Pedro Luís Ferronato Barreto

Desenvolvimento de sistema de gerência de movimentação financeira em pequenas propriedades rurais

Vilhena - RO

2021

Instituto Federal de Educação, Ciência e Tecnologia de Rondônia – IFRO

Campus Vilhena

Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas

Pedro Luís Ferronato Barreto

Desenvolvimento de sistema de gerência de movimentação financeira em pequenas propriedades rurais

Trabalho de Conclusão de Curso apresentado ao Instituto Federal de Educação, Ciência e Tecnologia de Rondônia – campus Vilhena, realizado em cumprimento de requisito parcial para a obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Orientador: Marco Antonio Augusto de Andrade

Vilhena - RO

2021

FICHA CATALOGRÁFICA

Biblioteca IFRO – Campus Vilhena

B273d

BARRETO, Pedro Luís Ferronato

Desenvolvimento de sistema de gerência de movimentação financeira em pequenas propriedades rurais / Pedro Luís Ferronato Barreto – Vilhena, Rondônia, 2021.

63f. ; il.

Orientador Prof. M.e. Marco Antonio Augusto de Andrade

Trabalho de Conclusão de Curso (Tecnólogo em Análise e Desenvolvimento de Sistemas) – Instituto Federal de Educação, Ciência e Tecnologia de Rondônia - IFRO

1. Desenvolvimento 2. Propriedade rural 3. Financeiro I. Instituto Federal de Educação, Ciência e Tecnologia de Rondônia – IFRO II. Título

005.1

Bibliotecária responsável Rosilene Maria do Couto Marques CRB 11/321

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE RONDÔNIA
Vilhena - Código INEP: 11107804
Rodovia BR 174, KM 3, CEP 76982-270, Vilhena (RO)
CNPJ: 10.817.343/0003-69 - Telefone: 69 2101-0703

ATA DE DEFESA DE TRABALHO DE CONCLUSÃO DE CURSO

Na data 13/12/2021 realizou-se a sessão pública de defesa do Trabalho de Conclusão de Curso intitul **Desenvolvimento de sistema de gerência demovimentação financeira em pequenas propriedades rurais** apres pelo aluno **Pedro Luis Ferronato Barreto (2019105053023-1)** do Curso **Superior de Tecnologia em Anális Desenvolvimento de Sistemas (Vilhena)**. Os trabalhos foram iniciados às **21:00** pelo Professor **Marco Antonio Au de Andrade** presidente da banca examinadora, constituída pelos seguintes membros:

- **Marco Antonio Augusto de Andrade** (Orientador)
- **Flavio de Almeida Andrade Lico** (Examinador Interno)
- **Juliano Fischer Naves** (Examinador Interno)

A banca examinadora, tendo terminado a apresentação do conteúdo do Trabalho de Conclusão de Cu passou à arguição do candidato. Em seguida, os examinadores reuniram-se para avaliação e deram o parecer final so trabalho apresentado pelo aluno, tendo sido atribuído o seguinte resultado:

[X] APROVADO **Nota: 92**

Proclamados os resultados pelo presidente da banca examinadora, foram encerrados os trabalhos e, constar, eu **Marco Antonio Augusto de Andrade** lavrei a presente ata que assino juntamente com os demais memb banca examinadora.

VILHENA / RO, 13/1

Documento assinado eletronicamente por **Pedro Luis Ferronato Barreto**, Discente, em 13/12/2021, às 21:36, conforme horário ofi Rondônia, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Marco Antonio Augusto de Andrade**, Orientador, em 13/12/2021, às 21:33, conforme h oficial de Rondônia, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Flavio de Almeida Andrade Lico**, Examinador Interno, em 13/12/2021, às 21:28, confor oficial de Rondônia, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Dedico este trabalho à minha família, por todo apoio e dedicação, essenciais para o sucesso deste.

Agradecimentos

Agradeço todo o corpo docente do curso Análise e Desenvolvimento de Sistemas, pelo conhecimento transmitido e dedicação à turma. Agradeço à minha familha por todo suporte em meu desenvolvimento, carinho, amor e fonte de modelo e inspiração. Agradeço a todos meus amigos e colegas, em especial Bruno Delani, João Lopes e Gabriel Brum, pelo apoio e experiências compartilhadas. Agradeço à banca examinadora pelo tempo dedicado ao julgamento deste trabalho. Especialmente, agradeço Marco Antonio Augusto de Andrade, meu professor orientador, por todo direcionamento e ajuda durante o desenvolvimento do projeto.

Quando aceitamos nossos limites,
conseguimos ir além deles

Albert Einstein

Resumo

Muitas vezes encontrar um sistema simples para gerenciar o financeiro de uma pequena propriedade rural pode ser uma tarefa difícil para o produtor, muitos sistemas existem porém são direcionados aos grandes produtores, latifundiários, exportadores de produtos, sendo extensivos e alguns até necessitando de equipe especializada e treinada para utilizá-lo. Este trabalho, por sua vez, busca desenvolver uma solução simples, para o pequeno produtor rural, um sistema cotidiano para registro de compras, vendas, produções realizadas, a fim de tornar simples visualizar movimentações financeiras e estado de caixa e estoque, trazendo a capacidade de tomada de decisão fundamentada para o usuário. Para tal, ficam definidas como tecnologias e metodologias utilizadas: UML (*Unified Modeling Language* - Linguagem de Modelagem Unificada) para modelagem do sistema, o sistema *Web Figma* para prototipação das telas, Flask, *framework Python* para desenvolvimento *Full Stack*, gerenciando o processo de desenvolvimento com *Kanban*. Tendo como resultado um sistema simples de registro e consulta das movimentações financeiras de pequenas propriedades rurais, bem como recursos estocados. Ademais o registro de desenvolvimento deste encontra-se descrito no presente trabalho. O produto gerado traz para o usuário, pequenos produtores, capacidade de decisão, baseando-se nos dados inseridos no sistema tratados em informações, trazendo noção da real situação de sua propriedade.

Palavras-chave: Pequenas propriedades. Sistema. Financeiro. *Kanban*.

Abstract

Often times finding a simple system to manage the financial of a small rural property can be a difficult task for the producer, many systems exist but are aimed at large producers, latifundium owners, product exporters, being extensive and some even requiring specialized and trained staff to use it. This work, by softness, seeks to develop a simple solution, for the small rural producer, a daily system for recording purchases, sales, productions carried out, in order to make it simple to visualize financial movements and cash flow and stock, bringing the capacity to making an informed decision for the user. For this, the technologies and methodologies used are defined as: UML (Unified Modeling Language) for system modeling, Web Figma system for screen prototyping, Flask, Python framework for Full Stack development, managing the development process with Kanban. The result is a simple system for recording and consulting the financial transactions of small rural properties, as well as stored resources. Furthermore, its development record is described in this work. The generated product brings to the user, small producers, decision capacity, based on the data entered in the system treated in information, bringing a notion of the real situation of their property.

Keywords: Small rural property. System. Finance. Kanban.

Lista de ilustrações

Figura 1 – Ciclo de vida 29
Figura 2 – Etapa de preparação 30
Figura 3 – Etapa de desenvolvimento 31
Figura 4 – Etapa de finalização 32
Figura 5 – Arquitetura do software 35
Figura 6 – Arquitetura da aplicação 35
Figura 7 – Diagrama de classe 37
Figura 8 – Diagrama de sequência 38
Figura 9 – Diagrama de entidade e relacionamento 39
Figura 10 – Gráfico de *throughput* do processo 44
Figura 11 – Gráfico de *cycle time* coluna "Novo"do processo 45
Figura 12 – Gráfico de *cycle time* coluna "Em andamento"do processo 45
Figura 13 – Gráfico de *cycle time* coluna "Análise"do processo 46
Figura 14 – Gráfico de *lead time* do processo 47
Figura 15 – Código base para cartões de redirecionamento 48
Figura 16 – Código base para campos de formulário 48
Figura 17 – Exemplo de código de teste 49
Figura 18 – Relatório de testes 50
Figura 19 – Cobertura dos testes 51
Figura 20 – Página de listagem de produções ativas 52
Figura 21 – Fragmento da página de detalhes de uma determinada produção 53
Figura 22 – Fragmento da página de *dashboard* 54
Figura 23 – Fragmento da página de *dashboard* - Cartões inferiores 55
Figura 24 – Tela de registro de vendas 56

