DESENVOLVIMENTO MOBILE UTILIZANDO FLUTTER E HASURA[1]

Darlan Michel da Silva[2]

André Fernando Rollwagen[3]

RESUMO

Este artigo aborda o desenvolvimento de um aplicativo *mobile*, com o enfoque no estudo da linguagem Flutter e na *engine*, ferramenta, de graphql, Hasura e um comparativo com outras tecnologias semelhantes presente no mercado. Tendo em vista o cenário ocasionado pela pandemia do COVID-19, o qual modificou a interação social, gerando a necessidade de distanciamento social, houve necessidade de reinventar formas de movimentar a economia. Para isso, foi idealizado e desenvolvido um aplicativo de e-commerce para comércios em geral, oportunizando realizar o pedido e receber as mercadorias em casa. O aplicativo possui a funcionalidade de acesso com uma conta cadastrada ou por meio de outras contas: Facebook ou Google. Ainda, por meio do aplicativo é possível ter acesso a promoções, busca de produtos por categorias, manutenção dos dados de acesso e de cadastro do usuário, realização e acompanhamento dos pedidos.

Palavras-chave: APP, Flutter, Hasura, e-comerce, COVID-19.

1 INTRODUÇÃO

Com a pandemia do COVID-19 as empresas precisam se reinventar para vender seus produtos. Com um aplicativo de vendas *online* o empresário consegue chegar até o seu cliente que está confinado em casa. Pensando nisso foi desenvolvida uma aplicação para fins comerciais, utilizando a linguagem Flutter, e como *back-end* utilizado o Hasura, que é uma *engine*, ferramenta, GraphQL tendo o Postgres como banco de dados.

Este trabalho tem como objetivo estudar as ferramentas elencadas acima, por meio do desenvolvimento de um aplicativo e-commerce, bem como comparar essas

---

[1] Trabalho de Conclusão de Curso (TCC) apresentado ao Curso de Tecnologia em Sistemas para Internet do Instituto Federal Sul-rio-grandense, Câmpus Passo Fundo, como requisito parcial para a obtenção do título de Tecnólogo em Sistemas para Internet, na cidade de Passo Fundo, em 2020.

[2] Aluno do curso de Tecnologia de Sistemas para Internet no Instituto Federal de Educação, Ciência e Tecnologia Sul-rio-grandense de Passo Fundo (IFSUL). E-mail: darlan.michel@hotmail.com.

[3] Orientador, professor do Instituto Federal de Educação, Ciência e Tecnologia Sul-rio-grandense de Passo Fundo (IFSUL). E-mail: andre.rollwa@gmail.com.

ferramentas com outras tecnologias em utilização no mercado, visando avaliar a eficiência das ferramentas escolhidas.

No Brasil, indicadores apontam que cerca de 74,7% das pessoas estão conectadas na internet. Esta facilidade de acesso demonstra que há oportunidades para o comércio crescer por meio das vendas digitais (FARTO, 2020).

Nos últimos meses, todos os setores do mercado foram afetados pela pandemia causada pelo novo coronavírus. O futuro ainda é incerto, mas os números mostram que o e-commerce disparou em tempos de pandemia (BRASIL, 2020a).

Segundo a Redação do e-commerce Brasil (2020b), a pandemia transformou os hábitos de consumo, gerando um notável aumento no número de consumidores *online*: o varejo passou de uma média de 5,1 milhões de consumidores mensais para 8,9 milhões, no mês de julho, em que as vendas registraram um crescimento superior a 50% na América Latina. As grandes varejistas conseguiram se adaptar rapidamente aos desafios da pandemia. Elas passaram de um faturamento mensal médio de US$ 19 milhões antes da Covid-19 para US$ 120 milhões, o que representa um crescimento superior a 500%. Da mesma forma, serviços de *delivery* se tornaram um forte apoio para os consumidores e comércios,registrando um crescimento acima de 100% em comparação ao mesmo período de 2019 (BRASIL, 2020b).

Entre os 15 setores analisados pela Conversion, agência de marketing digital no Brasil, 10 cresceram ao menos 10% YoY (ano a ano) quando o assunto é *e-commerce*. Entre os 10 setores que cresceram estão os segmentos de Comidas e Bebidas, Casa e Móveis, Pets, além de Calçados, Moda e Acessórios, Infantis, Jóias e Relógios, Eletrônicos e Eletrodomésticos, Importados e Varejo. Os números reforçam o quanto a pandemia impulsionou novos hábitos de consumo e compras no que muitos definem como um “novo normal” (BRASIL, 2020a).

Os clientes virtuais utilizam os dispositivos móveis para navegar nas lojas e realizar as suas compras. De acordo com o recente estudo The Covid-19 Series, do Itaú BBA, 43,1%das vendas do e-commerce acontecem via smartphones (FARTO, 2020).

Então, para aumentar e garantir a experiência no mobile commerce é preciso analisar as estratégias de venda, e proporcionar ao cliente fácil e rápida navegação, digitação simples e segurança.

A seguir será apresentado as tecnologias envolvidas no desenvolvimento do aplicativo, após uma comparação com outras frameworks. Seguindo será discutido a metodologia, resultados e considerações finais.

