Universidade de Brasília – UnB
Faculdade UnB Gama – FGA
Engenharia de Software

Agromart: Integração e Publicação do Aplicativo para Agricultura Familiar

Autor: Christian Fleury Alencar Siqueira e Thiago Siqueira Gomes
Orientador: Dr. André Luiz Peron Martins Lanna

Brasília, DF
2024

Christian Fleury Alencar Siqueira e Thiago Siqueira Gomes

Agromart: Integração e Publicação do Aplicativo para Agricultura Familiar

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Universidade de Brasília – UnB

Faculdade UnB Gama – FGA

Orientador: Dr. André Luiz Peron Martins Lanna

Coorientador: Dr. Rudi Henri van Els

Brasília, DF

2024

Christian Fleury Alencar Siqueira e Thiago Siqueira Gomes
Agromart: Integração e Publicação do Aplicativo para Agricultura Familiar/ Christian Fleury Alencar Siqueira e Thiago Siqueira Gomes. – Brasília, DF, 2024-
86 p. : il. (algumas color.) ; 30 cm.

Orientador: Dr. André Luiz Peron Martins Lanna

Trabalho de Conclusão de Curso – Universidade de Brasília – UnB
Faculdade UnB Gama – FGA , 2024.

1. Agromart. 2. CSA. I. Dr. André Luiz Peron Martins Lanna. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Agromart: Integração e Publicação do Aplicativo para Agricultura Familiar

CDU 02:141:005.6

Christian Fleury Alencar Siqueira e Thiago Siqueira Gomes

Agromart: Integração e Publicação do Aplicativo para Agricultura Familiar

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Trabalho aprovado. Brasília, DF, 17 de setembro de 2024:

Dr. André Luiz Peron Martins Lanna
Orientador

Dr. Rudi Henri van Els
Coorientador

Profa. Ma. Cristiane Soares Ramos
Convidado 1

Prof. Dr. Ricardo Ajax Dias Kosloski
Convidado 2

Brasília, DF
2024

Resumo

O Agromart é uma solução tecnológica com o objetivo de conectar produtores da agricultura familiar e consumidores de produtos orgânicos, por meio de um aplicativo que possibilita a compra e venda desses produtos. A ideia dessa aplicação surgiu em um Hackathon realizado na FGA no ano de 2020, desde então o seu desenvolvimento foi continuado por alunos da UnB-FGA ao longo de cinco trabalhos de conclusão de curso. A solução atualmente consiste em um aplicativo móvel para os consumidores, um servidor com uma API principal para cada CSA e uma API dicionário que conecta os consumidores ao servidor de cada CSA listada. Por se tratar de um trabalho desenvolvido por várias mãos diferentes ao longo do tempo sem seguir uma metodologia específica para conectar cada um dos trabalhos, a aplicação carece de documentação e uma política clara para orientar o desenvolvimento e dar unidade aos trabalhos desenvolvidos. Neste contexto, o presente trabalho de conclusão de curso em um trabalho de reengenharia, organizou os repositórios, documentou as funcionalidades, e integrou tudo que foi desenvolvido em um único ponto de convergência, para dar unidade ao projeto Agromart. Como efeito, este buscou viabilizar a publicação do Agromart, e possibilitou que o desenvolvimento de novas funcionalidades futuramente sigam essa mesma unidade.

Palavras-chave: Agromart. comunidade que sustenta a agricultura (CSA). agricultura familiar.

Abstract

Agromart is a technological solution aimed at connecting family farmers and consumers of organic products through an application that facilitates the buying and selling of these products. The idea for this application originated in a Hackathon held at FGA in the year 2020, and since then, its development has been continued by students from UnB-FGA over the course of four graduation projects. The current solution consists of a mobile app for consumers, a server for each Community Supported Agriculture (CSA), and a dictionary API that connects consumers to the server of each listed CSA. Due to being a project developed by various hands over time without following a specific methodology to connect each of the projects, the application lacks documentation and a clear policy to guide its development and provide unity to the work done. In this context, the present graduation project organized the repositories, documented the functionalities, and integrated the assets developed into a single point of convergence to bring unity to the Agromart project. As a result, this reorganization enabled the publication of Agromart and turned easier the development of new functionalities in the future following the same unity.

Key-words: Agromart. community-supported agriculture (CSA). family farming.

Lista de ilustrações

Lista de abreviaturas e siglas

MVP	Mínimo Produto Viável
CSA	Comunidade que Sustenta a Agricultura
TCC	Trabalho de Conclusão de Curso
API	Application Programming Interface
App	Aplicativo, no contexto do trabalho, refere-se ao Aplicativo Móvel
UnB	Universidade de Brasília
FGA	Faculdade do Gama
AWS	Amazon Web Services
EBS	Amazon Elastic Block Store
EC2	Amazon Elastic Compute Cloud
NPM	Node Package Manager
CPU	Central Processing Unit
APK	Android Package Kit
SGBD	Sistema Gerenciador de Banco de Dados

Sumário

1 Introdução

A agricultura familiar é um pilar fundamental da economia brasileira, sendo responsável pela produção de grande parte dos alimentos consumidos no Brasil, como feijão, arroz, mandioca, leite e verduras garantindo emprego e renda para milhões de famílias (SILVA, 2024). Além disso, a agricultura familiar também se destaca por ser uma alternativa produtiva mais sustentável que contribui para a preservação do meio ambiente e é a principal fonte de produtos orgânicos.

Com o intuito de apoiar a agricultura familiar, em um *Hackathon* realizado no Campus Gama da UnB, surgiu a ideia do Agromart, que na época foi um dos projetos vencedores da competição e posteriormente foi continuada ao longo de cinco projetos de conclusão de curso.

O Agromart consiste em uma aplicação que tem o objetivo de conectar agricultores e co-agricultores para facilitar as relações comerciais de produtos oriundos da agricultura familiar. Tendo sua estrutura desenvolvida para ser compatível com o modelo de CSA, um modelo de agricultura sustentável que conecta consumidores e agricultores locais, em que os consumidores se comprometem a comprar uma cesta de alimentos periodicamente, e os agricultores se comprometem a fornecer alimentos frescos nesse período estabelecido.

1.1 História do Agromart

No ano de 2020, em um *Hackathon* realizado no campus Gama da UnB cujo tema era: Cultivando Conexões e o objetivo era propor soluções tecnológicas que poderiam auxiliar a agricultura familiar, surgiu a ideia do Agromart. Inicialmente a proposta consistia de um aplicativo móvel em que os produtores pudessem divulgar os seus pontos de venda, as informações sobre os produtos vendidos, preços, localização e meios para o contato, permitindo que através dessas informações disponibilizadas, os consumidores interessados pudessem ter acesso aos agricultores mais próximos, facilitando assim a relação entre as partes. (RODRIGUES L. S.; MACEDO, 2021)

O projeto inicial conquistou o primeiro lugar do *Hackathon*, então os alunos decidiram dar continuidade ao projeto através de um TCC no qual foi desenvolvido um novo aplicativo que incorporava o design do primeiro e buscava atender os novos requisitos levantados através de conversas com agricultores e professores. (RODRIGUES L. S.; MACEDO, 2021)

O novo projeto tinha como ideia central o desenvolvimento de uma aplicação adequada ao modelo de parceria de uma CSA (Comunidade que Sustenta a Agricultura). Em

uma CSA, se estabelece uma relação de produção e distribuição de produtos agroecológicos, em que o consumidor assume o papel de co-agricultor ao se comprometer por meio de uma assinatura a receber uma cesta de produtos previamente definidos pelo agricultor de forma periódica, com a possibilidade de adicionar outros produtos da estação oferecidos e pagos em separado. Desse modo, o novo aplicativo proposto tinha como objetivo disponibilizar ao agricultor as seguintes funcionalidades: adição de novos produtos, definição da quantidade de cestas disponíveis, definição dos planos de assinaturas, e informações relativas à data, meio e local para realização da entrega ou coleta dos produtos. (RODRIGUES L. S.; MACEDO, 2021)

Nos anos seguintes, o desenvolvimento do Agromart continuou por meio de quatro trabalhos de conclusão de curso. O primeiro trabalho realizado no ano 2021, foi desenvolvido com o objetivo de alterar a arquitetura do sistema para que cada CSA pudesse criar sua própria instância do Agromart, retirando, assim a responsabilidade dos alunos e da faculdade pela manutenção dos servidores do aplicativo. Aos mantenedores do projeto Agromart caberia apenas a tarefa de manter uma API dicionário em que seriam registrados os endereços de cada um dos servidores das CSA's que possuem sua própria instância do Agromart (CELLA V. S. C.; FREITAS, 2023). O segundo trabalho desenvolvido no ano de 2022, tinha como objetivo propor uma forma de pagamento dentro do aplicativo para facilitar as relações comerciais entre agricultor e co-agricultor. (CORRÊA B. K. B.; VELUDO, 2022). Posteriormente, no ano de 2023 foi desenvolvido um trabalho com o objetivo de integrar o meio de pagamento dentro da aplicação do Agromart.(AGUSTINI F. B. S.; BOTTINO, 2023) Ainda no ano de 2023, foi desenvolvido um trabalho que tinha como objetivo utilizar um serviço de Lambda na AWS para hospedar o servidor da API dicionário do Agromart, com o objetivo de reduzir os custos de hospedagem. (RIBEIRO A. F. C.; MAGALHÃES, 2023)

1.2 Arquitetura e implementação do Agromart

O Agromart é composto por três componentes principais: o aplicativo móvel, a API Dicionário e a API principal de cada CSA, como mostra a Figura 1. Cada um desses componentes está descrito brevemente em seguida.

O aplicativo móvel é utilizado por todos os co-produtores cuja CSA utiliza a ferramenta Agromart. Nesse aplicativo eles podem realizar pedidos e comunicar com a CSA à qual pertencem. As tecnologias utilizadas no aplicativo móvel são: TypeScript, que é uma versão tipada da linguagem de programação JavaScript (SHUBEL, 2022); React Native, um *framework* para desenvolvimento mobile multiplataforma (BUDZIńSKI, 2024); e Expo, uma plataforma que simplifica o desenvolvimento de aplicativos usando React Native (FERNANDES, 2018). Atualmente, o aplicativo está disponível apenas na versão

Android, mas, por ter sido desenvolvido utilizando o React Native, que é uma ferramenta multi plataforma, é possível, futuramente, com poucas adaptações, disponibilizar uma versão iOS do aplicativo (BASUMALLICK, 2024).

A API principal da CSA, tem o objetivo de salvar os dados dos produtos, planos, clientes e lojas de uma CSA. Cada CSA deve ter seu próprio servidor, desse modo, o Agromart é uma aplicação que funciona de forma decentralizada, sem qualquer interação entre os servidores de cada instância CSA. A API da CSA é feita na linguagem JavaScript (GRIFFITHS, 2024), utilizando a ferramenta Strapi como Sistema Gerenciador de Conteúdo (GADHAVI, 2023). Além de ser consumida pelo aplicativo, a API principal também disponibiliza um painel de administração da CSA, onde é possível cadastrar produtos, visualizar pedidos entre outras funcionalidades.

A API Dicionário tem o objetivo de registrar as URLs de cada uma das CSAs que utilizam o Agromart para que, ao utilizar o App do Agromart, o usuário possa através dos endereços cadastrados na API Dicionário se conectar o servidor da CSA Desejada. A API Dicionário foi desenvolvida a linguagem JavaScript e o PostgreSQL como Sistema Gerenciador de Banco de Dados (SCOTT, 2024).

Anteriormente, existia um repositório web para os administradores das CSAs, mas, em trabalhos anteriores, esse painel foi arquivado, pois o painel de administrador disponibilizado pelo Strapi acessível através da API principal era uma forma mais eficiente de utilizar as funcionalidades de administrador da CSA.

1.3 Problema

Desde 2020, quando a ideia do Agromart surgiu no *Hackathon* que ocorreu na UnB-FGA, o projeto foi continuado ao longo de cinco trabalhos de conclusão de curso, tendo recebido contribuições de diversos alunos diferentes no desenvolvimento da aplicação. No entanto, esse desenvolvimento ocorreu sem seguir padrões claros para integrar as contribuições recebidas ao longo dos diferentes trabalhos realizados, o que comprometeu a organização do projeto como um todo.

A existência de diversos trabalhos que implementavam funcionalidades distintas mas não consideravam o Agromart como um todo, ao longo dos anos, comprometeu a unidade do projeto. Tal falta de unidade, pode-se notar através dos gráficos de *branches* dos repositórios do servidor (Figura 2) e do aplicativo mobile (Figura 3), onde pode-se verificar a existência de várias *branches* distintas que não foram unificadas no ramo principal de desenvolvimento.

Além disso, a ausência de testes gerais feitos a cada nova implementação que garantisse o funcionamento da aplicação como um todo, tanto servidor quanto aplicativo

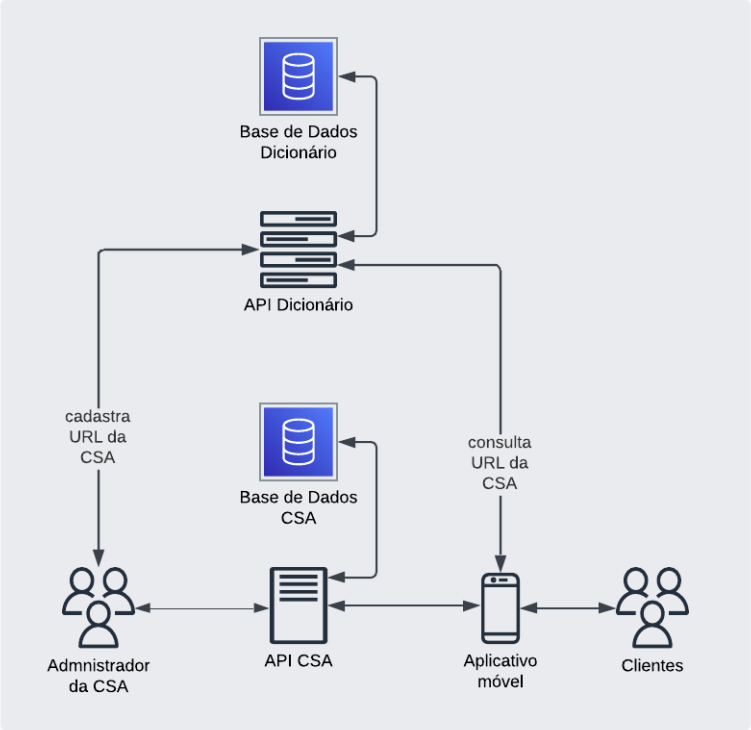

Figura 1 – Diagrama de Arquitetura de Software Agromart

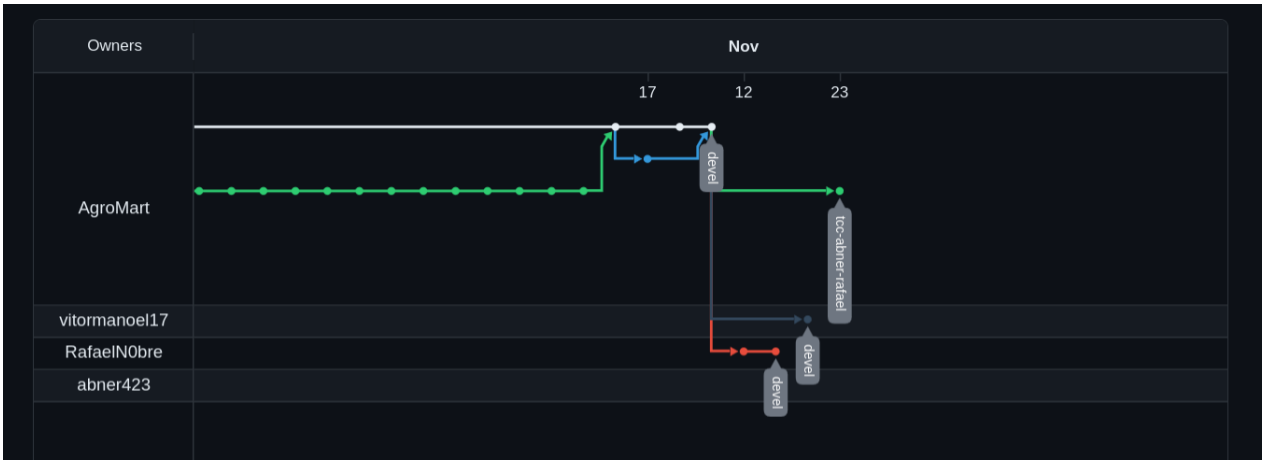

Figura 2 – Visualização gráfica das *branches* da API Agromart

mobile, causou incompatibilidades entre o aplicativo e o *back-end*, comprometendo o funcionamento da aplicação. Um exemplo dessa incompatibilidade, é a ausência de modificações para se integrar o aplicativo mobile com a nova arquitetura do *back-end*, implementada no trabalho (CELLA V. S. C.; FREITAS, 2023), que utiliza uma API dicionário para listar as instâncias de CSAs que estão rodando o Agromart, para que o usuário selecione a instância desejada. Como consequência dessa incompatibilidade, a API dicionário não está sendo consultada pelo aplicativo na *branch* devel, e mesmo nas *branches* mais atualizadas, o aplicativo não está 100% integrado com a arquitetura da API dicionário.