Lista de tabelas

Tabela 1 – Requisitos Funcionais..... 34

Lista de quadros

Quadro 1 – Plano de testes 40

Lista de abreviaturas e siglas

CRUD	Sigla para Create (criação), Read (consulta), Update (atualização) e Delete (exclusão) de dados
UML	Sigla para Unified Modeling Language - Linguagem de Modelagem Unificada
ORM	Sigla para Object-Relational Mapping - Mapeamento de Objeto-Relacional
SGBD	Sigla para Sistema de gerenciamento de banco de dados
DER	Sigla para Diagrama de entidade e relacionamento
RF	Requisitos Funcionais
MVC	Sigla para Model View Controller
MER	Sigla pra Modelo de entidade e relacionamento

Sumário

1 INTRODUÇÃO 23
1.1 Contexto e problema 23
1.2 Objetivos 23
1.2.1 Objetivo geral 23
1.2.2 Objetivos específicos 24
1.3 Justificativa 24

2 FUNDAMENTAÇÃO TEÓRICA 25
2.1 Trabalhos similares 27

3 MATERIAIS E MÉTODOS 29
3.1 Ciclo de vida e processos 29
3.2 Ferramentas e tecnologias utilizadas 32
3.3 Requisitos 33
3.4 Arquitetura do software 35
3.5 Modelagem 35
3.6 Persistência de dados 38
3.7 Plano de testes 39
3.8 Licença de uso 40

4 RESULTADOS E DISCUSSÕES 41
4.1 Gerenciamento de configuração e mudanças 41
4.2 Processo de desenvolvimento 43
4.3 Relatório dos testes 49
4.4 Implantação 51
4.5 Demonstração do software 52

5 CONSIDERAÇÕES FINAIS 57
5.1 Trabalhos futuros 57

Referências 59

ANEXOS 61

ANEXO A – LICENÇA MIT 63

1 Introdução

1.1 Contexto e problema

Encontrar um sistema informatizado, direcionado ao público rural e principalmente sendo um pequeno produtor, pode ser uma tarefa difícil, aplicações desenvolvidas para gerenciamento de propriedades rurais tem como público alvo grandes produtores ou então, são produtos de valores elevados.

Ademais, podem ser demasiadamente abrangentes, mesmo os sistemas modulares, onde você pode adquirir pequenas parcelas do sistema completo, atingem múltiplas áreas, como por exemplo, estoque completo, variabilidade de valores no mercado, pluviometria e muitas mais funcionalidades. Podendo tornar complexo a sistematização de processos simples dos pequenos produtores.

Porém, pequenos produtores não dispõem de equipe de apoio especializada para utilização do sistema, têm necessidades mais simples como verificação simples e principalmente, poder financeiro para adquirir os grandes sistemas.

A área da tecnologia da informação vem cada vez mais atingindo diferentes esferas, uma delas, a agricultura, a Embrapa pode ser destacada como exemplo de organização que busca o desenvolvimento de sistemas para esta área, como exposto por APOLINÁRIO et al. (2017) é notória a necessidade de aprimorar seus processos de trabalho, em especial o de desenvolvimento de software.

Sendo possível destacar também a importância de um processo definido para desenvolvimento dos sistemas, organizações que estendem suas atuações para informática a fim de promover a maior eficiência, definem a necessidade de estruturação dos projetos com base nas relações, desde culturais, passando por políticas, chegando à esfera organizacional (APOLINÁRIO et al., 2017).

Sendo assim este projeto tem como objetivo produzir uma solução de código aberto, gratuita e destinada ao auxílio financeiro do pequeno produtor rural, buscando trazer a fácil visualização das informações relevantes promovendo a capacidade de tomada de decisão.

1.2 Objetivos

1.2.1 Objetivo geral

Levando em consideração a situação apresentada anteriormente, define-se como objetivo geral, desenvolver um sistema capaz de registrar dados referentes às movimentações financeiras,

estoque e produção de uma determinada propriedade, a fim de auxiliar o pequeno produtor rural em suas decisões no que se refere a suas posses.

1.2.2 Objetivos específicos

A fim de alcançar o objetivo geral de desenvolvimento da plataforma foram elencados os seguintes objetivos específicos:

- Analisar os padrões de registro financeiro de uma propriedade;
- Elaborar diagrama de classe;
- Elaborar diagrama de Entidade e Relacionamento;
- Desenvolver protótipos das telas do sistema;
- Validar protótipo com público alvo;

1.3 Justificativa

Sistemas informatizados podem promover maior eficiência nos processos em que são introduzidos, portanto este trabalho busca expor e analisar o processo de desenvolvimento de um produto, destinado ao público de pequenos produtores rurais, produzindo assim um caso de exemplo de utilização das tecnologias e possivelmente um sistema útil para o usuário final.

2 Fundamentação teórica

Associar informática à agronomia e agropecuária é uma tarefa e intenção demonstrada pela comunidade há algumas décadas, principalmente grandes produtores e Embrapa, MEIRA et al. (1996) A informática poderá auxiliar para facilitar a gerência dos novos sistemas produtivos que surgirão e para agilizar o processo decisório, permitindo um melhor planejamento das atividades agropecuárias, em busca da otimização da aplicação dos conceitos embutidos nesses sistemas.

Porém, em seu princípio, não atingia diretamente as produções, MEIRA et al. (1996) A tecnologia da informação começou a ser aplicada com sucesso nas fazendas com a automatização das tarefas de contabilidade, de controle de recursos humanos e de controle de estoques e de maquinário. Só anos depois os agricultores e criadores puderam utilizar a informática diretamente na produção.

Ademais, alguns sistemas apresentavam alta complexidade na utilização, com o exposto por BECKER (2002) a ferramenta não pode vir antes da administração. É preciso primeiro profissionalizar o gerenciamento, depois instalar um sistema de informatização adequado para suas necessidades.

No entanto, faz-se notável nos dias hoje a potencialização a informatização do campo, movimento proporcionado pelas inovações desenvolvidas pela comunidade de tecnologia, como *drones* capazes de interagir com a produção SILVA; BOTELHO (2017), sistemas de gerência de propriedades e produções, não podendo excluir também o trabalho desenvolvido por órgãos públicos para levar essas inovações ao campo.

O desenvolvimento não detém-se somente aos latifundiários, mas também ao produtor familiar, evento observável por exemplo nos programas do SENAR (Serviço Nacional de Aprendizagem Rural), onde técnicos atuam junto ao produtor a fim de promover a evolução na propriedade, porém, não sendo completamente abrangente no quesito de não entregar o sistema utilizado para registros nas mãos do produtor tendo como ponte um dos técnicos do serviço, situação observada em visitas a propriedade exemplo, aliada a entrevistas.

A produção rural advinda dos pequenos produtores é essencial para o desenvolvimento e manutenção da sociedade, afinal, são estes que produzem grande parte do alimento que levamos às nossas mesas, legumes, vegetais, laticínios e demais produtos de origem animal e vegetal, nos mercados e feiras os produtos dispostos geralmente são provenientes das propriedades e cooperativas que circundam as cidades, ficando então os latifúndios destinados à produção em larga escala, principalmente de grãos, para exportação aos latifundiários.

Sistemas informatizados são conhecidamente promotores de aprimoramento de eficiência

das situações em que são inseridos ou implantados, portanto, adicionar esse fator à equação dos pequenos produtores a fim de notificar e expor dados e informações referentes a uma produção para fundamentar e promover tomadas de decisão traria não somente aumentar a quantidade produzida em uma safra, reduzir valores em caixa para o usuário, fatores desejáveis, mas também trazer esses efeitos benéficos para a sociedade que os rodeia.

Registro das produções, das atividades ligadas à mesma, podendo elencar cadastro de recursos e processos sanitários e zootécnicos se mostraram fonte de benefícios para produtores rurais, no que se refere a controle de custos e embasamento na tomada de decisões, segundo (CHIOZINI; CARVALHO, 2017).

A evolução não ficou restrita ao quesito de inserção de sistemas informatizados nas propriedades, os próprios produtos da área de tecnologia da informação aprimoraram, capacidade, velocidade, eficiência energética, alcance de transmissão, segurança dos sistema, metodologias de desenvolvimento foram criadas e aprimoradas para promover uma entrega constante e significativa das funcionalidades elencadas no levantamento dos requisitos.

