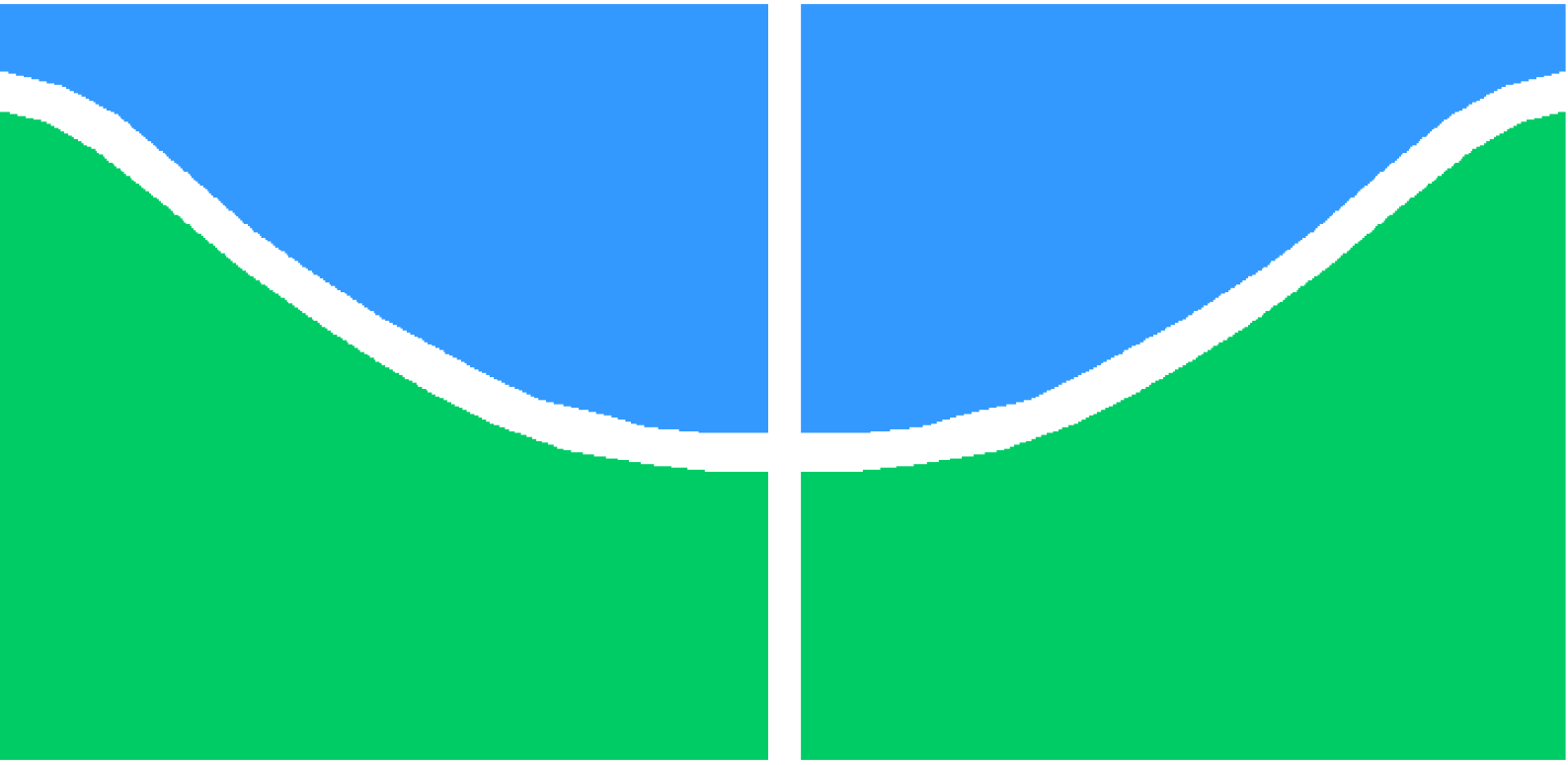


Universidade de Brasília – UnB

Faculdade UnB Gama – FGA

Engenharia de Software

Uma evolução do projeto Agromart: open source, meios de pagamento e gestão de co-agricultores

Autores: Byron Kamal Barreto Corrêa e Igor Guimarães Veludo

Orientador: Dr. André Luiz Peron Martins Lanna

Coorientador: Dr. Rudi Henri van Els

Brasília, DF

2022

Byron Kamal Barreto Corrêa e Igor Guimarães Veludo

Uma evolução do projeto Agromart: open source, meios de pagamento e gestão de co-agricultores

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

2022

Byron Kamal Barreto Corrêa e Igor Guimarães Veludo

Uma evolução do projeto Agromart: open source, meios de pagamento e gestão de co-agricultores/ Byron Kamal Barreto Corrêa e Igor Guimarães Veludo. – Brasília, DF, 2022-

59 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 , 2022.

1. desevolvimento tecnólogico. 2. desenvolvimento de software. 3. metodologias ágeis. 4. agricultura familiar. 5. csa. I. Dr. André Luiz Peron Martins Lanna. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Uma evolução do projeto Agromart: open source, meios de pagamento e gestão de co-agricultores

Byron Kamal Barreto Corrêa e Igor Guimarães Veludo

Uma evolução do projeto Agromart: open source, meios de pagamento e gestão de co-agricultores

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, 09 de Maio de 2022 – Data da aprovação do trabalho:

Dr. André Luiz Peron Martins Lanna
Orientador

Dr. Rudi Henri van Els
Coorientador

Profa. Me. Cristiane Soares Ramos
Convidado 1

Prof. Dr. Ricardo Ajax Dias Kosloski
Convidado 2

Brasília, DF
2022

Agradecimentos

Eu, Byron Kamal, agradeço primeiramente a Deus por toda a força que Ele me concedeu durante a jornada de graduação. Agradeço à minha família por todo apoio, incentivo e amor que me deram nos momentos difíceis e para prosseguir com meus sonhos. Agredeço aos meus amigos que torceram por mim e me apoiaram nos meus objetivos. Agradeço aos professores e à toda comunidade acadêmica por toda paciência e dedicação desempenhados em nos tornar pessoas e profissionais melhores. Agradeço também ao meu amigo Igor, pelo tempo de universidade e pela amizade que sei que levarei para a vida.

Eu, Igor Veludo, agradeço a minha família, em especial a minha mãe o ao meu pai, por sempre acreditarem no meu potencial. Agradeço aos amigos pelo apoio, pelas conversas e todo o suporte que me deram ao longo desses anos de caminhada da graduação. Agradeço ao meu orientador, por ter cedido o seu tempo e ter acreditado no nosso potencial para continuar com esse projeto. Agradeço também ao meu amigo Byron, por ter trilhado esse caminho ao meu lado e ao longo de anos na Universidade de Brasília.

Resumo

O projeto Agromart nasceu a partir de um evento de competição de programação (*hackathon*) realizado pela Faculdade do Gama (FGA), da Universidade de Brasília (UnB) no qual, em uma das categorias, o objetivo era o desenvolvimento de um *software* que facilitasse a conexão entre agricultores e consumidores. Após o evento, o projeto tornou-se um Trabalho de Conclusão de Curso (TCC) e contou com a parceria de uma Comunidade que Sustenta a Agricultura (CSA) do Distrito Federal. As CSAs são comunidades formadas por agricultores de produtos orgânicos ou agroecológicos e co-agricultores (consumidores) que investem mensalmente em um sistema sustentável de produção. O objetivo deste trabalho foi continuar o desenvolvimento do projeto bem como da solução tecnológica legada, que engloba por um lado um sistema web para os agricultores realizarem o gerenciamento de co-agricultores e de interação por mensagens com estes e, por outro lado, um aplicativo mobile multi-plataforma para os co-agricultores receberem notificações e realizarem suas assinaturas de cestas. Para o desenvolvimento dos projetos web e mobile foram empregadas as metodologias ágeis Scrum, Kanban e XP. Foram empregas também as técnicas de entrevistas com os clientes e o estudo das regras de negócio do funcionamento de uma CSA para levantamento das novas funcionalidades.

Palavras-chave: Desenvolvimento tecnológico. Desenvolvimento de software. Metodologias ágeis. Agricultura familiar. CSA.

Abstract

The Agromart Project was born from a programming competition event hosted by Universidade de Brasília (UnB), in which one of the categories had as its goal developing a software that would assist the connection between farmers and consumers. After the event, the project became a undergraduate thesis, and counted with the partnership of a Community-Supported Agriculture (CSA) from Brasília - DF. CSA are communities constituted by organic or agroecological products farmers and co-farmers (consumers) that invest monthly in a sustainable production system. Therefore, this work's goal was to continue the development of the legacy tech solution, that includes a web system to the farmers and a multiplatform mobile app for the consumers, and to suit the project to the standard of an open source community. For this project's tech development, the used agile methodologies were Scrum, Kanban and Extreme programming. Besides that, interview techniques were used with the clients, and the study of business rules of a CSA operation for survey of new features.

Keywords: Technological development. Software development. Agile methodologies. Family farming. Community-supported agriculture.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

API	Interface de Programação de Aplicações
CMS	*Content Management System*
CSA	Comunidade que Sustenta a Agricultura
CSS	*Cascading Style Sheets*
ER	Relacionamento de Entidade
HTML5	*Hypertext Markup Language 5*
SGC	Sistema de Gerenciamento de Conteúdo
TCC	Trabalho de Conclusão de Curso
UnB-FGA	Universidade de Brasília - Campus Gama
XP	*eXtreme Programming*
GCS	Gerência de Configuração de Software

Sumário

1 Introdução

O setor econômico de um país pode ser dividido em três partes: o setor primário, secundário e terciário. O setor primário é responsável por adquirir recursos oriundos da natureza. A agricultura compõe o setor primário e é um termo que se refere tanto ao cultivo da terra quanto às técnicas utilizadas para obter os seus recursos. Um dos conceitos de agricultura é ser um conjunto de técnicas utilizadas para cultivar plantas e vegetais a fim de gerar insumos para o consumo humano, sendo tais insumos destinados ao mercado alimentício ou à outras indústrias que os utilizarão como matéria-prima para os seus produtos (AGRIQ, 2022).

Segundo o Ministério da Agricultura (AGRICULTURA, 2020), a "Agricultura Familiar é a principal responsável pela produção dos alimentos que são disponibilizados para o consumo da população brasileira" sendo composta principalmente de pequenos produtores como pescadores, aquicultores entre outros. Esse setor se destaca pela produção de grãos como feijão, milho, arroz, café, trigo, e também pela criação de gados de corte, aves, ovinos, frutas e hortaliças, entre outros alimentos.

A COVID-19 gerou impactos negativos em todos os setores da economia. Um dos setores mais afetados pelo afastamento e isolamento social foi a agricultura, principalmente os pequenos agricultores que dependem das feiras para escoamento de sua produção.

Um sistema de escoamento interessante é o praticado pelas Comunidades que Sustentam a Agricultura (CSAs), sendo que atualmente existem 35 destas no DF (BRASíLIA, 2020). Para um breve entendimento, neste modelo o agricultor deixa de vender seus produtos através de intermediários e conta com a participação das pessoas para a organização e financiamento de sua produção, deixando de serem apenas consumidores e para se tornarem co-agricultores. Ainda sobre o funcionemento das CSAs, tem-se que:

> O funcionamento da CSA é simples: por meio de uma cota fixa mensal, os co-agricultores recebem uma caixa semanal ou quinzenal de produtos agrícolas, como frutas, verduras, legumes, ovos, leite e o que mais estiver combinado com seu agricultor. Tudo de acordo com a estação e com a safra do período, respeitando os tempos da natureza e também do produtor. Agricultores recebem uma renda mais estável e segura, além de uma conexão mais próxima com sua comunidade, enquanto os co-agricultores (antigos consumidores) se beneficiam com alimentos locais frescos, saudáveis e sustentáveis, sentindo-se mais conectados à natureza (WWF-BRASIL, 2018).

1.1 História do Agromart

A idéia do Agromart surgiu inicialmente quando os idealizadores do projeto (RODRIGUES; MACEDO, 2021) se inscreveram no tema Cultivando Conexões do *hackathon* da UnB-FGA 2020. O desafio consistiu no desenvolvimento de um software no contexto da agricultura familiar com o objetivo de estabelecer uma conexão entre os agricultores e os consumidores levando em consideração o isolamento social por conta da COVID-19, além da pré-existente dificuldade dos produtores rurais do DF e entorno em listar e divulgar seus produtos para seus clientes.

Durante o evento foi desenvolvido um aplicativo inicial onde o agricultor pudesse divulgar sua loja, barraca ou ponto de venda com seus devidos produtos, preços, localização, informação de contato e descrições adicionais. Com isso o consumidor poderia visualizar as lojas mais próximas dele através de mapas e filtros, entrar em contato com o agricultor através de 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.

Após o evento, (RODRIGUES; MACEDO, 2021) tiveram contato com profissionais da área que apresentaram abordagens diferentes da implementada. Por meio de entrevistas e conversas, os autores conheceram as regras de negócio das CSA's e com isso um novo projeto foi iniciado do zero, de modo a considerar as regras de negócio e aproveitando alguns elementos do *design* anterior, acrescentar uma interface *web* para o gerenciamento por parte dos agricultores e manter o aplicativo *mobile* apenas para os consumidores. Atualmente, o projeto é coordenado por professores da UnB-FGA que dão continuidade ao projeto juntamente com estudantes.

1.2 Problema

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 com o seu consumidor.

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.

1.3 Objetivos

1.3.1 Objetivo Geral

O objetivo geral deste trabalho foi dar continuidade ao desenvolvimento do projeto Agromart, evoluindo suas soluções *mobile* e *web* existentes, e adequar o projeto ao padrão *open source*.

1.3.2 Objetivos Específicos

Para este Trabalho de Conclusão de Curso (TCC) pretendeu-se cumprir com os seguintes objetivos específicos:

Em nível de projeto:

- adequar o projeto aos padrões *open source* para que haja contribuição de desenvolvedores interessados na temática apresentada neste trabalho;
- gerar artefatos relacionados à metodologia de desenvolvimento de software, bem como às ferramentas utilizadas.

Em nível de produto de software:

- integrar com sistemas de pagamento;
- facilitar a comunição entre agricultores e co-agricultores com uso de notificações;
- gerar artefatos, como documentos de arquitetura de software e de GCS, para que novos desenvolvedores tenham fonte de informação sobre o software.

1.4 Metodologia de pesquisa

A pesquisa tecnológica é um tipo de pesquisa aplicada que visa a materialização de um produto, protótipo, processo, instalação piloto ou um estudo de viabilização desses ou até mesmo o aperfeiçoamento de produtos, processos e/ou sistemas (adaptado de (VALERIANO, 2004)). Desse modo, este trabalho classifica-se como uma pesquisa tecnológica pois objetiva o aperfeiçoamento de um produto com aplicação de conhecimentos já existentes.

Os dados coletados que ajudaram no levantamento dos requisitos para o desenvolvimento da solução de software foram obtidos através de entrevistas e reuniões realizadas com os responsáveis pela CSA parceira ao projeto Agromart.

