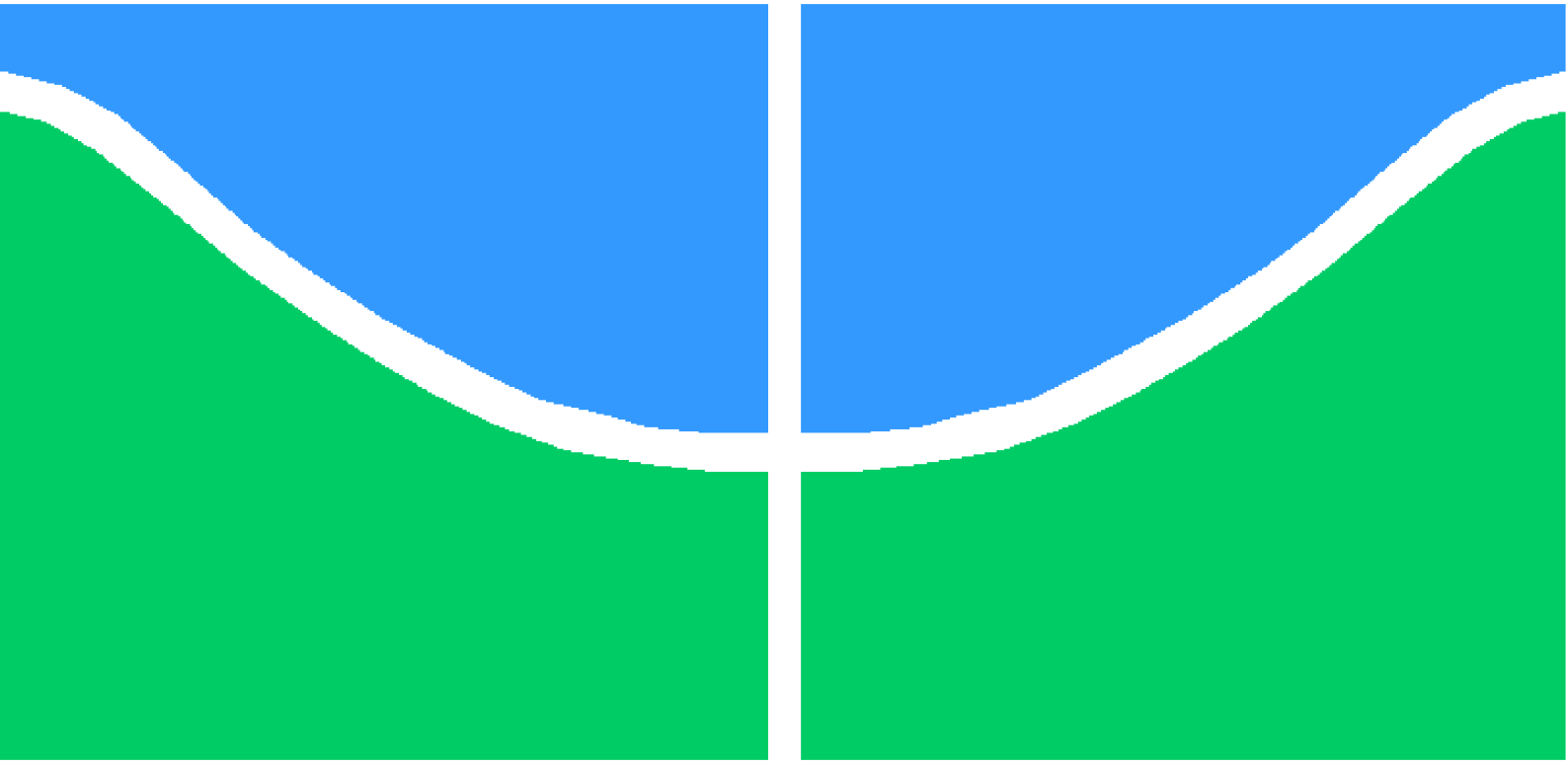


Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Engenharia de Software

Módulo de integração de pagamento e manutenção do software Agromart

Autores: Giovanna Borges Bottino e Felipe Boccardi Silva Agustini

Orientador: Prof. André Luiz Peron Martins Lanna

Co-orientador: Prof. Rudi Henri van Els

Brasília, DF

2023

Giovanna Borges Bottino e Felipe Boccardi Silva Agustini

Módulo de integração de pagamento e manutenção do software Agromart

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

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Orientador: Prof. André Luiz Peron Martins Lanna

Coorientador: Prof. Rudi Henri van Els

Brasília, DF

2023

Giovanna Borges Bottino e Felipe Boccardi Silva Agustini
Módulo de integração de pagamento e manutenção do software Agromart/ Giovanna Borges Bottino e Felipe Boccardi Silva Agustini. – Brasília, DF, 2023-
103 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. André Luiz Peron Martins Lanna
Coorientador: Prof. Rudi Henri van Els

Trabalho de Conclusão de Curso – Universidade de Brasília - UnB
Faculdade UnB Gama - FGA , 2023.

1. Agricultura familiar. 2. AgroMart. I. Prof. André Luiz Peron Martins Lanna. II. Prof. Rudi Henri van Els. III. Universidade de Brasília. IV. Faculdade UnB Gama. V. Módulo de integração de pagamento e manutenção do software Agromart

CDU 004.4:005.64

Giovanna Borges Bottino e Felipe Boccardi Silva Agustini

Módulo de integração de pagamento e manutenção do software Agromart

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

Trabalho aprovado. Brasília, DF, 28 de julho de 2023:

Prof. André Luiz Peron Martins Lanna
Orientador

Prof. Rudi Henri van Els
Co-orientador

Profa. Cristiane Soares Ramos
Convidado 1

Prof. Ricardo Ajax Dias Kosloski
Convidado 2

Brasília, DF
2023

Este trabalho é dedicado a todos os produtores de laranjas. Através do trabalho de vocês essa fruta nos uniu.

Agradecimentos

Eu, Giovanna Borges Bottino, sou grata por estudar na Universidade de Brasília. Agradeço à todas as mulheres na minha vida e ao meu pai. Minha mãe, Andréa Borges, por ter me apoiado e me oferecido diversas oportunidades de crescimento. Minha irmã, Isabella Maria Bottino, por mostrar o caminho de como ser melhor. Meu pai, por ser minha referência de superação. E principalmente a todos os professores que falaram que engenharia não é para mulheres: vocês me incentivaram.

Eu, Felipe Boccardi Silva Agustini, agradeço por todos aqueles que me ajudaram a trilhar o caminho que percorri. Agradeço aos meus professores por terem me guiado pela estrada do conhecimento, a minha família e amigos pelo apoio e incentivo que me ofereceram para estar aqui, e por estarem comigo nos momentos mais difíceis.

"Pedras no caminho? Eu guardo todas. Um dia vou construir um castelo.
(Nemo Nox)

Resumo

O AgroMart é um software que serve para auxiliar comercialização de cestas de produtos agroecológicos cultivados por agricultores familiares. Ele visa facilitar a relação entre os pequenos agricultores e os co-agricultores com uma interface web para os agricultores e um aplicativo mobile multiplataforma. Esta proposta pretende fazer a manutenção e a criação de um módulo de integração de pagamentos para o AgroMart. A manutenção refere-se a melhoria da segurança através da atualização dos pacotes encontrados na API feita em Strapi CMS. Já o módulo de integração de pagamento, um serviço em Node Express que se comunicará com os gateways de pagamento Mercado Pago, PagSeguro e PayPal escolhidos por uma matriz de decisão. Assim espera-se contribuir para AgroMart com evolução, manutenção e documentação.

Palavras-chaves: AgroMart. Comunidades que Sustentam a Agricultura. Manutenção de Software

Abstract

AgroMart is a software that serves to help commercialize baskets of agroecological products cultivated by family farmers. It aims to facilitate the relationship between small farmers and co-farmers with a web interface for farmers and a cross-platform mobile application. This proposal intends to maintain and create a payment integration module for AgroMart. Maintenance refers to improving security by updating the packages found in the API made in Strapi CMS. The payment integration module, a Node Express service that will communicate with Mercado Pago, PagSeguro and PayPal payment gateways chosen by a decision matrix. Thus, it is expected to contribute to AgroMart with evolution, maintenance and documentation.

Key-words: AgroMart. Communities That Support Agriculture. Software maintenance.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

API	Interface de programação de aplicações
APP	Aplicações Mobile
CD	Entrega contínua
CI	Integração contínua
CSA	Comunidade que Sustenta Agricultura
HTML	Linguagem de Marcação de HiperTexto
HTTP	Protocolo de Transferência de Hipertexto
IBGE	Instituto Brasileiro de Geografia e Estatística
IEEE	Instituto de Engenheiros Elétricos e Eletrônicos
MVP	Produto Mínimo Viável
PIB	Produto Interno Bruto
SGBD	Sistema Gerenciador de Banco de Dados
TCC	Trabalho de Conclusão de Curso
UNB	Universidade de Brasília
UML	Linguagem de Modelagem Unificada
URL	Localizador uniforme de recursos
XP	eXtreme Programming

Sumário

1 Introdução

Este capítulo de introdução tem como propósito apresentar os principais elementos que constituem a estrutura desta monografia, proporcionando ao leitor uma visão geral e norteando-o através dos principais aspectos abordados ao longo do trabalho.

Será Contextualizado o domínio de interesse deste trabalho Comunidades que Sustentam a Agricultura (CSA) e o Agromart. Seguido pela Justificativa para a realização deste Trabalho, para esclarecer as contribuições paro o Agromart e as CSAs. Apresentado a Objetivos Geral e Específicos, finalizado com a Organização desse trabalho.

1.1 Contextualização

A agricultura familiar desempenha um papel significativo na produção agropecuária brasileira. Conforme os dados do censo conduzido pelo IBGE (2017), cerca de 23% de toda a produção nacional é proveniente da agricultura familiar. Correspondendo a aproximadamente 80,9 milhões de hectares de área de produção e contribui com cerca de 107 bilhões de reais para o Produto Interno Bruto (PIB) brasileiro.

Anteriormente, até meados da década de 90, os agricultores familiares enfrentavam dificuldades no acesso a meios eficientes de escoamento e planejamento produtivo, o que limitava a otimização do retorno financeiro de suas atividades. Como resultado, muitas dessas famílias encontraram nas Comunidades que Sustentam a Agricultura (CSAs) uma forma de alcançar esse objetivo (SCHNEIDER, 2003).

As CSAs são associações estabelecidas por meio de parcerias comerciais entre agricultores e consumidores, visando estreitar as relações entre eles, transformando os consumidores em co-produtores. Dessa forma, eles assumem riscos e benefícios da produção, investindo em um sistema sustentável de produção de alimentos saudáveis e livres de agrotóxicos (TORUNSKY; NETO; AMORIM, 2015).

Segundo Torres (2017), as CSAs se baseiam em princípios de parceria. Em algumas dessas comunidades, as ferramentas de gestão utilizadas incluem planilhas e a rotatividade de membros da equipe de articulação. No entanto, essas atividades administrativas podem se tornar complicadas.

Com o intuito de atuar como intermediário entre os pequenos agricultores e os co-produtores, foi desenvolvida a solução de software Agromart. Essa aplicação planeja facilitar a interação entre os membros associados, permitindo que as CSAs melhorem o escoamento da produção dos agricultores, fornecendo alimentos mais saudáveis e de qualidade aos co-produtores (RODRIGUES; MACÊDO, 2021).

1.2 Justificativa

O Agromart tem em vista facilitar a comunicação entre produtores e co-agricultores, auxiliar no planejamento do escoamento da produção agrícola e oferecer ferramentas de gerenciamento para as CSAs. Ele proporciona aos administradores das comunidades uma forma de preservar os dados durante as trocas administrativas (RODRIGUES; MACÊDO, 2021).

Durante seu desenvolvimento, a aplicação passou por diversas mudanças. Inicialmente, Rodrigues e Macêdo (2021) iniciaram o desenvolvimento do software com o objetivo de criar uma solução capaz de intermediar a comunicação entre agricultores e co-agricultores, além de oferecer uma forma de visualização e solicitação de produtos agroecológicos por meio de um aplicativo móvel.

Corrêa e Veludo (2022) adaptaram o projeto para um formato de software livre e adicionaram ferramentas de pagamento e notificação. Posteriormente, Freitas e Cella (2023) propuseram a criação de uma versão personalizada e automatizada da aplicação para permitir que as CSAs disponibilizem sua própria versão do programa.

Apesar dos desenvolvimentos anteriores, o software ainda não atende a algumas necessidades de gestão das CSAs devido à descontinuação do módulo de pagamento, o gateway Juno. Esse módulo é essencial para a gestão financeira da aplicação, pois sem ele não é possível realizar pagamentos por meio da Interface de Programação de Aplicações (API), controlar as entradas de caixa e criar o extrato financeiro.

Substituir um módulo descontinuado por um novo módulo atende às necessidades dos usuários, aproveita avanços tecnológicos, corrige problemas conhecidos, integra-se a sistemas modernos e fornece suporte contínuo. Essa abordagem garante que o software esteja alinhado com as demandas e expectativas dos usuários, permitindo que eles aproveitem ao máximo as funcionalidades e recursos oferecidos. Portanto, é necessário desenvolver um novo plugin gerenciador de APIs de pagamento para suprir essa necessidade de um novo gateway de pagamento e evitar problemas semelhantes no futuro.

Além disso, o Strapi no projeto Agromart está na versão 3.4.6, uma versão antiga que eventualmente será descontinuada, o que significa que o suporte técnico não estará mais disponível. Manter o software na versão mais recente garante maior eficiência, funcionalidades aprimoradas e compatibilidade com as tecnologias em constante evolução.

A solução proposta neste Trabalho de Conclusão de Curso (TCC) visa realizar a manutenção do Strapi do AgroMart e implementar ajustes no sistema de pagamento digital. A manutenção envolve a atualização dos componentes da aplicação atual e a remoção de componentes não utilizados. Já os ajustes no sistema de pagamento digital incluem a criação de um integrador de APIs de pagamento.

1.3 Objetivo Geral

O objetivo principal deste trabalho é realizar a manutenção evolutiva do produto de software Agromart, por meio da atualização dos módulos e do aprimoramento das ferramentas de gestão financeira.

1.3.1 Objetivos Específicos

Para alcançar o propósito do projeto, pretende-se cumprir os seguintes objetivos específicos:

- Atualizar os frameworks utilizados para suas versões mais recentes, garantindo assim o uso de tecnologias atualizadas e aproveitando possíveis melhorias e funcionalidades adicionais;
- Remover módulos não utilizados, eliminando qualquer componente ou funcionalidade que não seja mais necessário, a fim de simplificar o sistema e melhorar sua eficiência;
- Implementar um integrador de pagamento digital, desenvolvendo uma funcionalidade que permita a geração de uma URL de pagamento por meio de uma API, facilitando a gestão financeira e possibilitando transações seguras e eficientes;
- Implementar um gerenciador de extratos financeiros, criando uma ferramenta que permita a visualização dos registros financeiros, auxiliando na análise das transações financeiras;
- Esquematizar e documentar a solução proposta, fornecendo uma descrição clara e completa da solução desenvolvida, incluindo diagramas, fluxos de processo e documentação técnica, facilitando o entendimento e o uso adequado do software.

Ao alcançar esses objetivos específicos, espera-se que o Agromart seja atualizado e aprimorado, proporcionando uma melhor experiência aos usuários e atendendo de forma mais eficaz às necessidades de gestão financeira das CSAs.

1.4 Organização da Monografia

O trabalho é composto por cinco capítulos principais e consiste na seguinte estrutura:

- **Capítulo 1 - Introdução:** O objetivo deste capítulo é fornecer os principais conceitos e esclarecimentos do que será abordado neste trabalho. Apresenta aspectos que serão fundamentais para o entendimento e contexto do projeto.

- **Capítulo 2 - Referencial Teórico:** O referencial teórico apresenta os levantamentos e pesquisas principais relacionadas aos assuntos/temas abordados neste trabalho. Para embasar as abordagens e discussões propostas ao projeto e desenvolvimento deste trabalho.
- **Capítulo 3 - Suporte Tecnológico:** Contém detalhes sobre as ferramentas e outros suportes utilizados.
- **Capítulo 4 - Metodologia:** A metodologia trata da organização da pesquisa a ser feita. Este capitulo especifica as escolhas metodológicas.
- **Capítulo 5 - Agromart:** este capítulo tem o objetivo de contextualizar o Projeto Agromart e seus componentes ao apresentar alguns dos os artefatos gerados pela metodologia.
- **Capítulo 6 - Resultado:** este capítulo descreverá os resultados alcançados no desenvolvimento da metodologia.
- **Capítulo 7 - Considerações Finais:** Por fim, as considerações finais tem o objetivo de apresentar os resultados obtidos ao longo do desenvolvimento deste projeto, e suas expectativa após a implementação.

2 Referencial Teórico

Neste capítulo, serão apresentadas as bases teóricas necessárias para compreender a manutenção e a evolução do Agromart. O objetivo é facilitar o entendimento dos termos utilizados e fornecer uma base conceitual para os tópicos abordados. As seguintes áreas serão abordadas: Engenharia de Software, Lean Inception, Práticas ágeis, Manutenção Evolutiva, Aplicações Web, Sistemas de Gestão de Conteúdo, Pagamento Digital e Testes. Por fim, serão apresentadas as considerações finais do capítulo, resumindo os principais pontos discutidos e estabelecendo a base teórica necessária para a compreensão dos tópicos subsequentes.

2.1 Engenharia de Software

Segundo Rajlich (2011), *software* é um produto muito diferente de outros produtos por estar entre os sistemas mais complexos já criados pela humanidade. Graças ao aumento da complexidade do *software*, a comunidade desenvolveu a consciência de que o *software* requer treinamento e habilidades especializadas. Essa consciência deu origem à engenharia de *software*.

A engenharia de *software* é uma disciplina que lida com dificuldades essenciais de complexidade, mutabilidade, conformidade e descontinuidade do *software*. Na história dessa ciência trabalhou-se com três paradigmas: improvisado, cascata, iterativo. Atualmente o paradigma mais utilizado é baseado em iterações, apoiada em períodos em que os programadores trabalham de acordo com um plano de iteração (RAJLICH, 2011).

O desenvolvimento de *software* iterativo e evolutivo foi a resposta à volatilidade. Os requisitos mudam constantemente, mas os engenheiros não podem operar em completo caos e não podem mudar constantemente as funcionalidades. Em vez de planejar todo o ciclo de vida, eles planejam por um período limitado, chamado de iteração. Os programadores pegam a versão atual dos requisitos, preparam um plano por tempo limitado e quando ficam desatualizados replanejam a próxima iteração (RAJLICH, 2011).

2.2 Lean Inception

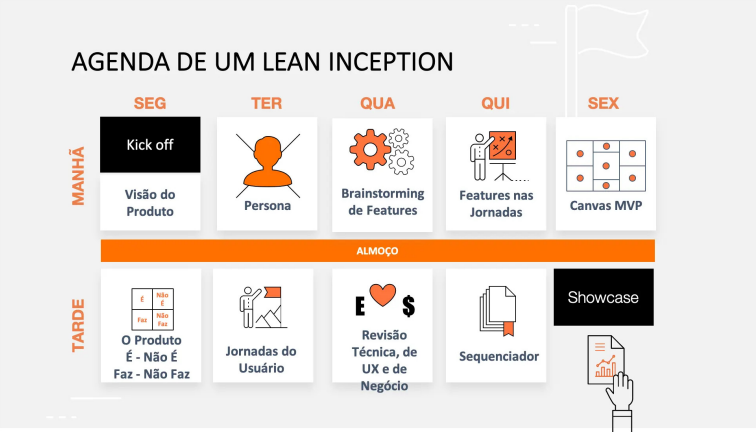
A Lean Inception[1] visa alinhar equipes sobre o produto mínimo viável (MVP). Foi criada para evitar o desperdício de tempo e de recursos com a criação de produtos

[1] Termo criado por Caroli (2018)

errados, porque foca na eficiência na criação e evolução iterativa e incremental para um produto de software eficaz (CAROLI, 2018).

De acordo com Caroli (2018), a Lean Inception tem o intuito de promover alinhamento e definição de objetivos, estratégias e o escopo do produto por meio de uma oficina colaborativa. A oficina é composta de uma sequência de atividades que usam de técnicas de *Design Thinking*[2] e *Lean StartUp*[3] para divergir e convergir o alinhamento sobre um produto. A figura 1 descreve um exemplo de agenda recomendada para a oficina de Lean Inception que contem: Apresentações, *Kick Off*[4], Visão do Produto, Objetivos do Produto, *Personas*, Jornadas, *Brainstorming*[5] de *Features*[6], Revisão Técnica, de UX e de Negócio, Sequenciador e *Showcase*[7] que serão explicadas a frente.

Figura 1 – Exemplo de agenda Lean Inception



Fonte: Caroli (2018)

- **Apresentações:** Atividade utilizada para conhecer os participantes antes de entrar mais fundo nas atividades da Inception.
- **Kick off:** Essa sessão serve para orientar os participantes com as razões do negócio, e dar início ao entendimento comum das suas principais necessidades e objetivos.
- **Visão do produto:** Aqui é listado a visão inicial que será refinada.
- **Objetivos do Produto:** Cada participante deve compartilhar o entendimento sobre os objetivos do produto, o que é discutido até entrar em um consenso.
- **Personas:** Nessa fase são criadas personas, onde simbolizara o usuário e seus objetivos com o produto.

[2] Design Thinking: técnica usada estimular ideação e propostas de soluções.
[3] Lean StartUp: conjunto de processos usados para desenvolver produtos.
[4] Kick Off: pontapé inicial.
[5] Brainstorming: sessão de debate.
[6] Features: funcionalidaes de um sistema.
[7] Showcase: apresentação do caso.

- **Jornadas:** Para cada uma das persona deve ser descrito a sequência de passos necessários para alcançar um objetivo. Alguns desses passos precisam ser pontos de interação do usuário com o produto.

- **Brainstorming de Features:** Nessa fase, com base nas jornadas, uma sessão de criatividade acontece, onde será listado várias funcionalidades, a descrição de uma ação ou interação de um usuário com o produto, e refinadas.

- **Revisão Técnica, de UX e de Negócio:** Aqui, os participantes devem entender a relação tecnica, negocial e de esforço para cada funcionalidade.

- **Sequenciador:** A partir das funcionalidades listadas deve-se elaborar um plano de entrega chamado sequenciador.

- **Canvas MVP:** Nessa fase os participantes devem se juntar para descrever um Canvas MVP.