2 TECNOLOGIAS PARA O DESENVOLVIMENTO MOBILE

Nesta seção são discutidos os conceitos envolvidos no desenvolvimento do aplicativo *mobile*. Em seguida, serão apresentados as ferramentas utilizadas para a execução do desenvolvimento. Por fim, é apresentado um comparativo dessas tecnologias com outras tecnologias atuais do mercado.

2.1 FLUTTER

Nos últimos anos criaram-se dezenas de *frameworks* que sempre prometem performance e agilidade no desenvolvimento mobile, com essa constante evolução fica atrativo a utilização de novos frameworks, em contrapartida há um certo receio de utilizar tecnologias que são recentes (OSSADA, 2019).

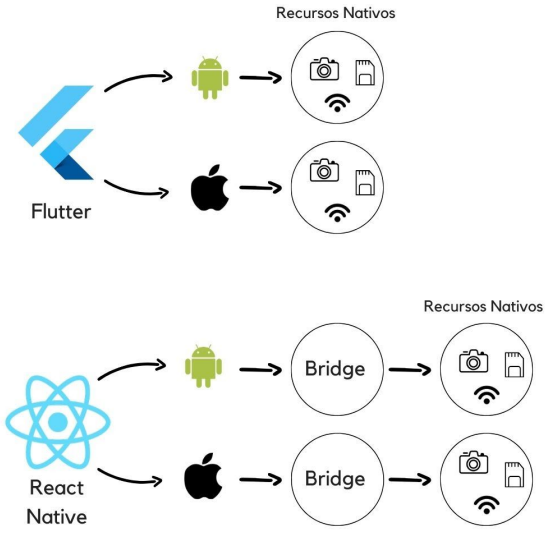
Framework é um facilitador no desenvolvimento de aplicações e, sua utilização poupa tempo e custos para quem utiliza, pois é um conjunto de bibliotecas utilizadas para otimizar recursos (ANDRADE, 2020).

Flutter é um *framework open-source* lançado em Dezembro de 2018 e mantido pela Google com a promessa de desenvolvimento *cross-platform* (plataforma cruzada) para Android, iOS, Windows,Mac, Fuchsia e WEB, podendo ser compilados para WEB, Mobile e Desktop com apenas um código sem a necessidade de adaptações para as plataformas (OSSADA, 2019). Nesse *framework* o Dart é a principal linguagem de desenvolvimento, criada em 2011 e também mantida pela Google. Ele utiliza uma abordagem até então única para lidar com os componentes nativos de cada plataforma, em que cada um deles é implementado pelo próprio *framework* é apresentado ao usuário por um motor de renderização próprio (DEVMEDIA, 2020).

Segundo Ossada (2019), o DART é simples e é bem parecida se comparada com as linguagens mais populares que temos hoje no mercado, como C#, Java, Javascript,etc. Ao criar um aplicativo com o Flutter, seu código é compilado para a

linguagem base do dispositivo, ou seja, as aplicações são realmente nativas e por isso conseguem acessar recursos do dispositivo sem a “ajuda” de terceiros e com o desempenho maior (ANDRADE, 2020). A Figura 1 abaixo ilustra uma das diferenças destes acessos entre o React Native e o Flutter, por exemplo o Flutter acessa diretamente os recursos nativos dos sistemas operacionais, enquanto o React tem uma ponte entre a aplicação e os recursos nativos do aparelho.

Figura 1 - Diferenças entre Flutter e React Native



Fonte: ANDRADE, 2020.

A ponte entre o código e a plataforma é o que faz as aplicações ficarem custosas, presentes no exemplo do react native. De uma lado temos o aplicativo utilizando tecnologias webs (Html, CSS e JS) e de outro lado temos o mundo nativo, esses dois mundos, individualmente, são performáticos, entretanto há sempre uma troca de contexto para que seu aplicativo execute no nativo, e é essa troca de contexto que são custosos a aplicação (OSSADA, 2019).

O Flutter trouxe a responsabilidade de renderização para o lado do aplicativo, visto que antes a responsabilidade de renderização ficava do lado da plataforma, não necessitando fazer a troca de contexto (OSSADA, 2019).

Ossada (2019) e Andrade (2020) citam algumas grandes empresas que já aderiram a essa tecnologia, como a própria Google que é desenvolvedora do flutter, Alibaba, Tencent, Nubank, Groupon, entre outras. o *Framework* demonstra ser uma

excelente opção de estudo para desenvolvedores que buscam melhor performance para suas aplicações.

2.2 HASURA

Hasura é uma empresa de tecnologia de software que cria produtos de ferramentas para desenvolvedores, Processo interno como serviço (*back-end as a service* - BaaS) e produtos de plataforma como serviço (*platform as a service* - PaaS). Em julho de 2018, a empresa anunciou o lançamento de código aberto do Hasura GraphQL Engine para permitir que os desenvolvedores configurem terminais GraphQL em seus aplicativos postgres existentes. Em setembro de 2018, a empresa introduziu um sistema de trigger no Postgres para a construção de aplicativos sem servidor (HALL ,2018).

