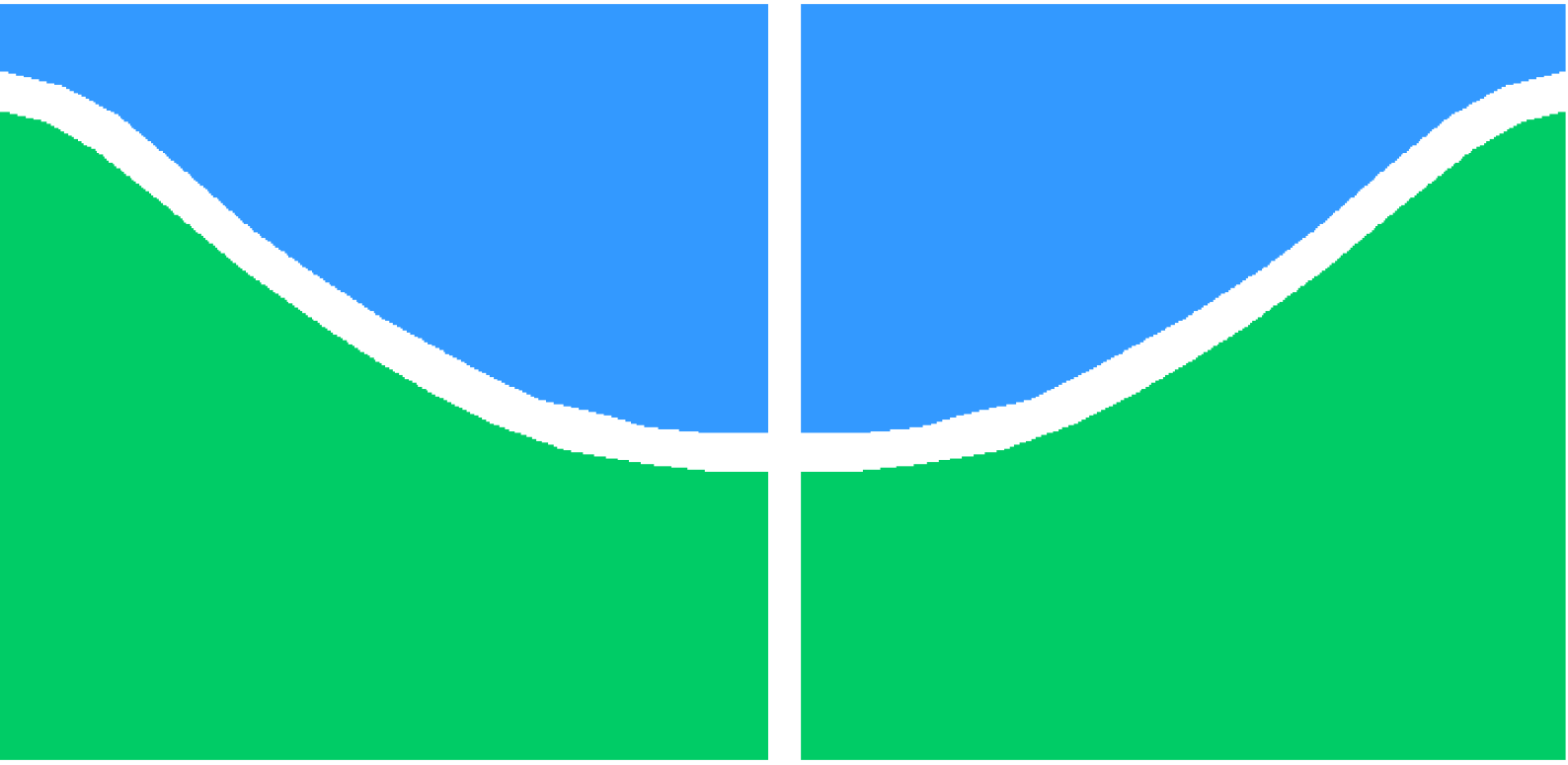

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Engenharia de Software

Inovações Tecnológicas na Agricultura Familiar: Agromart

Autor: Lucas Siqueira Rodrigues e Lucas Pereira de Andrade Macêdo

Orientador: Prof. Dr. André Luiz Peron Martins Lanna

Brasília, DF

2021

Lucas Siqueira Rodrigues e Lucas Pereira de Andrade Macêdo

Inovações Tecnológicas na Agricultura Familiar: Agromart

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: Prof. Dr. André Luiz Peron Martins Lanna

Brasília, DF

2021

Lucas Siqueira Rodrigues e Lucas Pereira de Andrade Macêdo
Inovações Tecnológicas na Agricultura Familiar: Agromart/ Lucas Siqueira Rodrigues e Lucas Pereira de Andrade Macêdo. – Brasília, DF, 2021-
59 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. André Luiz Peron Martins Lanna

Trabalho de Conclusão de Curso – Universidade de Brasília - UnB
Faculdade UnB Gama - FGA , 2021.

1. Agricultura Familiar. 2. Tecnologia. I. Prof. Dr. André Luiz Peron Martins Lanna . II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Inovações Tecnológicas na Agricultura Familiar: Agromart

CDU 02:141:005.6

Lucas Siqueira Rodrigues e Lucas Pereira de Andrade Macêdo

Inovações Tecnológicas na Agricultura Familiar: Agromart

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, 28 de maio de 2021:

Prof. Dr. André Luiz Peron Martins Lanna
Orientador

Prof. Dr. Rudi Henri Van Els
Convidado 1

Profa. Me. Cristiane Soares Ramos
Convidado 2

Prof. Me. Ricardo Ajax Dias Kosloski
Convidado 3

Brasília, DF
2021

Este trabalho é dedicado a todos os pequenos agricultores que foram impactados pela pandemia causada pela covid 19.

Agradecimentos

Eu, Lucas Siqueira Rodrigues, agradeço primeiramente aos meus pais, Gislene e Fernando, por nunca deixarem de me apoiar e incentivar os meus estudos. Agradeço meus familiares por sempre apoiarem minhas decisões e colaborarem com minhas conquistas. Agradeço aos professores pelo ensino recebido durante a faculdade, e por fim agradeço a todos os meus colegas de curso pelas experiências que compartilhamos juntos durante todo este período.

Eu, Lucas Pereira de Andrade Macêdo, agradeço principalmente aos meus pais Osmar e Ana Paula pelo apoio e incentivo, agradeço aos professores pelos conhecimentos compartilhados, agradeço aos colegas de curso por todas as experiencias, bons e maus momentos que vivenciamos durante essa jornada e por fim agradeço aos amigos e familiares por toda positividade transmitida por meio de seu apoio, incentivo e companheirismo.

Resumo

A agricultura familiar é responsável por grande parte dos alimentos consumidos no Brasil, ela é focada em produzir tipos variados de produtos orgânicos e artesanais. Este trabalho de conclusão de curso tem como objetivo desenvolver uma solução tecnológica denominada Agromart que foca facilitar a relação entre os pequenos agricultores e os consumidores. A solução engloba uma interface web para os agricultores e um aplicativo mobile multi-plataforma para os consumidores, com o propósito de proporcionar ao pequeno agricultor uma maior garantia para o escoamento de sua produção e ajudar pessoas que buscam uma alimentação mais saudável e de qualidade a encontrar seus produtos.

Palavras-chaves: Agricultura familiar. Tecnologia. React. React-Native.

Abstract

Family farming is responsible for a large part of the food consumed in Brazil, it is focused on producing varied types of organic and artisanal products. This course conclusion work has the objective to develop a technological solution called Agromart, which aims to mediate the relationship between small farmers and consumers, the solution includes a web interface for farmers and a multiplatform mobile application for consumers, with the purpose of providing the small farmer a greater guarantee for the flow of his production, and help people that are looking for healthier and better quality of his food to find their products.

Palavras-chaves: family farming. technology. React. React-Native.

Lista de ilustrações

Lista de abreviaturas e siglas

CSAs	Comunidades que Sustentam a Agricultura
TCC	Trabalho de Conclusão de Curso
UnB-FGA	Universidade de Brasília - Campus Gama
CMS	Content Management System
SGC	Sistema de Gerenciamento de Conteúdo
XP	eXtreme Programming
ER	Relacionamento de Entidade
API	Interface de Programação de Aplicações

Sumário

1 Introdução

Historicamente, os setores que abarcam a economia são classificados por três grandes eixos: primário, secundário e terciário. A agricultura faz parte do setor primário e compreende as atividades agrícolas, pecuárias e extrativas. De forma generalizada, costuma-se definir a agricultura como um conjunto de técnicas utilizada para cultivar plantas com o objetivo de obter alimentos, fibras, energia, matéria-prima para roupas, construções, medicamentos, ferramentas, dentre outros produtos (ROUDART., 2009).

Segundo a Organização das Nações Unidas para Alimentação e Agricultura (FAO), a agricultura familiar pode ser definida como um modo de produção agrícola, silvicultura, pesca, pecuária ou aquicultura; que administra e opera para uma família e que depende predominantemente de trabalho familiar incluindo homens e mulheres. Importante ressaltar que a agricultura familiar possui características que representam possibilidade de transição de um modelo de agricultura convencional, pautado no excessivo uso dos recursos naturais não-renováveis, para um sistema de produção agroecológico que tem como base os pilares da sustentabilidade (ecológica, econômica, social, cultural, espacial/geográfica).

Com o advento da pandemia de COVID-19 que se instalou no mundo e no Brasil os agricultores têm sido fortemente impactados no escoamento de sua produção, principalmente aqueles que não têm um sistema de escoamento independente e definido e dependem de feiras ou semelhantes para a distribuição da produção.

Um sistema de escoamento interessante é o das Comunidades que Sustentam a Agricultura (CSAs). Para um breve entendimento as CSAs são comunidades formadas por agricultores de produtos orgânicos ou agroecológicos e *coagricultores* (consumidores) que investem mensalmente em um sistema sustentável de produção. Esse modelo traz segurança e garantia de escoamento para o agricultor e impede que ele sofra com pressões de mercado, tudo isso em troca de alimentos saudáveis e livres de agrotóxicos. Esse sistema promove o que é conhecido como um encurtamento da cadeia já que o contato entre agricultor e consumidor é direto.

1.1 Justificativa

Grande parte dos agricultores familiares não possuem um sistema de produção bem definido como as CSAs de modo que alternativas voltadas para a inovação tecnológica têm sido essenciais para a continuidade de suas atividades comerciais. Algumas iniciativas nesse sentido já vêm sendo implementadas como, por exemplo, a colaboração de mapas virtuais de feiras orgânicas e páginas em redes sociais para divulgação. Porém

essas soluções não trazem ao pequeno agricultor nenhuma garantia para um melhor planejamento da sua produção além de não possibilitar uma comunicação efetiva entre o consumidor e o agricultor.

O Agromart, a solução de software deste trabalho, tem o propósito de proporcionar ao pequeno agricultor uma maior garantia para o escoamento de sua produção e ajudar pessoas que buscam uma alimentação mais saudável e de qualidade a encontrar seus produtos.

1.1.1 A História do Agromart

A ideia do Agromart surgiu inicialmente quando os autores deste trabalho se inscreveram no hackathon da UnB-FGA 2020, no tema: Cultivando Conexões, onde o desafio consistia no desenvolvimento de um software no contexto da agricultura familiar cujo o principal objetivo era estabelecer uma conexão entre os agricultores e os consumidores levando em consideração o isolamento social por conta da COVID-19 e a dificuldade dos produtores rurais no DF e entorno de conseguir listar e divulgar os seus produtos para seus clientes.

A ideia inicial surgiu a partir de uma reportagem transmitida no globo rural, onde Eleuza Fernandes, uma pequena agricultora de Goiás, montou uma barraca em que toda negociação era feita na base da confiança, onde o consumidor ao chegar recolhe o produto desejado e deixa o pagamento no local evitando desperdícios e proporcionando uma experiência compatível com as recomendações de distanciamento devido ao COVID-19, trazendo benefícios tanto para o agricultor quanto para o consumidor.

Essa história deu a motivação necessária para que fosse dado o primeiro passo do Agromart. Durante o evento, foi desenvolvido apenas um aplicativo onde o agricultor pudesse divulgar sua loja, barraca ou ponto de venda com seus devidos produtos, preços, localização, informação de contato e descrições adicionais. E com isso o consumidor poderia visualizar as lojas mais próximas dele através de mapas e filtros, entrar em contato com o agricultor por um link para iniciar um conversa direta por um aplicativo de mensagem, com o principal objetivo de confirmar a disponibilidade de produtos e adquirir informações sobre pagamentos. O usuário também poderia encontrar um ponto de venda através da localização e traçar uma rota a partir de seu destino atual, além disso, no aplicativo foi disponibilizado uma sessão de informações com recomendações quanto ao uso do mesmo durante a pandemia.

