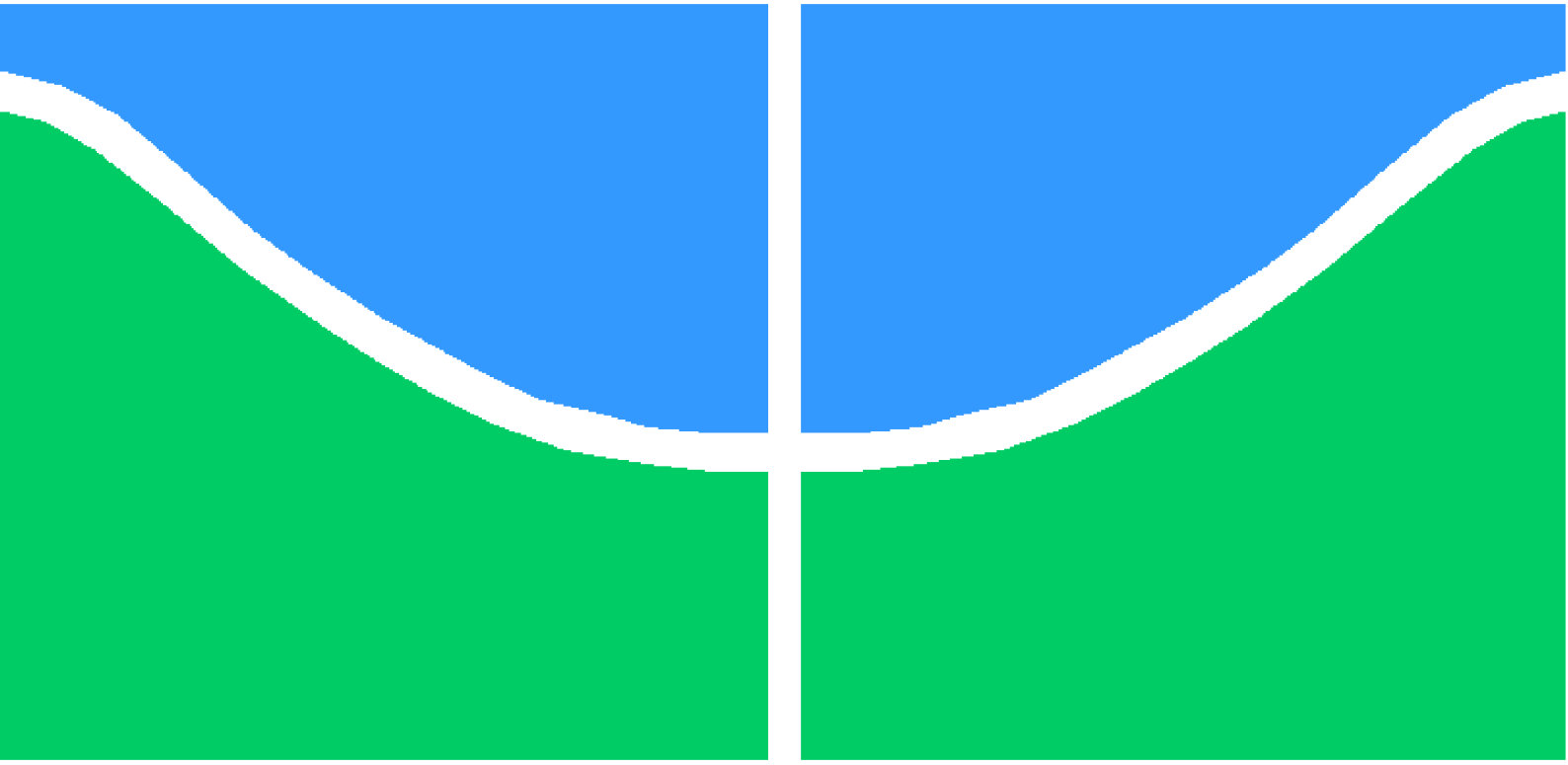


Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Engenharia de Software

Uma evolução do projeto Agromart: implantação individualizada e automatizada de um ambiente de CSA

Autores: André Aben-Athar de Freitas e Pedro Vítor de Salles Cella

Orientador: Dr. André Luiz Peron Martins Lanna

Coorientador: Dr. Rudi Henri van Els

Brasília, DF

2023

André Aben-Athar de Freitas e Pedro Vítor de Salles Cella

Uma evolução do projeto Agromart: implantação individualizada e automatizada de um ambiente de CSA

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

2023

André Aben-Athar de Freitas e Pedro Vítor de Salles Cella
Uma evolução do projeto Agromart: implantação individualizada e automatizada de um ambiente de CSA/ André Aben-Athar de Freitas e Pedro Vítor de Salles Cella. – Brasília, DF, 2023-
64 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 , 2023.

1. Palavra-chave01. 2. Palavra-chave02. I. Dr. André Luiz Peron Martins Lanna. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Uma evolução do projeto Agromart: implantação individualizada e automatizada de um ambiente de CSA

CDU 02:141:005.6

André Aben-Athar de Freitas e Pedro Vítor de Salles Cella

Uma evolução do projeto Agromart: implantação individualizada e automatizada de um ambiente de CSA

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).

Dr. André Luiz Peron Martins Lanna
Orientador

Profa. Dra. Cristiane Soares Ramos
Convidado 1

Prof. Dr. Ricardo Ajax Dias Kosloski
Convidado 2

Brasília, DF
2023

Agradecimentos

Eu, André Freitas, agradeço a minha família e namorada, por sempre me incentivarem e darem a fundação para eu ser o meu melhor. Agradeço aos meus amigos que já tinha antes e os que fiz durante a Faculdade por todas as risadas e bons momentos. Agradeço também especialmente meus amigos Lucas Ganda e Wictor Girardi, por caminharem este caminho ao meu lado, sempre me dando o suporte que precisei desde o início. Agradeço também ao meu amigo Pedro, que mesmo numa fase inicial do curso optou por fazer o Trabalho de Conclusão de Curso ao meu lado. Agradeço a ele também por sempre me ajudar em todas as Disciplinas que cursamos juntos, sempre me indicar as melhores literaturas possíveis e por ser um ótimo amigo.

Eu, Pedro Cella, agradeço ao meu amigo André por ter me chamado para participar desse Trabalho de Conclusão de Curso, sempre me ajudando e me explicando durante as fases de desenvolvimento, e principalmente por ter me dado a esperança de que estaria tudo bem no final desse trabalho, agradeço a ele também por ter os mesmos gostos que eu o que nos tornou ainda mais amigos e camaradas durante todas as etapas e fases que passamos durante o trabalho. Agradeço também aos meus amigos Paulo Gonçalves, Ricardo Venturini, Henrique Amorim e Nilvan Peres por me acompanharem durante todos os semestres da faculdade, escutando desabafos e tornando a experiência da faculdade ainda melhor e mais agradável. Por fim agradeço a minha família e aos meus amigos por me acompanharem e acompanharem essa grandíssima trajétoria e fase da vida.

"Enquanto você não desistir, nada acabou."
(Seo Koji)

Resumo

Este trabalho tem como objetivo apresentar a evolução do software Agromart, com foco em sua característica Open Source. A proposta é explorar como o software poderá ser modificado em seu código mobile React-Native e CMS Strapi, a fim de torná-lo personalizado e automatizado para cada CSA. O objetivo principal é permitir que cada CSA possa gerenciar sua própria aplicação através de um admin Strapi, otimizando a gestão de seus produtos e serviços. Para isso, o trabalho abordará as modificações necessárias no software, bem como a criação de uma documentação detalhada (que se encontra em <https://agromart.github.io/docs/>) para facilitar a utilização e o processo de implantação da aplicação da CSA. O trabalho é relevante para a agricultura sustentável e a gestão de produtos locais, trazendo uma solução moderna e eficiente para as CSAs. A partir deste trabalho, espera-se que outras CSAs possam utilizar o software Agromart de forma mais eficiente e personalizada, contribuindo para o fortalecimento da agricultura sustentável e para o desenvolvimento da economia local.

Palavras-chaves: deploy. CSA. Agromat. Strapi. React Native.

Abstract

This work contemplates the evolution of the Agromart software with an emphasis on the Open Source characteristic of the project. The main objective of this work is to explain how the current Agromart software will be modified in its mobile React-Native code and its CMS Strapi in order to make it individualized and automatic so that each CSA can manage it through its own Strapi admin, in addition to the software in itself, a documentation will be added that aims to instruct the use of the application as well as facilitating the deployment process of the application of the CSA itself.

Key-words: deploy. CSA. Agromat. Strapi. React Native.

Lista de ilustrações

Lista de abreviaturas e siglas

CMS	*Content Management System*
UnB-FGA	Universidade de Brasília - Campus Gama
XP	*eXtreme Programming*
API	Interface de Programação de Aplicações
CSA	Comunidade que Sustenta a Agricultura
APP	*Application*
PaaS	*Plataform as a Servive*
URL	*Uniform Resource Locator*
WIP	*Work in Progress*

Sumário

1 Introdução

O setor agropecuário familiar é visto como uma grande forma de absorção de emprego assim como uma importante fonte na produção de alimentos, sendo que uma boa parte dessa produção é realizada para o consumo próprio enquanto uma outra parte é destinada a venda para um público geral, o que faz com que o foco do desenvolvimento dessa atividade tenha um caráter mais social do que propriamente econômico. Todavia, é preciso destacar que a produção familiar, além de fonte de recursos para as famílias de baixa renda, é também uma grande fonte de riquezas onde tais riquezas são constadas pela economia tanto do setor agropecuário como do próprio país.

Desse modo, é importante destacar o papel dos agricultores familiares, afinal não são apenas a produção e exportação de alimentos, "a expressividade da atividade familiar quantificada pelo PIB do agronegócio familiar se torna mais ampla e define melhor como a produção dos pequenos produtores realmente interfere na economia"(GUILHOTO et al., 2017) .

CSA é a sigla para Comunidade que Sustenta a Agricultura. É uma alternativa de apoio à produção local de alimentos, promovendo espaços de interação entre as pessoas na cidade e no campo. Com o advento da doença causada pelo novo coronavírus (COVID-19), a agricultura familiar constatou a necessidade de se adaptar a uma nova realidade. Com um cenário de distanciamento social e uma economia frágil, as CSAs tiveram um importante papel para a sobrevivência de diversas famílias da área rural.

Elas funcionam de maneira relativamente simples (MEIRELES, 2018): 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 conforme 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.(MEIRELES, 2018)

1.1 História do Agromart

A concepção Projeto Agromart surgiu à partir de um *Hackathon* ocorrido na UnB-FGA 2020, cujo o tema era “Cultivando Conexões” Os idealizadores do projeto foram desafiados a desenvolver um software que coubesse no contexto de agricultura familiar. O objetivo era estabelecer uma conexão entre os agricultores e os consumidores levando em

consideração o isolamento social por conta da COVID-19. Naquele momento o principal problema enfrentado era em relação à divulgação e listagem de produtos dos produtores rurais 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, informações de contato e descrições adicionais. Logo, o co-agricultor poderia visualizar as lojas mais próximas dele através de mapas e filtros, entrar em contato com o co-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. (CORREA; VELUDO, 2021)

Após o evento, os idealizadores se juntaram aos professores da faculdade e entraram em contato com profissionais da área. Após a realização entrevistas, pesquisas e um profundo entendimento das regras de negócio que estão envolvidas em CSAs, foi gerado um novo projeto visando inovar com uma plataforma Web, um aplicativo apenas para co-agricultores e *design* aprimorado (CORREA; VELUDO, 2021). Desta forma, o projeto também estaria mais adequado ao dia a dia e regras de uma CSA.

Atualmente, o projeto Agromart já passou por mais uma interação em que recebeu uma abordagem *open source* e um meio de pagamento digital. Ele é coordenado por professores da UnB-FGA que dão continuidade ao projeto juntamente com estudantes.

1.2 Problema

Atualmente, diversos agricultores familiares enfrentam problemas relacionados à gestão de estoque dos produtos, escoamento das produções, uma comunicação efetiva com os co-agricultores, dentre outros, o que faz com que a existência de alternativas que utilizem inovações tecnológicas venham a ser consideradas uma solução, entretanto ainda existe uma barreira entre essas inovações e os agricultores, essa barreira se trata do conhecimento possuído por eles quanto a essas tecnologias.

A solução que vem sendo desenvolvida durante os últimos trabalhos de conclusão de curso, o Agromart, tem como seu principal objetivo facilitar a comunicação com os co-agricultores, ajudar no planejamento do escoamento de sua produção e também oferecer ferramentas de gerenciamento da CSA como um todo (produção, co-agricultores, pagamentos, dentre outros). A forma como esse trabalho foi desenvolvido trás como foco a resolução de problemas relacionados a CSA Floresta, e que atualmente possui um portal próprio que lida com as demandas dessa CSA, porém existem atualmente mais de 100 CSAs no Brasil sendo 24 localizadas no DF (WWF, 2018) pensando nisso a solução proposta para esse trabalho foi a de individualizar e automatizar o sistema do Agromart, utilizando da característica *open source*, cujo foco será disponibilizar o *software* para que

qualquer CSA possa utilizar e assim criar seu próprio sistema.

1.3 Objetivos

1.3.1 Objetivo Geral

O objetivo principal deste trabalho é incentivar a adesão de novas CSAs ao projeto Agromart. Isto é, colocar em prática a individualização e automatização do ambiente necessário para administrar uma CSA, de modo a permitir que uma CSA seja capaz de criar e controlar seu próprio ambiente de gerenciamento rapidamente.

1.3.2 Objetivos Específicos

- Alteração no código com o objetivo do funcionamento da aplicação a partir do novo modelo, sendo esse um modelo de individualização dos ambientes das CSAs;
- Automatizar ao máximo a instanciação de um ambiente Agromart individualizado, de modo que o administrador de uma CSA possa facilmente levantar sua dashboard integrada com o aplicativo mobile.

1.4 Metodologia de Pesquisa

Ao se falar sobre a manutenção de um *software* queremos nos referir a manutenabilidade podemos definir como "a facilidade que um dispositivo possa recolocado nas condições adequadas a cada vez que haja uma necessidade de manutenção e através dela é possível medir a eficiência da manutenção durante o reparo."(CAVALCANTE; FILHO,) Porém existem tipos de manutenção que um *software* pode vir a sofrer, dentre elas, temos as manutenções, adaptativas, corretivas, evolutivas e preventivas. Baseado no texto de (VARGAS; PEREIRA,) vamos explicar melhor cada uma e por fim relacionar qual dos tipos de manutenção apresentados se relacionam com o trabalho desenvolvido. Dessa forma temos:

- **Manutenção Adaptativa:** Essa manutenção tem como característica realizar a mudança de um *software* com base em um novo ambiente ou realidade em que o mesmo é imposto, ou seja, ele sofre alterações para que possa ser melhor adaptado a um novo ambiente em que será introduzido.
- **Manutenção Corretiva:** De maneira simples essa manutenção tem como objetivo corrigir alguma falha ou defeito encontrado no *software*, geralmente esse tipo de manutenção ocorre após ter sido encontrado um problema pelo usuário, impossibilitando-o de realizar alguma tarefa.

- **Manutenção Evolutiva:** Essa manutenção tem como objetivo introduzir novas tarefas, melhorias ou funcionalidades ao *software*, consequentemente essas evoluções resultam em atualizações, gerando assim uma nova versão do sistema.

- **Manutenção Preventiva:** Seu objetivo é o de prevenir que defeitos se tornem falhas mais agravantes para o sistema, por isso essa manutenção consiste em uma observação e acompanhamento do *software* já implementado por parte dos desenvolvedores .

