Pontifícia Universidade Católica do Rio de Janeiro

Desconto ou Cilada

Rede social de compartilhamento de promoções,
com catalogação de produtos.

Lucas Rebello Damo
Projeto Final de Graduação

CENTRO TÉCNICO CIENTÍFICO - CTC
DEPARTAMENTO DE INFORMÁTICA

Curso de Graduação em Ciência da Computação
Rio de Janeiro, Julho de 2022

PUC
RIO

Lucas Rebello Damo

Desconto ou Cilada

Rede social de compartilhamento de promoções, com catalogação de produtos.

Relatório de Projeto Final, apresentado ao curso de Ciência da Computação da PUC-Rio como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

Orientador: Alberto Barbosa Raposo

Rio de Janeiro
Julho de 2022

Resumo

Rebello Damo, Lucas. Barbosa Raposo, Alberto. Desconto ou Cilada - Rede social de compartilhamento de promoções e grupamento de produtos. Rio de Janeiro, 2022. Relatório Final de Projeto de Graduação - Departamento de Informática. Pontifícia Universidade Católica do Rio de Janeiro.

Este projeto visa criar um ambiente único para a tomada de decisão de compra do consumidor, amparada pela opinião de outros consumidores e a visualização de outras ofertas do mesmo produto ou de semelhantes nos diferentes e-commerces brasileiros. Para isso, foi desenvolvido um aplicativo em Flutter, focado na plataforma Android, um banco de dados relacional, uma API e um crawler.
O aplicativo almeja a fácil visualização dos dados armazenados no banco de dados e obtidos tanto por outros usuários quanto pelo crawler, criando um espaço organizado com elementos de uma rede social, como perfil, comentários e curtidas.

Palavras-Chave: Rede Social, E-Commerce, Banco de Dados, Flutter, Python.

Abstract

Rebello Damo, Lucas. Barbosa Raposo, Alberto. Desconto ou Cilada - Social network for sharing promotions. Rio de Janeiro, 2022. Relatório Final de Projeto de Graduação - Departamento de Informática. Pontifícia Universidade Católica do Rio de Janeiro.

This project aims to create a singular decision-making tool for consumers online, where they can base it on others opinion and all the offers of the same product of different brazilian e-commerces. With this goal the following was developed: An app, a relational database instance and a crawler.
The app aims to provide a simplified view of the data contained in the database and obtained by users' input and by the crawler. Providing an organized space with social networking features as profile, commentaries and likes.

Aos meus pais, por acreditarem e me apoiarem sempre.

Sumário

1 Introdução

Não é de hoje que os serviços vêm se posicionando no ambiente virtual. Cada vez mais se prova fundamental a presença online para a comunicação entre clientes e fornecedores.

O surgimento da pandemia com a nova doença COVID-19 foi um grande catalisador para que varejistas e clientes pulassem de vez para o espaço cibernético. A pandemia obrigou o distanciamento físico entre pessoas e, dessa forma, entre o provedor e o consumidor.

Para o varejo, essa situação foi ainda mais conturbada, pois muitos sequer possuíam a estrutura para realizar vendas online e outros já estavam exclusivamente nesse espaço. O que aconteceu, portanto, foi um amarrotamento de lojas competindo pela visualização do cliente e também pelo melhor preço. No que tange ao internauta em busca de um produto específico, isso causou uma infinidade de ofertas com condições não muito claras e comparação ainda mais difícil [1].

Aqueles, que antes podiam consultar o vendedor e conhecer o produto fisicamente, agora dependem do “boca a boca”, que ganha proporções inimagináveis por conta das inúmeras conexões e hiperlinks que existem no ciberespaço, elevando a uma escala exponencial o impacto das “dicas” [2].

Além disso, muitos varejistas adotaram um novo modelo de negócios, o famoso, marketplace. A adoção desse novo estilo de negócio trouxe mais problemas ao consumidor, que agora precisa identificar quem é de fato o responsável pela venda, o site ou um terceiro que está apenas utilizando a plataforma para oferecer suas próprias mercadorias.

Ao utilizar o google, por exemplo, o consumidor será bombardeado de anúncios e ofertas do mais variados e-commerces, alguns marketplaces e outros não. Se a busca for por avaliações, a situação ainda piora pois, com tantos anúncios pagos e com empresas brigando para aparecer na página de resultados, dificilmente ele encontrará uma opinião de uma parte idônea. Ou ainda, irá cair na página de avaliações de um dos sites de varejo e encontrará comentários que se referem mais ao serviço de venda do que do produto em si.

Com o intuito de melhorar a experiência dos consumidores, esse trabalho visa desenvolver justamente um espaço idôneo onde todos poderão contribuir com avaliações acerca dos produtos, lojas e preços. Dessa forma, pretende-se formar um novo ciberespaço organizado e propício a trocas genuínas entre os internautas que buscam melhores mercadorias ou preços.

Para atingir esse objetivo, será necessário um aplicativo que misture as funcionalidades de uma rede social, com as de uma ferramenta de busca. Para tal, serão implementadas quatro peças interdependentes: O banco de dados, a interface do usuário, o servidor e o *crawler*.

Este trabalho está organizado da seguinte forma: A seção 2 apresenta a situação atual, de quais ferramentas um internauta-consumidor dispõe, seus defeitos e pontos fortes. A seção 3 entra em mais detalhes da solução proposta e de sua implementação. A seção 4 traça um plano de ação com os cronogramas pretendidos ao longo do trabalho final I e II. E a seção 5 conclui com uma visão geral do que foi alcançado.

2 Situação Atual

Nesta seção, serão analisadas as principais ferramentas já existentes e mais utilizadas atualmente. O objetivo é evidenciar os pontos fortes, o que falta nelas e como poderia ser diferente de forma a conciliar os acertos com uma melhoria nas limitações.

Ao buscar por “comparar preços” no google [16], aparecem as quatro ferramentas mais conhecidas: JaCotei, Zoom, Buscapé e BondFaro. Dessas, as três últimas são da mesma empresa, a Mosaico, que recentemente entrou na bolsa de valores levantando mais de 1 bilhão de reais [3].

Ao acessar qualquer uma das quatro plataformas citadas, a primeira sensação é de que se trata do mesmo produto. O foco é o mesmo: Categorizar produtos, filtrá-los e comparar os preços. A interface se assemelha muito a um site de vendas normal, com a principal diferença de oferecer mais de um fornecedor e ver o histórico recente de preços. Entretanto, o fluxo de utilização é claro e exclusivo: Comparar preços de um produto que o cliente já escolheu.

Com um quase monopólio do segmento, a mesma interface em todas suas plataformas e alta popularidade, é difícil imaginar que em algum momento próximo o modelo de negócios da Mosaico mudará [3], e fica evidente que há pouco espaço para interação entre usuários em suas plataformas, e para recomendações de ofertas. Problema este, que uma quinta ferramenta busca solucionar: o Pelando [17].

O Pelando está muito mais próximo de uma rede social do que de uma plataforma de busca. Inclusive, não há categorização, comparativos de ofertas entre lojas e muito menos histórico de preços. Sua página inicial é um *feed*, contendo *posts*, que são promoções postadas por usuários. O processo acontece da seguinte maneira: Um internauta nota um preço atipicamente barato para um determinado produto, ou então alguma promoção por cupom ou outra campanha de marketing. Esse usuário vai até a plataforma e coloca o link para a página da oferta, o valor e uma pequena descrição, caso necessário, de como chegar a tal preço. Nesse post, outros usuários interagem *esfriando* ou *esquentando* as promoções, ou então, adicionando comentários a elas.

Dessa forma, pode-se dizer que existem dois modelos de soluções para a busca por menores preços. O primeiro modelo, exercido pela empresa Mosaico, é centrado na parte mais objetiva e automatizada, a fim de obter os preços das lojas, traçar históricos e agrupar os produtos. O segundo, do pelando, é centrado nos usuários, nas opiniões deles e em promoções que dificilmente seriam capturadas pelos algoritmos do primeiro, pois muitas das vezes precisam de cupons ou da utilização do aplicativo do e-commerce. Além disso, enquanto no primeiro o usuário normalmente já está buscando um produto específico, no segundo o usuário costuma ir em busca de bons negócios, e pode ser estimulado a comprar um produto que não precisava ou sequer tinha vontade, caso acredite ser uma oportunidade única.

O diferencial é gritante, o processo de busca pelo melhor preço no primeiro modelo é algo solitário e individual. Enquanto no segundo, há a participação de diversas pessoas que compartilham experiência com produtos, lojas, preços ou, até mesmo, sobre expectativas de futuras ofertas. Mas, apesar de solucionar o quesito de sociabilidade, o pelando falha em oferecer um ambiente democrático, ao impedir que os usuários se refiram a outras plataformas, e faltam as funcionalidades do primeiro modelo, de visualizar de forma agregada informações como histórico de preços. Outra funcionalidade que deixa a desejar é o modo de criar alertas para novas promoções, que só é possível a partir de tags. Estas

tags muitas das vezes alertam usuários interessados em processadores de computadores, sobre promoções de celulares, pois estes possuem processadores em suas descrições ou, até mesmo, alertam sobre promoções de processadores de comida.

3 Proposta e objetivos do trabalho

Como mencionado anteriormente, existem dois modelos de ferramentas para o consumidor que são extremamente opostos. Um é focado completamente na parte objetiva de comparar os preços entre as principais lojas, o outro é focado na parte social de compartilhar ofertas manualmente.

Cada vez mais, ferramentas auxiliam os seres humanos em todos os quesitos da vida, porém nenhuma tecnologia substitui a interação social. Não à toa, os maiores sites do mundo são justamente redes sociais, com uma a cada três pessoas sendo usuárias deste tipo de plataforma [12]. A ambição desse projeto é conciliar a conexão interpessoal com o lado mais prático de buscar por produtos específicos, categorias ou, até mesmo, de encontrar ofertas imperdíveis associado a uma experiência de compra online mais fácil, interessante e sociável.

Na situação atual de e-commerce, muitos seguem o padrão marketplace, o que dificulta o rastreamento dos menores preços oferecidos, uma vez que ofertas de marketplace podem ser de baixa confiabilidade. Além disso, muitas ofertas são oferecidas por meio de cashbacks, cupons e associadas a outras plataformas, que o consumidor pode não conhecer. A parte social da proposta deste trabalho, ataca justamente esse problema, uma vez que os usuários poderão compartilhar e relatar estas promoções “escondidas”, que seriam extremamente difíceis de serem descobertas por algoritmos ou outros meios automatizados.

Juntando a parte automatizada e objetiva com a interação de uma rede social, é esperado que a plataforma seja suficiente na tomada de decisão de um consumidor, tirando a necessidade de se buscar e juntar informações de diversas ferramentas. E, com isso, espera-se que a experiência se torne mais fácil, garantindo que cada compra esteja sendo um bom negócio a um preço justo.

Para isso, a solução proposta visa encontrar um meio termo entre ambos os modelos discutidos na seção anterior, um aplicativo capaz de fornecer os comparativos, histórico de preços e as atuais ofertas, sem perder o lado social, permitindo posts de novas promoções, comentários e outras interações entre consumidores. Será desenvolvido uma plataforma que não mais abre mão das comodidades tecnológicas, mas as utiliza para facilitar o processo de compra de qualquer usuário e sem deixar de lado a possibilidade de se contar com a opinião de terceiros para garantir uma boa compra.

Nesta seção será detalhado o produto final almejado para substituir as ferramentas existente, além disso também será levada em consideração os custos da estrutura necessária para manter a plataforma em funcionamento, além das possíveis fontes de receita, a fim de traçar de forma, ainda que superficialmente, o potencial de auto-sustento ou até lucro da mesma.

Dado que o tempo de desenvolvimento é limitado, a solução aqui proposta não será implementada em sua totalidade, em contrapartida um protótipo com as implementações consideradas mais fundamentais será implementado e o backend será configurado e executado. Dessa forma, o protótipo será uma versão inicial mas operacional da solução proposta, e as funcionalidades faltantes poderão ser desenvolvidas a partir da base fundada por esse projeto.

3.1 Interação Social

Como em toda rede social, o principal são os usuários, sem eles a plataforma fica sem conteúdo e perde seu sentido. Almeja-se incentivar a participação ativa e frequente dos usuários, para que os mesmos se sintam cada vez mais encorajados a opinar, postar e acessar o aplicativo.

As três principais atividades que se deseja do usuário é a de postar novas ofertas e curtir ou comentar as existentes Para incentivar essas ações, o aplicativo deve contar com formas fáceis e intuitivas de realizá-las, além da evidenciação dessa possibilidade em diversas telas. Além disso, é importante garantir a facilidade de login e registro no aplicativo, que pode ser obtido a partir a integração com métodos de autenticação de plataformas terceiras, como o Google, pois facilita e incentiva o processo feito pelo usuário, ao eliminar problemas como criar e gerenciar senhas, autenticar email e digitar manualmente todas as informações solicitadas.

Outro aspecto importante, é a fidelização do usuário, que pode ser obtida por uma experiência agradável de usabilidade e outros incentivos, como o reconhecimento dos que são mais ativos na plataforma, podendo se dar por disponibilização antecipada de funcionalidades, medalhas virtuais ou até mesmo descontos exclusivos [23]. Para a versão inicial, a única ferramenta viável seria a criação destas medalhas virtuais, em que apareceriam no perfil de cada usuário e seriam obtidas a partir de marcos como um certo número de postagens, comentários ou curtidas feitas pelo usuário.