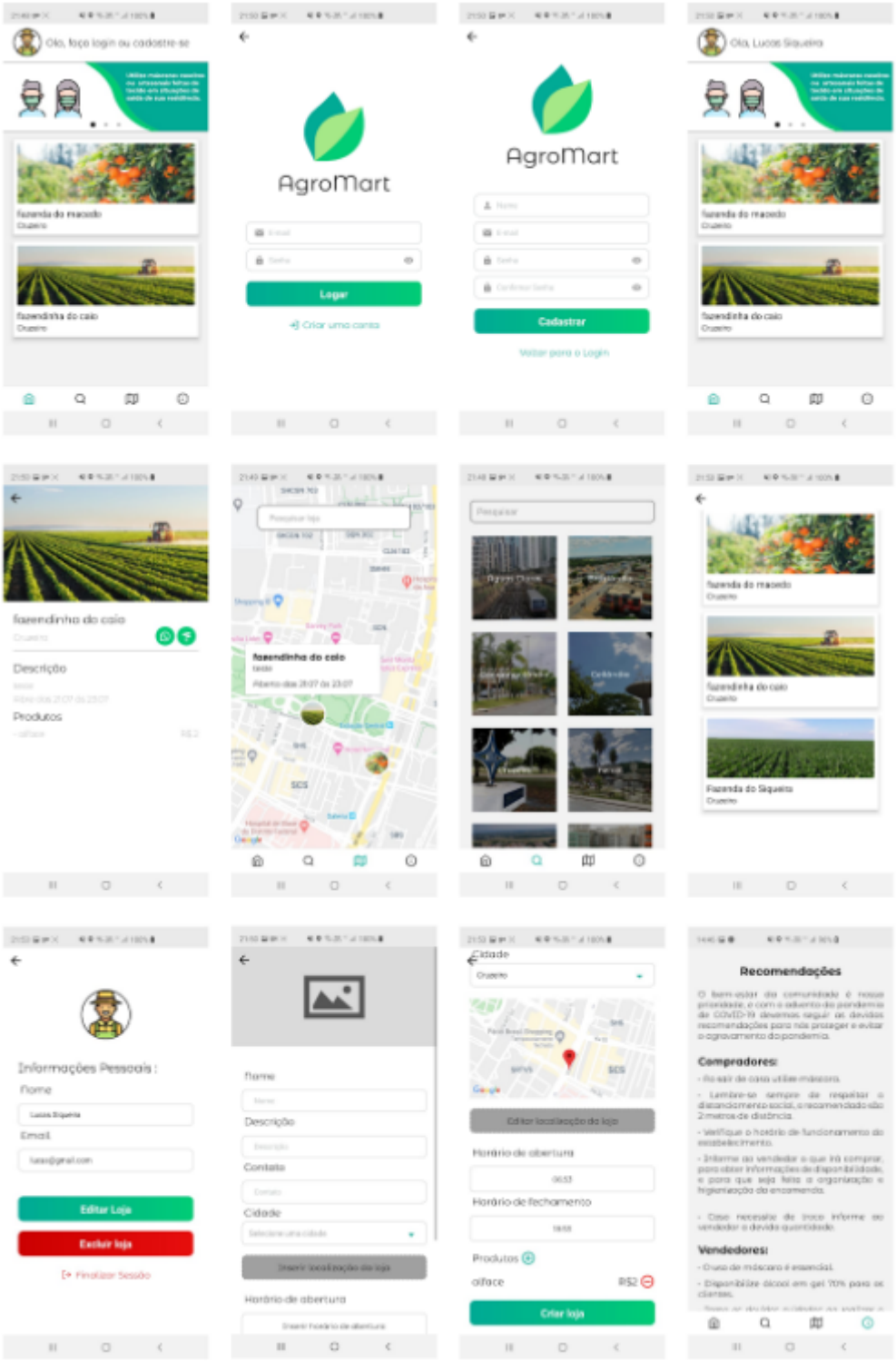

Figura 1 – Telas solução hackathon

Fonte: Autor

A solução apresentada para o hackathon conquistou o primeiro lugar da competição, e após o evento tivemos contato com profissionais da área que apresentaram abordagens diferentes da implementada. Por meio de entrevistas e conversas, conhecemos as regras de negócio das CSA's, com isso consideramos que a abordagem que utilizam com

a venda de cestas agroecológicas a partir da assinatura de planos trás para o agricultor uma maior garantia de escoamento dos seus produtos.

Então este trabalho propõe a adaptação do software para se adequar aos novos requisitos levantados, que são um aprimoramento mais flexível da abordagem utilizada pelas CSA'S. Para que essa adequação seja feita da melhor maneira, decidimos iniciar o projeto do zero, porém aproveitando alguns elementos do design anterior, acrescentando uma interface web para o gerenciamento por parte dos agricultores e mantendo o aplicativo mobile apenas para os consumidores.

1.2 Objetivos

1.2.1 Objetivo Geral

O objetivo deste trabalho é desenvolver uma solução em software que tenha em vista a resolução do problema existente na relação dos agricultores e consumidores, o qual foi acentuado com a pandemia. Tal solução deverá ser capaz de realizar a intermediação dessa comunicação, englobando um aplicativo de celular para os consumidores em que seria possível fazer a visualização e solicitação dos produtos agroecológicos. Para essa solicitação utiliza-se como base uma regra de negócio semelhante à das CSAs em que são feitas assinaturas de planos periódicos para a aquisição de cestas com produtos já definidos pelo agricultores, mas ainda sendo possível adicionar novos produtos nas cestas diante da disponibilidade dos mesmos. Pelo lado dos agricultores a solução a ser apresentada deve permitir o gerenciamento dessas cestas em que podem ser definidos:

- Possibilidade de adicionar novos produtos;
- Quantidades de cestas disponíveis;
- Quando e como será feita a coleta ou entrega das cestas e,
- Quais planos de assinatura serão disponibilizados para os seus consumidores;

1.2.2 Objetivos Específicos

- Compreender as necessidades e o contexto dos usuários para implementar interfaces adequadas, tendo como consequência uma boa usabilidade e fácil aprendizibilidade;
- Utilizar *frameworks* de desenvolvimento híbrido para a aplicação mobile, o que possibilita a geração do aplicativo para as plataformas *iOS* e *Android*;
- Implementar e documentar a solução proposta;
- Implantar a solução em um ambiente real;

1.3 Metodologia de Pesquisa

A metodologia de pesquisa deste trabalho é exploratória. Estas pesquisas têm como objetivo proporcionar maior familiaridade com o problema com vistas a torná-lo mais explícito ou a constituir hipóteses. Pode-se dizer que estas pesquisas têm como objetivo principal o aprimoramento de ideias ou a descoberta de intuições (GIL, 2002). Na maioria dos casos, essas pesquisas envolvem (a) levantamento bibliográfico, (b) entrevistas com pessoas que tiveram experiências práticas com o problema pesquisado e (c) análise de exemplos que "estimulem a compreensão"(GIL, 2002). Embora a pesquisa exploratória seja bastante flexível na maioria dos casos ela assume a forma de pesquisa bibliográfica ou de estudo de caso (GIL, 2002).

1.4 Fases do Trabalho

No curso de Engenharia de Software da UnB-FGA o TCC é dividido em duas etapas. A primeira é o TCC 1 em que o problema é analisado e uma solução é proposta, levando em conta a definição do escopo do trabalho, a metodologia adotada, a análise de referencial teórico sobre os conceitos que fundamentam o trabalho e a escolha da tecnologia para o desenvolvimento da solução. Na segunda etapa, o TCC 2, é a fase de desenvolvimento da solução (validada e na etapa anterior e incorporando alterações sugeridas quando for o caso) o software será desenvolvido segundo o planejamento realizado e utilizando as tecnologias e práticas definidas.

1.5 Organização do Trabalho

O trabalho será dividido em seis capítulos, sendo este inicial introdutório, descrevendo o contexto, a descrição do problema, os objetivos do trabalho e sua organização.

- Capítulo 2 - Referencial Teórico: Neste capítulo são apresentados os conceitos teóricos em que o trabalho será fundamentado com informações expostas por meio da pesquisa bibliográfica realizada.
- Capítulo 3 - Metodologia: Neste capítulo são descritos os procedimentos e práticas adotadas para realização deste trabalho.
- Capítulo 4 - Suporte Tecnológico: Neste capítulo são apresentados detalhes mais técnicos acerca da solução, evidenciando as ferramentas a serem utilizadas tanto na implementação do software quanto gestão da equipe e documentação da aplicação.
- Capítulo 5 - Resultados: Neste capítulo são apresentadas as funcionalidade do Agromart na versão mais recente.

- Capítulo 6 - Considerações Finais: Neste capítulo são apresentadas as considerações sobre o estudo e os trabalhos futuros a serem realizadas.

2 Referencial Teórico

2.1 Engenharia de Software

Engenharia de Software é o estabelecimento e uso de sólidos princípios de engenharia para que se possa obter economicamente um software que seja confiável e que funcione eficientemente em máquinas reais (BAUER, 1972) através da aplicação sistemática de conhecimentos científicos, tecnológicos, métodos e experiência para o projeto, implementação, teste e documentação de software (ISO/IEC/IEEE, 2010). Na visão de (PRESSMAN, 2011) a camada de processos é a base da Engenharia de Software pois é ela quem mantém as camadas tecnológicas coesas e possibilita o desenvolvimento de software de forma racional e dentro do prazo. É ainda na camada de processos que se define a metodologia que deve ser estabelecida para a entrega efetiva da tecnologia e que se constitui a base para o controle do gerenciamento de projetos de software, estabelecendo o contexto no qual são aplicados métodos técnicos, garantindo a qualidade e que as mudanças serão geridas de forma apropriada.

2.2 Aplicações mobile

Com os *smartphones* sendo o principal dispositivo de mão para mais de três bilhões de usuários (LTD, 2017) os aplicativos móveis já se tornaram uma necessidade tanto nos campos técnico quanto comercial. No entanto, existem muitas diferenças no desenvolvimento de aplicações mobile em relação ao desenvolvimento de aplicações tradicionais *desktop* ou *web* decorrentes, principalmente, devido às diversas limitações de hardware (capacidade de processamento, armazenamento, ambiente de utilização, tamanho da tela etc) e também características específicas como sensores, câmera, bateria (RAHIMIAN V.; RAMSIN, 2008), dentre outros.

O desenvolvimento de aplicações mobile pode ser dividido em desenvolvimento nativo e desenvolvimento híbrido. Aplicações nativas são aquelas específicas para determinado sistema operacional. Para esse tipo de aplicação os sistemas operacionais móveis desejam aplicativos específicos seus próprios ambientes de modo a aproveitar ao máximo seus recursos particulares, além de utilizar linguagens e ferramentas específicas para cada plataforma no desenvolvimento da aplicação (SERRANO N.; HERNANTES, 2013). Já aplicativos híbridos são desenvolvidos utilizando ferramentas de desenvolvimento web em conjunto com elementos nativos (SERRANO N.; HERNANTES, 2013). Esses aplicativos podem ser criados com o uso de *HTML, CSS e JavaScript* e ainda podem utilizar características e sensores nativos do aparelho (CHARLAND A ;LEROUX, 2011).

2.3 Aplicações Web

A aplicação web diz respeito a uma solução que é executada diretamente no *browser* (ou navegador) não sendo preciso realizar uma instalação na máquina do usuário. Pode-se, também, utilizar como definição "tudo aquilo que é processado em um servidor terceiro". As plataformas de *e-commerce* e as redes sociais são alguns dos exemplos que se enquadram nesse perfil. Para que uma aplicação web funcione ela depende de um servidor web, de solicitações realizadas pelos usuários, do uso de protocolos e métodos (normalmente o HTTP) e da resposta do protocolo. A aplicação deve permitir que os usuários consigam fazer uma solicitação e receber algo como resposta, ou seja, elas precisam mediar essa interação de forma natural, devolvendo o que a pessoa deseja como resultado (NOLETO, 2020).

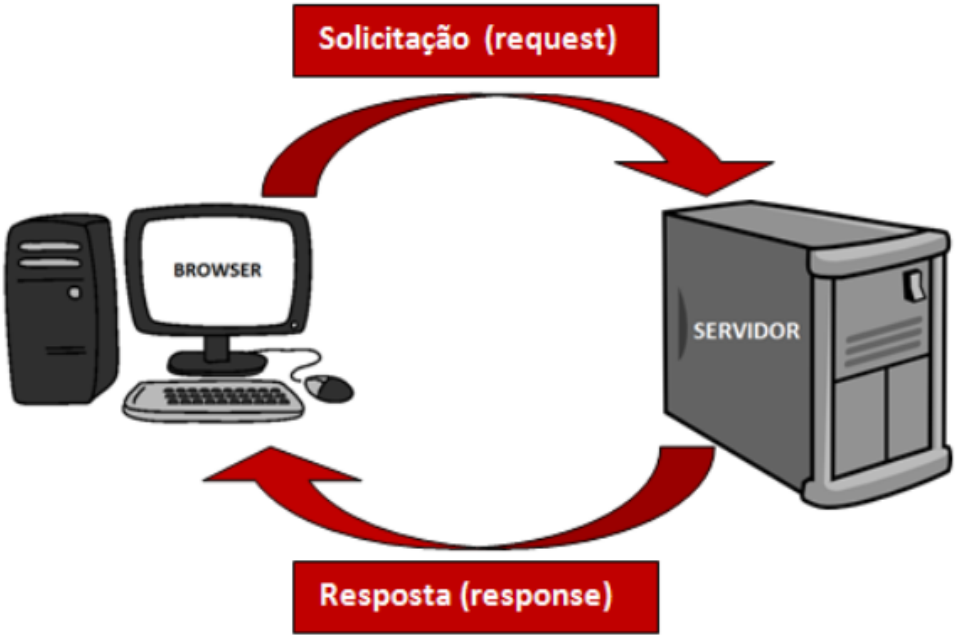

Figura 2 – Simulação de uma solicitação e resposta em aplicações web.