Com o conceito de manutenção apresentado anteriormente, assim como seus diferentes tipos, torna mais clara a visão de que o presente trabalho se trata de uma manutenção evolutiva, além disso ele seguirá uma metodologia de pesquisa semelhante as utilizadas anteriormente no projeto. Logo, este Trabalho de Conclusão de Curso possui as seguintes características:

Quanto à natureza: é uma Pesquisa Aplicada, pois tem como objetivo de gerar novos conhecimentos para aplicação prática dirigidos à solução de problemas específicos.(MORESI et al., 2003)

Quanto aos objetivos: É uma investigação intervencionista, pois tem como objetivo apresentar uma solução a um problema do ambiente o qual se dispõe a intervir, ou seja, nesse trabalho o objetivo é disponibilizar o *software* do Agromart de forma indidualizada para que CSAs com dificuldades relacionados a gestão, comunicação entre outros, possam utilizá-lo. Fazendo com que seja uma possível solução para os problemas citados anteriormente.(MORESI et al., 2003)

1.5 Propostas de Trabalho

Tendo como o objetivo do trabalho atual a individualização e automatização de um ambiente CSA, é necessário entender o que isso implica e o que será necessário fazer para alcançar tal objetivo.

1.6 Individualização

Quando é falada sobre a Individualização, deve-se recordar do projeto em sua primeira versão. Inicialmente o projeto foi feito com o planejamento de que várias CSAs pudessem se filiar a UNB de forma que todas iriam ser geridas por uma plataforma de administração feita em Strapi.

Como já mencionado, esta opção não seria escalável e nem ideal para o ambiente *open source* e acadêmico. A Universidade de Brasília não deve ser responsável pela ma-

nutenção e suporte de CSAs de modo que qualquer instabilidade do sistema empregaria danos no negócio e na carteira dos Administradores das CSAs.

Visando retirar a responsabilidade da Universidade de Brasília sobre as CSAs e criar uma independência sobre o código e ambiente de administração, foi que nasceu a ideia de individualizar o *dashboard* Strapi de cada CSA.

A nova forma individualizada funcionaria de forma que cada CSA teria seu próprio *deploy* de *dasboard* administrativa. Deste modo o responsável pela CSA consegue ver todos os pedidos e planos feitos para apenas a sua CSA sem depender da UNB para tal. Além disto, esta prática seria mais fiel à ideia de código livre, pois o próprio usuário pode melhorar sua *dashboard* e fazer o *deploy* da mesma, sem precisar do aval da UNB para tal.

Entretanto, existe um problema, a aplicação *mobile* "Agromart" publicada no Google Play não consegue ser facilmente multiplicada por N, da mesma forma que a *dashboard* Strapi. O processo de publicar uma aplicação no Google Play envolve criar conta na plataforma, gerar uma *build* do projeto, pagar taxas e outros passos mais técnicos. Isto foge da proposta do projeto Agromart de facilitar o escoamento de produtos de CSAs e formas simples. Para sanar tal problema, é necessário fazer com que o aplicativo atual se comunique com as "N"CSAs individualizadas.

1.6.1 Api-Dicionário

Após cada ambiente Strapi ser individualizado, o mesmo tem a sua própria Url-Base. UrlBase é o que se chama a URL que serve de início(base) para toda Requisição REST feita para uma aplicação. Logo, a api-dicionario foi idealizada de forma que a mesma guardaria a UrlBase de todas as CSAs que estão integradas com o ambiente Agromart junto com dados do responsável por ela.

Assim, a aplicação *mobile* será modificada para que antes de realizar a ação de login, o usuário deverá selecionar qual CSA ele está vinculado e após esta seleção, a api-dicionário passará a UrlBase da CSA selecionada, e a aplicação continuará funcionando normalmente.

1.7 Automatização

Com necessidade da individualização dos ambientes mostrada, seguimos para um outro ponto que também se deve levar em conta como motivação do projeto Agrmoart. Facilitar a vasão de produtos para pessoas que fazem parte de uma CSA, entretanto, com a individualização dos ambientes, um trabalho extra de fazer o *deploy* de cada *dashboard* Strapi surge.

A automatização no título do trabalho se refere justamente a iniciativa de diminuir o esforço para o usuário final. A ideia é que o Administrador tenha que fazer o menor número de passos possíveis sem conhecimento prévio para que sua *dashboard* se mantenha *online*. Para isto, serão criados tutoriais e um *script* que a partir de informações previamente dadas pelo usuário final, farão o *deploy* do produto na plataforma Heroku, que por si é um ambiente virtual que provará a presença online da CSA.

2 Referencial Teórico

2.1 Manutenção Evolutiva

A tecnologia atualmente vêm se atualizando cada vez mais, isso muitas vezes faz com que *softwares* que são utilizados no dia a dia se tornem obsoletos de forma rápida, o que os obriga a desenvolver atualizações mais frequentes porém nem sempre uma atualização é o suficiente para acompanhar as mudanças de uma determinada ferramenta ou pacote, por conta disso é necessário a relização de manutenções. Dentro de *software* uma explicação a cerca do que é manutenção seria "modificação de um produto de *software* depois de sua entrega (ao cliente) para corrigir erros, melhorar sua performance ou qualquer outro atributo, ou para adaptar o produto a um ambiente modificado."(ESPINDOLA AZRIEL MAJDENBAUM, 2004)

Após introduzido o conceito, dentro dessa mesma área é visível determinados tipos de manutenção que um *software* pode sofrer, dentre eles temos a que será utilizada nesse projeto, chamado de manutenção evolutiva, que consiste em "aperfeiçoar um software ou sistema implementando novos requisitos, bem como melhorar sua estrutura e desempenho."(BRASILEIRO, 2018) a caracterização dessa método se devem ao fato desse trabalho estar evoluindo o *software* Agromart, focando em sua individualização, ou seja, tornando-o capaz de ser usado sem haver uma entidade associada a ele, e também no aspecto *open source* do projeto.

2.2 Engenharia de Software

Engenharia de Software é uma área da Engenharia e da Computação que cuida de toda a parte técnica e científica dos sistemas, desde sua especificação, até o desenvolvimento e manutenção do software. Envolve a aplicação de tecnologias e práticas de gerência de projetos de modo que o projeto sempre esteja produtivo, organizado e com qualidade.

Na Engenharia de Software, o profissional é regularmente submetido a avaliar as vantagens e desvantagens entre vários caminhos a seguir. O trabalho de um engenheiro de software é buscar a sustentabilidade para o produto, gerenciando os custos e dimensionamento da aplicação e fluxo de desenvolvimento como um todo(WINTERS; MANSHRECK; WRIGHT, 2020). Deste modo, o atual trabalho irá utilizar conhecimentos da Engenharia de Software para que a individualização de ambientes de CSAs seja feita da melhor maneira possível.

2.3 Desenvolvimento Ágil de Software

Os métodos ágeis são uma reação às formas tradicionais de desenvolvimento de *software* e reconhecem a necessidade de uma alternativa aos processos de desenvolvimento de *software* pesados e orientados à documentação (SILVA, 2017). No desenvolvimento de software utilizando métodos tradicionais, o trabalho começa com a elicitação e documentação de um conjunto "completo" de requisitos, seguido das atividades de projeto, desenvolvimento e inspeção de arquitetura em alto nível. A partir de meados da década de 1990, alguns profissionais acharam esses passos iniciais de desenvolvimento frustrantes e, talvez, impossíveis. A indústria e a tecnologia se movem muito rápido, os requisitos mudam a taxas que superam os métodos tradicionais e os clientes se tornaram cada vez mais incapazes de declarar suas necessidades com antecedência e, simultaneamente, esperar mais de seu software (COHEN; LINDVALL; COSTA, 2003).

Métodos ágeis buscam solucionar este problema por meio de uma abordagem incremental para a especificação, desenvolvimento e entrega do *software*. Eles têm como meta realizar entregas mais rápidas aos clientes, e estes podem, em seguida, propor alterações e novos requisitos a serem incluídos nas iterações seguintes do sistema (SOMMERVILLE, 2010). Assim, com um contato maior com o cliente durante o processo de desenvolvimento, práticas ágeis como *ExtremeProgramming-XP*, *Scrum* e *KanBan* se tornaram cada vez mais utilizadas.

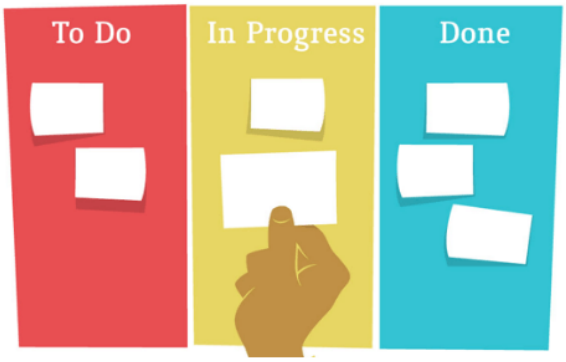
2.3.1 *KanBan*

O método *KanBan* é uma metodologia criada pela *Toyota* na década de 1960 para ajudar a empresa com problemas de estoque. Ele utiliza um quadro com cartões que representam tarefas em seus estágios de produção.(SILVA; ANASTÁCIO, 2019)

O quadro *KanBan* fornece visibilidade ao processo de *software*, ao mostrar o trabalho atribuído de cada desenvolvedor, comunica claramente as prioridades e destaca gargalos. Além disso seu objetivo é minimizar o *Work In Progress* (WIP), ou seja, minimizar o quantidade de tarefas sendo feitas ao mesmo tempo e desenvolver apenas os itens que são solicitados. Isso produz fluxo constante de tarefas liberadas para os clientes e os desenvolvedores podem se concentrar apenas nesses poucos itens em um determinado momento. O método *KanBan* visa se adaptar rapidamente o processo usando ciclos de feedback mais curtos (AHMAD; MARKKULA; OIVO, 2013).

2.3.2 Extreme Programming - XP

O XP é um processo de desenvolvimento que busca garantir que o cliente receba o máximo de valor de cada dia de trabalho da equipe de desenvolvimento. O processo tem como objetivo principal levar ao extremo boas práticas de programação e boas práticas



Fonte: (SITEWARE, 2016)

Figura 1 – Representação de um quadro *Kanban*

para o desenvolvimento de *software* no geral (EXTREME PROGRAMMING, 2013). Atuando de forma harmônica e coesa, certas práticas e conjunto de valores asseguram que o cliente sempre receba um alto retorno do investimento em *software* (TELES, 2017).

Existem cinco valores fundamentais no XP, que são:

- comunicação: boa comunicação entre os membros da equipe e com os clientes;
- simplicidade: o *software* deve ser simples e descrito de forma clara;
- coragem: conseguir apontar erros e simplificações necessárias no *software*;
- respeito: os membros da equipe devem se respeitar para que haja uma boa comunicação entre todas as partes envolvidas e,
- feedback: o desenvolvedor deve sempre informar o cliente.

Trabalhando com tais valores, o XP também diz que devem ser seguidas as seguintes práticas:

- cliente presente: o cliente deve conduzir o desenvolvimento a partir do *feedback* que recebe do sistema;
- jogo do planejamento: todos os detalhes sobre releases, iterações, histórias, pontos e velocidade devem ser descritos;
- *stand up meeting*: a equipe de desenvolvimento se reúne a cada manhã para avaliar o trabalho que foi executado no dia anterior e priorizar aquilo que será executado hoje;
- programação em par: os desenvolvedores implementam as funcionalidades em pares;
- desenvolvimento guiado pelos testes: os desenvolvedores escrevem testes para cada funcionalidade antes de codificá-las.;

- *refactoring*: modificar partes do sistema que estejam funcionando para facilitar a sua manutenção;

- código coletivo: os desenvolvedores têm acesso a todas as partes do código e podem alterar aquilo que julgarem importante sem a necessidade de pedir autorização de outra pessoa;

- código padronizado: a equipe estabelece padrões de codificação;

- design simples: a equipe deve sempre optar por um código que seja suficiente para atender às necessidades da funcionalidade que está implementando;

- ritmo sustentável: o XP recomenda que os desenvolvedores trabalhem apenas oito horas por dia e evitem fazer horas extras;

- integração contínua: os pares devem integrar seus códigos com o restante do sistema diversas vezes ao dia e,

- releases curtos: a equipe deve produzir um conjunto reduzido de funcionalidades e o coloca em produção rapidamente.

2.3.3 Scrum

O Scrum é um *framework* para projetos ágeis utilizado para o gerenciamento e desenvolvimento de produtos, com a característica de ser iterativo e incremental, além de focar na entrega de valor de um negócio no menor tempo possível. O *framework* sugere um conjunto de conceitos e práticas que se encaixa perfeitamente no desenvolvimento de produtos, propondo um auto-gerenciamento dinâmico, versátil e altamente adaptável que se torna muito eficiente durante a execução de projetos que possuem como objetivo final a entrega de um ou mais produtos (CRUZ, 2013).

No Scrum estão previstas três fases. A primeira é uma fase de planejamento geral, em que se estabelecem os objetivos gerais do projeto e da arquitetura do *software*. Em seguida, ocorre uma série de ciclos de *sprint*, sendo que cada ciclo desenvolve um incremento do sistema. Finalmente, a última fase do projeto encerra o projeto, completa a documentação exigida (SOMMERVILLE, 2010).

Sprints são periodos de tempo nos quais o time se compromete em realizar uma série de tarefas. Essas tarefas são chamadas de *sprint backlog* e são retiradas de um conjunto maior de tarefas chamadas de *product backlog*. Sendo assim o *product backlog* representa todos os requisitos para o produto que está sendo desenvolvido ficar pronto (SCHWABER; SUTHERLAND, 2010).

2.4 Sistema de gestão de conteúdo

Um Sistema Gerenciador de Conteúdo (do Inglês *Content Management System-CMS*) "é um aplicativo de computador usado para criar, editar, gerenciar, pesquisar e publicar vários tipos de mídia digital e texto eletrônico".(NAIK; SHIVALINGAIAH, 2022) Usualmente um CMS é responável pela gestão, coleta e publicação de informações conhecidas também como componentes de conteúdo.

O Projeto Agromart está estrututado com base em um CMS fortemente conhecido no mercado chamado Strapi. O Strapi nos fornece uma plataforma admnistrativa para que o responsável pela CSA possa criar cestas, organizar pedidos e compras via um só portal.

2.5 Aplicações mobile

Uma aplicação mobile é uma solução de *software* desenvolvida para ter como sua plataforma final smartphones ou tablets. Geralmente é baixado de uma loja on-line, como exemplos App Store Google Play, e instalada diretamente no dispositivo portátil desejado.

O grande desafio do desenvolvimento de aplicações web móveis é a heterogeneidade de dispositivos móveis e navegadores da web instalados nos dispositivos. As diferenças específicas de cada aparelho, como sistema operacional e possibilidade de diferentes inputs, por exemplo, leitor biométrico, câmeras, microfones e sensores de proximidade, fazem o desenvolvimento da aplicação ser mais demorado e complexo.

O desenvolvedor pode escolher abstrair a diferença entre sistemas operacionais e desenvolver a aplicação de forma híbrida. Deste modo, a aplicação é compilada como uma aplicação Web, recebendo e unificando a utilização de códigos específicos. Por mais que os dispositivos diferenciem em seus navegadores pré-instalados, a maioria dos dispositivos móveis atuais, tanto suportam WML ou um subconjunto de (X)HTML.(SPRIESTERSBACH; SPRINGER, 2004)

Outra opção a ser utilizada é desenvolver a aplicação de maneira nativa. Os aplicativos nativos são construídos para uma plataforma específica com ferramentas e linguagem fornecidos pelo fornecedor da plataforma. Por exemplo, um aplicativo móvel desenvolvido com Android Framework não é executado no *framework* iOS. Este modo é o ideal para o desempenho, pois pode utilizar totalmente a disponibilidade de recursos do sistema mobile em questão. No entanto, o tempo de desenvolvimento pode aumentar, pois temos que desenvolver de forma diferente para cada plataforma.(LIM, 2015)

2.6 Ambientes de Software

Um ambiente de *software* é uma coleção de programas, bibliotecas e utilitários que permitem aos usuários executar tarefas específicas. Ambientes de software são frequentemente usados por programadores para desenvolver aplicativos ou executar aplicativos existentes (BEAL, 2005). Um ambiente de *software* para uma aplicação específica pode incluir o sistema operacional, o sistema de banco de dados, APIs ou compiladores.

Comumente ambientes de *software* são divididos em pelo menos três:

- ambiente de desenvolvimento;
- ambiente de homologação;
- ambiente de produção.