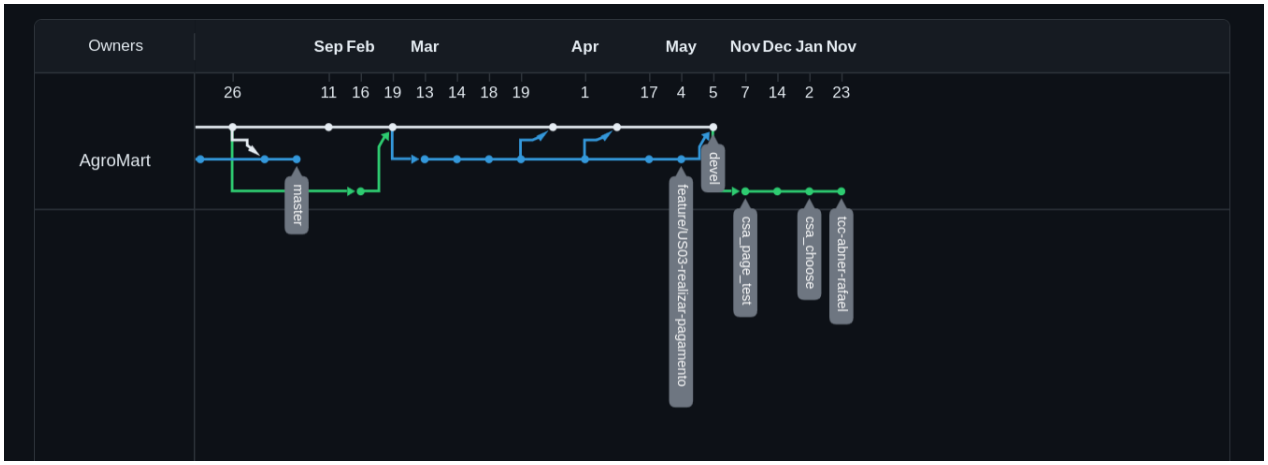

Figura 3 – Visualização gráfica das *branches* do Aplicativo Mobile

Desse modo, as inconsistências decorrentes da falta de unidade do projeto como um todo, inviabilizam a publicação do aplicativo para download em lojas virtuais e a distribuição de uma versão do servidor do Agromart que permita ser instanciada e executada por qualquer CSA interessada na aplicação.

Outro efeito negativo de haver um sistema que não está totalmente integrado e em pleno funcionamento é a inviabilidade de se receber contribuições externas no projeto, impossibilitando plena realização do carater *open-source* da ideia original, que teria como possibilidade a recepção de contribuições para além da FGA, já que torna-se extremamente complicado contribuir para um sistema que apresenta erros de compilação ou em sua execução.

1.4 Objetivo Geral

Diante do exposto, surgiu a necessidade da realização de um trabalho que tenha como objetivo dar a unidade necessária para aplicação do Agromart, possibilitando que aplicativo móvel seja disponibilizado para download em lojas virtuais, que o Agromart possa ser utilizado por uma CSA, além de garantir que os futuros trabalhos tenham um ambiente mais propício para o desenvolvimento.

Este trabalho envolve a análise dos repositórios do Agromart em cada *branch* existente, a realização de testes para detectar inconsistências nas funcionalidades do aplicativo móvel e da API principal, a correção de erros, defeitos e falhas para garantir o funcionamento adequado das funcionalidades básicas da aplicação, e a integração dos códigos de cada *branch* em um novo marco zero dos repositórios do Agromart. Além disso, inclui a publicação do MVP (mínimo produto viável) do aplicativo móvel e do servidor para as CSAs, juntamente com uma documentação detalhada deste processo e outras atualizações de documentação das funcionalidades existentes e da arquitetura nos repositórios da API,

do dicionário e do aplicativo móvel.

1.5 Objetivos Específicos

1.5.1 Unificação das *Branches*

Durante o desenvolvimento de software, é comum que diferentes versões do código sejam criadas em diferentes ramificações chamadas de *branches* para testar novas funcionalidades ou realizar correções sem afetar a versão principal do projeto. Neste objetivo, as diferentes *branches* serão mescladas, formando uma nova versão unificada do projeto. Isso inclui o entendimento do conteúdo de cada *branch* como também a resolução de conflitos de código que possam surgir durante a integração.

1.5.2 Testes Funcionais Pós-Unificação

Após a unificação do código, realizar uma série de testes no aplicativo. Esses testes são fundamentais para verificar se todas as funcionalidades estão operando conforme esperado e se as correções realizadas surtiram o efeito desejado. A qualidade do software será beneficiada através destes testes.

1.5.3 Correção de erros, falhas e defeitos

Após a unificação das *branches*, corrigir defeitos, erros e falhas identificadas no aplicativo móvel. O objetivo é garantir que o aplicativo opere de maneira estável e que os usuários possam utilizá-lo sem enfrentar problemas técnicos.

1.5.4 Configuração do Ambiente

A configuração do ambiente envolve preparar um espaço adequado onde o aplicativo possa ser desenvolvido e testado de forma eficaz. Isso inclui a configuração de servidores, bancos de dados e outros recursos necessários para garantir que a versão unificada do Agromart, possa ser desenvolvida e executada em um ambiente controlado.

1.5.5 Publicação do Aplicativo na Play Store

A Google Play Store é uma plataforma digital de distribuição de aplicativos para dispositivos Android. Publicar o aplicativo na Play Store significa disponibilizá-lo para que qualquer pessoa com um dispositivo Android possa baixá-lo e usá-lo. Esta etapa envolve não apenas a submissão do aplicativo para revisão pela Google, mas também garantir que ele esteja em conformidade com as políticas da plataforma e que sua versão publicada funcione corretamente.

1.5.6 Atualização das Documentações

Com a finalização das etapas anteriores, atualizar as documentações do projeto para refletir as mudanças realizadas. Documentar todas as funcionalidades existentes no aplicativo e na API principal, bem como uma explicação sucinta da arquitetura do sistema. Documentações atualizadas são essenciais para que outros desenvolvedores possam entender o estado atual do Agromart e contribuir com ele no futuro. Essas atualizações foram feitas no arquivo README de cada repositório.

1.6 Estrutura do Trabalho

A estrutura do presente trabalho se divide nos seguintes capítulos:

- **Referencial Teórico**: Este capítulo destaca a base conceitual e o referencial teórico usado para o desenvolvimento do trabalho, mostrando como cada fundamento foi utilizado.
- **Organização Github**: Este capítulo mostra o estado inicial dos repositórios da organização do Agromart no Github com a situação de cada ramo de desenvolvimento deixado em aberto ao longo do tempo.
- **Correção de erros, falhas e defeitos**: Esta seção mostra as correções de erros, falhas e defeitos realizadas para garantir o funcionamento do MVP do Agromart.
- **Alterações de documentação**: Este capítulo explica quais foram as alterações de documentação em cada repositório.
- **Publicação**: Este capítulo aborda o processo de publicação da versão inicial do aplicativo Agromart na loja PlayStore.
- **Produto Disponibilizado**: Este capítulo detalha as funcionalidades do produto disponibilizado na publicação da primeira versão do aplicativo Agromart.
- **Apresentação do produto**: Este capítulo aborda a apresentação do Agromart para a CSA da Florestta, após a disponibilização do produto.
- **Considerações finais**: Este capítulo discute sobre os pontos principais do trabalho realizado e discorre sobre sua conclusão.

2 Referencial Teórico

2.1 Engenharia de Software

A Engenharia de Software é uma disciplina de Engenharia que visa apoiar o desenvolvimento de produtos de software no âmbito profissional, que aplicamos também no mundo acadêmico. Tal estudo não considera apenas código, mas todo o ambiente de documentação e configuração necessário para o funcionamento de um sistema, dado que um sistema real, muitas vezes é composto por diversos programas menores, junto com seus arquivos de configuração (SOMMERVILLE, 2007). Isso é especialmente verdade no Agromart, que hoje em dia é composto de um aplicativo móvel, uma API dicionário, e uma API principal com os dados da CSA, cada um com diferentes tecnologias empregadas e arquivos de configuração.

Por ser uma disciplina da engenharia, e sendo o foco da engenharia a solução de problemas, não seria diferente na Engenharia de Software, sendo assim, ela abrange todos os aspectos da produção do Software, incluindo todo o ciclo de vida do produto (SOMMERVILLE, 2007).

2.1.1 Importância da Engenharia de Software

Frequentemente, a utilização de métodos e técnicas da Engenharia de Software para o desenvolvimento de sistemas é mais economicamente viável a longo prazo em comparação com a abordagem de escrever código como se fossem projetos pessoais. Em muitos casos, boa parte dos custos estão associados às alterações e correções no software após o início da sua implementação (SOMMERVILLE, 2007). No caso do Agromart que já está sendo desenvolvido há aproximadamente quatro anos, podemos pensar em termos de tempo gasto no projeto em vez de dinheiro, sendo assim a Engenharia de Software uma ferramenta poderosa para aumentar a eficiência dos colaboradores do projeto.

2.1.2 Diferença entre engano, erro, defeito e falha

De acordo com SOMMERVILLE [p. 207], é essencial compreender a distinção entre erros de usuário, erros de sistema, defeitos e falhas no contexto de sistemas de software.

- **Engano:** É o comportamento humano que resulta na introdução de defeitos em um sistema. Por exemplo, problemas de lógica que vão causar defeitos no sistema em casos específicos.

- **Defeito de sistema:** Uma característica do software que pode levar a um erro de sistema. É causado por um engano humano.
- **Erro de Sistema:** É a existência de estado errôneo do sistema que pode levar a um comportamento inesperado. Pode ocorrer quando o código defeituoso é executado.
- **Falha de Sistema:** É o evento que ocorre quando o sistema não fornece o serviço esperado pelos usuários, ou seja a consequência de erros de sistema.

Essas são importantes para uma melhor distinção dos termos, que em muitos contextos são utilizados com o mesmo significado. Nota-se também uma forte relação de causalidade nos termos, já que um engano pode causar defeitos, os defeitos geram erros e por fim os erros podem gerar falhas do sistema para o usuário final.

2.1.3 Atividades da Engenharia de Software

De acordo com SOMMERVILLE, numa abordagem sistemática de engenharia de software, existem quatro atividades fundamentais comuns que levam à produção de um produto de software, são elas:

1. Especificação de software, onde há trabalho em conjunto entre clientes e engenheiros para definir qual software a ser produzido e as restrições de sua operação.
2. Desenvolvimento de software, em que o software é programado a partir dos requisitos elicitados na atividade anterior.
3. Validação de software, que compreende entender se o software construído é o software é o que o cliente estava imaginando, ou em outras palavras, se o que foi feito está de acordo com o que o cliente tinha em mente.
4. Evolução de software, em que o software sofre manutenção e modificações de acordo com problemas encontrados, desejo do cliente, bem como mudanças do mercado e de tecnologias.

No Agromart, até por sua característica *open-source*, não há um cliente específico, onde quem desempenhou o papel de cliente até então foram os próprios contribuidores e desenvolvedores do projeto.

2.1.4 Evolução de Software

Alterações em código fonte de software não cessam quando o sistema é entregue e continuam por toda a vida útil do sistema. Isso quer dizer que, qualquer software que ainda

está sendo utilizado, sofrerá naturalmente modificações mesmo após seu desenvolvimento inicial.

SOMMERVILLE chama o processo geral de alterações em sistema após sua liberação inicial de manutenção de software, destacando três diferentes tipos de manutenção:

1. A correção de defeitos envolve o conserto de defeitos, erros e falhas que foram introduzidos através de enganos ou de fatores externos, esses defeitos geralmente são expostos através de falhas que os usuários identificam usando o sistema.

2. Adaptação ambiental. Esse tipo de manutenção é necessário quando algum aspecto do ambiente do sistema como hardware, sistema operacional ou outro software de apoio sofre uma mudança. Um exemplo do que podemos considerar software de apoio são as bibliotecas instaladas no sistema que consistem em conjunto de funções, classes ou outros componentes pré-escritos e empacotados de forma que pode-se incorporá-las ao sistema. O sistema deve ser modificado para se adaptar a essas mudanças de modo a oferecer uma experiência sempre satisfatória ao usuário.

3. Adição de funcionalidade. Esse tipo de manutenção ocorre quando os requisitos de sistema mudam, adicionando novas regras de negócio ou alterando regras existentes.

Este trabalho foca principalmente na evolução do software Agromart, mais especificamente na manutenção do sistema, atuando diretamente na correção de defeitos encontrados antes e depois da unificação do sistema e na adaptação ambiental.

A adaptação ambiental foi extremamente necessária no Agromart, especialmente no aplicativo móvel, onde o ecossistema baseado no React Native e Expo evolui rapidamente, juntamente com as diversas bibliotecas utilizadas. Uma porcentagem considerável desses softwares de apoio já se encontrava depreciada ou até mesmo com versões incompatíveis causando diversos problemas com o sistema. Tal adaptação ambiental, no caso do Agromart, está também diretamente ligada à correção de defeitos, já que muitos defeitos foram resolvidos simplesmente por fazer atualizações e melhorias no software de apoio do sistema.

2.2 Testes de caixa preta

Uma das abordagens mais comuns para testes de software são os teste de caixa preta. A técnica consiste em testar o sistema sem se importar com detalhes de sua implementação ou detalhes internos, mas apenas com a camada mais exterior do programa, a que é utilizada pelo usuário final, focando completamente no comportamento do software em suas interações com o usuário. O objetivo é identificar situações em que o programa não se comporta de acordo com seus requisitos (MYERS; BADGETT; SANDLER, 2012).

Os testes de caixa preta foram especificamente eficientes neste trabalho para identificação de falhas, onde o foco foi em utilizar o sistema e identificar os pontos de falha sem necessariamente olhar para implementação do código-fonte. Dessa maneira torna-se mais fácil a tarefa de localizar defeitos de código, uma vez identificados quais os componentes do sistema em estado de falha.

2.3 Metodologia Ágil

A metodologia ágil surgiu como uma abordagem que visava reduzir o retrabalho e a lentidão de metodologias mais tradicionais em caso de mudanças nos requisitos, permitindo que as equipes focassem mais no Software (SOMMERVILLE, 2007). Este trabalho se beneficiou da metodologia ágil dada a imprevisibilidade de possíveis problemas na integração das *branches* e das funcionalidades do Agromart, uma vez que não possuíamos conhecimento do código fonte do sistema e nem exatamente quais os erros, falhas e defeitos encontraríamos pelo caminho.

2.4 *Open-Source*

O código aberto, ou *open-source*, refere-se a uma abordagem de desenvolvimento de software em que o código-fonte é disponibilizado publicamente, permitindo que qualquer pessoa o visualize, modifique e distribua de acordo com diretrizes específicas.(OPENSOURCE, 2023) O *open-source* tem como vantagem a transparência do código, que permite uma colaboração ampla de desenvolvedores externos, resultando em soluções com mais inovações e qualidade. Isso também significa que erros e problemas de segurança podem ser identificados e corrigidos rapidamente pela comunidade.

No contexto do Agromart, o caráter *open-source* da aplicação é fundamental para a plena realização dos seus objetivos iniciais, de se tornar uma ferramenta útil para facilitar as relações de agricultores e co-agricultores no modelo de CSA. Para isso é necessário que o projeto tenha cada vez mais independência do contexto da universidade, tendo a possibilidade de receber contribuições externas e da formação de uma comunidade para a manutenção da aplicação.

3 Metodologia

3.1 Scrum

O Scrum é uma metodologia Ágil criada por Jeff Sutherland junto com Ken Schwaber em uma época onde o principal método de desenvolvimento de Software era o Cascata, uma abordagem linear, pouco flexível, o Scrum veio como uma mudança radical e muito mais flexível de desenvolver Software, contando com Sprints, adaptação contínua e reuniões regulares. As Sprints são ciclos de desenvolvimento breves, e no início de cada uma, acontece uma reunião de planejamento da Sprint, onde a equipe decide uma quantidade de trabalho que será possível realizar nas próximas duas semanas. (SUTHERLAND, 2014)

Dada a natureza do trabalho e o tamanho da equipe, não utilizamos efetivamente todas as práticas do Scrum como Sprints e suas cerimônias, uma vez que atuamos majoritariamente na resolução de erros, defeitos e falhas, que naturalmente são tarefas difíceis de estimar. Implementar o Scrum adicionaria uma complexidade desnecessária a um projeto com objetivos tão claros e um time de desenvolvimento tão pequeno.

Ainda assim optamos por incorporar alguns elementos do Scrum, como a alta adaptabilidade e flexibilidade e também uma equipe de desenvolvimento auto gerenciável e multifuncional, composta pelos autores deste trabalho. Esta abordagem proporcionou flexibilidade e adaptabilidade ao longo deste trabalho, permitindo ajustes em caso de eventuais mudanças de planos ou adversidades, e também com que a equipe fosse suficiente em si mesma.