Quanto às evoluções da infraestrutura de comunicações, tornou-se comum e viável a implantação de sistemas em localidades distantes dos centros urbanos, operadoras vêm instalando centros de transmissão e retransmissão cada vez mais distantes, ampliando a área de cobertura, começando a ser comum comunidades, cooperativas e propriedades fazerem parte das redes de telefonia, não podendo deixar de citar os sistemas de *internet* via rádio, que trazem não somente para cidades mas também para o campo a possibilidade de se conectar com a rede.

Sobre as metodologias de desenvolvimento, podemos citar exemplos como *Kanban* e *Scrum* que promovem métodos de gerenciamento de projetos, especificamente o primeiro citado define quadros de funcionalidades, divididos em colunas de estados, ou seja, o desenvolvedor ou equipe dispõem de um local especializados, extremamente gráfico, onde podem anotar tarefas e funcionalidades a serem desenvolvidas (*TO DO*), caso alguma entre em desenvolvimento ela é movida de coluna, (*DOING*) sinalizando seu novo estado, é comum serem utilizados nomes na língua inglesa, porém não é uma regra, bem como quantidade e função de cada coluna, ficando a critério dos desenvolvedores defini-los a fim de atender suas necessidades, prática comum no *Kanban* é a coleta de dados referentes aos tempo que cada tarefa passou em um estado, a fim de produzir métricas para determinação de fatores positivos e negativos do processo (SILVA; SANTOS; NETO, 2012).

Tendo em vista esse contexto, levanta-se a necessidade e possibilidade do desenvolvimento de um produto que atenda as necessidades de coleta e processamento dos dados das produções a fim de gerar informações fundamentadoras de decisões dos pequenos produtores rurais, ficando definido um sistema de tecnologia de informação como resultado esperado, que atinja o público e aproveite das evoluções de infraestrutura e metodológicas da área.

2.1 Trabalhos similares

Sistemas informatizados para gerência de propriedades são produtos disponibilizados no mercado, nem sempre atingindo todos os setores, mas sempre podendo servir de objeto de pesquisa e para desenvolvimento deste trabalho os seguintes *software* proprietários foram analisados e podem ser destacados:

- Aegro: Sistema proprietário, multiplataforma de planejamento financeiro de safra, com conexão com contas bancárias, Sefaz, NF-e, gerência de custos e estoque. Alta gama de abrangência na área financeira, com relatório extremamente detalhados, porém, valores devem ser tratados com consultores e a utilização pode demandar curva de aprendizagem alta.
- Agrisoft TI-Agro: Sistema agrícola, proprietário, baseado no sistema de módulos, sendo eles, rebanho, agrícola e de máquinas, apresentando profundidade na aplicação e suporte a manutenção ao banco de dados, porém apresentando visualização prévia do sistemas reduzida.
- CHBAGRO: Sistema *Web*, o primeiro no Brasil a abranger a plataforma, segundo os mesmos, extremamente modularizado, contando com quase duas dezenas deles e com conectividade *mobile*, podendo ser provável empecilho a grande quantidade de possibilidades algumas vezes próximas umas das outras.

Ademais, sistemas de código aberto, para gerência de propriedades rurais, também foram estudados para desenvolvimento:

- farmOS: Sistema *Web* capaz de associar imagens de satélite com demarcações de geolocalização para visualização gráfica das produções e registro de colheitas, gerenciamento de estoque, tendo como ponto forte as vantagens de agregação constante em decorrência do código aberto, porém baseado na língua inglesa, associada a interface minimalista pode não atender completamente o público brasileiro.
- Tania: Sistema *Web* de código aberto, destinado ao registro de pequenas produções, principalmente de mudas, atualizando necessidade de rega e *dashboard* para manutenção dos registros, com *design* agradável, porém em língua inglesa.
- LiteFarm: Sistema *Web* de código aberto, com comunidade ativa e *roadmap* bem definido, realiza registro detalhado das operações das produções e possibilita registro de vendas e despesas, bem como compartilhamento de certificados, com *design* agradável, porém em língua inglesa.

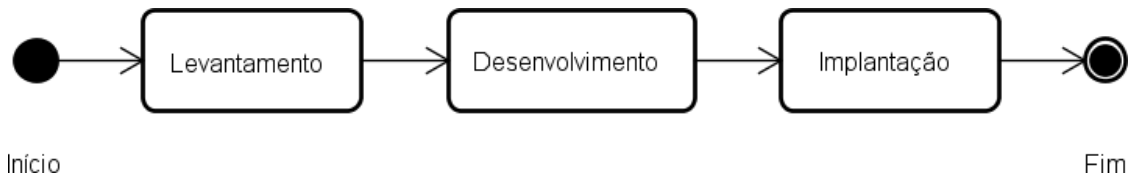
3 Materiais e métodos

Neste capítulo são descritos os materiais e métodos utilizados para o desenvolvimento da solução proposta.

3.1 Ciclo de vida e processos

Durante o planejamento inicial do desenvolvimento do sistema de gerência financeira para pequenos produtores definiu-se um ciclo de vida aproximado ao incremental, observar figura 1, onde o desenvolvimento busca se aproximar do usuário, separando funcionalidades, levantando requisitos e produzindo-a a cada ciclo. O diferencial definido para o projeto em questão foi o levantamento completo de requisitos inicialmente, validá-los novamente e caso necessário adaptá-los ao início de cada ciclo.

Figura 1 – Ciclo de vida



Fonte: elaborada pelo autor

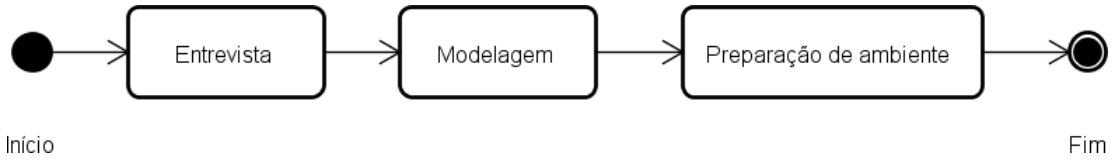
Portanto, a primeira etapa, figura 2, da produção se deu com o levantamento dos requisitos, visitas à propriedade exemplo e entrevistas apontaram as funcionalidades básicas e principalmente os dados necessários para registro dos objetos do sistema, sendo eles, produtor, sua propriedade, as produções realizadas nesta, compras, bem como os insumos e fornecedor e por fim, as vendas, relacionadas com produtos e clientes.

A propriedade exemplo escolhida para análise foi a chácara Trio parada dura, localizada a aproximadamente 10 km da zona urbana da cidade de Vilhena, no estado de Rondônia. A propriedade em questão, gerida por três associados, mantinha registros das notas fiscais de suas compras, tabelas simples em cadernos e em grande parte registros mentais das vendas. A partir das visitas e entrevistas foram gerados os gráficos de modelagem, alguns expostos a seguir.

Com os requisitos levantados, iniciou-se então a modelagem dos diagramas de classe e de entidade e relacionamento, produzindo protótipo de tela inicial, definindo identidade visual do sistema, a fim de basear a fase de desenvolvimento.

Em sequência da etapa inicial realizou-se a preparação do ambiente para desenvolvimento, construindo as pastas necessárias, ambiente virtualizado de desenvolvimento, baseado no *pipenv* para *Python*, instalando neste todas as bibliotecas necessárias: de *framework*, de ORM, testes automáticos, formulários e segurança. construindo então os banco de dados com a ORM *Flask-SQLAlchemy*[1].