1.5 Organização do trabalho

O desenvolvimento do trabalho está organizado da seguinte forma:

- No Capítulo 3 são abordados aspectos relacioanados ao gerenciamento do projeto. São apresentados conceitos e características em relação à metodologia de desenvolvimento adotada e as ferramenteas de comunicação e gestão de arquivos utilizadas nesse projeto.
- No Capítulo 4 são abordados aspectos relacioanados ao gerenciamento do produto. O desenvolvimento de um produto de software abrange o uso de repositório juntamente com suas políticas de uso, as funcionalidades que serão desenvolvidas e um roteiro de execução das tarefas.
- No Capítulo 5 são apresentadas as tecnologias que foram utilizadas na continuação do desenvolvimento do projeto.
- No Capítulo 6 são apresentados os resultados do que foi desenvolvido para evolução da solução de software.
- No Capítulo 7 são apresentadas as considerações finais sobre trabalho e sugestões de trabalhos futuros.

2 Referencial Teórico

2.1 Engenharia e desenvolvimento de software

A Engenharia de Software é uma área da engenharia cujo foco está em todos os aspectos da produção de software, indo dos estágios iniciais da especificação do sistema até sua manutenção e evolução, quando o sistema já está sendo utilizado. Ela trata da abordagem sistemática para a produção de software, analisando questões práticas de custo, prazo e confiança, assim como as necessidades dos clientes e desenvolvedores (SOMMERVILLE, 2011). Dentro dessa abordagem existem etapas e planejamentos que são importantes para o desenvolvimento de um software.

Segundo (EBERT, 2007) "o gerenciamento de produtos de software é a disciplina que governa um produto de software desde o início até o fechamento". Os artefatos gerados por esse gerenciamento suportam o planejamento, a construção e a garantia de qualidade do produto, incluindo desenvolvimento de código e controle de configuração (BERSOFF, 1984).

A gestão de projetos de software inclui um conjunto de práticas que serve de guia para um equipe trabalhar de maneira produtiva. Tal conjunto compreende métodos e ferramentas que organizam as tarefas, identificam sua sequência de execução e dependências existentes. Alguns artefatos que podem surgir dessa gestão são a metodologia de desenvolvimento e o plano de comunicação.

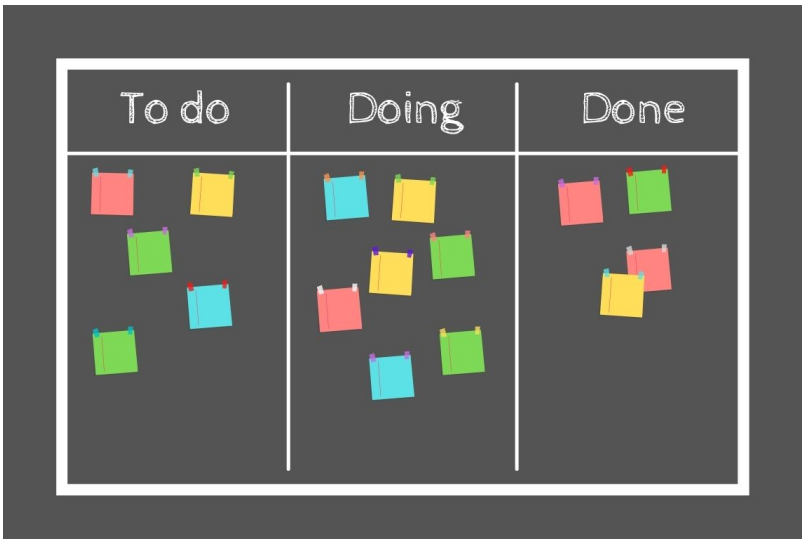
O desenvolvimento ágil de software é uma abordagem que surgiu após a publicação do Manifesto Ágil por um grupo de especialistas da área (MANIFESTO ÁGIL, 2001). As metodologias ágeis são baseadas em abordagens incrementais para a especificação, desenvolvimento e entrega do software e seu uso é comum na Engenharia de Software. Tais metodologias destinam-se a realizar entregas mais rápidas aos clientes e estes podem, em seguida, propor alterações e novos requisitos a serem incluídos nas iterações posteriores do sistema (SOMMERVILLE, 2011).

O *Scrum* é um método ágil geral cujo foco está no gerenciamento do desenvolvimento iterativo. Existem três fases sendo a primeira uma fase de planejamento em que se estabelecem os objetivos gerais do projeto e da arquitetura do software. Em seguida, ocorre uma série de ciclos de *sprints*, sendo que cada ciclo desenvolve um incremento do sistema. Finalmente, a terceira e última fase encerra o projeto, complementando a documentação exigida (SOMMERVILLE, 2011).

No método *Scrum* os projetos são divididos em ciclos chamados *sprints*, as funcionalidades e tarefas que serão desenvolvidas em um projeto são mantidas em uma lista

chamada de *product backlog* e os rituais consistem das reuniões e formalidades do método. No início de cada *sprint* acontece o ritual de *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 realizadas e implementadas durante a próxima *sprint*. As tarefas escolhidas são transferidas do *product backlog* para o *sprint backlog*, que é a lista de tarefas a serem realizadas na *sprint*. As *dailys* são reuniões curtas para alinhar o que já foi concluído de um dia para o outro e identificar pontos que estão dificultando a conclusão ou continuação de uma tarefa, caso haja. Ao fim de uma *sprint* a conclusão das tarefas é apresentada na *sprint review meeting* e então a equipe parte para o planejamento da próxima *sprint* (AGIL, 2014).

O *Extreme Programming* (XP) é uma metodologia que tem como objetivo principal levar ao extremo boas práticas de programação e de desenvolvimento de software no geral (EXTREME PROGRAMMING, 2013). É uma metodologia que pode facilmente ser adotada por diferentes níveis de desenvolvedores (experientes ou não) e em qualquer tamanho de equipe, sendo excelente para cenários em que é necessário adaptar-se a requisitos que podem mudar rapidamente (GUEDES, 2020). Existe um conjunto de princípios que devem ser seguidos por equipes que adotem o XP em projetos, são eles: *feedback* rápido, presumir simplicidade, abraçar mudanças, trabalho de alta qualidade, pequenos passos, melhoria e reflexão (GUEDES, 2020).



Fonte: Revista Exame, 2021

Figura 1 – Quadro Kanban

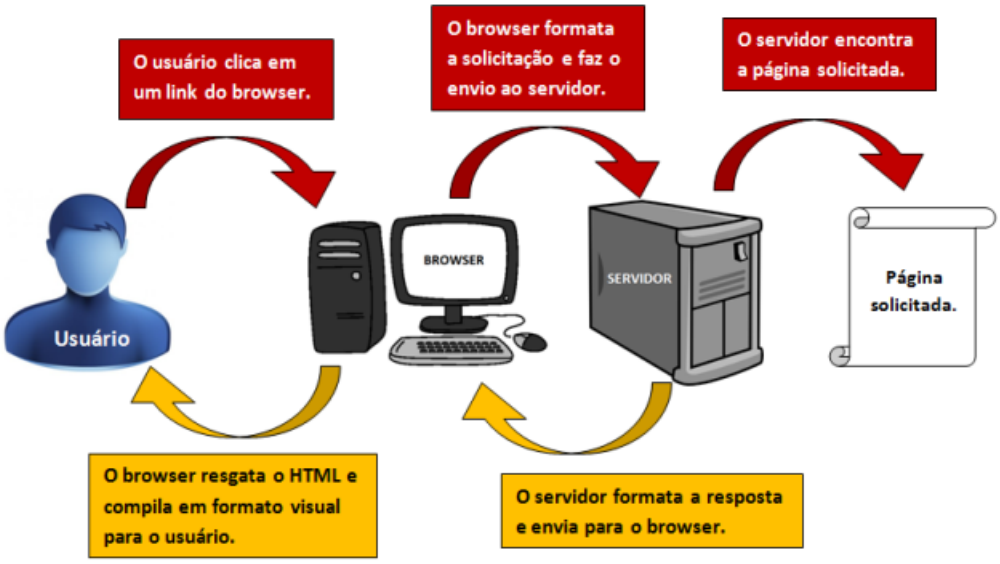
Embora não seja uma metodologia criada a partir do Manifesto Ágil, o método *Kanban* foi criado pela Toyota na década de 1960 para ajudar a empresa a lidar com problemas de estoque. É um sistema que utiliza cartões de cores ou tamanhos diferentes para designar e especificar tarefas. Dessa forma, se aprimora a administração a partir de cartões de sinalização para controle de fluxos. Assim se sabe quais tarefas precisam

ser feitas, quais estão atualmente sendo realizadas e aquelas que foram concluídas, como exemplificado na Figura 1.

O desenvolvimento de uma aplicação de software nasce da necessidade de se resolver um problema ou demanda de um cliente, empresa ou nicho de mercado, para que estes possam concretizar seus objetivos.

Aplicação *mobile* é um tipo de solução de software que é executada diretamente em dispositivos móveis como *smartphones* e *tablets* e pode ser classificada como nativa ou híbrida. Aplicações nativas são aquelas desenvolvidas especificamente para um determinado sistema operacional, como Android e iOS, de modo que utilizam recursos particulares da plataforma, além de possuírem ferramentas e linguagens de programação próprias (SERRANO; HERNANTES; GALLARDO, 2013). Já as aplicações híbridas são elaboradas com o uso de linguagens e ferramentas para o desenvolvimento web que unificam a utilização de códigos específicos. Esse tipo de aplicação realiza os seus processos como se fossem um *app* nativo, porém suas codificações são todas desenvolvidas em linguagens web, como HTML5, JavaScript e CSS (FERRONI, 2021).

Uma aplicação web é uma solução de software que é executada diretamente no *browser* (navegador) do usuário e alguns dos exemplos mais significativos são as plataformas de *e-commerce* e as redes sociais (NOLETO, 2020). Normalmente, a arquitetura utilizada por sistemas web é do tipo cliente-servidor em que há solicitações (requisições) enviadas pelo cliente (navegador) para o servidor. Este, por sua vez, processa as requisições e em seguida devolve algo como resposta para o cliente. O navegador permite ao usuário solicitar algum recurso e quando o servidor responde à solicitação são apresentados recursos como páginas HTML, figuras e documentos PDF (PALMEIRA, 2012), como representado pela figura 2.



Fonte: (PALMEIRA, 2012)

Figura 2 – Representação de solicitação e resposta do cliente.

Para que os usuários ou empresas possam alcançar seus objetivos, estas lançam mão do uso de meio de pagamentos integrados às suas aplicações. Uma maneira para realizar tal terefa é através do uso de *Application Programming Interfaces* (APIs) (em português, Interfaces de Programação de Aplicativos).

APIs referem-se a rotinas, instruções e padrões de programação estabelecidos por um software. Tais procedimentos formam uma interface de acesso a um aplicativo ou plataforma online, ou seja, eles disponibilizam suas funcionalidades a aplicativos de terceiros (ADDE, 2021). Uma *API* de pagamento online é utilizada para facilitar a comunicação entre uma loja virtual e um *gateway* de pagamento (por exemplo, PagSeguro ou Mercado Pago), isto é, uma ferramenta que possibilita a transmissão de dados entre os clientes, comerciantes e os bancos durante a realização das transações financeiras de um *e-commerce*.

2.2 Open source

Considerando-se a produção de software, esta pode ser dividida em dois tipos: software de conteúdo privado (comerciais) ou softwares de caráter livre (*open source*). Projetos *open source* divergem dos tipos comerciais ao tornar o código fonte disponível para quem quiser contribuir de modo que o conhecimento e a motivação bastam para que essas pessoas possam desenvolver em conjunto (HIPPEL; KROGH, 2003).

Iniciado em Fevereiro de 1998 por um grupo de especialistas na área de programação, o movimento *Open source* (código aberto) nasceu com o intuito de mudar a forma de distribuição de um programa e seu código, tornando-os aberto ao público. Esse movimento

é o oposto dos softwares proprietários, os quais possuem licenças exclusivas para quem o produziu. No *open source*, as pessoas podem ver o código, modificá-lo e distribuí-lo conforme suas necessidades. (REDHAT, 2019)

2.2.1 Comunidade open source

Projetos com caráter de código aberto funcionam de modo colaborativo e contam com a revisão e a produção pela comunidade. Normalmente, o código fonte é disponibilizado em algum repositório, como por exemplo o GitHub, onde desenvolvedores contribuem de maneira descentralizada.

Uma comunidade dentro de um projeto *open source* é formada por *users*, *contributors* e *maintainers*:

- *users*: usuários são pessoas que utilizarão o software e que podem relatar *bugs*, sugerir melhorias e novas funcionalidades;
- *contributors*: contribuidores são usuários que trazem discussões mais profundas e úteis para a organização e que alteram diretamente o código-fonte,
- *maintainers*: mantenedores são administradores responsáveis por manter o bom andamento do projeto e por mantê-lo "vivo", eles são membros efetivos da organização.

Além dos papéis desempenhados dentro de um comunidade *open source*, é comum a existência de documentos que orientem a contribuição e a participação no projeto. Tais documentos são;

- *README*: serve para registrar as informações a respeito do que se trata o projeto, quais são as tecnologias utilizadas, como executar o projeto na máquina local, dentre outros;
- *CONTRIBUTING*: apresenta informações relevantes para que contribuidores possam colaborar com o projeto,
- *CODE OF CONDUCT*: estabelece expectativas de comportamento dos participantes do projeto. O código de conduta pode ajudar a criar uma interação social positiva para a comunidade.

3 Gerenciamento do Projeto

Neste capítulo são apresentadas as metodologias e as ferramentas de comunicação utilizadas pela equipe para o gerenciamento do projeto Agromart.

3.1 Metodologia de desenvolvimento

Este trabalho utilizou uma abordagem ágil para dar continuidade ao projeto Agromart. Metodologias ágeis têm a preocupação em gastar menos tempo com documentação e colocar foco maior na implementação, tornando o processo de criação de software mais dinâmico. Outra característica de tais metodologias é a capacidade de adaptação às adversidades encontradas ao longo do projeto (SOARES, 2004).

Para a continuação do projeto foi utilizada uma combinação de elementos das metodologias ágeis apresentadas no Capítulo2, formando assim uma metodologia híbrida representada na Figura 3.

Scrum

Os conceitos utilizados foram:

- Sprints: com duração de 2 semanas, as *sprints* foram usadas para que não ficasse em aberto o que fazer e quando fazer, fazendo com que a progressão da aplicação tornasse iterativa.
- Sprint Planning: foi usado para que houvesse a priorização dos itens que seriam implementados e controle da quantidade de trabalho.
- Sprint Review: foi usada para avaliar a produtividade e qualidade de modo que erros fossem mitigados e decisões pudessem ser acertadas.
- Product backlog: foi utilizado para o controle das tarefas que precisavam ser desenvolvidas para se obter o produto final.
- Sprint backlog: foi utilizado para controlar os trabalhos que precisavam ser desenvolvidos na *sprint*.