Fonte: (PALMEIRA;, 2012)

2.4 Sistemas de Gestão de Contéudo

O *Content Management System* (CMS - em português, Sistema de Gerenciamento de Conteúdo - SGC) surgiu no final da década de 1990 para melhorar a gestão do conteúdo dos *websites* das organizações. Desde então diversos SGC surgiram, muitos deles de código livre e disponíveis na Web para cópia e uso gratuito. Um SGC possibilita a criação, o gerenciamento, a distribuição, a publicação e a recuperação de informações corporativas, sendo que o gerenciamento do conteúdo deve ser definido sob o ponto de vista das atividades das pessoas e dos seus objetivos.(SILVA, 2008)

2.5 Desenvolvimento Ágil de Software

O desenvolvimento ágil de software é uma abordagem que surgiu formalmente após a publicação do Manifesto Ágil por um grupo de especialistas da área. Os métodos ágeis focam os esforços na entrega contínua de software em curtos períodos de tempo, removendo a necessidade de processos pesados e vasta documentação. Além disso, tem como uma das principais características a alta capacidade de lidar com mudanças devido ao seu princípio de "responder à mudanças em vez de seguir um plano estritamente". Como esta capacidade pode ser ativada através de um desenvolvimento iterativo e incremental. Emergiram novas práticas ágeis como *Extreme Programming - XP*, *Scrum* e *KanBan*. Esses métodos ágeis abrangem práticas para aumentar a capacidade de entrega no desenvolvimento de software (JENTSCH, 2017)

2.5.1 Scrum

O Scrum é um *framework* dentro do qual pessoas podem tratar e resolver problemas complexos e adaptativos, enquanto produtiva e criativamente entregam produtos com o mais alto valor possível. Tendo como características o fato de ser leve, simples de entender e extremamente difícil de dominar, ele é um *framework* estrutural que está sendo usado para gerenciar o desenvolvimento de produtos complexos desde o início de 1990. Scrum não é um processo ou uma técnica para construir produtos, em vez disso, é um *framework* dentro do qual você pode empregar vários processos ou técnicas.(SCHWABER, 2013)

Nesse *framework* os projetos são divididos em ciclos chamados de *sprints*, as funcionalidades a serem implementadas em um projeto são mantidas em uma lista chamada de *product backlog*. No início de cada *sprint* é feita uma *sprint planning meeting*, que é uma reunião de planejamento em que os itens do *product backlog* são priorizados e a equipe define as atividades que serão implementadas durante aquele *sprint*. As tarefas alocadas em uma *sprint* são transferidas do *product backlog* para o *sprint backlog*, que é a lista de tarefas da *sprint* em andamento. Outro ritual do *Scrum* são as *dailys*, que são reuniões curtas para alinhar o que já foi concluído de um dia para o outro. Ao fim de uma *Sprint*, a conclusão das tarefas é apresentada na *Sprint Review Meeting*, e após faz-se a retrospectiva da *Sprint*, e então a equipe parte para o planejamento da próxima *Sprint*. (ÁGIL, 2020)

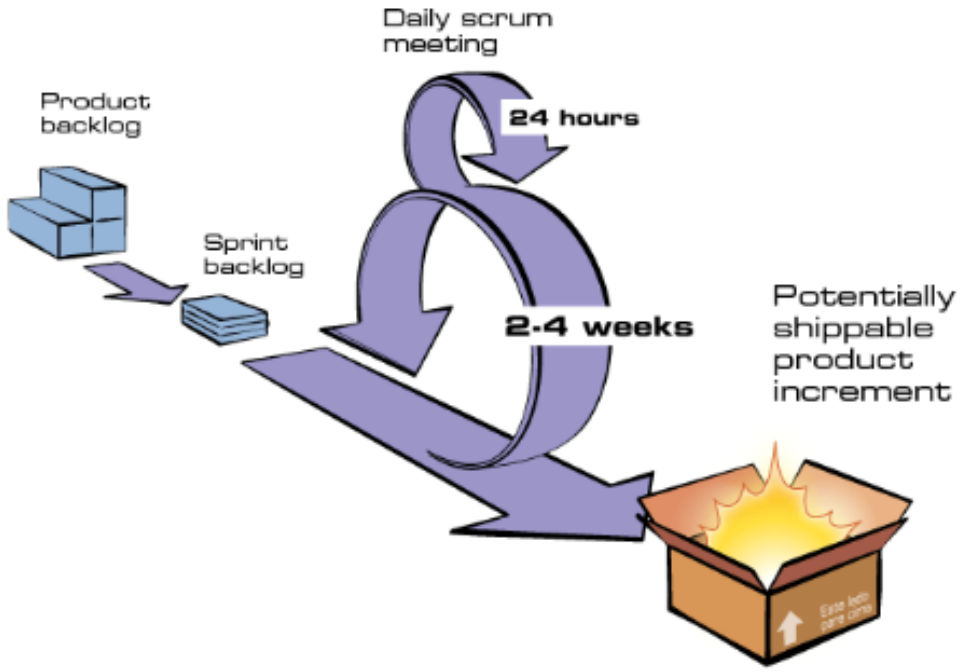

Figura 3 – Ciclo Scrum

Fonte: (ÁGIL, 2020)

2.5.2 Kanban

O método *Kanban* é um sistema utiliza cartões de cores diferentes ou tamanhos diferentes para designar e especificar tarefas. Dessa forma, se aprimora a administração a partir de cartões de sinalização para controle de fluxos. Assim, se sabe quais tarefas precisam ser feitas, estão sendo feitas e as que foram concluídas.(EGESTOR, 2020)

2.6 Entrega Contínua

Dentro do ciclo de desenvolvimento de um produto de software, existe a fase de entrega que consiste em disponibilizar a aplicação para uso, seja para usuários ou outros sistemas. Esta fase se resume ao conjunto de práticas, como integração e *deploy* contínuos que possibilitam que novos incrementos de software possam ser integrados ao código já existente. (HUMBLE J.; FARLEY, 2017)

A integração contínua é realizada através de compilações de software de vários tipos. As compilações de software típicas incluem a compilação do código-fonte para obter o programa ou componentes executáveis, compilar o código de teste para obter testes unitários executáveis, executar os testes unitários e testes de aceitação para obter relatórios de teste, realizar análise estática para encontrar problemas no código-fonte e produzir pacote instalável de componentes pré-construídos. (CHIN-YUN, 2015) Já o *deploy* contínuo é uma abordagem de engenharia de software em que as equipes continuam

produzindo software valioso em curtos período de tempo e garantem que o software possa ser liberado de forma confiável a qualquer momento.(CHEN, 2015)

3 Metodologia

Neste capítulo, a metodologia de desenvolvimento do Agromart será apresentada. A seção 3.1, apresenta o planejamento gerencial para o desenvolvimento do software.

3.1 Gerência do Projeto

A implementação da proposta apresentada por este trabalho utilizará uma abordagem ágil de desenvolvimento de software, porém não seguiremos à risca uma metodologia específica, serão utilizados recursos que se encaixam no nosso contexto diferentes dentre os presentes em diferentes metodologias ágeis, essa abordagem é inspirado no trabalho apresentado por (PAGOTTO, 2016) em "Scrum solo: Software process for individual develop", onde é proposto uma adaptação do *Scrum* para desenvolvimento de software de forma individual, onde o *Scrum* foi adaptado para um contexto diferente do convencional.

Dentro do *scrum* utilizaremos os recursos:

- Product Backlog - representa a lista contendo todas as funcionalidades desejadas para o produto, sendo que o conteúdo é dinâmico, podendo mudar ou ser incremental de acordo as necessidades dos usuários;
- Sprints - as *sprints* representam o período de tempo no qual o conjunto de atividades definido será realizado, para o nosso contexto utilizaremos duas semanas por o software se tratar de uma aplicação web e uma mobile;
- Sprint Backlog - representa a lista de tarefas que a equipe se compromete a realizar durante uma *Sprint*;
- Sprint Planning Meeting - ocorrerá no início de uma *sprint* e tem como objetivo organizar o funcionamento da *sprint*;
- Sprint Review Meeting - ocorrerá ao final de cada *sprint*, durante a reunião será discutido os pontos positivos e negativos, podendo sempre propor sugestões de melhoria para o projeto;

Em projetos de software, o uso do *KANBAN* está se tornando cada vez mais popular, independentemente das etapas do projeto ou dos métodos de produção. Um aspecto interessante desta técnica é que ela pode ser utilizada tanto em em processos ágeis como em modelos cascata de desenvolvimento de software. (MAJCHRZAK M.; STILGER, 2015)

Dentro do *KANBAN*, utilizaremos o quadro de tarefas para definir quais funcionalidades são do *Product Backlog*, quais estão no *Backlog da Sprint*, bem como aquelas que estão sendo feitas, as que estão para serem revisadas ou já foram concluídas, o quadro será dividido da seguinte forma:

- Backlog do Agricultor - representa o *backlog* do projeto web, aqui estarão todas as tarefas relacionadas ao desenvolvimento do produto que será utilizado pelos agricultores;
- Backlog do Consumidor - representa o *backlog* do projeto mobile, aqui estarão todas as tarefas relacionadas ao desenvolvimento do produto que será utilizado pelos clientes dos agricultores;
- Bugs / Melhorias - representa a lista de problemas encontrados que precisam ser corrigidos para um bom funcionamento do software, assim como sugestões de melhorias que visam aprimorar o produto;
- A fazer - representa o *backlog* da *sprint*, aqui estarão todas as tarefas que foram planejadas para a *sprint*;
- Fazendo - aqui estarão todas as tarefas que estão em execução;
- Em revisão - representa a lista de tarefas concluídas que estão aguardando validação para serem consideradas concluídas;
- Concluído - representa a lista de tarefas que estão completas;

O *eXtreme Programming* é uma metodologia que visa boas práticas de programação e desenvolvimento em geral, se baseia no valor do produto e priorizam o trabalho em equipe. Desta metodologia, utilizaremos sempre que possível a programação em pares, para incentivar a troca de conhecimento e aumentar o entendimento da equipe em relação a implementação do software. Outra prática presente nesta metodologia que abordaremos é a de refatoração e padronização, onde será utilizada uma folha de estilo para padronizar o código, assim como a prática da refatoração constante visando elevar a qualidade do mesmo.

3.2 Backlog

O *backlog* deste projeto usufruíra de conceitos das histórias de usuário que são utilizadas em metodologias ágeis, porém não seguiremos à risca as convenções de escrita das mesmas, o conceito aplicado refere se em que a descrição das tarefas do *backlog* deverão ser curtas, simples e de fácil entendimento.

3.2.1 Backlog do Agricultor

O *backlog* do agricultor representa as tarefas a serem realizadas para o desenvolvimento do projeto web, que será utilizado pelos produtores agrícolas para anunciar e disponibilizar seus produtos dentro das regras de negócio do software.

- Gerenciar conta - deve ser possível criar e editar a conta de um usuário, assim como recuperar o acesso em caso de perda da senha;
- Gerenciar planos - deve ser possível criar, editar, visualizar e remover um plano, onde os planos representam assinaturas das cestas agroecológicas;
- Gerenciar cestas - deve ser possível criar, editar, visualizar as cestas que estarão disponíveis para os assinantes dos planos e para compras avulsas dependendo da disponibilidade das mesmas;
- Gerenciar estoque de produtos avulsos - deve ser possível criar, editar, visualizar e remover produtos avulsos, que podem ser incluídos pelos clientes às cestas mediante da disponibilidade dos mesmos;
- Gerenciar assinantes - deve ser possível visualizar os assinantes dos planos;
- Gerenciar extratos - deve ser possível criar, editar e visualizar os extratos gerados por pedidos;

3.2.2 Backlog do Consumidor

O Backlog do Consumidor representa as tarefas a serem realizadas para o desenvolvimento do aplicativo mobile, que será utilizado pelos clientes dos agricultores para encontrar seus produtos e realizar as assinaturas dos planos, dentro das regras de negócio do software.

- Gerenciar conta - deve ser possível criar e editar a conta de um usuário, assim como recuperar o acesso em caso de perda da senha;
- Acesso às lojas e produtos - deve ser possível encontrar as lojas dos agricultores através de uma listagem geral, ou filtragem por pesquisa e localização;
- Gerenciar plano - deve ser possível encontrar os planos de assinatura de cada loja, assim como assinar e gerenciar uma assinatura;
- Pedidos - deve ser possível realizar o pedidos de cestas e produtos avulsos mediante da disponibilidade dos mesmos;

- Gerenciar endereços - deve ser possível adicionar, editar e remover um endereço de recebimento;
- Acesso ao suporte - deve ser possível visualizar informações de contato do agricultor responsável pelo plano assinado;

3.3 Trello

O *Trello* é a ferramenta de colaboração e gerência de projeto que iremos utilizar neste trabalho, esta ferramenta organiza projetos em quadros, com o objetivo de informar rapidamente o que está sendo trabalhado, quem está trabalhando em quê, e onde algo está em um processo. É a partir dele que será construído o *KanBan* que servirá para acompanhar o andamento do projeto.

4 Suporte Tecnológico

Este Capítulo tem como objetivo descrever os aspectos técnicos referentes ao desenvolvimento do Agromart.

4.1 Linguagens e Frameworks

O Agromart será desenvolvido utilizando *React* para a implementação da interface utilizada pelo agricultor e *React Native* para a aplicação mobile, pois ambos utilizam a mesma linguagem de programação *JavaScript* e possuem também uma sintaxe muito semelhante, o que facilita o entendimento e a aprendizibilidade dos códigos de cada interface.

O *React* foi criado em 2013 pelo Facebook, sendo uma biblioteca de código aberto focada em criar interface de usuário em páginas web (FACEBOOK, 2020a), já o *React Native* foi criado em 2015 também pelo Facebook, ambos com o conceito presente no slogan "Learn once, write anywhere.", que representa a capacidade de aprender apenas uma linguagem de programação e possibilitar o desenvolvimento de aplicações web, que também podem ser renderizadas por um servidor, e aplicações mobile, tanto *iOS*, quanto *Android*. (FACEBOOK, 2020b).