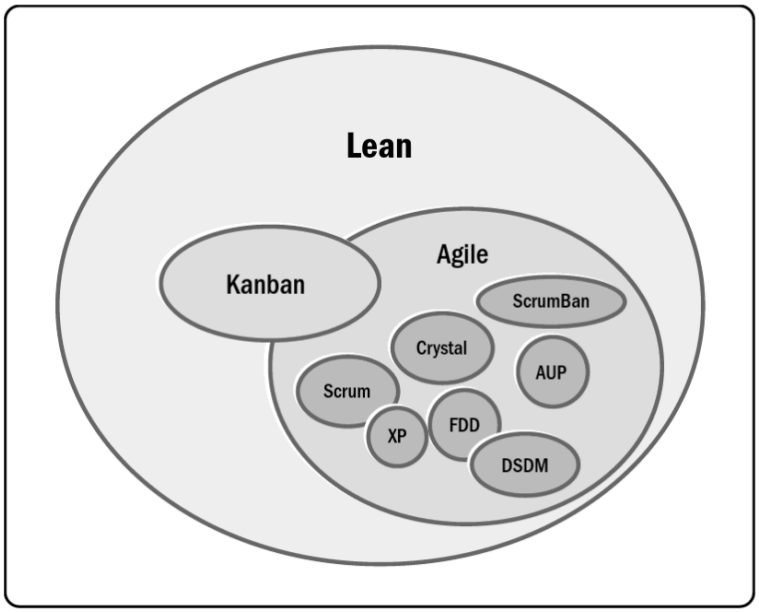
- **Showcase:** No show case o resultado é validado se faz sentido para o produto ou não.

2.3 Práticas Ágeis

Conforme a evolução do produto de *software*, o processo de desenvolvimento se tornou mais exploratório e em torno de solução de problemas. Alliance (2017) defende que projetos de alta incerteza têm altas taxas de mudança, complexidade e risco, por isso, foram criadas abordagens ágeis para explorar o desenvolvimento em ciclos curtos que se adaptam rapidamente com base na avaliação e no *feedback*.

O termo métodos ágeis cobre uma variedade de estruturas e métodos. A Figura 2 coloca o ágil como um termo geral, referindo-se a qualquer tipo de abordagem, técnica, estrutura, método ou prática que atenda aos valores e princípios do Manifesto Ágil. Como métodos ágeis e Kanban compartilham dos mesmo conceitos: “foco no valor”, “pequenos tamanhos de lote” e “eliminação de desperdício” de lean, foram colocados como subconjuntos. (ALLIANCE, 2017).

Figura 2 – Agile é um termo geral para muitas abordagens



Fonte: Alliance (2017)

2.4 Manutenção de Software

Segundo a norma IEEE Std 1219, do Institute of Electrical and Electronics Engineers[8] (IEEE), a definição básica de manutenção de *software* é a modificação do produto após a entrega para corrigir falhas, melhorar atributos como desempenho ou adaptar o produto a um ambiente modificado. Em outras palavras, um *software* que sofre modificação no código e documentação associada é considerado um *software* em manutenção (IEEE. . . , 1998).

Esta norma define um processo de manutenção que inclui as fases (IEEE. . . , 1998):

1. **Identificação, classificação e priorização do problema/modificação:** Aqui as modificações de *software* são identificadas, classificadas e recebem uma classificação de prioridade a partir dos tipos de manutenção que podem ser:

   - **Corretivo:** Modificação reativa realizada após a entrega para corrigir falhas descobertas
   - **Adaptativo:** Modificação realizada após a entrega para manter uma aplicação utilizável em um ambiente alterado ou em mudança
   - **Perfectivo:** Modificação após a entrega para melhorar o desempenho ou a capacidade de manutenção

[8] Instituto de Elétrica e Eletrônica.

- **Emergência:** Manutenção corretiva não programada realizada para manter um sistema operacional.

2. **Análise:** A fase de análise deve usar a documentação do sistema e do projeto, para estudar a viabilidade e o escopo da modificação e elaborar um plano preliminar para projeto, implementação, teste e Entrega.

3. **Desenho:** Deve-se usar da documentação atual do sistema, do projeto, do *software* e do banco de dados para arquitetar uma nova solução e modificação do sistema.

4. **Implementação:** Aqui, após o desenho da solução, os resultados devem ser usados para conduzir o esforço de implementação.

5. **Teste de regressão/sistema:** O teste de regressão faz parte do teste do sistema e deve ser realizado para validar que o código modificado não introduz falhas que não existiam antes da atividade de manutenção.

6. **Teste de aceitação:** Os testes de aceitação devem ser conduzidos em um sistema funcional pelas partes interessados.

7. **Entrega:** A fase final é a de entrega do produto de *software* atualizado.

2.5 Aplicações Web

As aplicações web são serviços de software projetados para oferecer suporte à interação máquina-a-máquina, permitindo que dois dispositivos eletrônicos se comuniquem pela Internet. As principais vantagens oferecidas pelas APIs são facilidade de acesso e facilidade de consumo (MCWHERTER; GOWELL, 2012).

Segundo McWherter e Gowell (2012), essas tecnologias explodiram no mercado porque provaram ser robustas e se adaptando muito bem para serviços da *Web*. Além disso, da consumabilidade, a capacidade de entender o que o servidor está comunicando funcionam muito bem para fornecer páginas da *Web*.

2.6 Sistemas de Gestão de Conteúdo

Eden (2008) defende que muitas páginas da Web surgiram de interesses pessoais de membros que podem não estar presentes no futuro. Essas paginas possuem um conjunto de usuários que as consideram parte da missão de serviço que desejam personalizar sua aparência para enfatizar a consistência e a marca. Isso pode ser muito difícil quando existem diferentes autores com diferentes criatividades e individualidades.

Um sistema de gerenciamento de conteúdo (CMS) é uma forma de gerência do conteúdo das páginas da *Web*. Um CMS auxilia a conformidade, fornecendo um sistema

gerenciado centralmente para exibir o conteúdo, que permanece sob o controle da equipe com experiência. Além disso, oferece uma maneira de evitam o fardo de codificar manualmente todas as informações em cada página em Linguagem de Marcação de HiperTexto (HTML) (EDEN, 2008).

2.7 Gateway de pagamento

O *Gateway* de pagamento é uma forma de orquestrar um pagamento digital que estabelece a comunicação entre um processador de pagamento e o comerciante. Ele visa fornecer serviços de *gateway*, porta de entrada, ou roteamento para os comerciantes (GOMZIN, 2014).

Diferente de um processador que opera o pagamento, um gateway encaminha a solicitação para o processador. Desse modo, também armazena lotes de transações e inicia e processa a liquidação (GOMZIN, 2014).

2.8 Testes

Segundo Copeland (2004) testar é o processo de comparar "o que é" com "o que deveria ser". É dividido em teste de caixa preta, caixa branca e caixa cinza. O teste de caixa preta é baseado nos requisitos e especificações, não requer conhecimento dos caminhos internos, estrutura ou implementação do software em teste. O teste de caixa branca é baseado nos caminhos internos, na estrutura e na implementação do software e requer habilidades de programação. O de teste de caixa cinza é uma união de caixa branca e preta, deve-se entender como o software foi implementado para escolher testes de caixa preta mais eficazes.

Os quatro níveis de testes são (COPELAND, 2004):

1. **Unidade** – Uma unidade de software armazenados em um único arquivo de disco;
2. **Integração** – juntam-se unidades em subsistemas e finalmente em sistemas, isso porque é possível que as unidades funcionem perfeitamente isoladamente, mas falhem quando integradas;
3. **Sistema** – Um sistema consiste em todo o produto entregue ao cliente, dentro do teste do sistema há muitos outros tipos de teste: funcionalidade, usabilidade, segurança, etc.;
4. **Aceitação** – O teste de aceitação quando concluído com sucesso, resultará na aceitação do software.

2.9 Resumo do capítulo

Neste capítulo, foi discutida a Engenharia de Software, uma disciplina que lida com a complexidade do software e que utiliza o desenvolvimento iterativo como paradigma mais comum. Esse paradigma permite que os programadores planejem por períodos limitados, conhecidos como iterações, acompanhando a volatilidade dos requisitos.

Também foram introduzidos conceitos básicos essenciais para a compreensão do trabalho. Foram abordados tópicos como a Lean Inception, uma abordagem que visa alinhar equipes e evitar a criação de produtos errados por meio do foco na eficiência e na evolução iterativa e incremental de um produto de software eficaz.

Outro aspecto abordado foi a Manutenção de Software, que compreende a modificação do produto após a entrega para corrigir falhas, melhorar o desempenho ou adaptá-lo a um ambiente modificado. A norma IEEE Std 1219 define um processo de manutenção que inclui fases como identificação, classificação e priorização do problema/modificação, análise, desenho, implementação, teste de regressão/sistema, teste de aceitação e entrega.

Além disso, foram mencionados o Gateway de Pagamento, responsável por orquestrar pagamentos digitais entre processadores e comerciantes, e as Aplicações Web, que possibilitam a interação entre dispositivos eletrônicos por meio da Internet, geralmente utilizando APIs para facilitar o acesso e o consumo de serviços.

Por fim, foram explorados os Sistemas de Gestão de Conteúdo, que auxiliam na criação e gerenciamento de conteúdo em páginas da web, garantindo consistência e personalização.

Esses conceitos fornecem uma base sólida para a compreensão do trabalho e sua aplicação no contexto abordado.

3 Suporte Tecnológico

Este capítulo visa descrever os aspectos técnicos referentes ao desenvolvimento do Agromart, no gerenciamento do projeto e configuração.

3.1 Lean Inception

A oficina de Lean Inception foi realizada utilizando o modelo de Agenda Lean de Caroli (2018) no quadro branco Mural. Mural é uma plataforma que oferece um espaço de colaboração na Internet por meio de um quadro branco digital. É utilizado para planejamento e facilitação de oficinas através da colaboração de equipes (MURAL, 2022).

Além de funcionar em todos os lugares, é a única plataforma que faz uma combinação de quadro branco digital e treinamento no Sistema Luma. Dessa forma, oferecem uma forma prática de contribuição conjunta (MURAL, 2022).

Para a comunicação durante as oficinas foi utilizado o aplicativo Discord. O Discord é um aplicativo gratuito que oferece voz e vídeo de baixa latência. Com esse aplicativo é possível compartilhar a tela e se reunir instantaneamente para conversar sem precisar ligar (DISCORD, 2023).

3.2 Práticas Ágeis

O código-fonte Agromart encontra-se hospedado no *GitHub*, uma ferramenta gratuita que utiliza o *Git* como recurso de controle de versão com integrações com diversas aplicações (GITHUB, 2022). Seguindo a prática do *GitHub* o rastreio do Kanban, do *Scrum* e os métodos de planejamento do *eXtreme Programming* deve ser através do *Github Projects* (GITHUB, 2023b).

Github Projects é uma planilha adaptável que se integra aos seus problemas e solicitações de *pull* no *GitHub* para ajudar você a planejar e acompanhar seu trabalho com eficiência. Além disso, é possível personalizar várias exibições adicionando um quadro Kanban com campos personalizados para acompanhar dados de *Scrum*. Como não empoe uma metodologia específica, fornece recursos flexíveis e personalizáveis para cada equipe (GITHUB, 2023b).

3.3 Arquitetura

3.3.1 App

O aplicativo Agromart disponível na loja Android foi desenvolvido em React Native um framework JavaScript. O React Native é renderizado com código nativo, o que significa que a interface criada para o usuário visa aplicações mobile tanto para iOS quanto Android. Usar React Native melhores partes do desenvolvimento nativo (NATIVE, 2023).

A publicação na loga Android foi realizada pelo Expo. Um framework de código aberto para aplicativos executados em React Native no Android, iOS e na Web. Ele disponibiliza recursos para criar e dimensionar um aplicativo. Entre esses recursos há atualizações *over the air* que disponibiliza a publicação do aplicativo sem a necessidade de revisão das lojas, pois traz facilitações na geração de certificados e executáveis (EXPO, 2023).

3.3.2 API

O Agromart é arquitetado para receber integração com API de cada CSA cadastrada (FREITAS; CELLA, 2023). Por ser mantida por cada CSA, a API do sistema Agromart é desenvolvido com o Strapi um CMS *headless* de código aberto. O Strapi possui uma comunidade grande com diferentes soluções customizadas. Ele é provisionado em JavaScript, simples e de alto desempenho (STRAPI, 2023).

3.3.3 Pagamento Digital

O pagamento digital acontecerá através de um integrador de gateway de pagamento desenvolvido em Node Express. Foi escolhido esse framework de desenvolvimento para manter o padrão de projeto escolhido na API dicionario. Esse serviço servirá para gerenciar os sistemas de terceiros de pagamento.

Os gateways de pagamento incorporados a essa API serão os indicados na matriz de decisão na seção 5.4: PayPal, MercadoPago e PagSeguro.

3.3.3.1 PayPal

O PayPal é um gateway de pagamento que atua atuando como operadores com as instituições financeiras para autorizar. É uma estrutura segura e usada para finalizar, parcelar e cobrar compras. Essa infraestrutura é disponibilizada em formato de API que permite integrar com muitas tecnologias e em qualquer infraestrutura (DEVELOPER, 2022).

3.3.3.2 Mercado Pago

O Mercado Pago também é um gateway de pagamento disponibilizado em formato API. Dessa forma, permitindo pagamentos digitais necessário apenas as credenciais atualizadas. Esse serviço não faz cobrança de mensalidade e possui proteção contra fraudes (DEVELOPERS, 2023).

3.3.3.3 PagSeguro

O PagSeguro, além de ser uma aplicação de pagamento com funcionalidades para cartão, permite o pagamento via pix. Isso porque possui à disposição mais de 25 meios de pagamento (PAGSEGURO, 2023).

3.4 Conteinerização

Conteinerização é utilizada para isolar os processos de desenvolvimento dos processos da maquina hospedeira. Um contêiner possui uma instância executável de uma sistema de arquivos personalizado que pode ser usado em diversas maquinas e em qualquer sistema operacional. Todos esses recursos são supridos e acessíveis usando Docker (DOCKER, 2023b).

Com o Docker é possível integrar vários contêineres separados de forma independente e em diferentes linguagens de programação usando o Docker Compose (DOCKER, 2023a). O Compose é usa um arquivo YAML para configurar os serviços do seu aplicativo. Então, com um único comando, você cria e inicia todos os serviços de sua configuração (DOCKER, 2023c).

3.5 Banco de Dados

O sistema gerenciador de banco de dados (SGBD) relacional utilizado pela Api do Agromart é o PostgreSQL (RODRIGUES; MACÊDO, 2021). O PostgreSQL é um SGBD de código aberto gratuito, confiável e robusto. Ele também possui uma grande comunidade que oferece tutoriais e auxílio aos desenvolvedores, sejam projetos comerciais ou não (POSTGRESQL, 2023). Por esses motivos essa proposta continuará seguindo esse banco de dados.

3.6 Documentação

Para disponibilizar a documentação e permitir futuras evoluções foi usado o Docusaurus em conjunto do GitHub Pages e PlantUML.

3.6.1 Site

O desenvolvimento da site foi realizado com o Docusaurus. Um gerador de site otimizado em React que converte HTML, Markdown e componentes React para a exibição. Também possui um grande facilitador é a barra de pesquisa que facilita aos leitores de encontrar a documentação específica (DOCUSAURUS, 2023).

3.6.2 Deploy

O deploy e hospedagem desse site desenvolvido em Docusaurus foi provido pelo GitHub Pages. O GitHub Pages disponibiliza sites diretamente do repositório GitHub, basta escolher um branch que o processo é automatizado (GITHUB, 2023c).

3.6.3 Linguagem de Modelagem Unificada

A documentação dessa proposta foi atualizada usando de Linguagem de Modelagem Unificada (UML) em conjunto com PlantUML. PlantUML é uma ferramenta de desenho que auxilia na modelagem de UMLs baseada em texto, facilitando para as próximas versões serem atualizadas. Também é de código aberto e altamente personalizável (PLANTUML, 2023).

3.7 Resumo do capítulo

Neste capítulo, foram abordados diversos aspectos relacionados à implementação do projeto. Inicialmente, foi discutida a utilização da Lean Inception, uma oficina realizada utilizando o modelo de Agenda Lean de Caroli (2018) no quadro branco digital Mural. O Mural é uma plataforma de colaboração online que oferece um espaço para planejamento e facilitação de oficinas.

Em relação às práticas ágeis, foi mencionado o uso do GitHub como uma ferramenta de controle de versão e rastreamento de projetos. O GitHub Projects foi destacado como uma planilha adaptável que integra problemas e solicitações de pull, facilitando o planejamento e acompanhamento do trabalho.

No que diz respeito à arquitetura do projeto, foi mencionado que o aplicativo Agromart para Android foi desenvolvido em React Native, um framework JavaScript que permite a criação de interfaces para iOS e Android. A publicação do aplicativo na loja Android foi feita por meio do Expo, um framework de código aberto que disponibiliza recursos para criação e dimensionamento de aplicativos.

A API do Agromart foi desenvolvida com o Strapi, um CMS headless de código aberto que permite a integração com diferentes soluções customizadas. Para o pagamento

digital, foi utilizado um integrador de gateway de pagamento desenvolvido em Node Express. Os gateways de pagamento escolhidos foram o PayPal, o Mercado Pago e o PagSeguro.

A conteinerização foi adotada como uma forma de isolar os processos de desenvolvimento dos processos da máquina hospedeira. O Docker foi utilizado para criar e integrar contêineres independentes usando o Docker Compose.

Quanto ao banco de dados, o PostgreSQL foi escolhido como o sistema gerenciador de banco de dados relacional para a API do Agromart. O PostgreSQL é um SGBD de código aberto conhecido por sua confiabilidade e robustez.

Em relação à documentação, o Docusaurus foi utilizado para desenvolver o site, que disponibiliza a documentação do projeto. O site foi hospedado no GitHub Pages, que permite a disponibilização direta do repositório GitHub. A Linguagem de Modelagem Unificada (UML) em conjunto com o PlantUML foi utilizada para atualizar a documentação, facilitando a modelagem baseada em texto.

Essas informações fornecem uma visão geral dos aspectos tecnológicos abordados na implementação do projeto

4 Metodologia

Neste capítulo, tem-se o detalhamento metodológico primeiramente, é apresentada a Classificação da Pesquisa, considerando. Na sequência, são tratados os Fluxos de Atividades em conjunto com os Cronogramas. Por fim, há o Resumo do Capítulo.

4.1 Classificação de Pesquisa

O modelo de pesquisa seguido nesse trabalho foi adotado segundo a organização proposta por Gerhardt e Silveira (2009).

A abordagem dessa pesquisa é classificada como qualitativa, por ter um intuito de compreender as CSAs com base em informações subjetivas coletadas por entrevistas com agricultores e co-agricultores. Com relação à natureza, o trabalho pode ser classificado como pesquisa aplicada, ao pretender-se propor uma solução para um contexto específico, utilizando-se de uma abordagem prática (GERHARDT; SILVEIRA, 2009).

No que se refere aos objetivos do TCC, classifica-se como exploratório. Esse tipo de pesquisa segundo Gil et al. (2002)

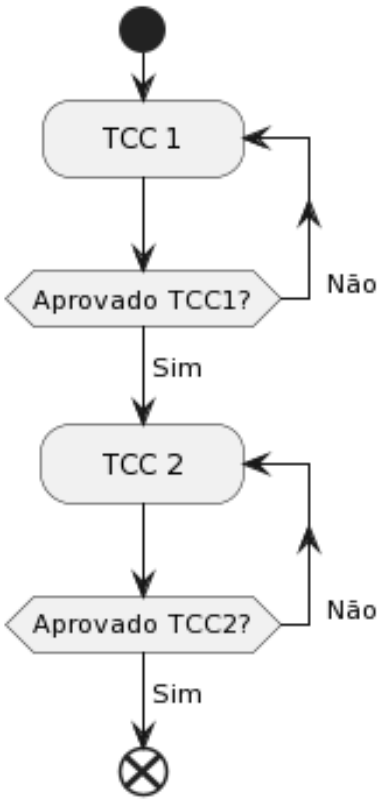
> tem como objetivo proporcionar maior familiaridade com o problema, com vistas a torná-lo mais explícito ou a constituir hipóteses. Pode-se dizer que estas pesquisas têm como objetivo principal o aprimoramento de idéias ou a descoberta de intuições. Seu planejamento é, portanto, bastante flexível, de modo que possibilite a consideração dos mais variados aspectos relativos ao fato estudado.

Uma vez que, pretende-se selecionar atributos de qualidades voltados para o perfil da CSA, para apoiar seus processos administrativos e comunitários. Este trabalho foca em uma mescla de pesquisa bibliográfica e estudo de caso, já que este TCC possui tanto o levantamento teórico de referências quanto o estudo de uma entidade, as CSAs (GERHARDT; SILVEIRA, 2009).

4.2 Fluxo de Atividades

O fluxo geral de atividades do Trabalho de Conclusão de Curso é apresentado na Figura 3. Nessa, é possível observar os dois subprocessos TCC 1 e TCC 2, descritos nas Figuras 4 e 5.