O ambiente de desenvolvimento é usado para construção dos aplicativo e é nele que os desenvolvedores completam a maior parte de seu trabalho. Normalmente, o ambiente de desenvolvimento é configurado localmente e o trabalho é facilitado por um repositório Git. Usuários e clientes não podem acessar nada feito no ambiente de desenvolvimento, a menos que o desenvolvedor os mostre (MYLONAS, 2017).

O ambiente homologação é o ambiente de intermédio entre desenvolvimento e produção. É nele que os testes de software são feitos e possíveis bugs são recolhidos. Assim a equipe tiver uma versão do produto pronta para lançamento, ela poderá realizar o *deploy* da nova versão do produto para o ambiente de produção

O ambiente de produção é a etapa final dentro do processo de desenvolvimento de software. É o onde o*deploy* da aplicação estará disponível para o público e será de fato utilizado pelo cliente final (IBM, 2022).

2.7 *Deploy* de Software

O *deploy* de sistemas de software é um processo complexo de pós-produção que consiste em disponibilizar o software para uso e mantê-lo operacional. Ele deve lidar com restrições relativas ao sistema e à(s) máquina(s) alvo, em particular sua distribuição, heterogeneidade e dinâmica, e satisfazer os requisitos de diferentes partes interessadas. No contexto de mobilidade e abertura, a implantação deve reagir à instabilidade da rede de máquinas (falhas, conexões, desconexões, variações na qualidade dos recursos, etc.)(ARCANGELI; BOUJBEL; LERICHE, 2015).

Quando se fala em *deploy* de software, geralmente é feita a pergunta "aonde o está o *deploy*?". Isso se deve ao fato de que a implantação de um sotware pode ser feita em diversos serviços disponíveis no mercado. Logo, quando é dito que o deploy do produto

foi feito na plataforma Heroku, quer se dizer que a aplicação está hospedada no ambiente de nuvem fornecido pela empresa Heroku.

2.8 Elicitação de Requisitos

De uma maneira simples, "elicitação de requisitos é um processo de coleta dos requisitos de um *software* de usuários, clientes e *stakeholders* ". (TIWARI; RATHORE; GUPTA, 2012) A utilização da elicitação de requisitos pode definir o sucesso ou fracasso de um projeto, isso porque é com essas técnicas que se tem definições do que se tornará o produto, ou seja, desde uma entrevista com o cliente, até a necessidade do usuário final, todos os aspectos que definem funcionalidades, formas de funcionamento, entre outros, são considerados requisitos de *software*.

A importância de uma elicitação se baseia no fato de tenta estabelecer o produto final, o mais fiel o possível ao que foi idealizado pelo cliente, e isso é feito através técnicas que visam tirar o máximo de informação de uma pessoa, ao mesmo tempo em que alimentam as novas funcionalidades e meios de funcionamento de uma determinada aplicação.

A modelagem dos requisitos de *software* é umas das etapas de elicitação, pode-se dizer que modelagem é um conjuntos de diagramas, que utilizando de especificidades representam o funcionamento de um *software*. Esse conceito se baseia em utilizar dos requisitos levantados para criar diagramas, a utilização desses diagramas se torna útil, por utilizarem figuras, além de um linguajar mais comum, sem muitos jargões de tecnologia. Dessa forma são feitos diagramas que podem ser analisados por qualquer pessoa, sem contar que esses diagramas representam o real funcionamento do *software* em si, o que além de facilitar a comprenssão de como ele funciona ainda é possível determinar o surgimento de possíveis erros.

A análise é uma etapa em que se validam os artefatos criados pela etapa de modelagem, ela serve para realizar os processos de verificação e validação, dessa forma tudo que foi criado será analisado para determinar se o *software* cumpri ou não com a finalidade proposta inicialmente.

3 Gerenciamento do Projeto

3.0.1 Metodologia de Desenvolvimento

Este trabalho utilizará uma abordagem considerada como híbrida, ou seja, nele serão utilizados vários conceitos existentes dentro da metodologia ágil. Desse modo existirá um equilíbrio no enfoque do projeto trazendo uma abordagem voltada para as pessoas mas também ao processo em si. A utilização dessas metodologias busca garantir qualidade na implementação daquilo que foi proposto mas também preza pela acessibilidade dos futuros usuários que utilizarão do Agromart. Por esse motivo existe um foco na implementação, na qualidade e na documentação desse trabalho, afinal por ser uma aplicação de caráter *open source* é necessário ter documentada a forma como ele deverá ser implementado para que funcione devidamente, além disso necessita-se ter uma documentação que possa servir como um tutorial para quem for utilizar o Agromart, afinal muitas vezes podem existir dúvidas durante o uso do *software*. Com as metodologias ágeis o modo como esse resultado será obtido é facilitado pela capacidade de tais metodologias em se adaptar às adversidades encontradas ao longo do projeto (SOARES, 2004), isso ocorre por conta das vantagens da metodologia ágil que tornam o ambiente de desenvolvimento propício a mudanças e inovações visando garantir uma satisfação e qualidade sobre o produto a ser entregue.

3.0.2 Scrum

Os conceitos utilizados serão:

- *Sprints*: com duração de 2 semanas, as *sprints* serão usadas para definirmos as tarefas e os objetivos tornando a progressão da aplicação mais iterativa. A escolha do tempo de duração das *Sprints* foi decidido pelos membros como uma boa perspectiva de tempo baseando-se nas experiências que ambos tiveram em outros trabalhos e projetos. Por esse motivo é estimado que em duas semanas as tarefas estejam concluídas e ao final da *Sprint* novas tarefas já tenham sido definidas.

- *Sprint Planning*: é nessa reunião que será decidido o escopo da próxima *sprint*, sendo estabelecida uma ordem de prioridade e o tempo estimado de cada tarefa pois podem acontecer casos em que tarefas exigem um tempo maior que a duração da própria *sprint*.

- *Sprint Review*: será usada para avaliar a progressão das tarefas, a produtividade dos membros e a qualidade dos artefatos produzidos. Nesse último caso, tal avaliação

será feita através da avaliação dos testes realizados. Desse modo pretende-se evitar ou mitigar possíveis erros.

- *Sprint Backlog*: será utilizado controlar as tarefas a serem desenvolvidos na *sprint*.
- *Product Backlog*: será utilizado saber quais tarefas precisam ser desenvolvidas para que se tenha o produto final.

3.0.3 Kanban

A execução do método Kanban será apoiada pela ferramenta Zenhub. Sua escolha deve-se à facilidade de integração ao Github o que torna descomplicado a visualização, do fluxo de trabalho, tanto pelos membros quanto para pessoas de fora. Além disso a plataforma ainda oferece um sistema de pontuação de tarefas, que podem ajudar a medir o nível de produtividade dos membros em cada *sprint* caso necessário.

3.0.4 Extreme Programming - XP

Do XP serão seguidos os 5 valores fundamentais, e apenas as práticas que são realmente necessárias ao projeto, listadas a seguir:

- jogo do planejamento: serve para estimar o nível de esforço da próxima *sprint* através de estimativas das atividades que deverão ser desenvolvidas.
- programação pareada: é um tipo de implementação que serve para que os membros se auxiliem durante o desenvolvimento de uma atividade ao ter alguma dificuldade ou dúvida.
- padronização de código: definição e utilização de regras de implementação, em que se busca uma garantia que qualquer membro conseguirá desenvolver ou manter aquele código.
- integração contínua: sua proposta é a de agir imediatamente sobre o que há de errado no código, para que dessa forma seja corrigido e disponibilizado com rapidez e novamente monitorado visando progresso nas mudanças realizadas.

3.1 Ferramentas de comunicação

3.1.1 Discord

Discord é uma plataforma focada em comunicação que foi criada para resolver um grande problema: se comunicar com amigos ao redor do mundo durante jogos online (DISCORD, 2022). Conforme o tempo ela foi se popularizando e se tornou um grande

portal de comunicação fora do âmbito dos jogos. Sua popularização se dá ao fato de dentro dos canais de voz ser possível compartilhar voz, textos e telas. Com isso, o Discord torna-se um ambiente propício para o desenvolvimento e prática do *pair programming*.

3.1.2 Zenhub

O ZenHub é uma plataforma de gerenciamento de projetos criada para equipes de desenvolvimento de software usando o GitHub. Fortemente integrado ao GitHub, o ZenHub permite que os usuários movam facilmente *issues* pelas pipelines em um quadro Kanban, organizem projetos complexos e personalizem fluxos de trabalho para atender às necessidades da equipe(LABELLE, 2019).

Ele será adotado para ser a ferramenta de kanban do time, para que seja possível acompanhar o progresso e identificar e eliminar gargalos por meio de métricas como *veolicty* e *burndown*.

3.1.3 Google Drive

Google Drive é a plataforma de armazenamento em nuvem do Google. Nele é possível que usuários compartilhem documentos por meio de diretórios compartilhados. Esta ferramenta será utilizada para que os integrantes do TCC guardem referências bibliográficas, relatórios de reuniões, arquivos de configuração de ambiente entre outros.

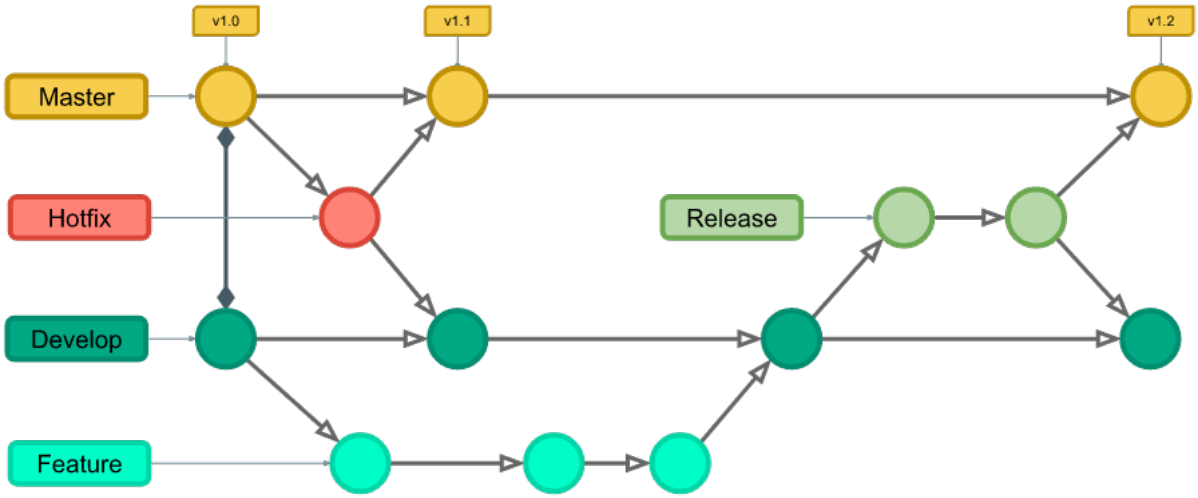
3.1.4 WhatsApp

O WhatsApp surgiu como uma alternativa ao sistema de SMS e agora possibilita o envio e recebimento de diversos arquivos de mídia: textos, fotos, vídeos, documentos e localização, além de chamadas de voz. Por meio desta aplicação, os integrantes do TCC1 irão manter contato constante com membros administrativos das CSAs. Desse modo, espera-se a comunicação entre os envolvidos nesse projeto facilite o desenvolvimento de modo que a aplicação final seja entregue dentro das expectativas do cliente

4 Gerenciamento do Produto

4.1 Gerência de Configuração de Software

Atualmente vem sido utilizado um padrão de política de *branches* que será mantido nesse projeto, com o objetivo de continuar minimizando a chance de possíveis impactos negativos durante o desenvolvimento. O *Gitflow* criado em 2010, apresenta uma política de *branching* interessante, que o torna uma ótima escolha para o projeto, visto que 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 (CORREA; VELUDO, 2021)



Fonte: (CORREA; VELUDO, 2021)

Figura 2 – Fluxo de trabalho - Gitflow

A *master* será a *branch* de produção do projeto. Isso significa que ela só será atualizada com códigos testados e estáveis.

A *branch* de *hotfix* é voltada apenas para a correção de possíveis *bugs* e erros. Ela deverá ser criada a partir da *master* e após a correção da falha em questão será aberto um *pull request* tanto para a *master* quanto para a *develop*.

A *branch develop* é a *branch* que irá acumular funcionalidade para que após uma certa quantidade de alterções, seja gerada uma versão estável do produto. Ela deve receber funcionalidades vindo de *branchs* de feature e possíveis *hotfixes* necessários.

Por fim temos a *branch* de *features* onde nela serão desenvolvidas funcionalidades necessárias para a aplicação. Esse tipo de *branch* só poderá ser mesclada com a branch *develop*.

4.1.1 Repositório

Por ser um projeto *open source* e que já foi previamente desenvolvido por outros alunos num trabalho anterior, foi decidido manter a utilização do repositório no Github que já criado, essa escolha se deve pelo fato do trabalho a ser desenvolvido apresentar evoluções do trabalho anterior, sem contar que isso garante um dos propósitos que seria a liberdade nas contribuições do projeto. Além disso a escolha se deve pela facilidade quanto ao uso da plataforma Github e uma de suas ferramentas, o Zenhub, que também será utilizado no projeto.

4.1.2 Análise estática de código

Ferramentas de análise estática tem como objetivo garantir a qualidade no desenvolvimento do código analisando se o mesmo segue os padrões existentes da linguagem de programação utilizada, como regras em relação ao nome de variáveis, disposição do código, dentre outros. É dado o nome de análise estática, pois usualmente essas ferramentas realizam essas correções sem a necessidade de o código ser executado, forma essa chamada de estática.

O ESLint é uma ferramenta *open source* que tem como objetivo encontrar possíveis erros na implementação do código, assim como boas práticas, a ferramenta está associada a linguagem *JavaScript* e utiliza do *ECMAScript*, um conjunto de padronizações criadas para a linguagem *JavaScript*, para fazer essa verificação. O ESLint já vem configurado ao criar o projeto, porém ele possui a opção de customização quando necessário utilizar para uma regra de negócio específica(RODRIGUES; MACEDO, 2021).

4.2 Backlog

O backlog de produto deste projeto utilizará conceitos das histórias de usuário, que são utilizadas em metodologias ágeis, porém as convenções de escrita das mesmas não serão seguidas à risca. O conceito aplicado refere-se à descrição das tarefas de backlog de modo que elas deverão ser curtas, simples e de fácil entendimento.

4.2.1 Backlog aplicação mobile

O backlog da aplicação mobile se refere a todas as atividades que deverão ser realizadas a fim de possuir um aplicativo que funcione condizente com o novo ambiente individualizado.

- Documentar Aplicação Mobile
  - Documentar possíveis jornadas de usuário. (*Mapping*).

- Novo Login no app
  - Nova tela de escolha de CSA pertencente,
  - Guardar no *AssyncStorage* a URL base da CSA pertencente.
- Fluxo de Pagamento
  - Contornar fluxo de pagamento para não precisar de cartão de crédito.
- Jornada Inicial do App
  - Retirar antiga tela de seleção de CSA por região após login.
- Criação da nova conta do Agromart no GoogleAPP
  - Criação da conta;
  - Novo meio de pagamento,
  - Upload do APP.

4.2.2 Backlog Admin Strapi

O backlog do *Admin Strapi* se refere a todas as atividades que serão realizadas com o intuito de deixar o administrador de cada CSA capaz de desempenhar suas funções.

- Criar serviço que guardará endpoints base de cada CSA.
  - Instância de banco de dados;
  - Api Node,
  - Deploy no Heroku.
- Documentar Admin Strapi
  - Documentar papéis do Admin:
  - Diagrama de dados do Strapi,
  - Documentação das regras negociais.
- Automatizar processo de Deploy
  - Criar arquivo onde serão colocadas as variáveis preenchíveis pelo Administrador da CSA
  - Criar Script que irá ser rodado para o deploy da aplicação,
  - Documentar como compilar o projeto

5 Suporte tecnológico

A linguagem *Javascript* é utilizada como base de todo o projeto Agromart. Ela está presente no atual servidor, aplicativo e também estará presente no novo serviço que será utilizado como dicionário de URL. Javascript é uma linguagem leve e interpretada que é amplamente utilizada para o uso de *scripting* para páginas web (DOCS, 2022). Entretando, com avanço e popularidade dos *frameworks Javascript* é muito comum a utilização da linguagem do lado do servidor.