3.2 Extreme Programming

O Extreme Programming (XP) é uma metodologia ágil de desenvolvimento de software, segundo Kent Beck, esta metodologia tem como valores: comunicação, simplicidade, *feedback*, coragem e respeito. As práticas do XP enfatizam a flexibilidade, comunicação do time, entregas contínuas e *feedbacks* rápidos, para melhorar a produtividade da equipe e a qualidade do software. Esta metodologia tem como algumas de suas práticas a programação em par, desenvolvimento orientado a testes, integração contínua e pequenas entregas frequentes. (BECK, 2004)

Com o objetivo de orientar o desenvolvimento do trabalho, incorporamos ao nosso fluxo de desenvolvimento duas práticas características do XP, a programação em par e a integração continua. A escolha dessas práticas específicas se deu devido às características do time e as necessidades encontradas no escopo do projeto. Por se tratar de um projeto

desenvolvido por duas pessoas, a adoção da programação em par ocorreu de forma natural e proveitosa. Além disso, a integração de cada nova versão ao aplicativo publicado na Google Play Store e ao servidor hospedado na AWS, foi essencial para evitar a descoberta tardia de incompatibilidades entre versão publicada e ambiente local.

3.2.1 Programação em par

A programação em par é uma prática do XP em que dois desenvolvedores trabalham juntos em uma mesma máquina para escrever o código. Durante o desenvolvimento, ambos devem estar concentrados no código desenvolvido dando sugestões e compartilhando conhecimento. Durante o desenvolvimento em par existe dois papéis, o piloto e o copiloto. O piloto é quem escreve o código e copiloto é quem revisa o código em tempo real, os papeis não são fixos e podem ser trocados a qualquer momento.

Devido às características do projeto, a programação em par trouxe grandes benefícios. Por se tratar de um projeto realizado em dupla, a sua adoção se deu de forma natural. Ela foi fundamental principalmente no momento de encontrar soluções para falhas críticas que demandam a análise de vários possíveis cenários para encontrar a causa, possibilitando pensar em uma quantidade maior de cenários através de duas perspectivas diferentes.

3.2.2 Integração contínua

A integração contínua consiste na prática de disponibilizar o código desenvolvido na aplicação principal o mais rapidamente possível, evitando o acúmulo de grandes quantidades de novas funcionalidades, correções de defeitos e melhorias para serem incorporadas de uma só vez à aplicação. Essa prática oferece vantagens significativas, como a detecção precoce de erros e rápida integração.

A integração contínua foi essencial para o projeto, especialmente devido ao processo necessário para a publicação de um aplicativo na Google Play Store, que envolve análises que podem levar até sete dias. Com a integração contínua, foi possível obter *feedbacks* mais rápidos a cada nova versão publicada, evitando surpresas que poderiam ocorrer caso todo o código desenvolvido fosse integrado apenas ao final do trabalho.

3.3 Roadmap

Um *roadmap* é um diagrama que define passos necessários para alcançar um determinado resultado ao longo de um período. Serve como um guia visual para mostrar a direção e o progresso do trabalho.

A partir dos nossos objetivos foi elaborado um *roadmap* como apresentado na Figura 4. O *roadmap* foi utilizado com uma ferramenta para auxiliar o desenvolvimento e a organização das tarefas, entretanto, naturalmente algumas tarefas foram feitas paralelamente, contrariando a linearidade proposta no *roadmap*. Isso foi feito para não permitir que o *roadmap*, que é um artefato para nosso auxilio, viesse a prejudicar a eficiência da equipe na realização das atividades.

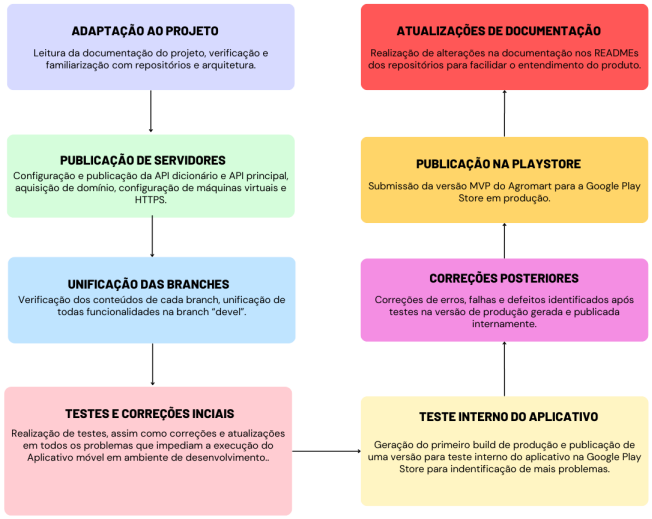

Figura 4 – *Roadmap* proposto

A seguir, cada subseção apresenta uma breve descrição das atividades propostas no *roadmap*.

3.3.1 Adaptação ao projeto

Essa atividade consiste na leitura da documentação do projeto e familiarização com repositórios, arquitetura e tecnologias empregadas.

3.3.2 Publicação de servidores

Se refere à configuração e publicação da API dicionário e API principal, aquisição de domínio e configuração de máquinas virtuais com certificado. Foi uma das atividades realizadas no início do trabalho para permitir maior produtividade na correção de erros, falhas e defeitos, já que os ambientes podiam ser acessados sem a necessidade de execução local.

3.3.3 Unificação das *branches*

A atividade consiste na verificação dos conteúdos de cada *branch*, unificação de todas funcionalidades na *branch* "devel". Essa atividade representa partida para o desenvolvimento.

3.3.4 Testes e correções

Realização de testes de caixa preta após a unificação das *branches*, assim como correções e atualizações em todos os problemas que impediam a execução do aplicativo móvel em ambiente de desenvolvimento. Essa atividade foi realizada em paralelo com as atividades descritas a seguir, pois os defeitos eram sempre corrigidos a medida com que eram descobertos.

3.3.5 Teste interno do aplicativo

Geração da primeiro versão de produção e publicação de uma versão para teste interno do aplicativo na Google Play Store para identificação de problemas antes da publicação definitiva.

3.3.6 Publicação na Play Store

Consiste na submissão da versão MVP do Agromart para a Google Play Store em produção.

3.3.7 Atualizações de documentação

Essa é a ultima atividade proposta no *roadmap*, que consiste na realização de alterações na documentação nos arquivos `README.md` dos repositórios para facilitar o entendimento do produto.

3.4 Versionamento de código

3.4.1 Git

O Git (GIT, 2024) é um sistema de controle de versão distribuído usado para gerenciar e rastrear alterações em projetos de software. Cada desenvolvedor mantém uma cópia completa do repositório, o que facilita o trabalho colaborativo de forma offline. Uma característica do Git muito importante no contexto deste trabalho é o conceito de *branches*, que são ramos de desenvolvimento separados, desse modo, é possível desenvolver funcionalidades de forma isolada e, posteriormente, integrar essas mudanças ao projeto principal através de uma ação que chamamos de *merge* ou mesclamento.

Nos dias de hoje o Git é uma ferramenta fundamental para o desenvolvimento de software, pois garante organização de versões e facilita o trabalho colaborativo '. Ele pode ser integrado com diversas plataformas que hospedam código-fonte como o Github, GitLab, Bitbucket e Azure Repos.

3.4.2 GitHub

O GitHub é uma plataforma que hospeda o código fonte de projetos que utilizam Git. Ele é usado para armazenar e facilitar o acesso à projetos de software que utilizam o Git para o seu versionamento (GITHUB, 2024). Além disso ele oferece recursos como *pull requests*, que possibilita solicitar a junção de *branches*, e revisão de código, em que é possível aprovar ou solicitar mudanças em uma *pull request*. Ele é amplamente utilizada tanto em projetos *open-source* quanto em ambientes corporativos, e é a plataforma onde o código fonte do Agromart é hospedado.

3.4.3 GitFlow

Após a unificação das *branches* já existentes, foi feito o uso do GitFlow, que é uma política de organização de versionamento de código criado com o intuito de melhorar o fluxo das *branches* nos repositórios. A Figura 5 apresenta a política de *branches* de forma detalhada. O GitFlow já é o padrão do Agromart, porém como ficaram muitas *branches* soltas sem serem devidamente fechadas, mostra que não foi utilizado corretamente em todos momentos, porém, neste trabalho, não foram deixadas *branches* soltas, proporcionando um ambiente onde a prática do GitFlow se torna muito mais fácil e intuitiva.

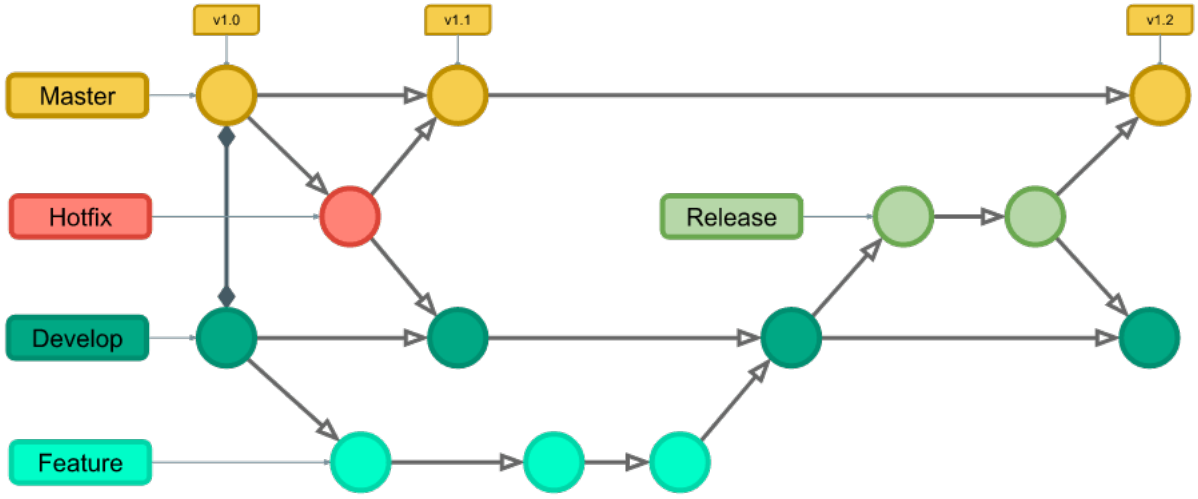

Figura 5 – Representação gráfica do GitFlow

Fonte: (ALURA, 2023)

4 Organização Github

4.1 Visão geral

A organização do Agromart no Github foi criada no ano de 2021, ela foi utilizada para o desenvolvimento do Agromart ao longo de alguns trabalhos de conclusão de curso. Durante esses trabalhos foram criados 7 repositórios. Repositórios do aplicativo móvel, API Dicionário, API principal, Frontend Web, que foi arquivado, dois repositórios de documentação e um repositório ".github" que insere a descrição da organização.

Ao longo dos diferentes trabalhos desenvolvidos, algumas funcionalidades foram descontinuadas, *branches* foram deixadas em aberto e não foram mescladas totalmente. Iremos focar esse trabalho nos três repositórios de código fonte utilizados pelo Agromart hoje, o da API dicionário, da API principal e do aplicativo móvel. Diante dessa situação surge a necessidade de analisar o que cada um desses repositórios e ramos de desenvolvimento contém, para sintetizar todo o trabalho feito de uma forma organizada para dar unidade ao Agromart e facilitar futuros desenvolvimentos.

4.2 Mapeamento de *branches* e unificação

Essa seção busca trazer mais informações sobre as *branches* existentes em cada repositório e qual seu conteúdo, bem como os resultados obtidos com a unificação destas na *branch* "devel". Como a API dicionário não possui mais de uma *branch*, ela não será abordada nessa seção.

4.2.1 Aplicativo móvel

O repositório do aplicativo móvel do Agromart se chama "mobile-client" e inicialmente continha quatro *branches* ativas. As *branches* ativas eram: "devel", "tcc-abner-rafael", "csa_choose", "csa_page_test", "feature/US03-realizar-pagamento" e "master", como é possível verificar na Figura 6.

Na *branch* "tcc-abner-rafael", havia algumas alterações, como atualização parcial da versão da SDK do Expo e algumas correções de erros em tempo de execução na tela de login.

Na *branch* "csa_page_test", havia a criação de um novo componente de seleção de CSA.

Na *branch* "csa_choose", haviam alterações da pastas *android* e *ios*, bem como

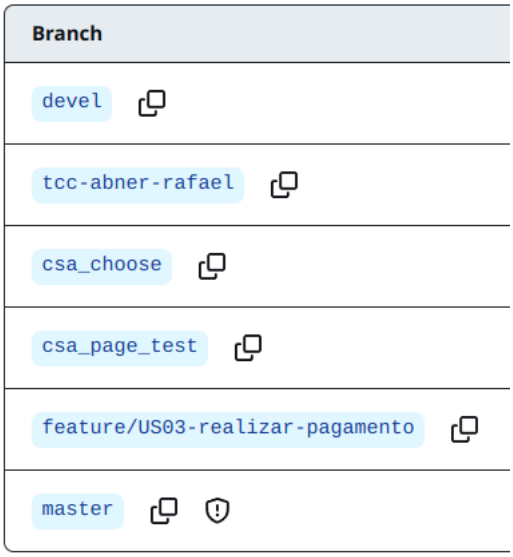

Figura 6 – Branches repositório do aplicativo móvel

alterações na seleção da url base através da API dicionário, todas alterações da *branch* "csa_page_test" também estão presentes nessa.

A *branch* "feature/US03-realizar-pagamento", já estava totalmente mesclada em "devel" e podia ser excluída de forma segura.

4.2.2 API principal

O repositório da API principal do Agromart se chama "api" e inicialmente estava com 4 *branches* ativas. As *branches* ativas eram: "devel", "tcc-abner-rafael", "automate-deploy" e "master", como é possível observar na Figura 7.

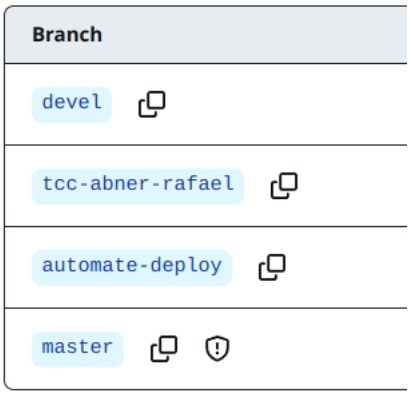

Figura 7 – Branches repositório da API principal

As *branches* "devel" e "master" são respectivamente as *branches* de desenvolvimento e principal

O código unificado dessas *branches* foi incorporado à *branch* de desenvolvimento, e neste repositório foram mantidas apenas as "devel" e "master".

A *branch* "automate-deploy" é onde foi desenvolvida a automatização de publicação, ela já estava com sua funcionalidade principal incorporada à "devel", tendo como diferença apenas alterações relacionadas à variáveis de ambiente.

A branch "tcc-abner-rafael" possuía algumas correções de erros da API principal.

4.2.3 Repositório API Dicionário

O repositório da API Dicionário do Agromart se chama "api-dicionario" e continha apenas uma *branch* "main", como é possível observar na Figura 8. Dessa forma, não foi necessário nenhum trabalho de análise e unificação de *branches*.

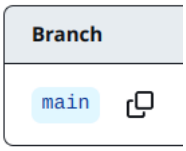

Figura 8 – *Branches* Repositório API Dicionário

4.2.4 Resultados

Após o mapeamento do que havia em cada *branch* e o entendimento de que elas possuíam códigos úteis, todas as *branches* foram mescladas em "devel" sem nenhum conflito, e a partir da *branch* de desenvolvimento, tanto no repositório do aplicativo quanto no repositório da API principal, iniciamos então a próxima etapa do trabalho, que seria a correção de defeitos até chegar em um MVP estável.

Ao final dos *merges*, como vemos na Figura 9, ambos repositórios ficaram apenas com duas *branches*, "devel" e "master", que serão o ponto de partida para o desenvolvimento das correções. Além disso, após todas as correções feitas e conclusão do trabalho, a *branch* "master" foi atualizada com o conteúdo de "devel".

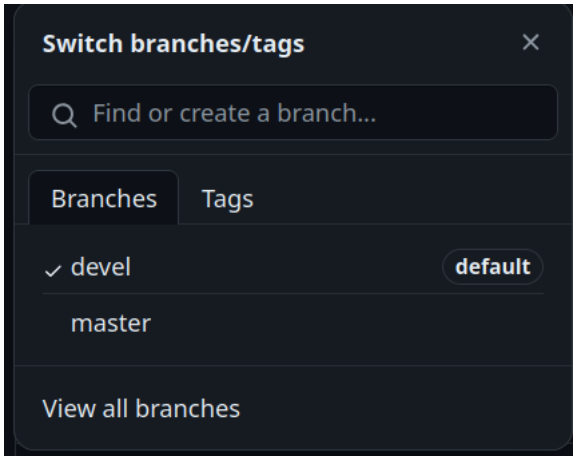

Figura 9 – Resultado após unificação

5 Correção de erros, falhas e defeitos

5.1 Visão Geral

Ao longo do projeto diversos defeitos, erros e falhas foram corrigidos, a maioria deles no aplicativo móvel. Logo nas primeiras tentativas de rodar o aplicativo localmente, foram identificados problemas relacionados ao Expo, plataforma utilizada para o desenvolvimento do aplicativo mobile, que impediam os testes, geração de *builds* e a sequência de desenvolvimento.

Build é o processo de conversão de arquivos de código em uma versão executável. Esses artefatos, também podem ser chamados de *builds*, e são o resultado final do processo de compilação. Então, o termo *build* pode se referir tanto ao próprio processo de geração quanto ao produto gerado ao final.