Figura 2 – Etapa de preparação



Fonte: elaborada pelo autor

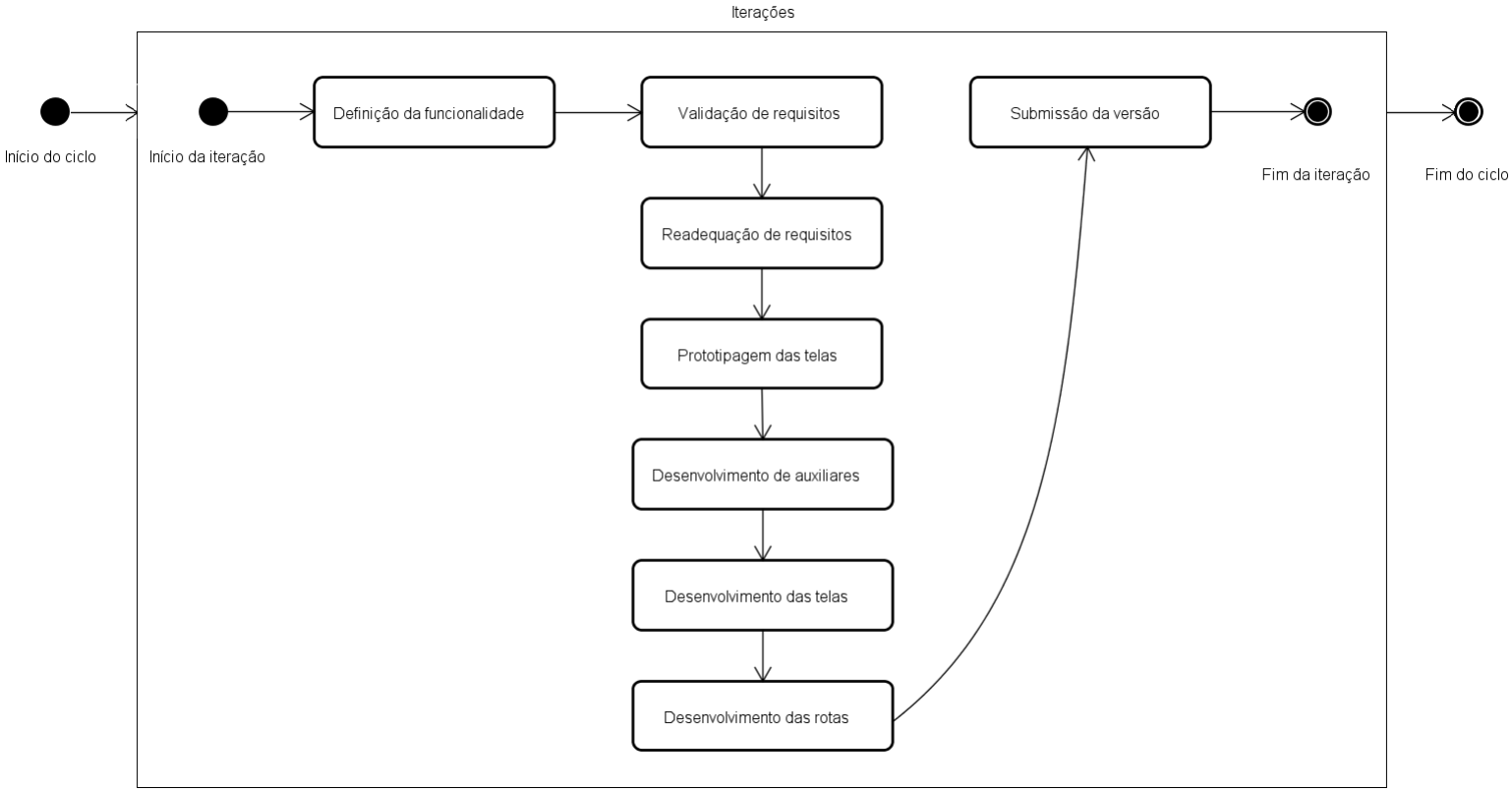
A etapa de desenvolvimento, figura 3, foi dividida em ciclos, onde funcionalidades levantadas anteriormente eram separadas, por exemplo, desenvolver a funcionalidade de registro de uma produção, iniciava-se então a revalidação dos dados definidos no levantamento, caso necessário sua readequação às necessidades reais da funcionalidade, principalmente caso tenha sido notada falta de informação ou novas partes da funcionalidades tenham sido notificadas.

Fundamentado, dava-se início o processos de prototipação da funcionalidade em questão, baseando-se na identidade visual anteriormente acordada e buscando uma boa experiência do usuário, quando necessário, arquivos de formulários, baseados nas bibliotecas *Flask-WTF*[2], eram desenvolvidos para auxiliar na construção das telas e rotas, já que dispunham da capacidade de validação e *render* de campos.

Por fim, realizava-se o desenvolvimento das rotas, com *Flask*[3], responsáveis em construir as telas utilizadas pelo usuário, bem como tratar dados enviados das mesmas e interagir, quando necessário, com a base de dados. Ao final de cada iteração o código era versionado na plataforma *GitHub*[4].

1 Disponível em https://flask-sqlalchemy.palletsprojects.com/en/2.x/
2 Disponível em https://flask-wtf.readthedocs.io/en/1.0.x/
3 Disponível em https://flask.palletsprojects.com/en/2.0.x/
4 Disponível em https://github.com/pedroferronato/gerenciamento-rural

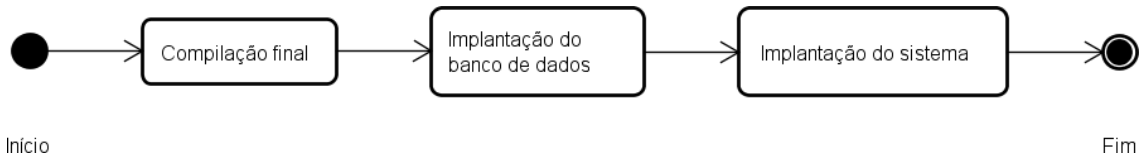
Figura 3 – Etapa de desenvolvimento



Fonte: elaborada pelo autor

Ao fim do desenvolvimento, iniciou-se a fase de implantação, figura 4, onde banco de dados e sistema foram compilados e implantados em instância virtual no servidor IFRO - *Campus* Vilhena, tornando possível o acesso dos usuário através do *link* público[5].

Figura 4 – Etapa de finalização



Fonte: elaborada pelo autor

3.2 Ferramentas e tecnologias utilizadas

A principal linguagem de programação utilizada no desenvolvimento foi Python[6], linguagem extremamente potente, com *frameworks* e comunidade ativa, auxiliando no desenvolvimento, códigos nesta linguagem buscam simplicidade e legibilidade, não deixando de lado a capacidade computacional de acesso e processamento de dados em informações, sua capacidade em tratamento de grande quantidade de dados também foi fator somatório na escolha, pois, trabalhos futuros com objetivos de gerar relatórios extensos dos dados gerados pelo sistema seriam processos simplificados pela utilização prévia da linguagem (MENEZES, 2010).

Flask é um dos *frameworks* para desenvolvimento *web* da linguagem definida, dispondo de funcionalidades de construção de rotas, *render* de *templates* e principalmente a facilidade de trabalho com dados das requisições e respostas, contando também com associação rápida a demais tecnologias, como *SQLAlchemy*[7], ORM para Python, contando também com a versão associada ao *framework*, *Flask-SQLAlchemy*[8], utilizada para CRUD dos objetos na base de dados.

O SGBD (Sistema de gerenciamento de banco de dados) escolhido para registro e manutenção dos dados foi o *MySQL*[9], devido ao alto nível de relacionamento entre os objetos

[5] Disponível em https://ruralfinance.fslab.dev/
[6] Disponível em https://www.python.org
[7] Disponível em https://www.sqlalchemy.org
[8] Disponível em https://flask-sqlalchemy.palletsprojects.com/en/2.x/
[9] Disponível em https://www.mysql.com

exigido pelo sistema, como supracitado, acessos, estruturação da base e operações de CRUD são realizadas pelo ORM *Flask-SQLAlchemy*.

Ademais, JavaScript, HTML e CSS, foram utilizados para estrutura, estilização e funcionalidades adicionais, como preenchimento automático dos campos referentes a valores monetários e máscaras de datas nas páginas construídas pelas rotas gerenciadas pelo *Flask*.

No processo de levantamento de requisitos, o sistema de modelagem da UML (Unified Modeling Language - Linguagem unificada de modelagem) escolhido foi o Astah[10], versão de comunidade, para prototipagem das páginas[11], utilizou-se o *software web* Figma, tendo como método de validação principal conferências com produtor de pequena propriedade, possível futuro usuário do sistema.

3.3 Requisitos

Baseando-se nas entrevistas com pequeno produtor e modelagens dos diagramas de classe e DER iniciais definiram-se os requisitos funcionais, que definem o que o sistema deve realizar e não funcionais, que ditam como as funcionalidades devem ser realizadas, por exemplo, velocidade dos processos e tecnologias utilizadas, necessários para atingir o objetivo de um sistema financeiro simples.