O Hasura GraphQL Engine é um servidor GraphQL extremamente rápido, e em tempo real de APIs GraphQL sobre bancos de dados Postgres, com *webhook triggers* em eventos de banco de dados, e esquemas remotos para lógica de negócios (DURAIRAJU,2020).

O GraphQL é uma linguagem de consulta para APIs e um *runtime* para atender a consultas com dados existentes. O esquema do GraphQL é formado com três tipos principais de ações:

- Mutations são usadas para fazer insert, update e delete no banco de dados;
- As query são utilizadas para solicitar os dados do *backend*;
- E as subscriptions são usadas para um relacionamento em tempo real com o servidor;

Segundo Batschinski (2020), o uso do BaaS fornece os seguintes benefícios: maior concentração no produto principal, automação em relação ao gerenciamento de bancos de dados, redução do custo total de desenvolvimento, reúne as vantagens de um *back-end* como serviço, essa configuração permite que um desenvolvedor crie seus esquemas e os implementa automaticamente em um ambiente de infraestrutura pronto para uso.

O Hasura GraphQL Engine pode ser configurado para funcionar com seu sistema de autenticação existente e pode manipular o controle de acesso usando

regras em nível de campo com variáveis dinâmicas do seu sistema de autenticação (DURAIRAJU, 2020).

2.3 FIREBASE AUTHENTICATION

O Firebase Authentication fornece serviços de *back-end*, Kits de Desenvolvimento de Software (SDKs) fáceis de usar e bibliotecas de interface de usuário (IU) prontas para autenticar usuários no seu aplicativo. Ele oferece suporte à autenticação usando senhas, números de telefone, provedores de identidade federados conhecidos, entre outros (FIREBASE, 2020).

O SDK do Firebase Authentication fornece métodos para criar e gerenciar usuários que utilizam os próprios endereços de e-mail e senhas para fazer login, e permite o envio de e-mails de redefinição de senha. Ele fornece métodos que permitem aos usuários fazer login com as Contas do Google, Facebook,Twitter e GitHub (FIREBASE, 2020).

Para conectar um usuário ao seu aplicativo, primeiro precisa ter as credenciais de autenticação do usuário. Essas credenciais podem ser o endereço de e-mail e a senha do usuário ou um token do OAuth. Os serviços de *back-end* verificarão essas credenciais e retornarão uma resposta ao cliente (FIREBASE, 2020).

Após fazer login, é possível ter acesso às informações básicas do perfil do usuário, podendo controlar o acesso dele aos dados armazenados. É possível também usar o token de autenticação fornecido para verificar a identidade dos usuários nos seus próprios serviços de back-end (FIREBASE, 2020).

2.4 COMPARATIVOS

Nesta seção será apresentado um comparativo entre o Flutter e outras *frameworks* utilizadas atualmente no mercado como React Native, Xamarin e Ionic, como também uma comparação entre os bancos de dados Hasura e o Firebase.

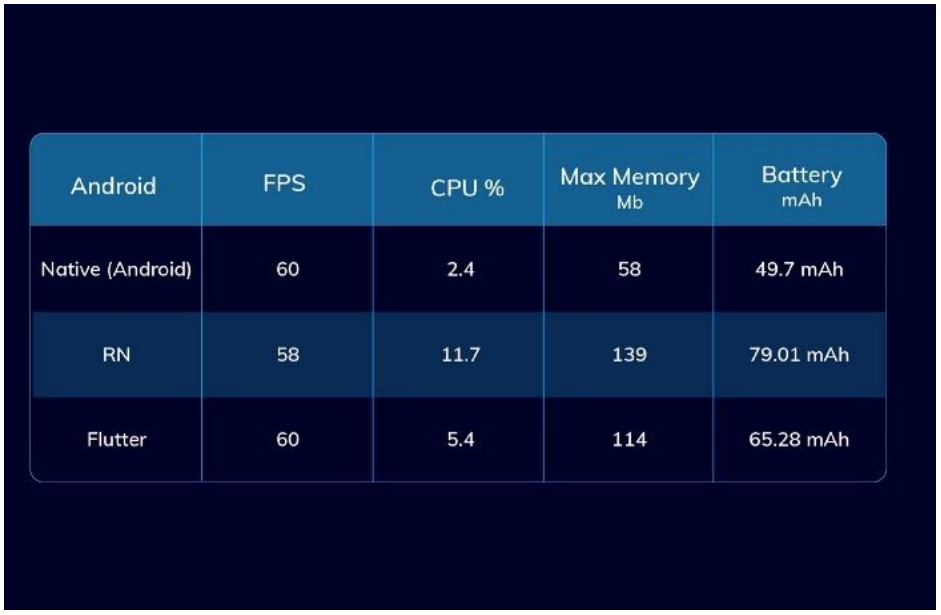
2.4.1 Flutter vs React Native

O React Native é mantido e apoiado pelo Facebook desde 2015, e possui como linguagem principal o javascript. Segundo CLARK (2020), as semelhanças entre Flutter e React são que ambos são *frameworks* multiplataformas, de código aberto, possuem uma ferramenta de compilação rápida e fornece uma experiência nativa.