O *JavaScript* é uma linguagem de programação interpretada e orientada a objetos. É uma das linguagens mais utilizadas no mundo, e apesar de seu uso ter sido durante muito tempo predominantemente no desenvolvimento *Web*, com o surgimento dos *frameworks* e bibliotecas para desenvolvimento de servidores e de aplicações mobile multiplataforma sua utilização tem aumentado cada vez mais em todas as áreas. (CASS, 2017)

4.2 Arquitetura

Nossa arquitetura vai seguir o modelo cliente-servidor que é uma estrutura de aplicação distribuída que distribui as tarefas e cargas de trabalho entre os fornecedores de um recurso ou serviço, designados como servidores, e os requerentes dos serviços, designados como clientes(BERSON, 1996), uma vez que vamos possuir dois *front-ends* cada um dedicado a um segmento de nosso usuários, o que gera a necessidade de um *back-end* atuando como servidor. A comunicação entre cliente e servidor se dará por meio do padrão http, que é um protocolo de transferência de hipertexto de nível de aplicação com a leveza e a velocidade necessárias para sistemas de informação hipermídia distribuídos e colaborativos. (FRYSTYK, 1996)

A arquitetura interna dos nossos clientes seguirá o padrão imposto pelos *frameworks* que vamos utilizar, pois consideramos adequada às nossas necessidades. Toda a comunicação com serviços externos será feita através do nosso *back-end* tornando-o a única fonte de informações e serviços consumida por nossos *front-ends*.

4.3 Gerência de Configuração de Software

A política de *branch* estabelecerá o fluxo de trabalho por meio de ambientes isolados, garantindo que o desenvolvimento seja feito de forma organizada e controlada, diminuindo os riscos que o desenvolvimento coletivo proporciona e assegurando a integridade do código que será utilizado em produção. A Figura 4 detalha a política de *branchs* dos nossos repositórios.

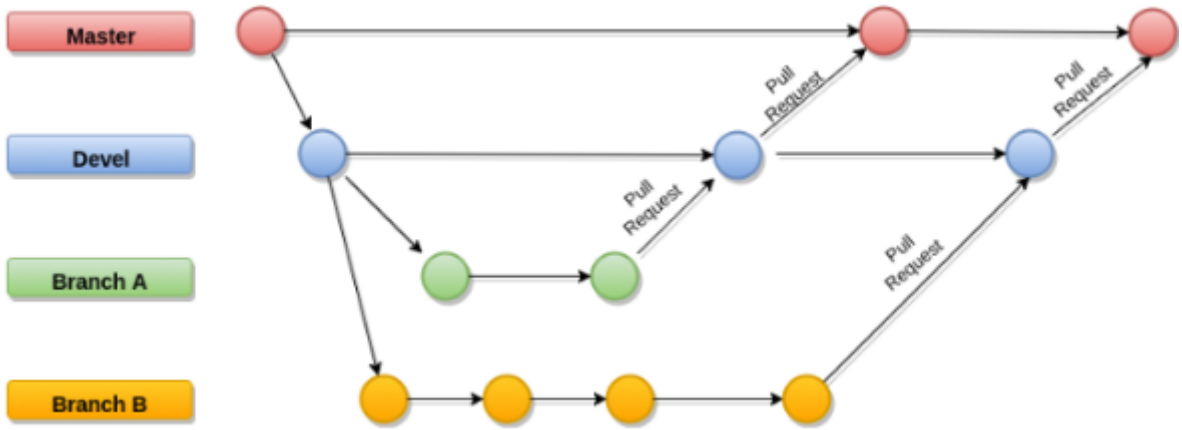

Figura 4 – Fluxo de trabalho

Fonte: Autor

O repositório do projeto terá uma *branch* principal, sendo ela a *branch* estável, a *master*. Haverá uma *branch* denominada devel destinada a ser a branch de desenvolvimento.

A *master* será a *branch* estável do projeto, sendo ela a proveniente da *devel* por meio de aprovação de *pull requests* ao fim de cada *release*. Nenhum *commit* será feito diretamente nesta *branch*, o código contido na mesma representará o produto final em produção.

As *branches* auxiliares são destinadas a implementação de funcionalidades, realização de tarefas de cada textitsprint e conserto de *bugs*. Cada uma dessas atividades terá sua própria *branch* , criada a partir da *devel* , após o fim do desenvolvimento nas *branches* auxiliares elas devem ser incorporadas a *devel* por meio de *pull request*.

4.3.1 Entrega Contínua

A entrega contínua do nosso *back-end* e da nossa interface do agricultor será feita utilizando o *heroku*. O *heroku* facilita a automação do fluxo de trabalho do software, funcionando para integração e deploy contínuos, esta ferramenta possibilita realizar a *build*, rodar testes e subir o código da aplicação direto do repositório do *github*, fazendo parte da revisão do código e gerenciamento de *branch* da forma que decidimos (HEROKU, 2021).

Para nossa aplicação mobile vamos delegar ao *Expo* a entrega contínua, aplicativos desenvolvidos com *Expo* podem fazer uso do seu sistema de atualizações *over the air* que facilitam a disponibilização de novas versões do aplicativo sem a necessidade de passar novamente pelo processo de revisão das lojas (EXPO, 2021). Outros fatores que ajudaram na escolha do *Expo* foram a facilidade que ele traz na geração de certificados e executáveis, tornando o processo de publicação dos aplicativos nas lojas de aplicativos muito mais fácil e amigável.

4.3.2 Análise Estática de Código

As ferramentas de análise estática de código tem por objetivo aumentar a qualidade de código por meio de processos que avaliam o código fonte e buscam falhas de implementação sem que este seja executado. Essas falhas podem estar relacionadas com variáveis não inicializadas, falhas de segurança, carregamento de recursos entre outros. A análise estática também pode ser utilizada para avaliar os padrões de implementação do código para determinadas linguagens (DELEY T.; GJORGJEVIKJ, 2017).

Utilizaremos o *ESLint* para padronizar nossa folha de estilo, esta ferramenta é *open source* e tem como objetivo a avaliação de estilo de código fonte e identificação de más práticas associadas ao padrões do *javascript*. O *ESLint* já vem por padrão pré-configurado, no entanto iremos especificar as regras dentro do nosso contexto a serem obedecidas e o *ESLint* fará a avaliação se o código desenvolvido está dentro dos padrões. Esta técnica ajuda a manter a padronização e organização do código, dando identidade ao código que mesmo escrito por vários de desenvolvedores terá um único padrão.

4.3.3 Repositório

O *GitHub* é uma ferramenta gratuita para hospedagem de código fonte e utiliza o *Git* como ferramenta de controle de versão, além disso possui integração com diversas ferramentas e *plugins* que auxiliam no desenvolvimento de aplicações.(GITHUB, 2020)

Hospedaremos o código fonte do Agromart no *GitHub*, a escolha se deu devido a popularidade da ferramenta no meio da engenharia de software e a familiaridade e as boas experiências passadas pela equipe com a ferramenta.

4.4 Sistema de Gestão de Conteúdo

Para a escolha do CMS a ser utilizado no projeto levamos em conta alguns pontos, o primeiro é que necessariamente deve ser um CMS *Headless* que é um tipo de CMS que se concentra inteiramente na interface administrativa para criadores de conteúdo, na facilitação de fluxos de trabalho de conteúdo e na colaboração e organização do conteúdo em taxonomias. Ele não se preocupa com camadas de apresentação, modelos, estrutura do site ou design, mas armazena seu conteúdo em formato puro e fornece acesso a outros componentes por meio de *APIs* sem estado ou fracamente acopladas (TAMTURK, 2016).

Nossa intenção é utilizá-lo como *back-end* em formato de *API* e interface provisória para acesso dos Agricultores, o segundo ponto é que a utilização de muitas das soluções de CMS disponíveis no mercado está atrelada a compra de acesso e/ou hospedagem de/em serviço específico, limitando a personalização e nos deixando presos a um serviço que muitas vezes pode não ser vantajoso para o nosso contexto e/ou não ser escalável o suficiente.

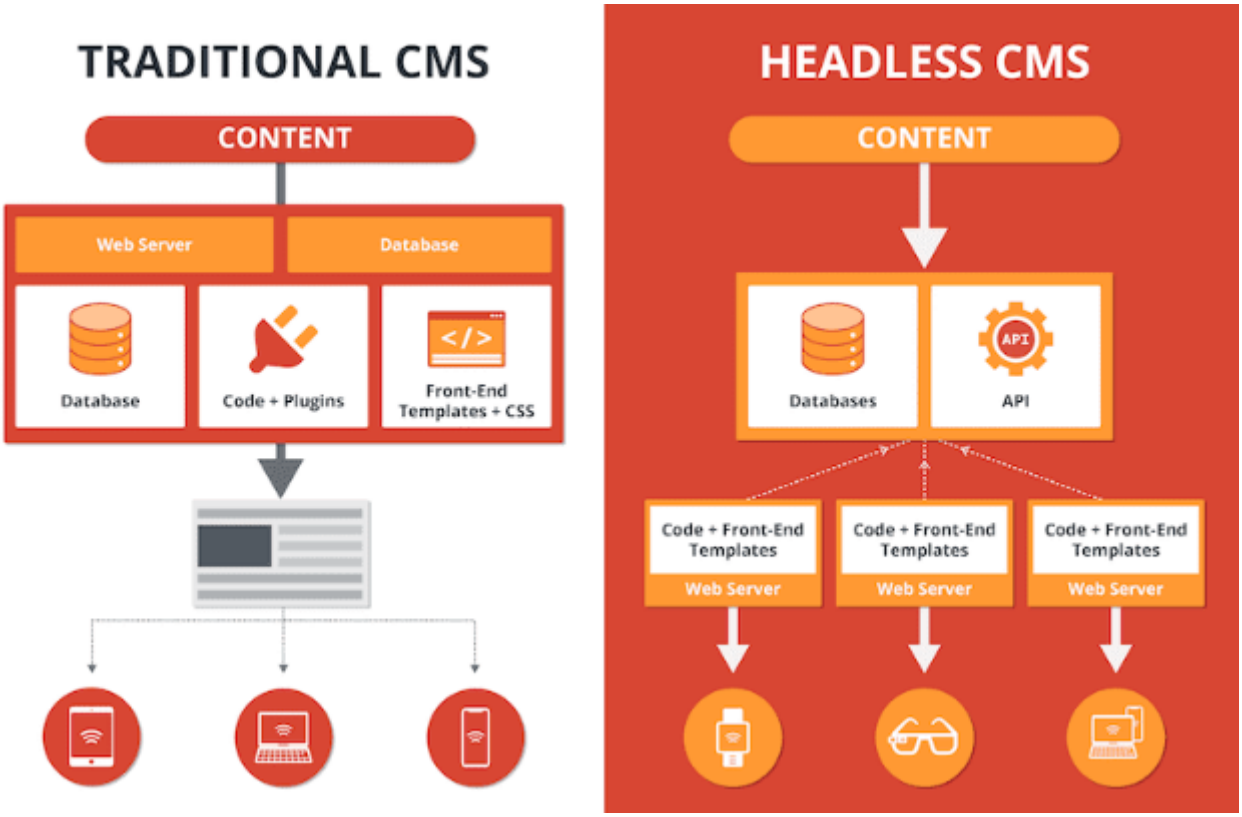

Figura 5 – Comparação entre CMS Tradicional e Headless CMS

Fonte: (JUSTEN, 2020)

Levando em conta os pontos levantados no parágrafo anterior tomamos a decisão de utilizar o *Strapi*, que é um CMS *Headless open source self-hosted* (STRAPI, 2020) ou

seja temos liberdade de customização e total controle sobre nossos dados e hospedagem. Além disso a linguagem base utilizada pelo *Strapi* é o *javaScript* o que vem de acordo com a *stack* que utilizaremos no projeto tornando nosso conhecimento reaproveitável e facilitando a adição de *plugins* ao código caso seja necessário.

4.5 Licença

Inicialmente planejamos o projeto com código fechado, porém com o passar do desenvolvimento vimos que seria mais vantajoso deixá-lo como software livre, visto que o projeto poderá ser continuado pela comunidade, com o incentivo dentro da própria UnB, onde foi feito o levantamento de recursos para a manutenção do mesmo. Com essa decisão, foi necessário escolher qual licença de software livre seria utilizada dentre as três categorias existentes:

- Licenças Permissivas
- Licenças Recíprocas Totais
- Licenças Recíprocas Parciais

Observando os conceitos das três categorias, escolhemos a licença GPL, que é uma licença recíproca total, onde o projeto será sempre livre e gratuito, com esta licença garantimos que toda a cópia, execução, modificação e redistribuição do nosso projeto deve continuar livre.

4.6 Banco de Dados

Como a nossa *API* será feita utilizando o *Strapi* como CMS, teríamos quatro opções de banco de dados:

- SQLite
- MongoDB
- MySQL
- PostgreSQL

Consideramos que para o projeto não seria interessante um banco de dados não relacional, pois teríamos muitas relações entre nossas tabelas, logo descartamos o *MongoDB* para escolha da nossa base da dados, as outras três opções possuem um comportamento

bem semelhante, porém levando em consideração a experiência da equipe, optamos pelo *PostgreSQL*.

O *PostgreSQL* é um poderoso sistema de banco de dados relacional de código aberto com mais de 30 anos de desenvolvimento ativo que lhe rendeu uma forte reputação de confiabilidade, robustez de recursos e desempenho. *PostgreSQL* vem com muitos recursos destinados a ajudar os desenvolvedores a construir aplicativos, administradores para proteger a integridade dos dados e construir ambientes tolerantes a falhas, e ajudá-lo a gerenciar seus dados, não importa o tamanho do conjunto de dados. Além de ser gratuito e de código aberto, o *PostgreSQL* é altamente extensível. Por exemplo, você pode definir seus próprios tipos de dados, construir funções personalizadas e até mesmo escrever código de diferentes linguagens de programação sem recompilar seu banco de dados (POSTGRESQL, 2021)

5 Resultados

5.1 Aplicativo

Nessa seção será demonstrada as funcionalidade do aplicativo Agromart.