O projeto utilizará o *Javascript* pelo lado do servidor, por meio do *Node.js*. Ele roda o *Javascript* de forma assíncrona e orientada a eventos e foi projetado para criar aplicativos de rede escaláveis (ORG, 2022). No geral ele é utilizado na criação de API's com o *Express*, um *framework* para Node.js que fornece recursos mínimos para construção de servidores web (EXPRESS, 2022).

A aplicação mobile é escrita utilizando o React Native. React Native é um *framework Javascript* usado para escrever aplicações mobile e com renderização nativa tanto para platformas iOS quanto Android. Ele é baseado em React, biblioteca *Javascript* do Facebook para criar interfaces de usuário, ao invés de ser utilizado em navegadores WEB, ele tem como alvo dispositivos móveis. Tal finalidade permite que os desenvolvedores WEB possam escrever aplicativos móveis que parecem verdadeiramente "nativos", a partir da utilização uma biblioteca *Javascript* que já conhecem e dominam. Além disso, por permitir que a maioria do código seja compartilhada entre plataformas mobile, o React Native facilita o desenvolvimento simultâneo para *Android* e iOS (EISENMAN, 2017).

Internamente ao React Native, é utilizado o *Typescript*. Este é um superset baseado em Javascript que permite um melhor controle dos tipos de variáveis e funções sem perder as características da linguagem e contando com as suas bibliotecas e *frameworks* (PIRES, 2012).

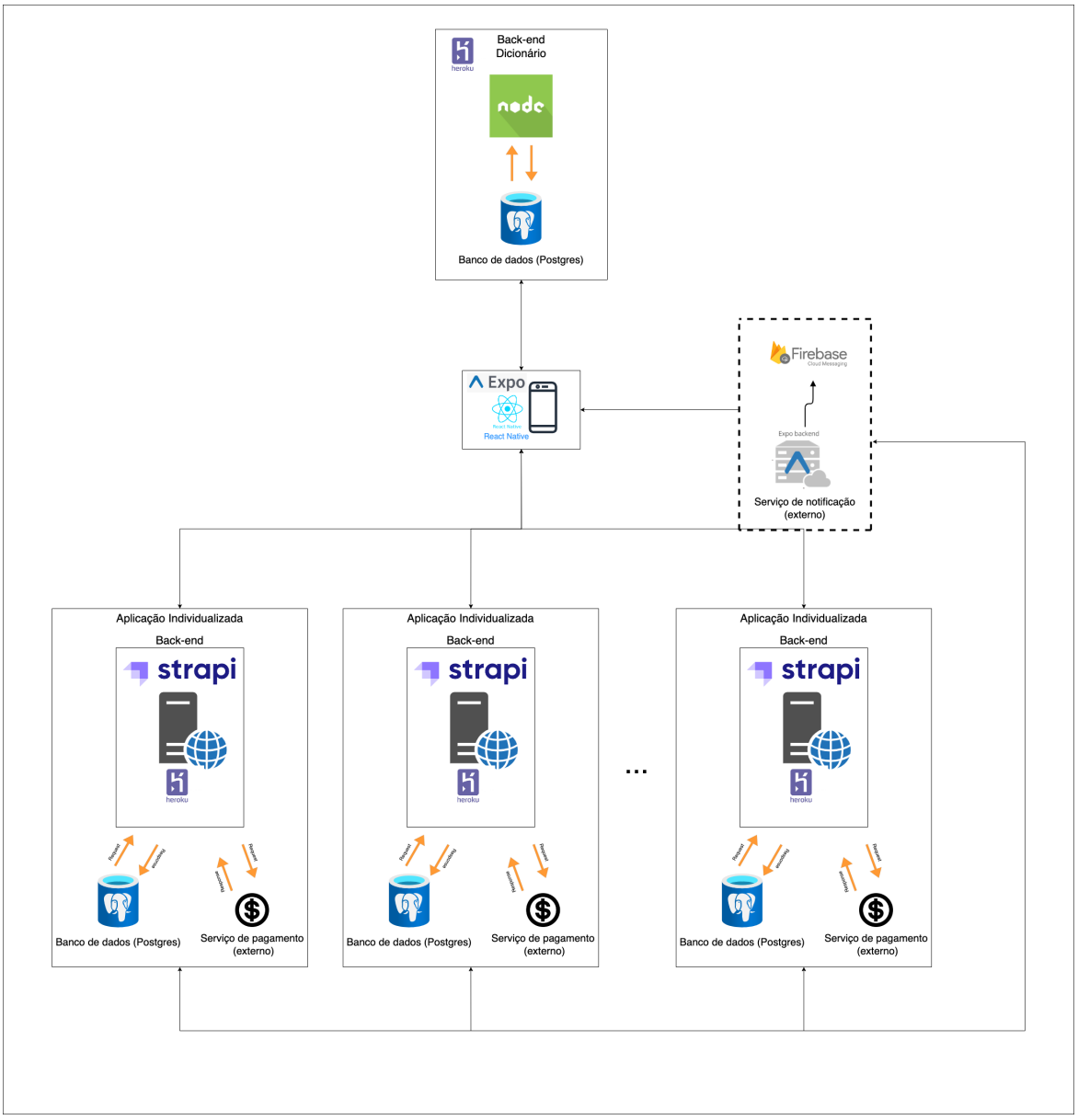
Entretanto é importante frisar que a aplicação Agromart estará disponível apenas para celulares rodando o sistema operacional *Android*. Isso se deve ao custo necessário para disponibilizar uma aplicação na loja virtual de aplicativos iOS.

5.1 Arquitetura

A arquitetura do projeto continuará utilizando o modelo cliente-servidor que segue o modelo de aplicação distribuída. Essa estrutura consegue dividir as tarefas e cargas de trabalho entre os fornecedores de um recurso ou serviço, servidores, e os que requereram esse serviço.

Entretanto, o Agromart não terá mais dois *front-ends*. A aplicação *mobile* se comunicará com um serviço central que servirá de dicionário para requerer a URL base de cada CMS Strapi individualizado. A partir deste momento, a comunicação cliente-servidor entre *app* e Strapi individualizado ocorrerá. Deste modo, cada aplicação Strapi terá seu próprio banco de dados e frontend disponibilizado pelo CMS.

A comunicação entre cliente-servidor será feito por padrão HTTP, que é um protocolo de hipertexto leve e veloz, necessário para sistemas de informação distribuídos e colaborativos.



Fonte: Autor, 2022

Figura 3 – Nova Arquitetura de software proposta

5.2 API REST

API (Application Programming Interface) é um mecanismo que permite que dois componentes de software se comuniquem usando um conjunto de definições e protocolos (AWS, 2023). Já REST (Representational State transfer), se refere a um padrão arquitetural sobre comunicações que ocorrem na WEB. Logo, uma API REST descreve um conjunto de recursos e um conjunto de operações que podem ser chamadas nesses recursos utilizando o padrão REST. As operações em uma API REST são chamadas via um cliente HTTP (SURWASE, 2016).

5.3 Strapi

A escolha do Strapi se deu pelo motivo do *backend* do Agromart já ter sido desenvolvido utilizando essa ferramenta, por isso ao invés de se criar um novo *backend* sendo criado do zero, até por questões de tempo, escolheu-se manter o Strapi, que funciona como um *dashboard*, onde o administrador conseguirá acessar para realizar diversas funções.

O Strapi é um CMS Headless, 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 utilizando seu painel de administração da aplicação (STRAPI, 2021).

5.4 Docker

O Docker é uma plataforma aberta para desenvolvimento, envio e execução de aplicações. O Docker permite que você separe suas aplicações de sua infraestrutura para que você possa fazer deploy de software mais rapidamente. Com o Docker, você pode gerenciar sua infraestrutura da mesma forma que gerencia suas aplicações (DOCKER, 2022).

O Docker permitirá que membros da comunidade *open-source* baixem o repositório e tendo apenas o docker instalado em suas máquinas, consigam testar localmente a aplicação.

5.5 Shell Script

Um shell script é um arquivo de texto que contém uma sequência de comandos para um sistema operacional baseado em UNIX (NEWHAM, 2005). Ele torna capaz a automatização de processos sequenciais que seriam digitados diretamente na linha do terminal.

5.6 Docker-compose

O Docker Compose é uma ferramenta desenvolvida para ajudar a definir e compartilhar aplicação de vários contêineres. Com o Compose, podemos criar um arquivo YAML para definir os serviços e com um único comando. Tendo como sua maior vantagem poder definir sua pilha de aplicações em um arquivo, mantê-la na raiz do repositório do seu projeto e rodar seu projeto inteiro com apenas um comando.(COMPOSE, 2022)

Deste modo, o projeto Agromart inteiro pode ser executado com seu banco de dados e CMS Strapi rodando em containers dockerizados e orquestradas por apenas um arquivo, o docker-compose.yml

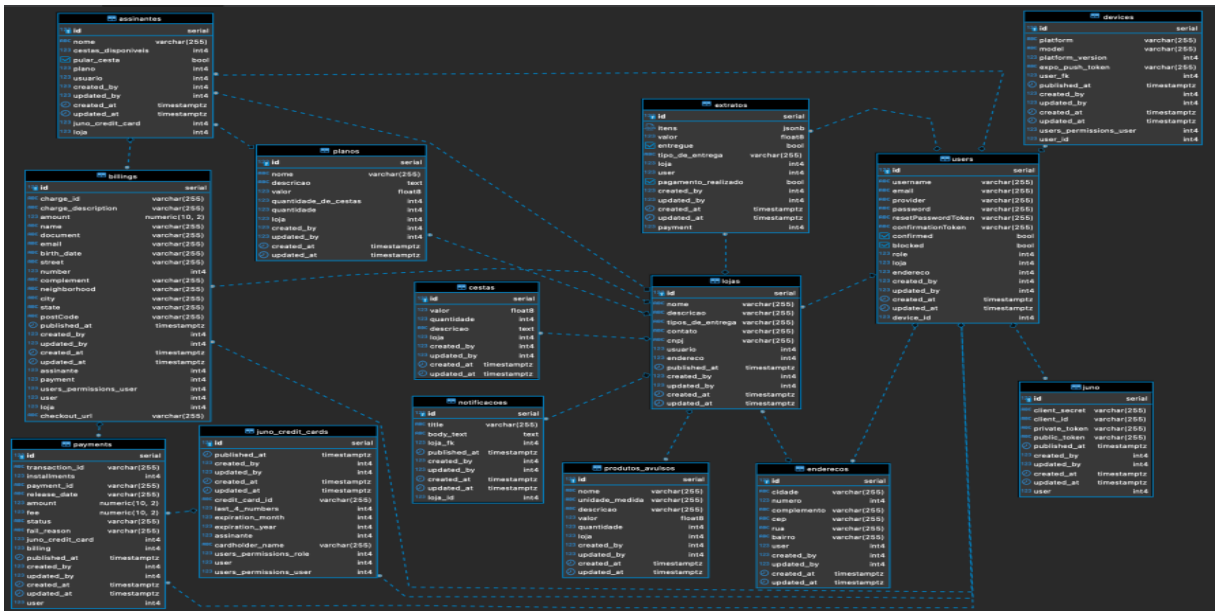
5.7 Banco de Dados

Nos baseando em projetos anteriores, a escolha de um sistema gerenciador de banco de dados continuou sendo o *PostgreSQL*, isso se deve pelo fato do projeto anterior já ter sido adaptado para o seu uso, ou seja, não necessitaria de uma refatoração e remanejamento dos dados que já existem, mas mais do que isso, é a característica *open source* que faz com que essa ferramenta seja mantida, visto que PostgreSQL é atualmente uma alternativa confiável de código aberto para bancos de dados comerciais (PAN, 2022). Sem contar que com o foco em transformar o Agromart em um *software open source* a utilização do *PostgreSQL* torna mais acessível e sem a necessidade de buscar um outro sistema gerenciador de banco de dados.

Além da escolha da ferramenta foi feito também uma análise do DLD, Diagrama Lógico de Dados, cujo objetivo é demonstrar via um diagrama, como as Entidades que representam objetos, pessoas, entre outros do mundo real, se relacionam entre si, além de mostrar quais características aquela entidade possui, essas características chamadas de atributos, são como regras que deverão ser seguidas ao ser feita a implementação do Banco de Dados a nível de código. Dessa forma pode-se verificar se haveria uma necessidade de alterar as bases de dados já criadas ou mantê-las. No caso optou-se por manter as bases criadas, visto que estão implementadas de maneira que possibilita seu uso sem a necessidade da criação de novas bases de dados além de já estar alinhada com a ferramenta *PostgreSQL* que será utilizada na implementação do banco de dados, com essa decisão realizada é possível verificar o diagrama do DLD abaixo.

O *PostgreSQL* é um sistema gerenciador de banco de dados relacional de código aberto que está há anos no desenvolvimento ativo o que fez com que tivesse uma forte reputação nas áreas de confiabilidade, robustez de recursos e desempenho. Essa ferramenta dispõe de uma abundância de recursos destinados a ajudar desenvolvedores em suas tarefas, tais como desenvolvimento de aplicativos, gerenciamento de dados, entre outros.

Além disso é uma ferramenta ,como foi dito anteriormente, de código aberto e gratuita.



Fonte: Byron e Igor, 2022

Figura 4 – Diagrama Lógico de Dados - Agromart

5.8 Heroku

Heroku é uma PaaS (Plataform as Service), ou seja, um ambiente de desenvolvimento e implantação completo na nuvem que permite hospedagem, configuração, testagem e publicação de projetos virtuais. Entre outras funções, ele facilita o trabalho dos desenvolvedores na configuração da infraestrutura para o *deploy*, ou seja, a implantação das aplicações (IMAGINEDONE, 2021).

O Heroku é totalmente gerenciado, dando aos desenvolvedores a liberdade de se concentrar em seu produto principal sem a distração de manter servidores, hardware ou infraestrutura (HEROKU, 2022). No ambiente Heroku, também são oferecidos serviços adicionais como, por exemplo, máquinas virtuais de fácil acesso para aplicação. Deste modo, um banco de dados pode ser criado facilmente ao lado de uma aplicação, sem a necessidade de o desenvolvedor mantê-lo.

5.9 *Continuous Development*

Quando se fala de *Continuous Deployment*, ou CD como é usado, refere-se a automatização de tarefas que são vistas como monótonas ou repetitivas, ou que, são tarefas simple a ponto de poderem ser automatizadas. Dentro do que considera-se o contínuo estão diversas tarefas de diferentes etapas e sub-etapas no ciclo de vida de um *software*. (FITZGERALD; STOL, 2014)

No desenvolvimento do trabalho, isso ocorreu em um diferente etapa do ciclo, sendo ela utilizada na parte de documentação, isso acontece, pois é um trabalho *open source*, ou seja, a criação de um ciclo de tarefas deve ser feito e mantido por alguém, porém o projeto tem o objetivo de ser mantido pela comunidade, logo esse ciclo não seria ideal no momento em que o projeto se encontra, visto que poderia atrapalhar no processo futuro de novas integrações e funcionalidades. O outro motivo é simples, existe uma necessidade de que sempre que houver uma alteração na documentação a mesma ter em sua url os documentos mais recentes, mantendo-a sempre atualizada, porém isso pode ser considerada uma tarefa repetitiva, sem contar que necessitaria de uma pessoa para relizá-la, pensando nisso foi criado um ciclo que realiza dois *scripts* de *deploy*.

Dentro desses *scripts* um seria de teste, que verifica se a página da documentação foi criada e compilada sem nenhum tipo de problema, e outro realmente de *deploys* que fica responsável por realizar a verificação de compilação, e publicar essa página atualizada na url, atualizando esse documento sempre que a *branch* principal sofrer alguma mudança.