Seguem os requisitos funcionais definidos inicialmente:

[10] Disponível em https://astah.net

[11] Disponível em https://www.figma.com/file/OneCOkOUbCspVXGgdqgroe/Sistema-de-gerência-de-movimentação-financeira-em-pequenas-propriedades-rurais?node-id=0%3A1

Tabela 1 – Requisitos Funcionais

RF	Descrição
1	Construir modelos para registro dos produtores
2	Construir telas para registro dos produtores
3	Construir telas para manutenção dos produtores
4	Construir rotas para manutenção dos produtores
5	Construir modelos para registro das propriedades
6	Construir telas para registro das propriedades
7	Construir telas para manutenção das propriedades
8	Construir rotas para manutenção das propriedades
9	Construir modelos para registro das produções
10	Construir telas para registro das produções
11	Construir telas para manutenção das produções
12	Construir rotas para manutenção das produções
13	Construir modelos para registro das movimentações financeiras
14	Construir telas para registro das movimentações financeiras
15	Construir telas para manutenção das movimentações financeiras
16	Construir rotas para manutenção das movimentações financeiras
17	Construir modelos para registro dos fornecedores
18	Construir telas para registro dos fornecedores
19	Construir telas para manutenção dos fornecedores
20	Construir rotas para manutenção dos fornecedores
21	Construir modelos para registro dos clientes
22	Construir telas para registro dos clientes
23	Construir telas para manutenção dos clientes
24	Construir rotas para manutenção dos clientes
25	Construir telas para conferência do estoque
26	Construir rotas para manutenção do estoque
27	Construir telas para dashboard
28	Exibir cartões com movimentações do mês atual
29	Exibir cartões com últimas compras
30	Exibir cartões com maiores vendas do mês
31	Gerar valor gasto por unidade produzida

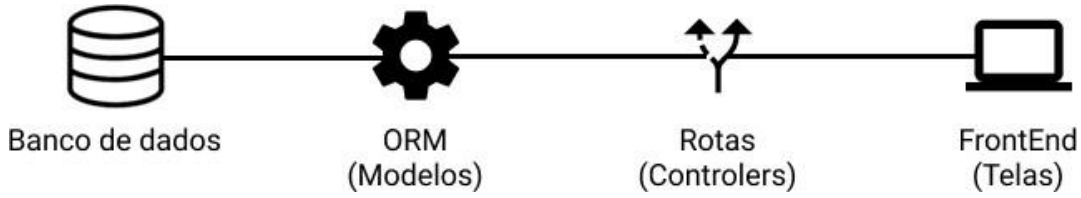
3.4 Arquitetura do software

O sistema desenvolvido foi uma aplicação monolítica, ou seja, acesso à base de dados e interfaces estão em uma mesma localidade, internamente o sistema dispõem de arquitetura MVC para promoção de suas funcionalidades, observar exemplos nas figuras 5 e 6.

Figura 5 – Arquitetura do software

Fonte: elaborada pelo autor

Figura 6 – Arquitetura da aplicação



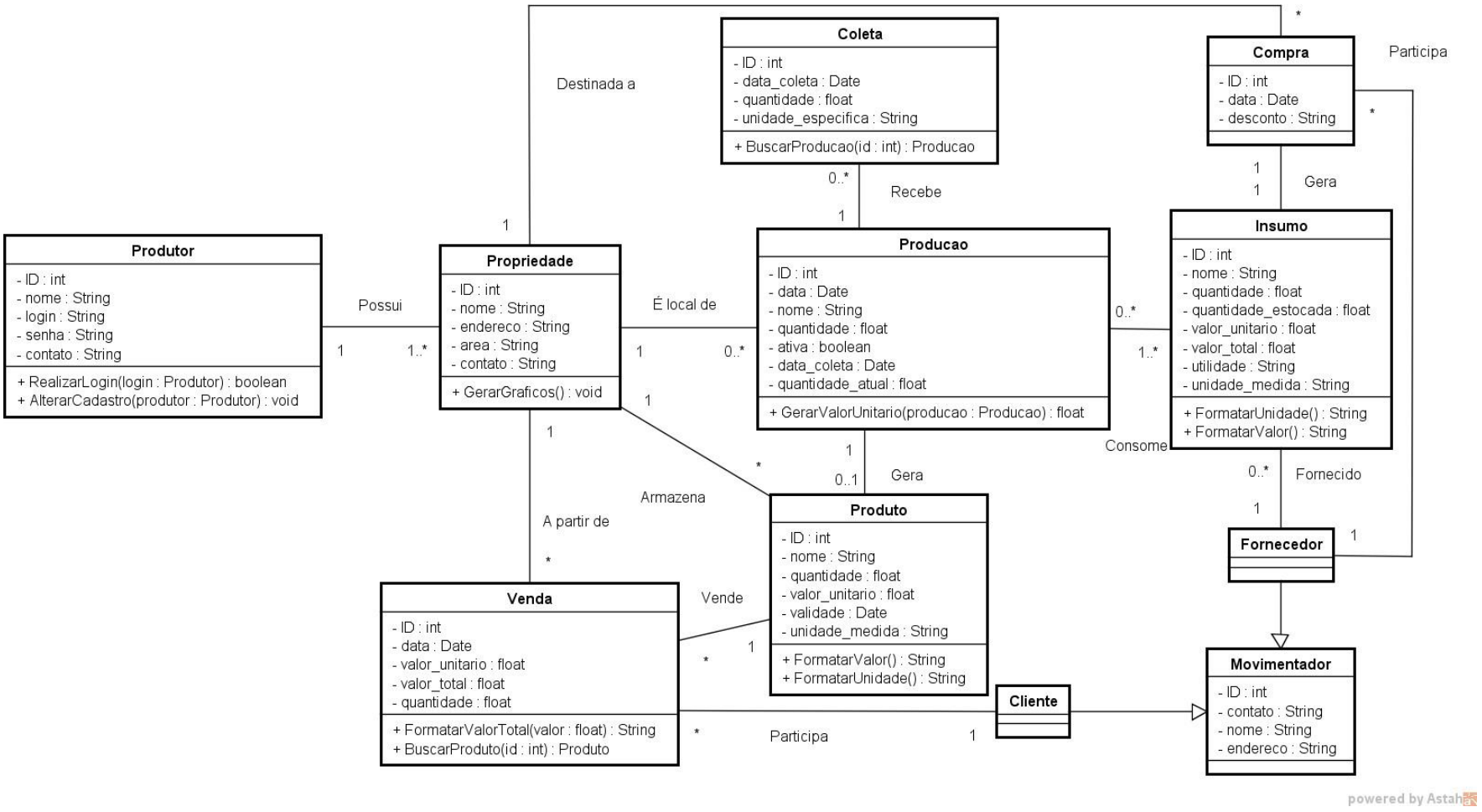
Fonte: elaborada pelo autor

3.5 Modelagem

A modelagem de dados promove documentação e visualização gráfica prévia dos dados necessários e procedimentos executados, o principal diagrama desenvolvido neste trabalho foi

o diagrama de classes, figura 7, disposto a seguir, responsável por elencar os elementos, seus dados, tipos de dados e relações entre os elementos.

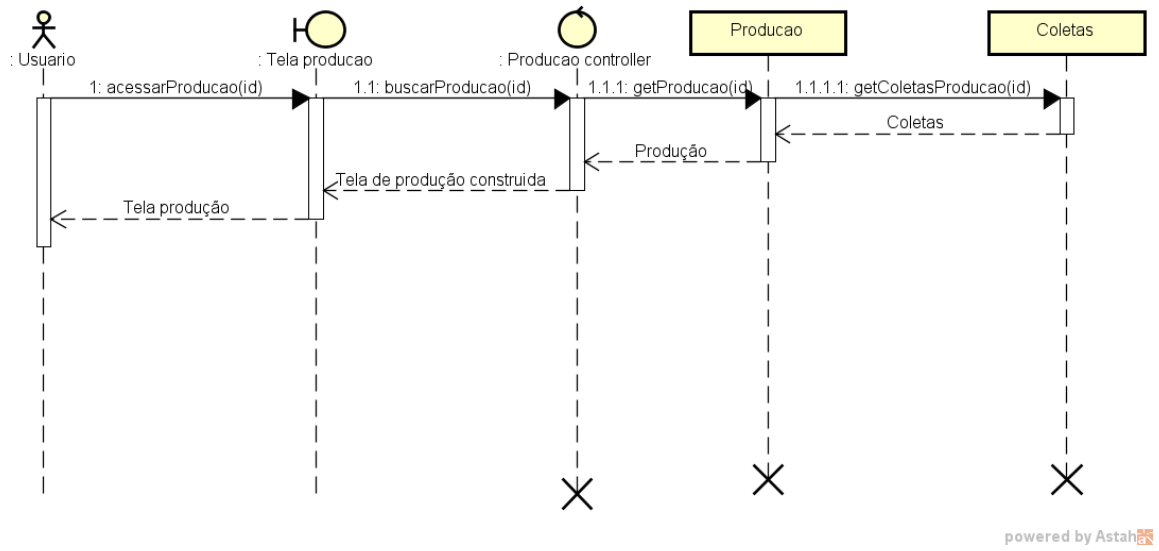
Figura 7 – Diagrama de classe



Fonte: elaborada pelo autor

A fim de modelar os procedimentos realizados pelo sistema foram desenvolvidos diagramas de sequência, que busca definir as ações realizadas pelos atores em determinada situação, por exemplo, a figura 8, a seguir, define os processos de carregamento da tela de detalhes de uma determinada produção.