3.2 Estrutura de backend para o servidor da API

O objetivo deste trabalho também inclui encontrar uma forma de fornecer toda essa estrutura a um custo reduzido sem perder sua performance. Para isso, um modelo de custos que seja diretamente proporcional ao número de acessos é essencial. Pois, ao ser publicado, o aplicativo pode buscar formas de receitas que também escalem de acordo com o número de acessos e visualizações, por meio de propagandas e programas de indicação, e basta que a receita dessa monetização seja superior ao custo de sua estrutura, ambas parametrizadas pelo número total de acessos.

O principal custo da plataforma é toda a parte relacionada a obtenção e armazenamento dos dados. Isto é, a API e o servidor em que se encontra o banco de dados [18]. Para isso foram estudadas algumas alternativas visando a performance e o baixo custo.

Uma máquina física, ou até mesmo uma VPS (Virtual Private Server) seria a solução mais clássica para a hospedagem de um servidor e o seu banco de dados [19]. Alguns de seus pontos positivos seriam a baixa latência entre o servidor e o banco de dados caso ambos estivessem alocados na mesma máquina. Além disso, a configuração da comunicação entre ambos, neste cenário, também seria facilitada.

Entretanto, esse é um modelo cada vez menos utilizado, por se tratar de uma configuração de difícil escalabilidade e que mantém uma capacidade uniforme tanto em horários de pico quanto horários de pouco acesso, o que resulta numa sobra de poder computacional nos horários vagos ou num gargalo nos horários de pico em serviços que não tem um acesso uniforme.

Por se tratar de um serviço oferecido exclusivamente a usuários brasileiros, é de se esperar que os horários de madrugada tenham pouco acesso, e por isso, utilizar essa

opção acabaria no problema relatado acima, isto é, pouca performance em horários de picos, e ociosidade nos horários de pouco acesso.

Uma outra opção, a que foi escolhida, é a dos serviços auto-escaláveis fornecidos tanto pelo Google quanto pela Amazon . Esses serviços são oferecidos em dois modelos:

O primeiro se trata de instâncias que são provisionadas sob demanda, podendo manter uma sempre ativa para garantir uma resposta mais rápida nos horários de pouco acesso. Nesse modelo, máquinas virtuais são “ligadas” conforme a necessidade, se tratando de um ambiente muito parecido com o da máquina física, porém com toda a estrutura de distribuição da carga já resolvida pela provedora.

O segundo é o modelo Serverless, que pode ser definido como uma plataforma que esconde o uso de servidores dos desenvolvedores e executa o programa sob-demanda, cobrando apenas pelo tempo de cpu utilizado [5]. Se encaixando, dessa forma, como a melhor alternativa para se estruturar um aplicativo cuja demanda inicial é nula, no estágio de desenvolvimento e que, no caso de se popularizar, possa alcançar um elevado número de acessos simultâneos.

Vale notar, que o modelo Serverless requer a atenção e a preocupação na modelação do sistema, por parte dos desenvolvedores, pois este impossibilita a utilização da memória principal para armazenamento de qualquer informação que possa ser necessária em outra chamada e dificulta a invocação de funções assíncronas de dentro do programa [21].

É essencial que a plataforma proposta consiga responder dinamicamente ao aumento ou diminuição de requisições, pois é de se esperar que estas não sejam uniformemente distribuídas ao longo das 24 horas de um dia. Além disso, como mencionado anteriormente, é importante que o custo esteja diretamente ligado ao número de acessos, permitindo também o desenvolvimento e o teste da infraestrutura a um baixo custo.

3.3 Interface do usuário

A interface do usuário é um dos pontos principais para esse aplicativo. Uma interface de baixa qualidade pode expulsar o usuário e levá-lo a buscar concorrentes.

A experiência do usuário está diretamente ligada a interface e ao que ele observa na tela de seu dispositivo. Uma experiência com muita lentidão pode exaurir o usuário, o excesso de informação pode confundi-lo ao invés de auxiliá-lo e uma rede social pode expô-lo a conteúdos indesejados.

A fim de minimizar os problemas mencionados, a interface deve contar com um visual limpo, balanceando a quantidade de informações e sua disposição, a fim de conter as principais informações necessárias ao usuário sem poluir a tela do seu dispositivo. A fim de evitar lentidão, a interface deve utilizar chamadas em *background* e assíncronas além de manter uma estrutura em cache, a fim de dar a sensação de instantaneidade ao usuário em interações como curtir e um carregamento progressivo em outras telas, a fim de exibir a informação que já possui e a medida que for possível carregando os elementos seguintes.

Quanto ao problema de conteúdos potencialmente ofensivos, por se tratar de uma rede social, o produto deverá, antes de sua publicação, possuir mecanismos que permitam usuários a denunciarem esses conteúdos e mecanismos automatizados que os identifiquem. Permitindo que estes conteúdos sejam rapidamente analisados e deletados a fim de minimizar o número de usuários expostos a estes.

O protótipo de baixa fidelidade, a seguir, foi criado a fim de ilustrar como o aplicativo deverá ficar em sua versão final, contando com todas as telas que um usuário comum poderá interagir.

3.3.1 Telas iniciais

Figura 1 - Telas principais do dispositivo para usuários não logados.

A tela à esquerda na Figura 1, é a tela que aparece assim que o usuário abre o aplicativo, exibirá as principais ofertas, ou como é chamado no aplicativo, os principais Descontos. O critério de exibição e ordenação pode ser alterado pelo usuário na caixa de seleção no canto superior direito.

A tela de login contará com diversas integrações para facilitar o cadastro e o login do maior número possível de usuários. E, caso ele prefira, poderá se cadastrar com o tradicional sistema de email e senha.

3.3.2 Telas de perfis

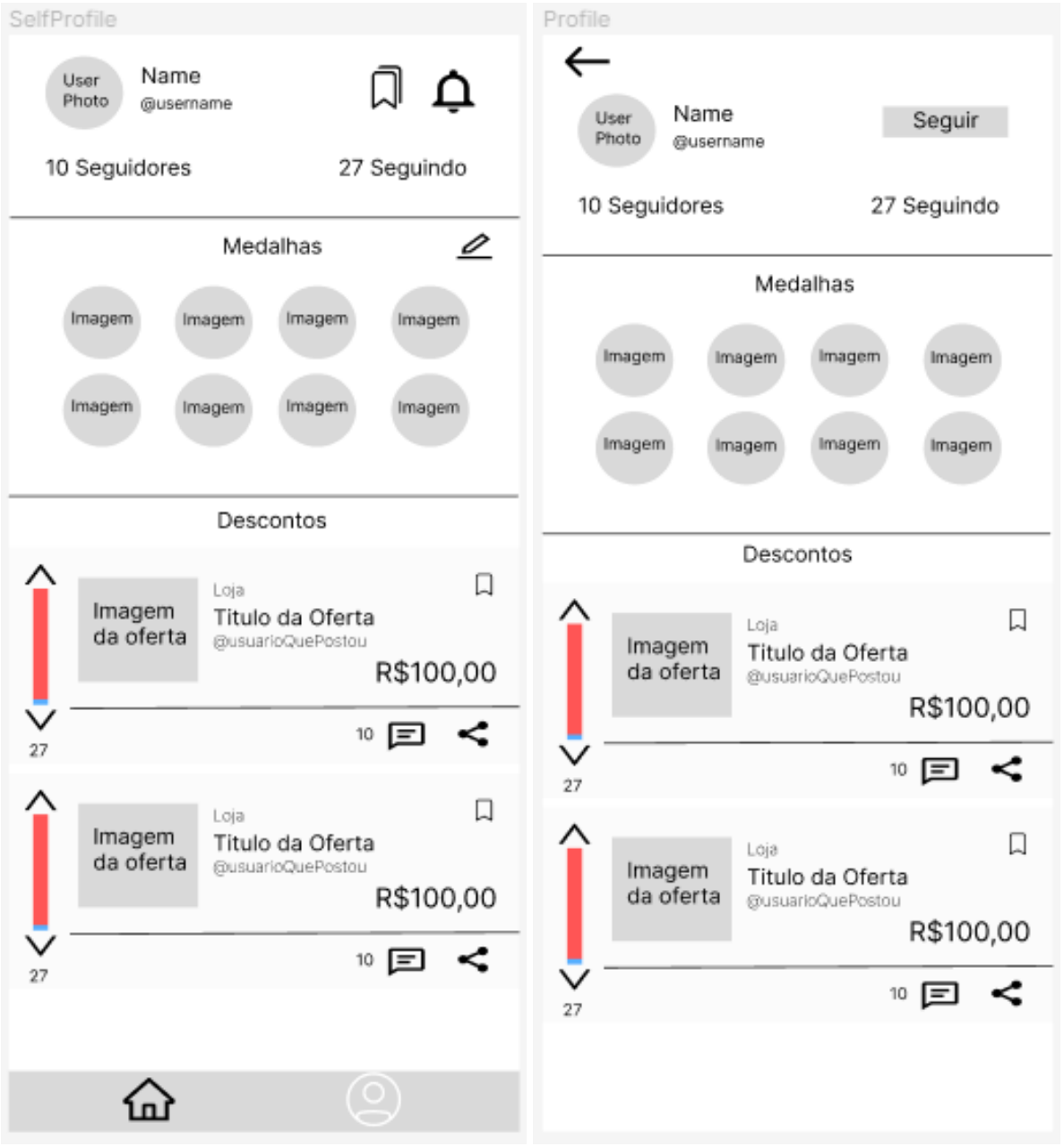

Figura 2 - Telas de perfil do usuário.

As tela de perfil tem duas variações: a primeira, da esquerda na Figura 2, é a visualização do usuário logado de seu próprio perfil, com opções de alterar as medalhas em exibições, ver as promoções salvas e as notificações recebidas. Algumas interações seguem padrões vistos em vários aplicativos e não são representadas visualmente. O usuário logado ao clicar no seu próprio nome ou imagem, deverá ver um menu com opções de alterar nome, foto e deslogar. À direita, é o perfil de um terceiro, que não o do usuário logado, nele as únicas interações possíveis são aquelas relacionadas às promoções e o botão de seguir ou deixar de seguir.

3.3.3 Telas de usuário logado

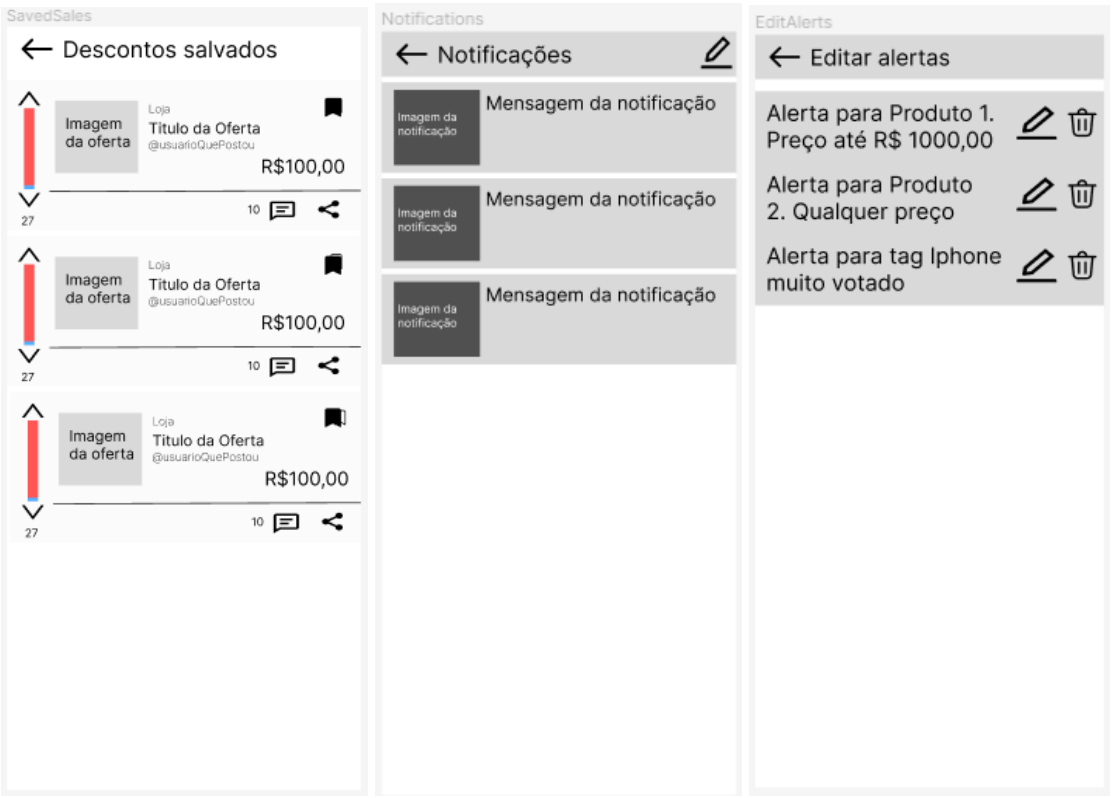

Figura 3 - Telas acessíveis apenas para usuários logados.

Estas outras telas aparecerão apenas para o usuário logado, e representam a visualização das promoções que o usuário salvou, com botões para deixar de salvá-las, as notificações recebidas, com opção de clicar nelas e ser redirecionado a uma promoção ou produto alvo da notificação, e os alertas configurados, com opção de editá-los ou apagá-los.

3.3.4 Telas de promoção e produto

Figura 4 - Telas de produto e promoção.

Ambas as telas seguem o mesmo padrão visual de manter a imagem em plano de fundo por debaixo de uma camada com borda curva contendo as informações. Foi escolhido esse padrão pela sua estética e por agregar a imagem do texto de forma mais natural. Além disso, o espaço ocupado pela imagem é aproveitado ao rolar a tela para baixo.

A tela de oferta contém as principais ações propostas pelo aplicativo. Nela o usuário pode compartilhar, abrir o link da promoção, salvar, votar se ele considera um desconto ou uma cilada, abrir outra oferta similar, comentar, reagir a comentários e abrir perfis de outros usuários. É esperada ser a segunda tela mais visualizada após o feed.