5.1.1 Cadastrar Usuário

Para efetuar um cadastro o usuário deve informar seu e-mail, nome, senha e confirmar a senha, nos campos apresentados na Figura 6, e o acesso é feito pela tela de login (Figura 7).

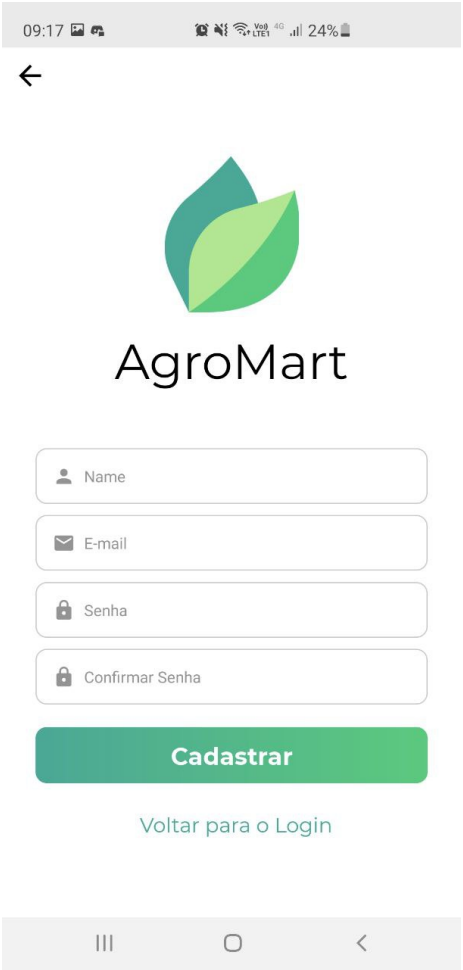

Figura 6 – Aplicativo: Cadastrar usuário

Fonte:Autor

5.1.2 Login de Usuário

Após o cadastro, o usuário poderá logar com e-mail e senha preenchendo o formulário representado na Figura 7, o acesso a tela de login pode ser feito pelo toque no texto "Olá, Faça Login ou cadastro", também pode ser acessado pelo toque no cartão escrito "Login"(Figura 8).

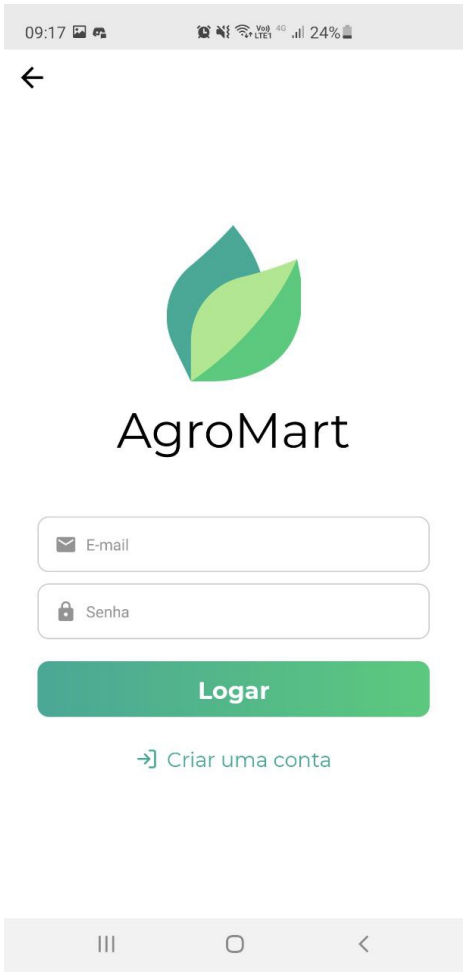

Figura 7 – Aplicativo: Login

Fonte:Autor

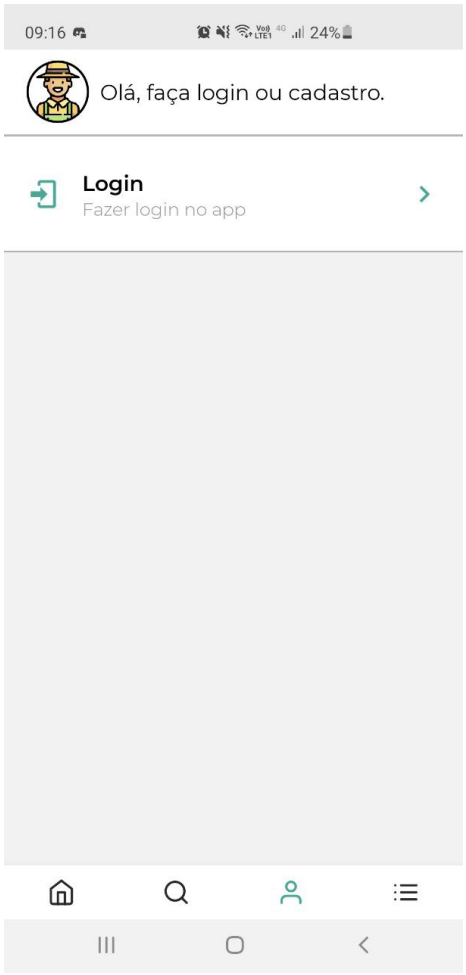

Figura 8 – Aplicativo: Perfil Sem Usuário Logado

Fonte:Autor

5.1.3 Lista de Lojas

A lista de lojas é representada na tela inicial (Figura 9), cada cartão representa uma loja, com um *banner*, nome e região administrativa em que a loja se encontra. Ao realizar o toque neste cartão, o usuário será redirecionado à tela de detalhes da loja (Figura 9), onde se encontraram as seguintes informações: nome da loja, região administrativa, descrição da loja, *banner* e um botão que redirecionará o usuário para um aplicativo de mensagens com o contato da loja.

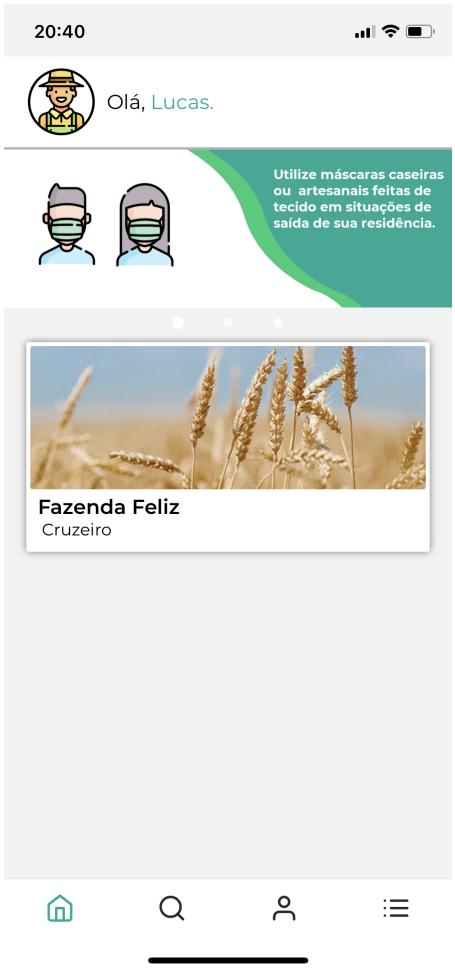

Figura 9 – Aplicativo: Listagem de lojas

Fonte:Autor

5.1.4 Pesquisar Lojas

A pesquisa pode ser encontrada na aba de busca na barra de navegação, e a pesquisa pode ser feita pela região administrativa onde a loja se encontra ou pelo nome da loja (Figura 10). Com a pesquisa feita, o usuário visualizará uma listagem de cartões semelhantes ao da página principal (Figura 9), e o toque no cartão também redirecionará o usuário a tela de detalhes da loja (Figura 11).

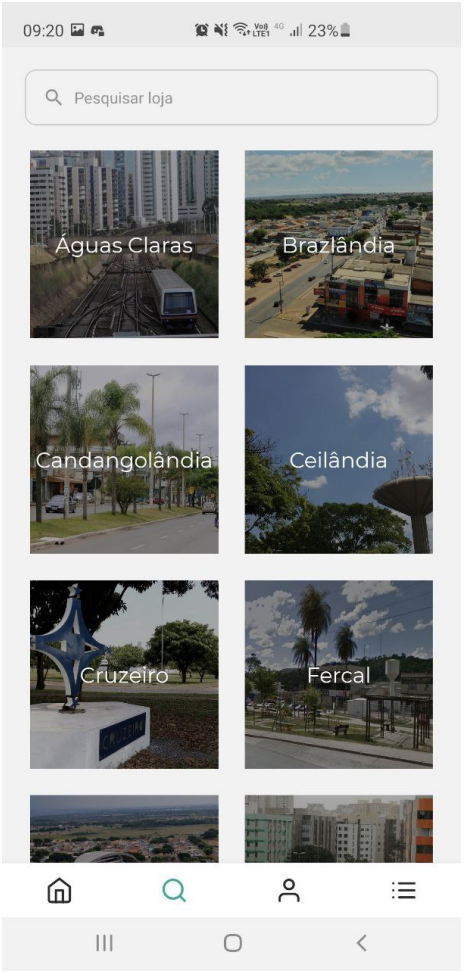

Figura 10 – Aplicativo: Pesquisar Lojas

Fonte:Autor

5.1.5 Fluxo Pedido

Para que um pedido seja feito, o usuário ao acessar a tela de detalhes da loja (Figura 11), deverá escolher os produtos que ele deseja, podendo navegar entre: cestas, planos e produtos avulsos. Ao selecionar um produto, o usuário irá visualizar uma imagem e uma descrição referente ao produto, com o objetivo de adquirir informações adicionais do mesmo (Figura 12), e também escolher a quantidade desejada e adiciona-lo o seu carrinho.

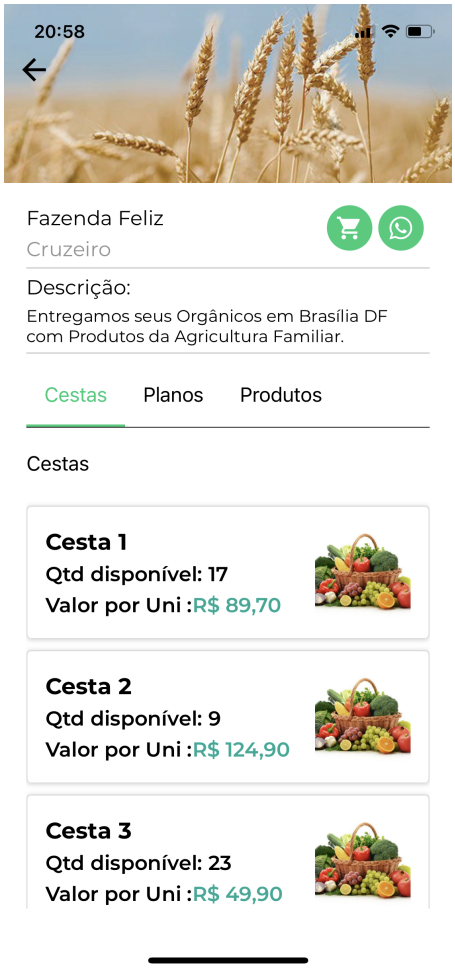

Figura 11 – Aplicativo: Detalhes de uma loja

Fonte:Autor

Figura 12 – Aplicativo: Detalhes de um produto

Fonte:Autor

Depois de escolher os produtos e suas quantidades o usuário poderá acessar o seu carrinho de pedidos (Figura 13), onde poderá gerenciar e visualizar o total do seu pedido, podendo adicionar novos itens, alterar os itens escolhidos ou remove-los. Caso necessário o usuário também pode alterar o seu endereço, na tela é possível visualizar se o endereço está correto, e quando o usuário estiver satisfeito basta finaliza o pedido.

Figura 13 – Aplicativo: Carrinho

Fonte:Autor

5.1.6 Histórico de Pedidos

O histórico de pedidos (Figura 14) pode ser acessado pela barra de navegação, todos os pedidos estarão presentes representados por cartões, informando o vendedor, a data do pedido e o valor, caso o usuário deseje visualizar mais informações, basta tocar em "Ver Detalhes"e uma modal será aberta com as informações adicionais informando se o pedido foi entregue e se o pagamento foi realizado (Figura 15).