Os desenvolvedores usam amplamente o JavaScript na comunidade. Portanto, ele permite que os desenvolvedores criem aplicativos móveis sem um treinamento extensivo. É por isso que muitas organizações foram capazes de se adaptar ao React Native (CLARK, 2020).

Se tratando de desempenho, o Flutter é particularmente útil para aplicativos com animações pesadas. Ele oferece melhor desempenho do que o React. NEVILLE (2020) não recomenda o uso do React Native em uma operação muito pesada da CPU, visto que o Flutter é um ótimo ajuste para essas tarefas do ponto de vista da CPU e da memória, como visto na Figura 2.

Figura 2 - Comparação de desempenho



Fonte: NEVILLE, 2020.

2.4.2 Flutter vs Xamarin

A Xamarin foi originalmente fundada em 2011, mas em 2016 a Microsoft adquiriu a Xamarin. Ela possui como linguagem principal o C#. O C# é uma

linguagem muito popular, e existe uma enorme comunidade de desenvolvedores no mundo (CODEMAGIC, 2019). O mesmo não ocorre com o Dart, visto que é uma linguagem razoavelmente nova, lançada em 2018.

Embora a Xamarin tenha alguns recursos interessantes, ela está disponível gratuitamente com limitações, por outro lado, o Flutter é *open source* (CODEMAGIC, 2019).

Em termos de desempenho, o Flutter pode ser considerado melhor do que os aplicativos Xamarin. Xamarin tem uma configuração no *framework* que pode relatar os problemas de desempenho mais cedo. Além disso, ao desenvolver aplicativos, o recurso de compilação rápida do Flutter contribui muito para a produtividade do desenvolvedor (CODEMAGIC, 2019).

2.4.3 Flutter vs Ionic

O Ionic foi lançado em 2013 pela empresa Drifty Company com o objetivo de criar aplicativos móveis multiplataforma utilizando linguagens web, entre as principais utilizadas são o HTML, CSS e JavaScript.

Segundo Netkow (2020), o Flutter possui um desempenho melhor para animações agressivas, por outro lado o Ionic possui aplicativos padrões. Outra consideração de Netkow é que os aplicativos do Ionic são relativamente menores em questão de tamanho, pois ele utiliza recursos dos navegadores padrões não requerendo códigos básicos.

2.4.4 Hasura vs Firebase

Hasura e Firebase são APIs de BaaS que ajudam a desenvolver com mais agilidade. As diferenças entre os dois são, segundo HASURA (2018):

- Hasura oferece um banco de dados Postgres, enquanto o Firebase é um banco próprio;
- Hasura tem a capacidade de fazer consultas em massa, enquanto o Firebase está restrito a consultas CRUD básicas;
- No Firebase é mais difícil modelar relacionamentos entre dados independentes, visto que cometido um erro no início da modelagem é difícil

migrar para um novo modelo. Hasura suporta modelagem relacional e tipos JSON. A modelagem relacional facilita a modelagem de dados com muitos relacionamentos inerentes sem duplicar dados ou se preocupar com o manuseio de atualizações.

- As permissões do Firebase só podem capturar regras com base nos dados dentro do 'path' atual (node). No Hasura, você pode especificar permissões granulares com base no ID do usuário em uma linha ou com base em relacionamentos com outras tabelas.

3 METODOLOGIA

Foi desenvolvido um aplicativo *e-commerce*, visando auxiliar os comércios locais em tempo de pandemia. O aplicativo foi desenvolvido utilizando o *framework* Flutter, para armazenamento dos dados foi utilizado o Hasura, sendo esse um banco de dados relacional e de fácil utilização.

Ainda, a aplicação foi focada na plataforma Android, visto que para poder desenvolver para IOS necessita-se de uma máquina com MAC. E visto que o Flutter está em fase beta para web e alfa para sistemas desktop.

Assim, o desenvolvimento do aplicativo foi segmentado em *Front-end* e *Back-end*, em que para o *Front-end* utilizou-se o *framework* Flutter, visto que o mesmo utiliza uma estrutura em forma de *widgets*, que são renderizados na tela. E para o *back-end* foi utilizado o Hasura, com banco de dados Postgres, o qual possui configurações de permissões de usuários através de um *JSON Web Token* (JWT) fornecido pelo Firebase.

O aplicativo não foi testado por outros usuários pois é apenas uma versão teste de protótipo, o qual será testado em uma próxima fase.

4 RESULTADOS E DISCUSSÕES

Foi desenvolvido um aplicativo de vendas *online* para o sistema operacional android, que contém uma tela de login, dada por meio do Firebase Authentication que utiliza uma decodificação jwt. O aplicativo também tem uma tela inicial de

promoções, seguida de uma tela de produtos e categorias, o qual o cliente pode filtrar a categoria desejada ou pesquisar pelo nome do produto.

Além disso, possui uma tela de carrinho de compras, onde o cliente poderá visualizar os produtos desejados e fazer a manutenção dos mesmos, além de selecionar o endereço em que receberá o pedido, a forma de pagamento na hora da entrega e também um campo para poder inserir algum cupom de desconto promocional.

