica das linguagens de programação que apresentam construções sintáticas mais fáceis de serem lidas, trazendo uma maior legibilidade ao código.

Ambas abordagens tratam a interação do cliente e servidor de forma parecida e o que as diferenciam são as linguagens e plataformas utilizadas para o desenvolvimento. Pelo fato de, neste trabalho, a maioria das requisições consistirem em consultas simples ao servidor, o *framework* Rails foi escolhido como ferramenta para sua construção. Outro aspecto importante para a decisão é a agilidade que a linguagem e o *framework* trazem para o desenvolvimento.

O **SGBD** utilizado na aplicação é o PostgreSQL. As tabelas do banco de dados foram criadas e modificadas a partir de *migrations*, um recurso do *framework* Rails que



permite a criação e alterações no modelo de forma otimizada, uma vez que permite a escrita dessas ações na linguagem Ruby. Os objetos do banco de dados foram criados de acordo com as convenções do Rails, que define algumas regras a respeito do nome das tabelas e seus atributos. Foram criadas um total de oito tabelas, das quais cinco representam as principais entidades do sistema e seus relacionamentos (*estabelecimentos*, *listas*, *precos*, *produtos* e *usuarios*). As demais tabelas (*estabelecimentos\_usuarios*, *listas\_produtos* e *produtos\_usuarios*), chamadas “tabelas de associação” são responsáveis por manter as chaves primárias dos registros que possuem um relacionamento do tipo muitos-para-muitos.

A aplicação conta também com um banco de dados interno (a memória *cache*) que foi criado seguindo o mesmo modelo do banco de dados externo. Nele são armazenadas todas as informações relacionadas ao usuário, de forma que sempre estejam em sincronia com o que está sendo mantido externamente. Uma das vantagens associadas à implementação de um banco de dados interno é a diminuição do risco de sobrecarga do servidor, que não terá de lidar com requisições desnecessárias. O usuário também é beneficiado, uma vez que o consumo de dados será reduzido consideravelmente, desde que o conteúdo armazenado em *cache* não seja deletado e, caso seja, será salvo novamente quando for consultado externamente.

## 4 Resultados

Neste capítulo serão descritos os resultados do trabalho, que incluem a arquitetura do sistema e o funcionamento das partes que o compõem, bem como a descrição do aplicativo e suas principais funcionalidades.

### 4.1 Sistema

O sistema é composto por três partes que interagem entre si para garantir o seu funcionamento. São elas: o aplicativo para dispositivos Android, o servidor e o banco de dados, que são descritas abaixo juntamente com a visão geral da aplicação.

#### 4.1.1 Visão Geral

O sistema é dividido em três camadas, seguindo as especificações do padrão [MVC](#). A camada de *view* é a camada de interação com o usuário, ou seja, é a interface do aplicativo, e tem como objetivo exibir informações em forma de textos ou representações gráficas, permitindo que o usuário visualize dados e realize operações sobre eles. O *controller* é responsável por interpretar as requisições dos usuário e, quando necessário, atualizar os modelos e o *layout*, ou redirecionar para uma nova *view*. A camada de *model* lida com a manipulação dos dados, ou seja, realiza a escrita e leitura no banco de dados e faz a validação necessária. Toda a interação entre componentes do sistema é ilustrada na Figura 1.

O aplicativo conta com uma memória *cache*, na qual são armazenadas todas as informações relacionadas ao usuário – isso inclui os dados informados no momento de cadastro, bem como suas listas e os produtos e estabelecimentos adicionados aos favoritos. Dessa forma, caso seja requisitada uma operação de consulta, a aplicação deve buscar registros armazenados em *cache* antes de realizar a consulta no banco de dados externo. Isso permite que as consultas sejam realizadas com um tempo de resposta menor e evita a sobrecarga do servidor.

O Rails facilita muito o processo de criação das partes que compõem o [MVC](#), uma vez que oferece comandos como o *scaffold* que cria todo o [CRUD](#) (acrônimo para *create*, *read*, *update* e *delete*), que abrange todas as operações básicas de inserção, leitura, atualização e exclusão de dados, além de gerar arquivos de *migration*, que se referem às modificações realizadas no banco de dados.