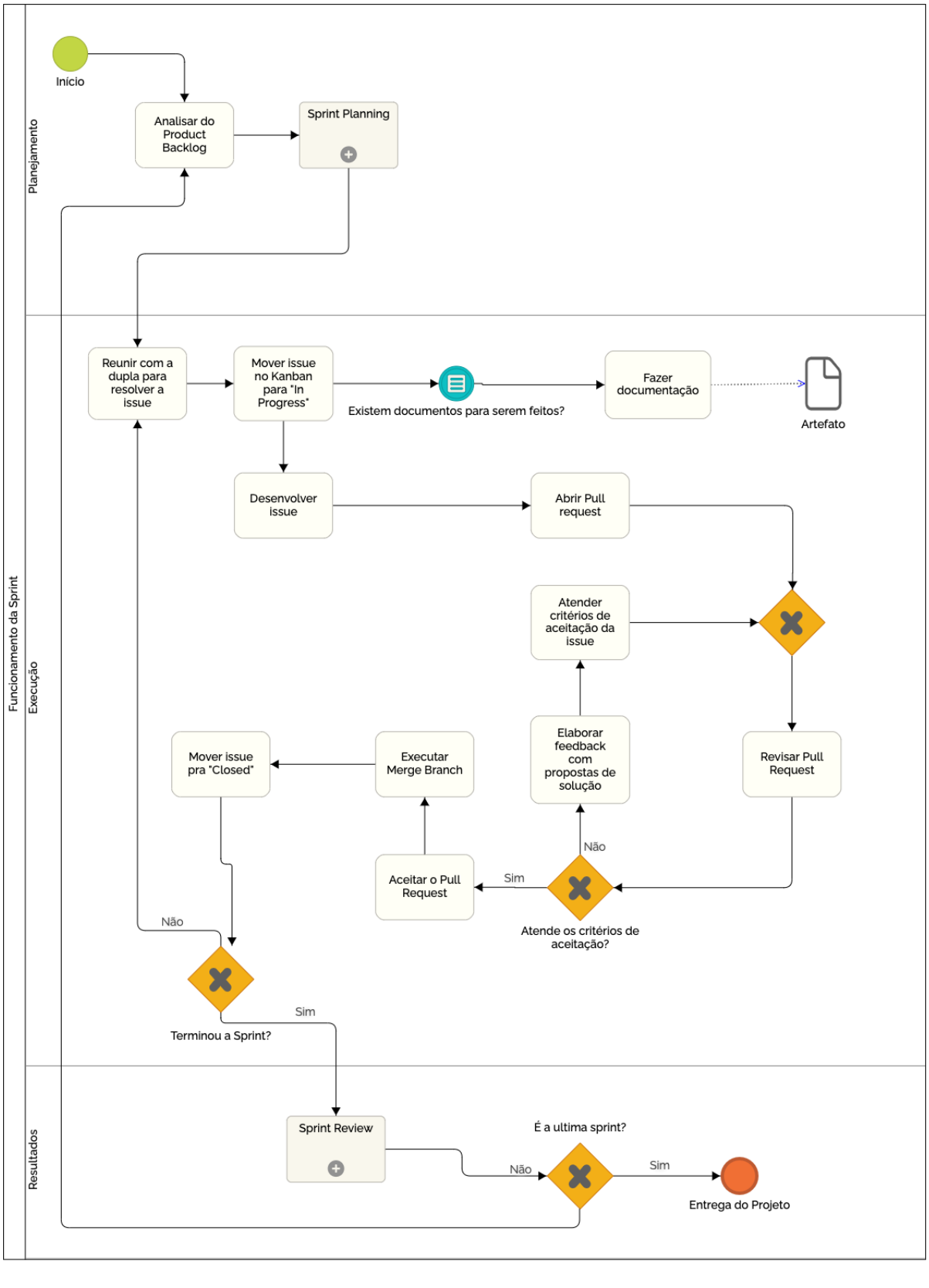
Kanban

Foi adotado o Kanban pois os elementos visuais ajudam no acompanhamento das tarefas e a manter o fluxo de trabalho. Para a implementação do Kanban foi utilizada a ferramenta Zenhub devido à facilidade e integração com GitHub.

Extreme Programming - XP

No XP, foram seguidos os 5 valores fundamentais, pois acredita-se que tais valores fazem a equipe crescer como um todo. Com relação às práticas, foram utilizadas apenas aquelas que foram necessárias no projeto, como, por exemplo:

- Jogo do planejamento: para que fosse possível estimar o trabalho e esforço do que foi desenvolvido nas *sprints*.
- Programação pareada: implementação em conjunto para que em caso de dúvidas ou dificuldades o par pudesse se ajudar e para que houvesse transmissão de conhecimento.
- Padronização de código: serviu para que o código pudesse ser mantido por qualquer membro da equipe.



Fonte: Autores, 2021

Figura 3 – Fluxo da metodologia híbrida do projeto

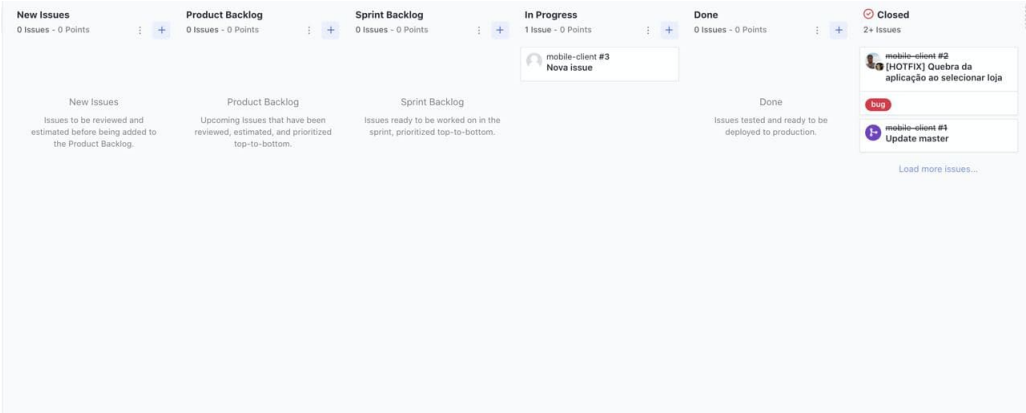
3.2 Ferramentas de Comunicação

3.2.1 Discord

Discord é um canal de comunicação de texto e voz baseados em servidores públicos ou privados. Dentro do servidor é possível criar canais de texto, voz, compartilhar telas, integração com *bots*, entre outras funcionalidades (DISCORD, 2021). Com isso, o Discord tornou-se um ambiente propício para o desenvolvimento e prática do *pair programming*.

3.2.2 Zenhub

Zenhub é uma ferramenta de Kanban para gerenciamento de projetos ágeis. Por meio de sua extensão no *Google Chrome* foi possível integrá-lo ao repositório do Agromart no GitHub, criando um Kanban personalizado para o projeto. A ferramenta conta com várias *features* interessantes, como Burndown e Velocity.



Fonte: Autores, 2021

Figura 4 – Zenhub Agromart

3.2.3 Google Drive

Google Drive é a plataforma de armazenamento em nuvem gratuita do Google. Esta ferramenta permite que o usuário guarde praticamente qualquer tipo de arquivo nos servidores e acesse remotamente em qualquer computador ou *smartphone* com conexão com a internet. Os arquivos armazenados no Google Drive, podem ser compartilhados com outros usuários e colaboradores através da conta do Google.

4 Gerenciamento do Produto

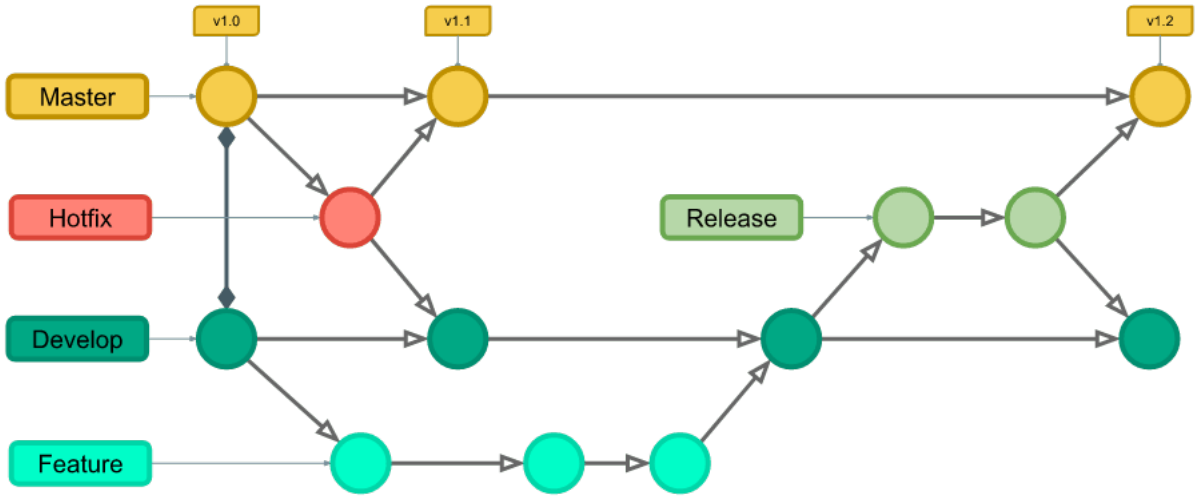
4.1 Gerência de Configuração de Software - GCS

4.1.1 Repositório

Como o projeto em desenvolvimento é de caráter livre, *open source*, o seu código fonte está hospedado no GitHub. Nesse ambiente de versionamento é possível entrar em contato com os desenvolvedores, abrir *issues* e também contribuir com o projeto, tudo de maneira simples e eficiente. Pelo GitHub é possível conectar-se à outras ferramentas que auxiliam na produtividade (como o Zenhub) e até mesmo com ferramentas de *deploy* como o Travis CI, auxiliando o desenvolvimento contínuo de software.

4.1.2 Política de Branch

Este tópico aborda o funcionamento da política de *branches* que foi definida para ser utilizada e dar continuidade ao projeto Agromart. Uma vez definida, todos os participantes do projeto seguiram esse fluxo para que se pudesse minimizar impactos negativos na fase de desenvolvimento e garantir que o projeto fosse desenvolvido de forma organizada e controlada, assegurando a integridade do código desenvolvido e disponibilizado em ambiente de produção. O *Gitflow* foi criado em 2010 e é considerado um ótimo modelo de *branching*. É um modelo fortemente baseado em *branches*, mas focados em entregas de projetos. Ele define os papéis de cada *branch* e como elas devem interagir. Apesar dele ser mais complexo quando comparado a outros *workflows*, ele disponibiliza um *framework* robusto para gerenciar projetos mais complexos. A Figura 5 detalha como está definida a política de *branches* dos repositórios do Agromart.



Fonte: Alura, 2020

Figura 5 – Fluxo de trabalho - Gitflow

A branch *master* foi definida como a *branch* estável do projeto, sendo ela proveniente da *develop* por meio de aprovação de *pull request*. Nenhum membro está autorizado a fazer *commits* diretamente na *master*.

A *branch* de *hotfix* é criada a partir da *master* sempre que houver algum *bug* em produção. Ao se resolver o *bug*, um *pull request* é aberto para *master* e para *develop*.

As *branches* de *feature* são *branches* para o desenvolvimento de novas funcionalidades. Essas *branches* são criadas sempre a partir da *branch develop* e uma vez finalizada, é feito um *pull request* para *develop*.

4.1.3 Política de Commits

Os *commits* devem ser atômicos e os comentários devem descrevê-los de forma sucinta. O texto deve descrever o que foi produzido de forma resumida e em português. Caso o *commit* não seja destinado à conclusão de funcionalidade ou documento, deve ser iniciado com o verbo no gerúndio, mas se o *commit* é destinado à conclusão de uma funcionalidade, documento ou correção, deve ser iniciado com o verbo no particípio. Além disso, deve conter o número da *issue* correspondente no seguinte formato:

`<#id da issue> <Texto começando com letra maiúscula, verbo no gerúndio ou particípio>`

Exemplo de *commit* destinado à conclusão:

`#1 Finalizada funcionalidade de cadastro de usuário`

Exemplo de *commit* não destinado à conclusão:

`#1 Corrigindo erros do documento de arquitetura`

4.1.4 Política de Issues

As *issues* são criadas com o objetivo de mapear e descrever o trabalho a ser desenvolvido, possibilitando controle e transparência do que está sendo feito. Com isso, é possível manter a rastreabilidade de tudo que foi planejado e executado.

As issues contém identificadores para que se possa indicar sua natureza. Os identificadores definidos para o projeto são:

- *[EPIC]* – utilizado para as *issues* que representam épicos.
- *[US]* – utilizado para as *issues* que representam histórias de usuário.
- *[TS]* – utilizado para as *issues* que representam histórias técnicas.

  O formato padrão de nomenclatura para essas issues é:

  `[<Identificador>] <Título com descrição sucinta>`

  Exemplo:

  `[US01] Prototipação das telas de cadastro`

- *[REFACTOR]* – utilizado para *issues* que representam refatoração.
- *[BUG]* – utilizado para *issues* que representam correções de *bugs*.
- *[DOC]* – utilizado para as *issues* que representam tarefas de documentação.
- *[TRAINNING]* – utilizado para *issues* que representam atividades de estudo e treinamento.
- *[QUESTION]* – utilizado para *issues* que representam perguntas que a comunidade deseja fazer aos mantenedores.
- *[SUGGESTION]* – utilizado para *issues* que representam sugestões que a comunidade deseja fazer aos mantenedores.

  O formato padrão de nomenclatura para essas *issues* é:

  `[<Identificador>] <Título com descrição sucinta>`

  Exemplo:

  `[BUG] Duplicação no Banco`

4.2 Backlog

O *backlog* de produto deste projeto utilizou conceitos de histórias de usuário, que são utilizadas em metodologias ágeis, porém não froam seguidas à risca as convenções de escrita das mesmas. O conceito aplicado refere-se à descrição das tarefas de *backlog* que são curtas, simples e de fácil entendimento. Convém ressaltar que os itens de *backlog* descritos abaixo referem-se aos itens desenvolvidos e implementados no escopo desse trabalho. Contudo, considerando-se que o Agromart é um projeto em desenvolvimento, os *backlogs* apresentados continuarão a serem desenvolvidos com a inclusão de novos itens em futuras evoluções do projeto.

4.2.1 Backlog do Agricultor

O *backlog* do agricultor representou as funcionalidades que foram implementadas para evolução da interface web que será utilizada pelo agricultor familiar. Tais funcionalidades estão descritas a seguir.

- acesso à plataforma web – os agricultores poderão realizar o *login* na plataforma web.
- enviar notificações para os co-agricultores – os agricultores podem enviar notificações para os co-agricultores sobre produtos e/ou informações relevantes sobre a CSA.
- visualizar uma lista com os co-agricultores integrantes da CSA – os agricultores poderão ver as informações dos integrantes da CSA, incluindo o tipo de plano e o *status* do pagamento.

4.2.2 Backlog do Co-agricultor

O *backlog* do agricultor representa as funcionalidades que foram implementadas para evolução do aplicativo *mobile* que será utilizado pelos co-agricultores, com o propósito de encontrar seus produtos e realizar as assinaturas dos planos dentro das regras de negócio do software.