A tela de produto contém as informações pertinentes a ele, como marca, descrição e ofertas. Mas contém também várias variações do mesmo, que podem ser do tipo cor, tamanho de memória no caso de eletrônicos, voltagem no caso de eletrodomésticos, entre

muitas outras. Ao selecionar uma variação, caso haja uma modificação visual a imagem desta tela deve atualizar para corresponder à variação selecionada

3.3.5 Telas de busca e de marca

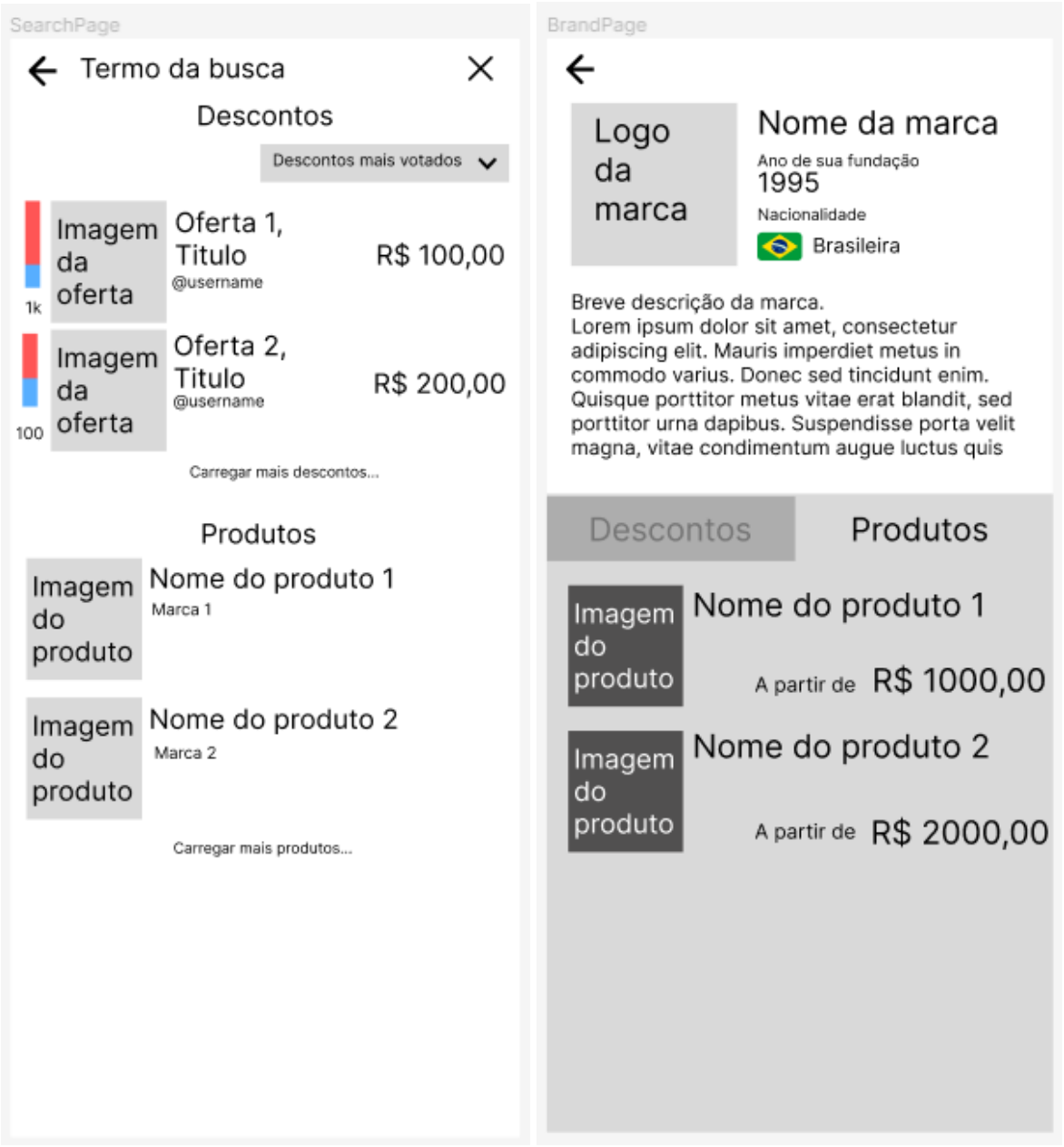

Figura 5 - Telas de busca e marca.

A tela de busca automaticamente faz a pesquisa pelos dois elementos principais do aplicativo, as ofertas e os produtos, e os exibe em uma mesma visualização. Para as ofertas, são exibidas versões reduzidas do que se tem no feed, para aumentar o número de visualizações, além disso não é necessário ter os botões de reações e compartilhamento que estarão disponíveis caso o usuário expanda clicando em uma delas. O usuário também pode escolher ver mais resultados para um ou outro tipo de resultado clicando no “Carregar mais descontos/produtos”.

A tela de marca permite a visualização das ofertas e produtos associados a ela, além de conter informações básicas sobre a mesma.

3.3.6 Tela de adicionar promoção

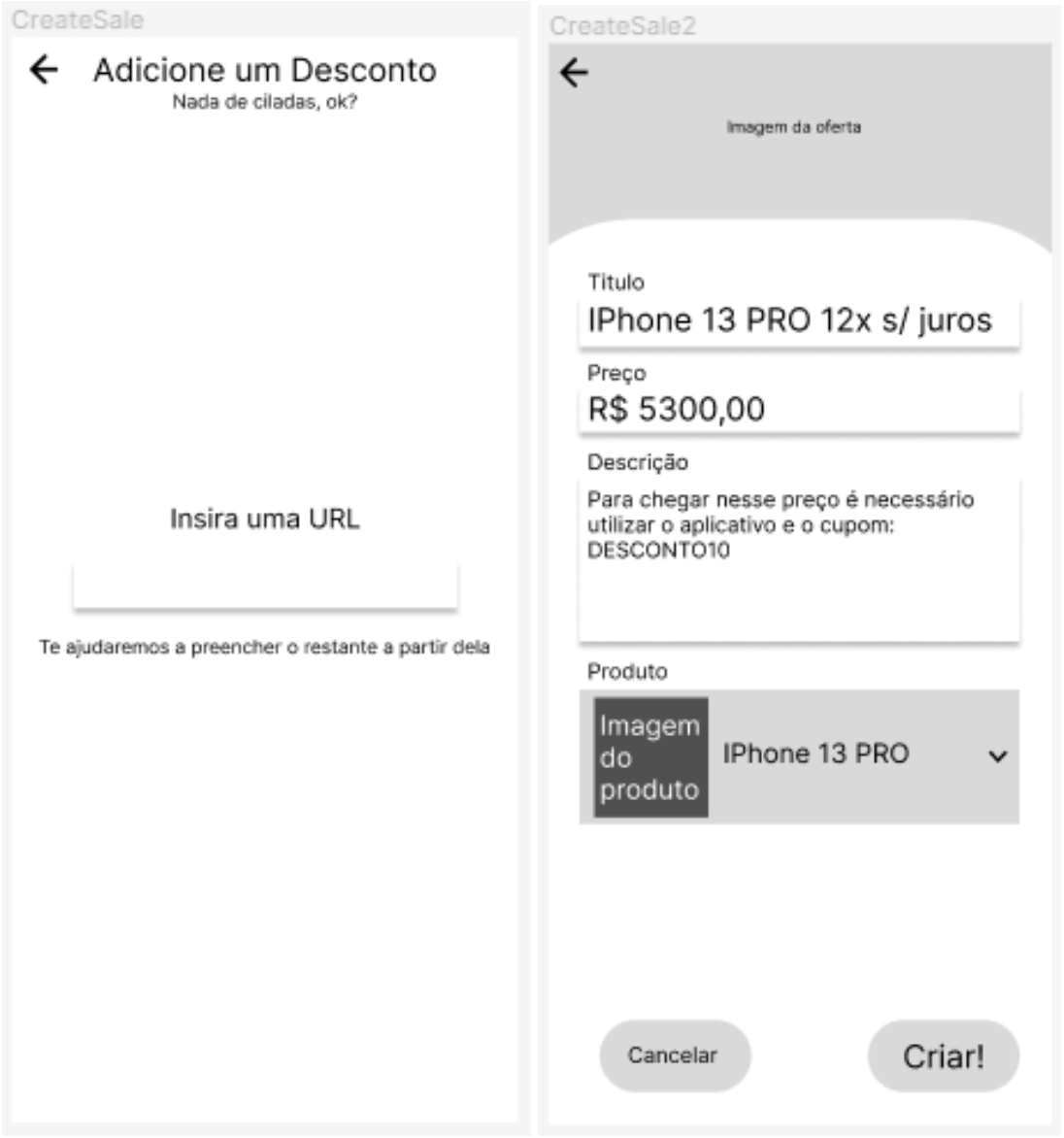

Figura 6 - Telas de adicionar promoção.

A tela de adicionar promoção tem como objetivo facilitar ao máximo o usuário a cadastrar uma nova promoção. Para isso, foi pensada uma lógica em que a partir da URL, o aplicativo pudesse extrair informações automaticamente, como título, preço, descrição e qual produto está associado, diminuindo o trabalho manual dos usuários e estimulando o compartilhamento de promoções.

Outra maneira do usuário enviar estas promoções, seria pela integração ao sistema de compartilhamento do celular, em que o aplicativo Desconto ou Cilada aparecesse com uma das opções junto a email, redes sociais e outros métodos integrados, ao clicar compartilhar em qualquer aplicativo de e-commerce. Por esse meio, o usuário poderia já cair nesta segunda tela e preencher ou corrigir as informações faltantes.

3.4 Banco de dados

O primeiro questionamento quanto ao modo de armazenamento dos dados foi em relação ao modelo utilizado. Os sistemas do tipo NoSQL têm ganhado cada vez mais espaço no mercado [6], devido à flexibilidade, escalabilidade e performance oferecidos pelo mesmo, grande parte devido à facilidade de particionar horizontalmente.

Entretanto, o SQL facilita a execução de consultas mais complexas, permitindo a extração de relatórios e insights de forma facilitada. Como um dos pontos principais da plataforma é o poder de agregar as informações por produtos e lojas, além de gerar insights a partir disto, como por exemplo um gráfico com a evolução do preço de algum produto, o SQL facilitará e trará performance a esse tipo de consulta.

No caso do aplicativo ser bem sucedido, é importante também poder escalar a performance do banco de dados para que o mesmo dê conta em momentos de pico na utilização. Para isso, pode-se utilizar um sistema de caching em consultas corriqueiras, como a exibição das ofertas mais populares ou então uma busca comum por algum produto ou promoção. Entretanto, em interações como a postagem de comentários e curtidas, necessitam a interação com o banco de dados todas as vezes.

3.5 Diferenciais

Outro aspecto fundamental, são os diferenciais que a plataforma tem quando comparada aos seus concorrentes. Como já existem algumas ferramentas próximas à ideia proposta, é fundamental que o Desconto ou Cilada ofereça mais funcionalidades ou melhorias em performance, regras de negócio e interface.

Os principais diferenciais propostos se resumem a criar um ambiente imparcial quanto aos tipos de descontos relatados, isto é, permitir o compartilhamento de ofertas que não gerem retorno nenhum à plataforma, permitir a busca por produtos exibindo as ofertas deles, assim como mapear as diferentes variações dos produtos como, por exemplo, cor e tamanho de armazenamento no caso de celulares.

3.1.1 Concorrentes

Primeiramente, um diferencial em relação a quase todas as plataformas de promoções é não impedir ou prejudicar a divulgação de promoções que não gerem retorno financeiro ao aplicativo ou ainda que referenciam um concorrente. Essa prática é adotada principalmente para garantir um maior retorno financeiro, pois as plataformas ganham com as compras realizadas a partir de usuários que vieram delas e, ao permitir links de outras plataformas, esse retorno financeiro seria dado aos concorrentes.

A fim de focar no usuário, e em sua experiência, tendência cada vez mais adotada pelas empresas, o Desconto ou Cilada não deverá limitar as promoções e dicas postadas pelos usuários, podendo utilizar programas de afiliação quando disponíveis mas não os exigindo. A receita por propagandas, através de plataformas como Google Ads, no aplicativo deverá ser o suficiente para custeá-lo, uma vez que uma das preocupações é garantir um baixo custo na manutenção desta plataforma.

Quando comparado com o Pelando, a principal diferença pode ser resumida em categorizar e identificar os produtos referenciados nas ofertas, dessa forma, será possível criar telas para que os usuários vejam todas as promoções e preços atuais do produto mencionado, além do histórico recente de preços e avaliações dos usuários.

Já em comparação com as outras plataformas do grupo Mosaico, as principais diferenças consistem justamente no elemento social e mais flexível de se considerar preços. Não haverá restrições quanto ao link para lojas, como Aliexpress, Shopee ou outros varejos alternativos e nem quanto ao tipo de promoção, como as que ocorrem por meio de cupons ou cartões e clubes de fidelidade.

4 Projeto e Implementação

Finalmente, nessa seção, apresenta-se o plano e a execução com o objetivo de atingir um MVP (Minimum Viable Product) da solução proposta, evidenciando o potencial oferecido frente aos concorrentes e a sustentabilidade de um projeto desse porte.

Foram necessários alguns ajustes nos requisitos e objetivos mencionados anteriormente, a fim de se obter um produto entregável dentro do prazo estipulado por um trabalho de conclusão de curso. Nesta seção, serão tratadas os requisitos cortados e mantidos para a versão apresentada à banca, além do detalhamento das implementações realizadas.