Figura 8 – Diagrama de sequência

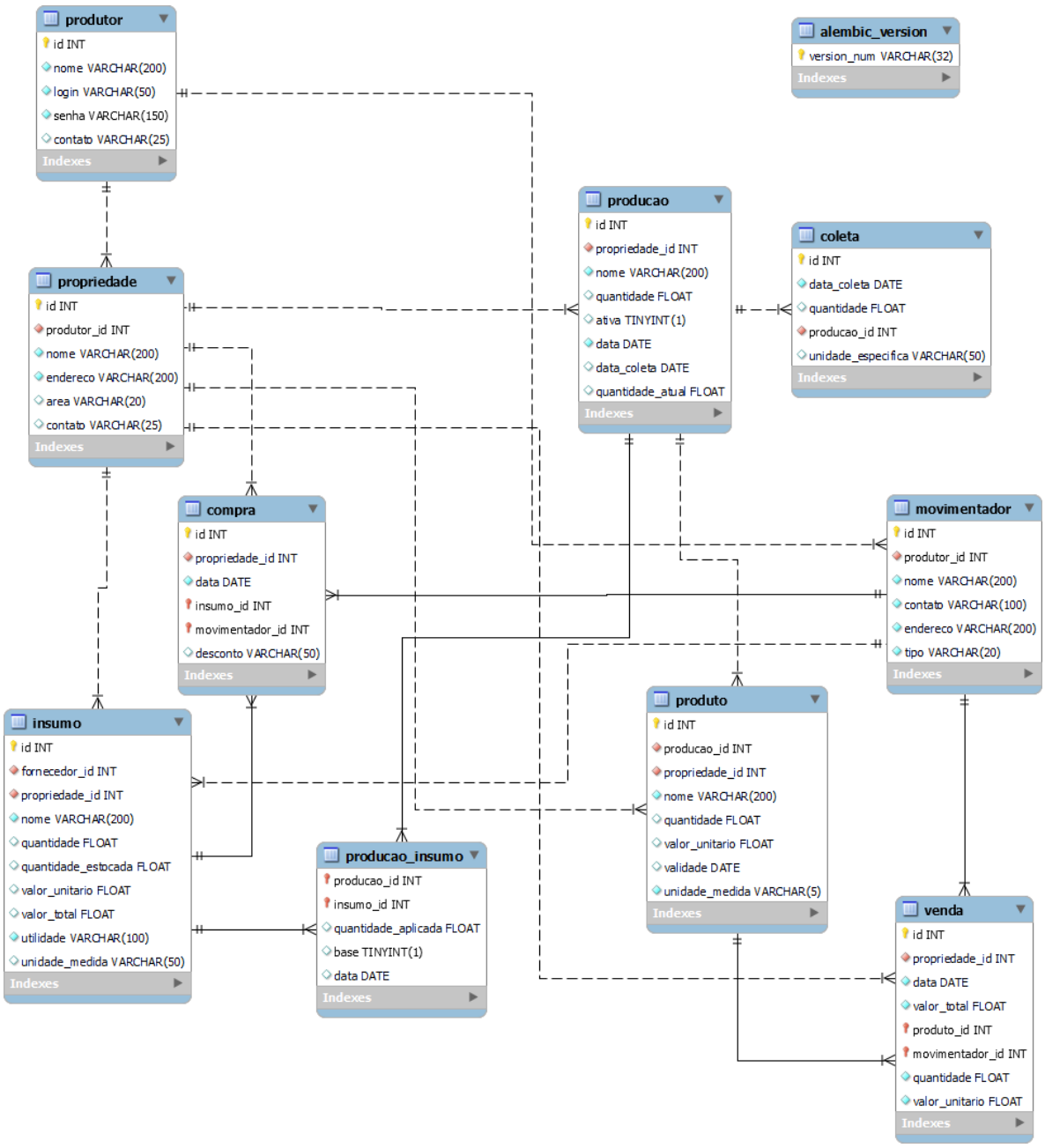


Fonte: elaborada pelo autor

3.6 Persistência de dados

Devido ao nível elevado de relacionamento apresentado no levantamento de requisitos, como pode ser observado no MER, figura 9, a seguir, registrando produtor, propriedade, fundamentais para registro das atividades seguintes e preparativos para futuros trabalhos que busquem escalar as funcionalidades, como múltiplas propriedades, movimentações financeiras, representadas por venda e compra que contam com a generalização chamada de movimentador que representa cliente ou fornecedor, produtos de estoque, insumos e produtos, oriundos de compras e produções, por fim a coleta, signo de colheita ou abate parcial de uma determinada produção.

Figura 9 – Diagrama de entidade e relacionamento



Fonte: elaborada pelo autor

3.7 Plano de testes

Testes unitários foram desenvolvidos principalmente para garantir o bom funcionamento das funções de tratamento de dados para construção das telas, construção de rota de login e redirecionamento automático de usuários não autorizados, para tal, a biblioteca *pytest*[12] foi utilizada. O quadro 1 apresenta os casos de teste.

[12] Disponível em https://docs.pytest.org/en/6.2.x/

Quadro 1 – Plano de testes

Título	**Execução**	**Resultado**
Resposta 200 para pagina login	Dado a rota "/login", o teste deve executar uma requisição à aplicação	O teste deve receber o status 200 como retorno da requisição
Resposta 302 para tentativa não autorizada de acesso página inicial	Dado a rota "/", o teste deve executar uma requisição à aplicação	O teste deve receber o status 302 como retorno da requisição aferindo que o redirecionamento acontece caso não esteja autenticado
Texto redirecionamento na resposta para tentativa de acesso não autorizada	Dado a rota "/", o teste deve executar uma requisição à aplicação	O teste aferir que: no corpo da resposta existe o comando de redirecionamento para a tela de login
Redirecionado para pagina login caso não autorizado	Dado a rota "/"e o comando para acompanhar redirecionamentos, o teste deve executar uma requisição à aplicação	O teste aferir que: a url de redirecionamento é a de login e o código de status da requisição é 200
Verificar método formatação quantidade insumo	Dada a quantidade de 10.0 e unidade de medida igual a sacos o teste deve executar a função quantidade_formatada()	O resultado aferido pelo teste deve ser igual a "10 sacos"comprovando que a função reconhece a falta de casas flutuantes e formata adequadamente
Verificar método formatação quantidade fracionada insumo	Dada a quantidade de 11.3 e unidade de medida igual a sacos o teste deve executar a função quantidade_formatada()	O resultado aferido pelo teste deve ser igual a "11.3 sacos"comprovando que a função reconhece as casas flutuantes e formata adequadamente
Verificar método formatação valor total insumo	Dado o valor de 10.0 o teste deve executar a função valor_final_formatado()	O resultado aferido pelo teste deve ser igual a "R$ 10.00"comprovando que a função formata os valores computados para a representação monetária correta
Verificar método valor total extenso insumo	Dado o valor de 11.2500001 o teste deve executar a função valor_final_formatado()	O resultado aferido pelo teste deve ser igual a "R$ 11.25"comprovando que a função formata os valores computados para a representação monetária correta, evitando exibição de casa flutuantes ínfimas utilizadas pela linguagem de programação.