Após as correções iniciais relacionadas ao Expo, os esforços foram concentrados na correção de defeitos relacionados às funcionalidades do Agromart. A API principal do sistema e a API Dicionário demandaram apenas algumas correções pontuais. Então a maior parte dos esforços foram direcionados para a correção de defeitos, erros e falhas no aplicativo móvel.

Essas correções simplificaram a estrutura do projeto, reduziram a complexidade no desenvolvimento e garantiram que o Agromart estivesse preparado para atualizações futuras de maneira mais simples e organizada. A integração com o EAS também proporcionou um fluxo de trabalho mais ágil para o desenvolvimento e a manutenção contínua do aplicativo.

5.2 Erros relacionados à tipagem

Ao longo do desenvolvimento foram identificados diversos erros de tipagem no aplicativo, que foram corrigidos para garantir melhor entendimento do código facilitando o desenvolvimento, reduzindo o tempo e esforço necessários para alterações no código fonte.

5.2.1 TypeScript

O TypeScript é uma linguagem de programação que adiciona algumas funcionalidades ao Javascript, como tipos e interfaces, permitindo que os desenvolvedores detectem erros durante o desenvolvimento.

O uso do TypeScript traz vários benefícios, como menor probabilidade de erros por acesso de variáveis, propriedade, funções ou objetos não definidos e maior legibilidade e clareza do código.

Diferente de outras linguagens fortemente tipadas, o sistema de tipos do Typescript é usado para checar possíveis erros e avisos em tempo de compilação, já que o compilador pode identificar inconsistências e problemas de tipagem antes mesmo do código ser executado, por exemplo acesso a uma propriedade que pode possivelmente não estar definida em um objeto.

Essa segurança não se propaga em tempo de execução, uma vez que o código Typescript é transpilado para Javascript antes de ser executado, e o JavaScript por sua vez tem a tipagem dinâmica.

Isso explica o motivo desses erros de tipagem, apesar de gerarem confusão entre os desenvolvedores, não necessariamente causarão falhas em tempo de execução.

5.2.2 Erros Corrigidos

Para resolver esses problemas, primeiramente foi adicionado um *script* de *typecheck*, que verifica os erros de tipagem em todo o projeto e traz o endereço e a quantidade de erros de cada arquivo. Após rodar o *typecheck*, foram quantificados 100 erros em 65 arquivos diferentes, como é possível ver nas Figuras 10 e 48.

Para corrigir estes erros foram importadas bibliotecas de tipagens que deveriam estar sendo utilizadas, criação de interfaces para atributos sem qualquer tipagem, correção de interfaces incorretas que estavam causando conflitos, além da correção de partes do código que apresentavam erros na lógica e estavam sendo apontados pela tipagem.

Após a correção de todos estes erros relacionados ao TypeScript, o *typecheck* foi rodado novamente eliminando todos eles, como se pode ver na Figura 12. Como resultado, todos os 100 erros dos 65 arquivos diferentes foram corrigidos.

5.3 Falhas e erros relacionados a versões depreciadas ou incompatíveis e funcionalidades nativas

5.3.1 Aplicativo ejetado

O Agromart foi inicialmente construído utilizando o Expo, uma plataforma que facilita o desenvolvimento de aplicativos mobile ao fornecer ferramentas e bibliotecas prontas para uso. No entanto, o aplicativo estava configurado de forma ejetada, ou seja, ele havia sido removido de alguns dos sistemas de conveniência do Expo para possibilitar a utilização de recursos nativos que o Expo não oferece suporte. No caso do Agromart, essa

```
Found 100 errors in 65 files.

Errors  Files
     1  App.tsx:81
     1  src/components/AddressShortView/styles.ts:1
     1  src/components/BackHeader/styles.ts:1
     1  src/components/Button/styles.ts:1
     1  src/components/Carousel/index.tsx:33
     1  src/components/CartItemCard/styles.ts:1
     2  src/components/DropdownComponent/index.tsx:50
     1  src/components/ExtractModal/styles.ts:3
     1  src/components/HistoryItemCard/styles.ts:1
     2  src/components/Input/styles.ts:1
     1  src/components/NotificationCard/styles.ts:1
     1  src/components/PlanCard/index.tsx:36
     1  src/components/PlanCard/styles.ts:1
     1  src/components/ProductCard/index.tsx:4
     1  src/components/ProductCard/styles.ts:1
     1  src/components/ProductViewCard/index.tsx:3
     1  src/components/ProductViewCard/styles.ts:1
     1  src/components/ProfileItemAccess/styles.ts:1
     1  src/components/QuantityInput/styles.ts:1
     1  src/components/RACard/styles.ts:1
     1  src/components/RadioButton/styles.ts:1
     1  src/components/RegularText/index.ts:1
     1  src/components/Select/styles.ts:3
     1  src/components/StoreCard/index.tsx:3
     1  src/components/StoreCard/styles.ts:1
     1  src/components/TextButton/styles.ts:1
     3  src/components/UserHeader/index.tsx:22
     2  src/components/UserHeader/styles.ts:1
     1  src/hooks/CartProvider.tsx:26
     1  src/hooks/StoresProvider.tsx:14
     4  src/hooks/index.tsx:7
     1  src/pages/AddressForm/index.tsx:36
     1  src/pages/AddressForm/styles.ts:1
     2  src/pages/BillingAddress/index.tsx:32
     1  src/pages/BillingAddress/styles.ts:1
     2  src/pages/Cart/index.tsx:149
     1  src/pages/Cart/styles.ts:1
     4  src/pages/CreditCardRegister/index.tsx:166
```

Figura 10 – Primeira imagem de erros do TypeScript

ejeção foi desnecessária, pois o aplicativo não utilizava nenhum recurso nativos adicional. Essa configuração acabou introduzindo complexidade desnecessária ao projeto, desorganizando os diretórios e tornando o processo de desenvolvimento mais complexo.

Ter um Aplicativo ejetado estava causando diversos problemas para gerar uma *build* de produção realmente funcional, como erros de parâmetros do java, erros diversos de compilação e incompatibilidade de versões. Especificamente para o Agromart, um Aplicativo ejetado pode ser um grande causador de problemas, pois é um Aplicativo que não recebe manutenção tão frequentemente, e com o tempo podem surgir erros de *build* no Android quando novas atualizações do Expo são lançadas.

Para resolver esse problema, e além disso, proporcionar um ambiente de desenvolvimento mais simples e produtivo para o Agromart, foi criada uma nova versão do projeto, sem a ejeção, utilizando a versão da SDK 49 do Expo. A SDK é um conjunto de ferramentas e bibliotecas que os desenvolvedores usam para construir e manter o aplicativo.

Além disso, uma conta no EAS (Expo Application Services) foi criada para o

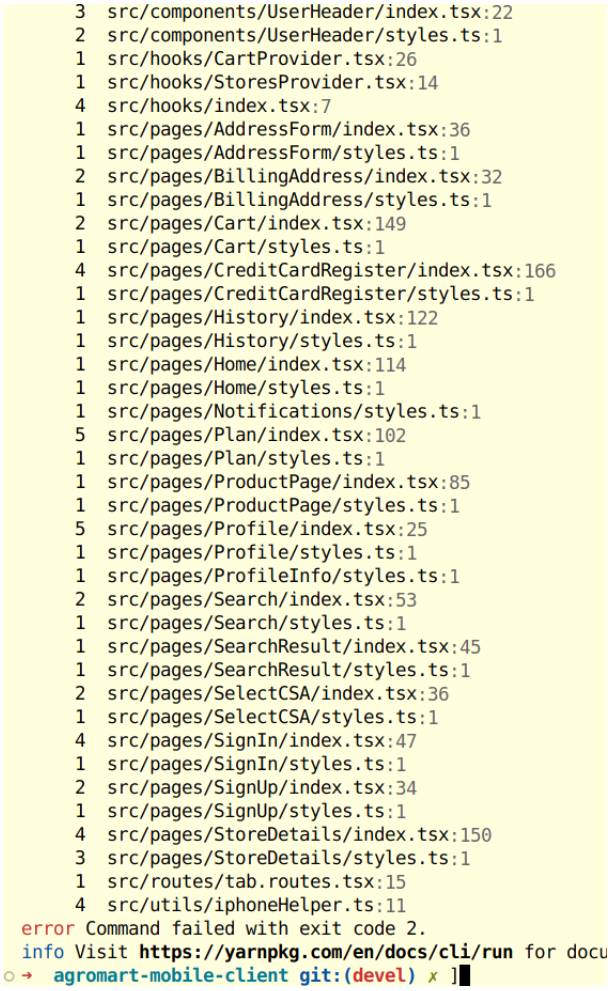

Figura 11 – Segunda imagem de erros do TypeScript

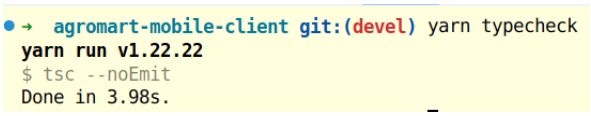

Figura 12 – Typecheck após correções

nosso projeto Agromart, note que a criação dessa conta não afeta nem possui relação com quaisquer outras contas criadas anteriormente para o aplicativo, pois utilizamos um nome de pacote diferente do original do Agromart, de forma que nosso projeto fique apartado. O EAS é um serviço do Expo que permite a gestão de *builds* e atualizações do aplicativo de forma centralizada e eficiente, facilitando o processo de desenvolvimento. Com o novo projeto configurado e o código do Agromart migrado para a SDK 49, todas as bibliotecas foram atualizadas para garantir a compatibilidade com a nova versão. Isso permitiu que o primeiro APK do aplicativo (arquivo de instalação utilizado em dispositivos Android) fosse gerado com sucesso na nova versão, resolvendo problemas de compatibilidade e trazendo maior organização ao projeto.

5.3.2 Padronização de gerenciamento de pacotes

Houve também a padronização do gerenciador de pacotes, o aplicativo do Agromart estava utilizando dois gerenciadores de pacotes diferentes, o NPM e o Yarn, e isso é uma má pratica. Por conta disso o arquivo `npm.lock` foi removido para manter apenas o `yarn.lock`, simplificando o gerenciamento das dependências do projeto. Além disso, uma série de bibliotecas foram atualizadas com as novas versões, e foi realizada a migração de bibliotecas que não eram mais suportadas, evitando problemas futuros de manutenção.

5.3.3 Expo Doctor

O Expo disponibiliza uma ferramenta chamada Expo Doctor, onde é possível visualizar problemas com o projeto, foram corrigidos todos os problemas que existiam, pois estavam nos impedindo de gerar versões de produção. Como podemos ver na Figura 13 , existiam muitos problemas de compatibilidade de versões e de bibliotecas instaladas. Já na Figura 14 vemos o comando rodando sem detectar nenhum problema após as correções efetuadas.

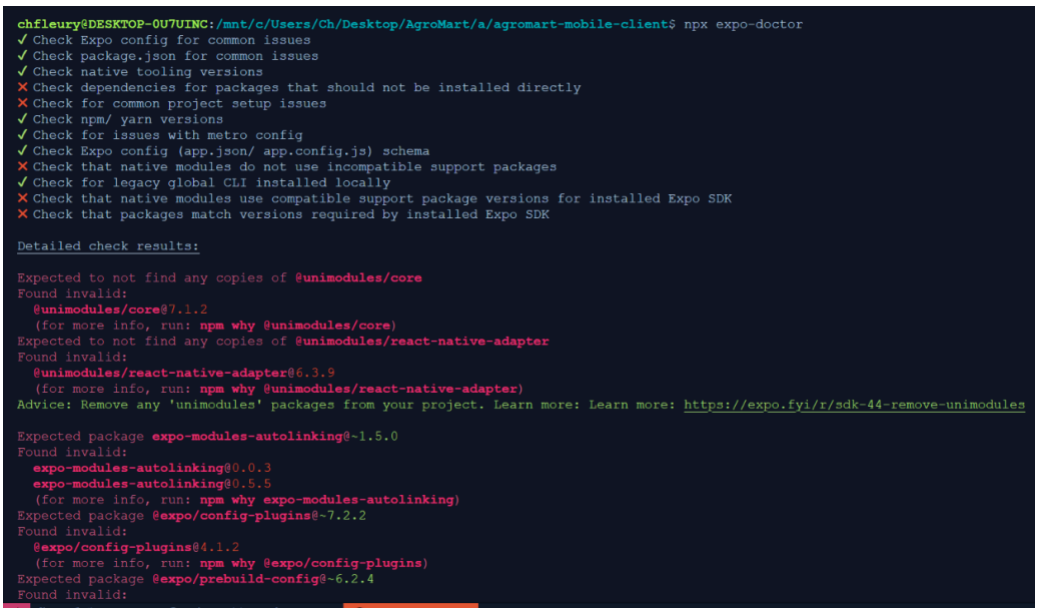

Figura 13 – Problemas encontrados pelo Expo Doctor

5.3.4 Problemas detectados após publicação

Além dos problemas detectáveis pelo Expo ou momento de compilação do projeto, existiam alguns erros em tempo de execução que eram causados por bibliotecas, como um *crash* no aplicativo que ocorria apenas em *builds* de produção causado por uma biblioteca depreciada, como mostrado na Figura 15. *Crash* se refere a uma falha abrupta que causa o fechamento do aplicativo móvel de repente. No defeito aqui descrito, o *crash* ocorria na tela

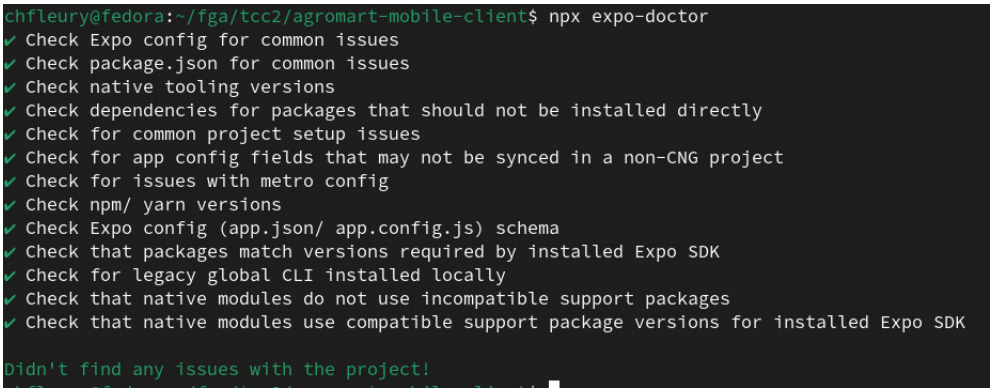

Figura 14 – Expo Doctor sem problemas após correções

de carregamento ao abrir o aplicativo. Esse problema só foi devidamente diagnosticado graças a uma versão de teste interno que foi publicada na loja antes da versão final de produção, através dela tivemos acesso ao relatório de causas de *crash* de aplicativo da Google Play Store, possibilitando a sua correção. Mais informações sobre o processo de publicação de versões de teste interno e sua importância podem ser encontradas no Capítulo 6 deste trabalho (ver seção de Publicação).

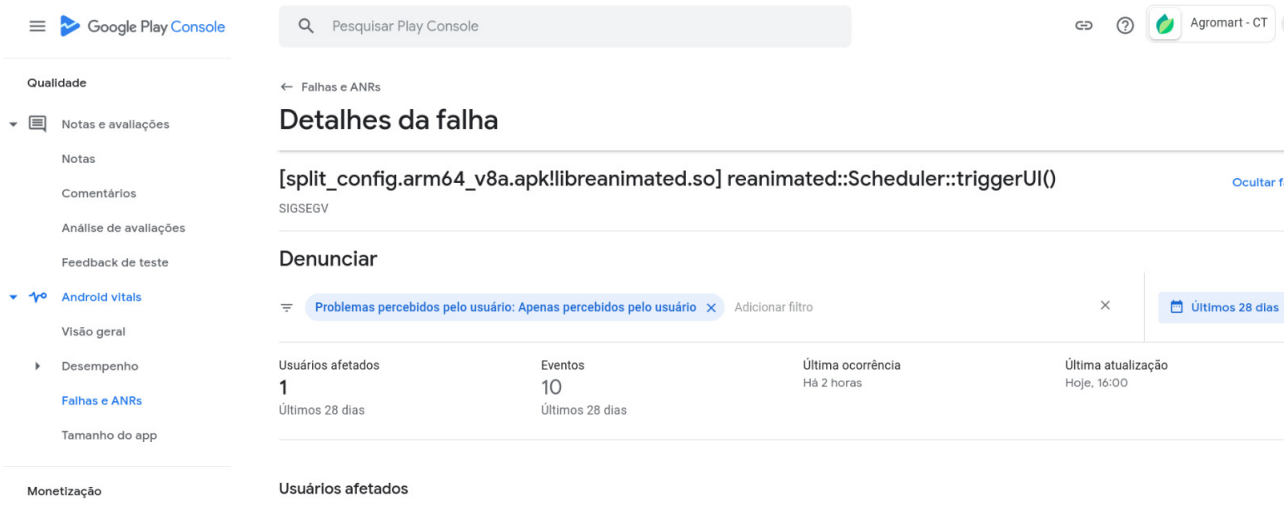

Figura 15 – Report de Crash causado pela biblioteca react-reanimated

5.4 Defeitos relacionados à funcionalidades do Agromart

Durante o desenvolvimento do aplicativo Agromart, foram realizadas diversas correções para garantir o correto funcionamento das funcionalidades essenciais do MVP.