Por existirem uma infinidade de produtos na internet, não seria viável catalogar todos eles, e qualquer solução automática para isso também seria demasiadamente complicada. Portanto, serão catalogados alguns produtos nas quatro principais categorias, ordenadas pelas receitas totais: Telefonia, eletrodomésticos, entretenimento e TI [4].

Idealmente, seria interessante ter tal plataforma tanto em dispositivos móveis quanto em computadores. Entretanto, o desenvolvimento será focado nos dispositivos móveis com sistema operacional Android, utilizando Flutter para garantir a usabilidade futura tanto nos sistemas iOS quanto Android. Esta escolha se dá em razão da crescente popularidade de smartphones e a possibilidade de criar experiências nativas nestes dispositivos, como notificar o usuário onde quer que ele tenha levado seu dispositivo. O foco em Android decorre da facilidade em testar, uma vez que não possuo os dispositivos necessários da Apple para o desenvolvimento de aplicações mas, por ser um framework multiplataforma, o mesmo código escrito nesta implementação pode ser facilmente adaptado para ser executado em dispositivos iOS.

Para o servidor, será utilizado Python com a biblioteca Flask, por ser uma biblioteca consolidada que oferece a criação simples de rotas e urls, além de ser compatível com o framework Zappa. O crawler também será em Python, facilitando futuras integrações com o servidor, e será utilizada a biblioteca Scrapy, que já oferece suporte a execução paralela.

Finalmente, para o banco de dados será utilizado PostgreSQL, por ser um gerenciador de banco de dados relacional gratuito e já estabelecido no mercado, com suporte a diversas funcionalidades como gatilhos, funções, visões materializadas e por ser suportado pelo AWS Aurora, que será utilizada como host para o mesmo.

4.1 Banco de dados

4.1.1 Modelo de Dados

Dois modelos de dados foram elaborados neste projeto. O primeiro é orientado ao produto final e o segundo representa o estado alcançado durante o desenvolvimento do mesmo. Muitas funcionalidades foram cortadas e diminuídas a fim de alcançar um produto considerado como mínimo produto viável (MVP).

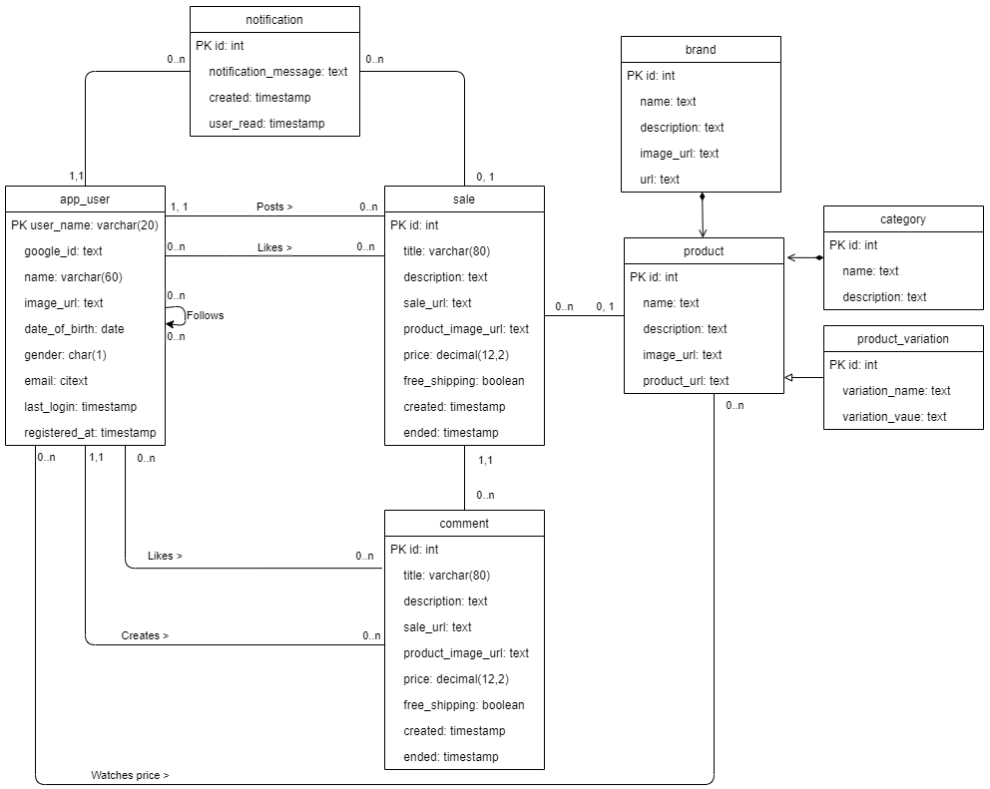

Figura 7 - Diagrama da versão final proposta.

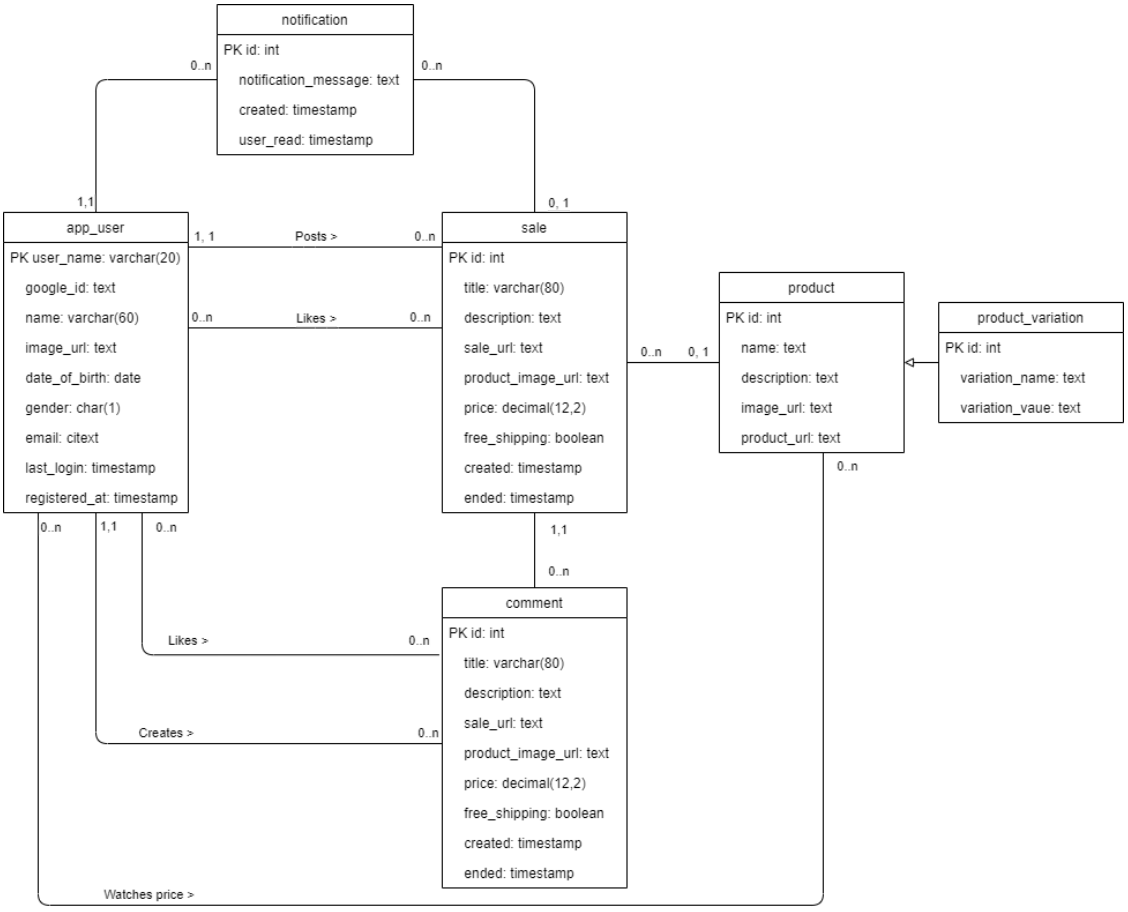

Figura 8 - Diagrama da versão implementada.

O plugin citext do Postgres foi usado para definir o que é email por uma expressão regular que garante a conformidade deste campo a nível de banco de dados, e o plugin pg_trgm foi utilizado a fim de possibilitar a busca por similaridade de palavras, auxiliando na busca de usuários por produtos e ofertas.

As contas de servidor e crawler também foram criadas, de forma a diminuir o acesso de cada um destes serviços e adicionar uma camada de segurança que pode minimizar o impacto caso qualquer uma destas seja comprometida.

4.1.2 AWS Aurora

Visando ambos alcançar tanto a performance quanto a estrutura SQL conveniente a extração de informações a respeito dos dados, escolhi utilizar o banco de dados Aurora da AWS.

Aurora é um gerenciador de banco de dados completamente gerenciado e compatível com PostgreSQL e MySQL, que combina a performance e confiança de gerenciadores comerciais com o custo benefício e simplicidade dos bancos de dados open-source [7].

Mesmo recebendo um único nome, o AWS Aurora possui diversas configurações completamente diferentes entre si, tendo como principal característica ser um sistema em Cluster com os dados também em um cluster de volumes auto-replicável, sendo oferecidos em dois tipos de cluster: o *single-master* e o *multi-master*.

No multi-master, todas as instâncias podem realizar as operações chamadas de Data Definition Language (DDL), isto é, alterações estruturais, como criar tabelas, e de Data Manipulation Language (DML), isto é, alterações dos dados. Cabendo à aplicação ou usuário o cuidado com a concorrência para que duas instâncias não escrevam sob o mesmo conjunto de dados ou esquema ao mesmo tempo.

No modelo serverless, oferecido apenas na arquitetura single-master, cada instância possui a capacidade de aumentar ou diminuir os seus recursos durante a execução. A AWS denomina esses recursos como Aurora capacity unit (ACU). Cada ACU possui 2GB de memória, mas a AWS não especifica quanto cada ACU representa em poder de processamento, dizendo apenas que escala o poder da rede e de processamento de forma correspondente à memória RAM. Cada instância pode variar em múltiplos de 0.5 ACUs tendo no mínimo 0.5 ACU no limite inferior e 128 ACU no limite superior. Além disso, um cluster pode ter inúmeras instâncias leitoras que podem ou não escalar junto da instância escritora. As instâncias leitoras que escalam junto da escritora são prioritárias para assumir o papel da escritora caso a mesma se torne indisponível, pois elas já terão os recursos provisionados na mesma quantidade [7].

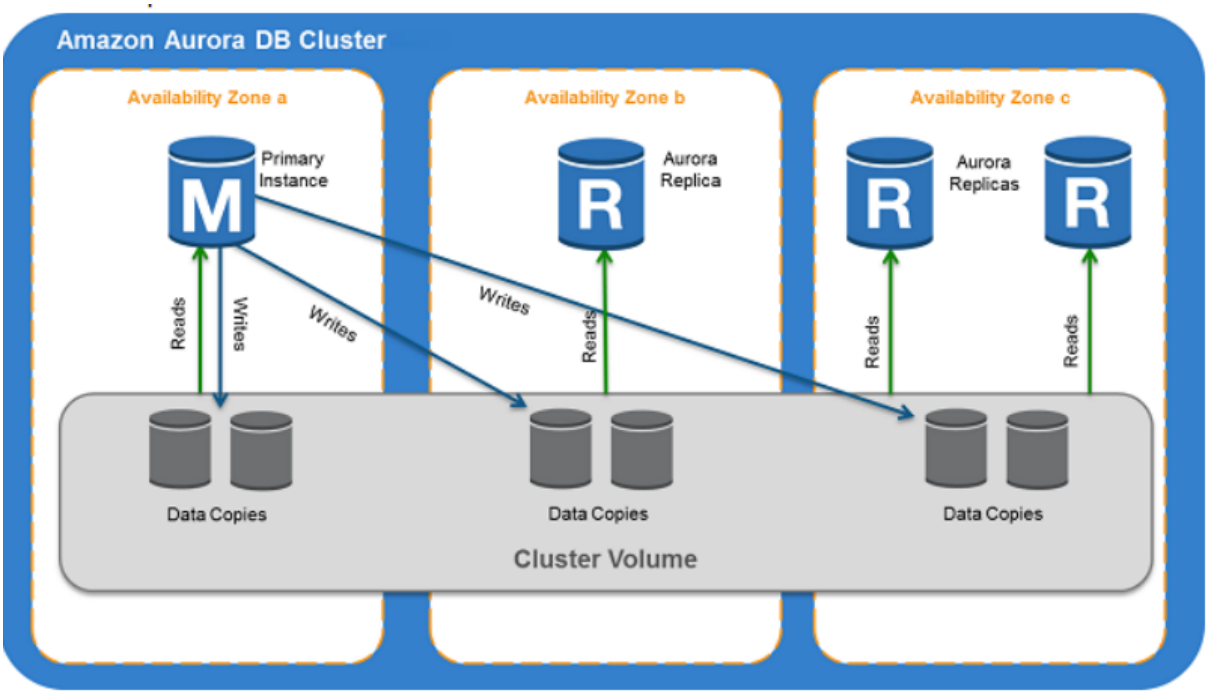

Figura 9 - Diagrama da AWS Aurora [7].

A aplicação proposta provavelmente terá mais solicitações de leitura do que de escrita, pois é de se esperar que os usuários vejam muito mais ofertas e pesquisem mais do que interajam curtindo, comentando ou postando. Dessa forma, caso se prove necessário, é possível separar a aplicação para que as instâncias leitoras sejam utilizadas sempre que possível enquanto a escritora fique encarregada apenas da parte de escrita e provisionar uma instância grande para lidar exclusivamente com a parte de escrita.