Há também uma tela de perfil do usuário, onde ele poderá cadastrar e editar seus endereços de entrega, assim como poderá acompanhar o status de seus pedidos, editar dados referente ao seu perfil e por fim fazer logout de sua conta.

As próximas seções apresentam com mais detalhes as funcionalidades do aplicativo.

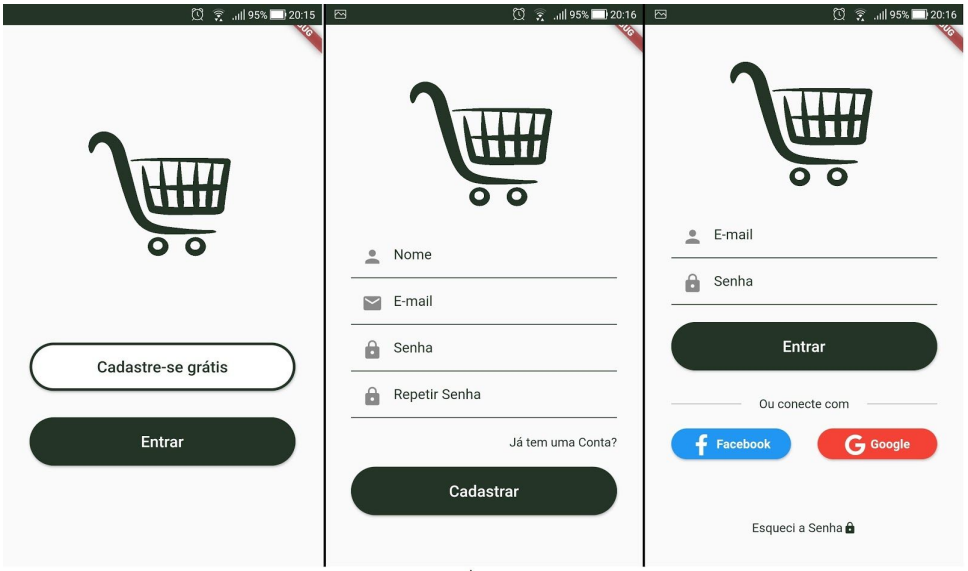
4.1 LOGIN

Ao iniciar o aplicativo o usuário vai se deparar com dois botões, um para efetuar o login (Entrar) e outro para se cadastrar conforme Figura 3a. Caso o usuário clique na opção de cadastro será direcionado para a screen de cadastro contendo os seguintes campos, nome, e-mail, senha e confirmação de senha, conforme Figura 3b, em que ele terá que preencher todos os campos com validação. Ao clicar no botão “Cadastrar” e todos os campos forem validados, ele será direcionado para a tela de promoções. Se clicar na opção “Já tem uma Conta?” o usuário será direcionado para a tela de Login. Na tela de Login, conforme Figura 3c, o usuário tem a opção de entrar no aplicativo usando uma conta que ele cadastrou ou utilizando a conta do facebook ou do google.

Para a autenticação foi usado uma biblioteca do Flutter chamada *firebase_auth*, que faz conexão com o Firebase authentication. Para fazer login com a conta do facebook foi utilizado a biblioteca *flutter_facebook_auth* e para a conta do google a *google_sign_in*, ambas fazem conexão com o Firebase.

No Firebase Authentication é necessário ativar as formas de Login, neste caso por e-mail, google e facebook. Além disso, é necessário liberar as permissões na Facebook Developers e adicionar o domínio do Hasura nos domínios autorizados do Firebase.

Figura 3 - Telas iniciais



Fonte: AUTOR, 2020.

E no Hasura é necessário alterar as configurações, incluindo a chave HASURA_GRAPHQL_JWT_SECRET e dentro dessa chave o valor demonstrado na Figura 4.

Figura 4 - Valor da chave HASURA_GRAPHQL_JWT_SECRET

```
{
    "type":"RS256",
    "jwk_url": "https://www.googleapis.com/service_accounts/v1/jwk/securetoken@system.gserviceaccount.com",
    "audience": "<firebase-project-id>",
    "issuer": "https://securetoken.google.com/<firebase-project-id>"
}
```

Fonte: AUTOR, 2020.

Com essas configurações o Firebase enviará o jwt para o Hasura e o mesmo irá decodificar e permitir o login do usuário para ter acesso ao banco de dados.

4.2 BANCO DE DADOS

Com essas configurações apresentadas na Figura 4, o Hasura pode receber o “X-Hasura-User-Id”, que é a identificação fornecida pelo Firebase Authentication após o login. Com posse dessa informação o Hasura consegue identificar o usuário e mostrar apenas as permissões concedidas nas configurações.

Cada tabela tem suas permissões e no painel é possível definir suas regras. Neste caso as permissões de insert, select, update e delete estão iguais, mas é possível definir uma configuração diferente para cada uma.