5.4.1 Alterações no aplicativo móvel

Inicialmente, foi corrigido o uso da URL base da API Dicionário, que estava comprometendo a seleção da CSA ao iniciar o aplicativo. Essa URL base deveria agora apontar

para a API Dicionário que nós fizemos o deploy, visto que outros deploys da API dicionário feitos em semestres anteriores não estavam mais acessíveis.

Uma das correções mais importantes foi a padronização das chamadas de API de CSAS, pois em algumas telas, algumas chamadas eram feitas incorretamente para a API Dicionário, quando na verdade, deveriam ser feitas para a API da CSA selecionada. Também foi resolvido um problema relacionado ao *token* de autenticação, que afetava a comunicação entre o aplicativo móvel e o backend das CSAs.

Além disso, foram solucionados *crashes* na tela de histórico e na tela de pedidos, e um defeito que causava a exibição incorreta de compras de valor zero no histórico, como mostrado na Figura 16.

Figura 16 – Histórico exibindo todas as compras com valor de R$ 0,00

O redirecionamento após o login foi ajustado para garantir que os usuários fossem direcionados corretamente para a tela inicial após se logarem no aplicativo.

Erros relacionados à cadastro e edição do endereço do usuário também foram corrigidos, como vemos na Figura 17 era exibido um erro que não tinha relaçao com endereço e não era possível cadastrar um endereço de usuário pelo aplicativo nem edita-lo posteriormente.

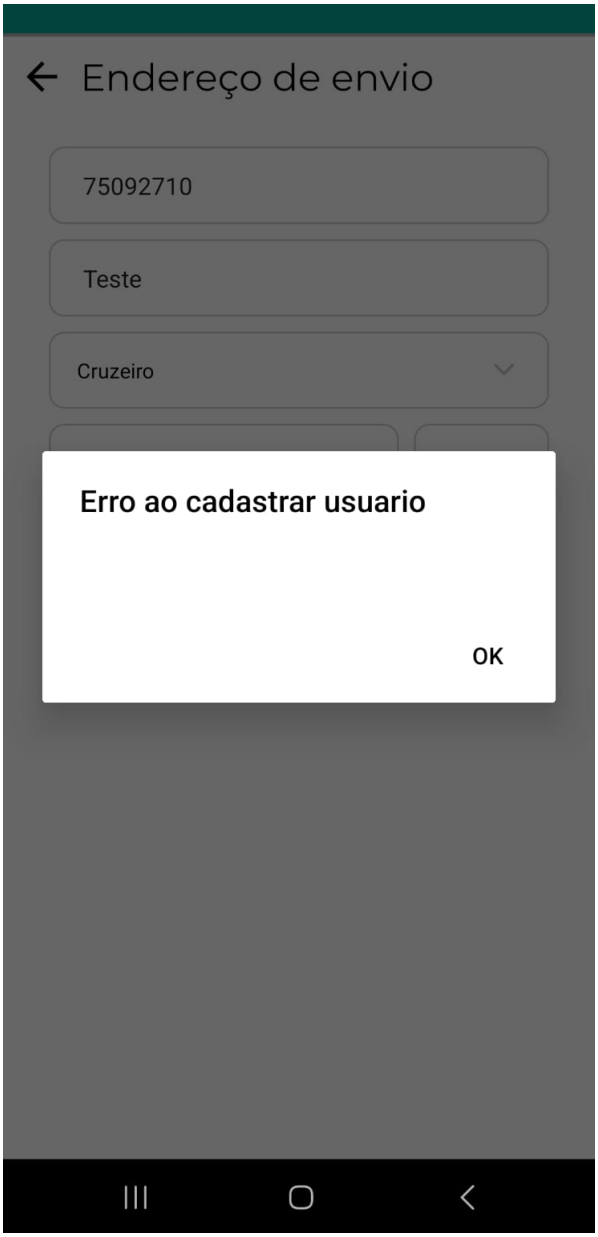

Figura 17 – Erro ao cadastrar usuário na tela de cadastro de endereço

Também houveram diversas correções de erros em tempo de execução causados por problemas de tipagem, acessos de propriedades que não estavam definidas, ou de fontes que não eram encontradas, como visto na Figura 18.

Outras correções incluíram a resolução de um problema em que o Aplicativo ten-

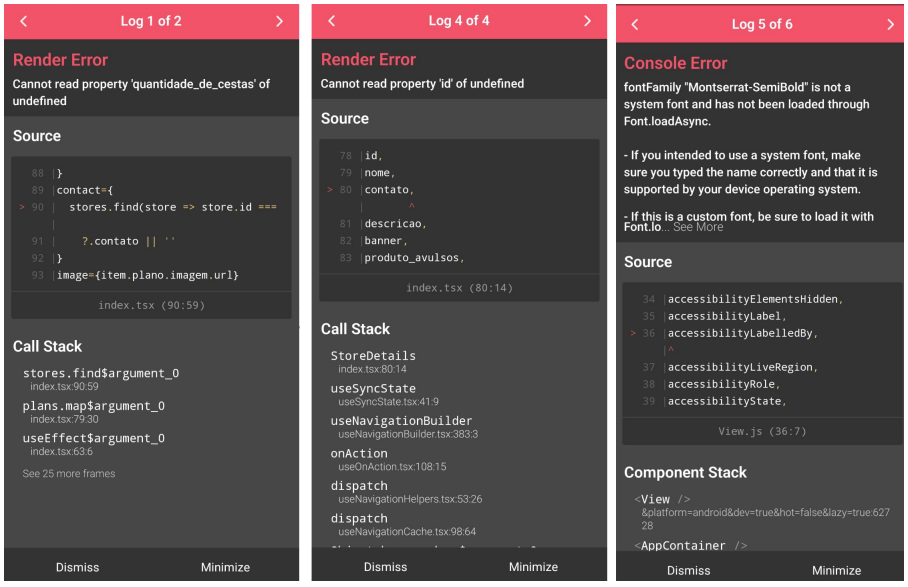

Figura 18 – Compilação de erros de execução

tava fazer a listagem adequada das lojas antes do login, o que não funcionava pois a URL base da CSA ainda não estava configurada. Melhorias na exibição de imagens em todo o aplicativo e a correção do filtro de histórico de compras. Também foi corrigido um problema relacionado à assinatura de planos, que anteriormente não salvava corretamente qual plano o usuário tinha assinado.

Essas correções foram essenciais para garantir uma experiência de uso fluida e confiável no Agromart.

5.4.2 Funcionalidade de pagamento por cartão de crédito

A funcionalidade de pagamento por cartão de crédito no Aplicativo, em vez de ser corrigida, foi desabilitada, uma vez que apresentava problemas críticos em ambiente de produção, impossibilitando com que pudéssemos receber pagamentos corretamente pelas plataformas do Mercado Pago ou PayPal. Foi tentado integrar a CSA fictícia que criamos com uma conta real do Mercado Pago, para que os pagamentos fossem feitos em ambiente de produção, porém apesar dos esforços, a funcionalidade não foi integrada corretamente.

Além disso o fluxo de pagamentos impedia a realização de qualquer pedido sem que fosse feito o pagamento por cartão adiantado pelo aplicativo, não possibilitando que fossem utilizados quaisquer outras formas de pagamento, como em dinheiro ou cartão na hora da entrega, o que compromete a usabilidade do aplicativo, e poderia frustrar tanto os usuários quanto as CSAs.

Dado esse cenário, a abordagem adotada foi de desabilitar essa funcionalidade, mas sem perder nenhum código que foi desenvolvido, o código apenas não é mais acessado. Dessa forma, o fluxo de pedidos fica mais simples e funcional, com o pedido sendo criado

com pagamento pendente, podendo ser pago de formas flexíveis como na entrega, em dinheiro, PIX ou qualquer outro meio de pagamento aceito pela CSA. Uma vez pago e/ou entregue, esses status podem ser atualizados pela administração da CSA no painel.

5.4.3 Alterações no Backend e na API dicionário

As principais correções necessárias se concentraram no aplicativo móvel, nas outras partes do sistema não houve a necessidade de muitas correções.

No Backend do Agromart foi realizada uma correção na exibição das imagens das cestas de produtos. Que não eram retornadas na resposta da requisição.

5.4.4 Pull-requests geradas e lista de alterações

Ao final de todo o trabalho, foram geradas duas *pull-requests*, já com todas as correções aplicadas, bem como código unificado de todas as *branches* existentes nos repositórios.

A *pull request* acessível em <https://github.com/AgroMart/api/pull/81> contém o código resultante da unificação das *branches* existentes no repositório e da correção de um defeito na exibição de imagens de cestas e produtos na API.

A *pull request* acessível em <https://github.com/AgroMart/mobile-client/pull/17> contém o código resultante da unificação das *branches* existentes no repositório do aplicativo e também das seguintes correções:

- Unificação das *branches* do aplicativo
- Correção no endpoint da API Dicionário
- Ajuste no uso da UrlBase proveniente do dicionário
- Correção no *token* de autenticação para a AGROMART-API
- Padronização do gerenciador de pacotes, removendo `npm lock` e mantendo apenas `yarn.lock`
- Criação de novo projeto com Expo na SDK 49, eliminando a necessidade de ejetar o aplicativo
- Migração do código do aplicativo para o novo projeto na SDK 49
- Atualização das bibliotecas compatíveis
- Migração de bibliotecas não suportadas
- Geração do primeiro APK após a atualização para SDK 49

- Correção de falha de *crash* na tela de histórico
- Correção de defeitos relacionados à compras de valor zero no histórico
- Correção de *crash* na tela de pedidos
- Ajuste no redirecionamento após login para a tela inicial
- Desativação do fluxo de pagamento por cartão de crédito
- Atualização das URLs no aplicativo
- Correção de defeito onde o aplicativo tentava listar lojas antes do login
- Correções na tela de planos
- Mensagem de "Erro ao cadastrar usuário" na criação/edição de endereço

6 Alterações de Documentação

O objetivo das alterações de documentação foi facilitar a vida dos desenvolvedores que contribuirão com o projeto no futuro. Facilitando o rápido entendimento do escopo do sistema e também sua arquitetura, o que é muito relevante visto que o sistema é dividido entre vários repositórios.

6.1 API principal e aplicativo móvel

Nos repositórios da API principal e do aplicativo móvel, foram realizadas atualizações nos arquivos `README.md`. Essas atualizações incluem imagens do funcionamento de algumas telas do aplicativo, além de breves descrições de todas as funcionalidades implementadas. Na API principal também foi atualizado para incluir instruções de como acessar o painel do administrador.

6.2 Repositório de documentação

No repositório de documentação, o arquivo `README.md` também sofreu atualizações, onde foi adicionada uma explicação detalhada sobre a arquitetura do Agromart, com um diagrama que descreve os principais componentes e como eles se relacionam, facilitando o entendimento da API dicionário e como ela faz o roteamento de múltiplas APIs principais que implementam diferentes CSAs, já que não é uma arquitetura muito usual. Dessa forma, a curva de aprendizado do sistema para novos desenvolvedores é menor.

7 Publicação

7.1 Visão Geral

O intuito dessa seção é documentar todo o processo de publicação dos três componentes essenciais, a API dicionário, API principal e o aplicativo móvel. Dessa forma, deixamos claro todos os passos e requisitos para ter o sistema Agromart rodando e funcionando em ambientes de produção, acessível ao público através da loja de aplicativos da Google.

7.1.1 Nome do aplicativo e do pacote java

Nosso objetivo nesse trabalho a nível de publicação foi corrigir os defeitos necessários para termos um MVP rodando, e publicar esse MVP custeando o valor dos servidores, e de forma com que essa publicação não interfira com publicações futuras do Agromart. Por isso decidimos publicar o aplicativo em uma conta pessoal, uma vez que ele não funcionará mais quando deixarmos de custear os servidores, e será removido da loja.

O nome do pacote deve ser único para cada aplicativo publicado na Google Play Store, por isso, utilizamos o nome de pacote `com.tcc.agromart` para gerar as *builds* do aplicativo, e não o nome de pacote padrão do Agromart que é `com.agromart`, dessa forma, o nosso deploy não interfere na publicação de outras versões futuras do Agromart em outras contas.

O aplicativo foi publicado com o nome Agromart - CT, onde CT significa Christian e Thiago, que foi usado para diferenciar o aplicativo da publicação oficial que será feita através de contas gerenciadas pela instituição UnB.

7.2 Processo de deploy dos servidores

A motivação de realizarmos a publicação dos servidores foi a necessidade de ambientes reais rodando, acessíveis via HTTPS, para que a versão de produção do aplicativo pudesse ser devidamente publicada na Google Play Store, uma vez que os aplicativos publicados passam por revisão manual. Sendo assim o aplicativo só poderia ser aprovado na loja se estivesse se comunicando com servidores reais e persistindo dados.

7.2.1 Serviço escolhido

O Amazon EC2 (Elastic Compute Cloud) é um serviço de computação em nuvem da AWS (Amazon Web Services) que permite aos usuários criar máquinas virtuais em nuvem com diferentes configurações de CPU, memória, armazenamento e rede. Essas instâncias podem rodar sistemas operacionais, aplicativos e outros serviços, permitindo que você execute cargas de trabalho como se estivesse utilizando servidores físicos, mas com a flexibilidade e escalabilidade da nuvem. Essas configurações podem ser facilmente manipuladas de acordo com a necessidade do usuário.

A principal razão para a escolha da EC2 na realização do deploy dos servidores foi a flexibilidade que o serviço oferece para modificar os recursos da instância alocada. Sem saber inicialmente quanto recurso seria necessário para o bom funcionamento do Agromart, a escolha de uma opção que permitisse a fácil manipulação dos recursos computacionais foi fundamental.

Levando em conta as necessidades listadas, foi alocada uma instância EC2 na AWS para disponibilizar a API Dicionário e o Backend do Agromart. A instância EC2 selecionada possui as seguintes características: tipo de instância t3.medium, com 2 vCPUs, 4 GB de memória RAM, e um volume de armazenamento de 30 GB em um SSD como visto na Figura 19. O sistema operacional escolhido foi o Linux, e a instância foi alocada na região da AWS da Virgínia do Norte (us-east-1).

Adicionalmente, também foram criados cópias do volume de armazenamento para recuperar o estado dos servidores em caso de falha. A configuração foi realizada com o objetivo de balancear adequadamente o custo e o desempenho.

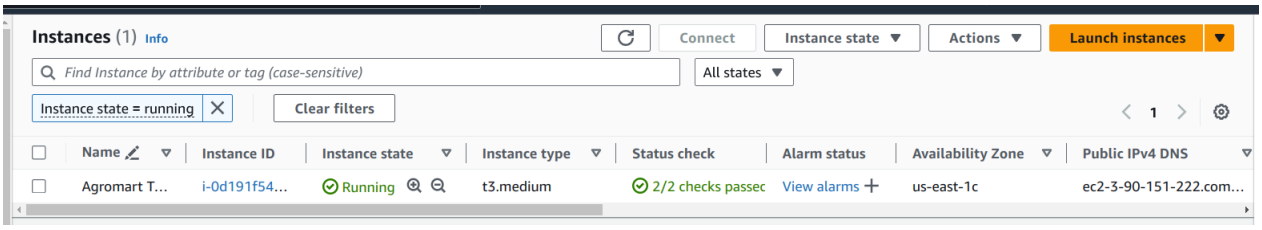

Figura 19 – Imagem da EC2 do Agromart

7.2.2 Preparação do Banco de Dados

Para a preparação do banco de dados, foi utilizado o SGBD PostgreSQL. A imagem oficial do PostgreSQL foi usada para criar o banco de dados em um ambiente conteinerizado. Essa abordagem foi escolhida por ser uma forma rápida de instalar e rodar o banco de dados no servidor.

7.2.3 Aquisição de Domínio

Um domínio foi essencial para conseguirmos um certificado válido, que é necessário para que possamos habilitar o HTTPS em nosso servidores. O domínio foi comprado através da plataforma Hostinger, e seu endereço é `agromarttcc.shop`, por um prazo de 1 ano, conforme apresentado na Figura 20.

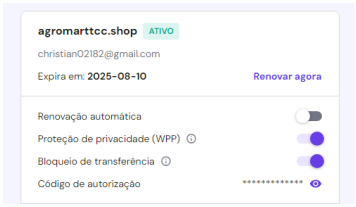

Figura 20 – Visualização do domínio na plataforma Hostinger

7.2.4 Configuração de HTTPS, domínio e certificado

HTTPS (Hypertext Transfer Protocol Secure) é uma versão segura do protocolo HTTP, usado para a comunicação entre navegadores web e servidores. A principal diferença é que o HTTPS criptografa os dados enviados e recebidos, garantindo maior segurança das informações transmitidas.

No caso do Agromart, disponibilizar nossos servidores através de HTTPS é extremamente necessário, pois chamadas puramente HTTP não são permitidas em aplicativos publicados na Google Play Store.