Além disso, é de se esperar que o uso oscile diária e ciclicamente em certos horários e dias, uma vez que o aplicativo estará restrito a usuários brasileiros, que estão sob apenas 3 fusos diferentes, e por apresentarem costumes similares no quesito de horários. Dessa forma, a economia prometida ao escalar para baixo a utilização de recursos nesses momentos é de grande importância para o quesito custo.

Portanto, pelas definições e argumentos acima, escolhi utilizar o AWS Aurora configurada como um cluster serverless possuindo apenas uma instância leitora, num primeiro momento, como solução para o armazenamento dos dados. Durante o desenvolvimento, esta única instância foi configurada para variar de 0.5 a 1 ACU, a fim de minimizar os gastos enquanto o aplicativo ainda não é disponibilizado ao público. Vale ressaltar que alterar esses valores e criar instâncias é extremamente fácil.

4.2 API

Como mencionado anteriormente, a API foi desenvolvida em Python utilizando o framework Flask e pensada como uma API serverless.

4.2.1 AWS Lambda

Visando comportar um possível crescimento na utilização do aplicativo, com picos imprevisíveis, o modelo serverless se apresentou como o mais adequado. Além da escalabilidade sob demanda, o seu custo também é dado pela utilização dos recursos, o que permite baixíssimo custo durante o desenvolvimento, e até gratuito.

A AWS, por ser a maior provedora desse ramo [8], além de possuir extensa documentação e um framework que auxilia a configuração com serviços feitos em Python e Flask, se provou a melhor escolha.

No quesito custos, AWS, Google e Azure oferecem planos muito semelhantes com até 400,000 GB-Segundos por mês de forma completamente gratuita. Entretanto o modelo de cobrança do Google arredonda as requisições para unidades de 100ms enquanto os outros dois apenas para 1ms, outra vantagem da AWS e Azure contra a Google é o menor preço pelo número de requisições [9].

Como contraponto ao modelo de cobrança, está a dificuldade de estimar o custo previamente, como seria facilitado por modelos mais tradicionais como servidores físicos ou virtuais. Este obstáculo, entretanto, é superado conforme o uso e a cobrança dos recursos utilizados, com uma base histórica desses dados, fica fácil estimar os próximos meses e comparar os custos com os recursos tradicionais mencionados..

4.2.2 Autenticação

Inicialmente, tinha sido utilizada a biblioteca flask-login para lidar com a autenticação de usuários. Entretanto a mesma era orientada a sessões webs e mantinha os tokens em memória local, o que não é compatível com a arquitetura serverless.

Inspirado nesta biblioteca, desenvolvi uma solução com o mesmo padrão e interface, bastando ao longo do programa utilizar um decorator em cima das funções que requerem login para serem executadas, este decorator intercepta a requisição antes da mesma entrar na função referida, autentica o usuário e, em caso de falha, retorna o código HTTP 401 - Not Authorized.

Para autenticar a requisição, verifica-se se a mesma possui o header contento o token previamente emitido pela aplicação após um login bem-sucedido. Para validar o token utiliza-se o padrão conhecido como JSON Web Tokens (JWT), que nada mais é do que uma forma compacta e segura para URL de transferir reivindicações, podendo ser criptografado com diferentes algoritmos de autenticação e ou criptografia, tanto simétrica quanto assimétrica [13]. Este padrão define os nomes e campos que podem ser utilizados na criação dos tokens, todos opcionais, com exceção do campo de expiração (exp), que determina a data e horário a partir da qual o token não deverá mais ser aceito. A seguir estão os campos escolhidos para esta aplicação [13]:

- Issued At (iat): Data e horário da emissão do token
- Expiration Time (exp): Único campo obrigatório pelo padrão JWT, contendo a data e horário da expiração do token.
- Subject (sub): O sujeito alvo do token, deve ser único no escopo aplicado.

No quesito do algoritmo utilizado, foi escolhido o “HS256”, por ser um algoritmo simétrico padrão na implementação da maioria das bibliotecas de criptografia e ser suficientemente seguro. Além disso, a escolha por algoritmo simétrico se dá pelo fato das informações contidas no token serem necessárias apenas ao servidor, isto é, o aplicativo não precisa decodificar nem validar o conteúdo do mesmo.

Este algoritmo utiliza uma chave secreta discricionária, no caso do aplicativo será gerada uma aleatória manualmente. Caso haja algum erro na “assinatura do token” ou o mesmo já esteja expirado, a autenticação falha. Uma etapa adicional serve para verificar se o token foi cancelado após sua emissão e antes de sua expiração. Necessário para implementação segura da funcionalidade de logout. Uma vez que se o usuário estiver em um ambiente inseguro, basta ele deslogar para que o acesso por aquele dispositivo seja

impossibilitado até um novo login. O que não seria verdade se apenas excluíssemos o token do lado do cliente, uma vez que, em um ambiente inseguro, esta informação poderia ter sido vazada ou armazenada por fora da aplicação.

Para essa validação contra tokens cancelados, o padrão é utilizar uma lista dos que o foram. Entretanto, isso significa manter uma lista infinitamente crescente ou um cuidado em retirar desta lista os tokens que naturalmente forem expirando. Para otimizar tanto a performance quanto o espaço armazenado, a solução implementada armazena uma informação contida no token válido em um banco não relacional do tipo chave valor, com chave sendo o nome de usuário e o valor sendo o iat (data e horário da criação do token). Esta informação é especialmente útil para isto, pois ela é única e infinitamente crescente, o servidor nunca irá gerar dois tokens ao mesmo instante e nem um mais antigo do que outro já gerado, pois ele estará sempre utilizando o horário de São Paulo. Desta maneira, ainda cria-se uma barreira adicional, caso algum agente mal intencionado consiga acesso ao segredo do servidor, seria necessário ainda descobrir o iat de um token válido de qualquer usuário para gerar um falso token crível ao servidor ou realizar um ataque de brute-force para tentar adivinhar este valor.

Para armazenar essa informação nesta etapa extra da validação, utilizei o AWS Dynamo, uma vez que o mesmo está na mesma provedora dos outros serviços que estou utilizando e possui uma latência extremamente baixa, comparável a manter a informação em memória primária, por se tratar de um banco não-relacional do tipo chave-valor e pelo seu uso ser apenas feito na consulta exata pela chave e na inserção de novos valores.

A estrutura geral fica da seguinte maneira: Quando um usuário realizar um login bem sucedido com suas credenciais, atualmente apenas o token do Google, o servidor validará este token junto ao Google e emitirá um token JWT com as informações de criação (iat), expiração (exp) e nome do usuário (sub), e colocará o iat no banco de dados não relacional, com a chave sendo o nome do usuário. Caso já exista a chave no banco, o valor será sobrescrito com o novo valor da criação do token.

Ao realizar logout, o registro chave-valor do usuário no banco não-relacional será excluído, e automaticamente o token antigo não será mais utilizável.

Dessa forma, o uso máximo do banco será o número de usuários. Em contrapartida, cada usuário fica limitado a ter um único login ativo simultaneamente e esta lógica de login terá que ser repensada se algum dia for necessária a possibilidade de se manter dois logins simultâneos. O diagrama a seguir, Figura 10, representa todo esse fluxo.

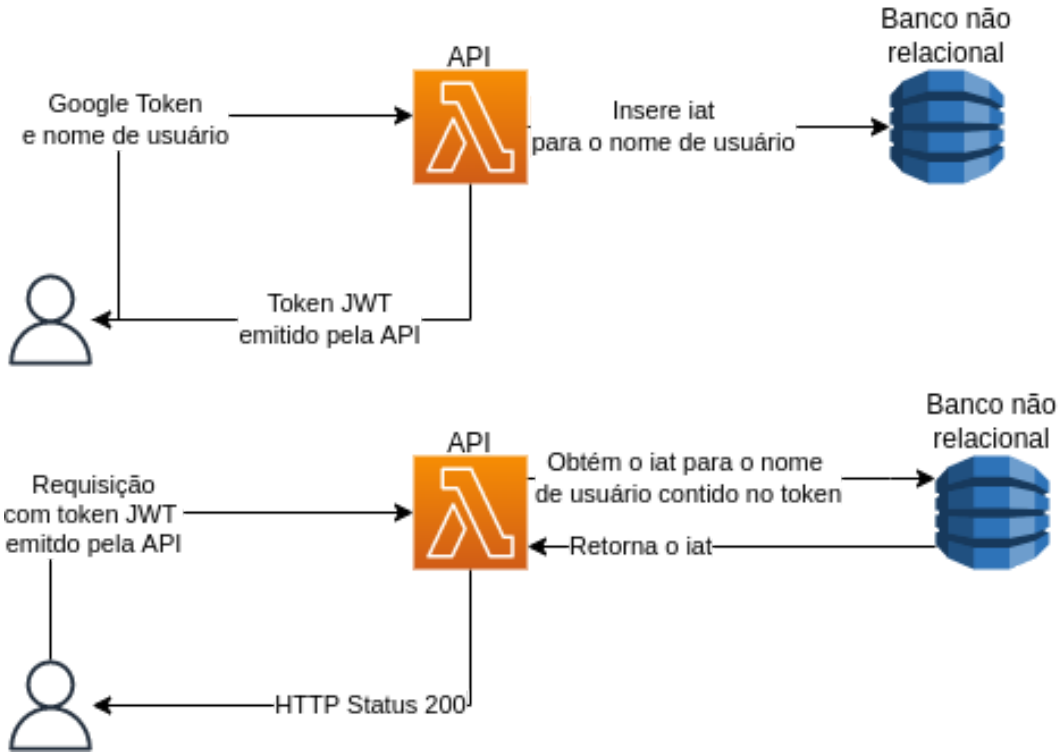

Figura 10 - Diagrama ilustrando o fluxo de autenticação.

4.3 Crawler

Na versão final, a plataforma deverá contar com a utilização de crawlers para atualizar de forma automatizada algumas informações. Alguns sites de e-commerce agrupam produtos e listam as ofertas dos marketplaces afiliados, outros possuem ofertas oficiais de produtos. Em ambos os casos, é possível e interessante extrair essas informações de forma automatizada a fim de rastrear o preço de produtos. A parte social de inserir manualmente promoções não é substituída por essa busca automatizada, uma vez que cupons e promoções de cashbacks não são identificadas desta maneira.

Um modelo inicial de crawler foi desenvolvido para o primeiro caso citado acima, na plataforma do MercadoLivre. Com o link de um produto o crawler desenvolvido é capaz de obter o nome do produto, a descrição, as imagens e a categoria. Além disso, o crawler também consegue ir para a página de ofertas do referido produto no MercadoLivre e extrair o nome da loja, o número de vendas nos últimos dias para tal loja, o preço e se o frete é grátis ou não para cada oferta.

Para o desenvolvimento deste crawler foi utilizada a biblioteca Scrapy em Python, um framework que auxilia na construção de crawlers, embutindo métodos para extração de informações por XPath, selectores de CSS, download automatizado de mídias e geração automática de relatórios [10]. E associado a este framework, para navegar pelas páginas HTML das lojas, foi utilizado XPath, uma notação flexível para selecionar diferentes partes de um documento em qualquer linguagem com formatação XML [11]. Além disso, o framework facilita a criação de variações. Como cada e-commerce constrói o website de uma maneira diferente, é necessário que, para cada um, o código seja manualmente modificado a fim de ser capaz de obter as mesmas informações a partir de estruturas completamente diferentes.

4.4 Interface

A interface foi um dos maiores desafios. A usabilidade do usuário é cada vez mais importante, sendo fator determinístico no sucesso ou fracasso de aplicativos. Como mencionado anteriormente, o framework escolhido para o desenvolvimento da interface foi o Flutter, por dar suporte ao desenvolvimento multiplataforma além de ter um grande potencial, por estar sendo cada vez mais utilizado e por ser desenvolvido pela gigante da tecnologia, Google.

Devido ao tempo limitado, optamos pela priorização das funcionalidades de login, agrupar ofertas por produtos, que é um dos maiores diferenciais, adicionar comentários, adicionar promoções, receber notificações e adicionar alertas de preço. Outras funcionalidades, menos prioritárias, infelizmente não foram concluídas, como as de salvar promoções e seguir usuários, e outras algumas foram reduzidas, por exemplo votar promoções como desconto ou cilada, aludindo ao nome do aplicativo, foi substituído por um botão básico de curtir.

Para o gerenciamento de estados, foi utilizado o padrão “BLoC” e a biblioteca “flutter_bloc”. O padrão BLoC foi criado pela Google para auxiliar na separação entre a parte visual e a parte lógica do código e reforça paradigmas de cada interação com o usuário gerar um estado para aplicação [15].

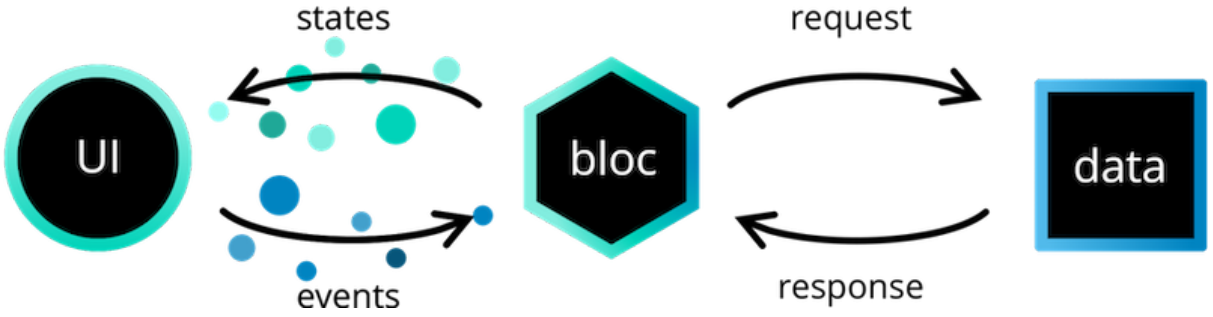

Figura 11 - Diagrama do padrão BLoC [14].