5.10 Licença

A licença a ser utilizada no projeto será a *GLP*, isso se deve ao fato da licença ser do tipo recíproca total, o que garante que o projeto será livre e gratuito. Essa licença garante que toda cópia, execução, modificação e redistribuição do projeto deve continuar com o caráter livre (RODRIGUES; MACEDO, 2021)

6 Resultados

6.1 Pré-desenvolvimento

Antes do desenvolvimento do projeto foram realizados algumas atividades, essas atividades serviriam para a dupla entendesse mais sobre o projeto e o conceito que ele trabalha, ou seja, o que de fato é uma CSA e como ela funciona, entender qual seria o objetivo da dupla nessa etapa do projeto e entender qual era a atual situação do projeto. Dessa forma essa pequena seção visa em explicar as atividades realizadas pela dupla em relação ao planejamento e organização dessa nova parte do projeto que viria a ser desenvolvido.

Antes de iniciar o projeto foi realizado uma reunião com o orientador e professor André Lanna, juntamente do co-orientador e professor Rudi van Els, nessa reunião foi explicado a história do Agromart, assim como o que era uma CSA. Com esses conceitos bem colocados foi feita então a discussão e escolha do tema do trabalho, tema esse que focaria na individualização e automatização do Agromart.

Com as escolhas feitas, foi então decidido a realização da visitação a CSA Floresta, a escolha dessa CSA em específico se deve ao fato do Agromart ter surgido como um projeto que a teria como foco de estudo, porém com o passar dos projetos e com a chegada da pandemia, acabou-se perdendo um pouco desse contato, porém a CSA em si se aprimorou ainda mais, tornando-a como uma grande referência. Durante a visitação, foram analisados alguns aspectos, como a forma como ela funcionava, de onde vinham os produtos, que são de agricultores parceiros a CSA, como funcionava sua metodologia de negócio e principalmente, como era feito o controle e administração de tudo isso. Durante essa visitação todas essas perguntas foram respondidas, o que corroboraram para alguns aspectos já implementados no projeto, que não necessitariam de mudança, mas sim de uma manutenção.

Com o foco projeto mais claro e com um entendimento melhor sobre o que se tratava a CSA, foir então executado uma análise do projeto atual, tal análise teve ajuda do Byron Veludo um dos membros que realizou o projeto anteriormente, com ele foi possível comprender mais sobre como rodar o projeto, quais ferramentas eram utilizadas, que problemas ele havia enfrentado, e com que problemas o projeto estaria sofrendo, dessa forma foi compreendido quais possíveis abordagens poderiam ser utilizadas.

Após esse longo caminho foi realizado uma análise do código, assim como uma verificação de quais artefatos já exisitiam. Durante a análise do código forma levantados problemas e suas possíveis soluções, além de uma compreensão ainda maior do que deveria

ser feito.

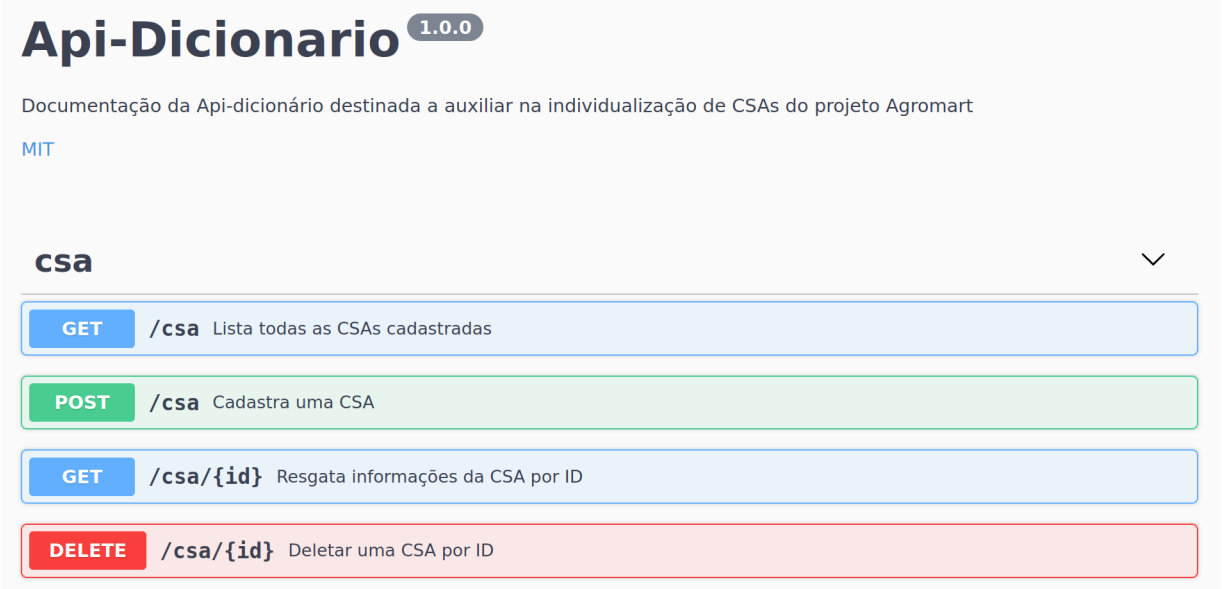
Por fim, a dupla fez a organização de como seriam feito o desenvolvimento, as abordagens que seria seguidas e principalmente o cronograma que deveria cumprido, levando ao restante do trabalho, que será mostrado adiante.

6.2 Api-Dicionário

Tento em vista a motivação de individualçização deste trabalho, para suprir a necessidade de múltiplos ambientes de CSA funcionarem num mesmo aplicativo mobile, foi criada a Api-Dicionário.

Esta aplicação é um serviço REST criado utilizando Node.js junto com o framework Express. É uma Api simples que se comunica com um banco de dados PostgresSql via a ORM squelize para salvar dados básicos sobre cada CSA vinculada ao projeto Agromart. Possuindo apenas 4 rotas de comunicação, é possível criar, deletar e listar CSAs. Tendo em vista que o projeto está inserido na comunidade OpenSource, foi criada uma documentação detalhada de cada rota e seu devido modo de uso utilizando o Swagger. Este, é uma ferramenta que tem o fim de auxiliar a documentação de APIs de forma visual e facilitada, pois toda a informação fica contida em apenas um arquivo json.

Abaixo se encontra a página inicial do documento gerado pelo *swagger*, onde pode-se ver todas a rotas disponíveis pela aplicação junto com uma breve explicação de sua funcionalidade:



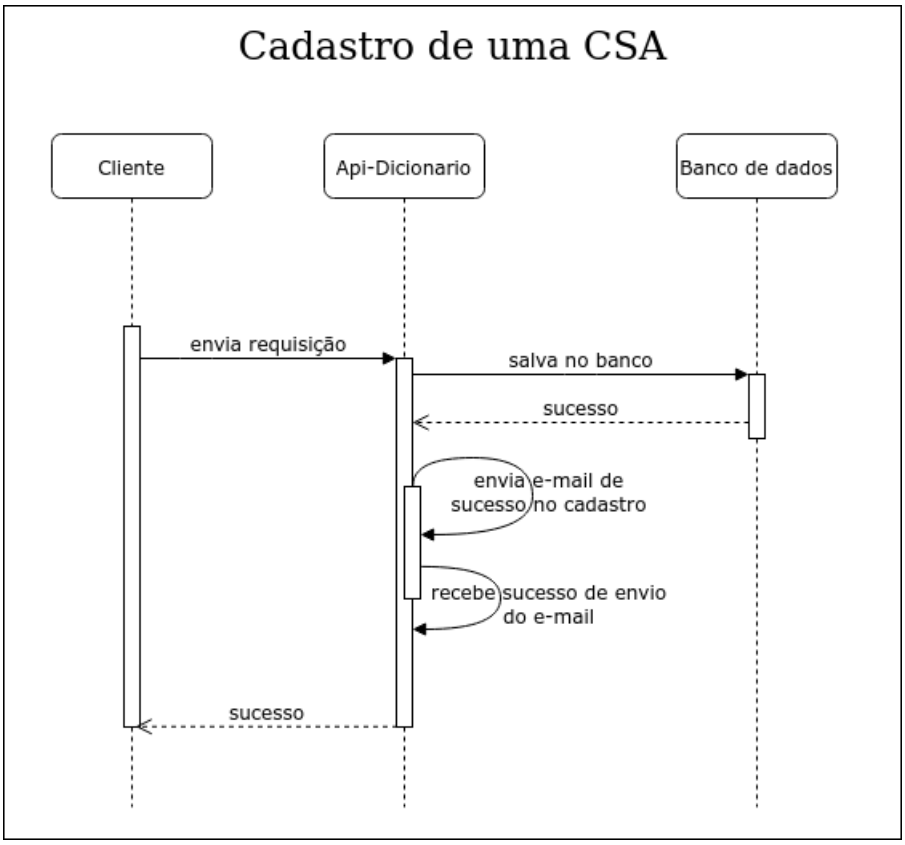
Fonte: Autor, 2022

Figura 5 – Swagger do projeto

6.2.1 Cadastro de uma CSA

É interessante notar que o método de cadastro de uma CSA possui um passo a mais do que apenas inserir no Banco de Dados. Como passo final no cadastro de uma CSA na API dicionário, é enviado um e-mail para responsável da comunidade. Este e-mail contém uma explicação básica sobre o projeto Agromart, com instruções para que o líder desta comunidade consiga explicar para seus co-agricultores como entrar no aplicativo utilizando o ID se sua CSA.

Este fluxo é representado pelo diagrama de sequência abaixo:



Fonte: Autor, 2022

Figura 6 – Diagrama de sequência da Api-Dicionario

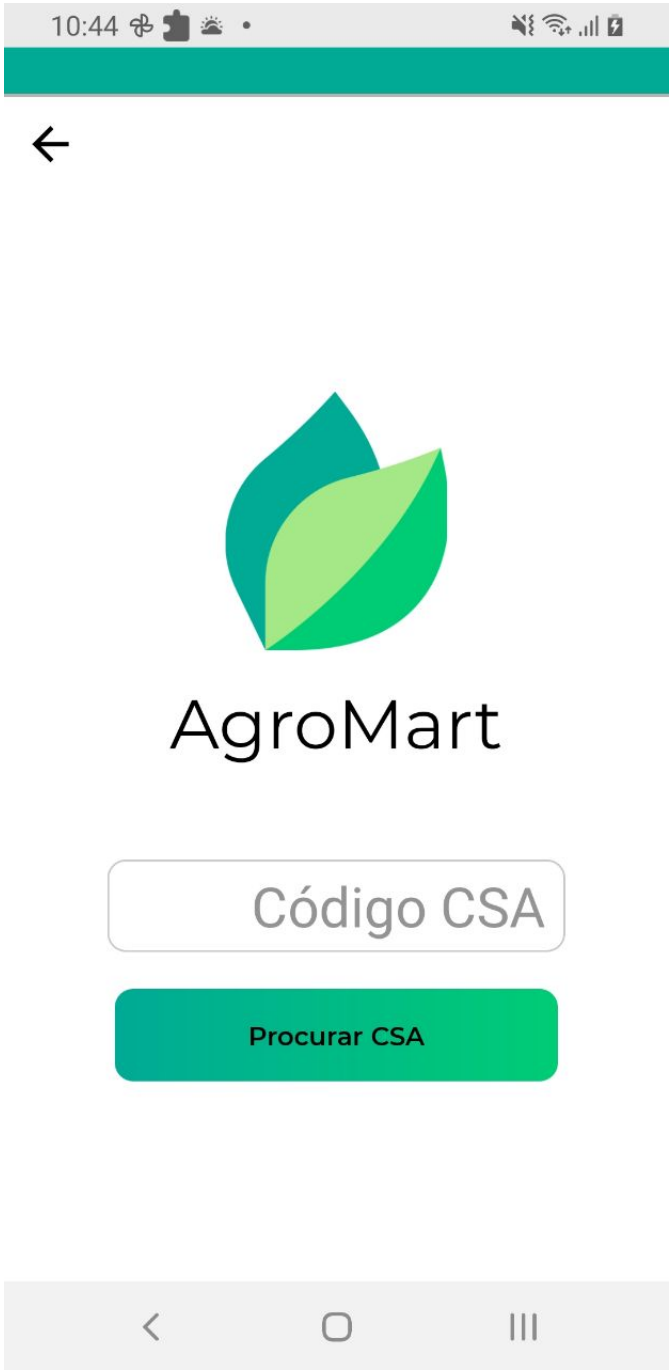
6.3 Aplicação Mobile

Com o intuito de vários usuários poderem utilizar o mesmo aplicativo para acessar diferentes CSAs, o fluxo de autenticação da aplicação mobile precisou ser alterada. Agora, um usuário ao clicar no ícone de *login* é primeiramente direcionado a uma página onde deve fornecer o ID da CSA que ele deseja se vincular.

Caso a CSA com o ID fornecido existir, o usuário é apresentado com a opção de utilizar esta CSA ou procurar por outra.

Com uma CSA escolhida para servir de URL base, o usuário pode de fato logar na aplicação.

Após o login ser efetuado com sucesso, o usuário é redirecionado para a página principal do aplicativo, já recebendo informações da CSA escolhida e podendo realizar seus pedidos de produtos, planos e cestas.



Fonte: Autor, 2022

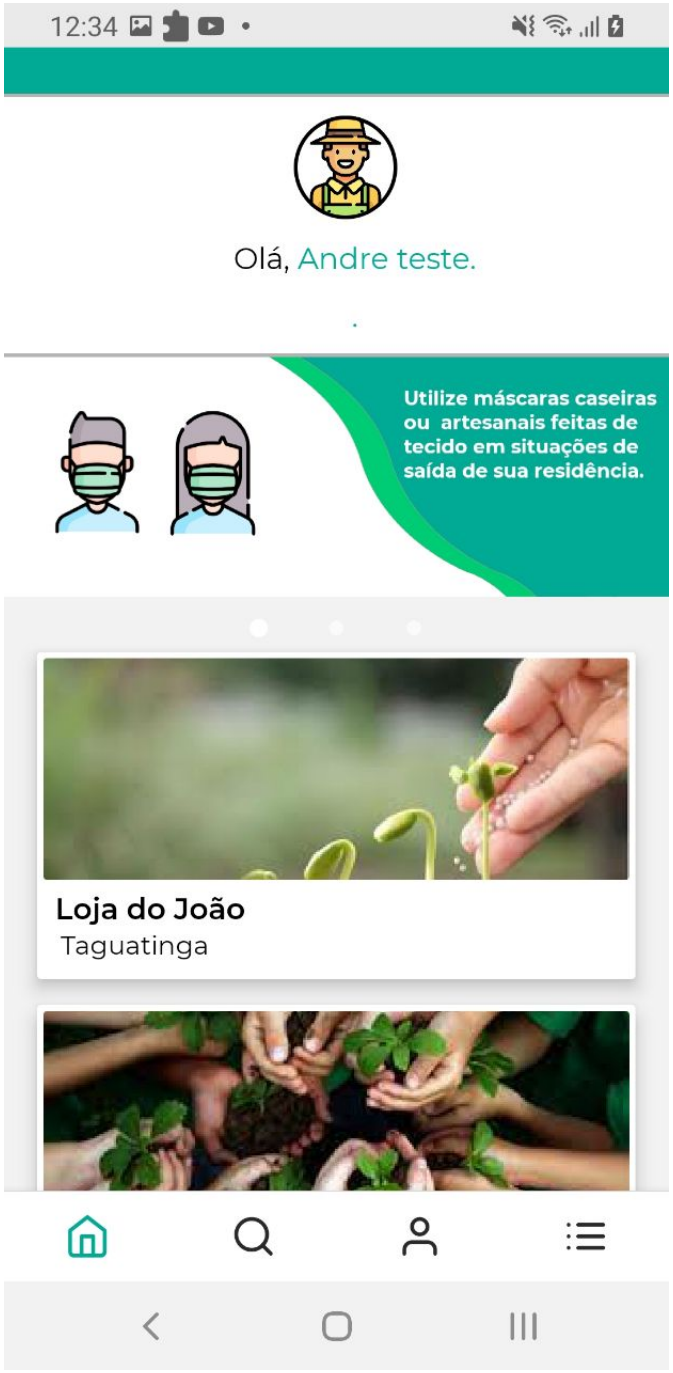
Figura 7 – Aplicativo: Tela de seleção de CSA

Fonte: Autor, 2022

Figura 8 – Aplicativo: Tela de confirmação de escolha de CSA

Fonte: Autor, 2022

Figura 9 – Aplicativo: Tela de Login



Fonte: Autor, 2022

Figura 10 – Aplicativo: Tela inicial após usuário logar

6.4 Automatização

Para cumprir o objetivo de Automatização na instanciação de uma Dashboard Strapi foi criado um script *Python* que, a partir de um arquivo previamente preenchido com dados e credencias do administrador da CSA em questão, fornecerá variáveis de ambiente e executará um *shell script* que por sua vez, possui uma sequência de comandos que farão a aplicação se tornar disponível no Heroku. A plataforma Heroku foi escolhida por sua fácil manutenção e incrível afinidade com o CMS Strapi. Entretanto, a partir do dia 28 de novembro de 2022, a empresa optou por retirar o plano grátis de banco de dados *Heroku Postgres* que o projeto utiliza. Como o atual trabalho de conclusão de curso já estava sendo executado, o mesmo foi mantido, porém, pago.