Figura 3 – Fluxo Geral de Atividades

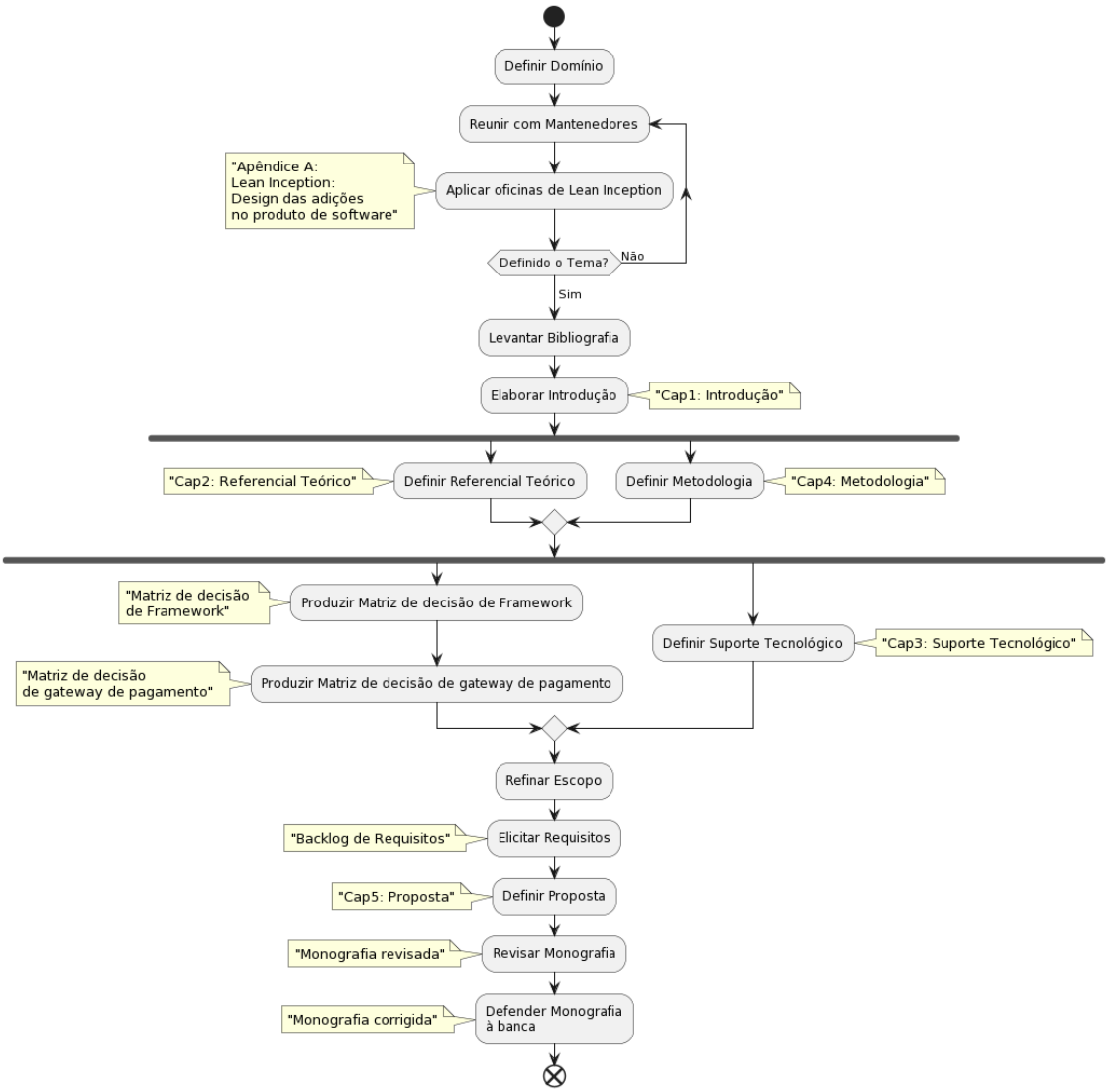


Fonte: autores

4.2.1 Trabalho de Conclusão de Curso 1

- **Definir Domínio:** Refere-se à definição do domínio de pesquisa do Trabalho de Conclusão de Curso;
- **Reunir com Mantenedores:** Este subprocesso referem-se as reuniões realizadas com os mantenedores do projeto: o professor orientador, alunos pesquisadores e desenvolvedores e representantes de CSA. Essas reuniões visam entender o projeto, seus pontos de melhoria e dificuldades.
- **Aplicar Oficinas de Lean Inception:** Representa o subprocesso de aplicar os encontros virtuais para a discussão de escopo e projeto seguindo a agenda da Lean Inception, apresentada na seção 2.2. Após a realização da agenda é alinhado os achados e escopo com o professor orientador. Se não aprovado deve-se Reunir com Mantenedores e repetir a agenda.
- **Levantar Bibliografia:** Essa etapa visa levantar estudos relacionados ao domínio da pesquisa do trabalho.
- **Elaborar Introdução:** Refere-se a escrita do capítulo de introdução ao trabalho pretendendo contextualizar, justificar e relatar os objetivos de pesquisa.

Figura 4 – Fluxo TCC1



Fonte: autores

- **Definir Referencial Teórico:** Refere-se a escrita do capítulo de Referencial Teórico que sumariza os conceitos fundamentais. Essa etapa é feita em paralelo à definição da Metodologia e resulta no Capítulo 2.

- **Definir Metodologia:** Aborda a definição de metodologias que guiam esse tralho. Essa etapa é feita em paralelo à definição do Referencial Teórico e resulta no Capítulo 4.

- **Produzir Matriz de Decisão de Framework:** Nessa etapa é produzido em paralelo com a definição do Suporte Tecnológico a matriz de decisão de framework. Essa matriz é usada como um suporte para a decisão tecnológica do trabalho.

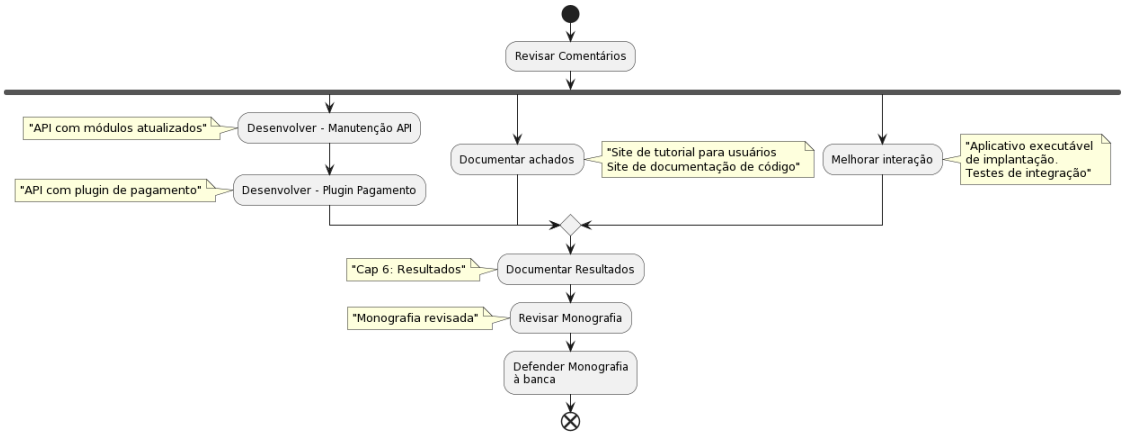
- **Produzir Matriz de Decisão de Gateway de Pagamento:** Nessa etapa é produzido em paralelo com a definição do Suporte Tecnológico a matriz de decisão

de gateway de pagamento. Essa matriz é usada como um suporte para a decisão tecnológica do trabalho.

- **Definir Suporte Tecnológico:** Esta atividade refere-se à definição das tecnologias utilizadas para o desenvolvimento do trabalho e resulta no Capítulo 3.
- **Refinar Escopo:** Refere-se ao refinamento do escopo, após um estudo mais aprofundado no referencial teórico e suporte tecnológico. Esse refinamento é realizado em conjunto ao orientador.
- **Elicitar Requisitos:** O backlog do produto é definido no fim da etapa de elicitação de requisitos.
- **Definir Proposta:** Refere-se a escrita do Capítulo de Proposta na Monografia de TCC1 substituído no TCC 2.
- **Revisar Monografia:** Nessa etapa é feito uma revisão final na primeira versão da Monografia.
- **Defender Monografia à banca:** Finalizando o processo é a etapa de apresentação da proposta para a Banca Avaliadora.

4.2.2 Trabalho de Conclusão de Curso 2

Figura 5 – Fluxo TCC2



Fonte: autores

- **Revisar Comentários:** Refere-se a aplicar as correções dos comentários realizados por parte da Banca Avaliadora.
- **Desenvolver - Manutenção API:** Neste subprocesso há o desenvolvimento de código para alcançar o Épico de Manutenção de Software, acontece em paralelo a Documentar Achados e Melhorar Interação.

- **Desenvolver - Plugin Pagamento:** Neste subprocesso há o desenvolvimento de código para alcançar o Épico de Pagamento, acontece em paralelo a Documentar Achados e Melhorar Interação.

- **Documentar achados:** Neste subprocesso há o desenvolvimento de uma nova página na internet que será usada pelos usuários finais e atualização da documentação atual dos desenvolvedores, acontece em paralelo a Melhorar Interação e Desenvolvimento.

- **Melhorar interação:** Neste subprocesso há o desenvolvimento de aplicação para melhorar a interação do usuário com o Agromart, acontece em paralelo a Documentar achados e Desenvolvimento.

- **Documentar Resultados:** Essa etapa tem em vista documentar na Monografia os resultados gerados do Desenvolvimento, Documentação e da Melhora de interação. Resultando no Capítulo 6.

- **Revisar Monografia:** Nessa etapa é feito uma revisão final da Monografia.

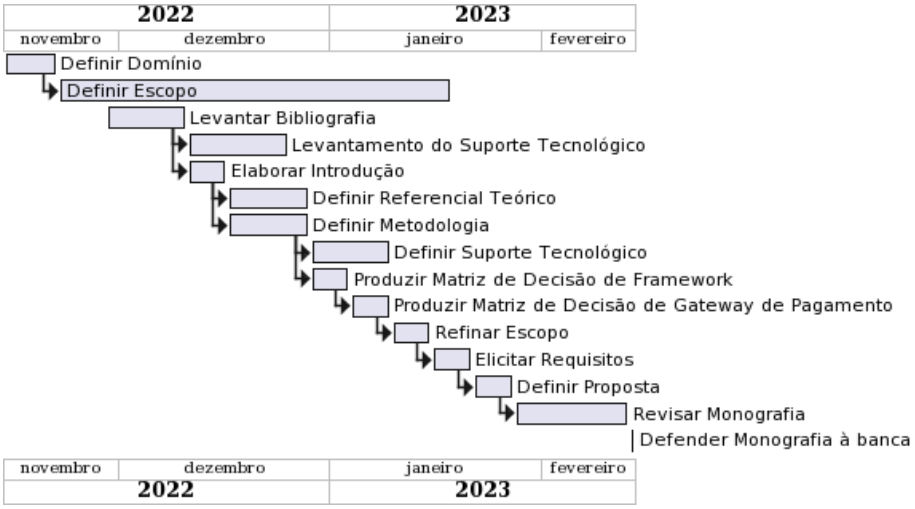
- **Defender Monografia à banca:** Finalizando o processo é a etapa de apresentação dos resultados para a Banca Avaliadora.

4.2.3 Cronograma

4.2.3.1 Cronograma TCC 1

O cronograma confere uma noção de tempo para as atividades a serem realizadas no decorrer do TCC e pode ser observado na Figura 6. Esse cronograma se relaciona ao Fluxo TCC1 na Figura 4 com o ajuste de definição de escopo que diz respeita as etapas de reunir com mantenedores e aplicar oficinas de Lean Inception.

Figura 6 – Cronograma do TCC 1



Fonte: autores

4.2.3.2 Cronograma TCC 2

Figura 7 – Cronograma do TCC 2

Fonte: autores

4.3 Metodologia de Desenvolvimento

4.3.1 Scrum

Scrum é uma estrutura de processo para gerenciar o desenvolvimento de produtos. O processo consiste em eventos, *Sprint planning*, *Daily*, *Sprint review* e *Sprint retrospective*, artefatos, *Product backlog*, *Sprint backlog* e *Increments*, e regras que usa uma abordagem iterativa para entregar o produto funcional. *Sprints* são periodos de 1 mês ou menos com durações consistentes onde um incremento é produzido. A equipe *Scrum* consiste em um proprietário do produto, responsável por maximizar o valor do produto, equipe de desenvolvimento, equipe multifuncional e auto-organizada, e *scrum master*, responsável por garantir que o processo *Scrum* seja mantido (ALLIANCE, 2017).

Nesse projeto o Scrum foi aplicado de maneira adaptada, pois a equipe é formada apenas de desenvolvedores. Os conceitos utilizados contém o *Product Backlog*, *Sprints* com duração de duas semanas começando com a *Sprint Planning* que gera o artefato *Sprint Backlog* e finalizando com uma *Sprint Review*.

4.3.2 Kanban

O Método Kanban é utilizado em ambientes que permite um fluxo contínuo de trabalho de valor para o cliente. Ao contrário da maioria das abordagens ágeis, o Método Kanban não prescreve o uso de iterações com períodos, mas iterações podem ser usadas dentro do Método Kanban. Isso requer que o princípio de puxar funcionalidades únicas através do processo e limitar o trabalho em andamento para otimizar o fluxo permaneça intacto (ALLIANCE, 2017).

As equipes precisam estar focadas em fluir o trabalho e não iniciar um novo trabalho até que o trabalho em andamento seja concluído. Isso por no Kanban ser mais importante concluir o trabalho do que iniciar um novo trabalho. A equipe trabalha em conjunto para implementar e aderir aos limites do trabalho em andamento (WIP) e fazer com que cada item seja concluído (ALLIANCE, 2017).

Em conjunto com o Scrum foi utilizado o Kanban para rastrear e acompanhar as tarefas e fluxo de trabalho. Para isso, foi utilizado o GitHubProject[1], ferramenta descrita na Seção 3.2 com um quadro composto seis colunas: Novo, Backlog, Backlog da Sprint, Em Progresso, Em revisão e Feito.

4.3.3 eXtreme Programming

EXtreme Programming (XP) é um método de desenvolvimento de software baseado em ciclos frequentes conhecido por popularizar um conjunto de práticas destinadas a melhorar os resultados. O método considera valores de comunicação, simplicidade, feedback, coragem, respeito e por princípios-chave de diversidade, oportunidade, falha, qualidade, pequenos passos e responsabilidade aceita (ALLIANCE, 2017). As práticas são dividias em organizacionais, técnicas, planejamento e integração:

- **Organizacional:** Sentam-se juntos; toda a equipe; espaço de trabalho informativo
- **Técnico:** Programação em par; programação de teste unitário; projeto incremental
- **Planejamento:** Histórias de usuários; ciclo semanal; ciclo trimestral;
- **Integração:** Integração contínua; teste de integração.

[1] Disponível em: https://github.com/orgs/AgroMart/projects/2. Acesso em: 15 fev, 2023.

Todos os valores do XP foram seguidos para melhorar os resultados desse projeto. Já para as práticas ágeis foram acolhidas conforme a necessidade. Essas práticas incluem programação em par, espaço de trabalho informativo com projeto incremental e com histórias de usuários em um ciclo semanal e semestral.

4.4 Resumo do capítulo

O capítulo em questão aborda a classificação da pesquisa realizada no trabalho. O modelo de pesquisa adotado é qualitativo e aplicado. A pesquisa qualitativa tem em vista compreender as Comunidades de Agricultura Sustentável (CSAs) com base em informações subjetivas coletadas por meio de entrevistas com agricultores e co-agricultores.

O objetivo do trabalho é exploratório, visando proporcionar uma maior familiaridade com o problema e gerar ideias e intuições. O planejamento da pesquisa é flexível para considerar vários aspectos relacionados ao tema estudado.

O capítulo também apresenta o fluxo de atividades do Trabalho de Conclusão de Curso (TCC), dividido em duas partes: TCC 1 e TCC 2. O TCC 1 envolve atividades como definir o domínio de pesquisa, reunir-se com os mantenedores do projeto, aplicar oficinas de Lean Inception, levantar bibliografia, elaborar introdução, definir referencial teórico, metodologia e suporte tecnológico, refinar o escopo, elicitar requisitos, definir proposta, revisar a monografia e defender a monografia perante a banca avaliadora.

Já o TCC 2 abrange atividades como revisar os comentários da banca avaliadora, desenvolver a manutenção da API, desenvolver o plugin de pagamento, documentar os achados, melhorar a interação, documentar resultados, revisar a monografia e defender a monografia perante a banca avaliadora.

Além disso, são apresentados cronogramas para o TCC 1 e TCC 2, fornecendo uma noção de tempo para as atividades a serem realizadas durante o trabalho.

No contexto do desenvolvimento do trabalho, é mencionado o uso da metodologia Scrum, uma estrutura de processo para gerenciar o desenvolvimento de produtos. O Scrum envolve eventos, artefatos e regras que permitem uma abordagem iterativa na entrega do produto funcional, utilizando sprints de duração consistente para produzir incrementos do produto. A equipe Scrum é composta por um proprietário do produto e outros membros responsáveis pelo desenvolvimento.

Em resumo, o capítulo apresenta a classificação da pesquisa, o fluxo de atividades do TCC 1 e TCC 2, cronogramas e a metodologia de desenvolvimento utilizada, fornecendo uma visão geral do trabalho realizado.

5 Agromart

Esse capítulo tem o objetivo de contextualizar o Projeto Agromart e seus componentes ao apresentar os artefatos gerados pelo metodologia. Sendo eles os artefatos do Lean Inception, a Matriz de Decisão de Framework e de Decisão de Gateway de Pagamento e o Backlog priorizado.

Com base nessas informações, descreve-se a Arquitetura do sistema desenvolvido, apresentando a Arquitetura Geral, as Classes desenvolvidos durante o trabalho. Por fim, têm-se as Iterações de Desenvolvimento e o Resumo do Capítulo.

5.1 Contexto

O projeto Agromart teve sua origem durante a participação dos idealizadores no hackathon da UnB-FGA 2020, no tema "Cultivando Conexões". O desafio proposto era desenvolver um software que estabelecesse uma conexão entre agricultores e consumidores, considerando o isolamento social causado pela COVID-19 e as dificuldades enfrentadas pelos produtores rurais do Distrito Federal e região em divulgar seus produtos para os clientes (RODRIGUES; MACÊDO, 2021).

A inspiração inicial veio uma pequena agricultora de Goiás, adotou um sistema de venda baseado na confiança. Os consumidores escolhiam os produtos desejados em sua barraca e deixavam o pagamento no local, evitando desperdício e seguindo as recomendações de distanciamento social impostas pela pandemia. Essa história motivou os idealizadores a dar o primeiro passo para a criação do Agromart (RODRIGUES; MACÊDO, 2021).

Durante o hackathon, foi desenvolvido um aplicativo que permitia aos agricultores divulgar suas lojas, barracas ou pontos de venda, incluindo informações como produtos, preços, localização e contato. Os consumidores podiam visualizar as lojas mais próximas por mapas e filtros, entrar em contato direto com os agricultores por meio de um link para iniciar uma conversa em um aplicativo de mensagens, com o objetivo principal de verificar a disponibilidade dos produtos e obter informações sobre os pagamentos. O aplicativo também fornecia orientações sobre o uso seguro durante a pandemia (RODRIGUES; MACÊDO, 2021).

A solução apresentada durante o hackathon conquistou o primeiro lugar na competição e, após o evento, os idealizadores tiveram contato com profissionais da área que apresentaram abordagens diferentes. Por meio de entrevistas e conversas, eles conheceram as regras de negócio das Comunidades que Sustentam a Agricultura (CSA's) e decidiram adaptar o software para atender a esses novos requisitos. Com essa nova abordagem, ba-

seada na venda de cestas agroecológicas por meio da assinatura de planos, os agricultores teriam uma garantia maior de escoamento de seus produtos (RODRIGUES; MACÊDO, 2021).

Para melhorar a adaptação do software, os idealizadores optaram por iniciar o projeto do zero, aproveitando alguns elementos de design anteriores. Além disso, adicionaram uma interface web para o gerenciamento por parte dos agricultores, enquanto mantinham o aplicativo móvel para os consumidores (RODRIGUES; MACÊDO, 2021).

O projeto Agromart é coordenado por professores da UnB-FGA, que continuam trabalhando nele juntamente com estudantes. Corrêa e Veludo (2022) trabalharam em uma interação que incluiu a implementação de uma abordagem de código livre e a integração de um sistema de pagamento digital que hoje já não é mais funcional.

Após a interação de Freitas e Cella (2023), o projeto Agromart recebeu uma nova abordagem com o objetivo de individualizar cada ambiente de Comunidade que Sustenta a Agricultura. Essa individualização foi planejada para que cada CSA possa ter seu próprio painel de administração no Strapi, um sistema de gerenciamento de conteúdo, conectado a um aplicativo central que fará requisições separadamente para cada CSA selecionada.

5.2 Lean Inception

A agenda utilizada para Aplicar Oficinas de Lean Inception, no Fluxo do TCC 1 na Figura 4, foi a recomendada por Caroli (2018). Com os dados levantados pelo agricultor na etapa de Reunir com Mantenedores foi possível gerar os artefatos de persona, jornada de usuário, idealização e MVP. Esses artefatos podem ser observados no Apêndice A[1].

O artefato de Visão do Produto expõe que o Agromart é para as CSAs que precisam organizar a relação entre pequenos agricultores e os co-agricultores. É um software de apoio à comercialização de cestas de produtos agroecológicos que facilita encontrar e organizar os pontos de encontro os participantes das CSAs. Diferentemente da csabrasilia e csadafloresta o nosso produto é um software livre e colaborativo que pode ser utilizado por qualquer CSA.

O produto mínimo viável (MVP) desse trabalho, apresentado no Apêndice A, descreve que a proposta é criar um gerenciador/integrador de gateway de pagamento. Para isso foi criado um conjunto de duas personas, a Ildete e a Raquel. A persona Ildete precisa controlar pagamentos de produtos e possa integrar com diferentes gateways de pagamento. Já a persona Raquel que se inscreve em cestas possa pagar os produtos pelo link gerado. Isso, resulta em um módulo de pagamento funcional.

[1] Disponível em: https://app.mural.co/t/tcc4451/m/tcc4451/1671668498239/77b23692216633e80f50f111bfdc63d1290babc1 Acesso em: 15 jan, 2023.

Essas informações foram cruciais para a validação do escopo e desenvolvimento da Matriz de Decisão de Framework e de Decisão de Gateway de Pagamento e o Backlog priorizado.

5.3 Matriz de Decisão de Framework

O MVP apresenta um módulo de pagamentos de produtos que possa integrar com diferentes gateways. No entanto, o Strapi encontrava-se com a versão desatualizada e com módulos não mais utilizados. Esse problema dificultava a inclusão de novos pacotes já que isso pode produzir incompatibilidade de versões. Levando a erros e dificuldades na integração.

Isso criou a necessidade de renovar a API do Agromart por completo. Na fase de Reunir com Mantenedores no Fluxo do TCC 1, houve conversas com outros desenvolvedores que apontaram muitas dificuldades com a utilização do Strapi. Pensando nisso, foi feita uma comparação apresentada na Tabela 1 para determinar se havia necessidade de trocar todo o *framework* ou apenas atualizar esse módulo.