Neste diagrama dá para notar a ideia central desta separação de deveres. No extremo da esquerda, está a interface que gera os eventos, seja através de interações com o usuário ou por mudanças no ambiente como a perda de sinal de telefonia. Esses eventos são transmitidos então à parte lógica, “BLoC”, que pode requisitar dados, sejam eles locais ou externos como em bancos de dados dedicados, e retorna estados à interface. A interface é construída então, em cima dos estados, abstraindo a parte que está contida no “BLoC” e na fonte dos dados. Vale notar, que a idéia do “BLoC” é poder emitir estados antes de sua requisição por dados, quando necessária, terminar, desta forma a interface pode exibir indicadores de carregamento. Além disso, essa estrutura facilita também o tratamento de “Streams” de dados e eventos. A parte lógica e central, chamada “BLoC” foi pensada justamente nestes casos mais complexos, onde a geração de eventos não espera um estado para poder prosseguir, mas funciona por meio de fluxos, onde muitas coisas estão acontecendo simultaneamente.

Esse paradigma também auxilia no desenvolvimento de interfaces mais fluidas e responsivas, pois estimula a criação de estados em todas as etapas de interação com o usuário, sendo a espera por dados um estado próprio que deve ser, também, representado em tela.

4.4.1 Telas iniciais

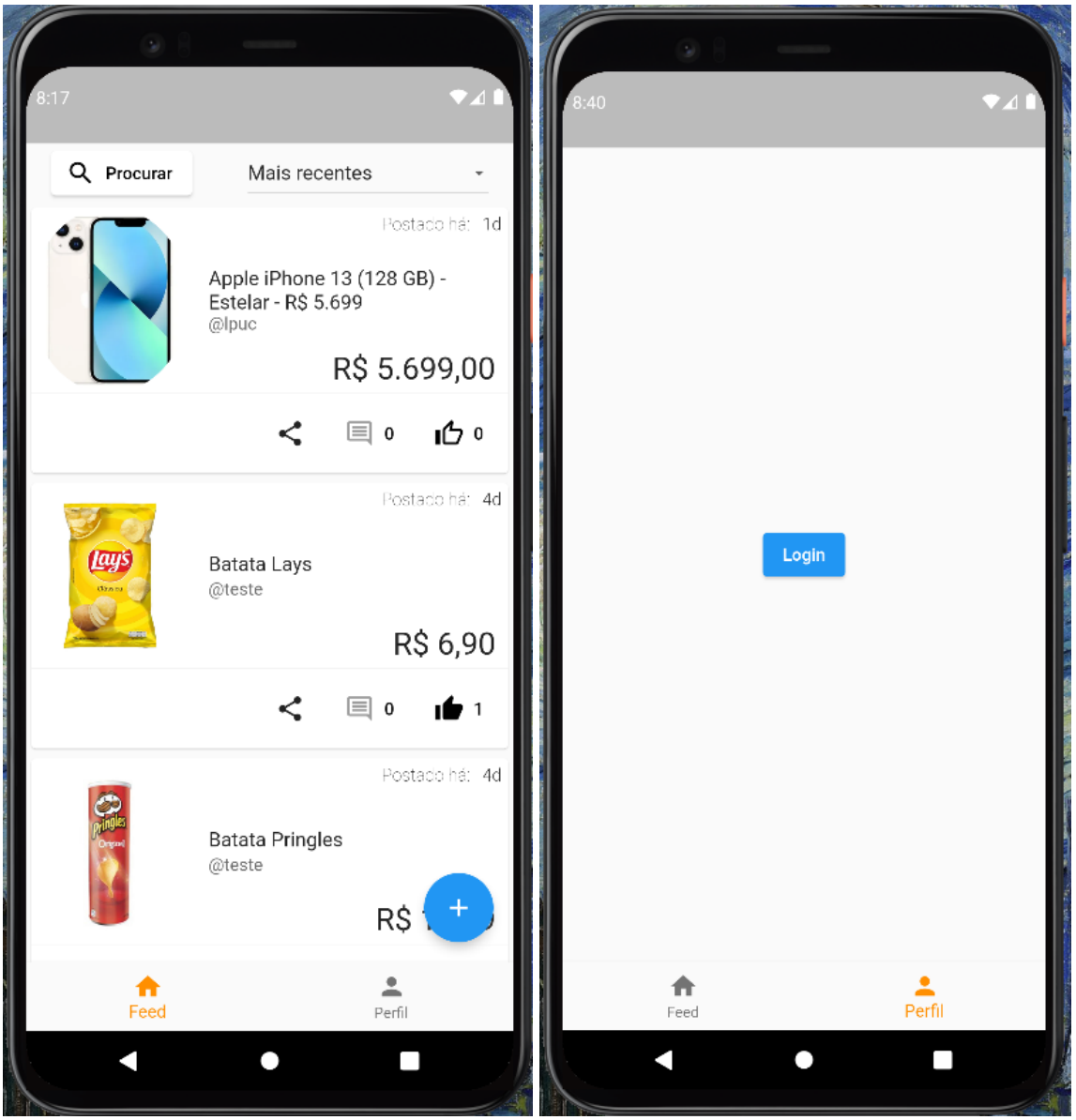

Figura 12 - Telas iniciais implementadas.

A tela de feed, primeira à esquerda na Figura 12, ficou próxima ao almejado pelo protótipo, com exceção da parte de votação, que foi simplificada para ser uma simples reação de curtir. Para ficar igual ao protótipo irá demandar um trabalho mais avançado e demorado, e por conta do tempo limitado, foi um dos vários detalhes cortados ou simplificados.

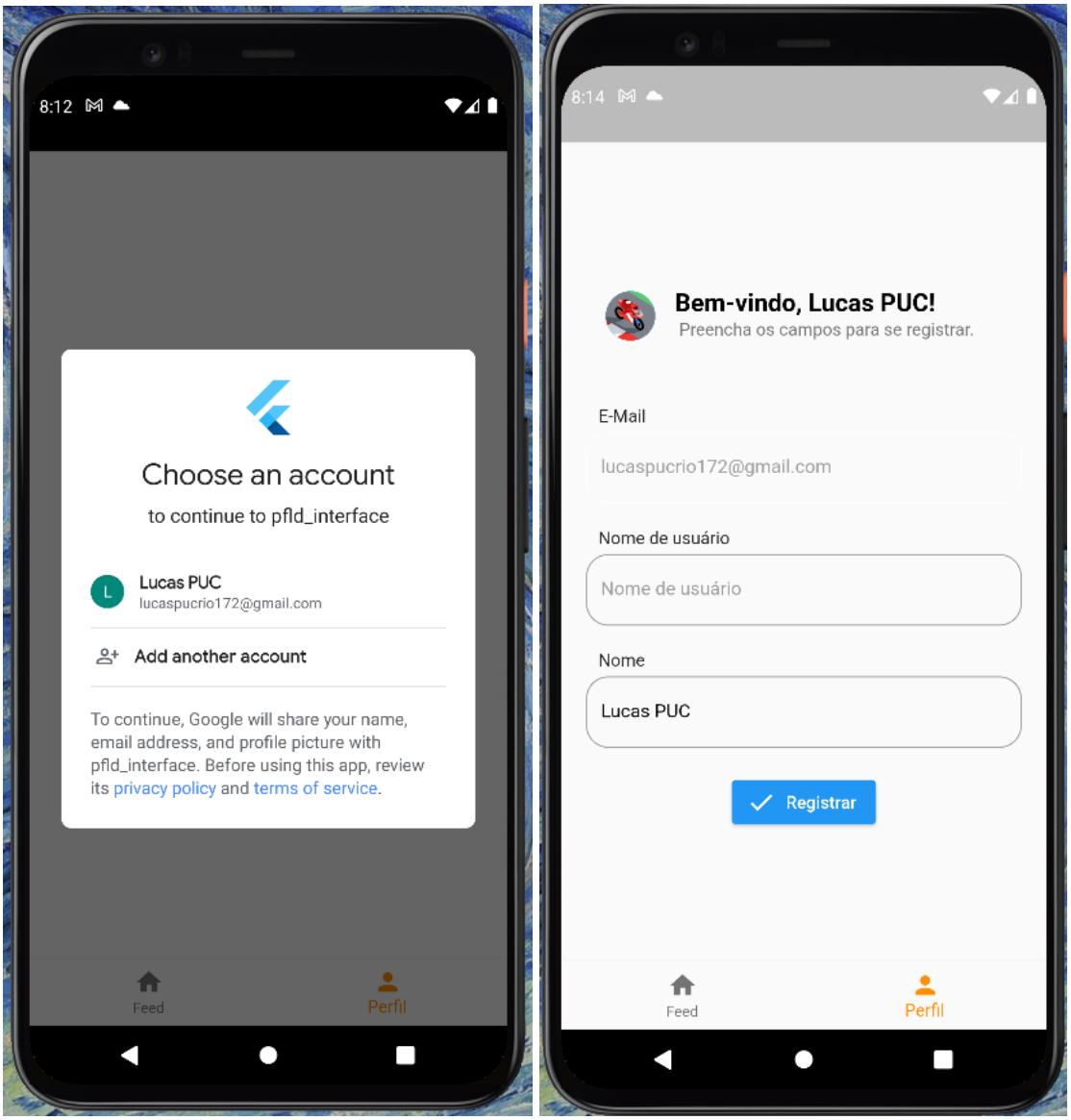

Figura 13 - Telas de login e registro implementadas.

A tela de login também perdeu várias integrações e o único login ou cadastro possível é por meio do Google. A tela de registro, por ter sido acessada a partir do login do Google, já preenche as informações de e-mail e nome, faltando apenas o usuário definir o seu nome de usuário, um identificador único que será exibido em suas postagens e comentários.

4.4.2 Telas de perfis

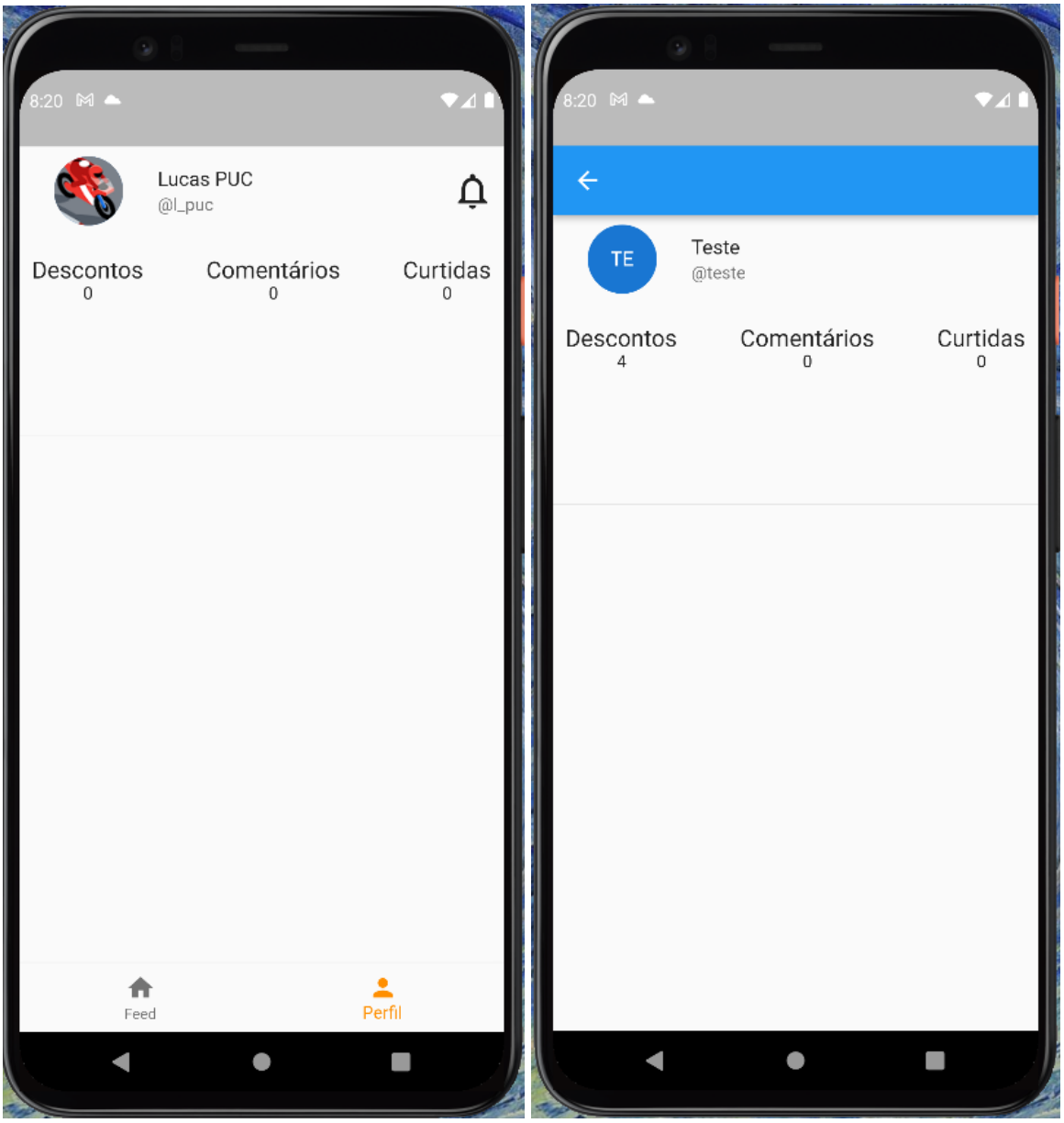

Figura 14 - Telas de perfil implementadas.