Fonte: Elaborado pelo autor

3.8 Licença de uso

A licença escolhida para o projeto foi a MIT[13], buscando manter a proposta inicial de construir uma solução de código aberto, desenvolvida pelo Instituto de Tecnologia de Massachusetts, garantindo direitos a cópia, modificação e redistribuição a todos e garantindo ao autor isenção de qualquer dano ao utilizar o produto disponibilizado.

[13] Disponível em https://opensource.org/licenses/MIT

4 Resultados e discussões

Neste capítulo são apresentados os resultados e discussões acerca da aplicação proposta.

4.1 Gerenciamento de configuração e mudanças

Como citado anteriormente, no que se refere ao controle das versões, ao final de cada ciclo de desenvolvimento uma nova versão do sistema era registrada no repositório[1] da plataforma *GitHub*, para cada uma reservava-se uma ramificação da versão anterior, chamada comumente de *branch*, recebendo todos os novos incrementos de código, retornando e sendo mesclada, *merge*, à última versão registrada.

A lista a seguir elenca simplificadamente as atividades desenvolvidas, separadas por ciclo:

- 08/08/2021 - 14/08/2021:
  - Criação do projeto.
  - Criação do repositório.
  - Protótipo da tela de login.
- 15/08/2021 - 21/08/2021:
  - Criação dos arquivos de modelo para, produtor, propriedade, produção, movimentadores, venda, compra, produtos, insumos e relacionamentos produto_produção e produção_insumo.
- 29/08/2021 - 04/09/2021:
  - *Template* e estilo para tela de login.
  - Rota para login.
- 05/09/2021 - 11/09/2021:
  - *Template* e estilo base para todo o sistema, com preparação para navegação.
- 12/09/2021 - 18/09/2021:
  - Rota e *template* de formulário para registro de produtores.
- 19/09/2021 - 25/09/2021:

[1] Disponível em https://github.com/pedroferronato/gerenciamento-rural

  - Prototipação das telas referentes à propriedades.
  - *Templates* para recepção, registro e conferência referentes à propriedades.
  - Modelo de formulário de registro de propriedades.
  - Rotas referentes às operações das clientes.

- 26/09/2021 - 02/10/2021:
  - Rotas para alterações em registro de propriedade.
  - Construção de modal para exclusões.

- 03/10/2021 - 09/10/2021:
  - Prototipação das telas referentes aos clientes.
  - *Templates* para recepção, registro e conferência referentes aos clientes.
  - Modelo de formulário de registro aos clientes.
  - Rotas referentes às operações dos clientes.
  - Construção do rodapé fixo.

- 10/10/2021 - 16/10/2021:
  - Rota de busca de clientes.
  - *Template* para paginação.

- 17/10/2021 - 23/10/2021:
  - Prototipação das telas referentes aos fornecedores.
  - *Templates* para recepção, registro, conferência e busca referentes aos fornecedores.
  - Modelo de formulário de registro e busca aos fornecedores.
  - Rotas referentes às operações dos fornecedores.

- 24/10/2021 - 30/10/2021:
  - Prototipação das telas referentes às produções.

- 31/10/2021 - 06/11/2021:
  - Prototipação das telas referentes às produções.
  - *Templates* para recepção, registro, conferência e listagem de estados referentes às produções.
  - Modelo de formulário de registro às produções.
  - Rota e *template* para verificação de estoque.

- 07/11/2021 - 13/11/2021:
  - Rotas referentes às operações das produções.
  - Rotas para adição de custeio.
  - Rotas para listagem das produções finalizadas e ativas.
  - *Templates* para recepção, registro, conferência e listagem referentes às compras.
- 14/11/2021 - 20/11/2021:
  - Modelo de formulário de registro às compras.
  - Rotas referentes às operações das compras.
  - Adequação dos formulários ao modelo de *datalist*.
  - *Templates* para recepção, registro, conferência e listagem referentes às vendas.
- 21/11/2021 - 27/11/2021:
  - Modelo de formulário de registro e busca às vendas.
  - Rotas referentes às operações das vendas.
  - Construção do *template* de *dashboard*.
  - Construção da rota de *dashboard*.

4.2 Processo de desenvolvimento

O gerenciamento do processo de desenvolvimento foi realizado por meio da metodologia *Kanban*, as colunas definidas foram:

- "Novo", onde as tarefas elencadas para resolução de uma funcionalidade, defeitos e tarefas secundárias ficavam descritas aguardando desenvolvimento.
- "Em andamento", coluna destinada a registrar atividades em processo de desenvolvimento.
- "Pronto para teste", soluções finalizadas, porém, aguardando maior bateria de teste, onde também foram desenvolvidos alguns dos testes automatizados.
- "Terminado", local onde ficavam descritas as funcionalidades submetidas ao repositório, ou seja, concluídas.

Gerenciar o projeto com *Kanban* se provou muito útil, principalmente se tratando de um projeto desenvolvido por único programador, tornando simples o registro de novas atividades para controle, produção de métricas referentes ao processo, produzindo noção e previsão de

entrega das funcionalidades, capacidade de tomada de decisão quanto à velocidade de produção das entregas (SILVA; SANTOS; NETO, 2012).

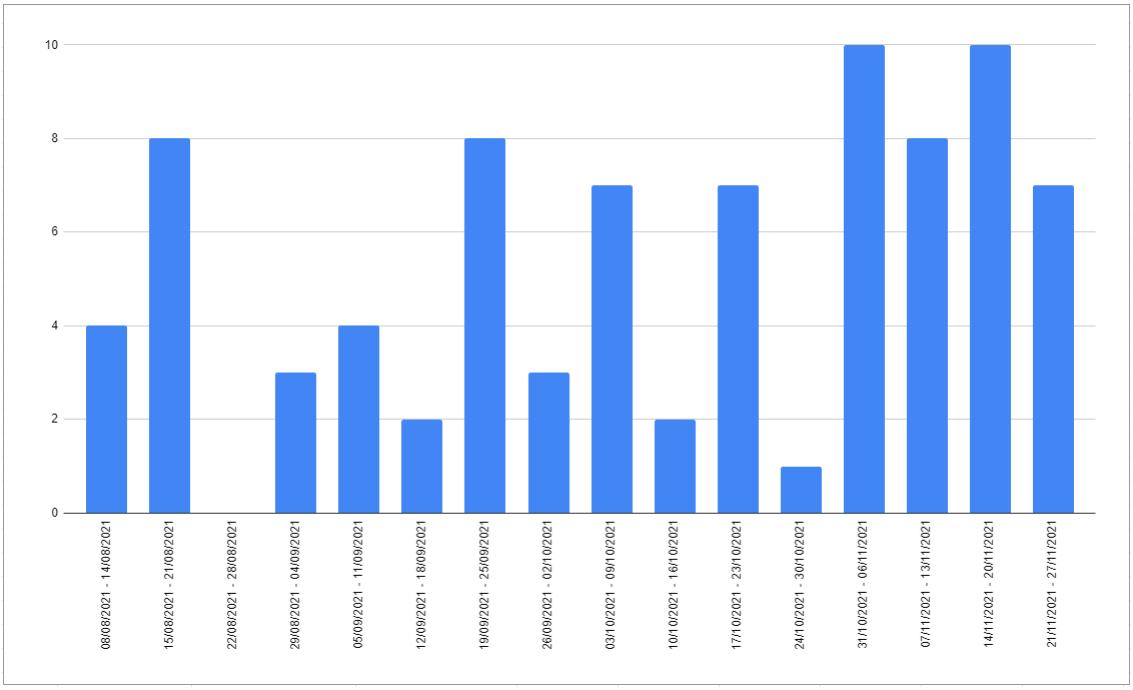
Inicialmente adotou-se um ritmo reduzido de desenvolvimento, ou seja, funcionalidades como construção do protótipo e construção de rotas tomavam um ciclo completo, aproximadamente uma semana, para serem desenvolvidas, tempo de entrega que reduziu muito ao final do processo de desenvolvimento em decorrência à utilização dos materiais anteriormente produzidos na solução, a figura 10 representa o gráfico de entrega de tarefas por ciclo.

Durante e ao fim do processo de desenvolvimento levantaram-se as métricas, baseadas no quadro *Kanban*, sendo elas:

- *Cycle time*, tempo que uma tarefa passou em cada coluna até chegar ao estado de "Terminado".
- *Lead time*, tempo total destinado da definição de uma tarefa até a sua conclusão
- *Throughput*, quantidade de tarefas desenvolvidas em um determinado espaço de tempo, para este projeto, um ciclo, aproximadamente uma semana.