- receber notificações dos agricultores – os co-agricultores poderão receber um *push notification* com as informações.
- visualizar central de notificações dos agricultores – os co-agricultores poderão visualizar um histórico de notificações.
- pagar planos – deve ser possível realizar o pagamento dos planos pelo aplicativo.

5 Suporte Tecnológico

Esse capítulo tem como finalidade trazer mais detalhes acerca das tecnologias utilizadas no desenvolvimento da solução de software Agromart. A seguir serão apresentadas cada uma dessas tecnologias e suas utilidades para o projeto.

5.1 Linguagens de programação e frameworks

O projeto Agromart utiliza Javascript como linguagem de programação tanto na parte do cliente (aplicação *mobile* e web) quanto na parte do servidor, fazendo com que toda a aplicação seja criada utilizando a mesma linguagem. Javascript é uma linguagem de *scripting* para páginas web amplamente utilizada por ser leve, interpretada, orientada a objetos e dinâmica (MDN WEB DOCS, 2021), sendo também utilizada no lado do servidor com o Node JS. Node é uma *runtime* projetada para desenvolvimentos de aplicações escaláveis de rede (NODE.JS ORG, 2021), de modo a permitir que o usuário consiga criar aplicações web sem depender do navegador.

React é uma biblioteca Javascript voltada para a criação de interfaces web, baseada em componentes e que leva o lema "Aprenda uma vez, use em qualquer lugar" pois, além de ser possível utilizá-la na programação web, ela também pode ser usada no desenvolvimento *mobile*.

React Native é uma biblioteca baseada no React que foi criada pelo Facebook em 2015 e surgiu como uma solução para permitir o desenvolvimento multiplataforma para os sistemas iOS e Android utilizando a mesma linguagem de programação. Uma vez escrito, o código é convertido em linguagem nativa dos aparelhos *mobile* (TREINA WEB, 2021). Uma das principais características do Javascript é ser uma linguagem de tipagem dinâmica, ou seja, não é necessário declarar o valor de uma variável ou função, o que pode causar alguns problemas de tipagem de dados em aplicações mais complexas. Nesse cenário o Agromart utiliza o Typescript que é um *superset* baseado em Javascript que permite um melhor controle dos tipos de variáveis e funções sem perder as características da linguagem e ainda conta com as suas bibliotecas e *frameworks*(GEEKHUNTER, 2021).

5.2 Análise estática de código

A análise estática de código serve para realizar a verificação da qualidade do código-fonte, buscando por falhas de implementação antes da execução do software. Essas falhas podem estar relacionadas com variáveis não inicializadas, atribuição incorreta de tipo de

dado à uma variável, chamadas erradas de funções, entre outros.

O ESLint é uma ferramenta *open source* que auxilia na análise de código Javascript ou Typescript, identificando más práticas quanto ao estilo e/ou padrões dessas linguagens. O uso dessa ferramenta ajuda a manter a padronização e organização do código promovendo identidade a este e, mesmo que escrito por vários desenvolvedores, mantendo um único padrão.

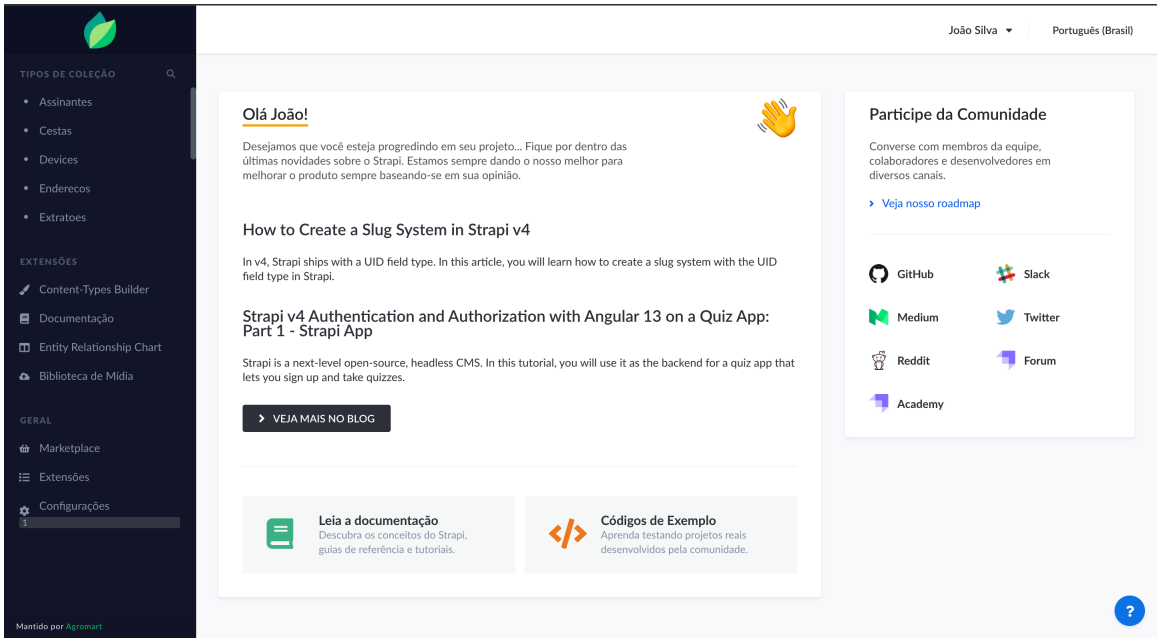
5.3 Análise da Qualidade de Código

O Code Climate fornece uma revisão de código automatizada que auxilia na manutenibilidade do software e fornece alguns parâmetros de qualidade como código duplicado, alta complexidade de funções e "maus cheiros" no código (*code smell*). A ferramenta apresenta um *feedback* visual das análises realizadas no código, facilitando assim o acompanhamento da qualidade do produto de software que será entregue e evitando futuros problemas na manutenção do projeto.

5.4 Sistema de Gestão de Conteúdo

Um sistema de gestão de conteúdo (*Content Management System - CMS*, em inglês) é uma ferramenta de software que permite criar, organizar, publicar e apagar conteúdos de maneira mais rápida quando comparado ao desenvolvimento de software tradicional.

Strapi é um CMS, *open source* escrita na linguagem Javascript, que permite a criação e manipulação de conteúdo de uma forma visual e iterativa por meio de uma aplicação web (STRAPI, 2021), como pode ser visto na Figura 6. Essa ferramenta já existia no projeto Agromart e passou a ser utilizada no lado do servidor como *back-end* e API.



Fonte: Autores, 2022

Figura 6 – Interface do CMS utilizado no projeto Agromart

5.5 Docker

O Docker é uma ferramenta de virtualização de projeto. Ele consegue criar aplicações em containers individuais utilizando uma imagem web ou uma imagem presente no próprio repositório do projeto (MICROSERVICEIT, 2021). Desse modo é possível criar um ambiente virtualizado e padronizado para todos os desenvolvedores, independente da arquitetura de seus computadores e assim manter um ambiente estável para o desenvolvimento da aplicação. O Agromart utiliza o Docker em sua etapa de construção do *backend* em ambiente de desenvolvimento, mantendo a mesma versão das tecnologias utilizadas.

5.6 Banco de Dados

Como a constução da API do projeto foi feita utilizando o Strapi, foi preciso escolher entre quatro banco de dados:

- SQLite
- MongoDB
- MySQL
- PostgreSQL

Entre os bancos listados estão bancos relacionais e não relacionais mas, para o projeto Agromart, foi retirada a possibilidade do banco não relacional devido à quantidade de relações entre as tabelas (RODRIGUES; MACEDO, 2021). O banco de dados relacional SQLite não seria uma boa escolha considerando a escalabilidade do projeto. Portanto, o banco de dados escolhido foi o PostgreSQL por ele ser um poderoso sistema de gerência de banco de dados, que possui confiabilidade, robustez e desempenho devido a recursos que protegem a integridade dos dados.

6 Resultados

Nesse capítulo serão apresentadas as funcionalidades desenvolvidas nas aplicações web, *mobile* e no *back-end*, além das adequações realizadas no projeto aos padrões *open source*. Os repositórios do projeto pode ser encontrados no GitHub através do link <https://github.com/AgroMart>

6.1 Aplicação Mobile

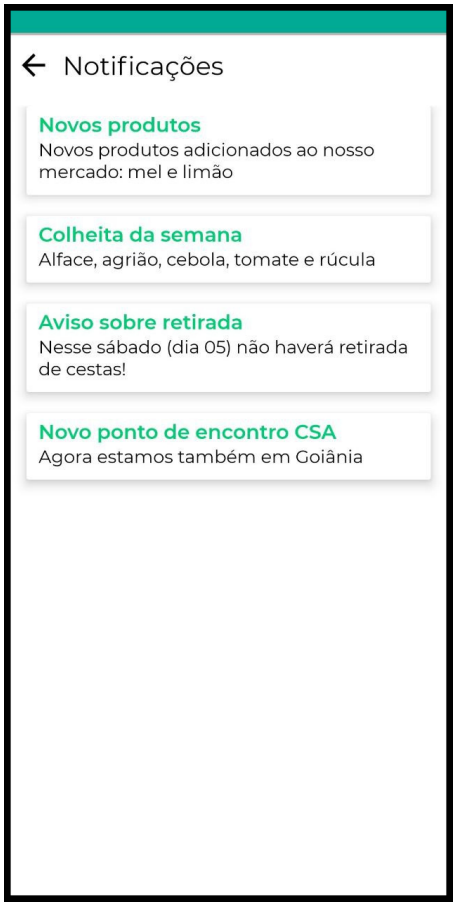
6.1.1 Notificações

Uma das principais *features* desenvolvidas para a aplicação *mobile* foi a possibilidade de enviar e visualizar as notificações sobre as colheitas ou novidades da CSA. Para acessar essa funcionalidade, ao abrir o aplicativo, o usuário deve clicar no ícone da sua conta que fica no canto inferior do aplicativo. Ao clicar uma nova página será mostrada como indicada na Figura 7.

Fonte: Autores, 2022

Figura 7 – Aplicativo: Acesso às notificações

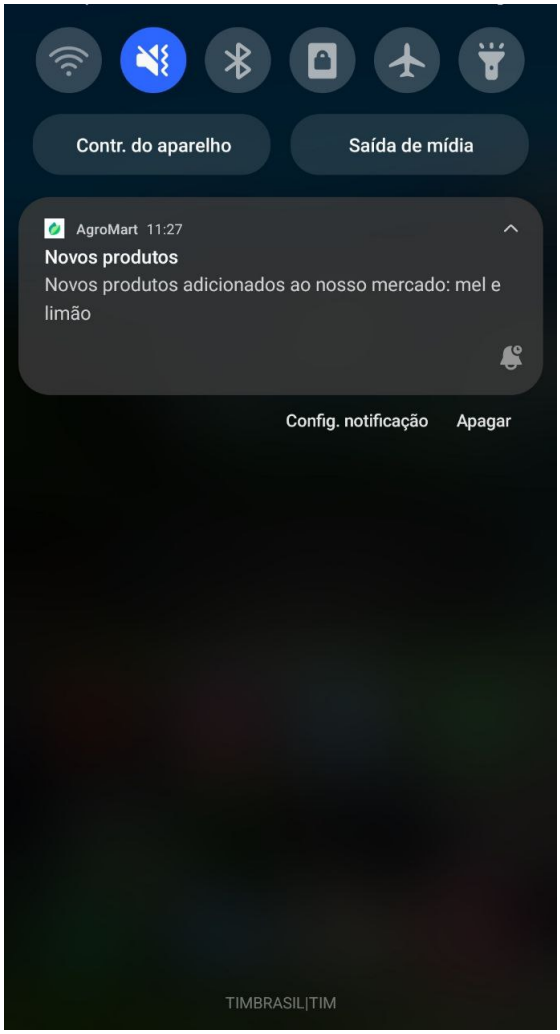
Ao clicar na opção de notificações a aplicação irá abrir um novo componente mostrando as últimas notícias da CSA para os usuários, conforme mostrado na Figura 8.



Fonte: Autores, 2022

Figura 8 – Aplicativo: Central de notificações

Sempre que houver uma nova notificação a aplicação irá avisar o usuário se o mesmo tiver ativado as notificações do Agromart. Uma mensagem com o título e o conteúdo da notificação serão mostrados na área de notificações do celular do usuário como mostrado na Figura 9.

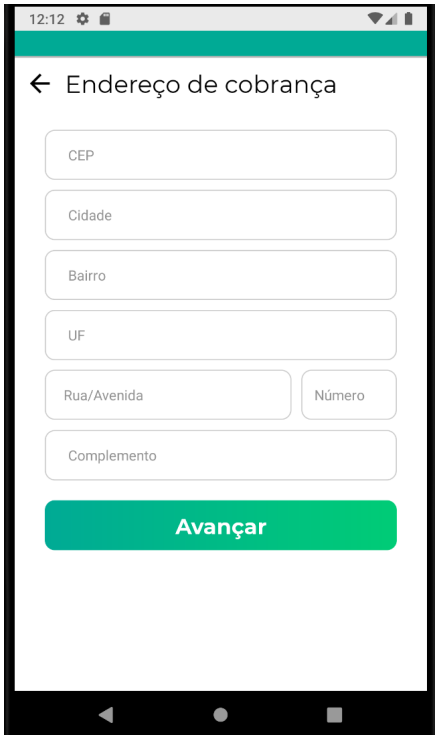


Fonte: Autores, 2022

Figura 9 – Aplicativo: Push notification

6.1.2 Pagamento

Outra *feature* que foi desenvolvida para o aplicativo foi a integração com API de pagamento para que os usuários possam pagar as mensalidades dos seus planos de forma mais rápida e em qualquer lugar. Para acessar essa funcionalidade o usuário deverá entrar na aplicação, clicar na opção *home* no canto inferior esquerdo, entrar em uma das lojas, escolher o plano, e depois clicar no botão "Realizar pagamento". Ao realizar esse procedimento a aplicação irá mostrar a tela representada na Figura 10. Nessa tela o usuário irá preencher os dados com o seu endereço. Com os dados preenchidos, ao clicar no botão "Avançar"a aplicação irá mostrar a última etapa do pagamento.



Fonte: Autores, 2022

Figura 10 – Aplicativo: Tela de Endereço de Cobrança

Com os dados de endereço preenchidos a última etapa do pagamento será a inserção dos dados do cartão de crédito do usuário para a confirmação do pagamento, conforme apresentado pela Figura 11. Vale ressaltar que as informações do cartão de crédito não podem e não são armazendas no banco de dados do projeto, sendo armazenados somente os dados provenientes da da API do Juno (o serviço de integração de meios de pagamento).