Essa pesquisa foi baseada nos seguintes requisitos:

- **Afinidade**
  Facilidade para tradução do código atual para o novo, considerado de grande relevância para os desenvolvedores com um peso 3. Para pontuar a afinidade foi usada a seguinte escala:
  - 0: Sem compatibilidade entre as estruturas antigas e novas, exigindo uma reescrita completa da base de código
  - 1-3: Alta compatibilidade, mas com uma quantidade significativa de modificações e testes necessários
  - 4-6: Compatibilidade moderada, com algumas alterações e testes necessários
  - 7-9: Boa compatibilidade, com alterações e testes mínimos necessários
  - 10: Compatibilidade perfeita, sem necessidade de alterações ou testes
- **Segurança**
  A pontuação dessa categoria foi dada conforme a biblioteca de segurança Snyk (2022). Ela possui a própria avaliação para as tecnologias de desenvolvimento de software. Considerado de relevância com um peso 2;
- **Popularidade**
  Já para a popularidade foi considerado a quantidade de sites rodando segundo Tech (2023a) usando a seguinte escala com um peso 3;

  - 0: Uma estrutura que não é usada por nenhum site.
  - 1-3: Uma estrutura usada por um pequeno número de sites, geralmente menos de 1% de todos os sites.
  - 4-5: Uma estrutura usada por um número baixo de sites, normalmente entre 1-5% de todos os sites.
  - 6-7: Uma estrutura usada por um número moderado de sites, normalmente entre 5-15% de todos os sites
  - 8-9: Uma estrutura usada por inúmeros sites, normalmente entre 15-30% de todos os sites.
  - 10: Uma estrutura usada por um número muito alto de sites, normalmente mais de 30% de todos os sites.

- **Suporte**
  O suporte é referente a comunidade vinculada ao framework, percebida pela quantidade de estrelas no Github (2023a). Considerado de muita importância para a utilização por uma CSA com um peso 4;

A Tabela 1 compara as tecnologias de desenvolvimento de software Strapi, Wordpress, Ghost, Joomla, PrestaShop e Drupal. Essas tecnologias são as mais populares no mercado, e compartilham similaridades com o software utilizado atualmente (TECH, 2023b). Pode-se visualizar que o mais seguro segundo Snyk (2022) é o Wordpress seguido do Drupal, enquanto o menos seguro é a Prestashop com 3,6 pontos. Em questões de popularidade, o Wordpress é a ferramenta com maior pontuação, tendo cerca de 45% das páginas disponíveis na web. É importante notar que empatado em segundo lugar em popularidade, tendo cerca de 1,8% do mercado, o Joomla possui uma alta popularidade, apesar de seu índice de segurança ser baixo, apenas 4,2 pontos. Para o suporte temos o Strapi seguido pelo Ghost mostrando um favorecimento da comunidade em CMS feitos em javascript.

Tabela 1 – Matriz de decisão de Framework.

CMS	Compatibilidade	Segurança	Popularidade	Suporte	Resultado
Strapi	9	7,1	1	51,3	5,4
Wordpress	0	9,5	10	17	4,7
Ghost	7	7,8	1	42,2	4,7
Joomla	0	4,2	4	4,4	1,8
PrestaShop	0	3,6	3	6,9	1,6
Drupal	0	8,1	4	3,8	2,5

Fonte: autores.

Comparando os resultados da Tabela 1, o Strapi encontra-se ainda como a melhor opção. Por isso, nesse trabalho foi atualizado os módulos do Strapi conforme recomendado pela OWASP (2021).

5.4 Matriz de Decisão de Gateway de Pagamento

Para a decisão de quais gateways de pagamento integrar inicialmente no gerenciador foi feita uma matriz de decisão. Apresentada na Tabela 2, ela foi construída com base nos seguintes requisitos:

- **Métodos de pagamento**
  Foram considerados os tipos de métodos de pagamento que o gateway suporta. foi dada a essa categoria o peso 5. E em seguida os gateways foram avaliados da seguinte forma:
  - 1-2: Métodos de pagamento limitados, com apenas algumas opções disponíveis.
  - 3-4: Métodos de pagamento abaixo da média, com algumas opções populares disponíveis, mas faltando algumas opções importantes.
  - 5-6: Métodos de pagamento mediano, com opções padrão como cartões de crédito, mas nada excepcional.
  - 7-8: Métodos de pagamento acima da média, com uma boa seleção de opções, incluindo métodos de pagamento alternativos, como ACH, carteiras eletrônicas, etc.
  - 9-10: Excelentes métodos de pagamento, com uma ampla variedade de opções disponíveis e suporte para os métodos de pagamento mais recentes.
- **Taxas**
  Para a avaliação das taxas foram comparadas as taxas de transação, taxas de configuração e taxas mensais ou anuais cobradas por cada gateway. Levando um peso 2 nessa categoria a avaliação seguiu a seguinte classificação:
  - 1-2: Taxas altas, com taxas de transação caras, taxas de configuração e taxas mensais ou anuais.
  - 3-4: Taxas acima da média, com algumas taxas altas, mas também com custos mais baixos.
  - 5-6: Taxas médias, com taxas padrão alinhadas com as médias do setor.
  - 7-8: Taxas abaixo da média, com custos mais baixos do que a média do setor, mas com algumas taxas adicionais
  - 9-10: Taxas baixas, com custos mínimos e preços competitivos.

- **Segurança**
  Com peso 2, foi avaliado a existência de medidas de segurança implementadas para proteger contra fraude e violações de dados em cada um dos gateways. Suas notas foram determinadas de acordo com a seguinte classificação:

  - 1-2: Medidas de segurança inadequadas, com alto risco de fraude e violação de dados.
  - 3-4: Segurança abaixo da média, com algumas vulnerabilidades e proteção limitada contra fraudes e violações de dados.
  - 5-6: Segurança média, com medidas de segurança padrão, mas nada excepcional.
  - 7-8: Segurança acima da média, com fortes medidas de segurança em vigor, mas com espaço para melhorias.
  - 9-10: Excelente segurança, com medidas de segurança líderes do setor e baixo risco de fraude e violação de dados.

- **Integração**
  Com peso 3 a integração foi avaliada considerando a facilidade de integrar o gateway na API do software de acordo com a documentação e suporte disponibilizados. sua integração foi classificada como:

  - 1-2: Difícil de integrar, com documentação ruim e suporte limitado para desenvolvedores.
  - 3-4: Integração abaixo da média, com alguns problemas, como falta de opções de integração ou suporte limitado ao desenvolvedor.
  - 5-6: Integração média, com opções de integração funcional e suporte adequado ao desenvolvedor, mas nada excepcional.
  - 7-8: Integração acima da média, com um processo de integração suave e bom suporte ao desenvolvedor, mas com espaço para melhorias.
  - 9-10: Excelente integração, com fácil processo de integração, boa documentação e suporte abrangente ao desenvolvedor.

- **Usabilidade**
  Para usabilidade foi considerada a experiência do usuário do gateway para os clientes. Incluindo o design do processo de check-out e a disponibilidade de atendimento ao cliente. Com o peso 3 os gateways foram avaliados de acordo seguinte escala:

  - 1-2: Experiência do usuário insatisfatória, com processo de checkout confuso ou difícil e mínimo ou nenhum suporte ao cliente.

  - 3-4: Experiência do usuário abaixo da média, com alguns problemas, como um processo de checkout desajeitado ou suporte ao cliente limitado.
  - 5-6: Experiência de usuário mediana, com processo de checkout funcional e suporte ao cliente adequado, mas nada excepcional.
  - 7-8: Experiência do usuário acima da média, com um processo de checkout tranquilo e bom suporte ao cliente, mas com espaço para melhorias.
  - 9-10: Excelente experiência do usuário, com um processo de checkout amigável e suporte ao cliente abrangente.

- **Confiabilidade**
  Para classificar a confiabilidade dos gateways de pagamento foram avaliados o tempo de atividade do gateway, a frequência e a duração de quaisquer interrupções e o nível de suporte ao cliente fornecido durante tais incidentes. prezando por uma alta confiabilidade no sistema essa categoria possui peso 4 e a seguinte escala:

  - 1-2: Não confiável, com interrupções frequentes e suporte ao cliente insatisfatório durante as interrupções do serviço.
  - 3-4: Confiabilidade abaixo da média, com interrupções ocasionais e suporte ao cliente limitado.
  - 5-6: Confiabilidade média, com algumas interrupções e suporte adequado ao cliente, mas nada excepcional.
  - 7-8: Confiabilidade acima da média, com interrupções pouco frequentes e bom suporte ao cliente, mas com espaço para melhorias.
  - 9-10: Extremamente confiável, com interrupções mínimas e suporte abrangente ao cliente.

Tabela 2 – Matriz de decisão de gateway de pagamento.

Gateways	**Métodos de pagamento**	**Taxas**	**Segurança**	**Integração**	**Usabilidade**	**Confiabilidade**	**Resultado**
Stripe	5	4	6	8	8	5	5,9
PagSeguro	9	5	5	5	7	9	7,2
MercadoPago	8	6	5	5	5	7	6,3
PayPal	7	5	6	6	6	8	6,6
Juno	1	9	7	7	7	6	5,4
Cielo	4	3	8	6	6	9	6

Fonte: De autoria própria.

A Tabela 2 compara os gateways de pagamento Stripe, PagSeguro, MercadoPago, PayPal, Juno, e Cielo. Esses gateways de pagamento foram escolhidos para comparação por não possuírem pagamentos mensais, e possuírem informações de fácil acesso para a avaliação.

Sendo assim foi escolhido para a integração inicial na API de gerenciamento os gateways PagSeguro, Mercado Pago e Pay Pal, por terem as maiores pontuações dentre os avaliados.

5.5 Backlog

Visando alcançar esse produto mínimo viável foi criado um Backlog com dois Épicos categorizados em Pagamento Digital e Manutenção. Cada Épico deve ter tarefas que podem ou não estar vinculadas a uma história de usuário representados na Tabela 3.

O épico Pagamento Digital é referente a funcionalidade que precisa de manutenção no AgroMart, pois não está funcionando. Possui quatro historias de usuário que contam a necessidade de poder pagar e visualizar extratos. Cada uma dessas US possuem sua definição de preparado e definição de finalizado.

Para o épico de Manutenção não há história de usuário, já que é uma necessidade de desenvolvimento e não de um usuário. No entanto, o épico possui definição de finalizado.

Tabela 3 – Épicos e US

Épicos	US	História de usuário	Prioridade	Definição de Preparado	Definição de Finalizado
Manutenção			Urgente		- Atualizar pacote do Strapi - Remover módulos não utilizados na API
Pagamento Digital	US01	Eu como agricultor gostaria de poder escolher qual gateway de pagamento utilizar para melhor comodidade	Urgente	- Definir gateway de pagamento	- Visualizar tela de configuração de gateway de pagamento na API - Integrar Api com gateway de pagamento - Salvar transação no Banco de Dados
Pagamento Digital	US02	Eu como agricultor gostaria de visualizar o extrato financeiro dos produtos vendidos para melhor controle financeiro	Média	- US01 Estiver finalizada	- Visualizar tela de extrato na Api - Ser possível filtrar e buscar no historio de pagamentos - Visualizar transações do Banco de Dados
Pagamento Digital	US03	Eu como agricultor gostaria de gerar link de pagamento para que o co-agricultor possa pagar o produto de forma integrada	Média	- US01 Estiver finalizada	- Visualizar tela de gerar pagamento - Escolher qual o gateway de pagamento - Escolher produto - Selecionar co-agricultor
Pagamento Digital	US04	Eu como co-agricultor gostaria de pagar meus produtos através do link gerado pelo agricultor para melhor conveniência	Baixa	- US03 Estiver finalizada	- Visualizar tela temporária de pagamento - Ser possível escolher meio de pagamento - Ser possível efetuar pagamento com sucesso - Salvar transação no Banco de Dados

Fonte: autores.

Com os Épicos e Histórias de Usuários bem definidas foram criadas tarefas para auxiliar no desenvolvimento descritas na tabela 4. Cada uma dessa tarefa precisou estar vinculada a um Épico.

Tabela 4 – Backlog Priorizado

Épicos	ID no Github	Título	Prioridade	Tamanho	Iteração
Manutenção	#29	Atualizar Versão do Strapi	Urgente	G-Grande	Sprint 1
Manutenção	#31	Migrar integração do Strapi com PostgreSQL	Urgente	Médio	Sprint 1
Manutenção	#43	Gerenciar extratos	Urgente	Pequeno	Sprint 2
Manutenção	#41	Gerenciar notificações	Urgente	Grande	Sprint 2
Manutenção	#39	Gerenciar endereços	Urgente	Pequeno	Sprint 2
Manutenção	#35	Gerenciar planos	Urgente	Médio	Sprint 2
Manutenção	#36	Gerenciar Cestas	Urgente	Pequeno	Sprint 2
Manutenção	#36	Gerenciar Produtos Avulsos	Urgente	Médio	Sprint 2
Manutenção	#36	Automatizar deploy no heroku	Urgente	Médio	Sprint 4
Manutenção	#48	Teste de aceitação criar conta de usuário	Alta	Médio	Sprint 5
Manutenção	#50	Teste de aceitação gerenciar assinantes	Alta	Médio	Sprint 5
Manutenção	#52	Teste de aceitação gerenciar endereços	Alta	Médio	Sprint 5
Manutenção	#54	Teste de aceitação gerenciar lojas	Alta	Médio	Sprint 5
Manutenção	#56	Teste de aceitação gerenciar produtos avulsos	Alta	Pequeno	Sprint 5
Manutenção	#58	Teste de aceitação de cestas	Alta	Pequeno	Sprint 5
Manutenção	#60	Teste de aceitação gerenciar planos	Alta	Pequeno	Sprint 5
Manutenção	#62	Teste de aceitação gerenciar extrato	Alta	Pequeno	Sprint 5
Manutenção	#66	Teste de aceitação notificação	Alta	Grande	Sprint 6
Pagamento Digital	#70	US01	Urgente	G-Grande	Sprint 7
Pagamento Digital	#72	US02	Média	Médio	Sprint 8
Pagamento Digital	#74	US03	Média	Grande	Sprint 9
Pagamento Digital	#76	US04	Baixa	Baixa	Sprint 9

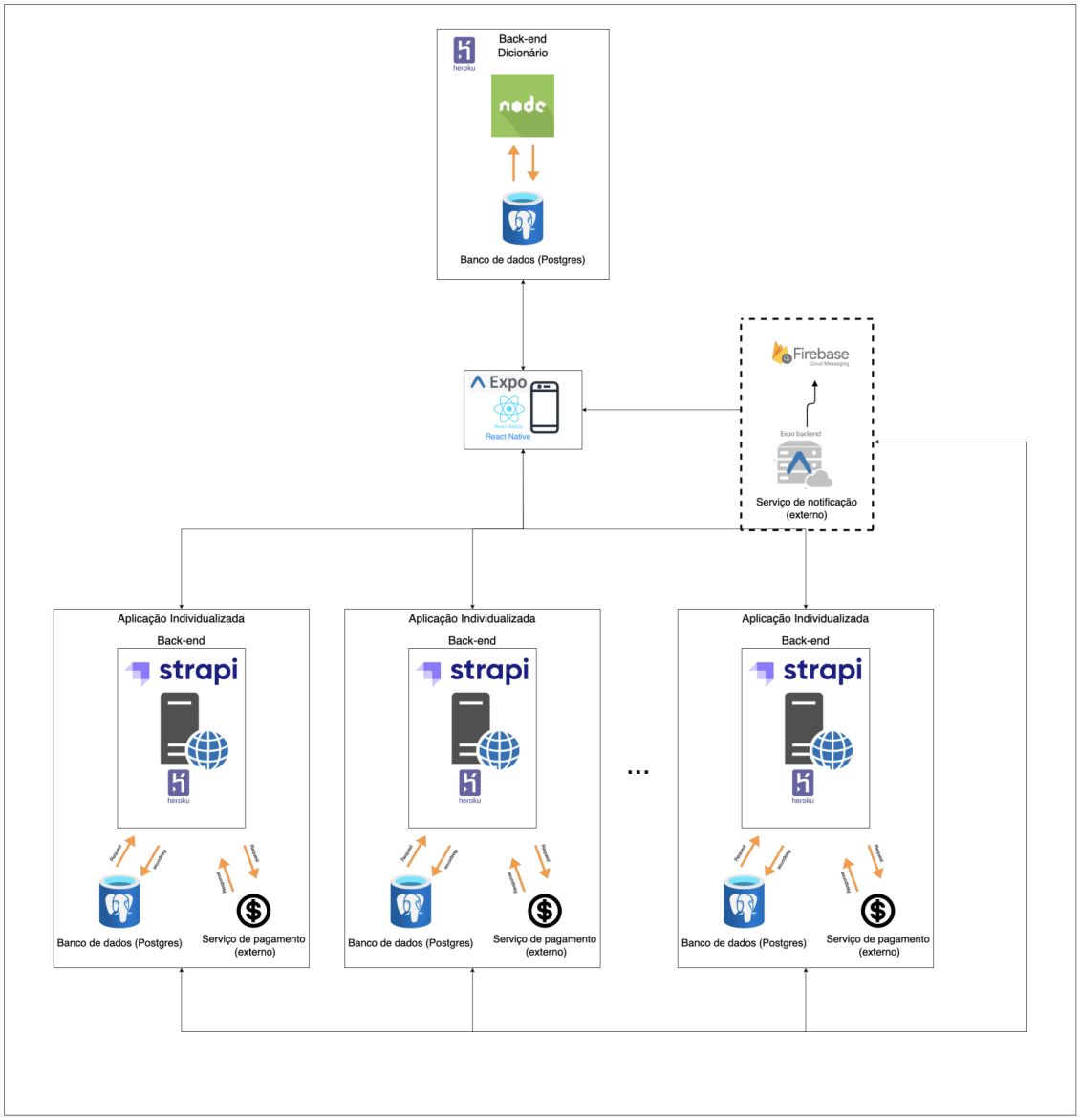
Fonte: autores.

Com base na norma Std 1219 da IEEE, mencionada na subseção 2.4 do capítulo 2, todas as tarefas foram classificadas individualmente. E em sua maioria foram categorizadas como adaptativas.

5.6 Arquitetura

A arquitetura planeja por Rodrigues e Macêdo (2021) e evoluída por Corrêa e Veludo (2022) e Freitas e Cella (2023) segue o modelo cliente-servidor com aplicação distribuída. É representada na Figura 8 por um diagrama de componentes que apresenta a estrutura decomposta em componentes arquitetonicamente significativos. A estrutura de implementação apresenta a organização das dependências e dos pacotes ilustrativamente.

Figura 8 – Diagrama de Componentes Agromart



Fonte: Freitas e Cella (2023).

A Figura 8, apresenta uma aplicação mobile que se comunicará com um dicionário que armazena o localizador uniforme de recursos (URL) de cada CMS Strapi individualizado. Com esse URL, a comunicação cliente-servidor entre App e a API ocorre com o Protocolo de Transferência de Hipertexto (HTTP). Cada API Strapi tem seu banco de dados PostgreSQL e interface providenciado pelo CMS (FREITAS; CELLA, 2023).

Os serviços possuem JavaScript como a linguagem de programação dominante no sistema Agromart. JavaScript é uma linguagem de programação de alto nível, interpretada de forma estruturada com tipagem dinâmica fraca. Ele é uma das três principais tecnologias da *World Wide Web* (JAVASCRIPT, 2022).

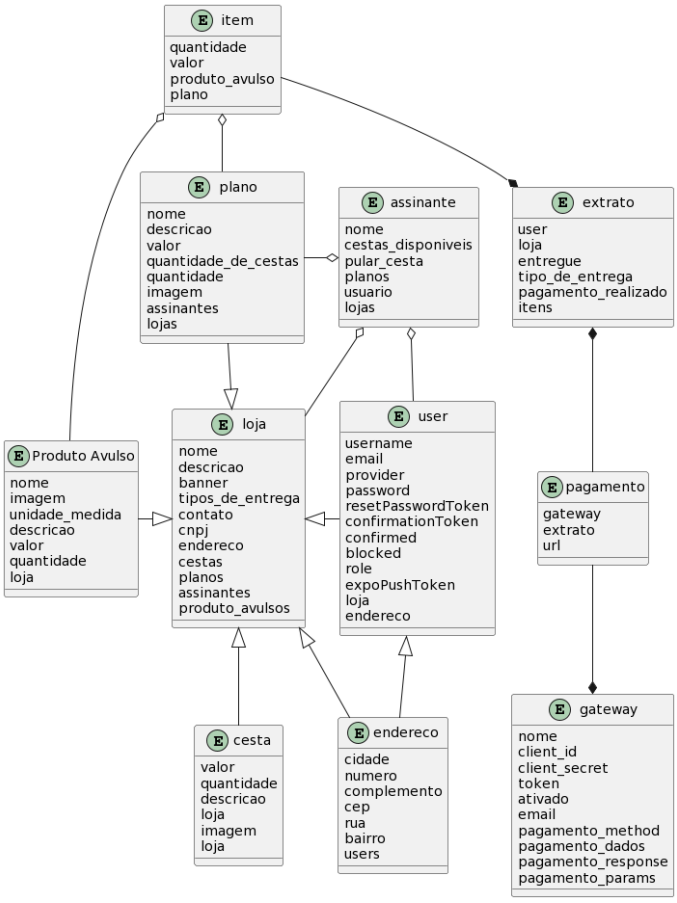
A proposta desse trabalho incluiu uma plataforma de pagamento digital. Para

isso, foi feito uma matriz de decisão entre as ferramentas de pagamento conforme descrito na Seção 5.4 onde os gateways selecionados foram PayPal, Mercado Pago e PagSeguro. Com essa escolha feita, pode-se desenvolver uma extensão no Strapi para Pagamentos, responsável por integrar pagamentos digitais.

5.6.1 Visão de Dados

Auxiliando o entendimento da solução, foi feito uma diagrama de entidade. Esse diagrama, representado na Figura 9, equivale a visão de dados que descreve como o sistema persiste as informações. Descrevendo a decomposição em entidades que fornece informações sobre como o sistema é estruturado e relacionado.

Figura 9 – Diagrama de Entidades



Fonte: autores.

Nesse sistema foram eliminadas as informações referentes a aplicação Juno, atualizado o pacote e entidade de notificação e adicionada informações sobre formas de pagamento e gateway com seus devidos simbolo de acesso. As entidades da API do Agromart foram despoluídas restando 8 entidades já existentes e adicionadas 3, são elas:

- **Item**: refere-se a um plano ou um produto gerando seu valor e quantidade em um pedido;
- **Pagamento**: refere-se a como o pagamento foi realizado, e a url temporária gerada;
- **Gateway**: trata-se da entidade reservada para salvar informações sobre o *Gateway* de Pagamento utilizado e seu *token* de acesso.

5.7 Resumo do capítulo

O capítulo descreve o projeto Agromart, que surgiu durante o hackathon da UnB-FGA 2020 com o objetivo de criar um software que conectasse agricultores e consumidores, facilitando a venda de produtos agrícolas durante a pandemia. O projeto passou por diversas evoluções e adaptações, sendo desenvolvido um aplicativo e uma interface web para gerenciamento.