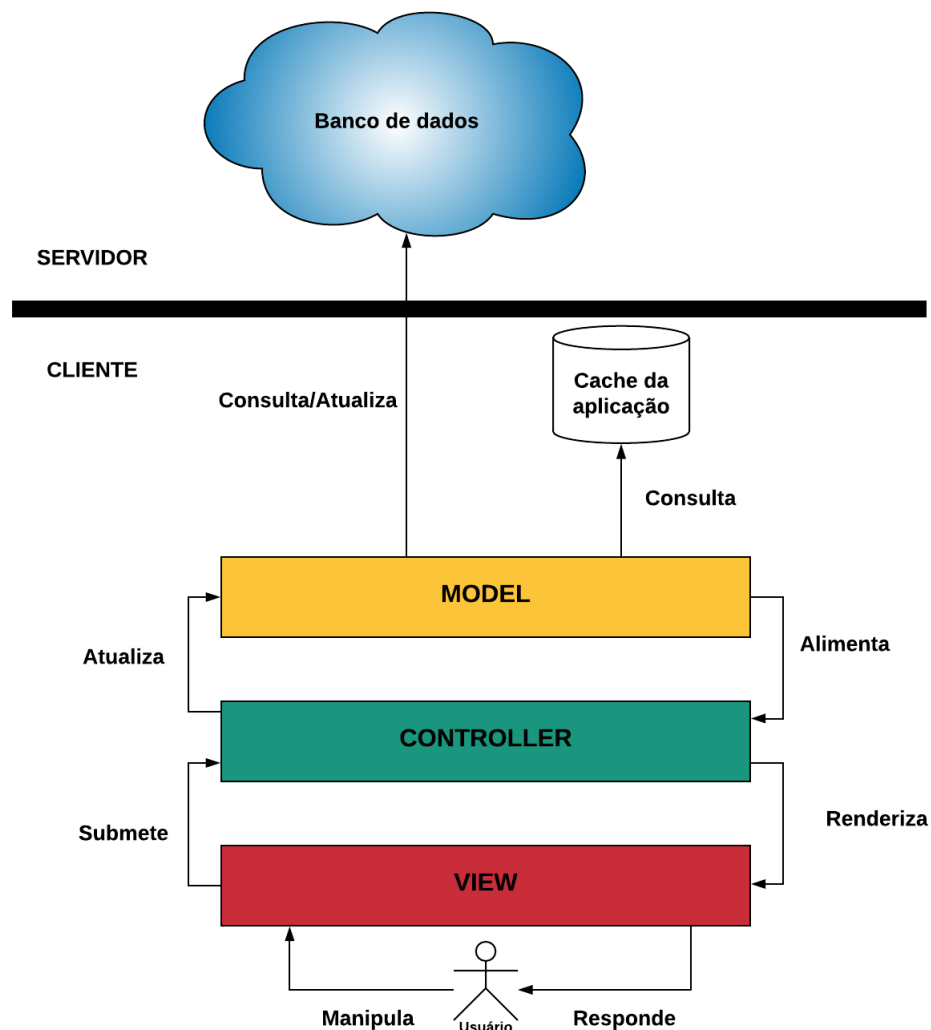


Figura 1 – Arquitetura da Aplicação

A troca de mensagens entre as partes que compõem o sistema é sempre iniciada pelo usuário, através do aplicativo. Nele, através da manipulação dos elementos presentes na *view*, são submetidas requisições ao servidor. Tais requisições podem conter instruções para modificar ou apenas consultar o modelo (banco de dados) da aplicação e fica a cargo do controlador identificar o tipo de operação a ser realizada. O sistema conta com dois controladores: (i) o primeiro, presente no aplicativo, que fica responsável por identificar a ação solicitada pelo usuário e, caso necessário, submetê-la ao servidor; e (ii) o segundo, que é mantido no servidor e é responsável por interpretar as requisições recebidas, consultar e/ou modificar o banco de dados e enviar uma resposta ao usuário. No caso do primeiro controlador, se a operação a ser realizada é uma consulta, este deve primeiramente consultar a memória *cache* da aplicação e, somente se não encontrar os dados buscados é que a requisição deve ser submetida ao servidor. Fica a cargo dos controladores interpretar os resultados das consultas e devolvê-los para o usuário através da atualização dos componentes da *view*. Esse processo pode ser visualizado na figura a

seguir:

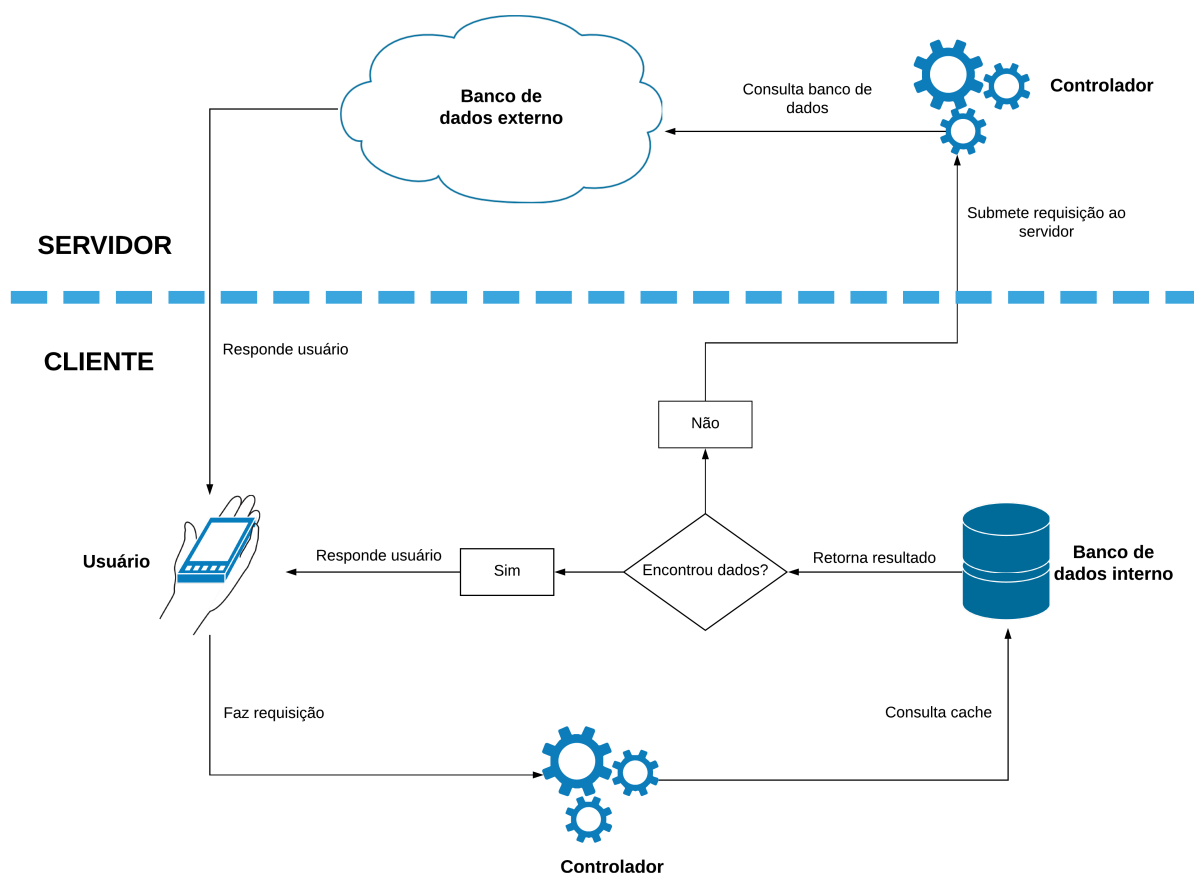


Figura 2 – Fluxo de comunicação entre componentes do sistema

Conforme ilustrado na Figura 2, o sistema é dividido em duas partes: (i) o cliente, que representa o usuário, o controlador local e a memória *cache* e abrange todas as operações realizadas localmente, como a atualização de componentes da *view*, consultas ao banco de dados interno e interpretação dos resultados obtidos; e (ii) o servidor, que é composto pelo controlador e banco de dados externos e é responsável por interpretar e responder as requisições enviadas pelo usuário, além de consultar e manipular o banco de dados.

O banco de dados da aplicação é composto por oito tabelas que representam as entidades que compõem o sistema, bem como seus atributos e relacionamentos. O Rails especifica alguns critérios para a nomeação dos componentes do banco de dados, como o uso de plural para nomear as tabelas e especificar alguns tipos de relacionamento e a nomeação de atributos que representam chaves estrangeiras que devem seguir o padrão *nomeDaTabelaReferenciada\_id*.