O Nginx é uma ferramenta poderosa e amplamente utilizada para servidores web, que pode ser usada como um servidor HTTP e *proxy* reverso, servidor *proxy* de e-mail e *proxy* genérico TCP/UDP. (NGINX, 2024). No contexto do Agromart, utilizamos o Nginx como *proxy* reverso dos nossos servidores (CSAs implantadas e API dicionário). Dessa forma todas chamadas feitas aos nossos servidores passam primeiro pelo Nginx, que encaminha para o servidor correto. O *proxy* reverso com Nginx também permitiu a implementação de um certificado que era necessário para o protocolo HTTPS.

O certificado necessário para habilitar o HTTPS foi obtido gratuitamente através da ferramenta Certbot, que nos proveu um certificado para o domínio `agromarttcc.shop`. A ferramenta Certbot é gratuita e de código aberto e serve para usar automaticamente os certificados do Let's Encrypt (uma Autoridade Certificadora sem fins lucrativos) em sites, a fim de habilitar HTTPS (FOUNDATION, 2024).

Com todas essas configurações feitas, a única coisa restante foi apontar o domínio para o IP público da máquina virtual onde os servidores estavam rodando, esse processo foi feito pelo painel da Hostinger.

7.3 Processo de deploy do Aplicativo na Google Play Store

7.3.1 Conta no Expo e EAS

O Aplicativo do Agromart foi criado utilizando uma ferramenta chamada Expo. O Expo é uma plataforma de desenvolvimento de aplicativos com React Native, que foi a tecnologia escolhida para o aplicativo em seu desenvolvimento.

o EAS (Expo Application Services) é um conjunto de ferramentas para deploy de aplicativos que utilizam o Expo. Utilizamos o serviço EAS Build, que permite compilar e gerar versões do aplicativo Android através da nuvem, sem necessidade de instalação de ferramentas para compilação localmente.

Para isso, foi criada uma conta do Expo específica para esse projeto de conclusão de curso, para *branch* da versão de produção que foi submetida à Google Play Store. Na figura 21, podemos ver a conta no Expo e algumas das *builds* de produção geradas.

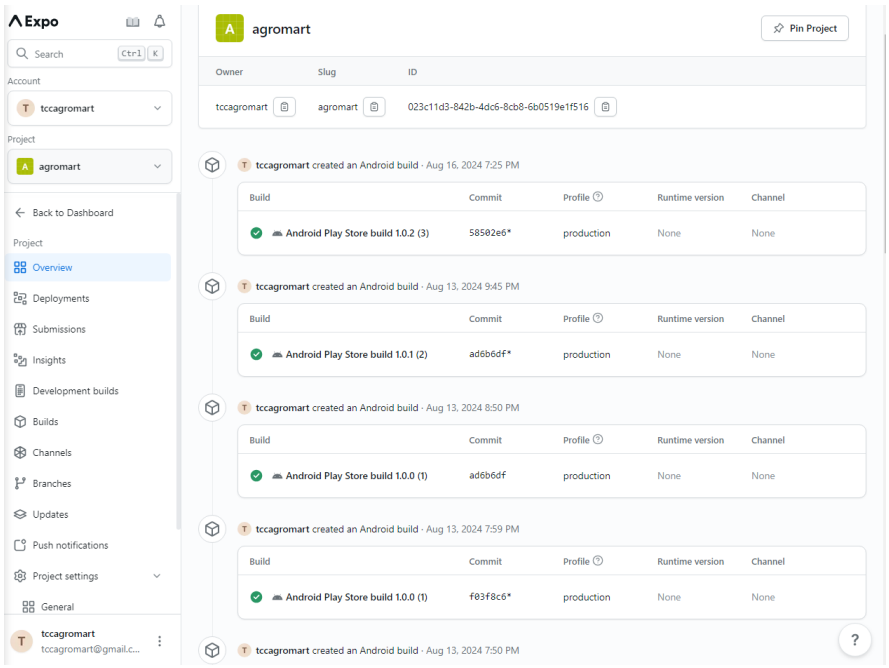

Figura 21 – Painel da conta Expo com *builds* realizadas

7.3.2 Geração de *builds* de produção

Uma *build* de produção é uma versão de um software que está pronta para ser lançada e usada pelos usuários finais. Após a correção de diversos defeitos, pudemos enfim concluir que tínhamos atingindo um MVP do Agromart, e gerar nossa primeira *build* de produção.

Após isso, executamos o comando `eas build --profile production --platform android`. Esse comando dispara a compilação em nuvem, gerando um arquivo com ex-

tensão `.aab`, que é a extensão aceita nas submissões à Play Store.

7.3.3 Conta de Desenvolvedor da Google Play Store

Para publicar aplicativos da Google Play Store, é necessário fazer um registro na plataforma Google Play Console e pagar uma taxa única de 25$, para isso, utilizamos uma conta de desenvolvedor que já havia sido adquirida anteriormente por um dos realizadores deste trabalho.

7.3.4 Versão de teste interno

O Google Play Console disponibiliza a possibilidade de serem publicadas versões para testes internos. Essas versões permitem os desenvolvedores lançar versões preliminares de seus aplicativos para um grupo restrito de usuários antes do lançamento oficial. Essa versão é ideal para realizar testes iniciais com um número limitado de testadores confiáveis, como membros da equipe de desenvolvimento, além de que não passa por uma revisão como a versão de produção, e podemos fazer uma publicação quase imediata. Através do teste interno, os desenvolvedores podem identificar e corrigir problemas, como defeitos e *crashes*. Os testadores são selecionados através de uma lista de endereços de email, e os dois autores deste trabalho foram incluidos nessa lista. Isso possibilitou que identificássemos um *crash* crítico ao abrir o Agromart que ocorria apenas em *builds* de produção.

7.3.5 Configuração do Aplicativo na Google Play Store e Publicação

É necessário a criação de um novo aplicativo através do painel do Play Console, e após isso fazer o *upload* do arquivo de *build* gerado, criando assim uma nova versão do aplicativo. Após isso, é necessário fazer uma configuração detalhada do aplicativo para enviar para revisão. Como visto na Na figura 22, há uma serie de informações obrigatórias como público alvo, recursos financeiros e política de privacidade. A política de privacidade foi gerada através de ferramentas online e está disponível em <https://www.freeprivacypolicy.com/live/b43f2c89-15ae-43e1-a55b-d3eec294da3b>.

Além disso, antes de enviar o aplicativo para revisão é necessário criar as instruções necessárias para o acesso ao aplicativo, uma vez que, de acordo com as políticas do Google, um revisor não pode criar contas nos aplicativos que irá revisar. Dessa forma, é necessário prover uma conta válida e instruções gerais de como se logar no aplicativo.

Após esse processo, o aplicativo pode ser submetido à revisão, e em um prazo de 7 dias úteis, será publicado na Google Play Store ou, caso haja alguma pendência, gerado um alerta para que sejam feitas correções no aplicativo antes de uma próxima revisão.

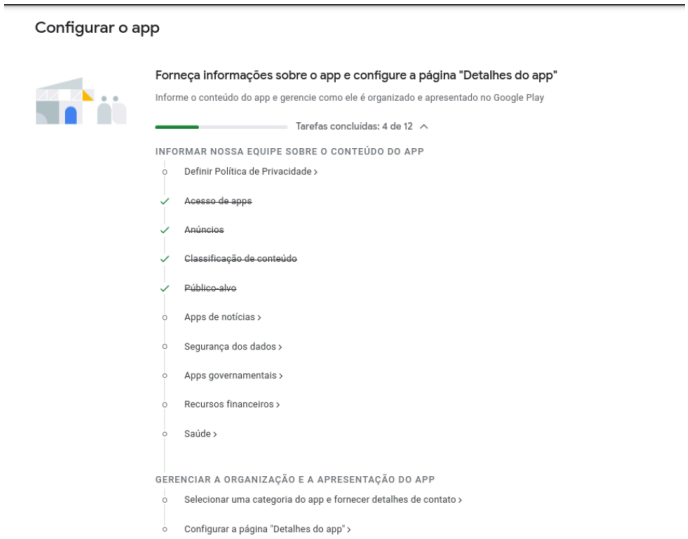

Figura 22 – Configuração do aplicativo no Play Console

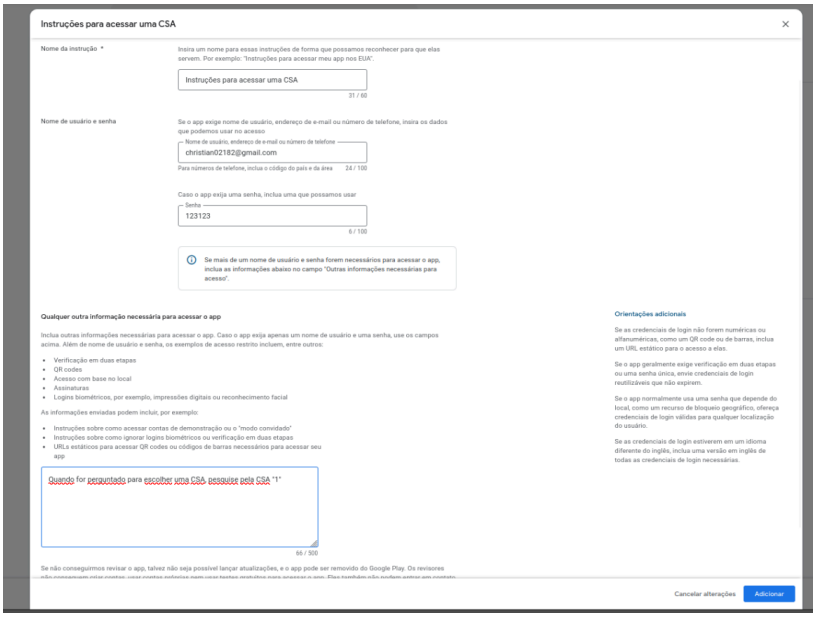

Figura 23 – Formulário de instruções de login

Nosso aplicativo foi aceito na revisão e publicado com sucesso. Ele encontra se disponível em Agromart - CT.

8 Produto Disponibilizado

Esta seção procura apresentar o produto que foi disponibilizado após a aplicação de todas as correções e ajustes necessários para garantir o funcionamento do Agromart. Além disso, foram realizadas atualizações na documentação dos arquivos README.md dos repositórios do aplicativo e da API principal. Tais atualizações foram realizadas para oferecer uma visão mais clara do funcionamento do Agromart, facilitando o entendimento e a utilização por parte dos usuários. Além disso, tal medida ajuda na compreensão dos futuros desenvolvedores colaboradores, de modo que possam entender rapidamente as funcionalidades do sistema.

8.1 Funcionalidades do Painel da CSA

O conteúdo de cada CSA é armazenado em um CMS (Sistema de Gerenciamento de Conteúdo), que possui um painel que possibilita que os administradores de uma CSA insiram as informações das lojas e dos seus produtos, além de possibilitar a visualização das informações de cada consumidor. Após o trabalho realizado, as seguintes funcionalidades estão disponíveis no MVP para serem utilizadas através do Painel de Gerenciamento da CSA:

- Gerenciamento de loja
- Gerenciamento de produtos avulsos
- Gerenciamento de cestas
- Gerenciamento de planos
- Gerenciamento de assinaturas
- Gerenciamento de usuários
- Gerenciamento de endereços
- Gerenciamento de pedidos

8.1.1 Gerenciar Lojas

Na tela de gerenciamento de lojas indicada na Figura 24, é possível ver as lojas já cadastradas pela sua CSA, criar uma nova loja, editar uma loja existente e excluir uma loja.

Figura 24 – Tela de lojas da CSA

8.1.2 Gerenciar Cestas

Na tela de gerenciamento de loja indicada na Figura 25, é possível ver as cestas já cadastradas pela sua CSA, criar uma nova cesta, editar uma cesta existente e excluir uma cesta.

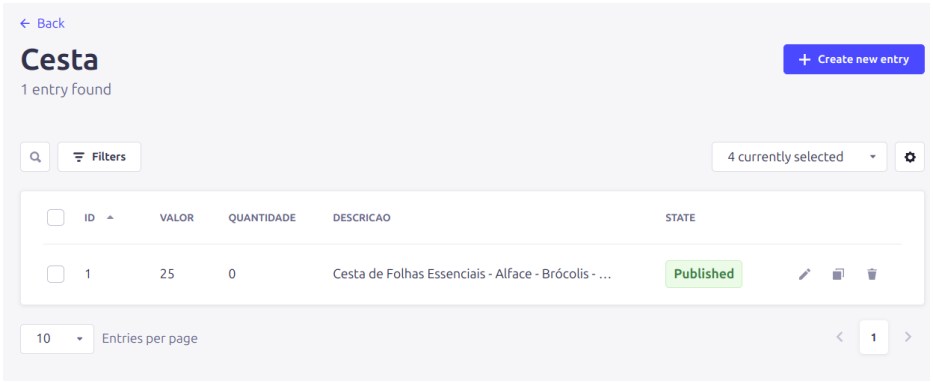

Figura 25 – Tela de cestas

8.1.3 Gerenciar Planos

Na tela de gerenciamento de planos indicada na Figura 26, é possível ver os planos já cadastrados pela sua CSA, também é possível criar um novo plano, editar um plano já existente e deletar o plano.

8.1.4 Gerenciar Produtos Avulso

Na tela de gerenciamento de produtos indicada na Figura 27, é possível ver os produtos avulsos já cadastrados pela sua CSA, também é possível criar novos produtos avulsos, editar um produto já existente e deletar um produto.

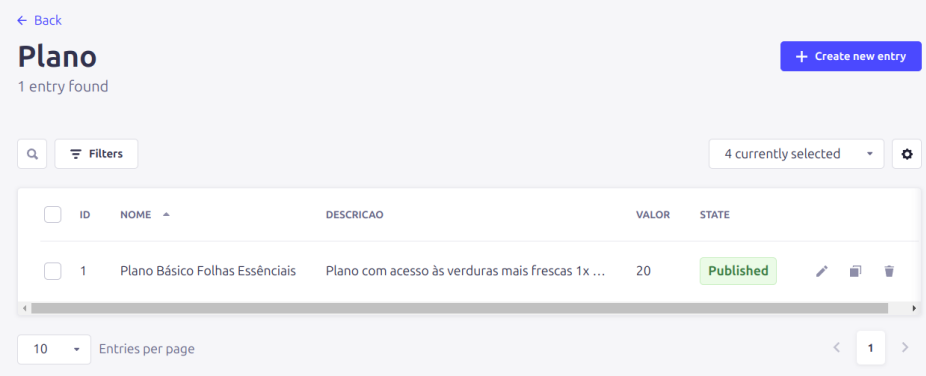

Figura 26 – Tela de planos

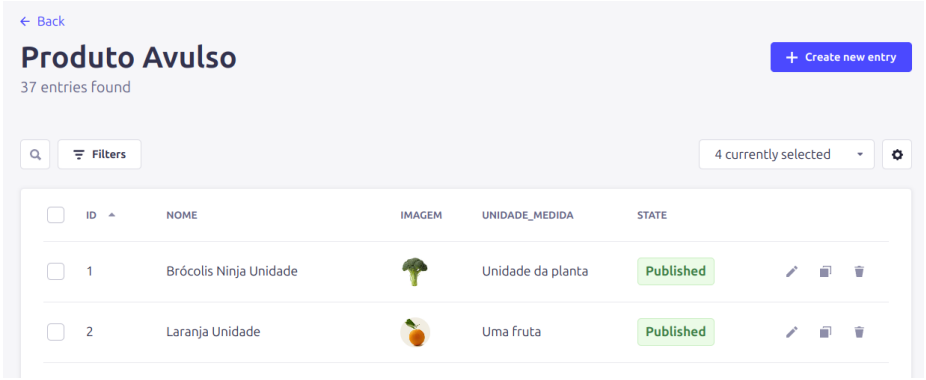

Figura 27 – Tela de produtos avulsos

8.1.5 Gerenciar Assinaturas

Na tela de gerenciamento de assinantes indicada na Figura 28, é possível visualizar quais são os assinantes de cada plano da CSA, além de possibilitar ao administrador saber quais assinantes desejam pular o recebimento da cesta semanal através do campo "pular_cesta".

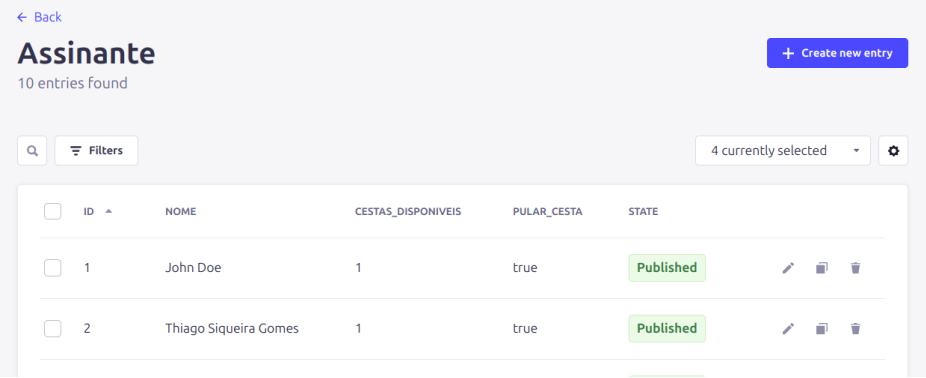

Figura 28 – Tela de assinaturas

8.1.6 Gerenciar Pedidos

Na tela de gerenciamento de extrato mostrada na Figura 29, é possível verificar os pedidos realizados por cada usuário, além disso, nesta tela também é possível marcar cada pedido com dois status diferentes, o status "pagamento_realizado" e o status "entregue", para que o administrador da CSA tenha controle da situação de cada pedido.