Fonte: Autores, 2022

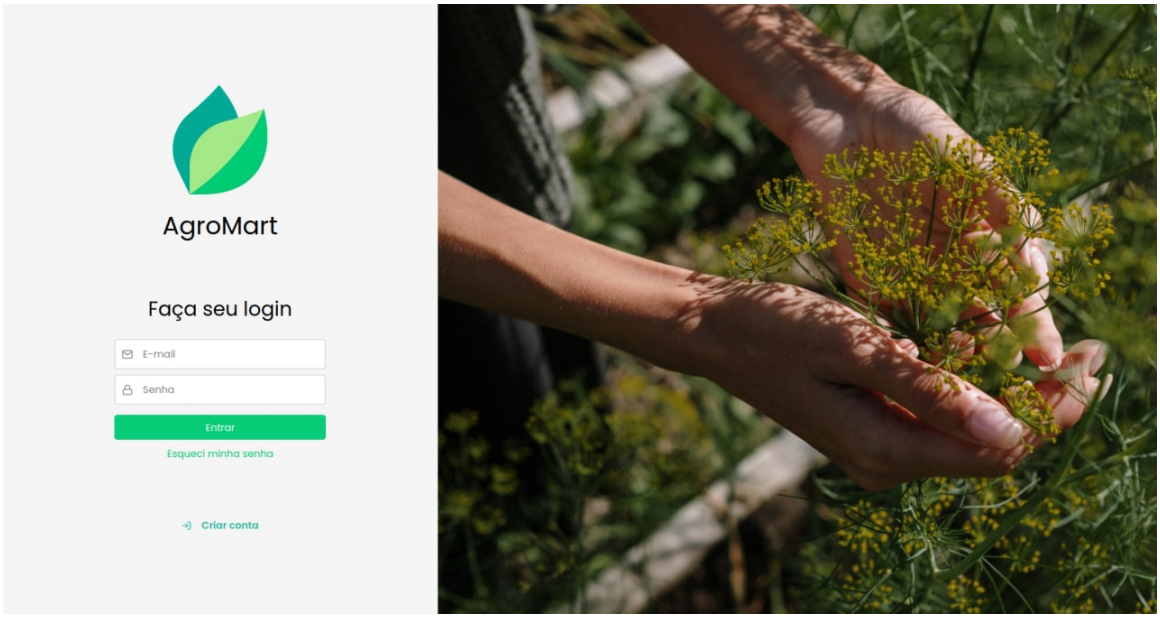
Figura 11 – Aplicativo: Tela de Cadastro de Cartão de Crédito

6.2 Aplicação Web

Com o intuito de fornecer uma ferramenta de apoio para o agricultor, foi criada uma aplicação web afim de centralizar algumas de suas atividades como, por exemplo, o envio de notificações para o aplicativo *mobile* e o cadastro de novos co-agricultores para a base de dados do projeto.

6.2.1 Login

Ao acessar o *link* da aplicação, o agricultor precisará efetuar o *login* para entrar no *site*. Para efetuar o acesso, o usuário precisará informar o seu email e a sua senha e em seguida apertar no botão "Entrar". A aplicação então irá enviar os dados via requisição para a API, para validação das informações, caso o usuário não esteja cadastrado, uma mensagem é enviada na tela pedindo para que crie uma conta. A imagem da página de *login* está apresentada na Figura 12.

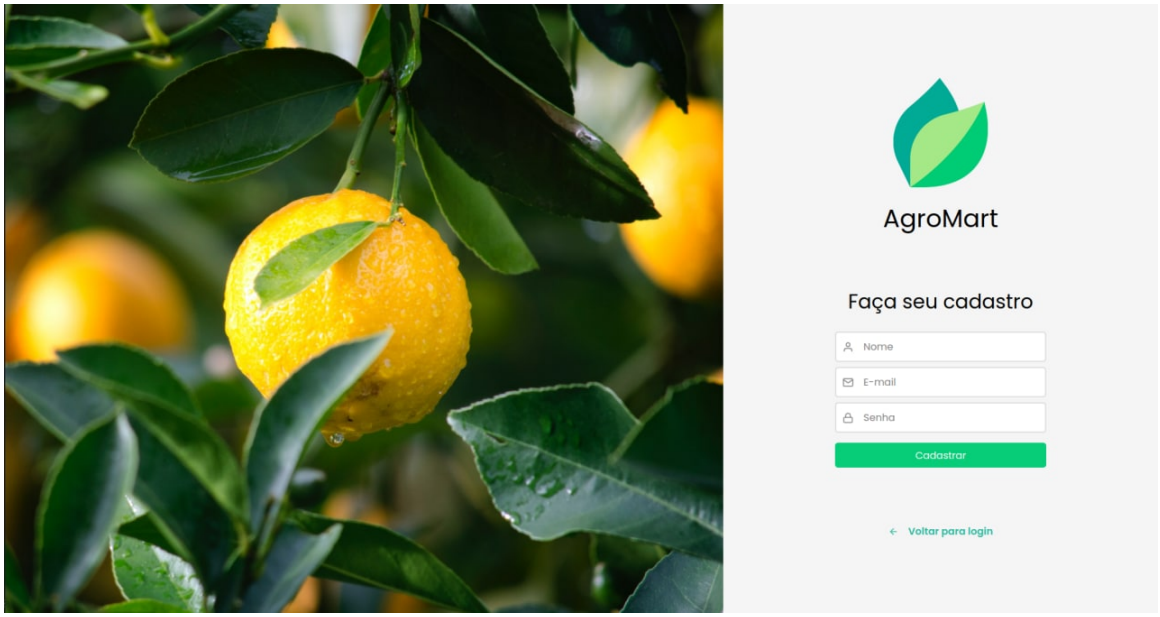


Fonte: Autores, 2022

Figura 12 – Web: Tela de Login

6.2.2 Cadastro

Caso o usuário ainda não esteja cadastrado na base ele pode ser cadastrado ao entrar na página de cadastro da aplicação web. Para realizar o cadastro será necessária a inserção de alguns dados como nome, email e criar uma senha de acesso. Ao clicar no botão "Cadastrar" a aplicação enviará os dados do cadastro para a API e uma nova conta será criada. Se algo der errado nesse fluxo, como o não preenchimento do email, ou senha ou nome, ou se a API retornar um erro ao tentar cadastrar, a própria aplicação web dará ao usuário um *feedback* sobre o ocorrido. A Figura 13 mostra a página de criação de um novo cadastro.

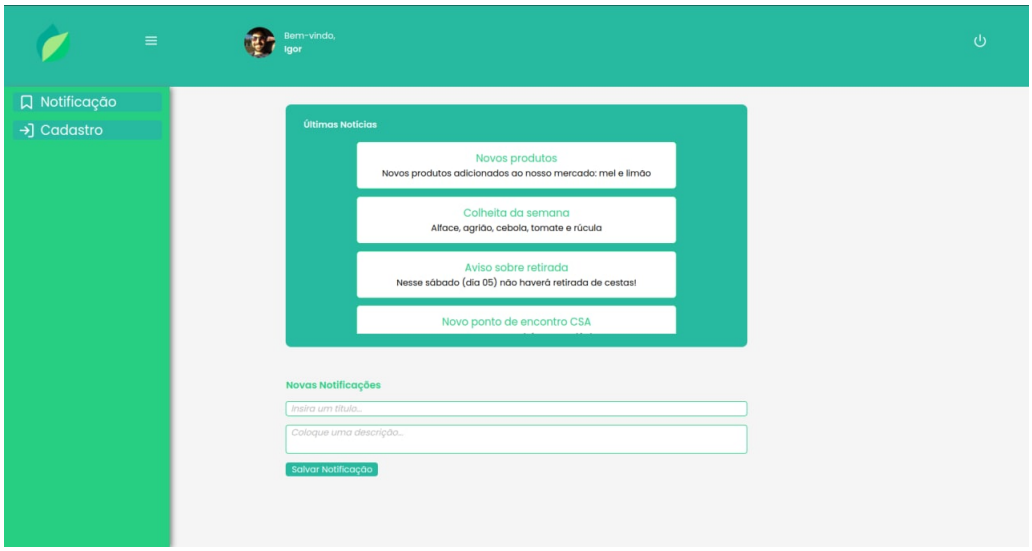


Fonte: Autores, 2022

Figura 13 – Web: Tela de cadastro

6.2.3 Notificações

Dentro da aplicação web o co-agricultor poderá visualizar as informações das CSAs e também poderá enviar uma nova notificação no campo de novas notificações. Uma vez enviada pelo portal web, as novas notificações irão aparecer no aplicativo *mobile*, informando os usuários sobre as informações mais recentes de sua CSA ou qualquer outro tipo de informação, como mostra a Figura 14.



Fonte: Autores, 2022

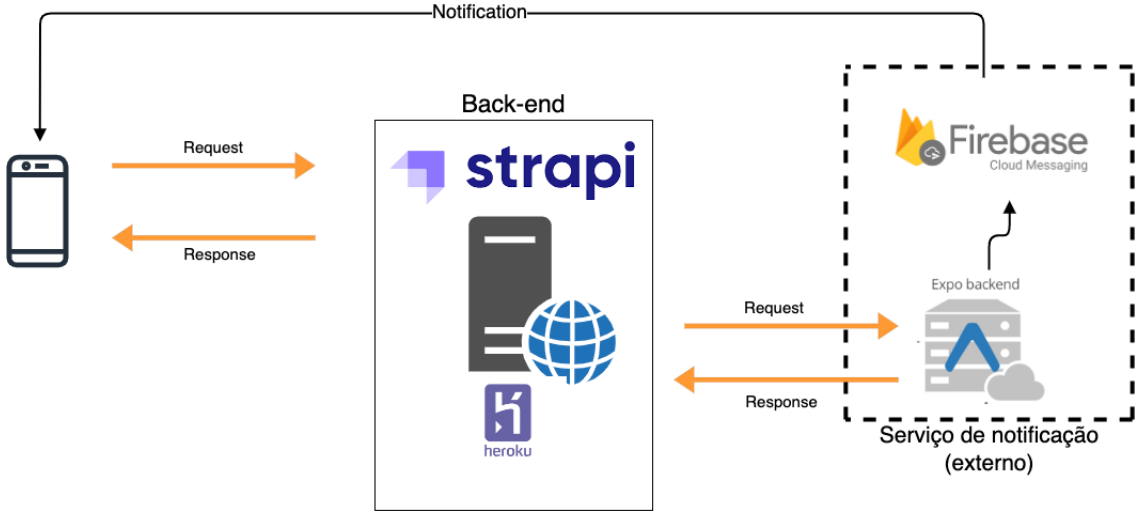
Figura 14 – Web: Tela de notificações

6.3 Back-end

Para evolução do projeto na parte de APIs, foram realizadas integrações com o meio de pagamento Juno e com os serviços de notificação disponibilizados pelo Expo e pelo Firebase Cloud Messaging.

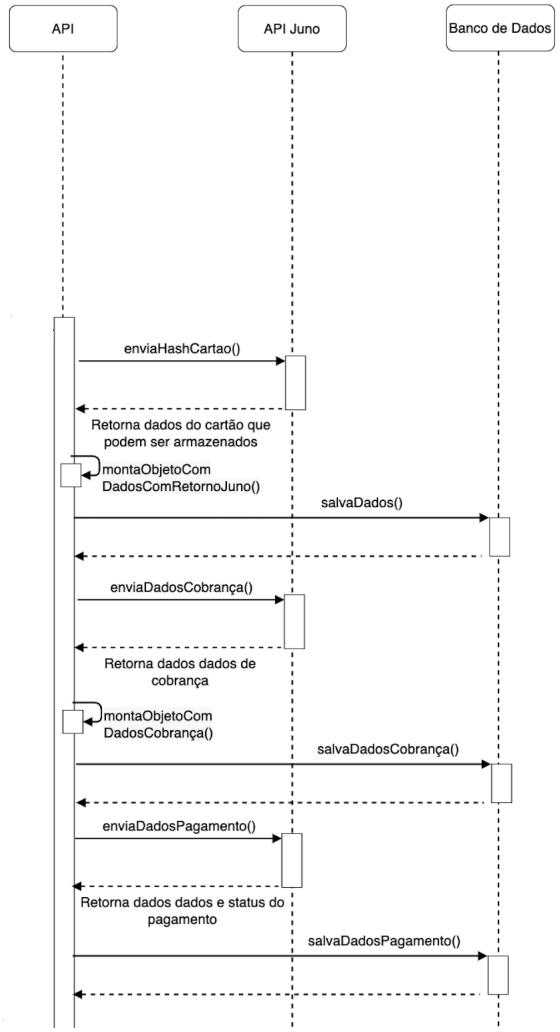
Para as notificações, o *backend* receberá da plataforma web os dados, o *backend* irá armazenar estes dados, e transmiti-los para os serviços externos de notificação, que disparam as notificações para os co-agricultores. Uma representação esquemática dessa integração é apresentada na Figura 15.

Já para o pagamento, o *backend* do projeto é responsável pela comunicação com a API do Juno. O *mobile* é responsável somente por gerar o *hash* do cartão e enviá-lo à API do Agromart. Após a chegada desse *hash* e das informações necessárias para a cobrança, o *backend* faz toda a tramitação de dados e requisições necessárias para processamento do pagamento. O fluxo de como ocorre o pagamento pode ser visto na Figura 16.



Fonte: Autores, 2022

Figura 15 – Esquema do processo de notificação



Fonte: Autores, 2022

Figura 16 – Fluxo de pagamento

6.4 Open Source

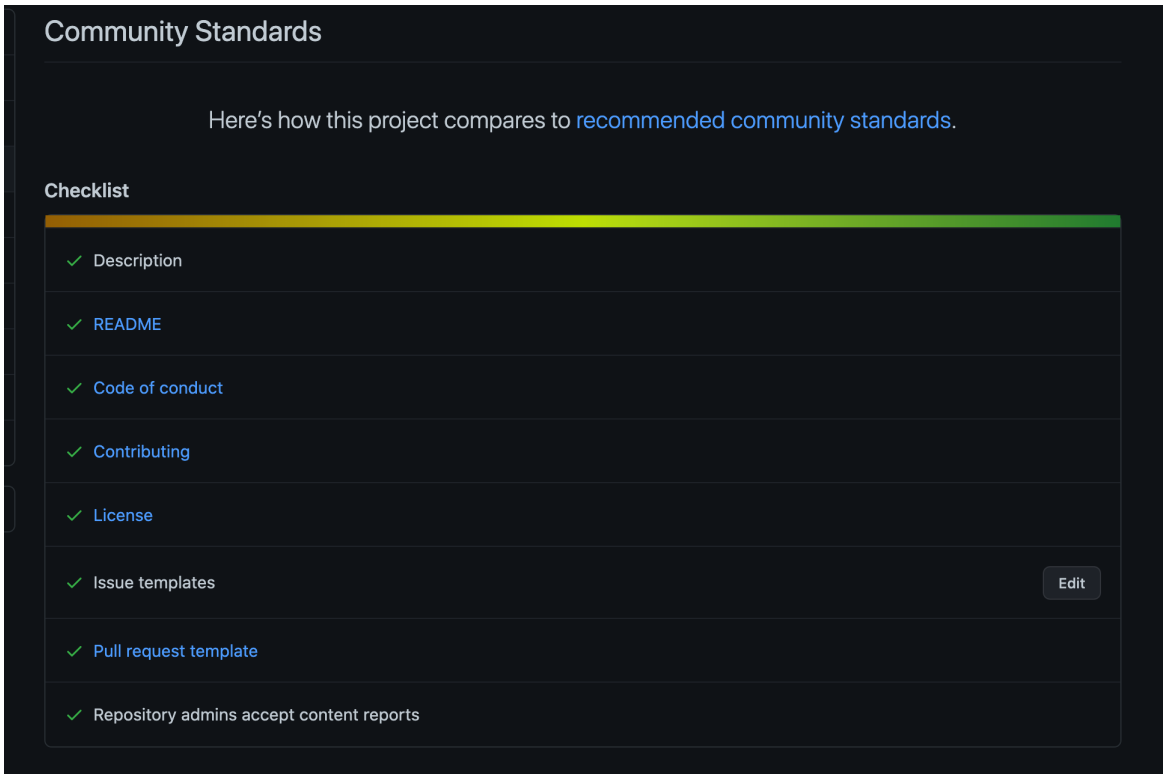
Inicialmente o código do Agromart ficava em um repositório fechado, mas com a iniciativa de torná-lo um projeto contínuo, o seu código foi migrado para um repositório público.