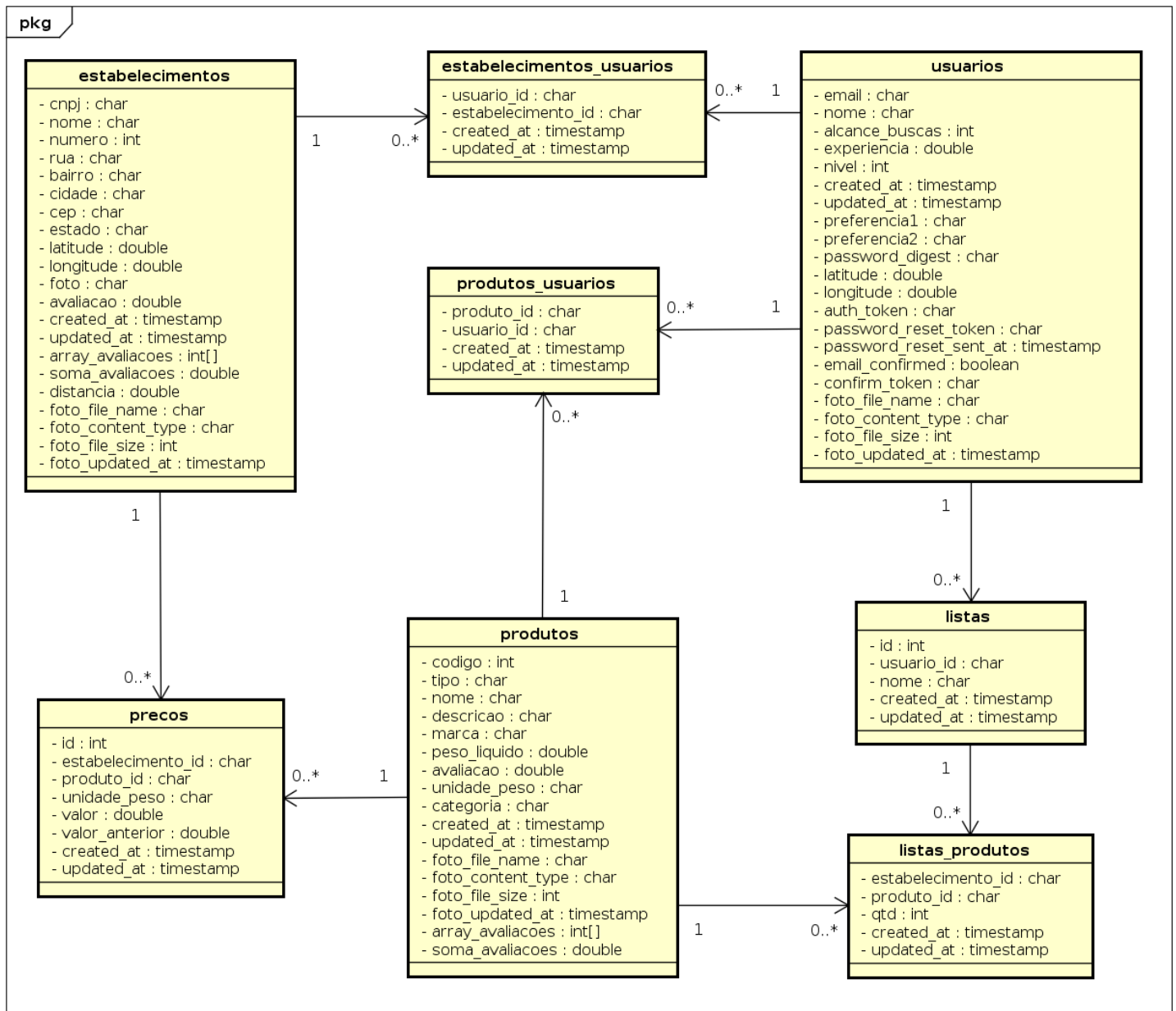


Figura 3 – Diagrama entidade-relacionamento

Como pode ser visto no diagrama entidade-relacionamento (ER) na Figura 3, existem três tabelas de associação. Esse tipo de tabela é responsável por armazenar os registros das entidades que possuem um relacionamento do tipo muitos-para-muitos e são compostas pelos identificadores das entidades que se relacionam. No caso desse projeto, as

tabelas `estabelecimentos_usuarios` e `produtos_usuarios` se referem aos estabelecimentos e produtos que o usuário pode definir como favoritos. Já a terceira tabela, `listas_produtos`, diz respeito aos produtos que uma lista pode conter. Nesses três casos, as entidades envolvidas não precisam obrigatoriamente relacionarem-se umas com as outras, de forma que o uso de chaves estrangeiras não se faz necessário. Dessa forma, quando o relacionamento entre esses objetos existir, este deve ser explicitado na terceira tabela.

A aplicação conta também com um banco de dados interno – a memória *cache*. Nele são armazenadas todas as informações relacionadas ao usuário, de forma que sempre estejam em sincronia com o que está sendo mantido externamente. Uma das vantagens associadas à implementação de um banco de dados interno é a diminuição do risco de sobrecarga do servidor, que não terá de lidar com requisições desnecessárias. O usuário também é beneficiado, uma vez que o consumo de dados será reduzido consideravelmente, desde que o conteúdo armazenado em *cache* não seja deletado e, caso seja, será salvo novamente quando for consultado externamente.

#### 4.1.2 Aplicação Móvel

O aplicativo foi desenvolvido utilizando o *Android Studio*, o ambiente para desenvolvimento de aplicações móveis da Google. Seu objetivo é fornecer uma interface amigável ao usuário e permitir que este faça requisições através da sua interação com o aplicativo. Para isso, o *layout* foi criado de forma que o usuário seja capaz de identificar as funcionalidades dentro de cada tela e utilizá-lo sem dificuldades.

Para cumprir com o seu objetivo de auxiliar seus usuários no momento das compras, o aplicativo conta com uma série de funcionalidades que visam facilitar esse processo, que vão desde a criação de listas e consultas individuais de preços de produtos à consultas mais elaboradas, nas quais são realizados cálculos para informar ao usuário o melhor estabelecimento para aquisição de uma lista de itens.