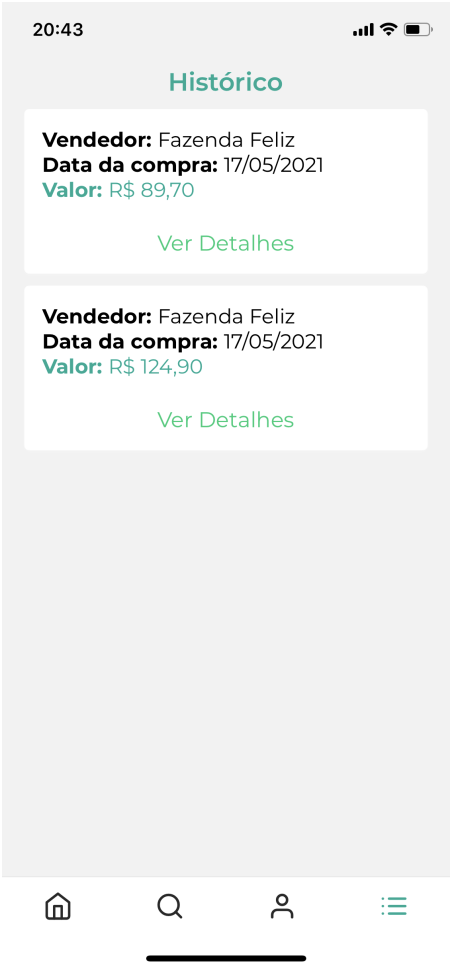

Figura 14 – Aplicativo: Histórico de pedidos

Fonte:Autor

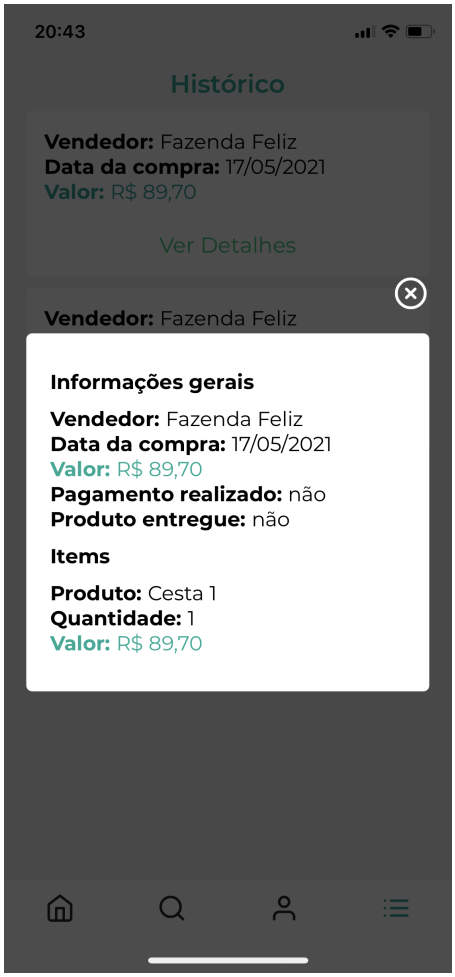

Figura 15 – Aplicativo: Modal com detalhes de um pedido

Fonte:Autor

5.1.7 Perfil

O acesso ao Perfil é feito pela barra de navegação, na tela de perfil o usuário terá as opções de gerenciar os dados de perfil, gerenciar o endereço de entrega, visualizar os planos assinados ou fazer logout da aplicação (Figura 16).

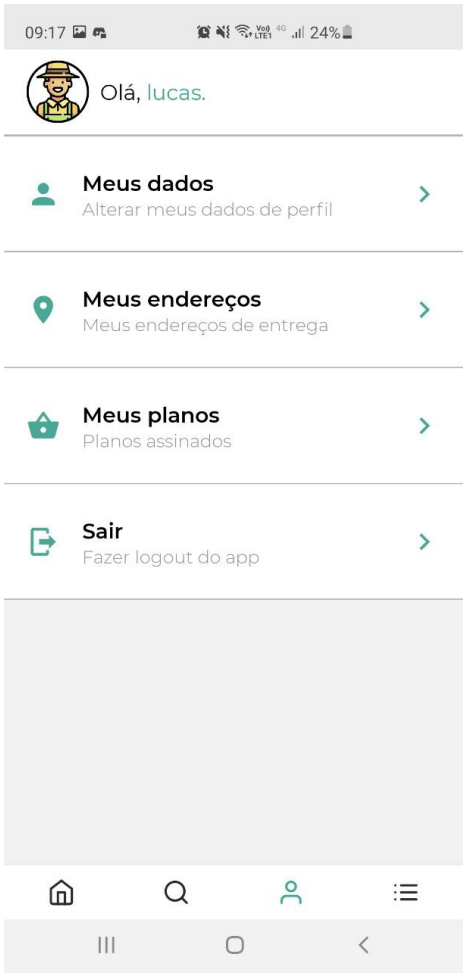

Figura 16 – Aplicativo: Perfil

Fonte:Autor

Ao tocar em "Alterar meus dados de perfil"o usuário será redirecionado para uma tela com informações de nome e e-mail (Figura 17), podendo alterar apenas o seu nome dentro da aplicação.

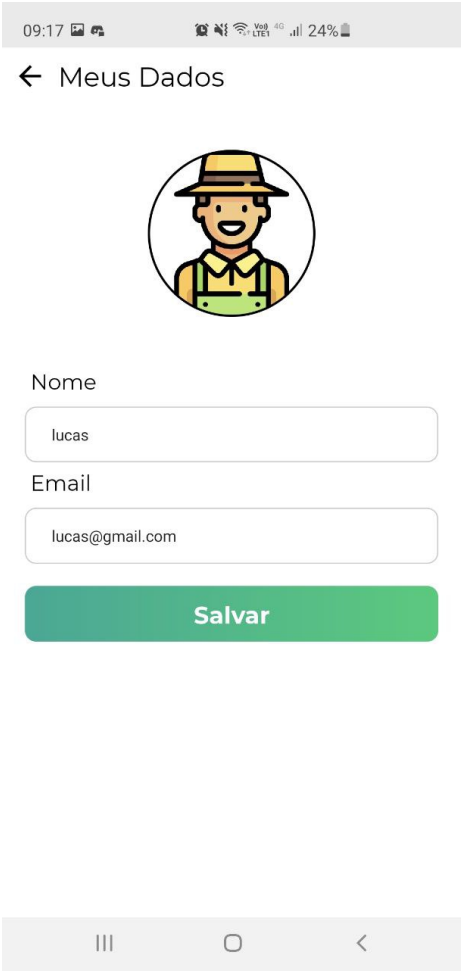

Figura 17 – Aplicativo: Informações do usuário

Fonte:Autor

Ao tocar em "Meu endereço", o usuário será redirecionado para uma tela com um formulário com os dados do seu endereço, podendo alterar ou registrar o endereço em que deseja receber os seus pedidos, os dados são: CEP, cidade, região administrativa, rua, número e complemento (Figura 18).

Figura 18 – Aplicativo: Formulário de Endereço

Fonte:Autor

Ao tocar em "Meus planos", o usuário será redirecionado para uma tela com um cartão com as informações referentes ao plano assinado: loja responsável pelo plano, número de contato, data de aquisição, cestas disponíveis, cestas recebidas, e opção para pular a cesta da semana (Figura 19).

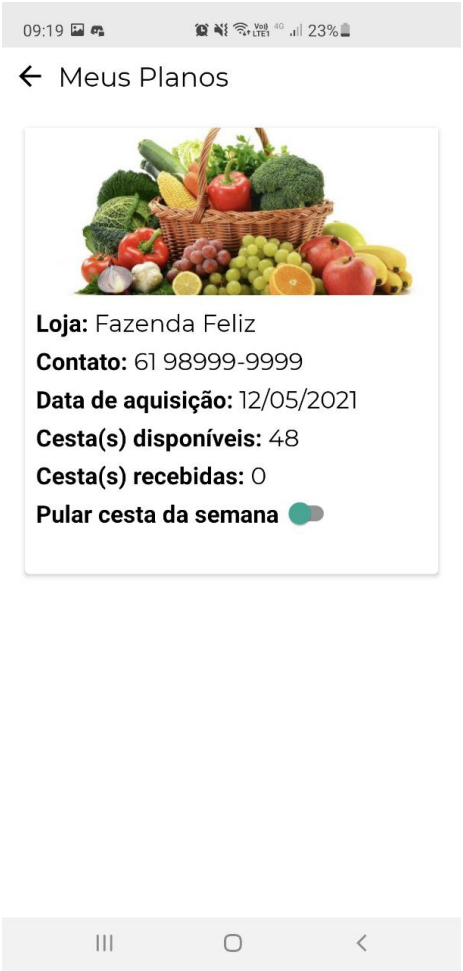

Figura 19 – Aplicativo: Planos Assinados

Fonte:Autor

Ao tocar em "Sair"o usuário realizará o logout do aplicativo, voltando para o estado sem usuário logado (Figura 8).

5.2 Interface Agricultor

Nessa seção será apresentada as funcionalidade da interface web do Agromart gerada pelo *strapi*, responsável pela gerencia pelo lado do agricultor.

5.2.1 Cadastrar Usuário

O cadastro de um agricultor deve ser feito por um super administrador, e confirmado pelo agricultor via e-mail, cada agricultor terá permissão para alterar apenas os dados criados por ele, dentro do *strapi* está permissão é chamada de *"Author"*.

5.2.2 Autenticar Usuário

A autenticação é feita com e-mail e senha, como representada na Figura 20, com a opção de se manter logado clicando em "Lembre-se de mim".

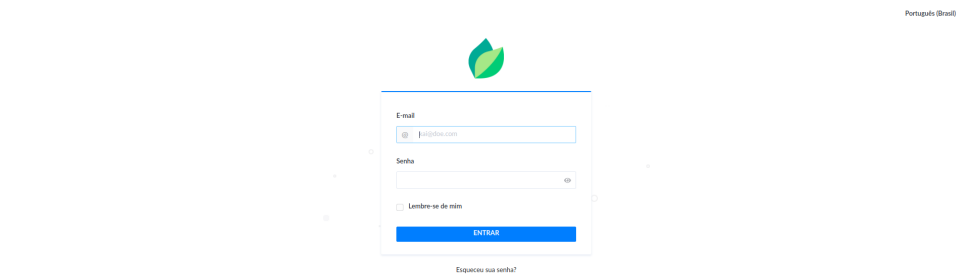

Figura 20 – Interface Agricultor: Login

Fonte:Autor

5.2.3 Padrão do Strapi

Com o usuário já autenticado, ele terá acesso ao nosso sistema de gerenciamento de conteúdo, como dito no capítulo anterior, utilizamos o *Strapi*, o uso dele pelo agricultor será feita a partir da criação de *"coleções"*, na barra lateral (Figura 21) estará presente todos os *"tipos de coleção"* da aplicação:

- Assinantes: referente aos assinantes de um plano, os dados desta coleção são: nome, cestas disponíveis, plano, e opção para pular a cesta.
- Cestas: Referente ao estoque de cestas de uma loja, os dados desta coleção são: valor, quantidade, descrição, loja.
- Endereços: referente ao endereço da loja e de usuários, os dados da coleção são: cidade, bairro, número, complemento, CEP e rua.
- Extratos: referente ao histórico de vendas de uma loja e de pedidos de um usuário, os dados da coleção são: itens, valor total, verificação de entrega, verificação de pagamento, tipo de entrega e a loja.
- Lojas: referente as lojas de um usuário, os dados da coleção são: nome, descrição, *banner*, tipos de entrega, contato, CNPJ, endereço, planos, produtos avulsos e cestas.

- Planos: referente ao estoque de planos de uma loja, os dados da coleção são: nome, descrição, valor, quantidade de unidades do plano, quantidade de cestas, assinantes e loja.
- Produtos Avulsos: Referente ao estoque de produtos avulsos, os da dados da coleção são: Nome, imagem do produto, unidade de medida, descrição, valor, quantidade e loja.

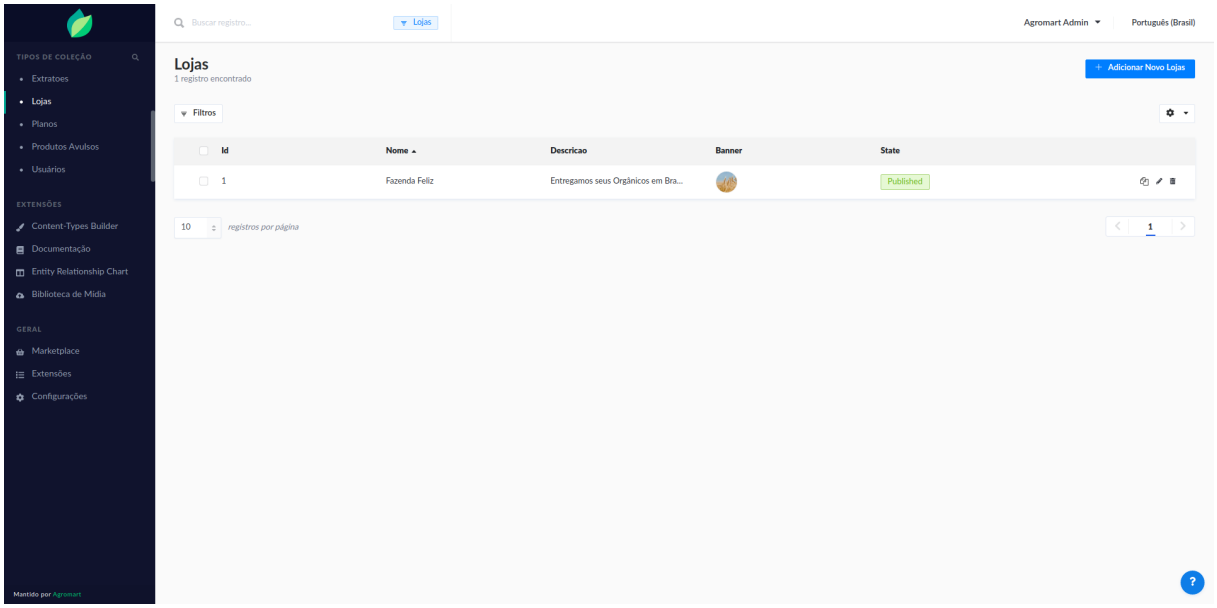

Figura 21 – Interface Agricultor: Visualização

Fonte:Autor

O usuário deve selecionar o tipo de coleção que deseja gerenciar, a estrutura da página pode ser visualizada na Figura 21, onde temos uma tabela com as informações do tipo de coleção, um botão azul no canto superior para cadastro de um novo tipo, que ao clicar o usuário será redirecionado para uma página de formulário com os campos referentes ao tipo de coleção (Figura 22). Cada item na tabela possui um conjunto de ações em seu final, onde é possível visualizar, editar ou excluir o item. Na barra superior é possível realizar uma busca pelo item desejado na tabela, e no canto inferior da tabela temos as opções de voltar para a página anterior e ir para a próxima página.