As telas de perfis não contam com a visualização dos posts nelas, e nem com as medalhas, que não chegaram a ser implementadas. Em contrapartida estatísticas visuais de posts, comentários e curtidas feitos pelo usuário. A tela de outro usuário não contém o botão perfil, como no protótipo, pois esta funcionalidade também não foi implementada.

4.4.3 Telas de usuário logado

Figura 15 - Tela de notificação implementada.

As notificações seguiram o padrão simples do protótipo e exibem as mensagens das últimas notificações. A tela de editar os alertas não foi implementada.

4.4.4 Tela de promoção

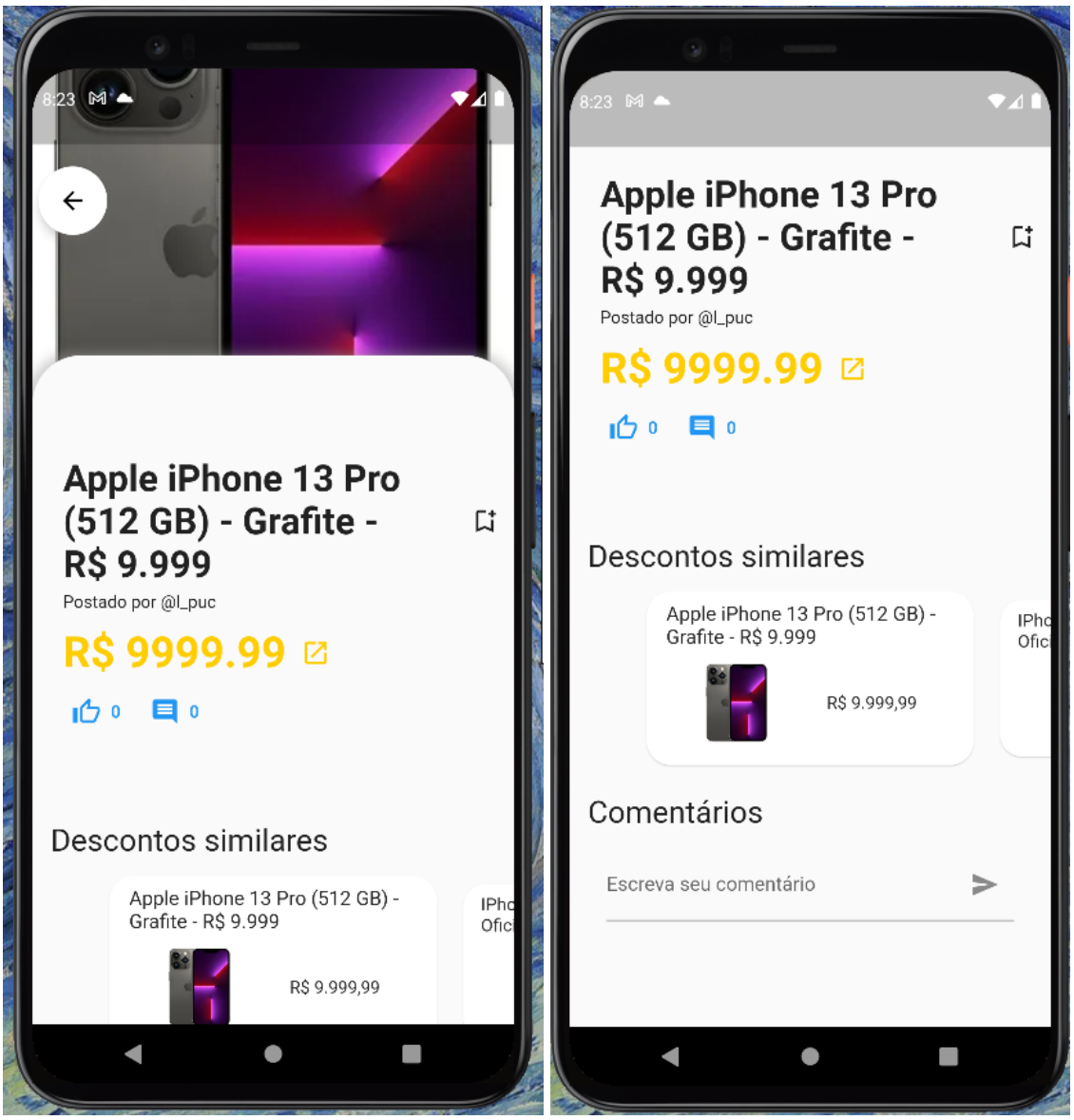

Figura 16 - Tela de promoção implementada.

A tela de promoção ficou bem próxima da versão do protótipo contendo todas as funcionalidades, com exceção de salvar, implementadas. A partir dela o usuário pode curtir a promoção, comentar, curtir comentários, visualizar promoções semelhantes e abrir os perfis de outros usuários. O usuário também pode, ao clicar na imagem, expandir a visualização da mesma em tela cheia.

4.4.5 Tela de produto

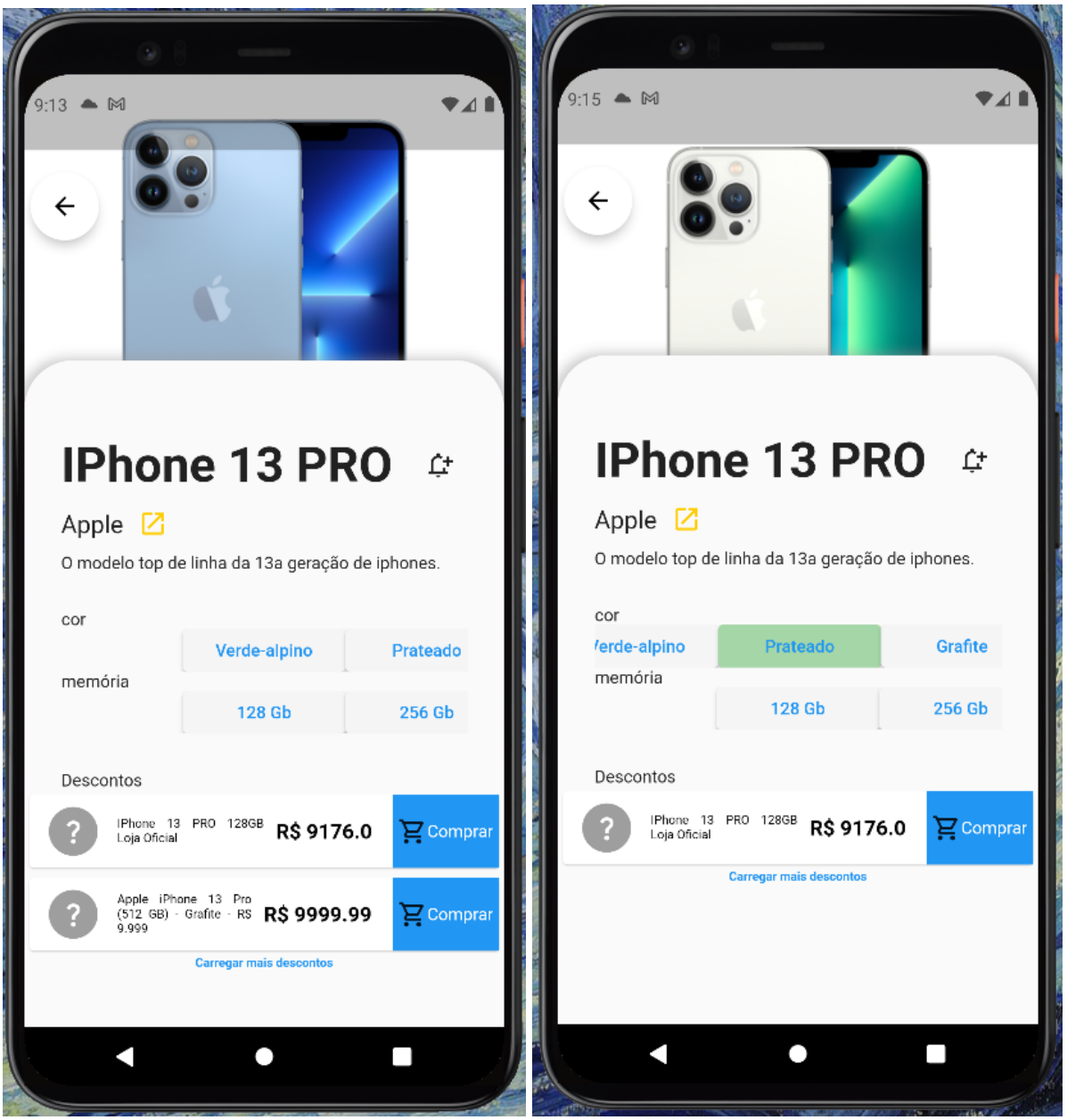

Figura 17 - Tela de produto implementada.

A tela de produto também ficou próximo do protótipo, contendo a opção pro usuário selecionar variações, que alteram a imagem contida na tela e as ofertas exibidas. As imagens que estão faltando ao lado do título das ofertas, eram para ser os logos das lojas, porém, não foi possível implementar a tempo da entrega deste relatório.

4.4.3 Telas de busca

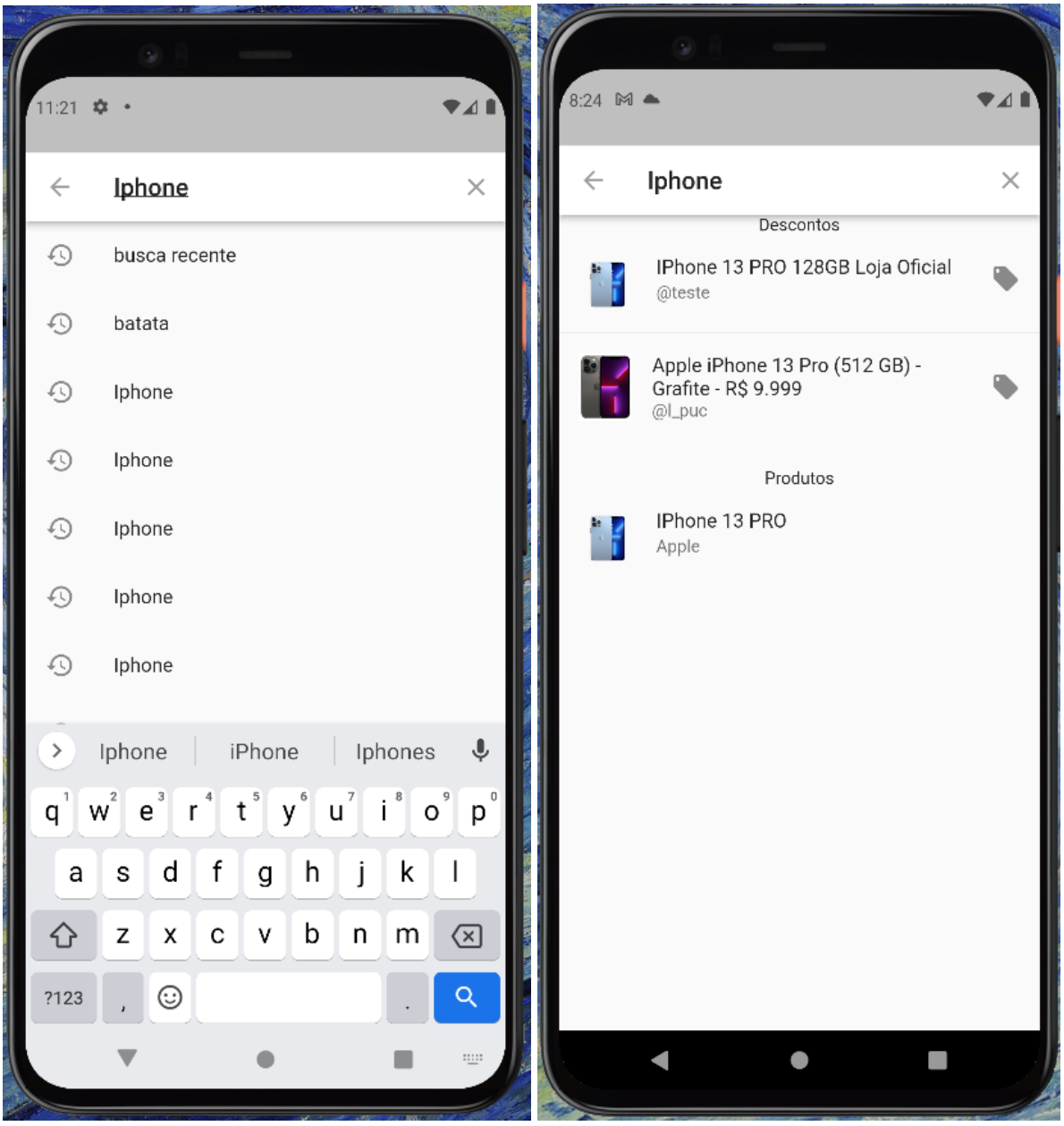

Figura 18 - Telas de busca implementadas.

A busca conta com uma lista de termos recentes, e também ficou próximo do protótipo, contendo a separação por ofertas e produtos, permitindo ao usuário visualizar mais de cada tipo sob demanda.