#### 4.1.3 Memória *cache*

Conforme mencionado anteriormente, o aplicativo conta com um banco de dados interno que é utilizado como memória *cache*. Nele são armazenadas todas as informações relacionadas ao usuário – dados pessoais, listas de compras, estabelecimentos e produtos definidos como favoritos, além das preferências de busca. Para isso, sempre que essas informações forem modificadas pelo usuário, será enviada uma requisição ao servidor que aplicará a mudança no banco de dados externo e, caso a operação seja bem sucedida, é replicada nos dados armazenados em *cache* e, caso contrário, os dados permanecem inalterados.

Pelo fato de informações relacionadas aos preços dos produtos mudarem com

frequência, esses dados sempre serão consultados externamente e não são armazenados em *cache*. O mesmo acontece para estabelecimentos e produtos que não fazem parte da lista de favoritos do usuário. Isso se deve, principalmente, à possibilidade de realizar buscas textuais dentro do aplicativo (tanto para produtos, quanto para estabelecimentos). Devido ao conceito de colaboratividade presente no aplicativo, é essencial que essas informações sejam consultadas externamente, uma vez que o usuário nunca possuirá armazenado em *cache* as informações de todos os produtos e estabelecimentos cadastrados no sistema – possuirá apenas aqueles de seu interesse, que foram adicionados aos favoritos.

Dentro desse contexto, é importante ressaltar que não é permitido ao usuário realizar cadastro ou alteração de informações sem que exista conexão com a internet. Todas as operações que envolvam manipulação dos dados é submetida ao servidor que aplica as devidas modificações no banco de dados externo e caso a operação seja bem sucedida, o servidor informa o aplicativo e este fica responsável por fazer as modificações no banco de dados local.

#### 4.1.4 Servidor

O servidor foi desenvolvido utilizando o *framework* Ruby on Rails. Seu principal objetivo é atender as requisições feitas pelos clientes e, para isso deve lidar com duas das partes do modelo MVC: o *controller* e o *model*. Conforme descrito anteriormente, ao utilizar o aplicativo, o usuário envia requisições para o servidor. Essas requisições são escritas na forma de rotas e devem, obrigatoriamente, especificar o controlador que contenha a função a ser executada, juntamente com o nome dessa função e o método de solicitação que será utilizado (*GET*, *POST*, *UPDATE*...). A partir da requisição recebida, o servidor executa a função especificada e faz operações de consulta, inserção ou atualização no banco de dados – o modelo da aplicação.

Ao gerar um *scaffold* no Rails, além do CRUD, que contém operações básicas do modelo, são criadas rotas padrões para cada uma dessas ações. No entanto, devido a complexidade de algumas consultas, que envolvem relacionamentos de um-para-muitos e muitos-para-muitos, foi necessária a construção de outras rotas e métodos para respondê-las.

O servidor responde as requisições seguindo o formato JSON (*JavaScript Object Notation*), um modelo para representação de dados utilizado para troca de informações. A principal vantagem na utilização desse modelo é o fato de os dados serem tipados, ao contrário de outros formatos de serialização como o XML (*eXtensible Markup Language*), que trata todos os dados como *string*. O servidor também é capaz de fazer a decodificação dos dados recebidos, graças ao suporte oferecido pelo *framework*.

## 4.2 O Aplicativo SocialMarket

O SocialMarket foi desenvolvido levando em consideração os aspectos positivos e negativos dos aplicativos descritos no capítulo 2, buscando trazer funcionalidades que pudessem proporcionar uma melhor experiência ao usuário. Dessa forma, são descritos abaixo o funcionamento e algumas das principais funcionalidades do aplicativo, juntamente com algumas questões que foram analisadas durante o desenvolvimento.

### 4.2.1 Cadastro e Autenticação

Para utilizar o aplicativo é indispensável que o usuário realize um cadastro, informando o nome, *e-mail* e senha a serem utilizados e, caso deseje, poderá também enviar uma foto que será utilizada em seu perfil, conforme ilustrado na Figura 4. O cadastro é exigido pois somente dessa forma é possível diferenciar os usuários que utilizam o aplicativo e os respectivos dados associados a cada um.

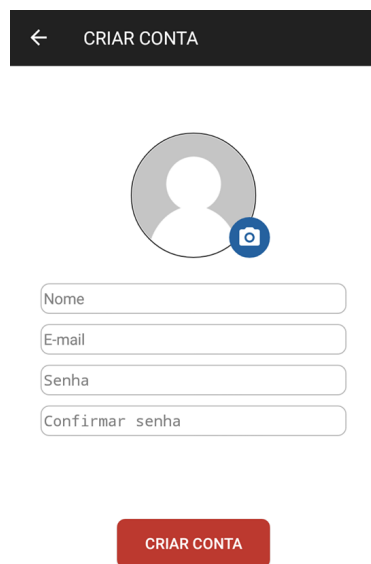
A tela de cadastro do aplicativo SocialMarket. No topo, há uma barra de navegação escura com uma seta para trás e o texto "CRIAR CONTA". Abaixo, há um ícone circular para upload de foto. Seguem quatro campos de entrada: "Nome", "E-mail", "Senha" e "Confirmar senha". No rodapé, há um botão vermelho com o texto "CRIAR CONTA".

Figura 4 – Cadastro

A tela de login do aplicativo SocialMarket. No topo, o texto "Bem vindo ao Social Market" em azul. Abaixo, há dois campos de entrada: "E-mail" e "Senha". Segue um botão vermelho com o texto "ENTRAR". Abaixo dele, há um botão com o ícone do Google e o texto "Fazer login". No rodapé, há dois links: "Primeiro Acesso? Clique aqui" e "Esqueci a senha".

Figura 5 – Tela de *Login*

Caso as informações fornecidas pelo usuário no momento de cadastro forem consistentes, ao clicar no botão “CADASTRAR”, é estabelecida a comunicação com o servidor, que receberá os dados e os armazenará no banco de dados. No entanto, o cadastro só será finalizado caso o usuário o confirme clicando no *link* de confirmação enviado para o *e-mail* informado. Dessa forma, evita-se que usuários utilizem *e-mails* de terceiros ou até mesmo inexistentes para acessar o aplicativo.