Atualmente o código possui os seguintes pre requisitos:

- Possuir uma conta no Heroku
- Possuir um meio de pagamento cadastrado no Heroku
- Heroku CLI instalado localmente
- Git instalado localmente

O *scrpit* pode ser resumido nos seguinte passos:

- Ler o arquivo de variáveis
- Exportar variáveis no ambiente local
- Logar no Heroku CLI
- Criar uma aplicação no Heroku
- Vincular um *Heroku Postgres* a aplicação
- Definir variáveis de ambiente no escopo da aplicação
- Fazer o *deploy* do código para o Heroku
- Fazer o *build* em modo de produção da aplicação
- Enviar para a Api-Dicionário os dados da nova CSA disponível

6.5 Documentação

No início do desenvolvimento do trabalho, foi visto uma preocupação quanto à forma que seria passada a informação para quem iria usar do Agromart futuramente, quando falamos dos possíveis usuários, nos referimos aos responsáveis por uma CSA e possivelmente desenvolvedores que ou continuam no processo de desenvolvimento por meio de trabalhos da UnB ou que se interessaram pelo objetivo do projeto e tentam contribuir no meio *open source* através de novas funcionalidades ou melhorias no código.

Foi pensando nesse tipo de abordagem que foi feito a documentação do Agromart, olhando os trabalhos anteriores, verificando a existência de artefatos, realizando uma análise no código, entre outras atividades, foi levantado quais artefatos seriam mais significativos, dessa forma desenvolvedores futuros teriam acesso ao processo das tomadas de decisão que foram realizadas, além da forma como funciona o Strapi e a aplicação *mobile* em si.

6.5.1 Artefatos

Para a criação dos artefatos, foi feita a escolha com base nas etapas de Requisitos de Software, nessa escolha foi feito uma análise sobre a etapa em que o projeto se encontrava, afinal algumas etapas e técnicas não se encaixariam, pensando nisso foram realizadas as etapas de Elicitação de Requisitos, Modelagem e Análise. Com essas fases seria possível demonstrar o processo de criação das ideias feito pela dupla, analisando quais funcionalidades estariam ou não nesse projeto, a parte de modelagem seria responsável por ilustrar essa decisão inicial, evidenciando o que foi mantido e a forma como o sistema funcionaria, assim como o ator a quem são atribuídos as atividades. Por fim temos a parte de análise, ela foi colocada para evidenciar o trabalho realizado na documentação, mas principalmente ela tem um carater avaliativo, sendo essa crítica feita pela própria dupla, em que são evidenciados pontos a serem melhorados nos artefatos, dessa forma quem for continuar o trabalho poderá ver possíveis sugestões de mudança.

Dentre os artefatos selecionados de cada etapa, foram feitos:

- **Elicitação de Requisitos**
  - **5W2H:** É um *Framework* utilizado para criar um plano de ação em um curto período de tempo, de maneira eficiente. Foi utilizado tal ferramenta para se ter uma visão inicial acerca do projeto, com base nas 7 perguntas, *What?* (O quê?), *Why?* (Porque?), *Who?* (Quem?), *Where?* (Onde?), *Wuen?*(Quando?), *How?*(Como?), *How Much?* (Quanto?). Dessa forma a medida que foram feitas as perguntas, foi se tornando ainda mais claro sobre o que de fato se trata o projeto, e um pouco da forma que ele funciona.

- **Storytelling:** É uma técnica que visa mostrar a forma como determinado sistema funciona através de uma história. O uso dessa técnica é interessante, pois ele visa ilustrar uma história que gira em torno do sistema a ser desenvolvido, dessa forma, pessoas que não são área de desenvolvimento podem entender de uma maneira mais simples, apenas relacionando a apliação com a história contada.

  A escolha dessa técnica se deu por conta dos futuros usuários do Agromart, que são os responsáveis pelas CSAs, muitos deles não são da área de tecnologia, então ter uma abordagem mais simples se tornou um bom objetivo, pois dessa forma, caso eles queiram entender mais sobre como alguma parte do sistema funciona, basta eles acessarem a documentação.

- **Introspecção:** A introspecção é uma técnica em que o desenvolvedor se põe no lugar do usuário para entender melhor quais seriam suas necessidades com a aplicação. No caso do projeto, existem dois usuários, que é o Administrador, que seria o responsável pela CSA, e o Co-agricultor, que seria o consumidor, logo a introspecção juntamente com a análise dos trabalhos anteriores nos fez levantar alguns novos requisitos.

  Essa técnica foi extremamnte importante para entendermos quais funcionalidades já existiam previamente e deveriam ter alguma manutenção, e quais seria novas funcionalidades de fato. Essa dúvida se deve ao fato da técnica de introspecção ser utilizada nas fases inicias do projeto, ou seja, quando não se tem nada, porém a utilização da mesma ilustrou melhor o que deveria ser feito e o que já tinha sido feito e deveria ser modificado dentro do contexto do projeto.

- **MoSCoW:** O MoSCoW de acordo com (MARTHASARI; SUHARSO; ARDIANSYAH, 2018): *"É um método de priorização de requisitos baseado no custo, risco e valor do negócio"*. Com ele, são utilizados os termos que indicam se determinado requisito será abarcado no projeto ou não. Possui as seguintes categorias:

  1. MUST - Prioridade mais alta
  2. SHOULD
  3. COULD
  4. WILL NOT - Prioridade mais baixa

  Apesar das outras técnicas já trazerem a ideia do que seria implementado ou não, a utilização dessa técnica de priorização trouxe uma ordem de prioridade ao que deveria ser implementado e o que deveria sofrer com uma manutenção, dessa forma criou-se uma ordem de prioridade em realação as funcionalidades, o que cooperou para a progressão do projeto em si.

- **Modelagem**

- **BPMN:** O *Business Process Model and Notation* ou BPMN como é mais conhecido, é um diagrama que tem como objetivo mostrar o funcionamento de um negócio utilizando ícones que facilitam a compreensão do processo pelo usuário. Usualmente ele é utilizado na área de negócios para ilustrar a ideia inicial ao mesmo em que conversa, através do diagrama, com aqueles que não são da área de tecnologia. Foi utilizado no projeto para ilustrar de uma generalizada, a forma como o sistema funcionaria, ao mesmo tempo em que demonstra a comunicação entre a aplicação *mobile* e o Strapi.

- **Casos de Uso:** O caso de uso é forma de modelagem que serve para ilustrar a do software ou dos requisitos levantados, é uma ferramenta útil para ilustrar o funcionamento da ideia e serve para analisar os possíveis erros que ele poderá ter. Por conta dessa modelagem, ela acaba sendo uma ótima ferramenta para se mostrar a alguém que não é da área de Tecnologia da Informação, visto que ela tem como objetivo demonstrar como funciona o *software*, ou seja, essa ferramenta ilustra os possíveis caminhos que o usuário poderá fazer.

  A utilização dessa técnica de modelagem é de extrema importância, como foi dito anteriormente, os futuros usuários do Agromart, não são da área de Tecnologia, dessa forma utilizando-se dos casos de uso, ficará mais fácil deles entenderem a forma como o sistema funciona, assim como quais caminhos são possíveis de se seguir na utilização do aplicativo/Strapi.

- **Cenários:** Essa técnica se assemelha muito ao *Storytelling* mostrado anteriormente, isso se deve ao fato dos cenários muitas vezes descreverem previamente as histórias que serão contadas ou vice-versa, no caso essa técnica visa descrever de forma mais textual as possíveis situações que o usuário poderá passar ao interagir com o sistema, e o que se espera dessa situação.

  A utilização dessa técnica foi escolhida por complementar mais a técnica de *Storytelling*, no caso muitas vezes a história em si não consegue ser interpretada pelo usuário, para mitigar essa situação utiliza-se os cenários, que servirão como uma instrução acerca da história contada anteriormente.

- **Histórias de Usuário:** As Histórias de Usuário seguem a mesma perspectiva da Introspecção, no caso cria-se, através de uma linguagem mais informal, possíveis necessidades do usuário, sua utilização se torna diferente, quando aplicada ao final do desenvolvimento, onde o objetivo é de aperfeiçoar foram abordados anteriormente, ou trazer novos requisitos.

  Essa técnica foi utilizada para aperfeiçoar mais os requisitos levantados, durante sua criação muitos requisitos acabaram se repetindo, porém outros novos acabaram surgindo, dessa forma a aplicação da técnica serviu para entender possíveis novas necessidades que o usuário teria, agora com a aplicação já mais

desenvolvida e encaminhando-se para a entrega final.

- **Análise**
  - **Verificação e Validação:** Essas técnicas trabalham juntas, porém de forma independente, e visam validar para entender so o produto atendeu a finalidade esperada.

    Usando da verificação foi feito a validação dos artefatos levantados na parte de modelagem, essa escolha se baseia no fato desses artefatos já se referirem ao produto final, diferente da parte de elicitação que tem como foco entender o que é o produto e quais são suas funcionalidades e regras. Já a validação visa validar com o responsável pelo projeto, seja ele uma empresa ou um *stakeholder*, para avaliar se o produto atinge a finalidade pela qual ele foi criado, usualmente, tem-se um protótipo e uma declaração externa, ou seja, uma declaração de algum responsável que não está na área de desenvolvimento.

    Como dito ateriormente, a utilização de ambas as técnicas serve para mostrar aos futuros desenvolvedores que utilizarão desse projeto, para entender, possíveis mudanças ou melhorias nos artefatos, assim como entender a que ponto chegou o projetoe quais resultados ele obteve, assim como na validação que serve para entender o que os responsáveis acharam e ver o atual protótipo da aplicação.

6.5.2 Docussaurus

O Docussaurus é um gerador de site-estático *open source* mantido pela Meta Verso, essa ferramenta é capaz de criar páginas *web* sem a necessidade de um conhecimento de desenvolvimento, seus documentos são escritos utilizando de *MarkDown* que é uma linguagem simples de marcação, que realiza uma conversão do seu texto para HTML.

No trabalho foi escolhida essa ferramenta por ela já ter sido utilizada anteriormente pelos grupos anteriores, além de ter um aprendizado fácil em relação ao seu uso, com a utilização de pastas para separar os arquivos em tópicos e subtópicos, além de permitir a criação de outros documentos, sendo eles utilizando de *React* ou apenas usando de *MarkDown*. Além da parte da criação, o *deploy* também pode ser realizado de uma maneira simples, tendo uma liberdade de escolha em relação ao local de *deploys* e podendo utilizar de *continuous development* ou não.

Durante a criação da documentação, todos os artefatos apresentados anteriormente forma criados e incluidos no Docussaurus, criando-se uma *branch* separada para realizar essas atuallizações, assim como a criação de um processo de CD para a realização do teste de *deploy* e do *deploy* final que será disponibilizado na URL e postado no Github Pages, uma ferramenta que permite a hospedagem da página criada.

7 Considerações Finais

Este Trabalho de Conclusão de Curso (TCC) tem como objetivo realizar a individualização do *software* Agromart, tornando único a forma como cada CSA irá administrar seus produtos e co-agricultores. A realização desse trabalho se deve ao fato do sistema ainda estar muito dependente da entidade associada a ele, que é a UnB, por isso um dos enfoques, também, é acabar com essa dependência direta, até porque, a universidade estava arcando com o suporte, manutenção e principalmente, em manter a infraestrutura do projeto rodando.

Durante o TCC1 foi idealizado a forma como seria organizado e planejado a parte de desenvolvimento que seria executado no TCC2, com isso as etapas desenvolvidas no TCC1 geraram em torno também da compreensão acerca do trabalho e do tema pela dupla, realizando contatos com a CSA Floresta, entendendo mais do projeto com o orientador e co-orientador, realizando a análise da forma como o projeto se apresentava, ou seja, entendendo o que foi feito, as ferramentas que o mesmo utilizava e principalmente como funcionava. Todas essas atividades marcaram o funcionamento do TCC1, para o TCC2 foi realizado a implementação da proposta inicial.

Quando é falado o que foi implementado, pode-se simplificar, mencionando os principais aspectos do trabalho, sendo eles a individualização, a automatização e a documentação, essas três atividades tiveram papéis importantes no desenvolvimento do projeto, isso porque a individualização traria a capacidade do projeto de se desvincular da UnB, tornando-o de fato um projeto *open source*, a automatização viria para facilitar a instalação e uso do projeto para que usuários que não tem um conhecimento da área de tecnologia pudessem utilizar de forma rápida e sem a necessidade de um conhecimento aprofundado, com a documentação, que serve para explicar o funcionamento do projeto, e como foi realizado a tomada de decisões para criar o resultado obtido, assim como a disponibilização de tutoriais para facilitar o uso do projeto para os futuros usuários. É importante ressaltar que a documentação visou não utilizar de conceitos de engenharia reversa, onde é um conceito que quando utilizado no ambiente de Tecnologia muitas vezes visa em criar artefatos, funcionalidades, entre outros, em cima de um projeto já pronto, no caso do projeto a documentação foi criada com embasamento em artefatos que já tinham sido criados anteriormente, mas que serviriam como base para documentar os novos artefatos que seriam criados a partir do trabalho gerado, logo o foco da dupla foi poder criar artefatos novos que ilustrassem a situação atual do projeto, ao em que ao mesmo tempo, utilizavam artefatos e funcionalidades criados anteriormente.

Durante a execução do atual trabalho, o gateway de pagamento JUNO sofreu

por mudanças em sua forma de operar e devido a isso, ele optou por não possuir mais pagamentos via cartão de crédito. Tendo em vista que o aplicativo tinha como *feature* o pagamento via cartão de crédito, foi necessária uma escolha baseada em escopo. Como o atual trabalho estava no escopo de arquitetura e funcionamento, e não no escopo em melhorias, foi decidido que tudo que envolvesse pagamentos seria contornado, a fim de deixar a aplicação funcional mesmo sem tal *feature*. Assim, as compras atualmente devem ser feitas pelo app e recebem uma *flag* de "não paga" inicialmente. Logo, quando o cliente for presencialmente buscar seu produto, ele realiza o pagamento e o Administrador da CSA altera o *status* do pedido para "pago".

Outro grande obstáculo encontrado foi a atualização de componentes visuais no React Native. Devido ao tempo de entre os Trabalhos de conclusão de curso, alguns pacotes utilizados foram notados como depreciados. Assim, foi necessário um estudo sobre as dependências dos pacotes utilizados pela aplicação até que chegasse em uma versão estável e onde o componente visual funcionasse da maneira correta.

Em sua totalidade o projeto foi realizado como esperado e proposto pela dupla, tendo sido benéfico no quesito de aprendizado de novas ferramentas, tecnologias e metodologias, além de ter contado com o apoio dos professores, orientadores e co-orientadores para o desenvolvimento e revisão do mesmo, e dos antigos estudantes que contribuíram para uma melhor compreensão das tarefas a serem realizadas.

Apêndices

APÊNDICE A – Primeiro Apêndice - Descrição da documentação

A.1 Introdução

Esse apêndice tem como objetivo demonstrar as realizações feita na parte de documentação, sendo mostrado o estado atual do Docussaurus bem como as páginas geradas com os artefatos gerados.