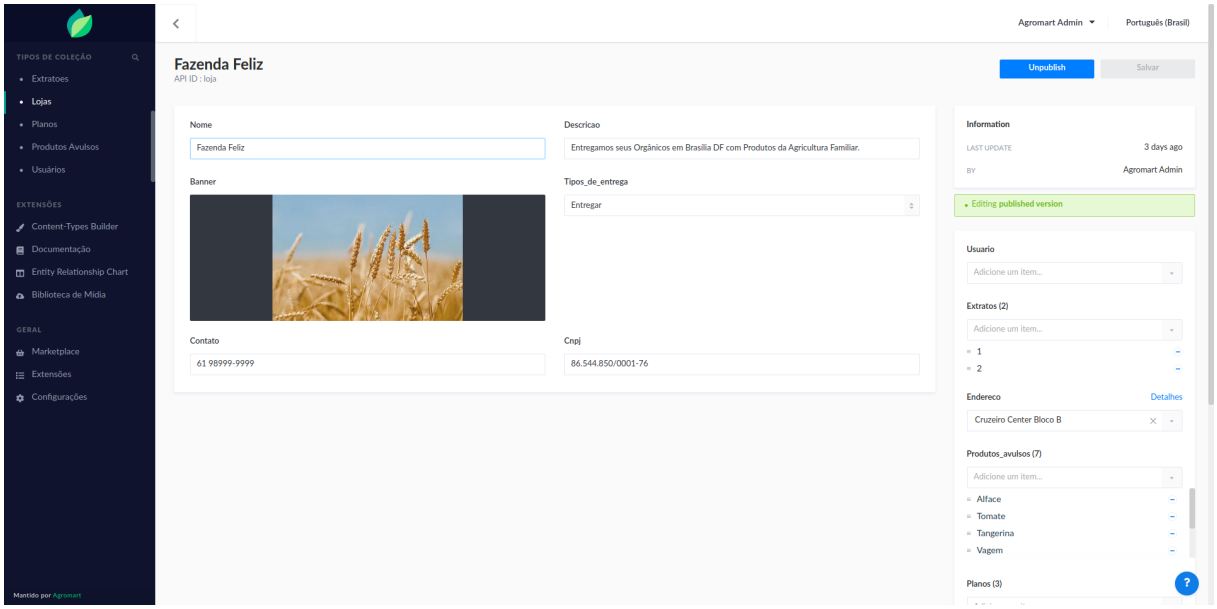

Figura 22 – Interface Agricultor: Formulário

Fonte:Autor

5.3 Estrutura do Projeto

5.3.1 Estrutura Aplicativo

O *React Native* apesar de ter uma estrutura inicial, não impõe uma estrutura rígida para o código fonte, porém com o objetivo de garantir uma melhor organização e manutenibilidade, foi utilizada uma estrutura de pastas para que os arquivos do projeto fossem separados de forma que os arquivos relacionados a mesma finalidade estivessem agrupados.

O código referente ao projeto e que não faz parte da estrutura padrão do *React Native* está presente na pasta *"src"* do projeto, e nela é feita a divisão categórica do código fonte.

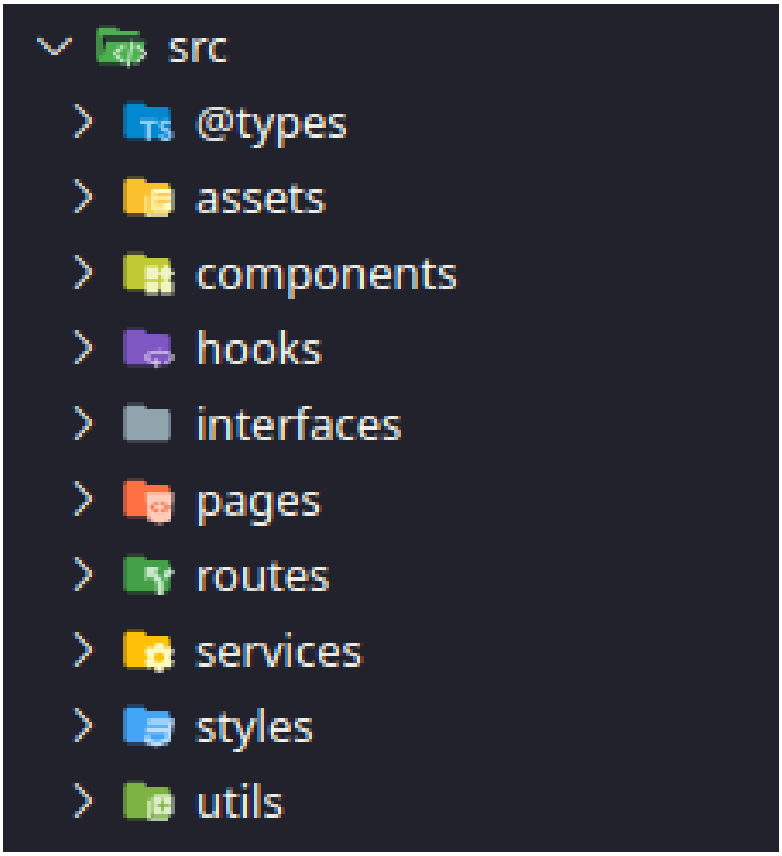

Figura 23 – Estrutura Código Fonte do Aplicativo

Fonte:Autor

- @Types: a pasta é referente a adição de tipagem vindas de bibliotecas externas.
- assets: está pasta é responsável pelo armazenamento das imagens utilizadas diretamente do código fonte, ou seja, as que não são consumidas da nossa *API*.
- components: contém os componentes isolados e reaproveitáveis do aplicativo. Todo componente é constituído por 2 arquivos, são eles o *index.tsx* e *styles.ts*, o primeiro é responsável pela exportação e codificação do componente, já o segundo tem a responsabilidade de conter o código de estilização.
- hooks: contém o *Hooks* customizados, a criação de *Hooks* próprios permite que a extração da lógica de um componente em funções reutilizáveis, que também é responsável pelo gerenciamento de estado (FACEBOOK, 2020a), esse diretório possui, além de arquivos para cada*Hook*, um arquivo *index.tsx*, responsável pela exportação do uso das permissões para utilização dos *hooks*.
- interfaces: contém as interfaces comuns utilizados para tipagem dos dados da aplicação.

- pages: esta pasta é responsável pelo código das telas do aplicativo, que também são componentes e possuem a mesma estrutura, com um arquivo principal e outro para os estilos.
- routes: contém a lógica responsável pela criação de rotas, onde cada rota tem uma pagina como acesso.
- services: contém a criação do acesso a *API* que será consumida pelo aplicativo.
- styles: esta pasta é responsável por exportar e criar estilos globais, que podem ser utilizados por todos os componentes.
- utils: contém logicas isoladas que podem ser utilizadas por qualquer componente.

5.3.2 Estrutura Interface do Agricultor

O *strapi*, diferente do *React Native*, tem uma estrutura muito rígida e especifica de pastas para o seu funcionamento, logo devemos seguir seu padrão para codificação da *API* e de sua interface.

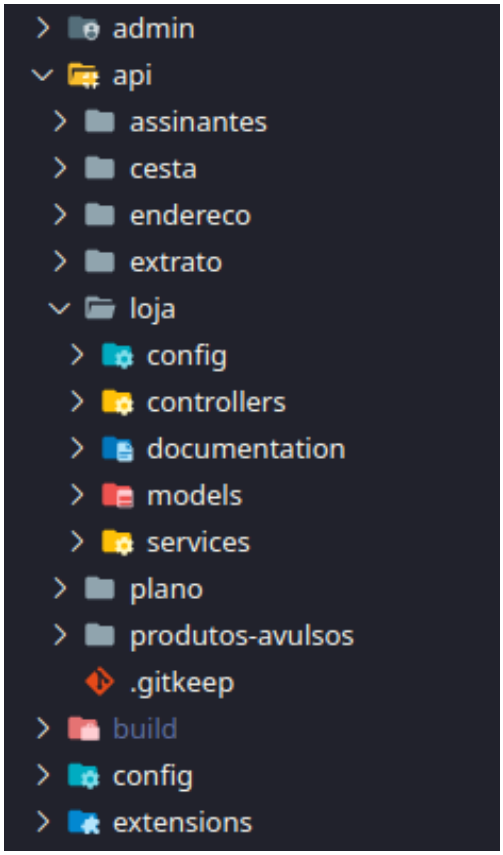

Figura 24 – Estrutura Código Fonte da Interface do Agricultor

Fonte:Autor

- admin: este diretório é responsável pelo código da customização da interface, o código é feito em *javascript* com *react*, e as alterações devem seguir exatamente o mesmo padrão de arquivos da pasta *admin* contida no "node modules", pois é uma dependência do projeto, que estamos alterando para uma melhor experiência para o usuário.
- api: contém todos os tipos de coleção, onde cada um possui sua própria estrutura que é gerada pelo strapi, após a criação de um tipo de coleção:
  - config: contém um arquivo *routes.json*, que possui a configuração de suas rotas.
  - controllers: possui um arquivo com o nome do tipo de coleção, onde é possível criar funções costumizadas para novas rotas.
  - documentation: contém um arquivo *json*, responsável pela documentação de suas rotas.
  - models: contém a estrutura dos dados do tipo de coleção.
  - services: possui um arquivo com o nome do tipo de coleção, onde é possível criar métodos isolados em serviços para utilizar em *controllers*.
- config: este diretório contém a configuração para execução da aplicação, como: conexão com banco de dados, porta de execução e chaves para autenticação.
- extensions: este diretório contém a instalação de extensões disponíveis na loja do *strapi*.

5.3.3 Documentação da API

Uma documentação de *API* facilita que desenvolvedores tenham um entendimento maior de um sistema, possibilitando a criação de interfaces para integração entre um ou mais sistemas, é uma entrega técnica de conteúdo com instruções sobre como usar e integrar efetivamente uma solução (VERSIANI, 2021).

A *API* do Agromart utiliza a tecnologia *Swagger*. As ferramentas *Swagger* eliminam o trabalho árduo de gerar e manter documentos de *API*, garantindo uma documentação sempre atualizada conforme a evolução da *API* (SWAGGER, 2021). Essa documentação possibilita que os agricultores tenham a liberdade de providenciar sua própria interface customizada, invés de utilizar a interface proposta pelo *strapi*.

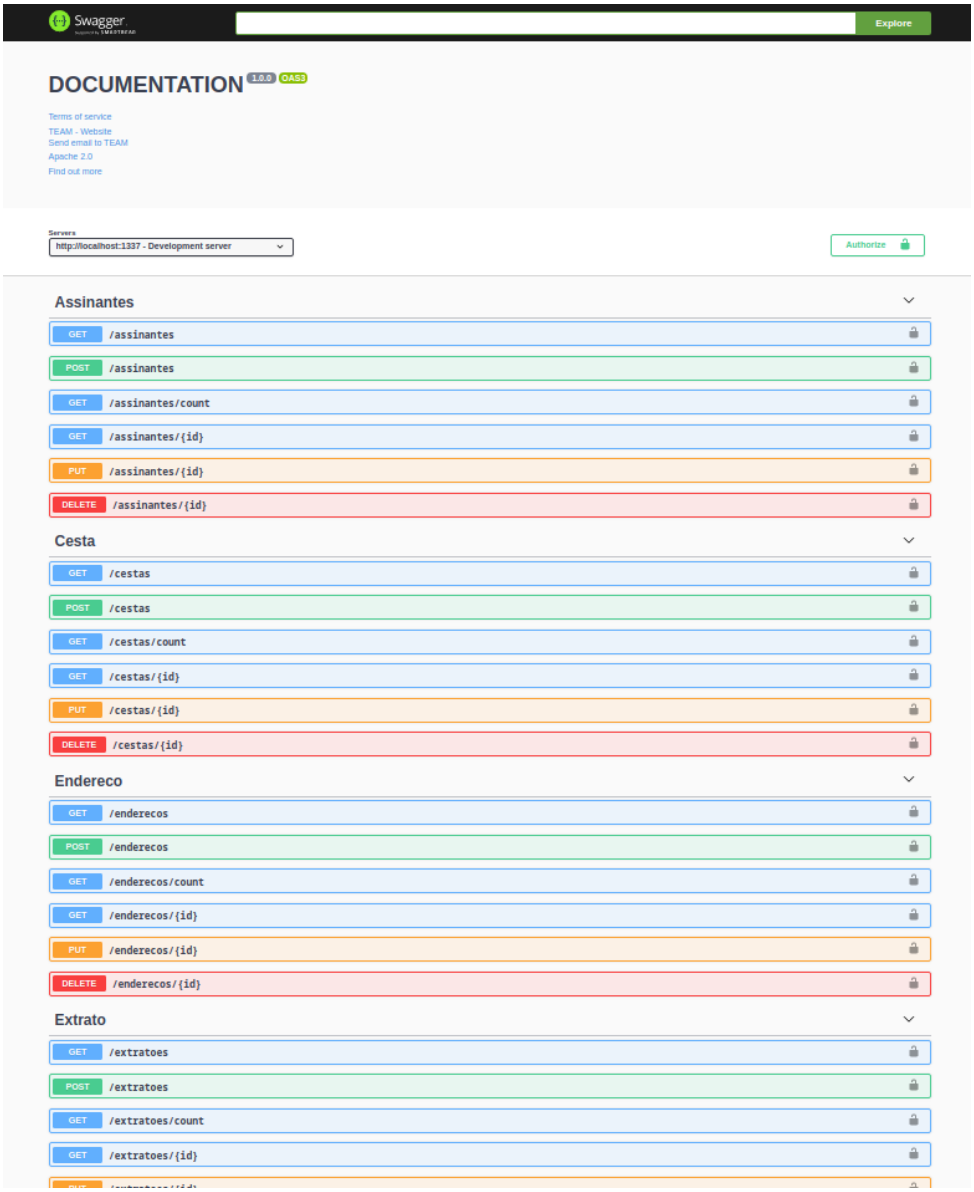

Figura 25 – Documentação API do Agromart

Fonte:Autor

5.3.4 Relacionamento de Entidades

Um Diagrama de Relacionamento de Entidade (ER) é um tipo de fluxograma que ilustra como “entidades” como pessoas, objetos ou conceitos se relacionam entre si dentro de um sistema. Os diagramas ER são usados com mais frequência para projetar ou depurar bancos de dados relacionais nas áreas de engenharia de software, sistemas de informação de negócios, educação e pesquisa (LUCIDCHART, 2021).