Uma vez cadastrado é permitido ao usuário acessar o aplicativo, através da tela de *Login* (Figura 5). Para isso, o usuário deverá informar o *e-mail* e senha cadastrados e, caso estejam corretos, o acesso à aplicação e suas funcionalidades são liberados. Existe



também a possibilidade de efetuar *login* através de uma conta Google, clicando no botão correspondente. Caso o usuário opte por essa opção, é exibido uma janela onde deverá selecionar uma das contas atreladas ao seu dispositivo e, em seguida o *login* é efetuado.

Ao optar por acessar o aplicativo através de uma conta Google é imprescindível realizar o cadastro do usuário no banco de dados do sistema através do *e-mail* associado à essa conta. Isso se faz necessário pelo fato de existirem diversas informações relacionadas aos usuários do *SocialMarket* que não podem ser atreladas à uma conta Google. Dessa forma, quando essa opção é selecionada, é verificado se existe algum cadastro que utiliza o *e-mail* da conta selecionada e, caso não exista, é gerada uma senha aleatória que é submetida ao servidor juntamente com as outras informações necessárias (nome e *e-mail*) e o cadastro é realizado. É importante ressaltar que a senha é gerada aleatoriamente e o usuário não tem conhecimento acerca disso pelo fato dessa senha ser desnecessária para esse caso, mas se faz necessária pelo fato de ser um campo obrigatório no banco de dados.

#### 4.2.2 Listas de Compras

O aplicativo permite a criação de listas de compras a partir do botão localizado no canto superior direito da tela principal do aplicativo (Figura 6), onde são exibidas todas as listas já criadas pelo usuário. Para criar uma nova lista é necessário informar apenas um nome que ainda não esteja sendo utilizado em nenhuma de suas outras listas. O usuário tem ainda opção de excluir uma lista de compras e adicioná-la aos favoritos – funcionalidade que permite a adição de todos os produtos da lista de compra à lista de produtos favoritos. Essas funcionalidades ficam “ocultas” na tela e só podem ser visualizadas a partir do recurso de *swipe*, que permite que o usuário deslize algum item para a esquerda, de forma a exibir essas opções. Esse recurso foi utilizado como forma de deixar o *layout* menos poluído e com excesso de informações.

Em cada uma das listas de compras são adicionados os produtos que o usuário tem intenção de comprar e é possível acessá-las individualmente (Figura 7), de forma que seja possível visualizar as informações dos produtos adicionados em cada lista, como nome, marca, peso líquido, avaliação e foto (se houver). Ao lado de cada produto é exibida a quantidade de itens que foi adicionado à lista que pode ser modificada clicando nos botões “+” e “-”. No lado direito de cada item da lista existe uma seta, que ao ser clicada pelo usuário, exibe as ofertas (4.2.5) do produto segundo o critério de busca definido pelo usuário. Ao clicar no ícone localizado no canto superior direito dessa tela é possível acessar o menu de configurações de busca, descrito em 4.2.3.

Existe ainda um botão no canto inferior direito da tela mostrada na Figura 7, que ao ser clicado oferece outras duas opções: buscar produtos ou ir às compras, descritas nas Seções 4.2.4 e 4.2.7, respectivamente.

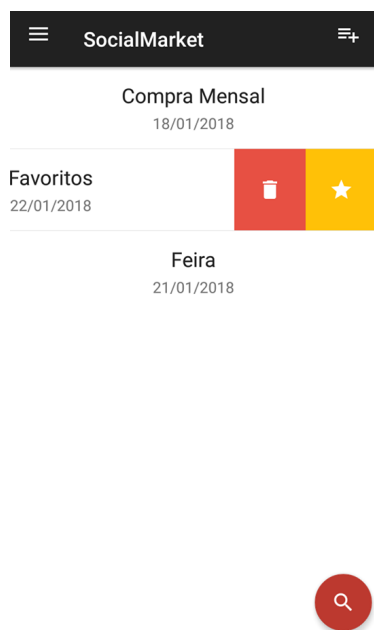


Figura 6 – Listas do usuário

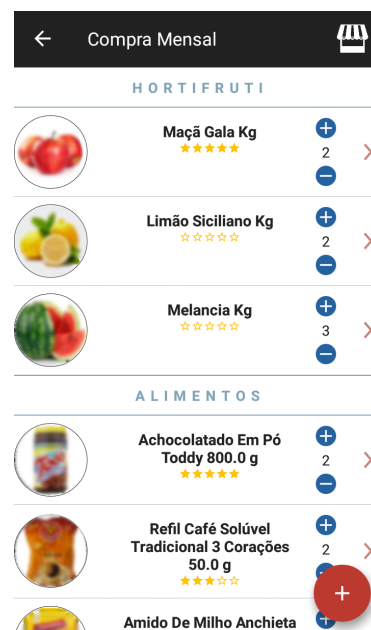


Figura 7 – Produtos da Lista

### 4.2.3 Configurações de Busca

O aplicativo conta com dois menus de configurações para a busca por preços. O primeiro (Figura 8) permite ao usuário configurar alguns parâmetros gerais da busca por ofertas: se deverão ser considerados vários estabelecimentos ou apenas um e se os estabelecimentos considerados na busca serão aqueles definidos como favoritos ou os que estão mais próximos do usuário. Caso o usuário opte pela última opção, deverá também especificar o raio de busca, que pode variar entre 0 e 20 quilômetros.



Figura 8 – Menu de configurações gerais

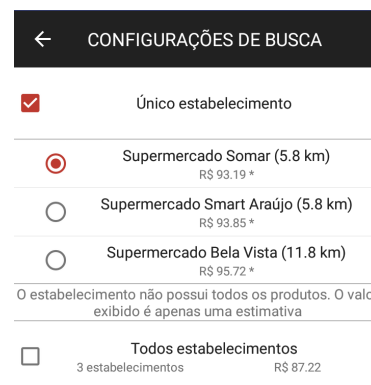


Figura 9 – Menu de configurações específicas

O segundo menu de configuração é mais específico e pode ser acessado a partir de

uma lista de compras, conforme explicado na Seção 4.2.2. Nesta tela (Figura 9), são exibidos o valor total da compra caso sejam considerados todos os estabelecimentos que atendam ao critério de busca definido no primeiro menu e o valor total da compra considerando esses mesmos estabelecimentos de forma individual. Neste momento, o usuário deve selecionar a opção que mais se adeque às suas necessidades e essa escolha é levada em consideração no momento das compras (Seção 4.2.7).