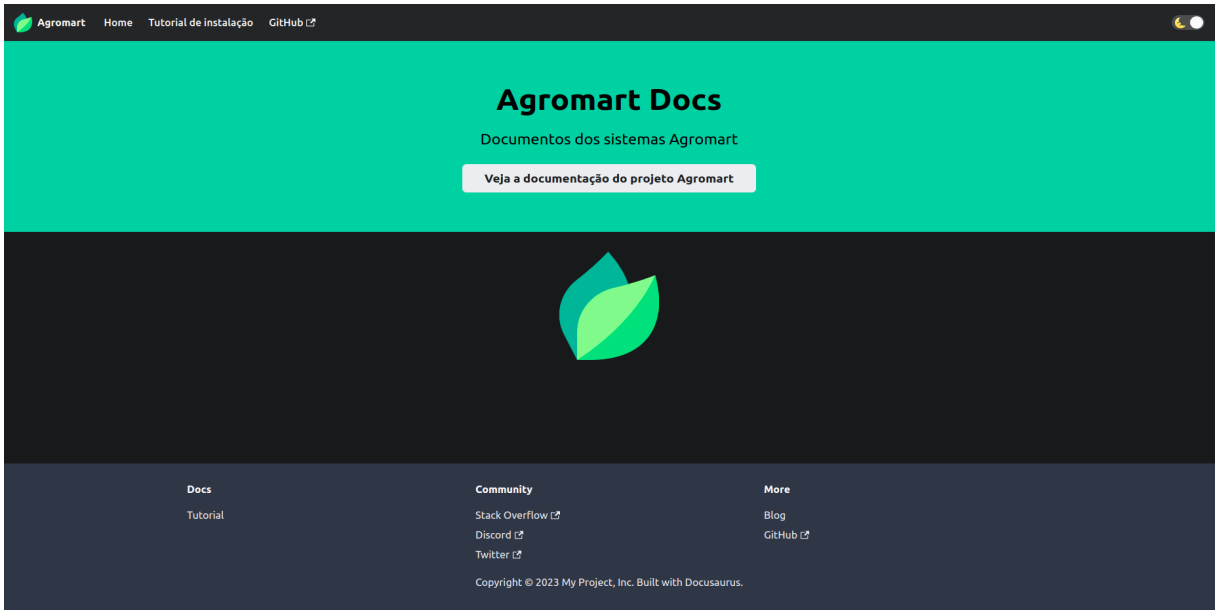
A documentação apresentada nesse apêndice pode ser encontrada no link

A.1.1 Siglas

BMPN - *Business Process Model and Notation*

A.1.2 Home

Como muitos sites, a página *Home* serve para explicar mais sobre o projeto em si, a ponto de abordar a história do Agromart, e a forma como o projeto funciona em parceria com a UnB.



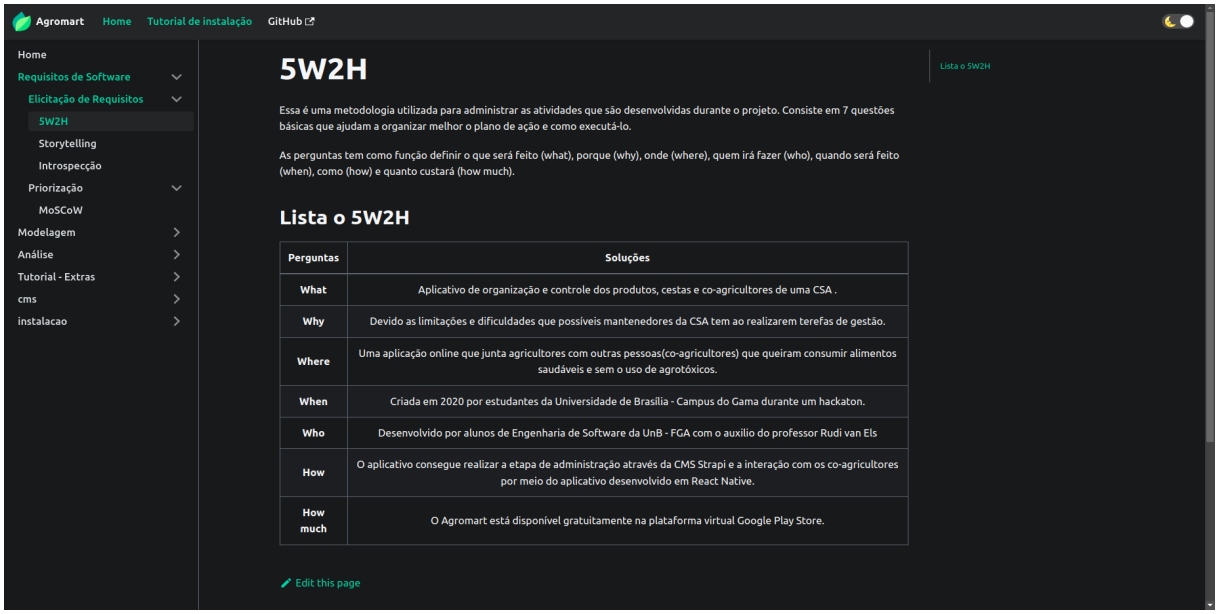
Fonte: Autor, 2023

Figura 11 – Página Home da Documentação

A.1.3 Requisitos de *Software*

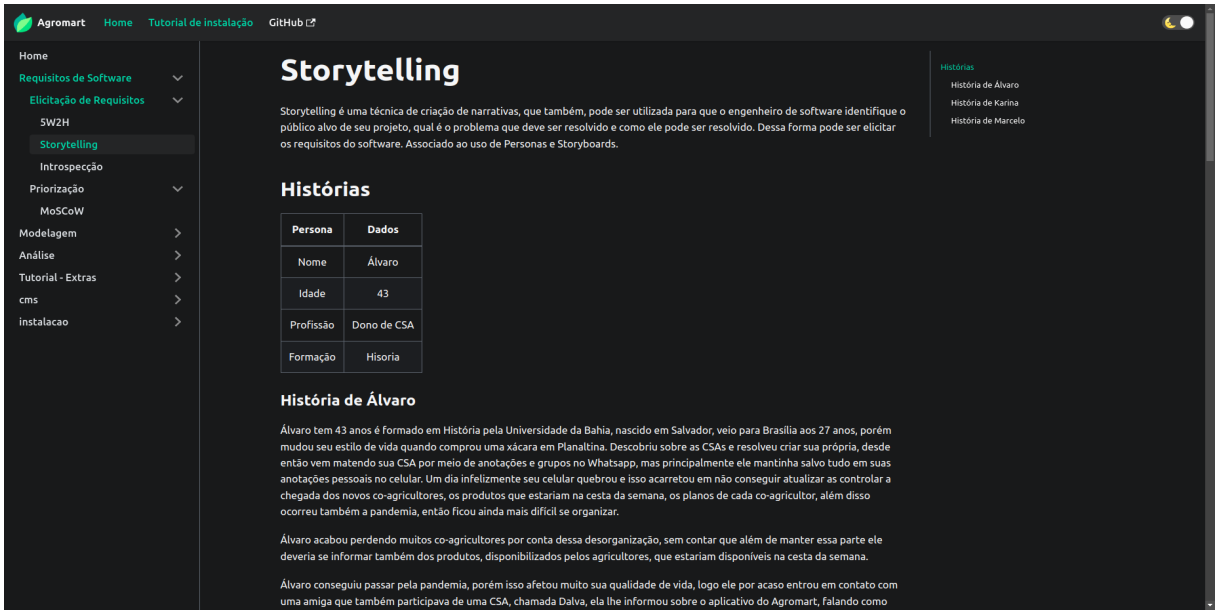
Esse tópico se refere a etapa de elicitação de requisitos, onde são abordadas as técnicas utilizadas pela equipe no quesito de compreensão do projeto, verificando as funcionalidades que sofreriam com as manutenções. Além dessa etapa existe a etapa de Priorização que mostra uma ordem de prioridade em relação as funcionalidades já implementadas ou que devem ser implementadas em prol do avanço do objetivo do projeto.

A.1.3.1 Elicitação de Requisitos



Fonte: Autor, 2023

Figura 12 – Página de 5W2H da Documentação



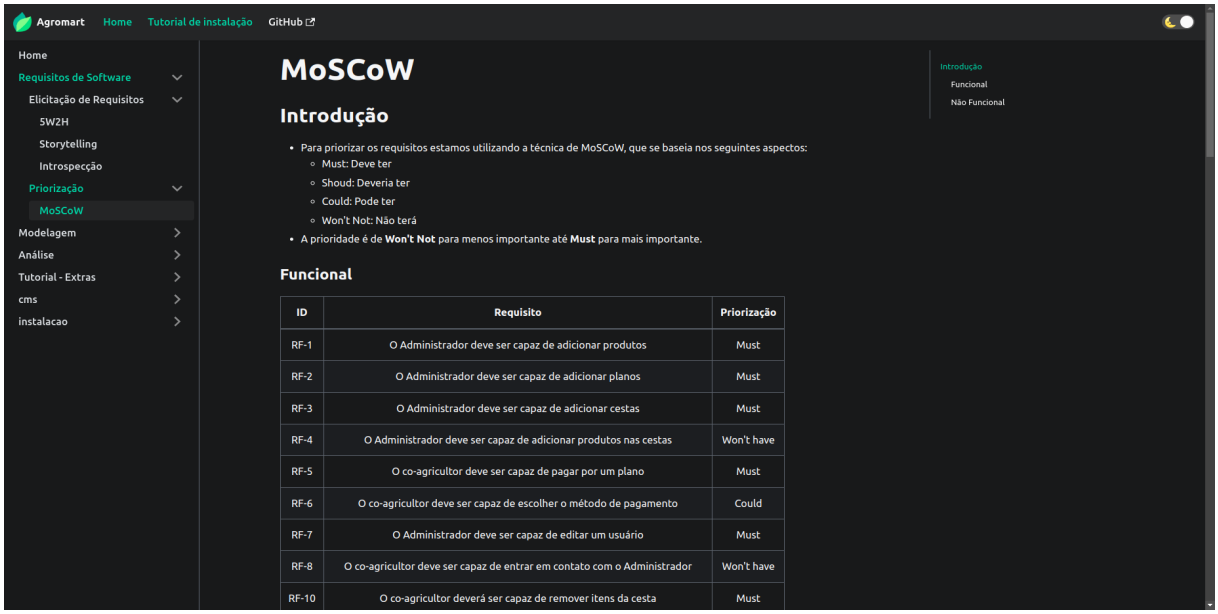
Fonte: Autor, 2023

Figura 13 – Página de Storytelling da Documentação

Fonte: Autor, 2023

Figura 14 – Página de Introspecção da Documentação

A.1.3.2 Priorização



ID	Requisito	Priorização
RF-1	O Administrador deve ser capaz de adicionar produtos	Must
RF-2	O Administrador deve ser capaz de adicionar planos	Must
RF-3	O Administrador deve ser capaz de adicionar cestas	Must
RF-4	O Administrador deve ser capaz de adicionar produtos nas cestas	Won't have
RF-5	O co-agricultor deve ser capaz de pagar por um plano	Must
RF-6	O co-agricultor deve ser capaz de escolher o método de pagamento	Could
RF-7	O Administrador deve ser capaz de editar um usuário	Must
RF-8	O co-agricultor deve ser capaz de entrar em contato com o Administrador	Won't have
RF-10	O co-agricultor deverá ser capaz de remover itens da cesta	Must

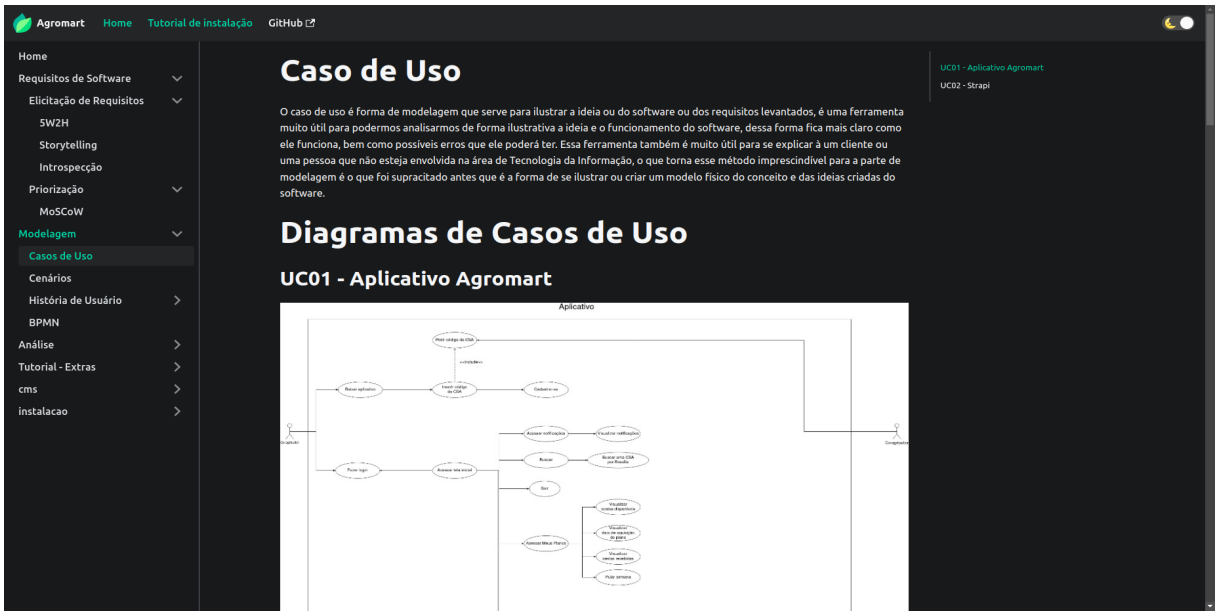
Fonte: Autor, 2023

Figura 15 – Página de Priorização em MoSCoW da Documentação

A.1.4 Modelagem

O processo de modelagem visa diagramar tudo o que foi feito na parte de elicitação, dessa forma serão mostrados os artefatos resultantes desse nível.