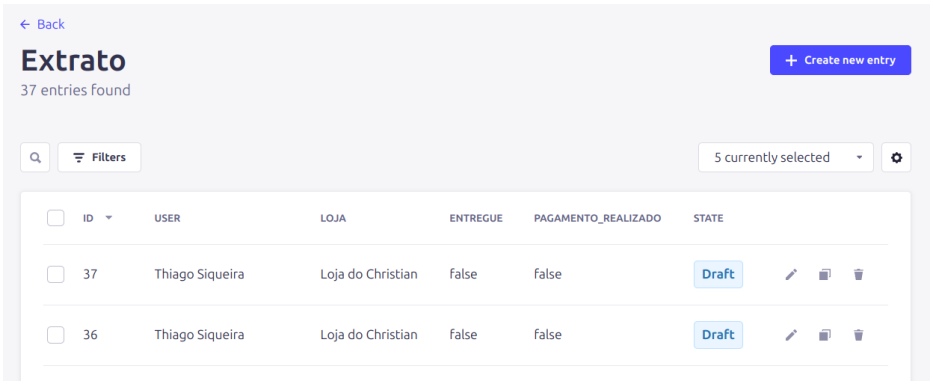

Figura 29 – Tela de pedidos

8.1.7 Gerenciar Usuários

Na tela de gerenciamento de usuários mostrada na Figura 30, é possível ver informações de um usuário, nesta tela também é possível criar, excluir e editar um usuário caso necessário.

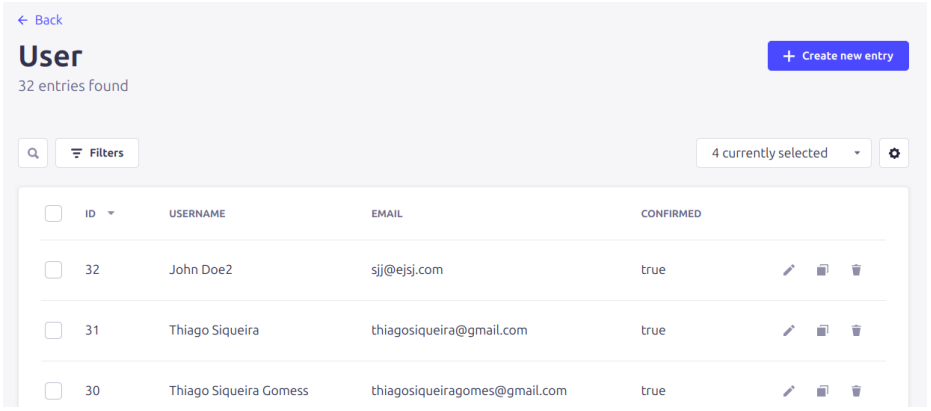

Figura 30 – Tela de usuários

8.1.8 Gerenciar Endereços

Na tela de gerenciamento de endereços mostrada na Figura 31, é possível verificar o endereço de um usuário, nesta tela também é possível criar, excluir e editar o endereço de um usuário caso necessário.

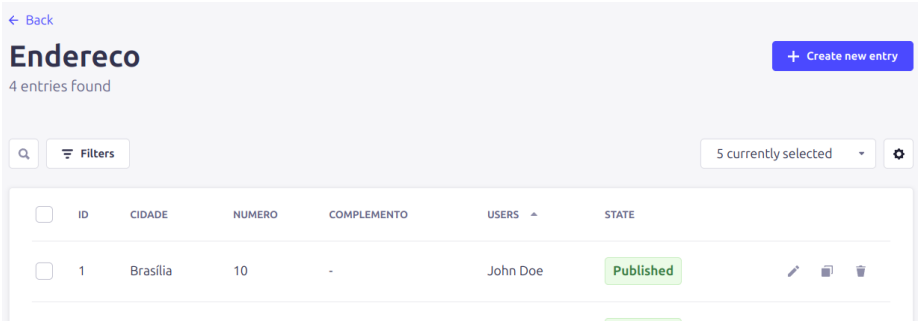

Figura 31 – Tela de endereço

8.2 Funcionalidades do Aplicativo

Após todas as correções realizadas para garantir o funcionamento do MVP do aplicativo móvel do Agromart, as seguintes funcionalidades estão disponíveis para uso:

- Se conectar a uma CSA
- Visualização de todas as lojas de uma CSA na página principal
- Pesquisar lojas por nome
- Pesquisar lojas por região administrativa
- Visualização de loja com produtos, planos e cestas
- Link para contato com o dono da loja
- Realizar pedidos
- Visualizar histórico de pedidos
- Visualizar planos assinados e pular cesta da semana
- Cadastrar e editar endereço
- Editar perfil

8.2.1 Se Conectar a uma CSA

Na tela inicial mostrada na Figura 32, utilizando o botão "Escolha sua CSA", o usuário será redirecionado para uma tela em que é possível digitar o código de uma CSA e pesquisar por ela, como mostrado na Figura 33.

Ao encontrar a CSA desejada, é possível se conectar a essa CSA utilizando o botão "Utilizar esta CSA", como se pode ver na Figura 34. Após se conectar à uma CSA, o usuário será redirecionado para tela de login e registro.

Figura 32 – Tela Inicial

8.2.2 Criar Conta na CSA

Após escolher a CSA desejada, caso ainda não tenha um cadastro de usuário nesta CSA, é possível realizar o cadastro do seu perfil utilizando: nome, email e senha.

Na tela de criação de conta, mostrada na Figura 35, é possível cadastrar seu perfil na CSA escolhida. O cadastro é realizado dentro do servidor de cada CSA individualmente, então as credenciais cadastradas poderão ser usadas apenas para realizar o acesso à CSA cadastrada.

8.2.3 Login na CSA

Após escolher a CSA desejada, caso já tenha um cadastro de usuário nesta CSA, é possível realizar o login no seu perfil utilizando email e senha, como mostra a Figura 36.

Nesta tela de Login também é possível conferir o nome e o domínio da CSA em que está se realizando login.

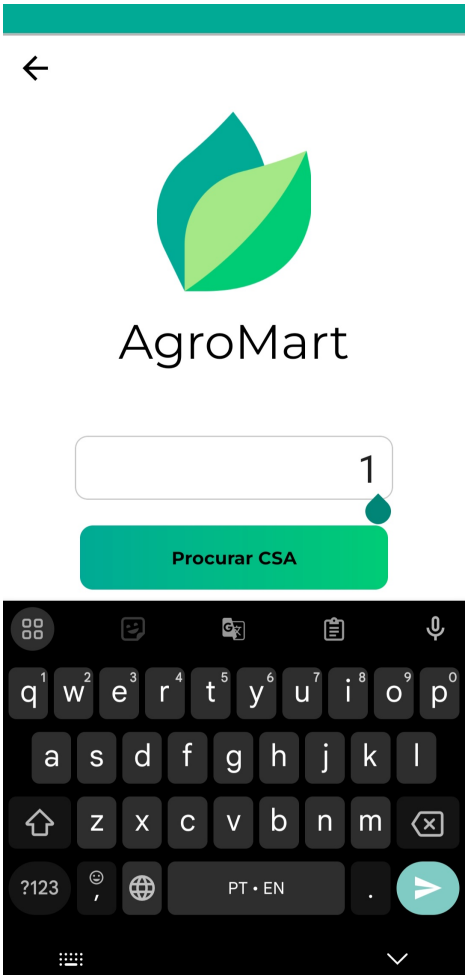

Figura 33 – Tela de Busca de CSA

8.2.4 Visualizar Lojas na Página Principal

Uma CSA pode ter várias lojas diferentes. Na página principal, mostrada na Figura 37, é possível visualizar uma lista das lojas que esta CSA possui.

8.2.5 Pesquisar Loja por Nome

Clicando no ícone em formato de lupa e digitando no campo de pesquisa, é possível pesquisar uma loja pelo nome ou por parte do nome, como mostra a Figura 38.

8.2.6 Pesquisar Loja por Região Administrativa

Na tela mostrada na Figura 39, é possível visualizar todas as regiões administrativas do Distrito Federal, clicando na região administrativa desejada é possível ver uma lista de todas as lojas disponíveis nesta região.

Figura 34 – Tela de Escolha de CSA

8.2.7 Adicionar e Editar Endereço

Na tela de configurações existe a opção de endereço, como mostra a Figura 40. Ao clicar nesta opção, o Aplicativo irá redirecionar para a tela de cadastro e edição de endereço, onde é possível adicionar ou alterar o endereço vinculado ao seu perfil de usuário nesta CSA, como é possível ver na Figura 41.

8.2.8 Editar Perfil

Na tela de configurações mostrada na Figura 40, existe a opção de edição de perfil, que ao ser selecionada redireciona o usuário para uma tela onde é possível editar seu nome e email, como mostra a Figura 42.

8.2.9 Contatar Dono da Loja

Dentro da tela principal de uma loja, mostrada na Figura 43, é possível observar que no canto superior direito existe um ícone do WhatsApp. Clicando neste ícone, o

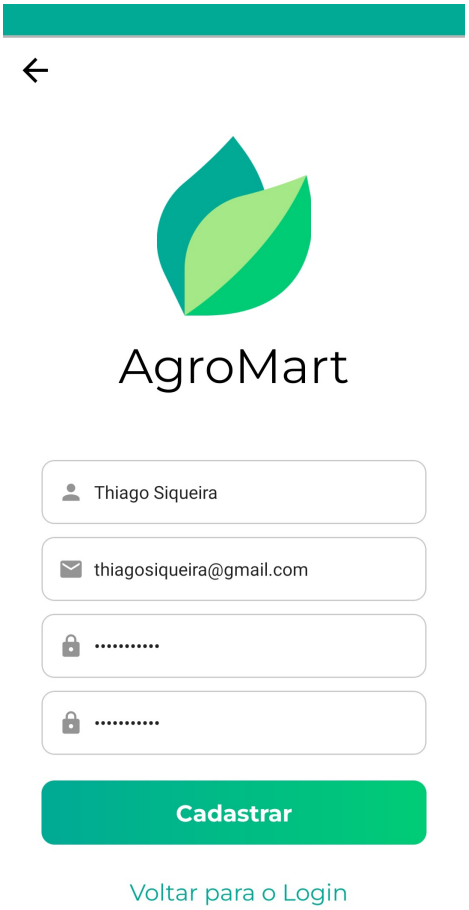

Figura 35 – Tela de cadastro de usuário

aplicativo irá redirecionar o usuário para o WhatsApp cadastrado pelo dono loja. Esta funcionalidade possibilita uma comunicação direta entre usuário e loja.

8.2.10 Visualizar Planos, Cestas e Produtos

Dentro da tela principal de uma loja, existe uma barra de navegação contendo três categorias: planos, cestas e produtos. Selecionando a categoria desejada, é possível visualizar os itens disponíveis, com as quantidades e preços de cada um deles, como mostra a Figura 43.

8.2.11 Realizar Compra

Ao clicar em um produto, plano ou cesta, é possível escolher a quantidade que deseja e adicionar ao carrinho, como mostra a Figura 44. Ao escolher todos os itens desejados, é possível ir até a tela do seu carrinho, como mostra a Figura 45, e finalizar sua compra. Após finalizar a compra seu pedido será enviado para a loja.

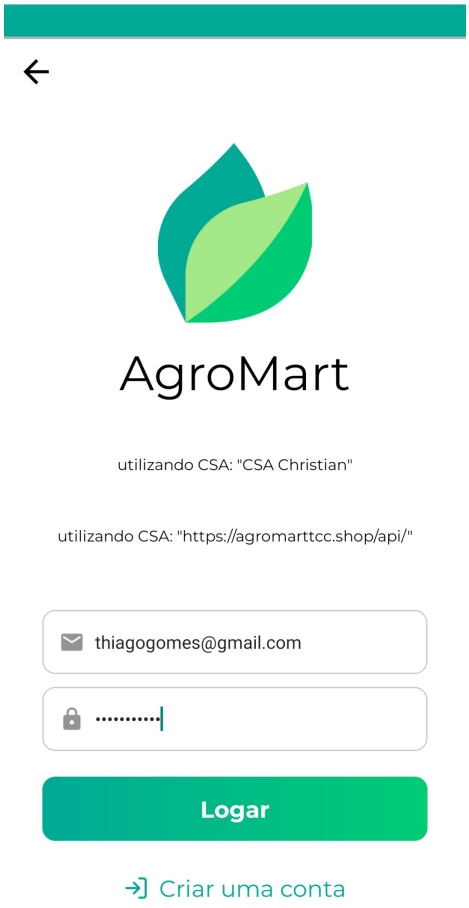

Figura 36 – Tela de login de usuário

8.2.12 Visualizar Histórico de Compras

Após a realização de uma compra, o usuário será redirecionado para uma tela de histórico, onde é possível ver todas as compras realizadas por ele até o presente momento, como é possível ver na Figura 46.

8.2.13 Visualizar Planos Assinados e Pular Cesta da Semana

Na tela de configurações, indicada na Figura 40, é possível selecionar a opção de visualizar os planos assinados. Ao acessar a tela de planos assinados, mostrada na Figura 47, é possível visualizar a lista de todos os planos assinados pelo usuário.

Além disso, nesta tela é possível utilizar o campo de "pular cesta semanal", que é um marcador para indicar ao dono da loja que nesta semana o usuário não deseja receber a cesta.