Menciona o uso da metodologia Lean Inception para aplicar oficinas durante o desenvolvimento desse trabalho. E Como os dados levantados pelos agricultores na etapa de reunião com mantenedores foram utilizados para criar artefatos como personas, jornada do usuário, idealização e Produto Mínimo Viável. Relatando que o MVP desse trabalho é desenvolver um gerenciador/integrador de gateway de pagamento para o Agromart.

Na fase de Reunir com Mantenedores, foi identificado que o framework utilizado (Strapi) estava desatualizado e com módulos não mais utilizados, o que dificultava a inclusão de novos pacotes e poderia causar incompatibilidade de versões. Houve uma comparação entre o Strapi e outros frameworks populares como WordPress, Ghost, Joomla, PrestaShop e Drupal, considerando critério como afinidade, segurança, popularidade e suporte. O Strapi foi considerado a melhor opção com base nos resultados da comparação.

Dessa forma, foi decidido atualizar os módulos do Strapi. Além disso, foi desenvolvida uma Matriz de Decisão de Gateway de Pagamento para auxiliar no processo de escolha do gateway de pagamento a ser utilizado no MVP. Considerando critério como compatibilidade, segurança, popularidade, suporte e integração. Com base na pontuação obtida, os gateways PagSeguro, MercadoPago e PayPal são escolhidos para a integração inicial na API do AgroMart, por terem as maiores pontuações entre os avaliados.

Além disso, é apresentado um backlog com épicos e histórias de usuário relacionados aos pagamentos digitais e manutenção do sistema. O backlog é dividido em duas partes e inclui tarefas que devem ser realizadas para o desenvolvimento do produto mínimo viável.

A arquitetura do sistema é representada por um diagrama de componentes, que mostra a estrutura da aplicação distribuída, com comunicação cliente-servidor utilizando o protocolo HTTP.

Também é apresentada uma visão de dados por meio de um diagrama de entidades, que descreve como as informações são persistidas no sistema. As entidades existentes são refinadas e três novas entidades são adicionadas: Item, Pagamento e Gateway.

No próximo capítulo, será abordada a implementação das funcionalidades relacionadas aos pagamentos digitais no AgroMart.

6 Resultado

Nesse capítulo será descrito os artefatos gerados na execução da metodologia no Fluxo de TCC 2 apresentado na Figura 5. Serão apresentados as tarefas elaboradas durante o desenvolvimento da Manutenção API, Plugin Pagamento, Documentar Achados e Melhorar interação através das seções de Manutenção API, Testes, Middlewares, Deploy, Casos de Uso, Interface, Visão de Processo, Visão de Dados, Documentação de tutorial e Resumo do Capítulo.

6.1 Manutenção API

Ao atualizar pacotes de desenvolvimento cria-se o risco de as bibliotecas terem falta de compatibilidade e por isso uma funcionalidade parar de funcionar. Por esse motivo, nesse trabalho, como descrito na Tabela 5 ao atualizar o Strapi da versão legado para a versão mais atual, foi necessário atualizar também o Node e as dependências.

Tabela 5 – Versões de pacotes

Pacotes	Versão legado	Versão atualizada
@strapi/strapi	3.4.6	4.6.2
Node	>=10.16.0 <=14.x.x	>=18.x.x
axios	0.26.1	1.4.0
expo-server-sdk	3.6.0	"@surunnuage/strapi-plugin-expo-notifications- 1.0.7v
faker	5.4.0	
firebase-admin	9.12.0	
jsrsasign	10.5.17	
knex	0.20.0	
qs	6.10.3	
strapi-plugin-entity-relationship-chart	3.1.0	
text-encoding	0.7.0	
yarn	1.22.17	1.22.xx
pg	latest	8.9.0
jest		29.5.0
supertest		6.3.3
nock		13.3.1
mercadopago		1.5.16
paypal-rest-sdk		1.8.1

Fonte: De autoria própria.

Nessa Tabela 5 é possível observar que foram removidos pacotes que não eram utilizados, como: faker, firebase-admin, jsrsasign, knex, qs, strapi-plugin-entity-relationship-chart e text-encoding. No entanto, foram adicionados novos pacotes para a realização de testes de integração e simulação de requisições nos testes, esses pacotes são: jest, supertest e nock. Os pacotes mercadopago e paypal-rest-sdk também foram adicinados, nesse caso para fazer a integração com os gateways de pagamentos.

O pacote expo-server-sdk foi removido para ser substituído pelo "@surunnuage/strapi-plugin-expo-notifications"na versão 1.0.7. Isso ocorreu, pois o novo módulo é um pacote exclusivo do Strapi que se integra de melhor forma a aplicação Agromart.

6.2 Testes

Conforme a norma IEEE Std 1219 do IEEE descrita na sessão 2.4, para validar se as modificações nos pacotes e dados não introduzem falhas ou combinações tóxicas foi necessário realizar testes de aceitação. As funcionalidades do AgroMart para esses testes são:

- Criação de conta de usuário e Autenticação;
- Gerenciar assinantes;
- Gerenciar endereços;
- Gerenciar lojas;
- Gerenciar cestas;
- Gerenciar produtos avulsos;
- Gerenciar planos;
- Gerenciar extratos;
- Gerenciar notificação;

Durante o subprocesso de Melhorar Interação, conforme apresentado na seção 4.2.2, foram realizados testes, que incluíram dois tipos distintos: testes de integração e testes aceitação. O objetivo desses testes foi verificar o funcionamento da versão 3 do Strapi e compará-lo com a versão 4.

Os cenários utilizados para os testes de aceitação foram especialmente desenvolvidos com base no uso do aplicativo Android, levando em consideração as necessidades dos usuários. O foco era assegurar que o software atendesse de forma adequada às suas demandas e expectativas. Esses testes concentraram-se em validar as funcionalidades-chave do AgroMart. Dessa forma, verificando se as mudanças realizadas não alteraram o comportamento esperado do software.

Os testes de integração foram fundamentais para verificar a interação harmoniosa entre os diferentes módulos e componentes do sistema, garantindo a integridade e consistência das funcionalidades após as atualizações e aprimoramentos.Além disso, foi

utilizado de um Banco de Dados durante a realização dos testes para garantir a integridade e consistência dos dados. A configuração desse Banco foi realizada por um novo *docker-compose* apresentado na Figura 10, usando de um arquivo *docker-compose.dev.yml* para não alterar o comportamento do desenvolvimento local.

Figura 10 – Docker Compose Dev

```yaml
version: "3.9"

services:
  agromart_db_service_test:
    container_name: agromart_db_container_test
    restart: always
    image: postgres
    env_file: .env
    environment:
      POSTGRES_DB: ${DATABASE_NAME}-test
      POSTGRES_USER: ${DATABASE_USERNAME}
      POSTGRES_PASSWORD: ${DATABASE_PASSWORD}
    ports:
      - '5432:5432'
    volumes:
      - ../data:/var/lib/postgresql/test
```

Fonte: autores.

Os pacotes jest e supertest foram utilizados para a configuração e desenvolvimento de código e o nock para a simulação de requisições. Durante essa estruturação foi encontrado a necessidade de excluir os registro após a execução do teste, por isso foi criado um código apresentado na Figura 11 para efetuar a configuração e limpeza do banco de dados automaticamente.

Figura 11 – Configuração de Testes

```
const Strapi = require("@strapi/strapi");
const { Pool } = require('pg');

let instance;

async function setupStrapi() {
  if (!instance) {
    await Strapi().load();
    instance = strapi;

    await instance.server.mount();
  }
  return instance;
}

const cleanupStrapi = async () => {
  const tableNamesQuery = strapi.db.connection.raw(`
  SELECT tablename
  FROM pg_tables
  WHERE schemaname = 'public'
  `);

  const resp = await tableNamesQuery.then();

  const tableNames = await resp.rows.map(row => row.tablename);

  // Itera sobre cada tabela e exclui seus registros
  for (const tableName of tableNames) {
    const clearTableQuery = strapi.db.connection.raw(`
      TRUNCATE ${tableName} RESTART IDENTITY CASCADE;
    `);

    await clearTableQuery;
  }
};

async function destroyStrapi() {
  await strapi.server.httpServer.close();
  strapi.db.connection.destroy();

  // Disconnect from the Strapi app
  await strapi.destroy();
}


module.exports = { setupStrapi, cleanupStrapi, destroyStrapi };
```

Fonte: autores.

Além disso, foi criado um usuário de teste padrão para uniformizar a forma de autenticação das requisições apresentado na Figura 12.

Figura 12 – Usuário de teste padrão

```
const request = require('supertest');

var user;
var jwt;
const mockUserData = {
    username: "userteste",
    email: "userteste@strapi.com",
    password: "1234userteste"
};

async function setupUser() {
    response = await request(strapi.server.httpServer)
            .post("/auth/local/register")
            .send({
                username: mockUserData.username,
                password: mockUserData.password,
                email: mockUserData.email,})
            .set("accept", "application/json")
            .set("Content-Type", "application/json");

    user = response.body.user
    jwt = response.body.jwt
}

function getUser(){
    return user
}

function getJWT(){
    return jwt
}

module.exports = { getUser, getJWT, setupUser };
```

Fonte: autores.

Com essas etapas realizadas foi possível escrever testes de integração para todos os endpoints que o aplicativo do Agromart realizados para a API. Na Figura 13 é possível ver o teste criado para o endpoint de devices. Nessa imagem, representa dois teste, um quando o sistema não encontrar o device e outro para quando encontrar.

Figura 13 – Parte do teste de device

```
const request = require('supertest');

describe('Tester para achar o User Expo Push Token', () => {
  const path = "/devices/user/";
  const mockUserData = {
    username: "userExpo",
    email: "userExpo@strapi.com",
    provider: "local",
    password: "1234userExpo",
    confirmed: true,
    blocked: null,
    expoPushToken: null
  };

  it("Não deve encontrar o device", async () => {
    await request(strapi.server.httpServer)
      .get(path + user.id)
      .set("accept", "application/json")
      .set("Authorization",`Bearer  ${jwt}`)
      .set("Content-Type", "application/json")
      .expect("Content-Type", /json/)
      .expect(200)
      .then((data) => {
        expect(data.body).toBeDefined();
        expect(data.body.mensagem).toBe("Device não encontrado!");
        expect(data.body.status).toBe(404);
      });
  });

  it("Deve encontrar o device", async () => {
    const user = await strapi.plugins["users-permissions"].services.user.add({
      ...mockUserData,
      username: 'user2',
      expoPushToken: 'expoPushToken'
    });

    await request(strapi.server.httpServer)
      .get(path + user.id)
      .set("accept", "application/json")
      .set("Authorization",`Bearer  ${jwt}`)
      .set("Content-Type", "application/json")
      .expect("Content-Type", /json/)
      .expect(200)
      .then((data) => {
        expect(data.body).toBeDefined();
        expect(data.body.mensagem).toBe("Device encontrado!");
        expect(data.body.device_id).toBe("expoPushToken");
        expect(data.body.status).toBe(200);
      });
  });
```

Fonte: autores.

Com o desenvolvimento desses teste foi possível perceber que houveram muitas mudanças de integração entre as versões do Strapi. Surgiu então a necessidade da criação de funções de permissão inicial, pois as entidades geradas automaticamente não possuem essa possibilidade. A Figura 14 representa essa função de permissão que precisa ser chamada para permitir requisições autenticadas para as entidades criadas pelo próprio Strapi.

Figura 14 – Função de Bootstrap

```
async function setAuthenticatedPermissions(newPermissions, publicRole) {
  const allPermissionsToCreate = [];
  Object.keys(newPermissions).map((controller) => {
    const actions = newPermissions[controller];
    const permissionsToCreate = actions.map((action) => {
      return strapi.query("plugin::users-permissions.permission").create({
        data: {
          action: `api::${controller}.${controller}.${action}`,
          role: publicRole.id,
        },
      });
    });
    allPermissionsToCreate.push(...permissionsToCreate);
  });
  await Promise.all(allPermissionsToCreate);
}
```

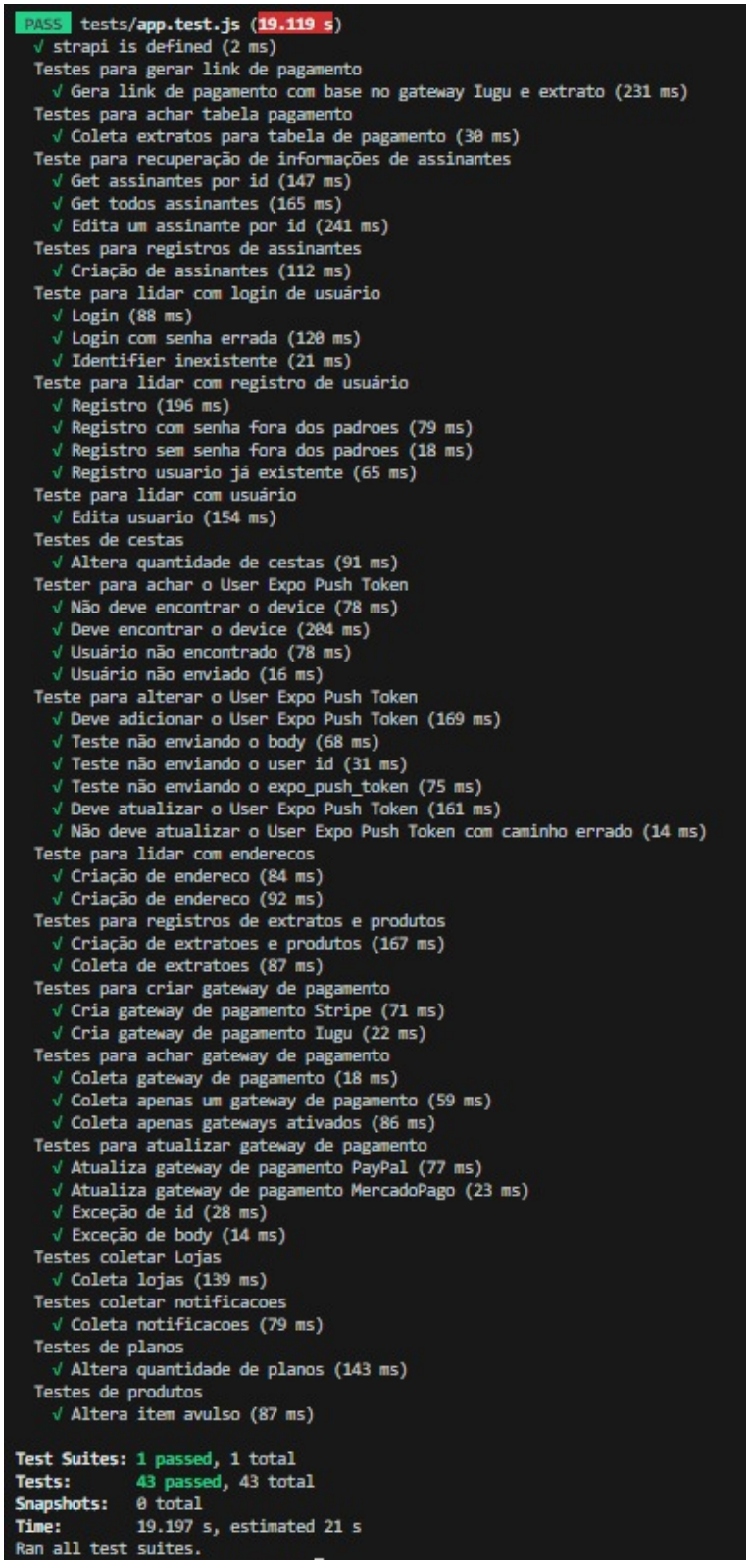
Fonte: autores.

Além disso, outra solução para garantir a inatingibilidade de integração entre a API com o aplicativo Android foi a adição de *middlewares* descritos na seção 6.3.

No caso do teste de aceitação, foi realizado após a finalização de todos os testes de integração. Foi feito a implantação do API no Heroku descritos na seção 6.4. Usando a API dicionário e a API Agromart no ambiente de produção foi executado o aplicativo Android localmente no ambiente de desenvolvimento. Dessa forma, foi possível testar e aceitar todas as mudanças feitas.

Devido a dificuldades técnicas não foi possível separar os testes em suítes diferentes, ou seja os testes se encontram na mesma coleção de casos de testes. Isso se deu pelo fato do comportamento do strapi com a ferramenta de testes jest. Apesar disso foram realizados testes de todos os módulos disponíveis. Totalizando 43 testes diferentes conforme mostra a Figura 15.

Figura 15 – Resultado dos testes na api Agromart



Fonte: autores.

6.3 Middlewares

Middleware é uma camada de software que funciona como intermédio entre diferentes componentes de um sistema. Fornecendo no caso desse trabalho transformações de dados e gerenciamento do fluxo de informações entre componentes. A criação desses intermediadores foram necessárias, pois da versão 3 para a versão 4 houve a mudança da estrutura de dados usada entre requisições.

Na versão 4, foi adicionado a qualquer requisição às entidades customizadas no lado do servidor um prefixo *api/*. Como no aplicativo de Android não havia esse prefixo e pensando em não alterar integrações já existentes, foi adicionado o intermediador *apiPrefix* representado na Figura 16. O objetivo deste intermediador é verificar se a URL já não começa com *api/* e se não faz parte das rotas padrão, apresentado na Figura 17. E após as verificações a rota é redirecionada com o prefixo *api/* quando necessário.

Figura 16 – API Prefix

```
const defaultRoute = require('./helpers/defaultRoutes');

module.exports = () => {
  return async (ctx, next) => {
    // Verifica se a URL não começa com "/api"
    if (!ctx.url.startsWith('/api/')
        && !defaultRoute.isDefaultRoute(ctx.url)) {
      // Redireciona para a mesma URL com o prefixo "/api" adicionado
      ctx.url = `/api${ctx.url}`;
    }
    await next();
  };
};
```

Fonte: autores.

Figura 17 – Rotas padrão

```
function isDefaultRoute(url) {
    if (!url.startsWith('/admin')
        && !url.startsWith('/i18n')
        && !url.startsWith('/content')
        && !url.startsWith('/upload')
        && !url.startsWith('/expo-notifications')
        && !url.startsWith('/plugins')
        && !url.startsWith('/pagamento')
        && !url.startsWith('/users-permissions')
        && !url.startsWith('/auth/google')
        && !url.startsWith('/auth/facebook')
        && !url.startsWith('/auth/github')
        && !url.startsWith('/email')
        && !url.startsWith('/_health')
        && url != '/'
        && url != '' ){
        return false
    } else{
        return true
    }
}

module.exports = { isDefaultRoute };
```

Fonte: autores.

Outra mudança entre as versões 3 e 4 se encontra no envio dos dados em requisição. Os quais devem estar envelopados em uma chave chamada *data*. O *middleware wrapBody* verifica se a propriedade *data* não está presente no corpo da solicitação. Se essa condição for verdadeira, significa que o corpo da solicitação não está no formato esperado. Esse código também utiliza da função das rotas padrão, se não for uma rota padrão o corpo original da solicitação será encapsulado em um objeto com a propriedade *data*, como mostrado Figura 18.

Figura 18 – wrapBody

```
1   const defaultRoute = require('./helpers/defaultRoutes');
2
3   module.exports = () => {
4       return async (ctx, next) => {
5           if (!("data" in ctx.request.body)
6               && !defaultRoute.isDefaultRoute(ctx.url)
7               && !ctx.url.startsWith('/api/auth/local')
8               && !ctx.url.startsWith('/api/users')
9               && !ctx.url.startsWith('/api/devices')) {
10              ctx.request.body =  { data: ctx.request.body };
11          }
12          await next();
13      };
14  };
```

Fonte: autores.

Por fim, o último intermediador que foi desenvolvido foi o *transformResponse*. Esse *middleware* tem a função oposta do *WrapBody*, enquanto um envelopa os dados enviados ao Strapi este envelopa os dados que o Strapi retorna. Em outras palavras, ele realiza algumas transformações no corpo da resposta antes de enviá-la ao cliente. A Figura 19 representa esse código em que verifica se a resposta possui um corpo e se contém uma propriedade *data*, além de não corresponder a uma rota padrão. Essa verificação é usada para determinar se a transformação é necessária.

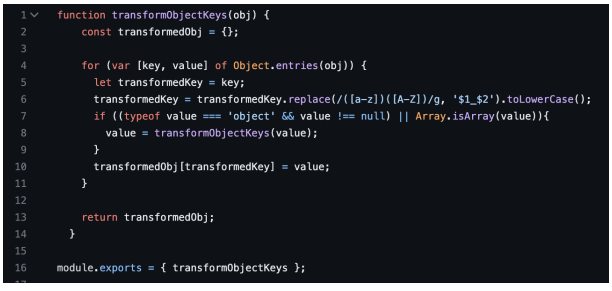
Figura 19 – transformResponse

```
const defaultRoute = require('./helpers/defaultRoutes');
const transformObject = require('./helpers/transformObject');

module.exports = () => {
    return async (ctx, next) => {
        await next();

        if (ctx.res.headersSent) {
          return;
        }
        if (ctx.response.body
            && ctx.response.body.data
            && !defaultRoute.isDefaultRoute(ctx.url)) {
            let data = ctx.response.body.data;

            if(data.attributes){
                const id = data.id;
                data = data.attributes;
                data.id = id;
                data = transformObject.transformObjectKeys(data)
            } else if(Array.isArray(data)){
                arr = []
                data.forEach(item => {
                    if(item.attributes){
                        const id = item.id;
                        temp = item.attributes;
                        temp.id = id;
                        arr.push(transformObject.transformObjectKeys(temp));
                    }
                })
                data = arr;
            }

            ctx.response.body = data;
        }

    };
};
```

Fonte: autores.

Se for necessário a função *transformObject* é chamada. A qual recebe um objeto como argumento e retorna um novo objeto com as chaves transformadas. Apresentada na Figura 20, a função itera sobre todas as propriedades do objeto fornecido e para cada par chave-valor, realiza transformações, por exemplo, a chave "nomeCompleto"seria transformada em "nome_completo".

Figura 20 – transformObject



```
function transformObjectKeys(obj) {
    const transformedObj = {};

    for (var [key, value] of Object.entries(obj)) {
      let transformedKey = key;
      transformedKey = transformedKey.replace(/([a-z])([A-Z])/g, '$1_$2').toLowerCase();
      if ((typeof value === 'object' && value !== null) || Array.isArray(value)){
        value = transformObjectKeys(value);
      }
      transformedObj[transformedKey] = value;
    }

    return transformedObj;
  }

module.exports = { transformObjectKeys };
```

Fonte: autores.