O banco de dados tem as seguintes tabelas:

- carrinho: Onde ficam gravados os produtos que os clientes selecionam para serem salvos no carrinho;
- categoria: Onde estão salvos as categorias dos produtos;
- cliente: Onde são gravados os dados dos usuários;
- cupomDesc: São as configurações dos cupons de descontos;
- cupomUsados: São gravados os usuários e quais cupons eles usaram;
- enderecos: Onde são armazenados os endereços dos usuários;
- formaPagamento: É as formas de pagamentos estipuladas na hora da entrega;
- pedido: Onde ficam gravadas as informações dos pedidos finalizados pelos usuários;
- produtos: Onde ficam gravadas as informações sobre os produtos e suas ofertas;

4.3 PROMOÇÕES

Após o usuário fazer o login, ele será direcionado para a tela principal, que é a tela de promoções, conforme Figura 5. No centro dessa página contém um carrousel que é animado passando um produto de cada vez num período de 3 segundos. Para esse efeito foi utilizado a biblioteca *flutter_swiper*.

Caso o usuário deseje comprar um desses produtos da promoção, clicando sobre o produto, abrirá uma caixa de diálogo perguntando a quantidade desse produto que deseja adicionar no carrinho de compras, o qual será detalhado na seção 4.5.

Figura 5 - Tela de promoções

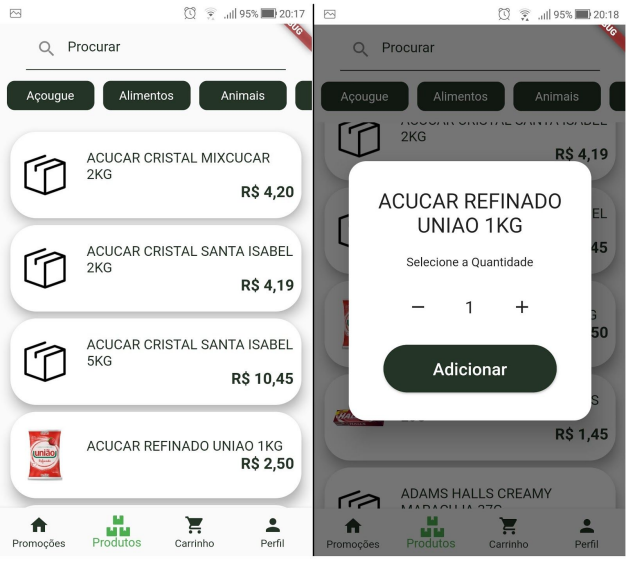
Fonte: AUTOR, 2020.

4.4 PRODUTOS

A próxima aba conforme Figura 6 é a aba dos produtos que contém um campo de texto para procurar um produto específico.

Abaixo do campo de pesquisa estão dispostos botões para acesso às categorias dos produtos, assim, clicando sobre uma delas, o usuário visualiza os produtos apenas daquela categoria. E abaixo das categorias temos a listagem de produtos contendo imagem, descrição e preço. Clicando sobre o produto será aberto uma caixa de diálogo para selecionar a quantidade daquele produto para ser adicionado ao carrinho, conforme ilustrado na Figura 6b.

Figura 6 - Tela de produtos



a b

Fonte: AUTOR, 2020.

4.5 CARRINHO

Após o usuário adicionar o produto no seu carrinho de compras, Figura 7, a tela será preenchida com os produtos adicionados, podendo ser alterados a quantidade ou excluídos clicando em seus botões respectivos.

Abaixo dos produtos está a opção de selecionar o endereço cadastrado pelo cliente, sendo que sem um endereço selecionado não é possível finalizar o pedido. O usuário também terá que selecionar a forma de pagamento na hora da entrega, o qual é um campo obrigatório.

Se desejado pode ser incluído um cupom de desconto, em que o usuário digitará a palavra chave do desconto, e receberá um desconto em porcentagem referente ao cadastro do banco.

Figura 7 - Tela do carrinho de compras

Fonte: AUTOR, 2020.

E por final um cartão contendo o resumo do pedido, com os valores do subtotal, desconto se houver, valor de frete fixo pela aplicação e o total, quando há alteração de produtos todos os campos se adaptam e atualizam seus valores.

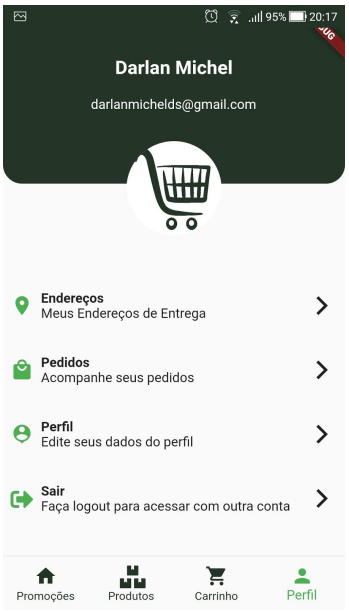
Quando o usuário clicar no botão “Finalizar Pedido”, os dados do carrinho são salvos na base de dados, e num futuro próximo, mostrados em outra aplicação para serem disponibilizados para a separação e envio deste pedido ao cliente.

4.6 PERFIL

A última tela do aplicativo é a página do perfil do usuário, conforme Figura 8, contendo seu nome e seu email na parte superior. Em seguida temos a manutenção dos endereços do usuário, a opção de acompanhar em tempo real os pedidos realizados, a edição dos dados do perfil e o logout.