Cada repositório do Github apresenta uma interface visual (Figura 17) que facilita o acompanhamento dos itens que são necessários para que o projeto adeque-se aos padrões de comunidade de código aberto. Além desses itens é preciso também que o repositório esteja com a visibilidade pública. Até o momento do desenvolvimento desse trabalho o projeto Agromart conta com pessoas dos perfis de usários e mantenedores, além dos documentos citados acima como é possível ver na Figura 18.

6.4.1 Licença

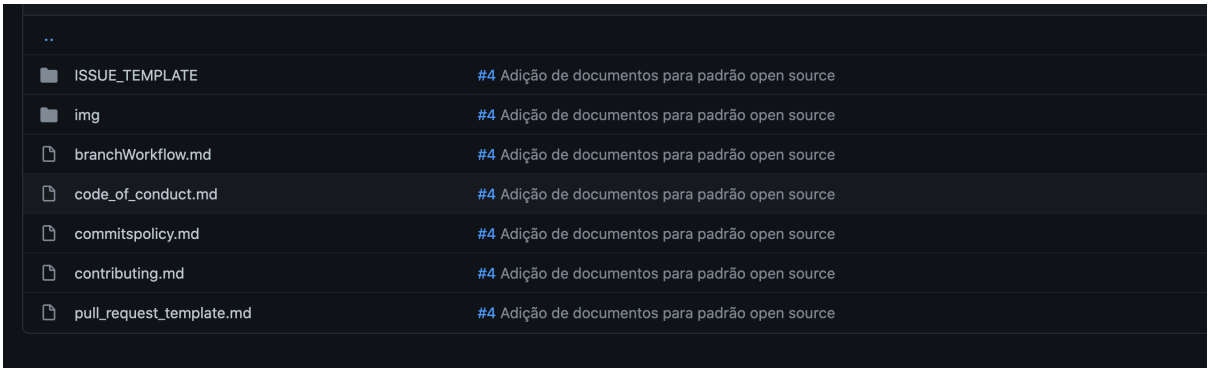
Uma licença é a autorização (ou restrição) de determinadas ações de uso, definidas por quem desenvolveu o software e com o uso do código aberto foi necessária a escolha de uma licença para o projeto.

Dentre as categorias existentes, a licença escolhida foi a GLP que é uma licença recíproca total, onde o projeto será sempre um projeto livre e gratuito. Essa licença garante que toda cópia, execução, modificação e redistribuição do projeto deve continuar com o caráter livre.



Fonte: Autores, 2022

Figura 17 – Inteface visual de acompanhamento do padrão de comunidade open source



Fonte: Autores, 2022

Figura 18 – Documentos para padrão open source - Agromart

7 Considerações Finais

As Comunidades que Sustentam a Agricultura tem desempenhado um papel importante no escomento da produção de agricultores familiares, eliminando intermediários e garantindo uma maior geração de renda. Diante desta temática, este trabalho teve por objetivo a continuação do desenvolvimento do sistema de sofware legado do projeto Agromart, visando auxiliar o agricultor no escoamento e comunição com os co-agricultores e ajudar pessoas que buscam uma alimentação mais saudável a encontrar seus produtos.

O desenvolvimento de novas funcionalidades na solução tecnológica existente visam trazer mais ferrementas para que as partes envolvidas possam usufruir melhor do contexto de uma CSA. A integração com meios de pagamento trazem mais agilidade no pagamento dos planos. As notificações melhoraram a comunição entre as partes.

Adequar os repositórios para o padrão de uma comunidade *open source*, trouxe uma maior maturidade ao projeto. Agora, desenvolvedores e/ou pessoas interessadas na temática abordada nesse trabalho podem contribuir e tornar o projeto mais bem-sucedido e para tal, existe a documentação de como contribuir.

Lidamos também com vários obstáculos ao longo do desenvolvimento do trabalho, o primeiro deles foi entender toda uma estrutura do código legado deixada pelos antecessores desse projeto. Com a criação da estrutura inicial do projeto, tivemos que ir atrás de um novo conhecimento afim de entender e consequentemente criar novas funcionalidades para a aplicação. Estudos sobre o CMS e como o *Strapi* funciona tornou este primeiro obstáculo bem difícil de ser vencido, uma vez que nunca havíamos trabalhado com um sistema gerenciador de conteúdo. Houveram dificuldades também em decidir quais novas *features* deveriam ser focadas no desenvolvimento deste trabalho, uma vez que foram levantados diversos novos requisitos junto aos clientes.

Outro grande obstáculo encontrado foi a criação da lógica do meio de pagamento utilizando o *Strapi* como API, pois o mesmo se mostrou bem deficiente em alguns aspectos. O primeiro ponto fraco do *Strapi* é não possuir uma documentação tão clara para criação de novas rotas e serviços, se mostrou também uma ferramenta muito complicada de se ter acesso aos métodos de criação, inserção e consulta no banco de dados, mais uma vez por ter uma documentação não tão clara nesse sentido, e por último a sua arquitetura se mostrou muito limitada para aplicações que irão escalar em quantidade de usuários.

Dentro das funcionalidades que deveriam ser desenvolvidas, apenas a "Visualizar uma lista com os co-agricultores integrantes da CSA"não foi completamente implementada, mas continuará a ser feita após o final deste trabalho.

De forma geral, o projeto foi muito enriquecedor e o resultado alcançado foi muito satisfatório. Ao final foram criadas novas *features* bem relevantes que ajudarão o projeto Agromart a crescer ainda mais. Diante disso, recomendamos como trabalhos futuros:

- dar continuidade à aplicação web que foi iniciada neste projeto, focando-se em gerar informações que sejam úteis para o agricultor a respeito dos co-agricultores integrantes da CSA em que ele é responsável;
- iniciar uma nova estrutura para substituir o Strapi como API, onde seja mais fácil criar rotas, *controllers, middlewares* e torne a escalabilidade mais viável;
- criação de teste unitários e de integração;
- validar as funcionalidades juntamente aos usuários, recebendo *feedbacks* para melhoria dos softwares.

Mesmo após o fim deste TCC, iremos continuar atuando e contribuindo com projeto, pois acreditamos em seu potencial.

Referências

ADDE, T. **O que é API de pagamento e por que integrá-la ao seu e-commerce?** 2021. Acessado em 21 de Outubro de 2021. Disponível em: <"https://www.nuvemshop.com.br/blog/api-de-pagamento/">. Citado na página 20.

AGIL, D. **SCRUM**. 2014. Acessado em 13 de Outubro de 2021. Disponível em: <"http://www.desenvolvimentoagil.com.br/scrum">. Citado na página 18.

AGRICULTURA, M. da. **Agricultura Familiar**. 2020. Acessado em 09 de Abril de 2022. Disponível em: <"https://www.gov.br/agricultura/pt-br/assuntos/agricultura-familiar/agricultura-familiar-1">. Citado na página 13.

AGRIQ. **Agricultura: tipos, práticas mais comuns e outras curiosidades**. 2022. Acessado em 09 de Abril de 2022. Disponível em: <"https://agriq.com.br/agricultura/">. Citado na página 13.

BERSOFF, E. H. Elements of software configuration management. **IEEE Transactions on Software Engineering**, SE-10, n. 1, p. 79–87, 1984. Citado na página 17.

BRASíLIA, R. C. **Comunidades**. 2020. Acessado em 09 de Abril de 2022. Disponível em: <"https://csabrasilia.wordpress.com/csabrasilia/comunidades/">. Citado na página 13.

DISCORD. **Sobre Discord**. 2021. Disponível em: <"https://discord.com/company">. Citado na página 25.

EBERT, C. The impacts of software product management. **Journal of Systems and Software**, v. 80, n. 6, p. 850–861, 2007. Citado na página 17.

EXTREME PROGRAMMING. **Extreme Programming: A gentle introduction**. 2013. Acessado em 13 de Outubro de 2021. Disponível em: <"http://www.extremeprogramming.org/">. Citado na página 18.

FERRONI, C. **Híbrido ou nativo: qual desenvolvimento escolher?** 2021. Acessado em 11 de Outubro de 2021. Disponível em: <"https://blog.tecnospeed.com.br/aplicativo-hibrido-ou-nativo">. Citado na página 19.

GEEKHUNTER. **Introdução a Typescript.** 2021. Acessado em 09 de Outubro de 2021. Disponível em: <"https://blog.geekhunter.com.br/introducao-a-typescript/">. Citado na página 30.

GUEDES, M. **O que é XP - Extreme Programming?** 2020. Acessado em 13 de Outubro de 2021. Disponível em: <"https://www.treinaweb.com.br/blog/o-que-e-xp-extreme-programming">. Citado na página 18.

HIPPEL, E. v.; KROGH, G. v. Open source software and the "private-collective" innovation model: Issues for organization science. **Organization Science**, v. 14, n. 2, p. 209–223, 2003. Citado na página 20.

MANIFESTO ÁGIL. **Manifesto para Desenvolvimento Ágil de Software**. 2001. Acessado em 30 de Março de 2022. Disponível em: <"https://agilemanifesto.org/iso/ptbr/manifesto.html">. Citado na página 17.

MDN WEB DOCS. **Sobre Javascript**. 2021. Disponível em: <"https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/About_JavaScript">. Citado na página 30.

MICROSERVICEIT. **Sobre Docker**. 2021. Acessado em 09 de Outubro de 2021. Disponível em: <"https://www.microserviceit.com.br/docker/">. Citado na página 32.

NODE.JS ORG. **Sobre Node JS.** 2021. Acessado em 09 de Outubro de 2021. Disponível em: <"https://nodejs.org/pt-br/about/">. Citado na página 30.

NOLETO, C. **Aplicações web: entenda o que são e como funcionam!** 2020. Acessado em 09 de Outubro de 2021. Disponível em: <"https://blog.betrybe.com/desenvolvimento-web/aplicacoes-web/">. Citado na página 19.

PALMEIRA, T. V. V. **Aplicações web: entenda o que são e como funcionam!** 2012. Acessado em 09 de Outubro de 2021. Disponível em: <"https://blog.betrybe.com/desenvolvimento-web/aplicacoes-web/">. Citado 2 vezes nas páginas 19 e 20.

REDHAT. **O que é open source?** 2019. Acessado em 02 de Abril de 2022. Disponível em: <"https://www.redhat.com/pt-br/topics/open-source/what-is-open-source">. Citado na página 21.

RODRIGUES, L. S.; MACEDO, L. P. de A. **Inovações Tecnológicas na Agricultura Familiar: Agromart**. Trabalho de Graduação — Universidade de Brasília, Brasília, DF, 2021. Citado 2 vezes nas páginas 14 e 33.

SERRANO, N.; HERNANTES, J.; GALLARDO, G. Mobile web apps. **IEEE Softw.**, IEEE Computer Society Press, Washington, DC, USA, v. 30, n. 5, p. 22–27, set. 2013. ISSN 0740-7459. Citado na página 19.

SOARES, M. dos S. Metodologias Ágeis extreme programming e scrum para o desenvolvimento de software. Minas Gerais, 2004. Citado na página 22.

SOMMERVILLE, I. **Software engineering**. 9ª. ed. [S.l.]: Pearson, 2011. Citado na página 17.

STRAPI. **Sobre Strapi**. 2021. Acessado em 13 de Outubro de 2021. Disponível em: <"https://strapi.io/documentation/developer-docs/latest/getting-started/introduction.html">. Citado na página 31.

TREINA WEB. **O que é React Native**. 2021. Acessado em 09 de Outubro de 2021. Disponível em: <"https://www.treinaweb.com.br/blog/o-que-e-o-react-native">. Citado na página 30.

VALERIANO, D. L. **Gerencia em Projetos: Pesquisa, Desenvolvimento E Engenharia**. [S.l.]: Pearson Education, 2004. Citado na página 15.

WWF-BRASIL. **Você já ouviu falar na Comunidade que Sustenta a Agricultura?** 2018. Acessado em 13 de Outubro de 2021. Disponível em: <"https://www.wwf.org.br/?65282/CSA-Comunidade-que-Sustenta-a-Agricultura">. Citado na página 13.

Apêndices

APÊNDICE A – Documento de Arquitetura de Software

A.1 Introdução

Este apêndice visa apresentar detalhes da arquitetura de software do projeto Agromart, de forma que facilite a visualização dos requisitos e da estrutura para os envolvidos.

A.1.1 Finalidade

Este documento oferece uma visão geral arquitetural abrangente do sistema, usando diversas visões arquiteturais para representar diferentes aspectos do sistema. O objetivo deste documento é capturar e comunicar as decisões arquiteturais significativas que foram tomadas em relação ao sistema.

A.1.2 Escopo

O Agromart 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 produtos ecológicos fornecidos pelos agricultores.

A.1.3 Definições, Acrônimos e Abreviações

UC: User Case (caso de uso).