6.4 Implantação no Heroku

Parte dos testes de aceitação requeriam que fosse realizado em um ambiente de produção. Por esse motivo, foi necessário implantar a API do Agromart atualizada no

Heroku. Essa implantação era realizada por meio de um arquivo em *shell* que precisa de um interpretador na máquina para ser executado.

A partir do sub processo de Melhorar Interação apresentado na seção 4.2.2 e pensando no perfil de usuário final que utilizaria essa ferramenta esse arquivo em *shell* foi subsistido por um executável. Um arquivo executável é um arquivo binário que pode ser executado em diferentes sistemas operacionais sem a necessidade de ter um interpretador específico instalado. Isso torna o arquivo executável mais portátil ao poder ser distribuído e executado em diferentes plataformas sem preocupações com dependências externas.

A forma de implantação por meio do executável facilitou esse processo para que apenas a alteração do arquivo de configuração intitulado *config.ini*, observado na Figura 21, fosse necessária. E em seguida um clique duplo no arquivo *deploy.exe*

Figura 21 – Aquivo confi.ini

```
[heroku]
api_key = INSIRA_AQUI_SUA_API_KEY_DO_HEROKU

[csa]
nome_csa = NOME_DA_CSA
responsavel_csa = RESPONSAVEL_DA_CSA
email = EMAIL_DA_CSA
```

Fonte: autores.

Como pode ser observado na Figura 21, o arquivo consiste apenas de quatro campos que requerem o nome da csa, o seu responsável, o e-mail e a chave da api do Heroku. A qual pode ser encontrada facilmente na pagina inicial do perfil do usuário Heroku.

Esse executável foi desenvolvido na linguagem de programação Python em um novo repositório GitHub[1], privado, na organização Agromart. Foi feito dessa forma para impedir alterações indesejadas. O executável teve todas suas funções criadas com tratamento de exceção, o que ajuda na resistência a erros, e traz ao usuário uma ferramenta de instalação mais robusta e confiável.

Além disso foi criado um tutorial, que será abordado mais profundamente na Sessão 6.6, com a finalidade de facilitar ainda mais a implementação por parte do usuário final. Contudo, o executável sofre de algumas limitações, funcionando somente para publicação da api na plataforma Heroku. E, assim como no shell script, houveram dependências externas que não puderam ser eliminadas como a dependência do Git pelo Heroku, e a dependência do Heroku CLI para a realização da implantação.

[1] Disponível em: https://github.com/AgroMart/deploy-automatico. Acesso em: 10 jul, 2023.

6.5 Módulo de Pagamento

Após a finalização da manutenção e atualização dos módulos foi possível trabalhar no desenvolvimento do módulo de pagamento. Esse módulo visa permitir a criação de páginas de pagamento de contas gerenciadas pelos gateways de pagamento. Conforme apresentado na Seção 5.3 foram escolhidos para a integração inicial na API de gerenciamento os gateways PagSeguro, Mercado Pago e Pay Pal.

A seguir será apresentado os casos de uso desse novo módulo, a visão de dados, a visão de processo, as interfaces geradas e um sumário de desenvolvimento.

6.5.1 Visão de Casos de Uso

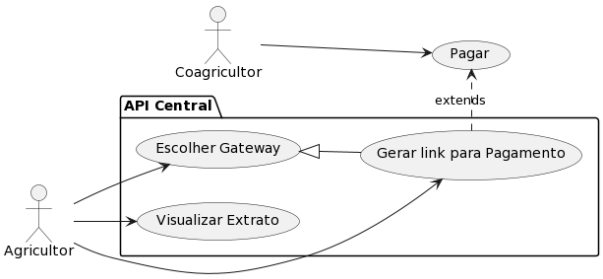
A fim de representar os possíveis comportamentos da aplicação central em Strapi ao integrar com o novo serviço API de Pagamento em Node, foi usado o diagrama de casos de uso. Isso porque esses diagramas representam algumas funcionalidades que são significativas do sistema final. Essas funcionalidades são:

- Caso 1 - Escolher Gateway
- Caso 2 - Visualizar Extrato
- Caso 3 - Gerar link para Pagamento
- Caso 4 - Pagar

Para esse projeto, foi criado um diagrama com os atores: agricultor, administrador do sistema; o co-agricultor, usuário cliente que utiliza para pagamento.

Na Figura 22, é possível ver o diagrama dos casos 1, 2, 3 e 4, cenário onde o ator agricultor escolhe o gateway de pagamento e gera link para o pagamento onde o co-agricultor pode pagar. Esse diagrama também indica que quando o caso 3 for executado, deve-se ter escolhido um gateway e o caso 4 poderá ser executado. Além disso, o agricultor pode visualizar o extrato.

Figura 22 – Diagrama de Caso de Uso

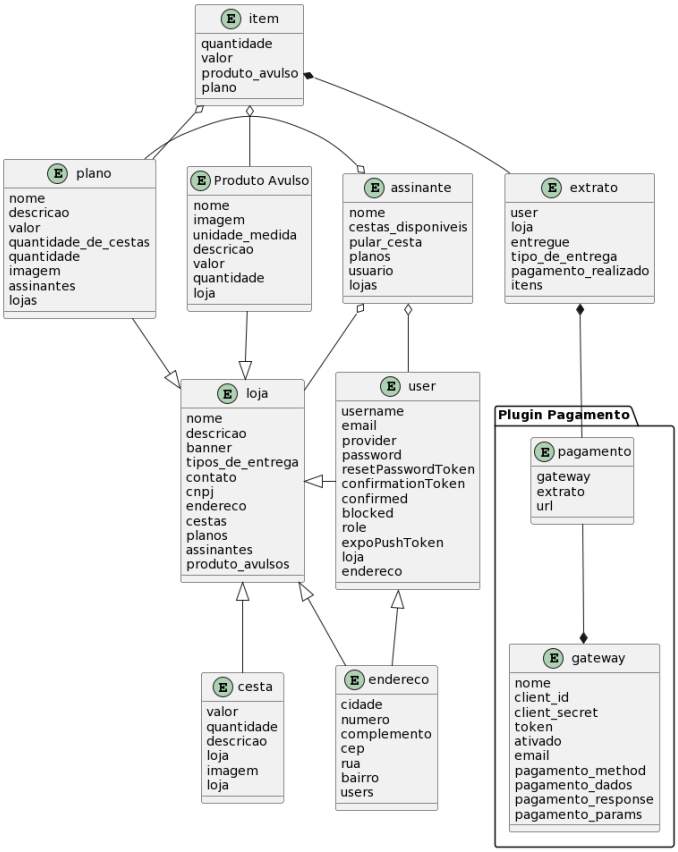


Fonte: De autoria própria.

6.5.2 Visão de Dados

Ao compreender os casos de uso da extensão de Pagamento na API do Agromart foi desenvolvido o diagrama de entidades apresentado na Figura 9. Após eliminado as entidades descontinuadas ou não utilizadas foram adicionas 3 entidades para o desenvolvimento do plugin de Pagamento: Item, Pagamento e Gateway. Essas entidades foram enfatizadas na Figura 23.

Figura 23 – Diagrama de Entidades - Módulo de Pagamento



Fonte: autores.

Nessa Figura 23, é possível observar que apesar da entidade Item ter sido adicionada para o desenvolvimento do módulo ela não pertence ao módulo. Isso acontece, por ser necessário modificar o tipo de dados do atributo itens na entidade Extrato de indp de um JSON para um objeto de itens. Essa atualização foi feita para prevenir que os usuários finais inserissem dados incorretos o que faria com que o poderia causar falhas no sistema.

Nas novas entidades adicionadas temos o Gateway que armazena as informações de um gateway externo de pagamento para configurar a comunicação e a entidade Pagamento que armazena a relação de um Extrato com um Gateway.

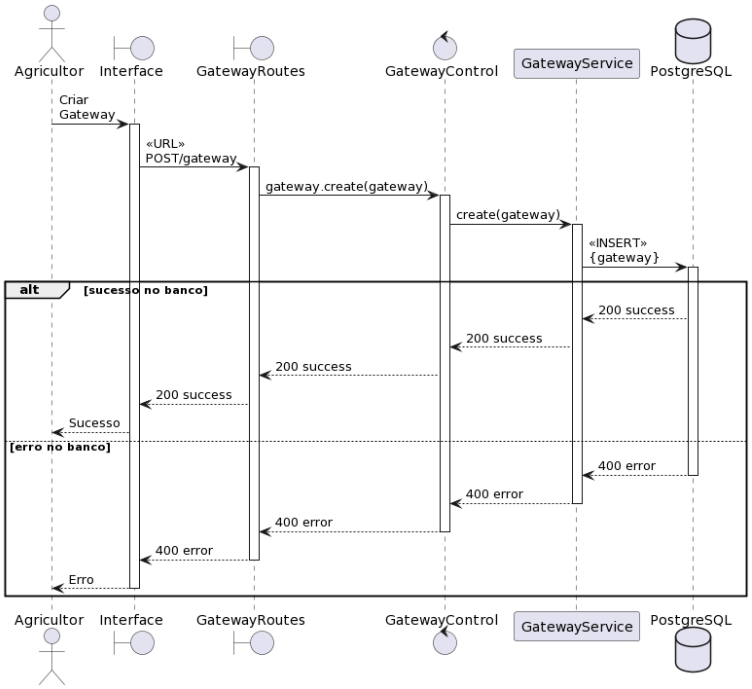
6.5.3 Visão do Processo

Representando a visão do processo foi utilizado o diagrama de sequência que é um diagrama de UML dinâmica que mostra uma sequência de eventos. Isso é, enfatiza a ordenação temporal das mensagens ao mostrar interações na ordem em que ocorrem. Essa categoria de diagrama auxilia a entender os requisitos de um sistema. Além disso, mostra como mensagens são trocadas entre os componentes.

Foram utilizados, porque permitem um melhor entendimento do projeto e aplicação. Com ele pode-se identificar como os objetos no sistema interagem entre si para implementar as funcionalidades.

Para esse projeto, foram criados 2 diagramas de sequência. Na Figura 24 é possível observar a sequência de eventos que ocorre para o registro de um gateway na perspectiva de AgroMart. Nesse processo é preciso que o Agricultor através da interface solicite a integração de um gateway ao seu token de acesso. Visando que esse token seja salvo no banco de dados, a interface do Strapi requisita à API que acione o banco de dados. Esse diagrama representa o caso de uso 1.

Figura 24 – Diagrama de Sequência: Registro de Gateway

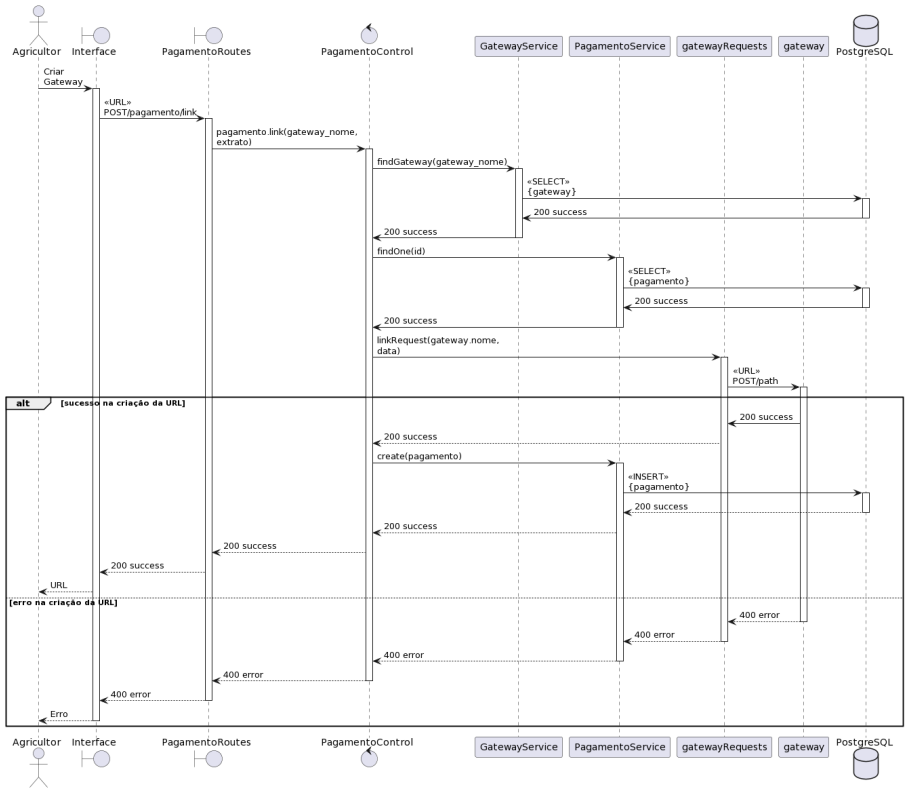


Fonte: De autoria própria.

Com o intuito de gerar a URL de pagamento, o caso de uso 3, na Figura 25 tem-se o processo que começa com a interação do usuário com a tela. O controlador de pagamentos é acionado que atribui o trabalho aos *services* de gateway e pagamento. Esses *services* buscam na base de dados as informações para serem enviadas à API de pagamento e

após requisitar ao gateway encarregado. Por fim, se tudo ocorrer conforme o espera, as informações serão salvas no banco de dados e será retornado a URL gerada para o usuário.

Figura 25 – Diagrama de Sequência: Gera URL

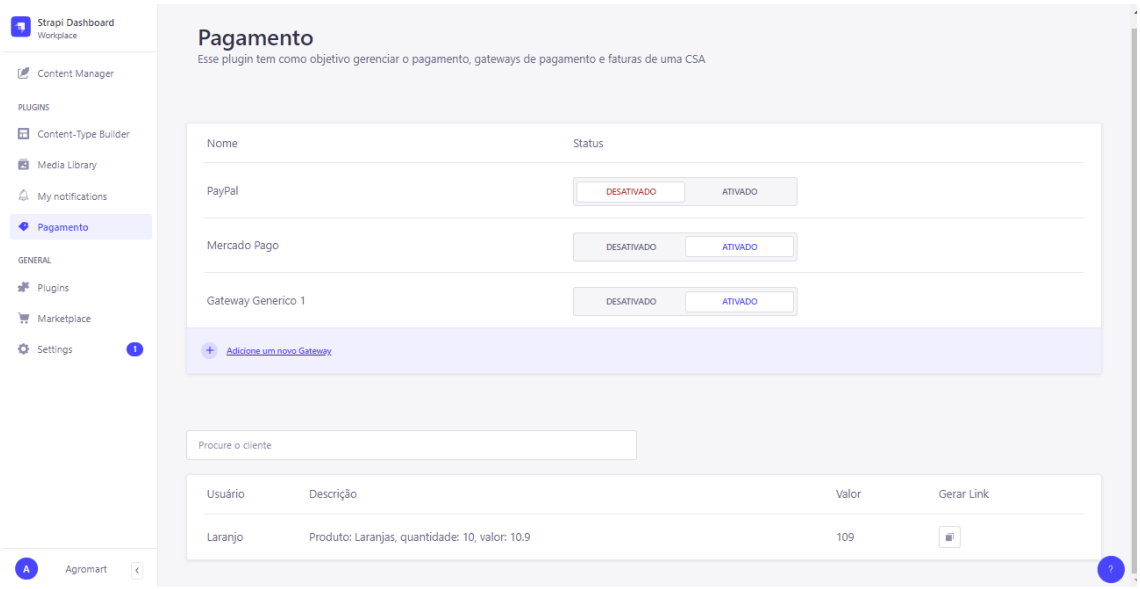


Fonte: De autoria própria.

6.5.4 Interface do Módulo

O Módulo de Pagamento encontra-se na página inicial do administrador na sessão de plugin. Essa página inicial do plugin possui informações dos extratos que ainda não foram pagos e os gateways integrados sejam eles ativos ou não, representa o caso de uso 2 visualizar extrato. Ainda nessa janela, é possível na lista de extratos, filtrar por cliente, ordenar pagamentos por valor, status, usuário ou descrição, e Gerar o link para Pagamento, o caso de uso 3. É possível ver essa página representada na Figura 26.

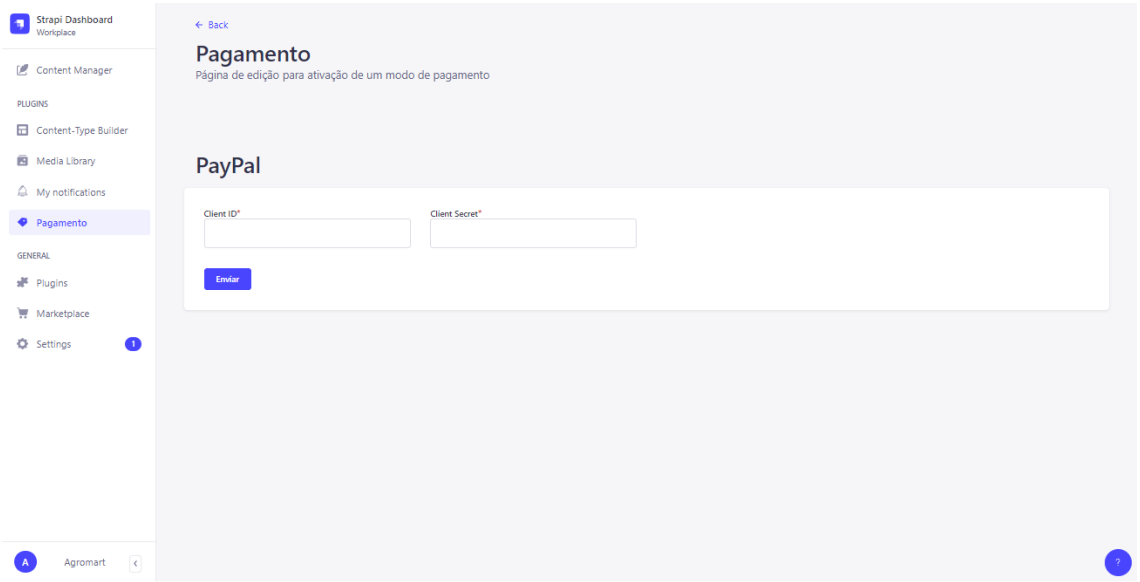
Figura 26 – Página Inicial do Plugin



Fonte: autores.

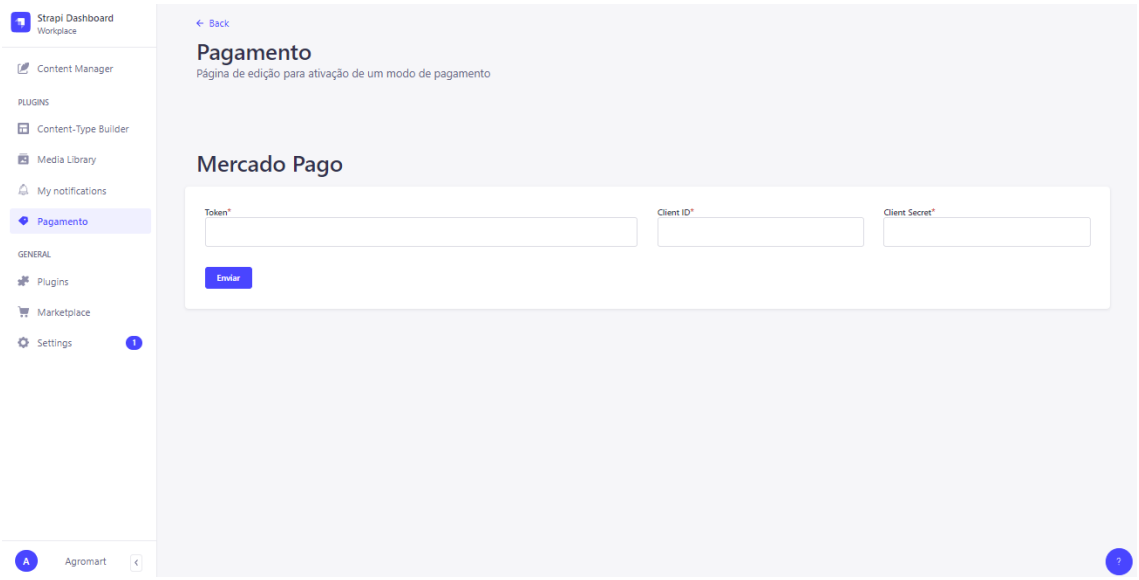
A partir da página inicial é possível entrar nas páginas de configuração dos gateways, seja na Figura 27 do Pay Pal, na Figura 28 do Mercado Pago e na Figura 29 de Gateways Customizados. Essas páginas são essenciais para configurar as informações fundamentais para a integração e representam parte do caso de uso 1, escolher gateway.

Figura 27 – Página de Configuração do Pay Pal



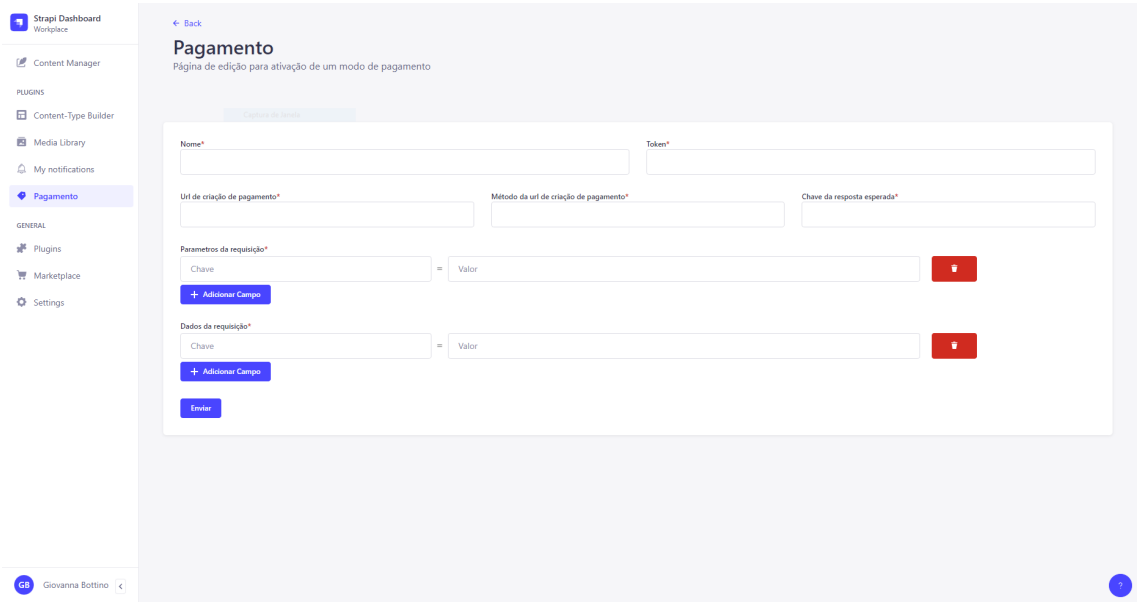
Fonte: autores.