Existem situações em que um estabelecimento não possui preços cadastrados para todos os itens da lista e, ao realizar o cálculo do valor total da compra, tais estabelecimentos poderiam ser classificados equivocadamente como sendo os de melhores ofertas. Como forma de contornar a situação, foi definido que o valor do produto cujo preço não está cadastrado para um determinado estabelecimento seria o menor valor cadastrado desse produto acrescido de 10%. Dessa forma, evita-se discrepâncias muito grandes nos valores exibidos, fator que poderia influenciar na decisão do usuário.

O cálculo do valor total da compra em vários estabelecimentos demanda um alto custo computacional quando são considerados diversos estabelecimentos. A solução adotada para contornar esse problema consiste basicamente em limitar o número de estabelecimentos avaliados para cinco nos dois casos – compra em vários estabelecimentos ou em apenas um. Diferentemente da situação em que são considerados estabelecimentos próximos, não existe nenhum critério que possa colocar um estabelecimento definido como favorito a frente de outro. Dessa forma, optou-se por limitar a quantidade de estabelecimentos favoritos como sendo cinco, impedindo que o usuário favorite novos estabelecimentos quando este limite for atingido. Apesar de simplória, a solução adotada atende aos objetivos do aplicativo, uma vez que dificilmente um usuário se disporia a visitar mais de cinco estabelecimentos para realizar uma única compra.

#### 4.2.4 Mecanismos de Busca

A busca por produtos pode ser realizadas de três formas: através do leitor de código de barras, pelo nome do produto, ou pela inserção manual do código. Todas essas opções são exibidas para o usuário na tela exibida na Figura 10. Para utilizar o *scanner*, basta posicionar o código de barras em frente ao leitor, aguardar que a imagem foque, para que seja possível realizar a leitura do código. Em seguida, o usuário é redirecionado diretamente para as ofertas do produto ao qual o código de barras lido pertence.



Figura 10 – Tela de busca

Caso o usuário opte pela inserção manual do código de barras, deverá informar o código de barras do produto buscado para ser redirecionado para a tela de ofertas. Na busca por nome, é necessário informar o nome ou alguma informação referente ao produto a ser buscado e, como podem existir diferentes produtos que correspondam ao texto informado pelo usuário, o resultado é exibido em forma de lista, onde são listados todos os produtos que contenham o termo buscado. Neste momento, o usuário poderá adicionar os produtos à alguma de suas listas, utilizando o recurso *swipe* para exibir essa opção. Para visualizar as ofertas de um determinado item, basta que o usuário o selecione.

#### 4.2.5 Ofertas

Na tela de ofertas (Figura 11) são exibidas as informações do produto e logo abaixo são listadas as ofertas existentes para aquele produto. Neste momento é permitido ao usuário adicionar o produto aos favoritos, a partir do recurso de *swipe* que exibe essa opção ao deslizar o produto para a esquerda.

As ofertas exibidas são aquelas que correspondem aos critérios de busca do usuário, obedecendo sua preferência pelos estabelecimentos favoritos ou próximos e o raio máximo de busca. Cada oferta é descrita pelo seu valor, o nome do estabelecimento e sua respectiva avaliação. As ofertas são ordenadas por preço e aquela que possuir o menor valor fica em destaque como sendo o primeiro item da lista.

O usuário pode ainda acessar as informações do estabelecimento de cada oferta, clicando em cima do seu nome e mudar sua avaliação a partir da barra de classificação. Para visualizar informações do produto, o usuário deve clicar na imagem do produto e sua avaliação pode ser modificada da mesma forma que a do estabelecimento. Para adicionar o produto à uma lista de compras, o usuário deve clicar no ícone localizado no canto superior



Figura 11 – Lista de Ofertas

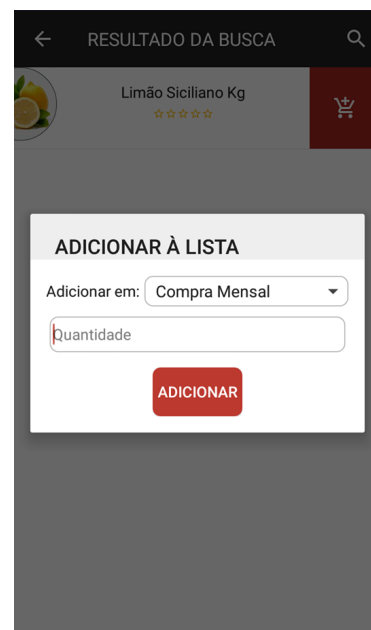


Figura 12 – Dialog para adição produto à lista

direito da tela e, em seguida, selecionar a lista desejada e especificar a quantidade de itens, conforme exemplificado na Figura 12.

Para cada oferta, existe a opção de atualizar o preço informado e confirmá-lo. Ambas as opções podem ser acessadas através do recurso de *swipe*. Ao optar por confirmar o preço, é exibida uma caixa de diálogo, solicitando que o usuário confirme a ação e, caso o faça, é exibida uma mensagem informando se a operação foi bem sucedida ou não. Caso o usuário opte por atualizar o preço, ele é redirecionado para uma outra tela onde é solicitado o novo valor da oferta.

Ao clicar no botão localizado no canto inferior direito, são exibidas duas opções: buscar produto e cadastrar um novo preço que são descritas nas seções 4.2.4 e 4.2.6, respectivamente.

#### 4.2.6 Cadastro de Produtos e Preços

O aplicativo leva em consideração que existem dois grupos de produtos nos supermercados: aqueles que possuem código de barras e os que não possuem – como os do setor hortifruti, por exemplo. O que diferencia essas categorias não é apenas o código de barras, uma vez que um produto que o possua já tem todas as suas características, como peso líquido e marca associadas a esse código. Dessa forma, antes de realizar o cadastro de algum produto, o usuário deve selecionar o grupo ao qual ele pertence (Figura 13).