A.1.4.1 Caso de Uso



Fonte: Autor, 2023

Figura 16 – Página de Casos de Uso da Documentação

A.1.4.2 Cenários

Fonte: Autor, 2023

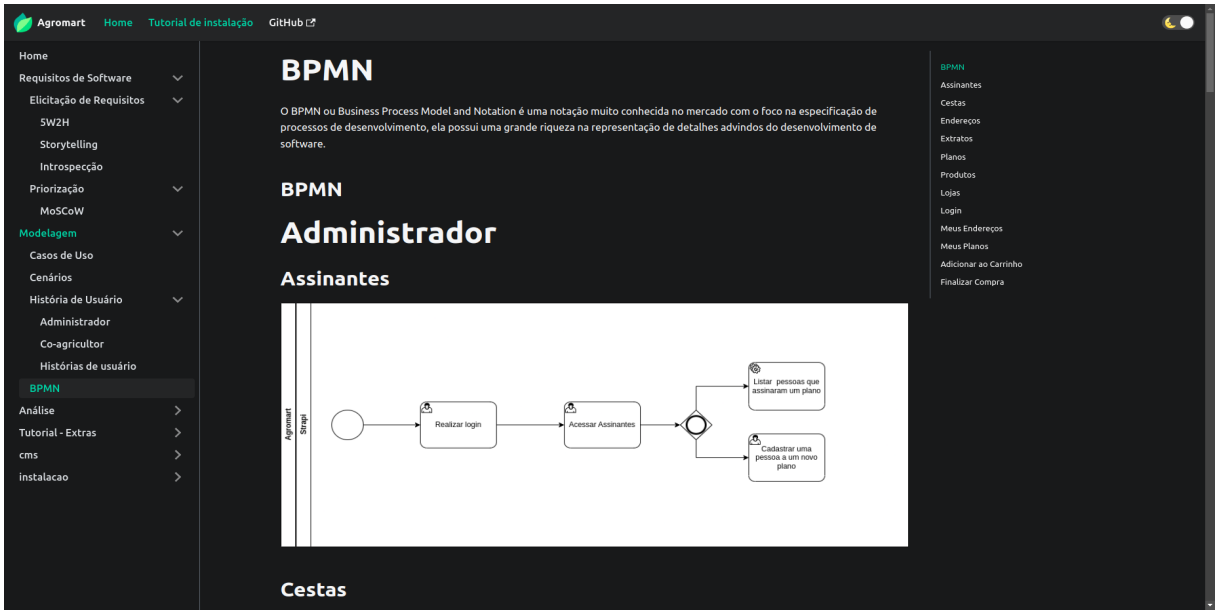
Figura 17 – Página de Cenários da Documentação

A.1.4.3 Histórias de Usuário

Fonte: Autor, 2023

Figura 18 – Página de Histórias de Usuário da Documentação

A.1.4.4 BPMN



Fonte: Autor, 2023

Figura 19 – Página de BPMN da Documentação

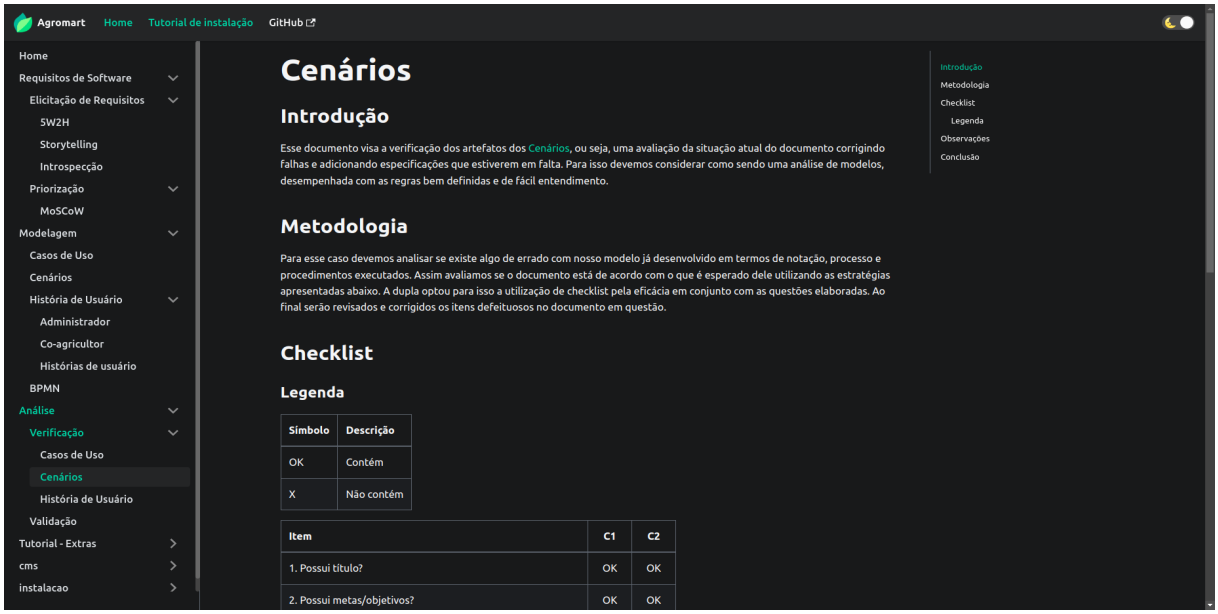
A.1.5 Análise

O processo de analise serve para realizar as etapas de verificação e validação que servem para que os próprios desenvolvedores avaliem se o projeto criado cumpriu com sua finalidade, além de ocorrer uma avaliação dos aretafos criados para que possa ser entendido pontos a serem melhorados.

A.1.5.1 Verificação

Fonte: Autor, 2023

Figura 20 – Verificação dos Casos de Uso



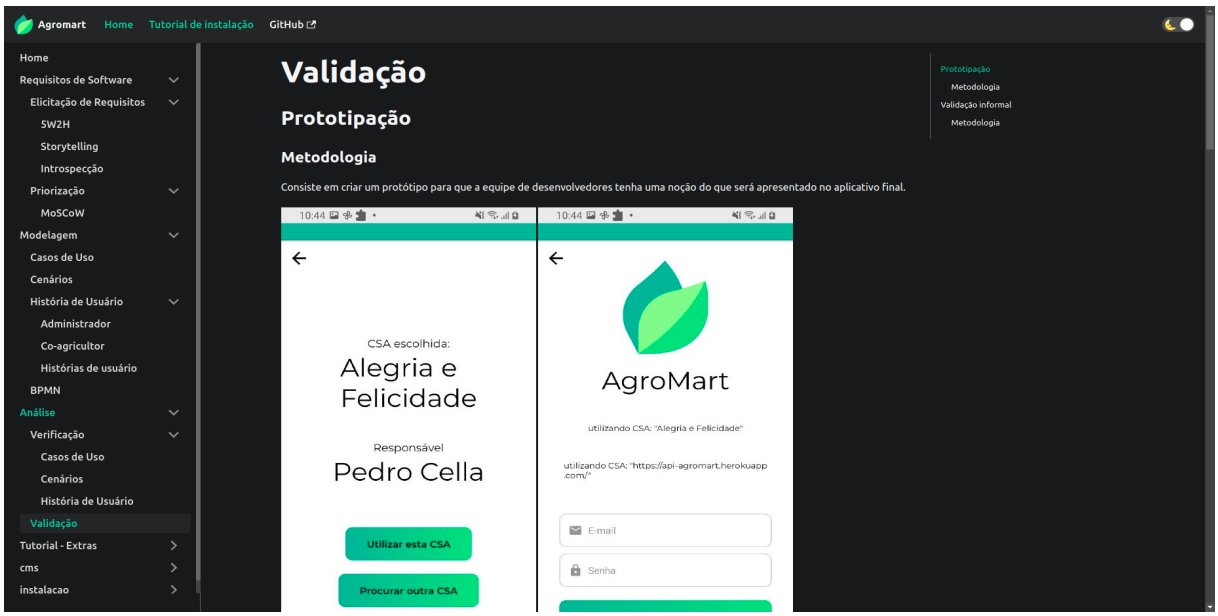
Fonte: Autor, 2023

Figura 21 – Verificação dos Cenários

Fonte: Autor, 2023

Figura 22 – Verificação das Histórias de Usuário

A.1.5.2 Validação



Fonte: Autor, 2023

Figura 23 – Validação das Histórias de Usuário

Referências

AHMAD, M. O.; MARKKULA, J.; OIVO, M. Kanban in software development: A systematic literature review. In: *2013 39th Euromicro Conference on Software Engineering and Advanced Applications*. [S.l.: s.n.], 2013. p. 9–16. Citado na página 20.

ARCANGELI, J.-P.; BOUJBEL, R.; LERICHE, S. Automatic deployment of distributed software systems: Definitions and state of the art. *Journal of Systems and Software*, v. 103, p. 198–218, 2015. ISSN 0164-1212. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0164121215000308>. Citado na página 24.

AWS. *O que é uma API?* 2023. Disponível em: <https://aws.amazon.com/pt/what-is/api/>. Citado na página 34.

BEAL, V. *Software Environment*. 2005. Disponível em: <"https://www.webopedia.com/definitions/software-environment/#What_is_a_Software_Environment">. Citado na página 24.

BRASILEIRO, G. da Manutenção Evolutiva do S. R. D. T. U. P. E. Rafael villar oliveira, raphael alves da silva. 2018. Disponível em: <bdex.eb.mil.br>. Citado na página 19.

CAVALCANTE, R. C.; FILHO, J. R. de F. Inserção da manutenabilidade no gerenciamento dos projetos complexos: Proposta de modelagem utilizando a metodologia fel. Citado na página 15.

COHEN, D.; LINDVALL, M.; COSTA, P. Agile software development. *DACS SOAR Report*, Citeseer, v. 11, p. 2003, 2003. Citado na página 20.

COMPOSE, D. *Use Docker Compose*. 2022. Disponível em: <https://docs.docker.com/get-started/08_using_compose/>. Citado na página 35.

CORREA, B. K. B.; VELUDO, I. G. Proposta para gestão de co-agricultores e meios de pagamentos na agricultura familiar: uma evolução do agromart. 2021. Citado 2 vezes nas páginas 14 e 29.

CRUZ, F. *Scrum e PMBOK unidos no Gerenciamento de Projetos*. [S.l.]: Brasport, 2013. Citado na página 22.

DISCORD. *Sobre o Discord*. 2022. Disponível em: <https://discord.com/company>. Citado na página 27.

DOCKER. *Docker overview*. 2022. Disponível em: <https://docs.docker.com/get-started/overview/>. Citado na página 34.

DOCS, M. W. *O que é JavaScript?* 2022. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Learn/JavaScript/First_steps/What_is_JavaScript>. Citado na página 32.

EISENMAN, B. *Learning React Native: Building Native Mobile Apps with JavaScript (English Edition)*. [S.l.]: O'Reilly Media, Inc., 2017. Citado na página 32.

ESPINDOLA AZRIEL MAJDENBAUM, J. L. N. A. Rodrigo Santos de. Uma análise crítica dos desafios para engenharia de requisitos em manutenção de software. 2004. Disponível em: <http://www.metodoiron.com.br/iron/wp-content/uploads/2014/03/Artigo_EngReq_ManutencaoSoftware.pdf>. Citado na página 19.

EXPRESS. *Express*. 2022. Disponível em: <https://expressjs.com/pt-br/>. Citado na página 32.

EXTREME PROGRAMMING. *Extreme Programming: A gentle introduction*. 2013. Acessado em 13 de Outubro de 2021. Disponível em: <"http://www.extremeprogramming.org/">. Citado na página 21.

FITZGERALD, B.; STOL, K.-J. Continuous software engineering and beyond: trends and challenges. In: *Proceedings of the 1st International Workshop on Rapid Continuous Software Engineering*. [S.l.: s.n.], 2014. p. 1–9. Citado na página 36.

GUILHOTO, J. J. M. et al. A importância da agricultura familiar no brasil e em seus estados (family agriculture's gdp in brazil and in it's states). 2017. Disponível em: <https://ssrn.com/abstract=2408072orhttp://dx.doi.org/10.2139/ssrn.2408072>. Citado na página 13.

HEROKU. *What is Heroku?* 2022. Disponível em: <https://www.heroku.com/about>. Citado na página 36.

IBM. *Planning your development and production environments*. 2022. Disponível em: <"https://www.ibm.com/docs/da/case-manager/5.3.2?topic=system-planning-your-development-production-environments">. Citado na página 24.

IMAGINEDONE. *O que é o Heroku e como a ferramenta revolucionou o desenvolvimento escalável?* 2021. Disponível em: <https://imaginedone.com.br/blog/o-que-e-o-heroku/>. Citado na página 36.

LABELLE, D. *"What Is ZenHub? Detailed ZenHub Overview & Explanation Of ZenHub Features"*. 2019. Disponível em: <https://thedigitalprojectmanager.com/tools/zenhub-overview/>. Citado na página 28.

LIM, S.-H. Experimental comparison of hybrid and native applications for mobile systems. *International Journal of Multimedia and Ubiquitous Engineering*, v. 10, n. 3, p. 1–12, 2015. Citado na página 23.

MARTHASARI, G.; SUHARSO, W.; ARDIANSYAH, F. A. Personal extreme programming with moscow prioritization for developing library information system. *Proceeding of the Electrical Engineering Computer Science and Informatics*, v. 5, n. 1, p. 537–541, 2018. Citado na página 47.

MEIRELES, T. *Você já Ouviu Falar na comunidade que sustenta a Agricultura?* 2018. Disponível em: <"https://www.wwf.org.br/?65282FCSA-Comunidade-que-Sustenta-a-Agricultura">. Citado na página 13.

MORESI, E. et al. Metodologia da pesquisa. *Brasília: Universidade Católica de Brasília*, v. 108, n. 24, p. 5, 2003. Citado na página 16.

MYLONAS, T. *"What are environments in software development? A guide to the development, beta, and production environments."*. 2017. Disponível em: <"https://codebots.com/app-development/what-are-environments-in-software-development-a-guide-to-the-development-beta-and-production-e
Citado na página 24.

NAIK, U.; SHIVALINGAIAH, D. Software open source para sistemas de gerenciamento de conteúdo (open source software for content management system). 2022. Disponível em: <https://ir.inflibnet.ac.in:8443/ir/bitstream/1944/1028/1/29.pdf>. Citado na página 23.

NEWHAM, C. *Learning the bash shell: Unix shell programming*. [S.l.]: "O'Reilly Media, Inc.", 2005. Citado na página 34.

ORG, N. *About Node.js*. 2022. Disponível em: <https://nodejs.org/en/about/>. Citado na página 32.

PAN, M. L. Z. C. S. Desenvolvimento de um sistema gerenciador de banco de dados paralelo ao básico do sgbd postgresql open-source (development of a parallel database management system on the basis of open-source postgresql dbms). 2022. Disponível em: <http://www.mathnet.ru/links/a8543b03291636795cb014a41cbe5a1e/vyuru62.pdf>. Citado na página 35.

PIRES, E. *TypeScript – O superset de JavaScript*. 2012. Disponível em: <https://www.eduardopires.net.br/2012/10/typescript/>. Citado na página 32.

RODRIGUES, L. S.; MACEDO, L. P. Inovações tecnológicas na agricultura familiar: Agromart). 2021. Citado 2 vezes nas páginas 30 e 37.

SCHWABER, K.; SUTHERLAND, J. Scrum. *Siehe: http://www. scrum. org/Resources/What-is-Scrum*, 2010. Citado na página 22.

SILVA, J. B. da; ANASTÁCIO, F. A. de M. Método kanban como ferramenta de controle de gestão. *ID on line. Revista de psicologia*, v. 13, n. 43, p. 1018–1027, 2019. Citado na página 20.

SILVA, M. A. A. M. d. *Aplicação do Earned Value Management na Metodologia Scrum*. Tese (Doutorado), 2017. Citado na página 20.

SITEWARE. *Metodologias de Gestão: Otimize processos com a metodologia Kanban*. 2016. Disponível em: <https://www.siteware.com.br/metodologias/metodologia-kanban/>. Citado na página 21.

SOARES, M. d. S. Metodologias Ágeis extreme programming e scrum para o desenvolvimento de software. 2004. Citado na página 26.

SOMMERVILLE, I. *Software Engineering*. 9. ed. Harlow, England: Addison-Wesley, 2010. ISBN 978-0-13-703515-1. Citado 2 vezes nas páginas 20 e 22.

SPRIESTERSBACH, A.; SPRINGER, T. Quality attributes in mobile web application development. In: BOMARIUS, F.; IIDA, H. (Ed.). *Product Focused Software Process Improvement*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004. p. 120–130. ISBN 978-3-540-24659-6. Citado na página 23.

STRAPI. . *Sobre Strapi*. 2021. Disponível em: <https://strapi.io/documentation/developer-docs/latest/getting-started/introduction.html>. Citado na página 34.

SURWASE, V. Rest api modeling languages-a developer's perspective. *Int. J. Sci. Technol. Eng*, v. 2, n. 10, p. 634–637, 2016. Citado na página 34.

TELES, V. M. *Extreme Programming: Aprenda como encantar seus usuários desenvolvendo software com agilidade e alta qualidade*. [S.l.]: Novatec Editora, 2017. Citado na página 21.

TIWARI, S.; RATHORE, S. S.; GUPTA, A. Selecting requirement elicitation techniques for software projects. In: IEEE. *2012 CSI Sixth International Conference on Software Engineering (CONSEG)*. [S.l.], 2012. p. 1–10. Citado na página 25.

VARGAS, A. A. F.; PEREIRA, J. V. d. S. Gerenciamento da manutenção de software. Citado na página 15.

WINTERS, T.; MANSHRECK, T.; WRIGHT, H. *Software Engineering at Google*. [S.l.: s.n.], 2020. ISBN 9781492082798. Citado na página 19.

WWF. *Taís Meireles*. 2018. Disponível em: <https://www.wwf.org.br/?65282/CSA-Comunidade-que-Sustenta-a-Agricultura>. Citado na página 14.