Figura 28 – Página de Configuração do Mercado Pago



Fonte: autores.

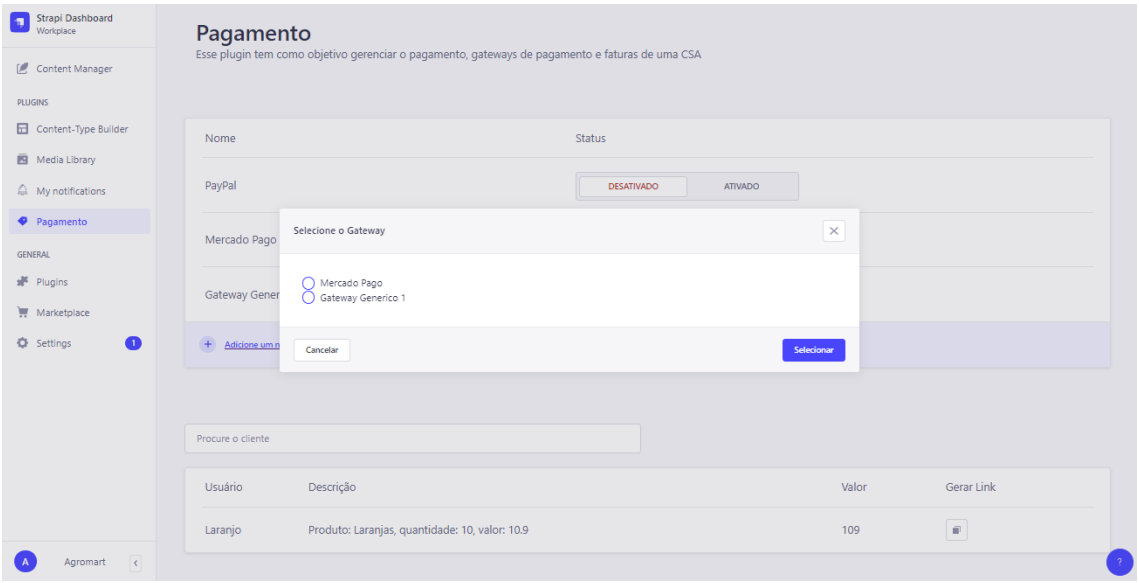
Figura 29 – Página de Configuração do Gateway Customizado



Fonte: autores.

O caso de uso 1, escolher Gateway, também é utilizado na Figura 30 que representa uma janela sobreposta. Essa janela, apenas aparece no momento que existe pelo menos dois gateways ativados. Quando o gateway é selecionado é possível realizar o caso de uso 4, Pagar, com o link gerado.

Figura 30 – Janela sobreposta



Fonte: autores.

6.5.5 Módulo de Pagamento

O objetivo do módulo é permitir a criação de páginas de pagamento gerenciadas por gateways de pagamento, como PagSeguro, Mercado Pago e PayPal.

Foram identificados quatro casos de uso significativos para o sistema: escolher gateway, visualizar extrato, gerar link para pagamento e pagar. Esses casos de uso foram representados em um diagrama de casos de uso, destacando as interações entre os atores envolvidos.

Um diagrama de entidades foi desenvolvido para representar as informações do módulo de pagamento. Três entidades foram adicionadas: Item, Pagamento e Gateway. Essas entidades foram utilizadas para o desenvolvimento do plugin de pagamento, estabelecendo a relação entre um Extrato e um Gateway.

A visão do processo foi representada por meio de diagramas de sequência, que demonstram a sequência de eventos e interações entre os componentes do sistema. Foram criados dois diagramas: um para o registro de um gateway e outro para a geração da URL de pagamento.

A interface do módulo de pagamento está disponível na página inicial do administrador, na seção de plugins. Nessa página, são exibidas informações es sobre os extratos não pagos e os gateways integrados. Além disso, é possível acessar as páginas de configuração dos gateways para definir as informações necessárias para a integração.

O módulo de pagamento foi desenvolvido genérico o suficiente para ser possível adicionar novas integrações. Para isso, basta adicionar no arquivo gatewayRequest a con-

figuração necessária, representado na Figura 31. Esse processo foi aplicado para todos os gateways PagSeguro, Mercado Pago e Pay Pal. No entanto, é importante destacar que a funcionalidade de checkout transparente, usada para gerar a URL no PagSeguro foi descontinuada. Por esse motivo, o PagSeguro foi removido para a entrega do trabalho, apesar de ter sido integrado.

Figura 31 – gatewayRequest

```
/**
 * Axios with a custom config.
 */
const mercadopago = require('./gateways/mercadoPago');
const paypal = require('./gateways/paypal');
const padrao = require('./gateways/padrao');

const gatewayRequests = {
  authRequest: (gateway, data) => {
    switch (gateway) {
      case 'PayPal':
        return paypal.authRequest(data.client_secret, data.client_id);
      default:
        return false;
    }
  },
  linkRequest: (gateway, data) => {
    switch (gateway) {
      case 'PayPal':
        return paypal.linkRequest(data.gateway, data.extrato.itens);
      case 'Mercado Pago':
        return mercadopago.linkRequest(data.gateway, data.extrato.itens);
      default:
        return padrao.linkRequest(data.gateway, data.extrato);
    }
  }
};

module.exports = gatewayRequests;
```

Fonte: autores.

Além disso, esse módulo de pagamento possui a forma "padrão", quando o usuário final, não desenvolvedor, quer cadastrar um novo gateway de pagamento. Para isso ele precisa fornecer informações de URL, método, dados, parâmetros, reposta e de token da requisição ao gateway.

Em resumo, esta seção apresenta os resultados do desenvolvimento do módulo de pagamento, incluindo os casos de uso, a visão de dados, a visão do processo e a interface do módulo. O Mercado Pago e Pay Pal foram integrados com sucesso e o módulo permite a adição de novas integrações.

6.6 Ajuda Agromart

A página de Ajuda Agromart é uma documentação abrangente desenvolvida durante sub processo de Documentar Achados apresentado na Seção 4.2.2. O seu objetivo é fornecer suporte e orientações aos usuários finais do Agromart, os agricultores. Essa pá-

gina contém tutoriais e instruções detalhadas sobre como utilizar os recursos do Agromart eficientemente e maximizar os benefícios oferecidos pela plataforma.

O objetivo do módulo Ajuda Agromart é proporcionar aos usuários uma experiência de navegação mais intuitiva ao buscar informações específicas sobre o deploy (implantação) e o plugin de pagamento. Foi desenvolvido para disponibilizar informações e instruções sobre o uso da plataforma, adaptadas conforme o perfil e contexto do usuário. Essas informações são apresentadas claramente e passo a passo para facilitar a compreensão.

O conteúdo presente na Ajuda Agromart é organizado de maneira clara e transparente, garantindo que os usuários encontrem as informações relevantes de maneira rápida e fácil. Os tutoriais são estruturados em sequências lógicas de ações, facilitando o aprendizado e a aplicação prática das funcionalidades do Agromart.

Foi desenvolvido usando Jekyll e Bundler e implantado usando o GitHub Pages. Mantido pelo GitHub no repositório do ajuda-agromart[2] na organização do Agromart.

6.7 Resumo do Capítulo

O capítulo descreve os artefatos gerados na execução da metodologia no Fluxo de TCC 2 apresentado na Figura 5. Ele inicia explicando a atualização de pacotes de desenvolvimento e as mudanças necessárias para atualizar o Strapi de uma versão legada para uma versão mais atual. Apresentando as versões dos pacotes antes e depois da atualização. Alguns pacotes foram removidos por não serem utilizados, enquanto outros foram adicionados para realizar testes de integração e simulação de requisições. O pacote expo-server-sdk foi substituído pelo "@surunnuage/strapi-plugin-expo-notifications"na versão 1.0.7, pois o novo módulo é exclusivo do Strapi e integra-se melhor com a aplicação Agromart.

Foi necessário realizar testes de aceitação para validar as modificações nos pacotes e dados. Os testes incluíram diferentes funcionalidades do Agromart, como criação de conta de usuário, autenticação, gerenciamento de assinantes, endereços, lojas, produtos avulsos, cestas, planos, extratos e notificações. Foram utilizados testes de integração com um banco de dados para garantir a integridade dos dados e os pacotes jest, supertest e nock para configuração e desenvolvimento de código.

Foi desenvolvido um conjunto de middlewares para lidar com as mudanças na estrutura de dados entre as versões do Strapi. Esses middlewares incluem a verificação e redirecionamento de URLs com prefixo "api/", encapsulamento dos dados enviados em requisições e transformação das respostas antes de enviá-las ao cliente.

[2] Disponível em: https://github.com/AgroMart/ajuda-agromart. Acesso em: 10 jul, 2023.

Para a implantação no Heroku, foi desenvolvido um arquivo executável em Python que simplificou o processo de implantação, permitindo que o usuário final alterasse apenas o arquivo de configuração. Um tutorial foi criado para auxiliar na implementação do usuário final.

O capítulo também descreve o desenvolvimento do módulo de pagamento, que permite a criação de páginas de pagamento gerenciadas por gateways como PagSeguro, Mercado Pago e PayPal. Foram identificados casos de uso, criado um diagrama de entidades, diagramas de sequência e desenvolvida a interface do módulo. O módulo foi integrado com sucesso aos gateways Mercado Pago e PayPal, e o gateway PagSeguro foi removido devido à descontinuação de uma funcionalidade necessária.

Além disso, o capítulo apresenta o módulo de Ajuda Agromart, uma página de documentação abrangente desenvolvida para fornecer suporte e orientações aos usuários finais do Agromart. A página contém tutoriais e instruções detalhadas sobre o uso da plataforma, organizados de maneira clara e intuitiva. O módulo foi desenvolvido usando Jekyll e Bundler e implantado no GitHub Pages.

Em resumo, o resultado da realização da metodologia pode ser considerada aplicada com sucesso.

7 Conclusão

O presente trabalho teve como objetivo principal a criação de um módulo de integração de pagamentos bem como realizar a manutenção evolutiva do software Agromart, visando o aprimoramento de suas funcionalidades de gestão financeira e a atualização de seus componentes tecnológicos. Para alcançar esse objetivo, foram definidos objetivos específicos, como atualizar os frameworks utilizados, remover módulos não utilizados, implementar um integrador de pagamento digital e um gerenciador de extratos financeiros, além de esquematizar e documentar a solução proposta.

Ao longo do desenvolvimento do trabalho, foram realizadas diversas atividades, incluindo reuniões com os mantenedores do projeto, aplicação de oficinas de Lean Inception, levantamento bibliográfico, definição de frameworks e gateways de pagamento, refinamento do escopo, elicitação de requisitos, desenvolvimento de código, documentação do projeto e revisão da monografia.

Foi realizada uma análise comparativa entre frameworks populares, sendo escolhido o Strapi como a melhor opção para o Agromart. Além disso, foram selecionados os gateways de pagamento PagSeguro, Mercado Pago e PayPal para integração com a aplicação.

Foram desenvolvidos os módulos de pagamento e de ajuda, além de serem atualizados os pacotes e removidos os componentes não utilizados. Através dessas atividades, foi possível aprimorar a experiência dos usuários e atender às necessidades de gestão financeira das Comunidades que Sustentam a Agricultura (CSAs).

Assim, foi implementado um módulo integrador de pagamento digital, que se vincula a diferentes tipos de gateways permitindo a geração de uma URL de pagamento por meio de uma API. E gerenciador de extratos financeiros que possibilita a visualização dos registros financeiros, auxiliando na análise das transações financeiras.

Sendo assim, o objetivo principal deste trabalho foi atingindo, ao realizar a manutenção evolutiva do produto de software Agromart, por meio da atualização dos módulos e do aprimoramento das ferramentas de gestão financeira. O software Agromart foi atualizado e aprimorado, proporcionando uma melhor interação entre produtores e co-agricultores, facilitando o planejamento do escoamento da produção e oferecendo ferramentas de gestão financeira. Com a implementação do integrador de pagamento digital, as CSAs podem realizar transações seguras e eficientes, contribuindo para o desenvolvimento sustentável da agricultura familiar no Brasil.

O trabalho também contribuiu para responder à questão de pesquisa, que envolvia

a busca por soluções que melhorassem a gestão financeira das CSAs e a comunicação entre produtores e co-agricultores. Através da atualização e aprimoramento do Agromart, foi possível atender a essas necessidades e proporcionar uma melhor experiência aos usuários.

Apesar dos avanços alcançados com a atualização e aprimoramento do software Agromart, ainda existem oportunidades de melhoria para garantir a robustez e a confiabilidade do sistema. Uma das áreas que podem ser aprimoradas é a realização de testes não funcionais, como testes de desempenho, segurança e escalabilidade. A implementação desses testes permitiria avaliar o comportamento do sistema sob diferentes condições de carga, garantindo a estabilidade e o bom desempenho mesmo em situações de alta demanda.

Outra limitação identificada no software está relacionada aos gateways de pagamento padrão vinculados a um pacote específico da API. Essa dependência pode trazer desafios em caso de necessidade de alteração do comportamento dos gateways ou de adoção de novos provedores de pagamento. Nesses casos, a equipe de desenvolvimento precisaria realizar manutenções no código para garantir a compatibilidade com os novos serviços. Para mitigar esse problema, sugere-se buscar uma abordagem mais flexível e modularizada, permitindo a fácil substituição ou atualização dos gateways de pagamento sem grandes impactos no restante do sistema.

Além disso, é importante destacar que o desenvolvimento do Agromart foi realizado considerando as necessidades específicas das CSAs e sua interação com produtores e co-agricultores. No entanto, o software pode enfrentar desafios ao se adaptar a diferentes contextos e exigências de outras iniciativas agrícolas ou modelos de negócio. Portanto, é fundamental continuar aprimorando a flexibilidade do sistema, tornando-o mais adaptável e personalizável para atender a uma ampla gama de cenários agrícolas.

Dessa forma, ao continuar a evolução do software Agromart, é essencial considerar essas oportunidades de melhoria, abordando testes não funcionais para garantir a qualidade do sistema em diferentes aspectos e buscando uma arquitetura mais modular e flexível para lidar com possíveis alterações nas dependências do software. Essas medidas garantirão que o Agromart permaneça como uma ferramenta eficiente e confiável para o apoio as CSAs.

Referências

ALLIANCE, A. *Agile Practice Guide, Project Management Institute, 2017: Agile Practice Guide*. [S.l.]: Bukupedia, 2017. v. 1. Citado 4 vezes nas páginas 31, 32, 48 e 49.

CAROLI, P. *Lean Inception: Como alinhar pessoas e construir o produto certo*. [S.l.]: Editora Caroli, 2018. Citado 4 vezes nas páginas 29, 30, 37 e 52.

COPELAND, L. *A practitioner's guide to software test design*. [S.l.]: Artech House, 2004. Citado na página 34.

CORRêA, B. K. B.; VELUDO, I. G. Uma evolução do projeto agromart: open source, meios de pagamento e gestão de co-agricultores. 2022. Citado 3 vezes nas páginas 26, 52 e 59.

DEVELOPER, P. *Get started with PayPal Developer*. 2022. Disponível em: <https://developer.paypal.com/api/rest/>. Citado na página 38.

DEVELOPERS, M. P. *SAIBA COMO INTEGRAR O MERCADO PAGO AO SEU SITE*. 2023. Disponível em: <https://www.mercadopago.com.br/developers/pt>. Citado na página 39.

DISCORD. *Discord*. 2023. Disponível em: <https://discord.com/>. Citado na página 37.

DOCKER. *Develop faster. Run anywhere.* 2023. Disponível em: <https://www.docker.com/>. Citado na página 39.

DOCKER. *Overview*. 2023. Disponível em: <https://docs.docker.com/get-started/>. Citado na página 39.

DOCKER. *Overview*. 2023. Disponível em: <https://docs.docker.com/compose/>. Citado na página 39.

DOCUSAURUS. *Build optimized websites quickly, focus on your content*. 2023. Disponível em: <https://docusaurus.io/>. Citado na página 40.

EDEN, B. L. *Content management systems in libraries: case studies*. [S.l.]: Scarecrow Press, 2008. Citado 2 vezes nas páginas 33 e 34.

EXPO. *What is Expo?* 2023. Disponível em: <https://docs.expo.dev/introduction/expo/>. Citado na página 38.

FREITAS, A. A.-A. d.; CELLA, P. V. d. S. Uma evolução do projeto agromart: implantação individualizada e automatizada de um ambiente de csa. 2023. Citado 5 vezes nas páginas 26, 38, 52, 59 e 60.

GERHARDT, T. E.; SILVEIRA, D. T. *Métodos de pesquisa*. [S.l.]: Plageder, 2009. Citado na página 43.

GIL, A. C. et al. *Como elaborar projetos de pesquisa*. [S.l.]: Atlas São Paulo, 2002. v. 4. Citado na página 43.

GITHUB. *Let's build from here*. 2022. Disponível em: <https://github.com/about>. Citado na página 37.

GITHUB. *Framework*. 2023. Disponível em: <https://github.com/topics/framework>. Citado na página 54.

GITHUB. *Sobre Projects*. 2023. Disponível em: <https://docs.github.com/pt/issues/planning-and-tracking-with-projects/learning-about-projects/about-projects>. Citado na página 37.

GITHUB. *Websites for you and your projects*. 2023. Disponível em: <https://pages.github.com/>. Citado na página 40.

GOMZIN, S. *Hacking Point of Sale: Payment Application Secrets, Threats, and Solutions*. [S.l.]: John Wiley & Sons, 2014. Citado na página 34.

IBGE. *Censo Agropecuário 2017 Resultados Definitivos - Agricultura Familiar*. 2017. Disponível em: <https://censoagro2017.ibge.gov.br/templates/censo_agro/resultadosagro/pdf/agricultura_familiar.pdf>. Citado na página 25.

IEEE Std 1219-2018: IStandard for Software Maintenance. 1998. IEEE Standards Association. Disponível em: <https://standards.ieee.org/ieee/1219/1842/>. Citado na página 32.

JAVASCRIPT. *An Introduction to JavaScript*. 2022. Disponível em: <https://javascript.info/intro>. Citado na página 60.

MCWHERTER, J.; GOWELL, S. *Professional mobile application development*. [S.l.]: John Wiley & Sons, 2012. Citado na página 33.

MURAL. *Extraordinary teamwork made surprisingly simple*. 2022. Disponível em: <https://www.mural.co/why-mural>. Citado na página 37.

NATIVE, R. *React Native*. 2023. Disponível em: <https://reactnative.dev/>. Citado na página 38.

OWASP. *A06:2021 – Vulnerable and Outdated Components*. 2021. Disponível em: <https://owasp.org/Top10/A06_2021-Vulnerable_and_Outdated_Components/>. Citado na página 55.

PAGSEGURO. *Desenvolva e amplie seu negócio com o PagBank PagSeguro*. 2023. Disponível em: <https://dev.pagseguro.uol.com.br/>. Citado na página 39.

PLANTUML. *Frequently Asked Questions (FAQ)*. 2023. Disponível em: <https://plantuml.com/faq>. Citado na página 40.

POSTGRESQL. *New to PostgreSQL?* 2023. Disponível em: <https://www.postgresql.org/>. Citado na página 39.

RAJLICH, V. *Software engineering: The current practice*. [S.l.]: CRC Press, 2011. Citado na página 29.

RODRIGUES, L. S.; MACÊDO, L. P. d. A. Inovações tecnológicas na agricultura familiar: Agromart. 2021. Citado 6 vezes nas páginas 25, 26, 39, 51, 52 e 59.

SCHNEIDER, S. Teoria social, agricultura familiar e pluriatividade. *Revista brasileira de ciências sociais*, SciELO Brasil, v. 18, p. 99–122, 2003. Citado na página 25.

SNYK. *strapi vulnerabilities*. 2022. Disponível em: <https://security.snyk.io/package/npm/strapi>. Citado 2 vezes nas páginas 53 e 54.

STRAPI. *Manage Any Content. Anywhere.* 2023. Disponível em: <https://strapi.io/>. Citado na página 38.

TECH, W. *Technologies Overview*. 2023. Disponível em: <https://w3techs.com/technologies>. Citado na página 53.

TECH, W. *Web Technologies of the Year 2022*. 2023. Disponível em: <https://w3techs.com/blog/entry/web_technologies_of_the_year_2022>. Citado na página 54.

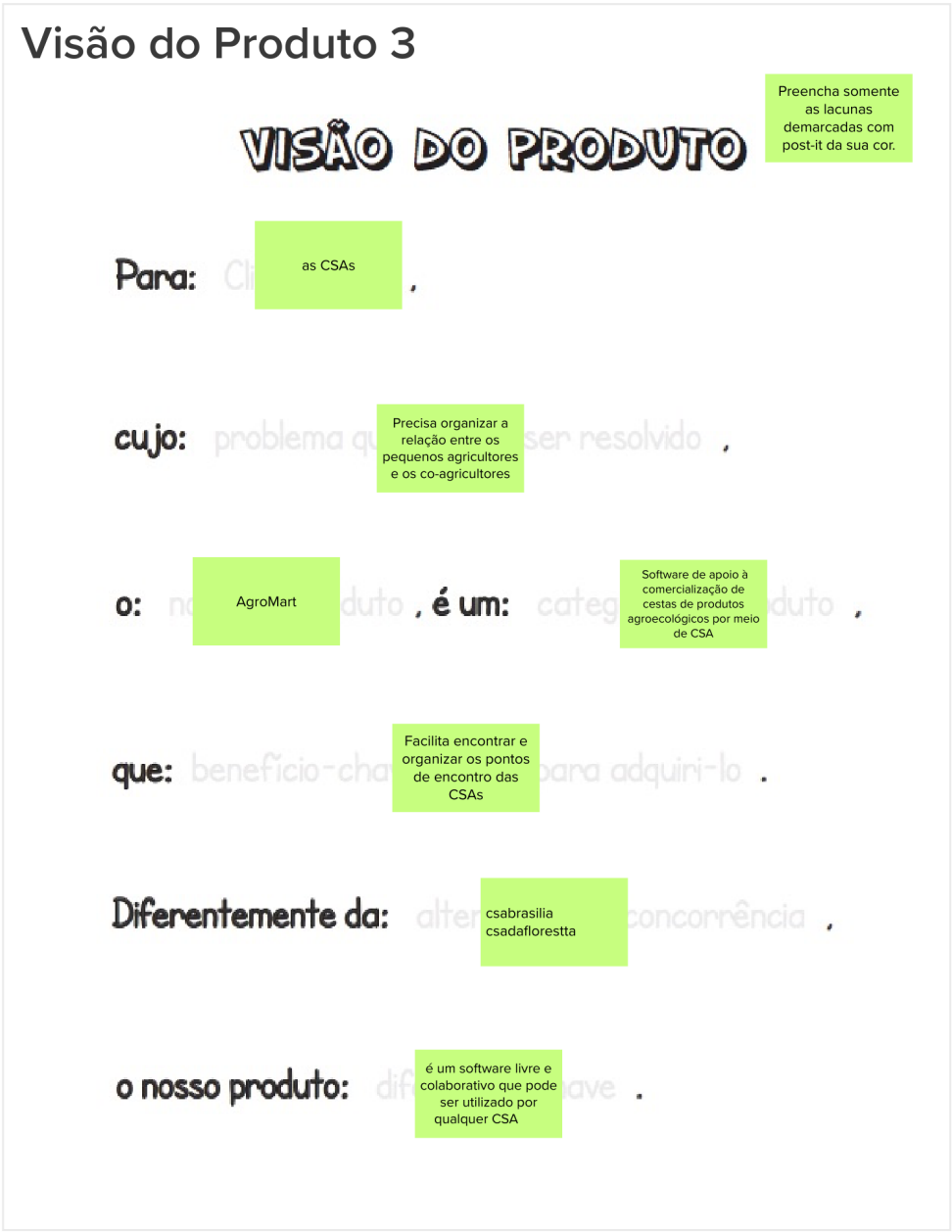
TORRES, C. L. Comunidade que sustenta a agricultura: a reaplicação da tecnologia social a partir dos casos pioneiros em brasília. 2017. Citado na página 25.