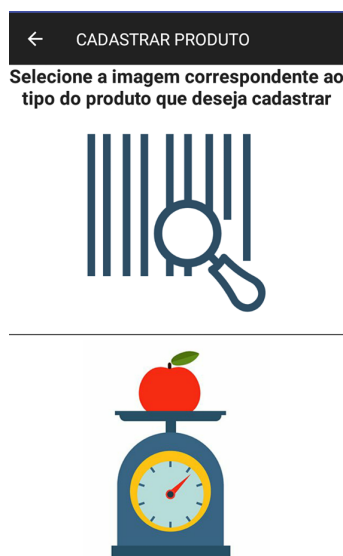


Figura 13 – Seleção do tipo de produto a ser cadastrado  
Arte: katemangostar / Freepik <sup>1</sup>(Balança), Freepik (Código de Barras)

Caso o usuário opte por cadastrar um produto que possua código de barras, é necessário que informe todas as informações do produto: nome, marca, peso líquido, unidade de peso e categoria, conforme exemplificado na Figura 14. No entanto, para os produtos sem código de barras não é exigida nenhuma outra informação além de nome e categoria, uma vez que as outras características como marca e peso não pertencem à esse tipo de produto.

Figura 14 – Cadastro de Produto com Código de Barras

Figura 15 – Cadastro de preço para Produto sem Código de Barras

<sup>1</sup> <<https://www.freepik.com>>

O cadastro de preços para os dois tipos de produto também é realizado de maneira diferente. Produtos que contenham código de barras já possuem todas as suas informações atreladas ao seu código e para cadastrar um novo preço, o usuário deve apenas informar o seu valor e o estabelecimento ao qual esse valor pertence. No entanto, para aqueles que não possuem código de barras, na maioria das vezes, o seu valor está associado à sua unidade de peso. Dessa forma, ao cadastrar um novo preço o usuário deve informar a unidade de peso do produto (quilograma, grama, litro, mililitro, unidade ou dúzia) e valor correspondente (Figura 15).

Um aspecto notado durante o desenvolvimento do aplicativo foi que um mesmo produto pode ser vendido em diferentes unidades de peso e que isso poderia prejudicar o usuário. Por exemplo, considerando o cenário em que o produto “Ovo” esteja cadastrado no aplicativo e que um determinado estabelecimento X venda ovos por unidade e o estabelecimento Y venda por dúzia. Ao cadastrar o preço da unidade, esse teria um valor muito abaixo do valor da dúzia e isso poderia fazer com que o usuário optasse por comprar no estabelecimento que vende a unidade acreditando que esse valor corresponda à dúzia ou qualquer outra unidade de medida. Dessa forma, optou-se por criar um novo produto para cada preço cadastrado cuja unidade de medida não corresponda à nenhuma outra cadastrada no banco de dados, de forma que o novo produto seja descrito por seu nome e sua unidade de peso.

#### 4.2.7 Compras

A tela de compras (Figura 16) só pode ser acessada a partir de uma lista de compras (Seção 4.2.2) e exibe os produtos de forma semelhante, adicionando a informação de preço. No entanto, neste momento é levado em consideração os critérios de busca definido pelo usuário descritos na Seção 4.2.3 e caso o usuário opte por comprar em vários estabelecimentos, os produtos são separados de forma que seja possível visualizar aqueles a serem comprados em cada estabelecimento, cujo nome é exibido acima da lista de produtos e para alternar entre um estabelecimento e outro é necessário clicar nas setas ao lado.

Por outro lado, se o usuário optar por comprar em um único estabelecimento é exibida uma única lista com todos os produtos. Caso existam produtos na lista cujo valor não está cadastrado para o(s) estabelecimento(s) que atendam o critério de busca, estes são adicionados em uma outra lista que é adicionada ao final da Compra, conforme exemplificado na Figura 17.

Acima da lista de produtos é possível visualizar o valor total da compra, que é calculado com base no valor de cada produto e na quantidade adicionada à lista. Uma outra funcionalidade que pode auxiliar o usuário no momento de compras é a possibilidade de selecionar os produtos da lista, marcando o *checkbox* ao lado de cada item, indicando que aquele produto será comprado. Baseando-se nos itens selecionados, o aplicativo informa

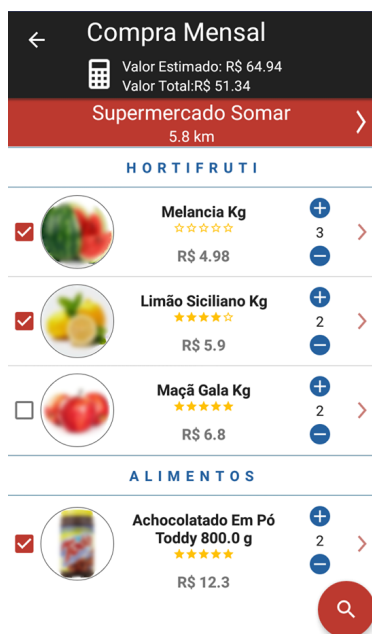


Figura 16 – Momento das compras

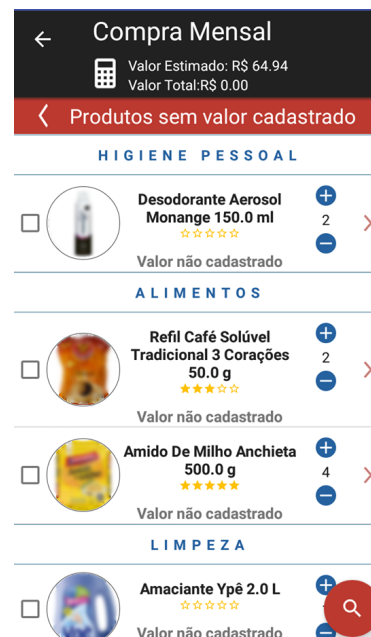


Figura 17 – Produtos que não possuem preço cadastrado

ao usuário o valor da compra apenas para os produtos que foram selecionados.

### 4.3 Considerações

A partir das funcionalidades apresentadas percebe-se que o SocialMarket cumpre com os objetivos propostos. Ao utilizá-lo, o usuário é capaz de visualizar as ofertas existentes para um produto específico, acessando-o através dos mecanismos de buscas ou de suas listas. A partir dos menus de configuração de busca é possível definir o raio máximo da busca por ofertas e caso o estabelecimento desejado pelo usuário não esteja próximo, existe ainda a possibilidade de adicioná-lo aos favoritos e selecionar a opção de buscar ofertas em estabelecimentos favoritos. Dessa forma, o aplicativo não restringe o usuário mostrando apenas informações de preços nos estabelecimentos de sua região. A funcionalidade “Compras” se destaca como sendo o diferencial do SocialMarket, por permitir que o usuário visualize, separadamente, os produtos da lista e seus respectivos preços nos estabelecimentos com as melhores ofertas.