Na manutenção de endereços há uma listagem de endereços cadastrados, conforme Figura 9a, em que ao clicar nos três pontinhos de cada cartão de endereço é possível editar ou excluir o mesmo. E mais abaixo há um botão para cadastrar um novo endereço.

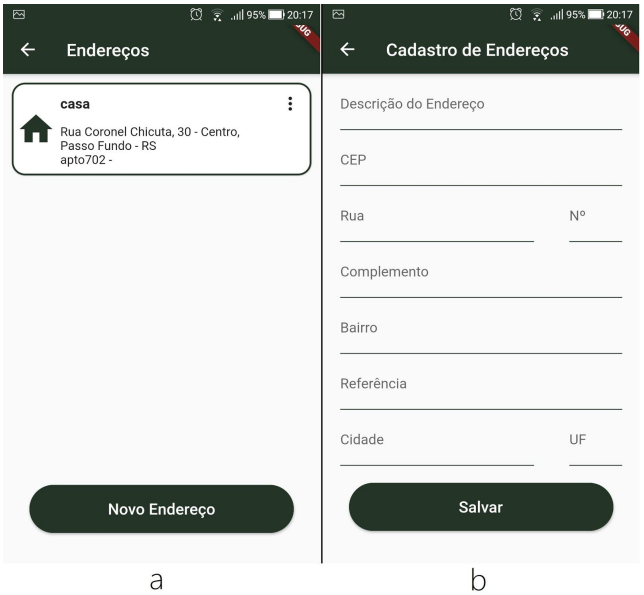
Figura 8 - Tela Perfil



Fonte: AUTOR, 2020.

Na tela de cadastro de endereço, Figura 9b, temos um formulário com descrição do endereço, CEP, rua, numero, complemento, bairro, referência, cidade e UF. Quando o usuário preencher o campo CEP com um número válido, a biblioteca *via_cep* buscará informações desse endereço e preencherá automaticamente os campos do endereço como rua, bairro, cidade, e UF.

Figura 9 - Tela de endereços



a b

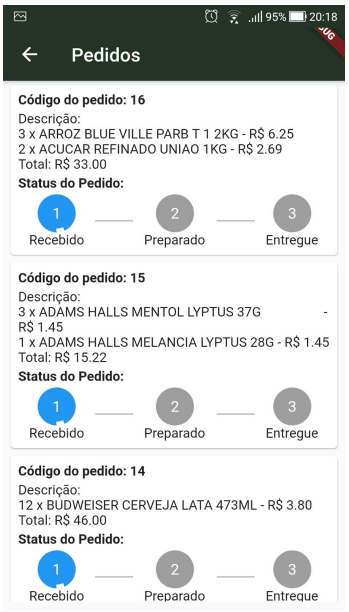
Fonte: AUTOR, 2020.

Na tela de acompanhamento de pedidos, conforme Figura 10, o usuário poderá acompanhar em tempo real como está a preparação do seu pedido. Nesta tela cada pedido terá seu código identificador, os produtos solicitados na hora da finalização do pedido, o total a pagar, e o *status* do pedido.

Os *status* mudam de cor conforme forem sendo atualizados, a bolinha azul significa que o pedido está pendente naquele *status*, essa bolinha azul tem uma animação de giro. Após concluída a etapa a bolinha muda para a cor verde, e a cor cinza significa que ainda está esperando aquela etapa.

Graças ao *subscription* do Hasura, assim que for mudado o *status* do pedido no banco de dados, a aplicação muda em tempo real também.

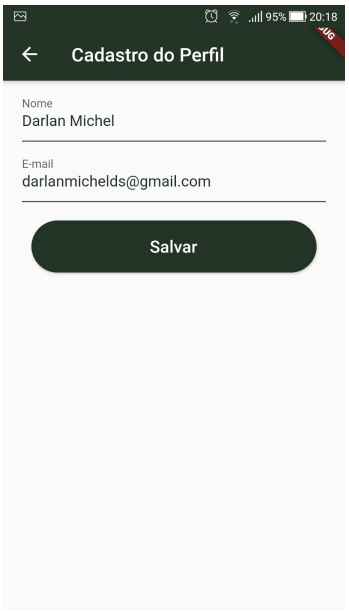
Figura 10 - Tela de pedidos



Fonte: AUTOR, 2020.

E por fim a tela para editar os dados do usuário, conforme Figura 11, é um formulário simples com apenas nome, endereço de email e um botão para salvar as informações no banco de dados.

Figura 11 - Tela de cadastro do perfil



Fonte: AUTOR, 2020.

5 CONSIDERAÇÕES FINAIS

Com o desenvolvimento deste trabalho foi percebido que o Flutter é um framework eficiente e de fácil aprendizado, apesar das outras ferramentas utilizar linguagens mais conhecidas, o que tornaria o desenvolvimento mais rápido, o Dart mostrou-se muito eficiente, ele é muito parecido com javascript, o que facilitou para o desenvolvimento e aprendizagem rápida dessa nova tecnologia. Além de que a maneira em que o flutter trabalha, em widgets, torna mais prático o desenvolvimento orientado a objetos e o aproveitamento de código.