Figura 37 – Página Principal

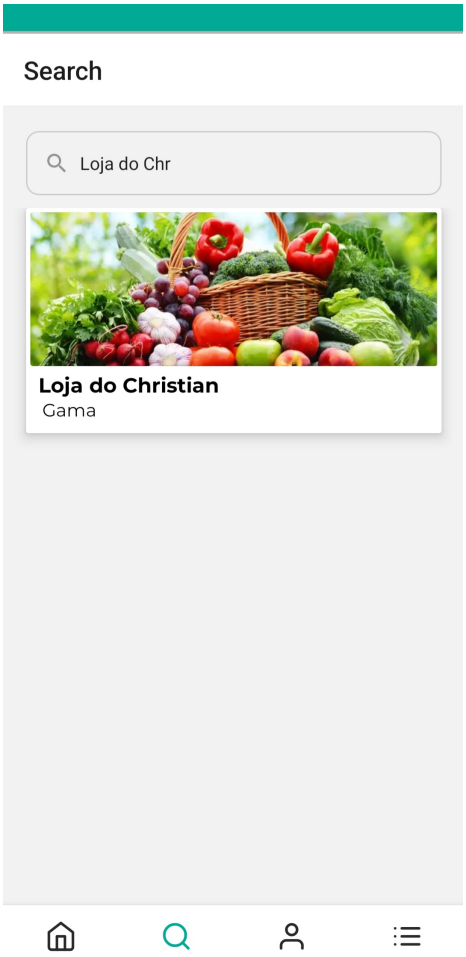

Figura 38 – Tela de pesquisa de loja por nome

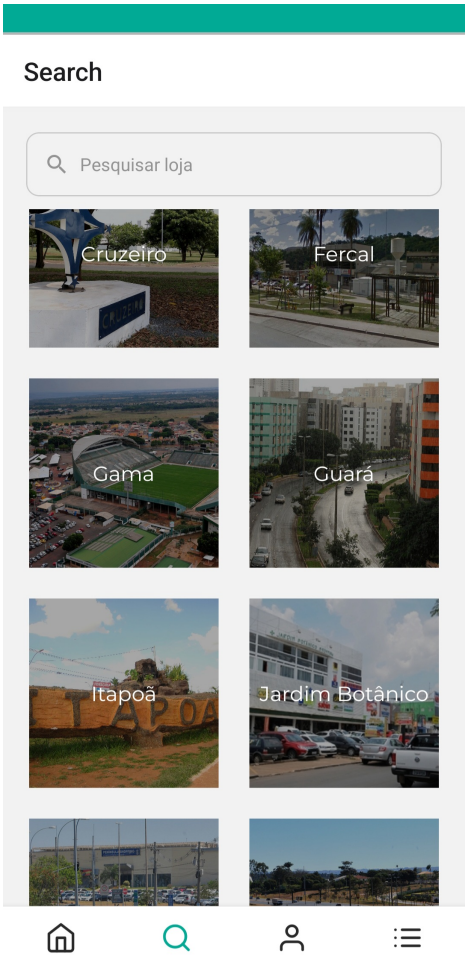

Figura 39 – Tela de pesquisa de lojas por região administrativa

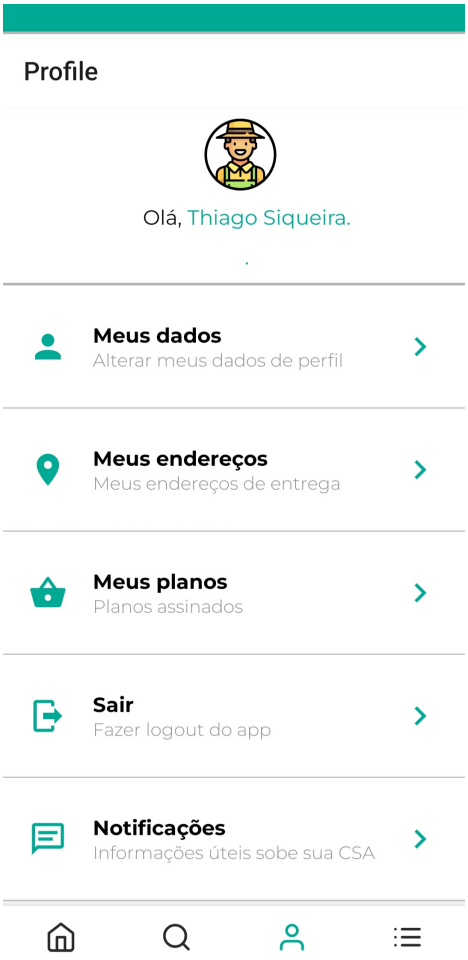

Figura 40 – Tela de configurações

Figura 41 – Tela de endereço

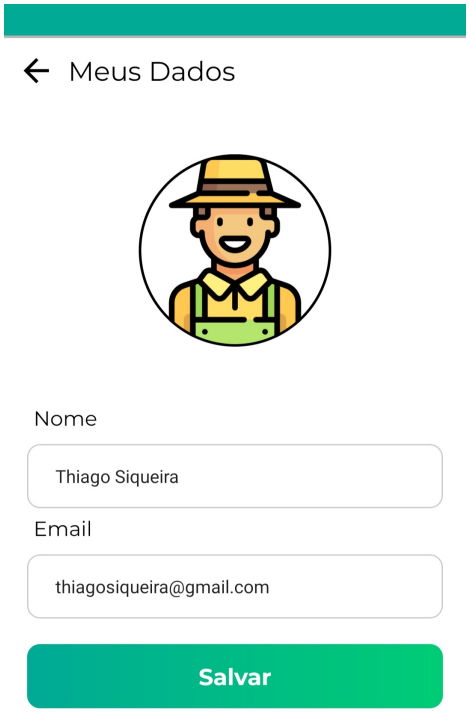

Figura 42 – Tela de perfil

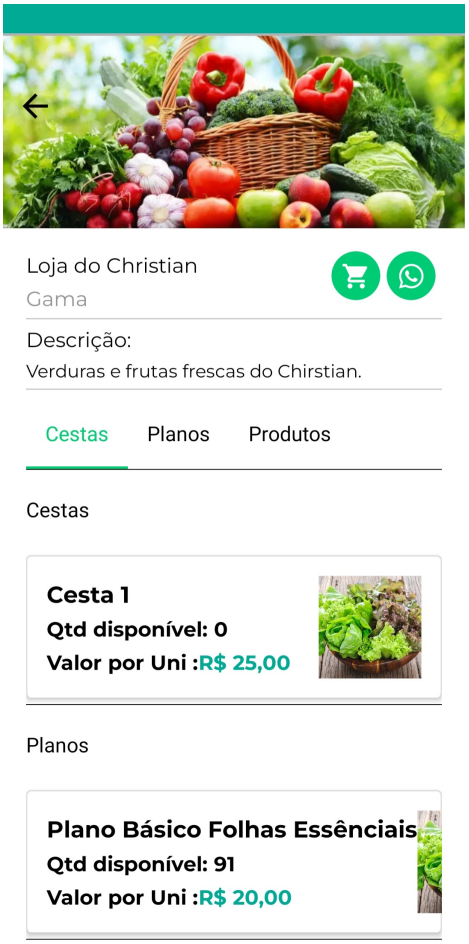

Figura 43 – Tela principal de uma loja

Figura 44 – Tela de adicionar produto

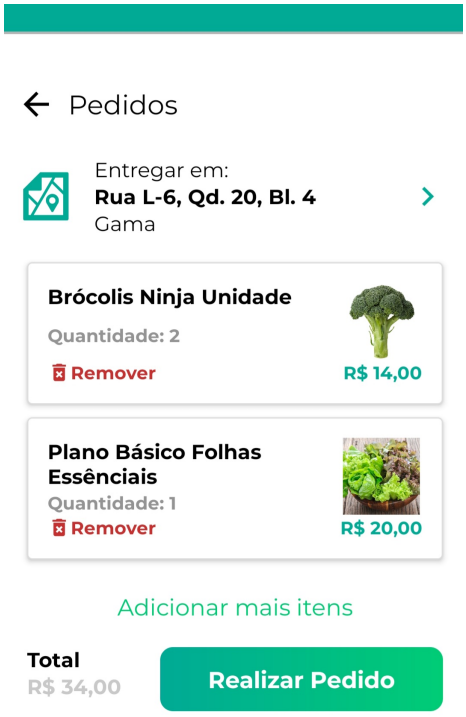

Figura 45 – Tela de finalizar compra

Figura 46 – Tela de histórico de compras

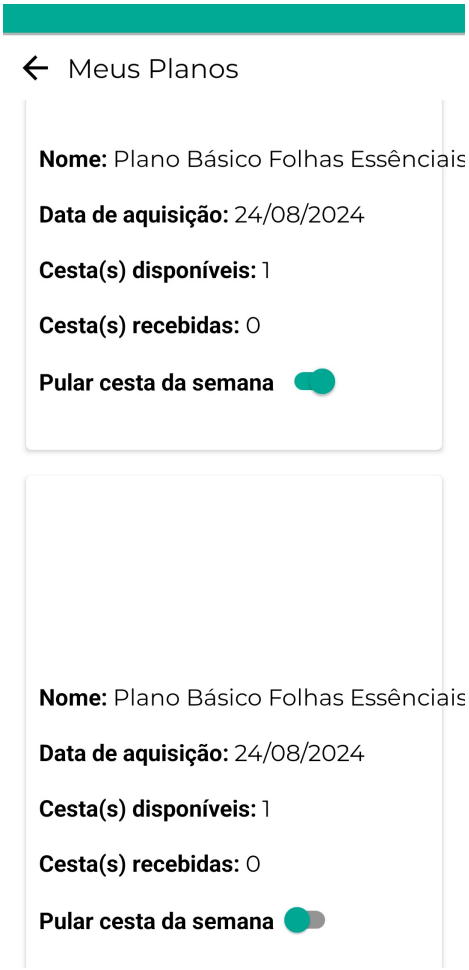

Figura 47 – Tela de planos assinados

9 Apresentação do produto

9.1 Visão Geral

Embora a apresentação do Agromart para uma CSA real não tenha sido um objetivo principal deste trabalho, tivemos a oportunidade de apresentá-lo à CSA da Florestta no dia 05/09/2024, na sede do Instituto Florestta em Taguatinga-DF, com a presença do professor e coorientador, Rudi Henri van Els. Após a apresentação, a atmosfera foi bastante positiva, e concluímos que o Agromart já possui as funcionalidades principais para ser utilizado por agricultores individuais. No entanto, constatou-se que o Agromart ainda não atingiu o nível de maturidade necessário para ser implementado em uma CSA real, com alguns pontos importantes levantados pela CSA da Florestta como essenciais para sua implantação.

Figura 48 – Foto na sede da CSA da Florestta após a apresentação

9.2 Melhorias identificadas pela CSA da Florestta

Essa seção tem como objetivo apresentar os pontos de melhoria destacados pelo administrador da CSA da Florestta para que o sistema do Agromart possa atender às necessidades de uma CSA real. Foram esses:

- **Substituir o termo loja por pontos de convivência**: Dentro do contexto de uma CSA, os espaços em que o agricultor e o co-agricultor interagem são chamados de pontos de convivência ou pontos de coleta, o termo loja é evitado nesse contexto para evitar um entendimento equivocado de como uma CSA funciona.
- **Vincular os produtos à CSA**: Atualmente os planos e produtos estão sendo cadastrados dentro uma loja (ponto de convivência), no entanto, conversando com

o administrador da CSA da Florestta, foi observado por ele que os planos e produtos devem estar vinculados à CSA, sendo cada ponto de convivência apenas o local onde o co-agricultor vai escolher receber os produtos.

- **Lembretes do recebimento das cestas**: Para o administrador da CSA, outro ponto muito importante que poderia ser incorporado ao Agromart, seria a funcionalidade de lembrete para o dia de entrega das cestas, que poderia ser tanto por meio de uma notificação no aplicativo quanto por uma mensagem no WhatsApp do co-agricultor.
- **Escolha do tipo de entrega**: No momento da finalização de um pedido, adicionar a funcionalidade de escolha do tipo de entrega, onde o co-agricultor informa se deseja realizar a retirada do produto ou recebê-lo em sua casa.
- **Flexibilizar periodicidade dos planos**: Atualmente os planos cadastrados no Agromart são sempre semanais, para atender as demandas de uma CSA real, é necessário possibilitar que a adesão aos planos tenha opções de diferentes periodicidades, como, por exemplo, o recebimento de cestas quinzenais.

Essas melhorias foram recebidas como sugestões importantes que podem guiar o desenvolvimento futuro do Agromart.

10 Considerações Finais

O desenvolvimento do Agromart, iniciado em 2020 durante um *Hackathon* na UnB-FGA, representa um esforço contínuo de alunos e professores em busca de soluções tecnológicas para a agricultura familiar. A proposta original do sistema evoluiu ao longo de cinco trabalhos de conclusão de curso, abrangendo aspectos como arquitetura, integração de pagamentos, hospedagem e funcionalidades específicas para as CSAs.

No entanto, ao longo desse processo, a falta de integração desses trabalhos, bem como a ausência de manutenções e de constantes de atualizações de ambientes no sistema, uma vez que os trabalhos de conclusão de curso são periódicos, comprometeu a integridade do projeto. As múltiplas *branches* nos repositórios, sem uma integração e testes efetivos, geraram incompatibilidades entre funcionalidades e novas arquiteturas prejudicando o funcionamento do sistema.

Diante desse cenário, os objetivos deste trabalho foram claros: analisar e unificar as *branches* existentes nos repositórios, realizar testes para identificar e corrigir inconsistências e todas as falhas que impediam o uso e publicação do MVP, efetuar a publicação desse MVP na Play Store e realizar mudanças na documentação. A busca pela unidade do projeto visa não apenas a viabilização da distribuição do aplicativo e do servidor, mas também facilitar o ambiente para receber contribuições externas, fortalecendo o caráter *open-source* do projeto.

A análise e correção de inconsistências é essencial para garantir o funcionamento adequado do Agromart. Repositórios coesos e integrados, para que realizar uma contribuição *open-source* se torne mais simples, um MVP, com o aplicativo disponível para download na Google Play Store e o servidor apto a ser utilizado por CSAs, representa a concretização dos esforços deste trabalho.

Durante o desenvolvimento do presente trabalho, diversas situações desafiadoras foram encontradas, dentre elas, pode-se destacar a etapa de atualização e troca de algumas bibliotecas utilizadas pelo sistema que estavam depreciadas, respeitando a linha tênue entre abandonar as versões antigas prejudiciais ao sistema, mas sem incorporar novas dependências com diferenças que demandassem alterações críticas no projeto que fossem maiores do que o escopo definido inicialmente. Além disso, mesclar diferentes *branches* de trabalhos desenvolvidos anteriormente em um projeto único, demandou um olhar analítico para entender cada um dos diferentes trabalhos realizados, para enfim realizar a síntese de todos eles em uma ramificação única.

Além disso, apesar de não estar nos planos iniciais, a apresentação do produto desenvolvido para CSA da Florestta foi um passo importante para o desenvolvimento

do Agromart, já que funcionalidades cruciais para o funcionamento pleno do sistema em uma implantação comercial foram identificadas. Dessa forma, o Agromart pode ser incrementado a fim de satisfazer esses requisitos futuramente e ser utilizado por CSAs reais. Apesar disso, a sensação de positividade é grande pelo fato do sistema já possuir as funcionalidades básicas para o uso de agricultores individuais.

Para trabalhos futuros, destaca-se a implementação das funcionalidades e melhorias sugeridas pelo administrador da CSA da Florestta durante a apresentação do sitema Agromart, que visam atender às necessidades de uma CSA real. Além disso, a reintegração do meio de pagamento, atualmente desativado por questões técnicas, é um ponto de evolução que abre um leque de possibilidades de futuras funcionalidades de gestão financeira da CSA dentro do Agromart.

Os ótimos resultados alcançados até agora são apenas o começo de um grande ciclo, em que o Agromart poderá se tornar uma ferramenta extremamente útil para a agricultura familiar e pode realmente fazer a diferença para cada agricultor e suas comunidades.

Referências

AGUSTINI F. B. S.; BOTTINO, G. B. Módulo de integração de pagamento e manutenção do software agromart. Brasília, Brasil, 2023. Citado na página 18.

ALURA. *Git Flow: entenda o que é, como e quando utilizar*. 2023. Disponível em: <https://www.alura.com.br/artigos/git-flow-o-que-e-como-quando-utilizar>. Acesso em: 18 dez. 2023. Citado na página 33.

BASUMALLICK, C. *What Is Android OS? History, Features, Versions, and Benefits*. 2024. Disponível em: <https://www.spiceworks.com/tech/tech-general/articles/android-os/>. Acesso em: 10 set. 2024. Citado na página 19.

BECK, C. A. B. K. *Extreme Programming Explained: Embrace Change*. 2th. ed. [S.l.]: Addison Wesley Professional, 2004. ISBN 0-321-27865-8. Citado na página 29.

BUDZIńSKI, M. *What Is React Native? Complex Guide for 2024*. 2024. Disponível em: <https://www.netguru.com/glossary/react-native>. Acesso em: 09 set. 2024. Citado na página 18.

CELLA V. S. C.; FREITAS, A. A. Uma evolução do projeto agromart: implantação individualizada e automatizada de um ambiente de csa. Brasília, Brazil, 2023. Citado 2 vezes nas páginas 18 e 20.

CORRÊA B. K. B.; VELUDO, I. G. Uma evolução do projeto agromart: open source, meios de pagamento e gestão de co-agricultores. Brasília, Brasil, 2022. Citado na página 18.

FERNANDES, D. *Expo: o que é, para que serve e quando utilizar?* 2018. Disponível em: <https://blog.rocketseat.com.br/expo-react-native/>. Acesso em: 10 set. 2024. Citado na página 18.

FOUNDATION, E. F. *About CertBot*. 2024. Disponível em: <https://certbot.eff.org/pages/about>. Acesso em: 11 set. 2024. Citado na página 55.

GADHAVI, M. *What is Strapi, and Why You Should Use it?* 2023. Disponível em: <https://radixweb.com/blog/what-is-strapi>. Acesso em: 10 set. 2024. Citado na página 19.

GIT. *About - Git*. 2024. Disponível em: <https://git-scm.com/about>. Acesso em: 09 set. 2024. Citado na página 32.

GITHUB, I. *About GitHub and Git*. 2024. Disponível em: <https://docs.github.com/en/get-started/start-your-journey/about-github-and-git>. Acesso em: 09 set. 2024. Citado na página 33.

GRIFFITHS, T. *What is JavaScript? Definition and How To Learn It*. 2024. Disponível em: <https://www.indeed.com/career-advice/career-development/what-is-javascript>. Acesso em: 10 set. 2024. Citado na página 19.

MYERS, G. J.; BADGETT, T.; SANDLER, C. *The Art of Software Testing*. 3rd. ed. USA: John Wiley & Sons, Inc, 2012. ISBN 978-1-118-13313-2. Citado na página 27.

NGINX. *nginx*. 2024. Disponível em: <https://nginx.org/en/>. Acesso em: 11 set. 2024. Citado na página 55.

OPENSOURCE. *The Open Source Definition*. 2023. Disponível em: <https://opensource.org/osd/>. Acesso em: 18 dez. 2023. Citado na página 28.

RIBEIRO A. F. C.; MAGALHÃES, R. F. T. Associação para aplicações agromart de uma csa em cloud. Brasília, Brasil, 2023. Citado na página 18.

RODRIGUES L. S.; MACEDO, L. P. d. A. novações tecnológicas na agricultura familiar: Agromart. Brasília, Brasil, 2021. Citado 2 vezes nas páginas 17 e 18.

SCOTT, P. *What is PostgreSQL? Everything You Need to Know*. 2024. Disponível em: <https://www.percona.com/blog/what-is-postgresql-used-for/>. Acesso em: 09 set. 2024. Citado na página 19.

SHUBEL, M. *What Is TypeScript?* 2022. Disponível em: <https://thenewstack.io/what-is-typescript/>. Acesso em: 09 set. 2024. Citado na página 18.

SILVA, R. C. *A real contribuição da agricultura familiar no Brasil*. 2024. Disponível em: <https://www.embrapa.br/agropensa/busca-de-noticias/-/noticia/27405640/a-real-contribuicao-da-agricultura-familiar-no-brasil>. Acesso em: 17 set. 2024. Citado na página 17.

SOMMERVILLE, I. *Software Engineering*. 9th. ed. USA: Addison-Wesley Publishing Company, 2007. ISBN 978-0-321-31379-9, 0-321-31379-8. Citado 4 vezes nas páginas 25, 26, 27 e 28.

SUTHERLAND, J. *Scrum : a arte de fazer o dobro do trabalho na metade do tempo*. São Paulo, Brazil: LeYa, 2014. ISBN 978-85-441-0088-2. Citado na página 29.