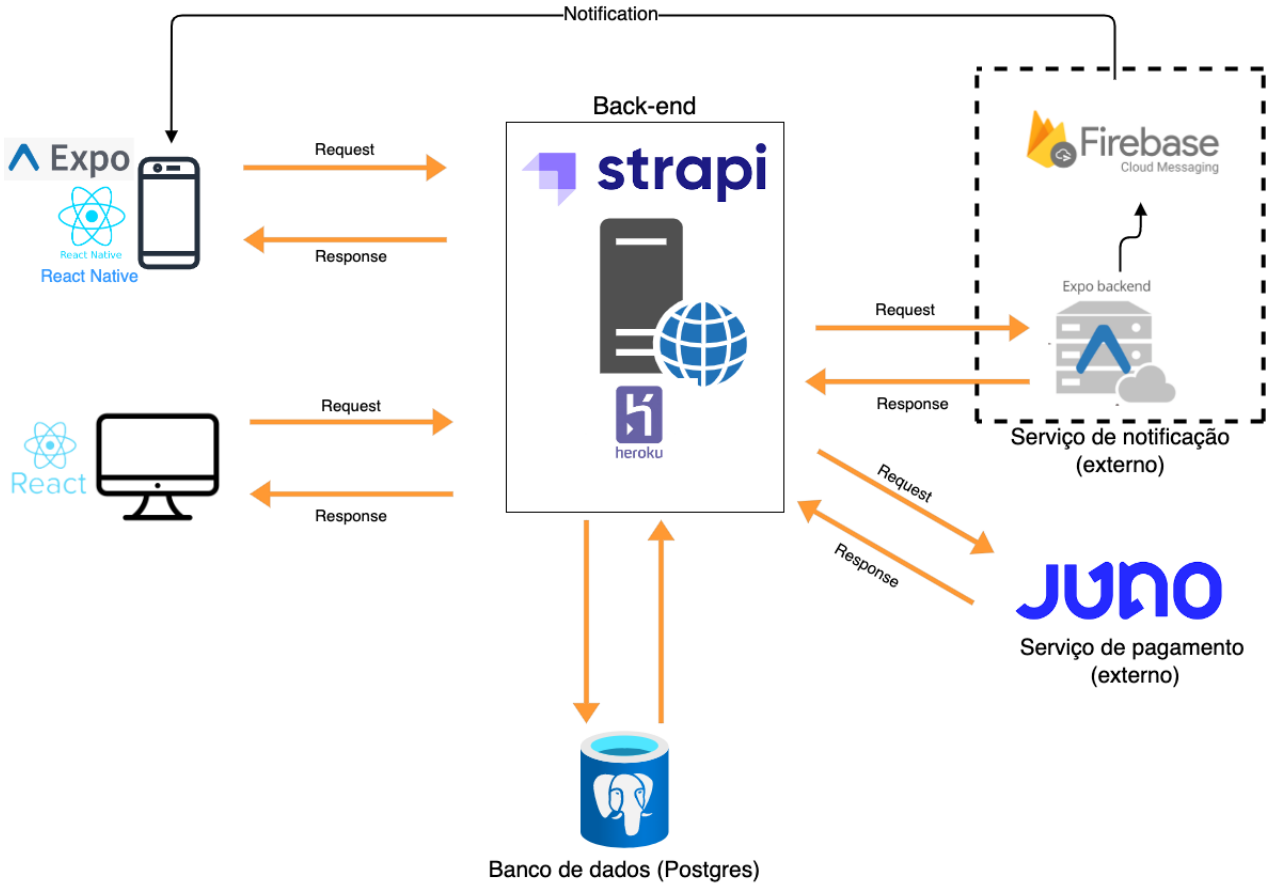
Push notification: notificação que o usuário recebe em um aplicativo de *smartphone*, *tablet* ou navegador sem requisitá-la.

API: Application Programming Interface.

Agricultor: usuário que é um profissional responsável pelo manejo dos mais diversos tipos de plantações e utiliza a plataforma web para venda da sua produção;

Co-agricultor: pessoa que dá suporte na organização e financiamento da produção do agricultor e é usuário da aplicativo.

A.2 Representação da arquitetura



Fonte: Autor, 2022

Figura 19 – Representação da arquitetura - Agromart

O projeto foi modelado seguindo a arquitetura cliente-servidor. No lado do servidor foi utilizado o Strapi para ser executado no *back-end* (servidor) hospedado em um sevidor no Heroku, sendo responsável pela a comunicação com o bancos de dados Postgres, integração com o meio de pagamento Juno e disparo de notificações através das APIs do Expo e do Firebase Cloud Messaging. O *front-end* (cliente) apresenta a versão web, na qual foi utilizada a biblioteca React, e versão *mobile* utilizando React Native e Expo.

A.2.1 Tecnologias

A.2.2 React (17.0.2) e React Native (0.64.3)

O React é biblioteca JavaScript de UI (*User Interface*), que realiza o papel de organizar e enviar e receber informações da API, possibilitando a criação de telas de forma declarativa, ou seja, de como a tela vai reagir ao estado ou dados da aplicação. Essa biblioteca é baseada em componentes, ou seja, é criada partes da interface e ao final juntam-se todos os componentes correspondentes formando, assim, uma interface

maior e mais complexa. React Native, que é uma biblioteca baseada no React, permite o desenvolvimento multiplataforma para os sistemas iOS e Android, utilizando a mesma linguagem de programação. Uma vez escrito, o código é convertido em linguagem nativa dos aparelhos *mobile*.

A.2.3 Expo (44.0.0)

O Expo é uma ferramenta utilizada no desenvolvimento *mobile* com React Native que permite o fácil acesso à rercursos nativos do dispositivo sem precisar instalar qualquer dependência.

A.2.4 Strapi (3.4.6)

Strapi é um CMS que utiliza Node.JS como motor de execução e permite a criação e manipulação de conteúdo de uma forma visual por meio seu painel de administração. Para este projeto, o Strapi foi estruturado para executar somente a API.

A.2.5 Juno

Juno é uma plataforme de pagamentos online que fonerce acesso à sua API de pagamento para integração, possibilitando assim os pagamentos por meio do aplicativo.

A.2.6 Backend Expo

O Expo conta com um servidor de serviços que pode ser acessado através de sua API de integração.

A.2.7 Firebase Cloud Messaging

O Firebase ofecere serviços em nuvem e dentre eles está o Cloud Messaging no qual é possível enviar e receber notificações e mensagens.

A.3 Metas e Restrições da Arquitetura

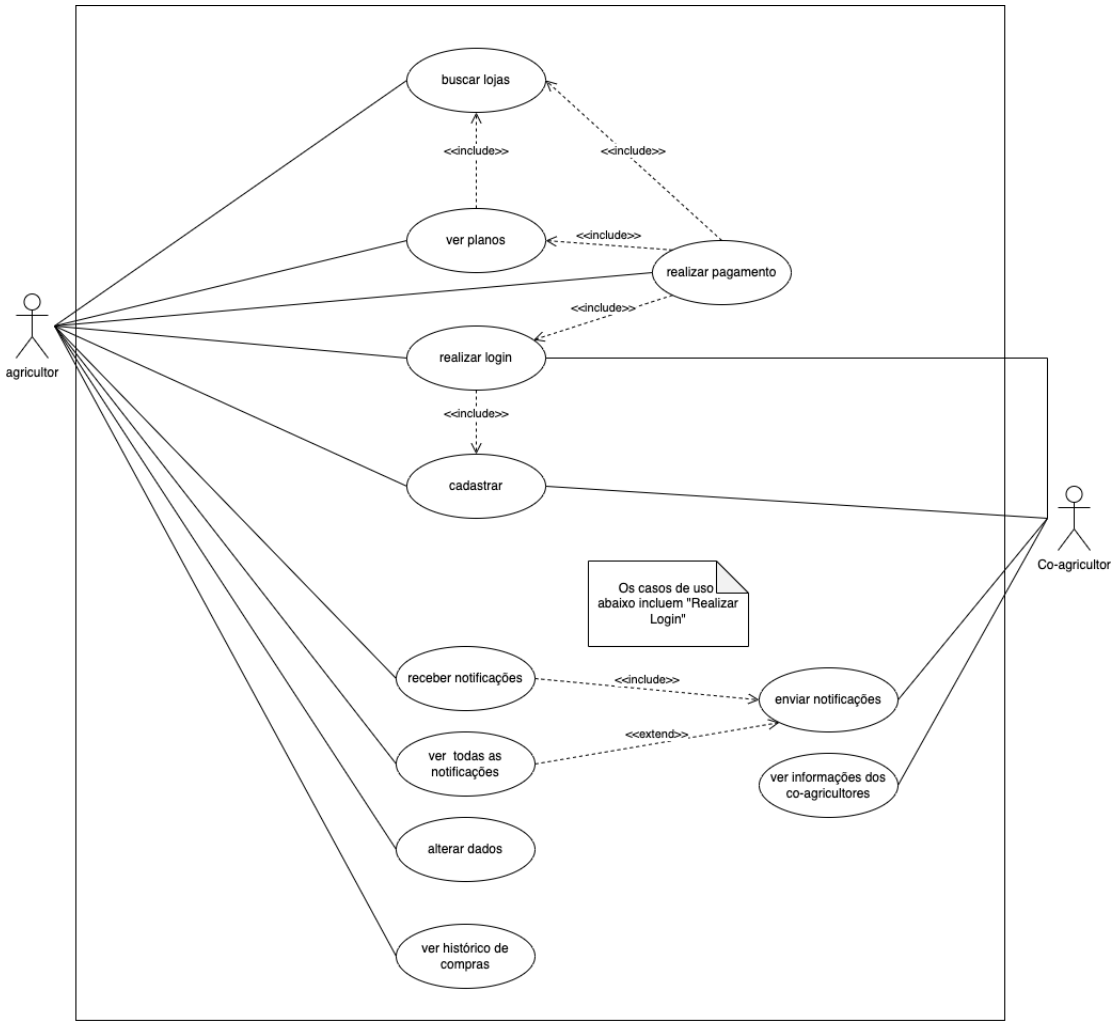
Aqui estão descritos os requisitos e objetivos do software que têm algum impacto sobre a arquitetura.

- O Aplicativo será funcional em dispositivos celulares Android aos quais devem ter acesso a internet.
- É necessária a conexão com internet para utilização do App.

- Os dados extraídos do *front-end* serão armazenados no banco de dados PostgreSQL.
- A informação pessoal do usuário será armazenada no banco de dados.
- As informações do cartão de crédito não devem ser amazenados no banco de dados da aplicação. Esses dados devem ser enviados à API de pagamento, esta devolve algumas informações que podem ser amazenadas no banco de dados do projeto.
- O *hash* do cartão de crédito deve ser criado no *front-end* (aplicativo).

A.4 Visão dos Casos de Uso

O diagrama de casos de uso serve para representar como os casos de uso interagem entre si no sistema e com os usuários (atores), ou seja, como as funcionalidades se relacionam umas com as outras e como serão utilizadas pelo usuário. Os casos de uso do sistema podem ser visualizado na figura 20.



Fonte: Autor, 2022

Figura 20 – Diagrama de casos de uso

A.4.1 Descrição dos casos de uso

A descrição dos casos usos apresentada aqui serve para detalhar melhor o UC, apresentando informações de descrição e os atores envolvidos.

Casos de Uso	Descrição	Ator
UC 01 - Registrar	O usuário registra suas informações na aplicação.	Agricultor e co-agricultor
UC02 - Realizar Login	O usuário valida suas informações e consegue acesso às outras funcionalidades	Agricultor e co-agricultor
UC03 - Buscar Lojas	O usuário consegue buscar e ver as lojas (CSA) por cidade.	Co-agricultor
UC04 - Alterar Dados	O usuário consegue editar seus dados pessoais.	Co-agricultor
UC05 - Receber Notificações	O usuário recebe push notifications e consegue visualizar todas as notificações.	Co-agricultor
UC06 - Ver Todas Notificações	O usuário recebe push notifications.	Co-agricultor
UC07 - Ver Planos	O usuário consegue ver os planos das lojas.	Co-agricultor
UC08 - Ver Histórico de compras	O usuário consegue ver o histórico de compras já realizadas.	Co-agricultor
UC09 - Realizar pagamento	Após ir em um loja e escolher um plano, o usuário consegue realizar o pagamento utilizando cartão de crédito.	Co-agricultor
UC10 - Enviar notificações	O usuário consegue enviar notificações para os co-agricultores.	Agricultor
UC11 - Ver informações dos co-agricultores	Ver as informações dos integrantes da CSA, incluindo o tipo de plano e o status do pagamento.	Agricultor

Fonte: Autor

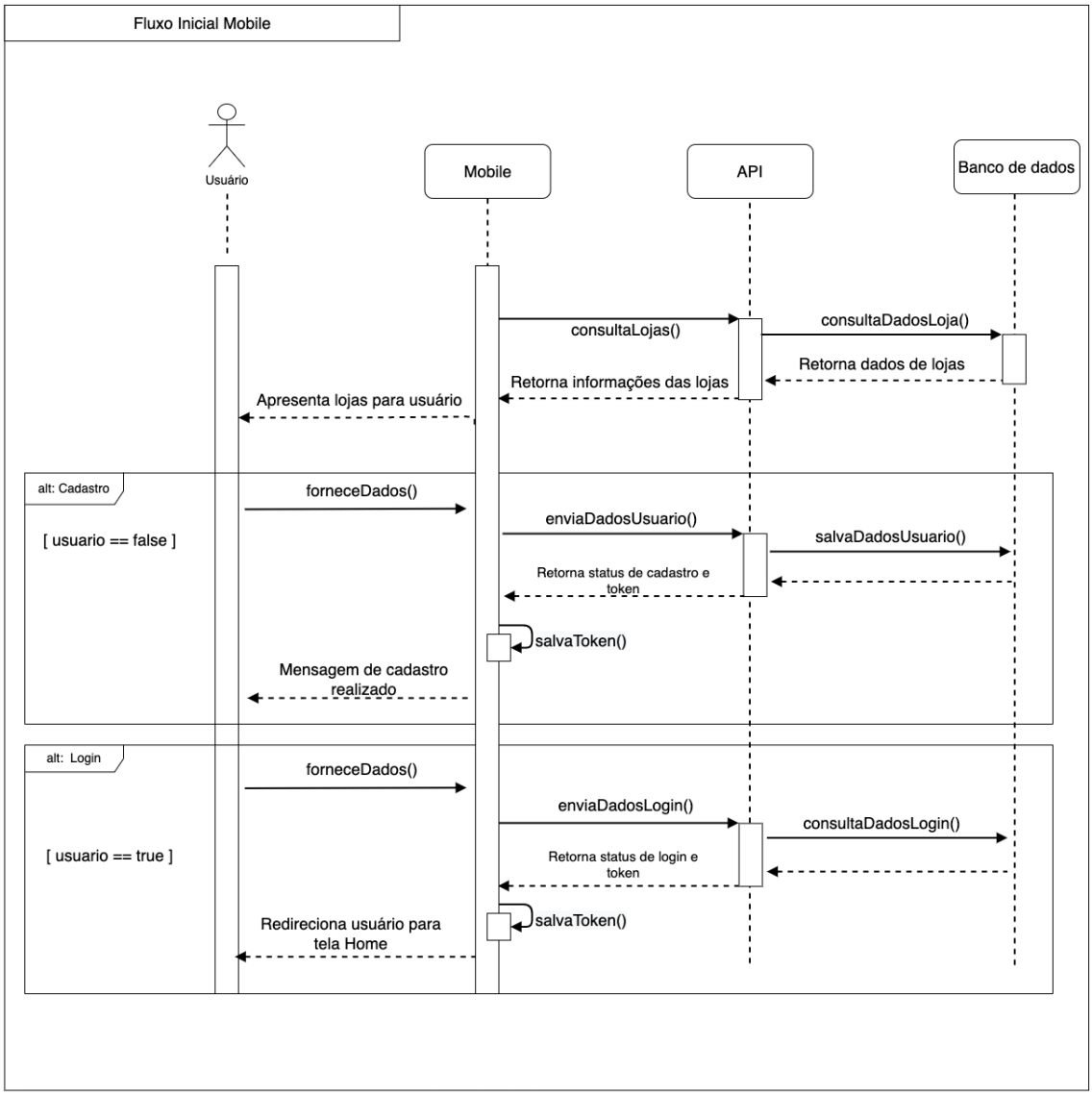
Tabela 1 – Descrição de Caso de Uso

A.5 Diagramas de Sequência

O diagrama de sequência é uma solução de modelagem UML dinâmica que incide na representação das interações entre os objetos em várias linhas de vida. Descreve como objetos e componentes intetagem uns com os outros para concluir um processo, função ou operação.