Quanto ao Hasura, ele facilitou e agilizou o processo de desenvolvimento do aplicativo, pois as regras de negócios de cada usuário estavam prontas após configuradas na própria tabela. E com a integração com o Firebase authentication facilitou ainda mais os métodos de login que consequentemente ajudaram nas regras de negócios do próprio banco de dados.

A única dificuldade encontrada foi que como o Flutter é uma tecnologia recente e vem sofrendo atualizações constantes, em que algumas bibliotecas vão sendo atualizadas também e consequentemente resultando em alguns bugs, além

de ter uma comunidade muito pequena até o momento, levando alguns dias para ser respondido quanto a minhas dúvidas.

Como trabalho futuro pretende-se desenvolver uma extensão contendo a manutenção do *e-commerce* para os lojistas, permitindo o cadastro de produtos, alteração de preços, criar promoções, ajustar estoque, e a manutenção dos pedidos feitos pelos clientes nesta aplicação desenvolvida para esse trabalho.

ABSTRACT

This article discusses the development of a mobile application, focusing on the study of the Flutter language and the graphql engine, Hasura and a comparison with other similar technologies present in the market. The developed application is an e-commerce for businesses in general, where the user can place his order and receive the goods in his own home through tele delivery, because in the current moment with the pandemic of COVID-19 there is a need for social distance , but the economy needs to keep turning. This app will contain a login screen and Facebook and Google accounts can be used to access, a promotions screen where the establishment can show its prices, a product screen separated by categories, a profile screen where the user you can maintain your data and track your orders, and finally a shopping cart screen where the user can finalize their purchase.

Keywords: APP, Flutter, Hasura, e-comerce, COVID-19.

REFERÊNCIAS

ANDRADE, A. P. D. *O que é Flutter? - Blog da TreinaWeb*. 2020. <https://www.treinaweb.com.br/blog/o-que-e-flutter/>. (Accessed on 09/23/2020).

BATSCHINSKI, G . *GraphQL o que é? Visão geral, prós e contras | Back4App Blog*. 2020.<https://blog.back4app.com/pt/graphql-o-que-e/>. (Accessed on 09/25/2020).

BRASIL, R. E.-C. *Conversion: e-commerce atinge 1,27 bilhão de acessos em agosto.* 2020a.

<https://www.ecommercebrasil.com.br/noticias/conversion-e-commerce-acessos-agosto/>. (Accessed on 09/22/2020).

BRASIL, R. E.-C. *E-commerce na América Latina cresce mais de 50% na pandemia*. 2020b. <https://www.ecommercebrasil.com.br/noticias/e-commerce-america-latina-pandemia-coronavirus/>. (Accessed on 09/22/2020).

CLARK, J. *Flutter vs. React Native | Segredos Desvendados | Back4App Blog*. 2020. <https://blog.back4app.com/pt/flutter-vs-react-native-comparacao/>. (Accessed on 12/24/2020).

CODEMAGIC. *Flutter vs Xamarin: A Developer's Perspective*. 2019. <https://blog.codemagic.io/flutter-vs-xamarin-a-developer-s-perspective/>. (Accessed on 12/04/2020).

DEVMEDIA. *Guia Completa de Flutter: Aprenda Flutter do Básico ao Avançado*. 2020. <https://www.devmedia.com.br/guia/flutter/40713>. (Accessed on 09/23/2020).

DURAIRAJU, P. *Hasura GraphQL Engine - GitHub*. 2020. <https://github.com/hasura/graphql-engine/blob/master/translations/README.portuguese_br.md>. (Accessed on 09/25/2020).

FARTO, M. *Por que você deve estar atento ao mobile commerce?* 2020. <https://www.ecommercebrasil.com.br/artigos/por-que-voce-deve-estar-atento-ao-mobile-commerce/>. (Accessed on 09/22/2020).

FIREBASE *Authentication*. 2020. <https://firebase.google.com/docs/auth>. (Accessed on 09/28/2020).

HALL, S. *Hasura Focuses on Making Kubernetes Easier with GitOps*. 2018. <https://thenewstack.io/hasura-focuses-on-making-kubernetes-easier-with-gitops/>. (Accessed on 09/25/2020).

HASURA. *Data APIs: Hasura vs Firebase. Hasura and Fire-base both offer database...* 2018. <https://hasurahq.medium.com/comparing-data-apis-on-hasura-to-data-apis-on-firebase-f565c32bc0f9>. (Accessed on 12/24/2020).

NETKOW, M. *Ionic vs Flutter - Ionic Framework*. 2020. <https://ionicframework.com/resources/articles/ionic-vs-flutter-comparison-guide>. (Accessed on 12/24/2020).

NEVILLE, R. *Flutter vs React Native vs Native: comparação de desempenho aprofundada*. 2020. <https://medium.com/@rodneyneville/flutter-vs-react-native-vs-native-comparaC3A7C3A3o-de-desempenho-aprofundada-156f9a6f0bd9>. (Accessed on 12/24/2020).

OSSADA, T. *Por que Flutter? - Medium.* 2019. <https://medium.com/toshiossada/por-que-flutter-8f17cc2bb02e>. (Accessed on 09/23/2020).