Portanto, caso o aplicativo se popularize e a comunidade de usuários coopere com a atualização das informações, acredita-se que o SocialMarket se tornará uma ferramenta de grande auxílio para aqueles que buscam economizar no momento das compras, tornando-se um atrativo tanto para as pessoas que praticam a pesquisa de preço, quanto para aquelas que não a praticam, devido à simplicidade que traz ao processo.



## 5 Conclusão

A pesquisa de preço faz parte da rotina da maioria das pessoas e traz ganho significativo para aqueles que a praticam. No entanto, o uso dessa prática para aquisição de itens em estabelecimentos físicos se torna impraticável para a maioria da população, principalmente em situações em que existem vários produtos a serem adquiridos. Nessas circunstâncias, existe um dispêndio de tempo e recursos que pode ser ampliado a depender da frequência de realização da prática.

Os impedimentos para a realização da pesquisa de preço poderiam ser amenizados com o uso de ferramentas que armazenem e gerenciem dados relacionados à ofertas de produtos em estabelecimentos físicos. Diferente dos itens comercializados em lojas virtuais, que podem ter suas informações obtidas através de uma consulta na *web*, os dados relacionados aos produtos vendidos em lojas físicas não podem ser facilmente adquiridos. Baseando-se nisso, foi proposto o SocialMarket, um aplicativo colaborativo par auxiliar nas compras em supermercados, cujo objetivo é prover funcionalidades que amparem seus usuários em todas as etapas do processo de compra, que vão desde a pesquisa de preço até o momento de aquisição dos produtos.

Neste trabalho foram apresentados todos os aspectos relacionados ao desenvolvimento do SocialMarket, desde sua relevância até a construção do sistema em si. Foram descritas também as principais funcionalidades do aplicativo, bem como as decisões de projeto que levaram a construção da aplicação final.

O aplicativo foi desenvolvido buscando apresentar uma *interface* amigável ao usuário, de forma a facilitar o seu uso. O *framework* utilizado possui uma ampla documentação disponível e uma comunidade ativa, fazendo com que a manutenção do sistema seja facilitada. Tais características compõem o conjunto de requisitos não funcionais que o sistema deve possuir, que engloba também questões relacionadas a sua integridade e segurança dos usuários, que diz respeito à restrição de acesso não autorizado ao aplicativo e armazenamento seguro de senhas.

A versão básica do aplicativo foi finalizada e cumpre com o objetivo proposto. No entanto, algumas das funcionalidades que foram listadas como requisitos do sistema não foram implementadas pelo fato de não serem relevantes para a primeira versão do aplicativo ou por demandarem muito tempo para serem implementadas – aspecto que poderia prejudicar a implementação das funcionalidades básicas e essenciais para o funcionamento do SocialMarket.

## 5.1 Trabalhos Futuros

Como resultado deste trabalho, é apresentada apenas a primeira versão do aplicativo, que possui vários aspectos a serem melhorados. Dessa forma, como trabalhos futuros, são sugeridos os seguintes tópicos:

- Implementação de políticas de atualização e cadastro de informações no aplicativo, principalmente no que diz respeito a produtos e preços que são os principais elementos da aplicação;
- Otimização de consultas no servidor e a inserção e atualização dos dados em *cache* devem ser aprimorados, de forma a reduzir o tempo de resposta ao usuário e evitar requisições desnecessárias ao servidor;
- O *layout* do aplicativo pode ser melhorado, de forma a garantir uma melhor experiência ao usuário; e
- Desenvolvimento de um mecanismo de gamificação, que deve servir como incentivo para os usuários cadastrarem e atualizarem as informações exibidas no aplicativo.

# Referências

- ARAÚJO, R. B. Computação ubíqua: Princípios, tecnologias e desafios. *XXI Simpósio Brasileiro de Rede de Computadores*, 2003. Nenhuma citação no texto.
- BEMBEM, A. H. C. et al. Inteligência coletiva: um olhar sobre a produção de pierre lévy. *3º Simpósio Hipertexto e Tecnologias na Educação: redes sociais e aprendizagem*, 2010. Nenhuma citação no texto.
- BRING! <<https://www.getbring.com>>. Acesso em 16/01/2018. Nenhuma citação no texto.
- COMPARE Preços em Supermercados e Lojas. Nota Fiscal Paulista (NFP) | Meus Preços. 2014. Disponível em: <<http://meusprecos.com.br>>. Nenhuma citação no texto.
- COSTA, M. F. da et al. Determinantes da decisão de compra do consumidor no setor supermercadista. *Seminário em Administração*, São Paulo, 2007. Citado na página 14.
- EMERCADO. <[www.emercadoapp.com](http://www.emercadoapp.com)>. Nenhuma citação no texto.
- KOTLER, P.; KELLER, K. L. *Administração de marketing*. 12. ed. [S.l.]: Pearson Prentice Hall, 2006. Nenhuma citação no texto.
- LÉVY, P. *A Inteligência Coletiva: Por uma antropologia do ciberespaço*. 5. ed. São Paulo: Edições Loyola, 2007. Citado na página 16.
- MOWEN, J. C.; MINOR, M. *Consumer Behavior*. 5. ed. [S.l.]: Prentice-Hall, 1995. Nenhuma citação no texto.
- OURGROCERIES. <<https://www.ourgroceries.com>>. Acesso em 16/01/2018. Nenhuma citação no texto.
- OUTOFMILK. <<https://www.outofmilk.com/>>. Nenhuma citação no texto.
- PEDROZO, S. A. *Pesquisa de preços pode gerar economia de 60%, em média, no supermercado*. 2016. Acesso em 05/11/2017. Disponível em: <<http://www.dgabc.com.br/Noticia/2139933/pesquisa-de-precos-pode-gerar-economia-de-60-em-media-no-supermercado>>. Citado na página 14.
- PFLEEGER, S. L. *Engenharia de Software: Teoria e Prática*. 2. ed. [S.l.]: Prentice Hall do Brasil, 2004. Citado na página 20.
- SCHIFFMAN, H.; KANUK, L. L. *Comportamento do Consumidor*. 9. ed. [S.l.]: LTC, 2009. ISBN 9788521616849. Citado na página 16.
- SOFTLIST. <[www.softlist.aptoide.pt](http://www.softlist.aptoide.pt)>. Acesso em 13/08/2017. Nenhuma citação no texto.