A.5.1 Fluxo Incial - Mobile

A figura 21 apresenta a troca de informações entre os objetos relacionados ao fluxo inicial no aplicativo.

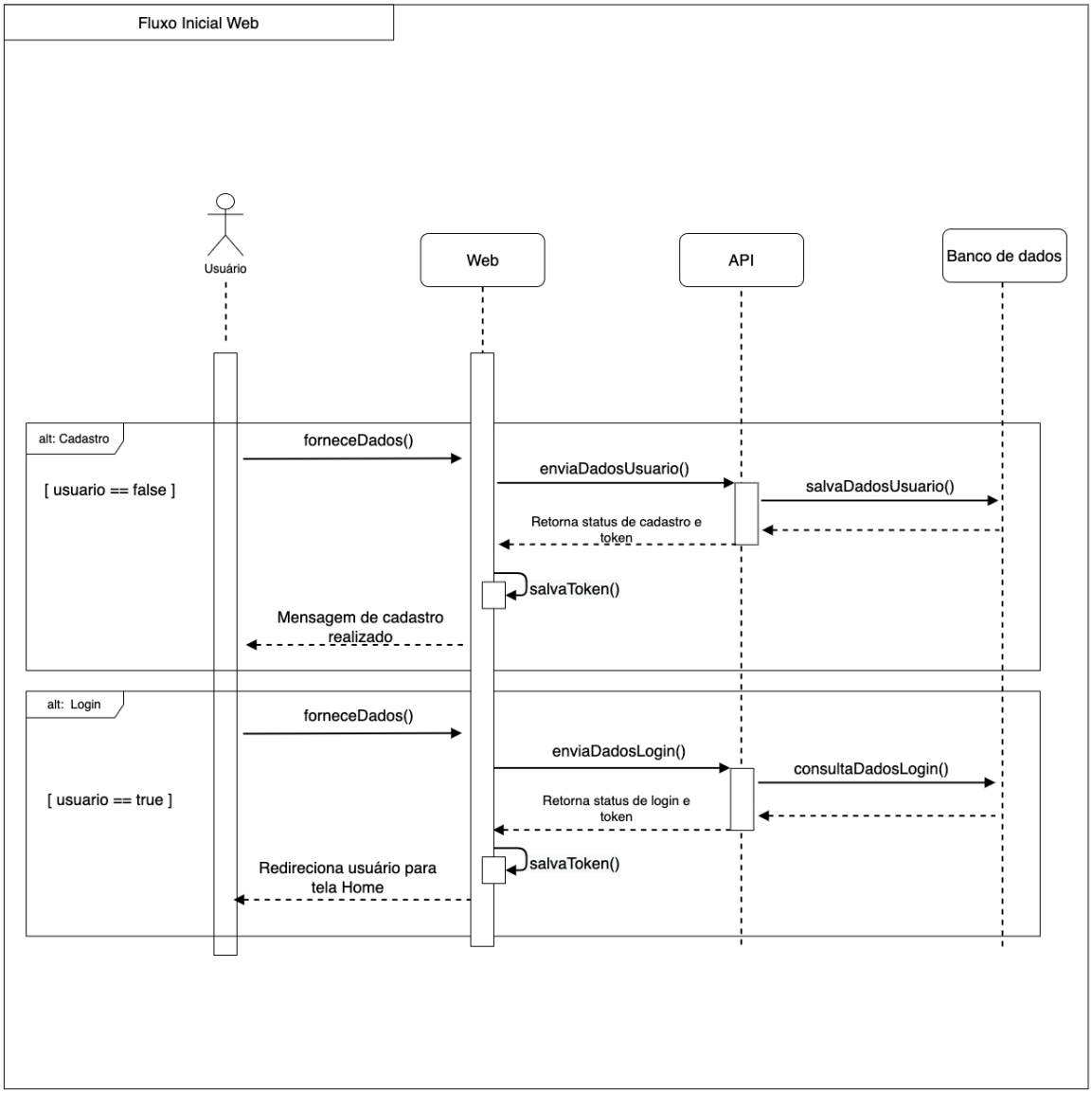


Fonte: Autor, 2022

Figura 21 – Diagrama de sequência - Fluxo inicial mobile

A.5.2 Fluxo Inicial - Web

A figura 22 apresenta a troca de informações entre os objetos relacionados ao fluxo inicial no sistema Web.

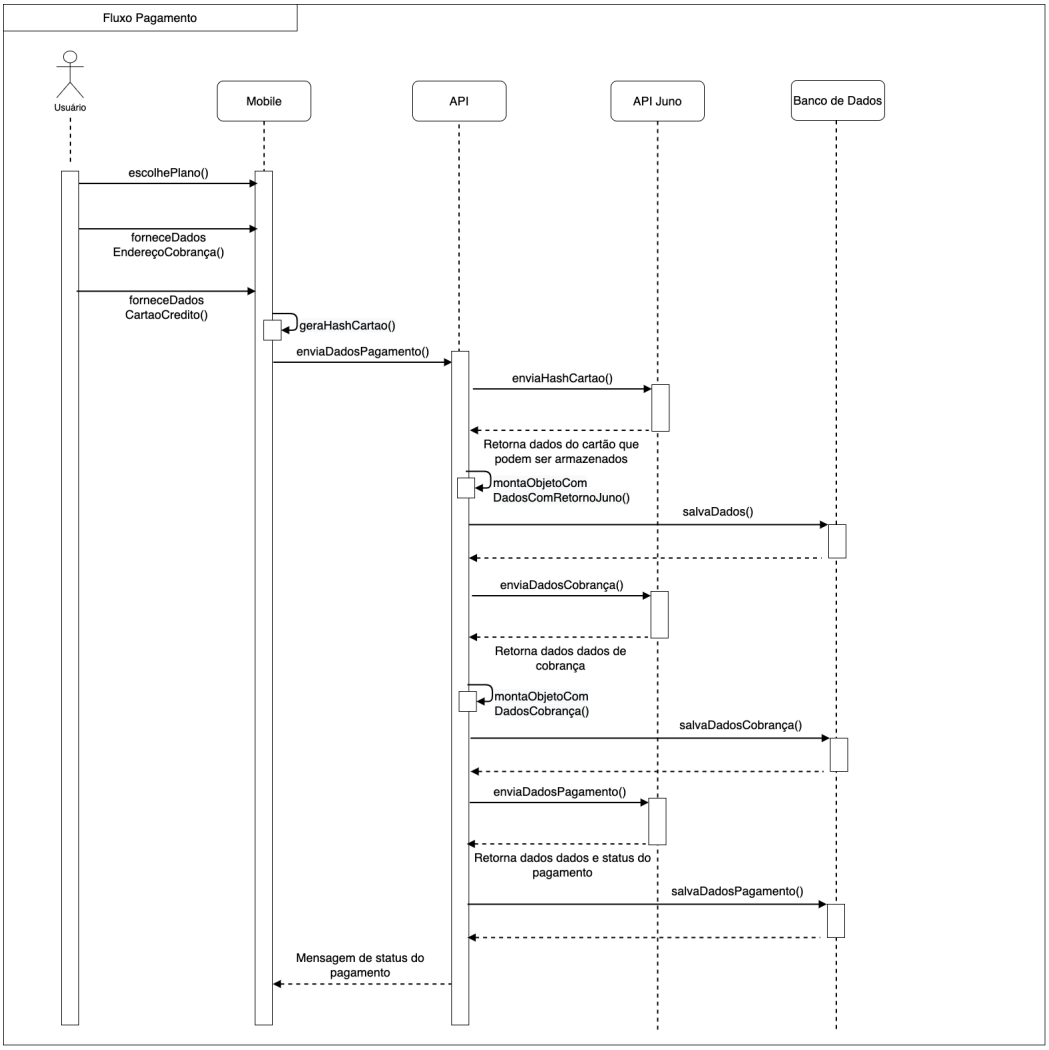


Fonte: Autor, 2022

Figura 22 – Diagrama de sequência - Fluxo inicial web

A.5.3 Fluxo Pagamento

A figura 23 apresenta a troca de informações entre os objetos relacionados ao fluxo de pagamento de plano.

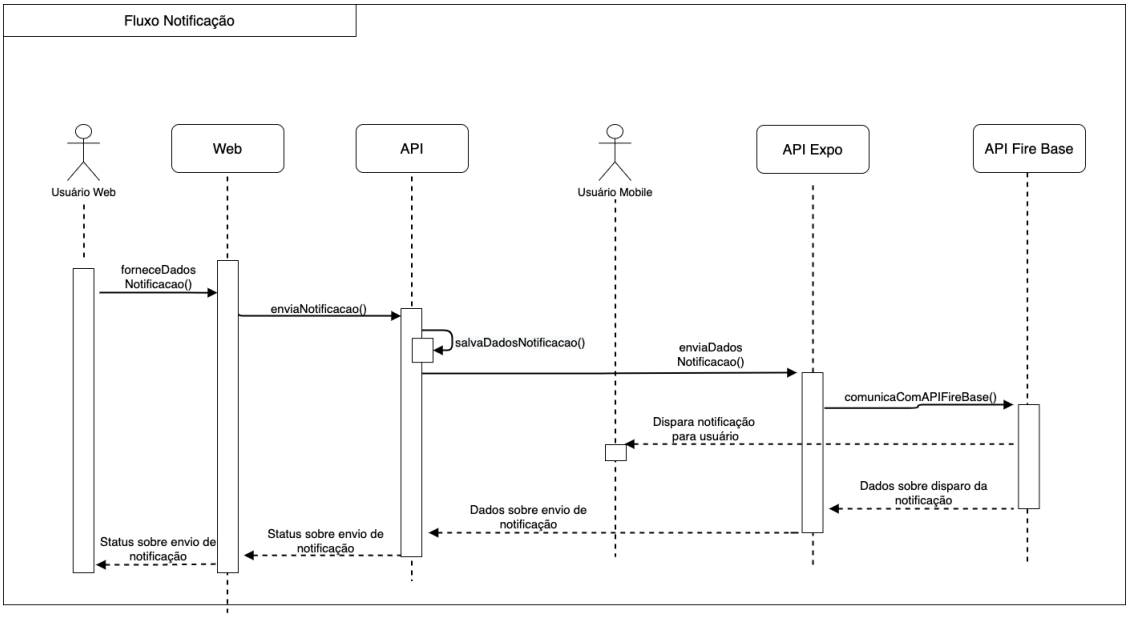


Fonte: Autor, 2022

Figura 23 – Diagrama de sequência - Fluxo pagamento de plano

A.5.4 Fluxo Notificação

A figura 24 apresenta a troca de informações entre os objetos relacionados ao fluxo de notificações.

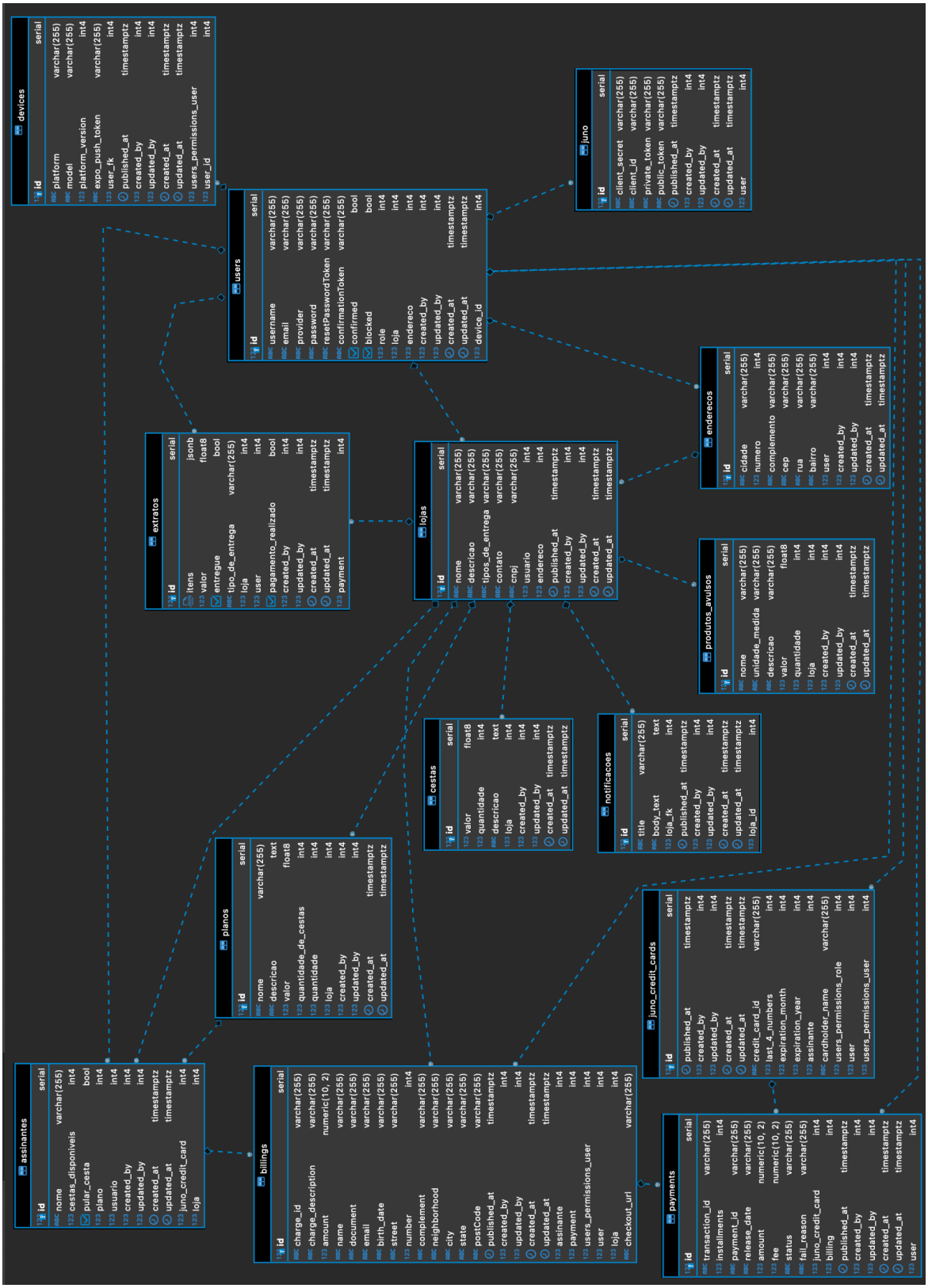


Fonte: Autor, 2022

Figura 24 – Diagrama de sequência - Fluxo de notificação

A.6 Visão de Dados

Diagrama de Relacionamento de Entidade (ER) é um tipo de fluxograma que ilustra o relacionamento entre "entidades" como pessoas, objetos ou conceitos, dentro de um sistema. A figura 25 apresenta este diagrama do projeto.



Fonte: Autor, 2022

Figura 25 – Diagrama de Relacionamento de Entidade - Agromart