TORUNSKY, F.; NETO, D. N. F.; AMORIM, J. O. de L. Csa: Comunidade que sustenta agricultura, uma experiência em são carlos. *Cadernos de Agroecologia*, v. 10, n. 3, 2015. Citado na página 25.

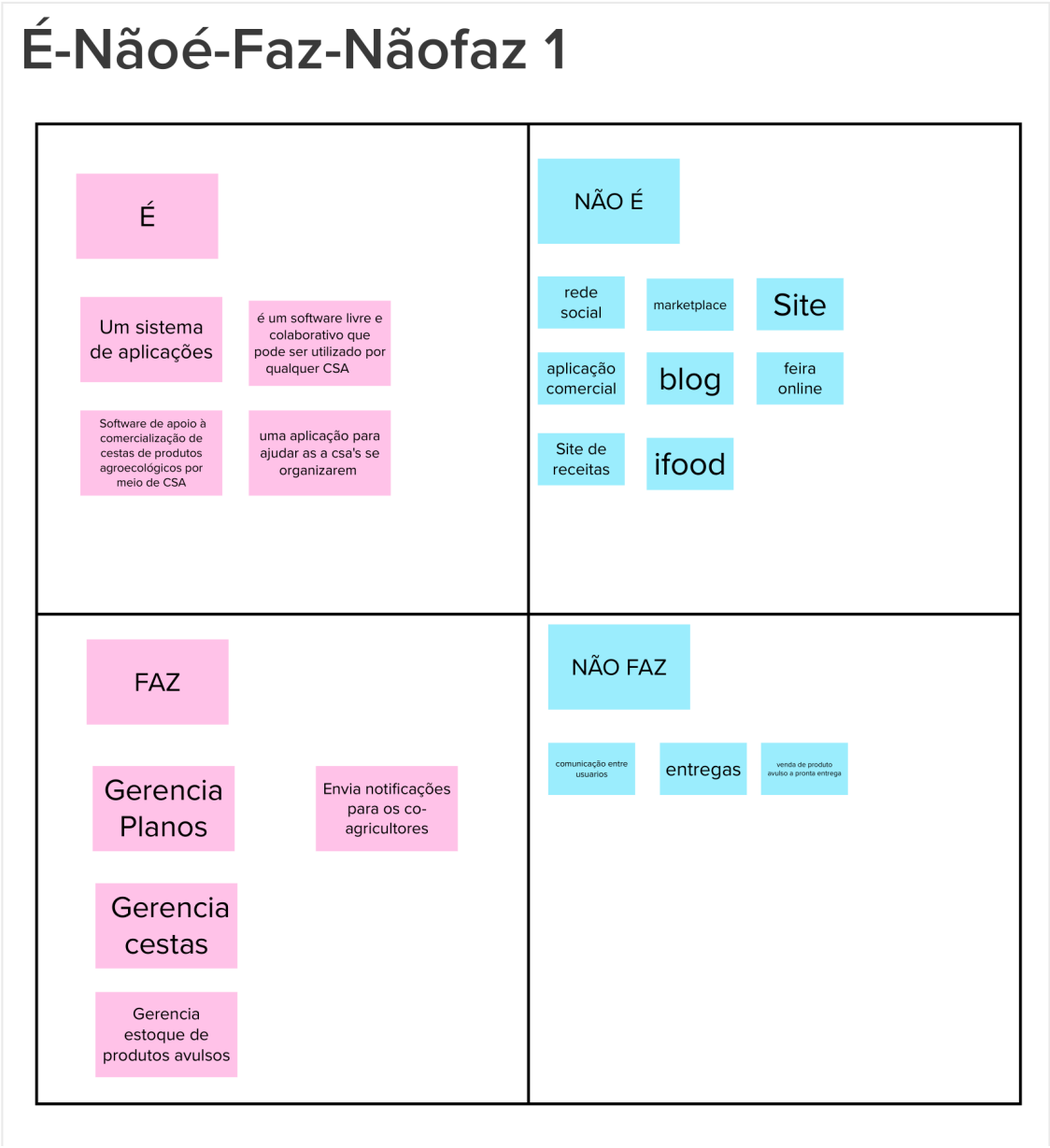
Apêndices

APÊNDICE A – Lean Inception: Design das adições no produto de software

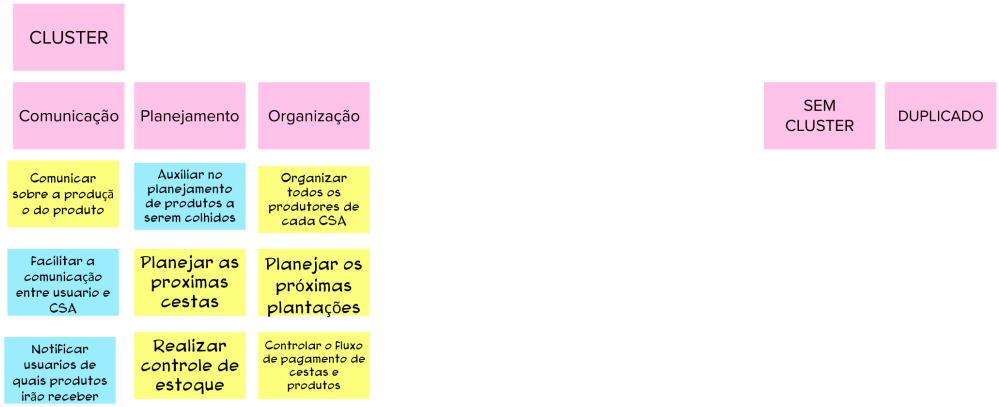
Visão do Produto



Aplicação da técnica "É não é"



Elaboração de objetivos de negócio



Personas

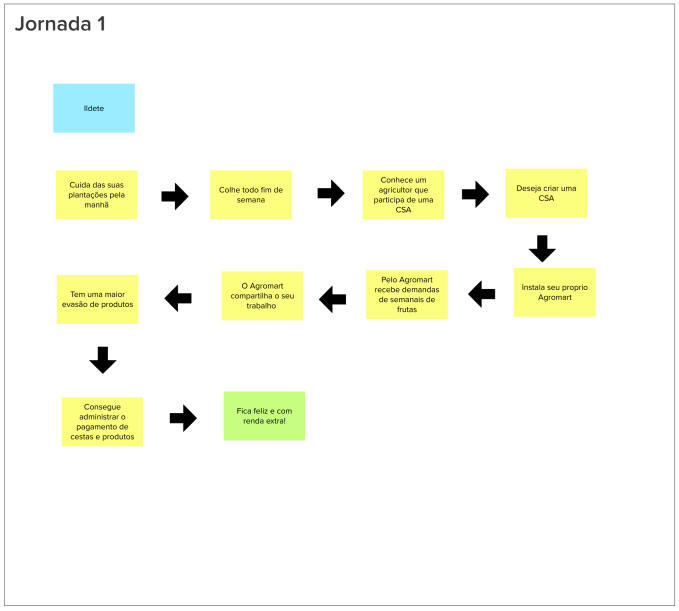
Persona 1

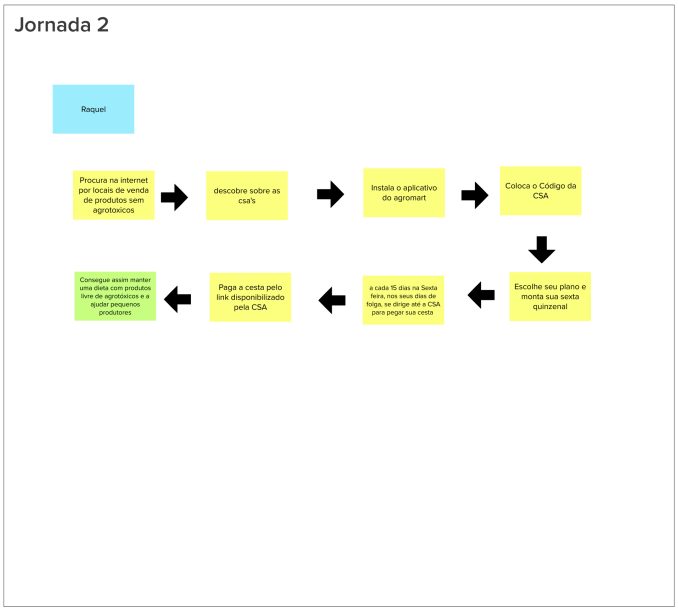
Ildete 	Perfil - 70 anos - Viúva - Tem uma fazenda familiar
Comportamento - Planta frutas na fazenda - Passa a maioria do tempo na fazenda - Da parte da produção para a família	Necessidades - Vender o restante de produtos - Organizar a próxima plantação - Encontrar pessoas para vender seus produtos - Espalhar a importância de seu trabalho - Gerenciar o pagamento de cestas e produtos

Persona 2

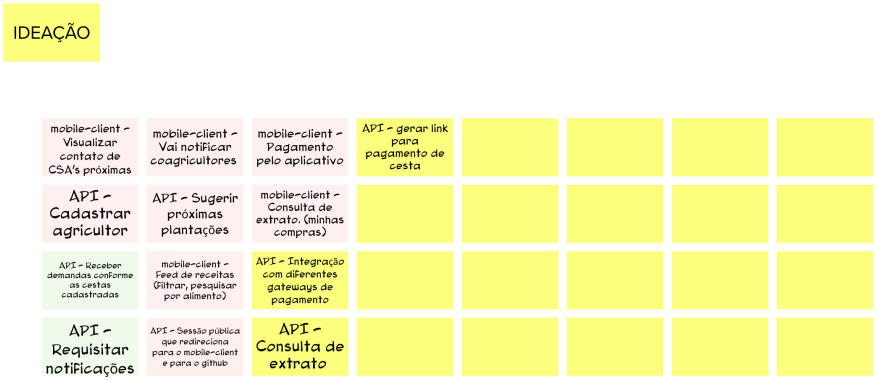
Raquel andrade 	Perfil - 29 anos - Solteira - Medica - tem 3 gatos - vegetariana
Comportamento Fitness Gosta de comer comida saudável gosta de fazer trilhas nos fins de semana faz compras 2x no mês	Necessidades - Comodidade para encontrar produtos sem agrotóxicos - Produtos frescos - Variedade de frutas e verduras em sua dieta

Jornada das personas

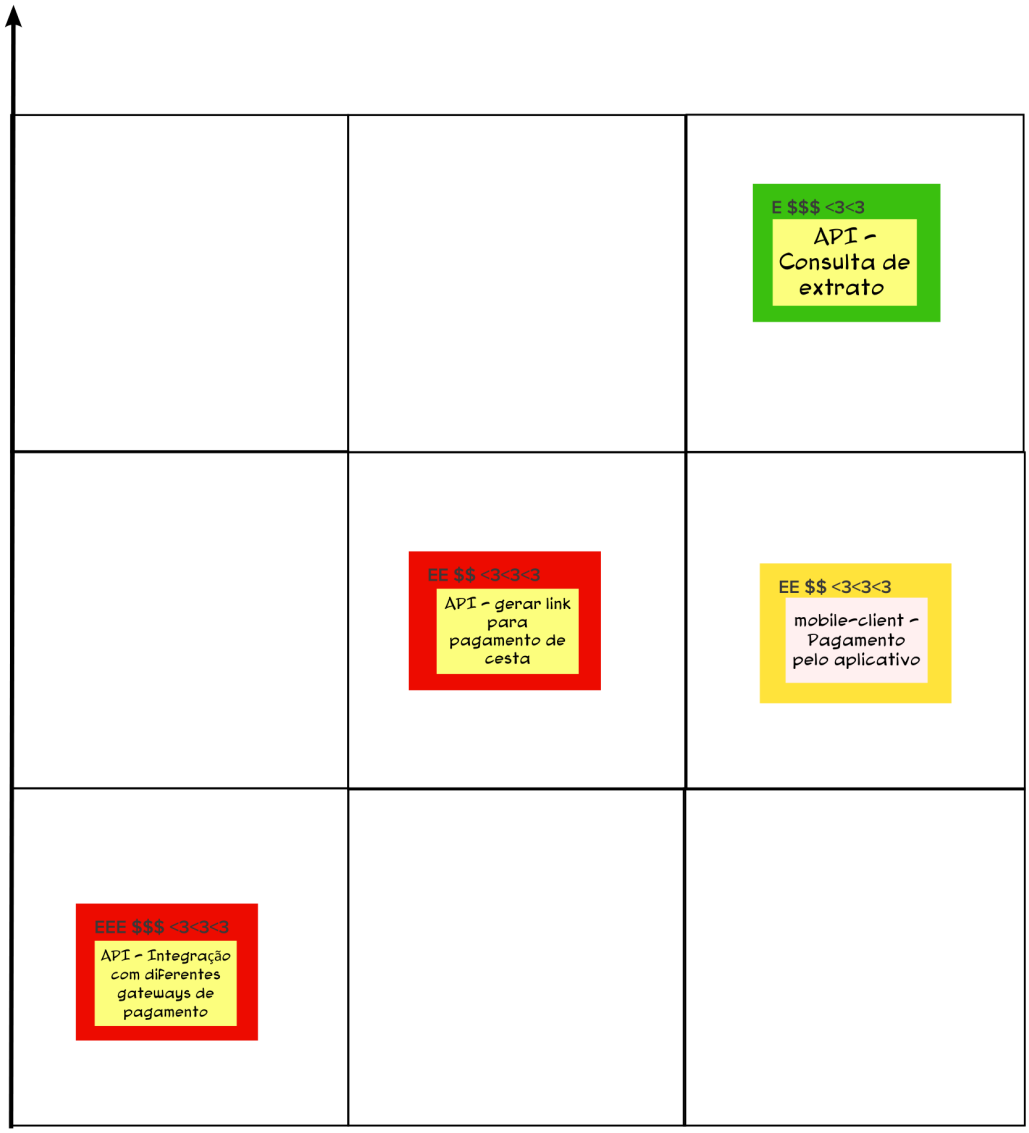




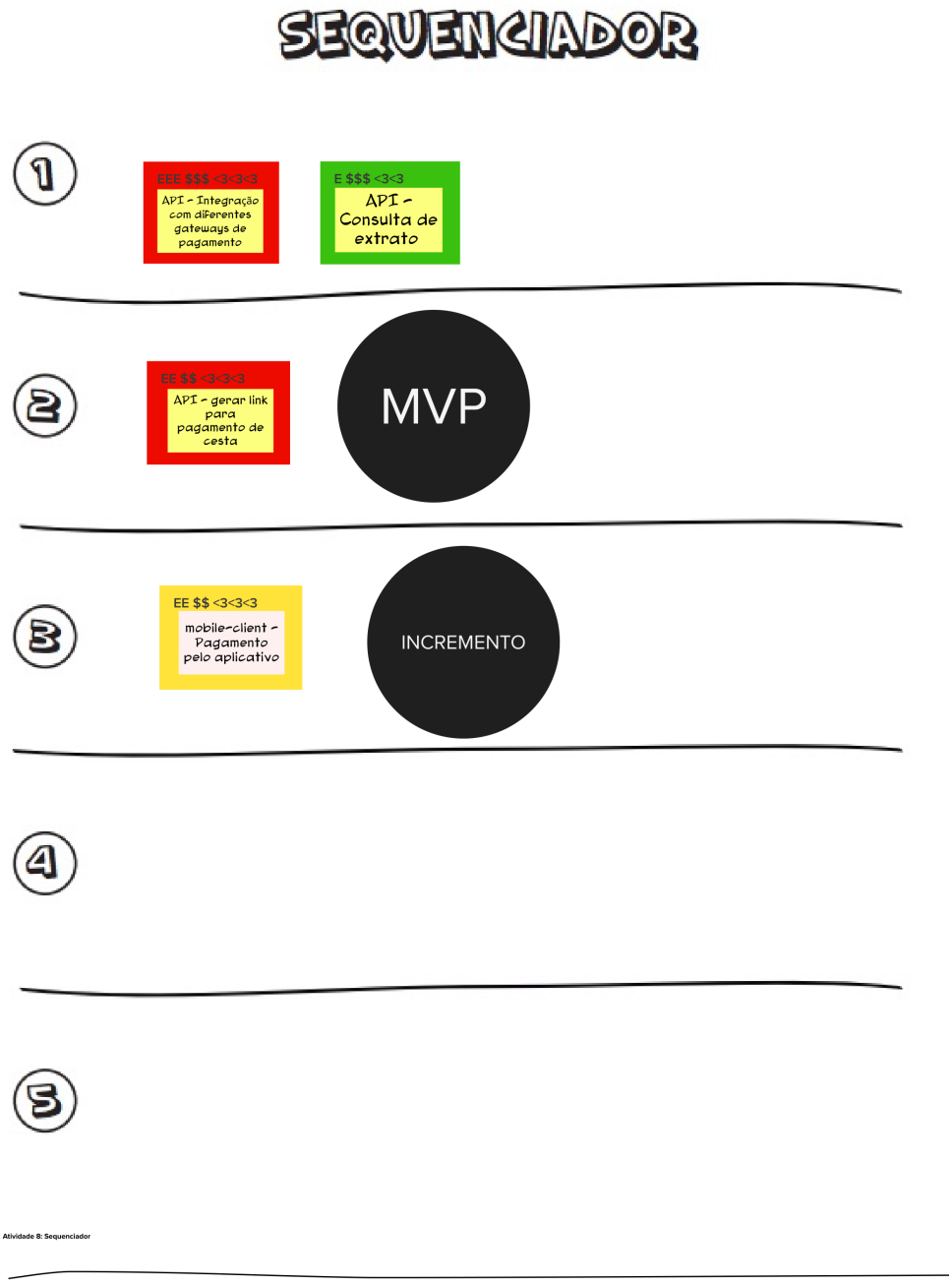
Brainstorm de funcionalidades



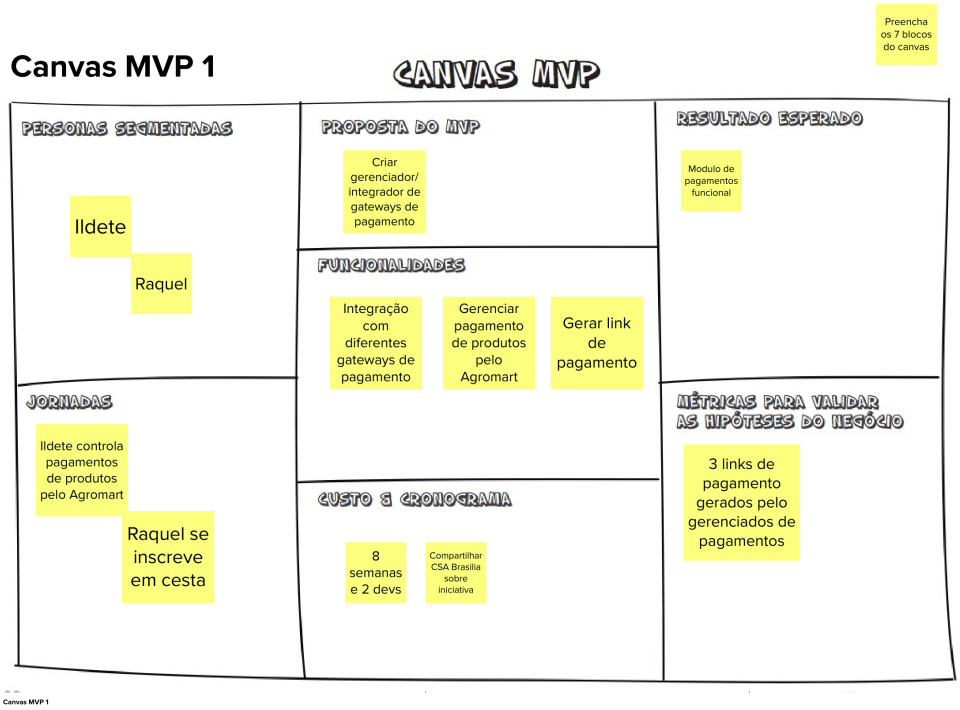
Revisão técnica



Sequenciador



Canva MVP


Canvas MVP 1
CANVAS MVP
Preencha os 7 blocos do canvas
PERSONAS SEGMENTADAS
Ildete
Raquel
PROPOSTA DO MVP
Criar gerenciador/ integrador de gateways de pagamento
RESULTADO ESPERADO
Modulo de pagamentos funcional
FUNCIONALIDADES
Integração com diferentes gateways de pagamento
Gerenciar pagamento de produtos pelo Agromart
Gerar link de pagamento
JORNADAS
Ildete controla pagamentos de produtos pelo Agromart
Raquel se inscreve em cesta
CUSTO E CRONOGRAMA
8 semanas e 2 devs
Compartilhar CSA Brasília sobre iniciativa
MÉTRICAS PARA VALIDAR AS HIPÓTESES DO NEGÓCIO
3 links de pagamento gerados pelo gerenciados de pagamentos
Canvas MVP 1