4.4.5 Adicionar promoção

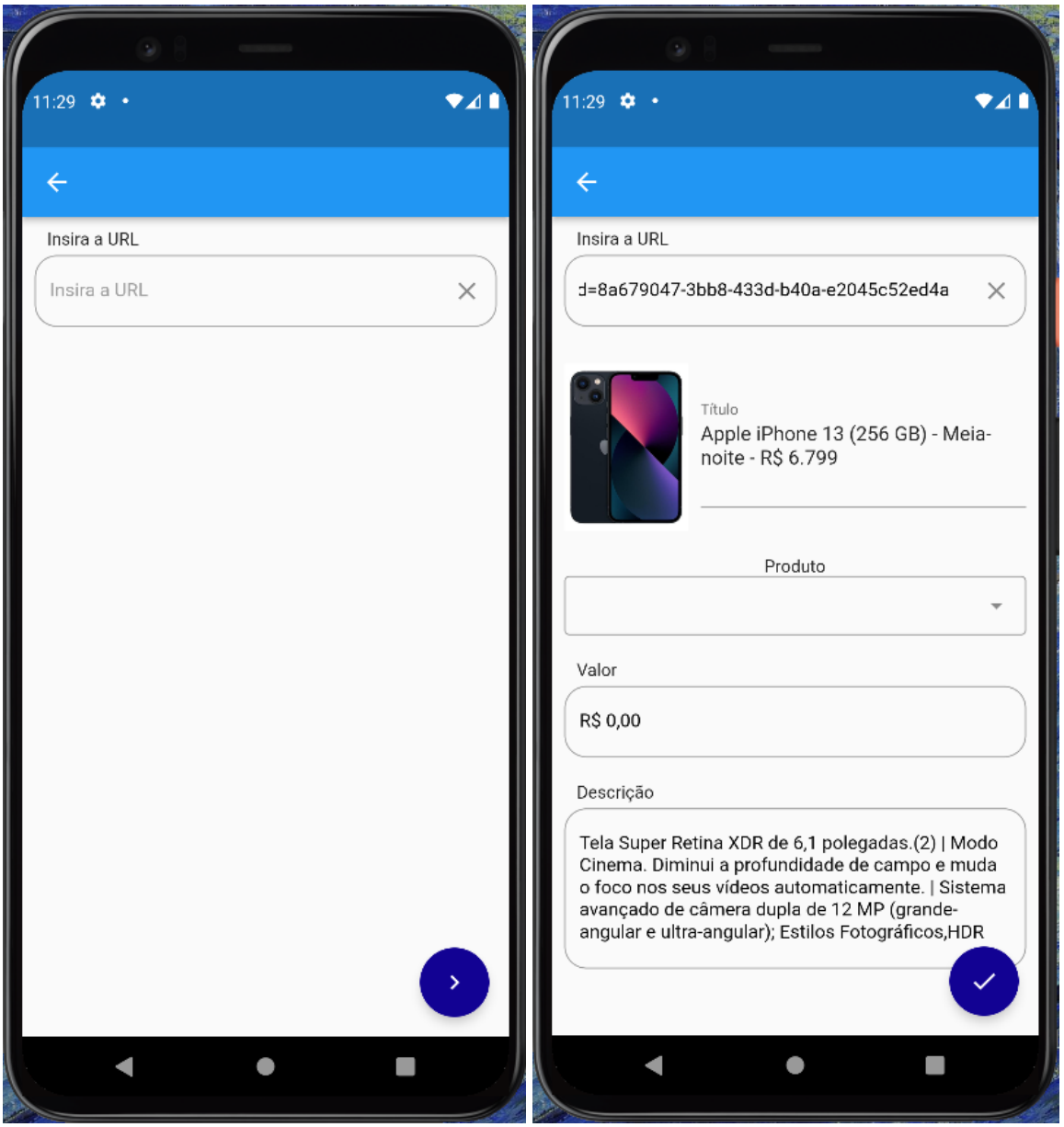

Figura 19 - Telas de adicionar promoção implementadas.

A tela de criar promoção também ficou próxima ao protótipo e obtém informações automaticamente da URL, como imagem, título e descrição. O usuário também pode escolher o produto a ser associado a partir de termos digitados por ele e, caso o produto possua variações, selecionar as correspondentes a aquela oferta.

4.4.6 Notificações

As notificações foram desenvolvidas de modo que o usuário possa determinar um preço máximo para o qual ele deseja ver promoções relacionadas a um produto. E sempre que uma promoção deste produto for postada abaixo do preço selecionado, ele receberá uma notificação.

As notificações foram desenvolvidas a partir da utilização do FireBase, que é um conjunto de ferramentas desenvolvido pela google para facilitar o desenvolvimento do back-end de aplicações [22]. Dentre essas ferramentas, o FireBase messaging possibilita o envio remoto de notificações a todos os dispositivos ou a um subgrupo selecionado arbitrariamente. Para isso, ele oferece também bibliotecas em flutter que devem ser instaladas juntas ao aplicativo no dispositivo do usuário, a partir delas, pode-se obter um token FCM que identifica o dispositivo e permite o envio de notificações por meio do FireBase Messaging.

Desta forma, foi programado para que este token fosse enviado ao servidor toda a vez que o usuário fizesse login e atualizando-o no banco de dados. Quando, por algum critério, o servidor decidir notificar este usuário, basta fazer uma requisição ao serviço do FireBase messaging com as credenciais do projeto que o mesmo se encarrega do resto.

Do lado da interface, bastou escrever os métodos que tratam o recebimento das notificações e do clique feito pelos usuários nas mesmas. A ideia é que a notificação ao ser clicada já abra o aplicativo na tela de interesse, seja ela uma promoção, produto ou perfil de um novo usuário que o segue. Nesta versão inicial, foi implementada apenas a notificação por preços inseridos em novas promoções de um produto que o usuário tenha cadastrado um alerta. Estes alertas podem ser cadastrados na página de qualquer produto e o usuário define o valor máximo para o qual ele deseja receber as notificações, desta forma, valores altos não serão notificados.

5 - Conclusão

A versão inicial alcançada cumpre com as principais necessidades destacadas e propostas e algumas características diferenciais em relação aos competidores ficaram bem evidentes nesta versão. Entretanto, para alcançar o nível de atenção aos detalhes requeridos por uma aplicação comercial e pública.

O fato de catalogar produtos, permitiu desenvolver funcionalidades voltadas a estes que podem ser de grande ajuda a um consumidor. O usuário consegue inserir alerta de preços para produtos específicos, com variações que ele selecionar, enquanto nas alternativas testadas o usuário ficava restrito a definir tags para as quais gostaria de receber notificações. Outro ponto, é que ele pode acessar a página de um produto e visualizar todas as ofertas relacionadas a tal, além de poder filtrar pelas variações disponíveis do mesmo. E ainda podem ser desenvolvidos muitos outros recursos a partir desta orientação associativa, como gráficos de preço por exemplo.

As seguintes funcionalidades ficaram faltando ou precisam ser aprimoradas, apesar de não essenciais, são funcionalidades úteis que foram pospostas por falta de tempo hábil: a possibilidade de seguir outros usuários e salvar promoções, algoritmos melhores capazes de exibir, em uma das opções de ordenação, conteúdo personalizado com base nos produtos que o usuário tem interesse e quem ele segue, permitir configurar um mínimo de curtidas e interações para receber alertas de algum produto, evitando notificações excessivas e desnecessárias. Outro detalhe visual importante é que as promoções ainda precisam ser visualmente identificadas quando finalizadas, além de criar filtros para que o usuário possa escolher visualizar apenas as ativas.

Para publicar o aplicativo, também seria necessário criar um meio de controle de conteúdo para evitar materiais inapropriados e uma moderação eficiente. Isto poderia ser feito com auxílio de algoritmos, mas inevitavelmente é necessário, também, um controle manual e, para isso, ficou faltando desenvolver opções de excluir e banir usuários quando logado em uma conta autorizada para tal.

Um ponto ignorado neste projeto, foi a questão visual da marca do aplicativo. Para o mesmo ser atrativo, é importante criar esquema de cores, logo e ícones. Além disso, é importante escrever os termos de uso e a política de privacidade adotados.

Referências

[1] COURA, P. Em tempos de lojas fechadas, comércio migra para o digital para sobreviver. 25 jan. 2021. Otempo.com.br. Disponível em: https://www.otempo.com.br/coronavirus/em-tempos-de-lojas-fechadas-comercio-migra-para-o-digital-para-sobreviver-1.2435862. Acesso em: 01 out. 2021.

[2] NOGUEIRA, P. Os Aplicativos dos Grandes Varejistas Brasileiros: Uma oportunidade de relacionamento ou apenas um folheto de ofertas digital?, 2018. Disponível em http://www.intercom.org.br/sis/eventos/2018/resumos/R13-1021-1.pdf. Acesso em: 23 set. 2021.

[3] CAMPOS, Álvaro. Mosaico Tecnologia estreia hoje na bolsa após levantar R$ 1,08 bilhão. 5 fev. 2021. Valor Investe. Disponível em: https://valorinveste.globo.com/mercados/renda-variavel/empresas/noticia/2021/02/05/mosaico-tecnologia-estreia-hoje-na-bolsa-apos-levantar-r-108-bilhao.ghtml. Acesso em: 1 out. 2021.

[4] MATA, Késley Brenner da Costa, et al. E-commerce: análise de dados sobre o comércio eletrônico no Brasil, 2021. Disponível em https://repositorio.pucgoias.edu.br/jspui/handle/123456789/1761. Acesso em: 01 out. 2021.

[5] CASTRO, P. The Rise of Serverless Computing. dez. 2019. Acm.org. Disponível em: https://cacm.acm.org/magazines/2019/12/241054-the-rise-of-serverless-computing/fulltext. Acesso em: 13 jul. 2022.

[6] ANDLINGER, P.. RDBMS dominate the database market, but NoSQL systems are catching up. 2013. Db-engines.com. Disponível em: https://db-engines.com/en/blog_post/23. Acesso em: 7 jun. 2022.

[7] Amazon Aurora User Guide for Aurora Amazon Aurora: User Guide for Aurora. Disponível em: https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/aurora-ug.pdf. Acesso em: 17 Jun. 2022.

[8] TUNG, L. Cloud computing: AWS is still the biggest player, but Microsoft Azure and Google Cloud are growing fast. 4 fev. 2021. ZDNet. Disponível em: https://www.zdnet.com/article/cloud-computing-aws-is-still-the-biggest-player-but-microsoft-azure-and-google-cloud-are-growing-fast/. Acesso em: 17 jun. 2022.

[9] RIFAI, M.. Serverless showdown: AWS Lambda vs Azure Functions vs Google Cloud Functions. 23 abr. 2021. A Cloud Guru. Disponível em: https://acloudguru.com/blog/engineering/serverless-showdown-aws-lambda-vs-azure-functions-vs-google-cloud-functions. Acesso em: 17 jun. 2022.

[10] Scrapy at a glance — Scrapy 2.6.1 documentation. 2022. Scrapy.org. Disponível em: https://docs.scrapy.org/en/2.6/intro/overview.html. Acesso em: 2 jul. 2022.

[11] XPath | MDN. 24 maio 2022. Mozilla.org. Disponível em: https://developer.mozilla.org/en-US/docs/Web/XPath. Acesso em: 2 jul. 2022.

[12] ORTIZ-OSPINA, E. The rise of social media. 2019. Our World in Data. Disponível em: https://ourworldindata.org/rise-of-social-media. Acesso em: 2 jul. 2022.

[13] JONES, M., BRADLEY, J., SAKIMURA, N. "JSON Web Token (JWT)", maio 2015. DOI: 10.17487/rfc7519. Disponível em: https://www.rfc-editor.org/rfc/rfc7519. Acesso em: 13 jul. 2022.

[14] Bloc State Management Library. 2022. Bloclibrary.dev. Disponível em: https://bloclibrary.dev/#/architecture. Acesso em: 14 jul. 2022.

[15] ANINDYA PRADNYA PARAMITHA. Getting Started with Flutter Bloc Pattern | Mitrais Blog. 22 out. 2021. Mitrais.com. Disponível em: https://www.mitrais.com/news-updates/getting-started-with-flutter-bloc-pattern/#:~:text=Bloc%20is%20a%20design%20pattern,and%20maintained%20by%20Felix%20Angelo.. Acesso em: 14 jul. 2022.

[16] Google. 2022. Disponível em: https://www.google.com. Acesso em: 2 ago. 2022.

[17] Pelando. 2022. Pelando.com.br. Disponível em: https://www.pelando.com.br/. Acesso em: 2 ago. 2022.

[18] O que é API? 2022. Redhat.com. Disponível em: https://www.redhat.com/pt-br/topics/api/what-are-application-programming-interfaces#:~:text=API%20significa%20interface%20de%20programa%C3%A7%C3%A3o,e%20integrar%20softwares%20de%20aplica%C3%A7%C3%B5es.. Acesso em: 2 ago. 2022.

[19] POSEY, B. virtual private server (VPS) or virtual dedicated server (VDS). 2021. SearchITOperations. Disponível em: https://www.techtarget.com/searchitoperations/definition/virtual-private-server-VPS-or-virtual-dedicated-server-VDS. Acesso em: 2 ago. 2022.

[20] O que é AWS? Como funciona Amazon Web Services. 2014. Amazon Web Services, Inc. Disponível em: https://aws.amazon.com/pt/what-is-aws/?nc1=f_cc. Acesso em: 2 ago. 2022.

[21] JEREMYLIKNESS. Serverless architecture considerations - Serverless apps. 14 abr. 2022. Microsoft.com. Disponível em: https://docs.microsoft.com/en-us/dotnet/architecture/serverless/serverless-architecture-considerations. Acesso em: 2 ago. 2022..

[22] SILVA, E. Firebase: o que é e como configurar no seu app Android. 2020. Geekhunter.com.br. Disponível em: https://blog.geekhunter.com.br/firebase-o-que-e-e-quando-usar-no-desenvolvimento-mobile/. Acesso em: 2 ago. 2022.

[23] PATEL, N. Usabilidade: O Que É, Conceito e Como Funciona. 30 dez. 2019. Neil Patel. Disponível em: https://neilpatel.com/br/blog/usabilidade-o-que-e/. Acesso em: 3 ago. 2022.