As entidades e relações do Agromart podem ser visualizadas na figura 26, onde uma loja se relaciona com suas cestas, produtos avulsos, planos, extratos e com um usuário que é o responsável por ela, já o usuário comum se relaciona com a entidade "Assinantes", para o caso de assinatura de plano, e também se relaciona com as entidades endereço,

extrato, e com as entidades *"Role"* e *"Permission"* que foram geradas pelo *strapi*.

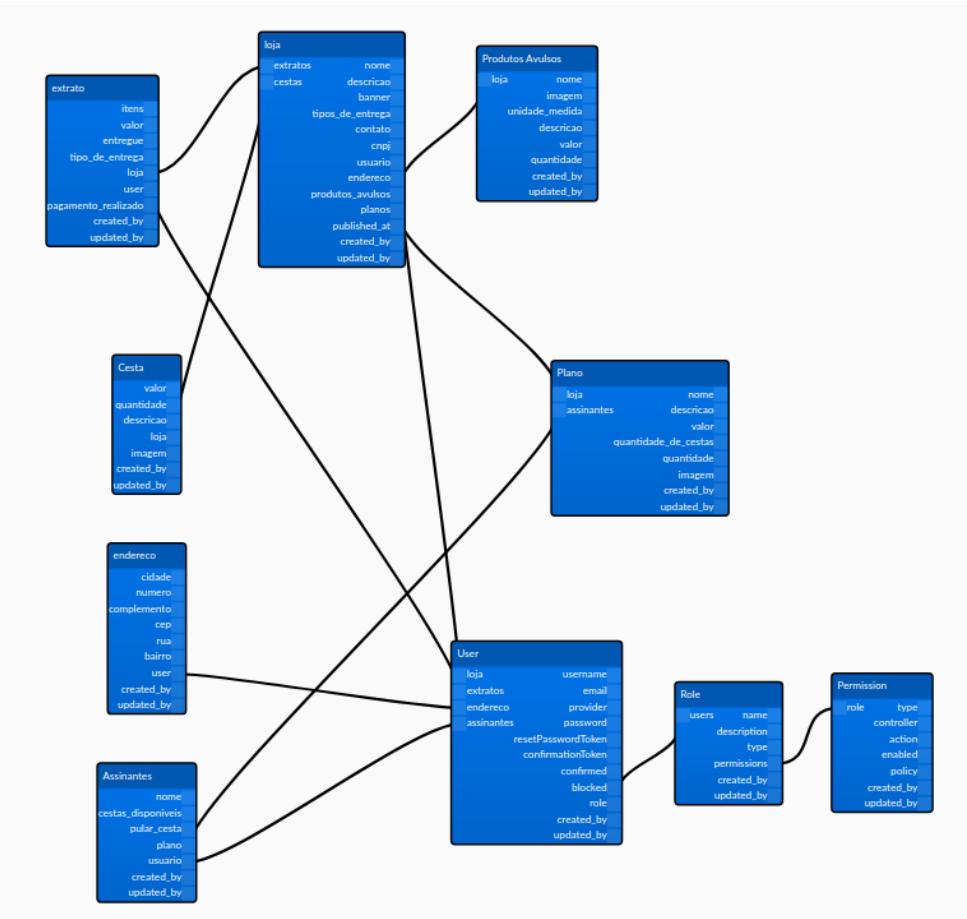

Figura 26 – Diagrama de Relacionamento de Entidades do Agromart

Fonte:Autor

6 Considerações Finais

Ao desenvolver esse trabalho passamos por vários obstáculos tais como adaptar a solução entregue no *Hackathon* para um contexto do mundo real, no caso o dos pequenos agricultores e modelar o projeto de uma forma que se enquadre com o nosso trabalho de conclusão de curso.

Outras dificuldades que tivemos foi a falta do contato direto entre os envolvidos no projeto devido ao isolamento social ocasionado pelo momento que estamos passando nesta pandemia e o encurtamento do semestre letivo para compensar a paralisação que ocorreu na universidade. Este encurtamento influenciou em todo planejamento inicial, com isso ajustes foram feitos para que fosse possível entregar o projeto com um escopo compatível com o tempo em questão.

Com isso foi tomada a decisão de utilizar a interface do *strapi* ao invés de codificar uma personalizada como era no planejamento inicial, apesar da solução ter se mostrado funcional, esta interface não é de fácil uso, ainda mais levando em consideração que parte do público alvo não é tão familiarizado com tecnologias, então uma melhoria futura para o projeto seria a implementação de uma interface mais agradável para os agricultores.

Outra decisão importante que foi tomada ao longo do projeto é a de tornar o produto em software livre, esta escolha torna o software mais acessível para a comunidade, além de ser uma tentativa de manter o software vivo, recebendo *feedbacks* e atualizações vindas da comunidade, e incentivando outros estudantes da UnB a contribuírem com a pesquisa em questão, tendo como beneficio que o projeto possui recursos financeiros a serem usados.

Como tivemos a ajuda de profissionais da área no levantamento do escopo, além de entrevistas e validações do nosso *backlog*, acreditamos que o produto tem um grande potencial de inovação e poderá impactar positivamente a vida dos pequenos agricultores do Brasil, uma vez que pode encurtar a distancia entre o agricultor e o consumidor final, eliminando a necessidade de intermediários, diminuindo assim o custo dos produtos para a população e aumentando o lucro para os agricultores.

A interface do agricultor pode ser acessada em: <https://agromart-cms.herokuapp.com/admin> e o aplicativo está publicado na *Play Store*, e pode ser encontrado pesquisando por *Agromart*, já a publicação na loja de aplicativos da plataforma *iOS* não foi realizada neste primeiro momento devido ao preço envolvido no processo e ao tempo de analise por parte da *App Store*.

Apesar dos obstáculos encontrados, estamos gratos com o resultado alcançado,

onde conseguimos entregar uma solução pensada em ajudar a sociedade em um momento de dificuldade, além de incentivar alunos a contribuir com pesquisas dentro da Universidade de Brasília, e por fim acreditamos que o projeto agrega em nosso portfólio, podendo abrir portas para novas oportunidades no futuro.

Referências

BAUER, F. *Software Engineering. [S.l.]*. [S.l.], 1972. Citado na página 18.

BERSON, A. *Client/Server Architecture*. [S.l.], 1996. Citado na página 27.

CASS, S. The top programming languages. 2017. Disponível em: <https://spectrum.ieee.org/computing/software/the-2017-top-programming-language>. Citado na página 27.

CHARLAND A ;LEROUX, B. *Mobile application development: Web vs. native. Queue, ACM, New York, NY, USA, v. 9, n. 4, p. 20:20–20:28, abr. 2011. ISSN 1542-7730*. [S.l.], 2011. Citado na página 18.

CHEN, L. *Continuous delivery: Huge benefits, but challenges too. IEEE Software, v. 32, n. 2, p. 50–54, Mar 2015. ISSN 0740-7459*. [S.l.], 2015. Citado na página 22.

CHIN-YUN, H. C.-T. *C.Patterns for Continuous Integration Builds in Cross-Platform Agile Software Development*. [S.l.], 2015. Citado na página 21.

DELEY T.; GJORGJEVIKJ, D. *tatic analysis of source code written by novice programmers. In: 2017 IEEE Global Engineering Education Conference (EDUCON). [S.l.: s.n.], 2017. p. 825–830*. [S.l.], 2017. Citado na página 29.

EGESTOR. Kanban: O que é, como funciona e como implantar esse método. 2020. Disponível em: <https://blog.egestor.com.br/kanban/>. Citado na página 21.

EXPO. Documentação do expo. 2021. Disponível em: <https://docs.expo.io/>. Citado na página 29.

FACEBOOK. React. 2020. Disponível em: <https://pt-br.reactjs.org/>. Citado 2 vezes nas páginas 27 e 50.

FACEBOOK. React native. 2020. Disponível em: <https://reactnative.dev/>. Citado na página 27.

FRYSTYK, T. B.-L. R. F. H. *RFC1945: Hypertext Transfer Protocol – HTTP/1.0*. [S.l.], 1996. Citado na página 27.

GIL, A. C. *Como elaborar projetos de pesquisa. 4th. ed. [S.l.]: Atlas, 2002. ISBN 85-224-3169-8*. [S.l.], 2002. Citado na página 16.

GITHUB, I. About github. 2020. Disponível em: <https://github.com/about>. Citado na página 29.

HEROKU. What is heroku? 2021. Disponível em: <https://www.heroku.com/what>. Citado na página 29.

HUMBLE J.; FARLEY, D. *tatic analysis of source code written by novice programmers. In: 2017 IEEE Global Engineering Education Conference (EDUCON). [S.l.: s.n.], 2017. p. 825–830*. [S.l.], 2017. Citado na página 21.

ISO/IEC/IEEE. *Systems and software engineering. [S.l.]*. [S.l.], 2010. Citado na página 18.

JENTSCH, C. S. The impact of agile practices on team interactions quality - insights into a longitudinal case study. 2017. Disponível em: <https://aisel.aisnet.org/amcis2017/SystemsAnalysis/Presentations/5/>. Citado na página 20.

JUSTEN, W. Introdução ao strapi - headless cms. 2020. Disponível em: <https://willianjusten.com.br/introducao-ao-strapi-headless-cms/>. Citado na página 30.

LTD, W. A. S. D. Global overview. 2017. Disponível em: <https://wearesocial.com/special-reports/digital-in-2017-global-overview>. Citado na página 18.

LUCIDCHART. What is an entity relationship diagram (erd)? 2021. Disponível em: <https://www.lucidchart.com/pages/er-diagrams>. Citado na página 53.

MAJCHRZAK M.; STILGER, L. *Experience report: Introducing kanban into automotive software project*. [S.l.], 2015. Citado na página 23.

NOLETO, C. Aplicações web betrybe. 2020. Disponível em: <https://blog.betrybe.com/desenvolvimento-web/aplicacoes-web/>. Citado na página 19.

PAGOTTO, T. F. J. A. . L. A. . G. J. A. *Scrum solo: Software process for individual development*. [S.l.], 2016. Citado na página 23.

PALMEIRA;, T. Como funcionam as aplicações web. DevMedia, 2012. Disponível em: <https://www.devmedia.com.br/como-funcionam-as-aplicacoes-web/25888>. Citado na página 19.

POSTGRESQL. *About*. [S.l.], 2021. Disponível em: <https://www.postgresql.org/about/>. Citado na página 32.

PRESSMAN, R. S. *Software Engineering: a Practitioner's Approach. 7th. ed. [S.l.]: The McGraw-Hill Companies, 2011. ISBN 0073375977/9780073375977*. [S.l.], 2011. Citado na página 18.

RAHIMIAN V.; RAMSIN, R. *Designing an agile methodology for mobile software development: A hybrid method engineering approach. In: 2008 Second International Conference on Research Challenges in Information Science. [S.l.: s.n.], 2008. p. 337–342. ISSN 2151-1349*. [S.l.], 2008. Citado na página 18.

ROUDART., M. M. E. L. *História das agriculturas no mundo*. [S.l.], 2009. Citado na página 12.

SCHWABER, J. S. K. *Um guia definitivo para o scrum: As regras do jogo*. [S.l.], 2013. Citado na página 20.

SERRANO N.; HERNANTES, J. G. G. *Mobile web apps. IEEE Software, v. 30, n. 5, p. 22–27, Sept 2013. ISSN 0740-7459*. [S.l.], 2013. Citado na página 18.

SILVA, F. C. C. J. D. *Um estudo sobre os sistemas de gerenciamento de conteúdo de código aberto*. [S.l.], 2008. Disponível em: <http://ww2.inf.ufg.br/sites/default/files/uploads/relatorios-tecnicos/RT-INF_002-08.pdf>. Citado na página 19.

STRAPI. Strapi. 2020. Disponível em: <https://strapi.io/>. Citado na página 30.

SWAGGER. Api documentation. 2021. Disponível em: <https://swagger.io/solutions/api-documentation/>. Citado na página 52.

TAMTURK, V. The ultimate guide for headless content management systems. 2016. Disponível em: <https://www.cms-connected.com/News-Archive/December-2016/The-Ultimate-Guide-for-Headless-Content-Management>. Citado na página 30.

VERSIANI, R. Como documentar uma api? 2021. Disponível em: <https://enotas.com.br/blog/documentar-uma-api/>. Citado na página 52.

ÁGIL, D. Scrum. 2020. Disponível em: <https://www.desenvolvimentoagil.com.br/scrum>. Citado 2 vezes nas páginas 20 e 21.