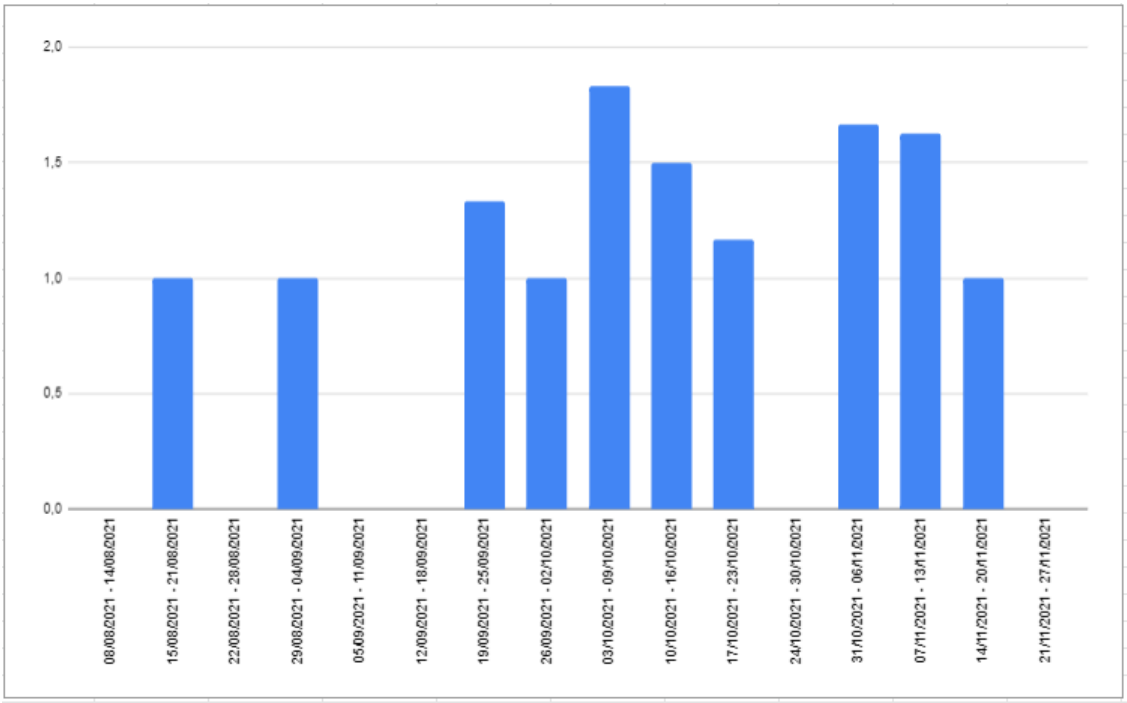
As figuras a seguir, demonstram os gráficos destas métricas.

Figura 10 – Gráfico de *throughput* do processo



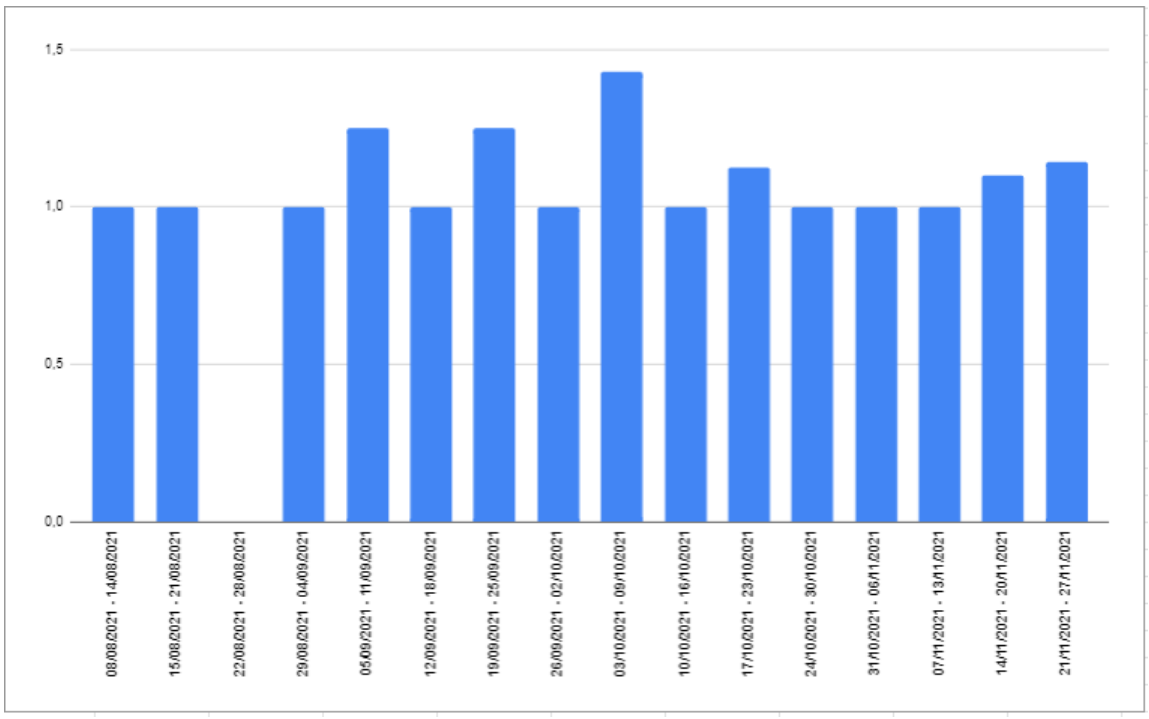
Fonte: elaborada pelo autor

Figura 11 – Gráfico de *cycle time* coluna "Novo"do processo



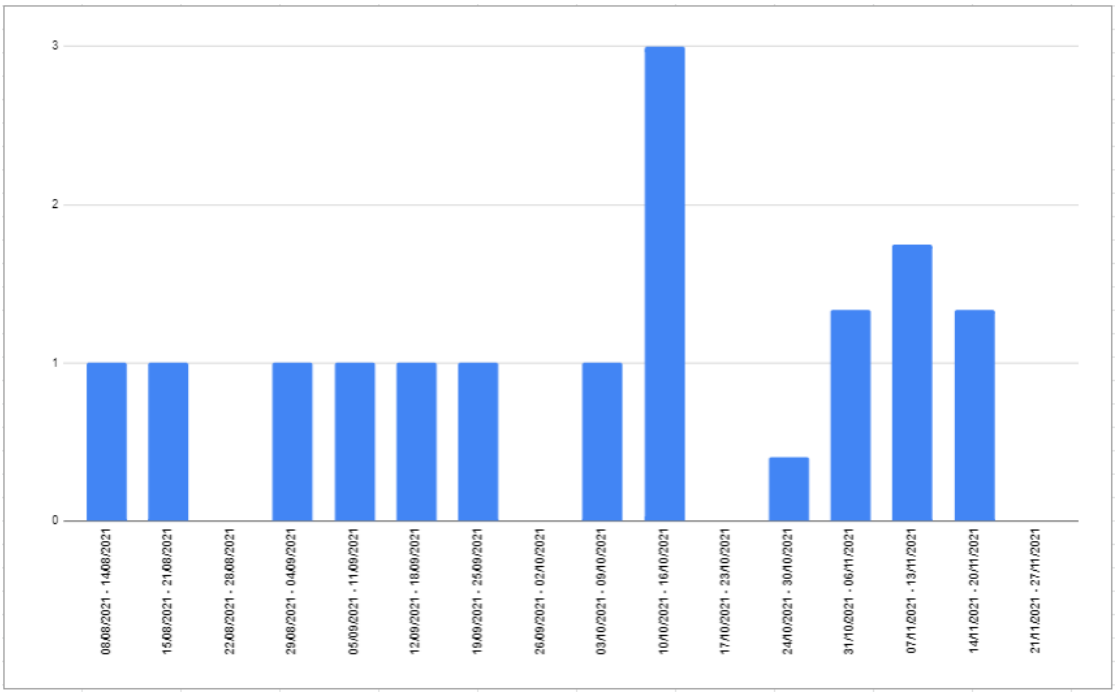
Fonte: elaborada pelo autor

Figura 12 – Gráfico de *cycle time* coluna "Em andamento"do processo



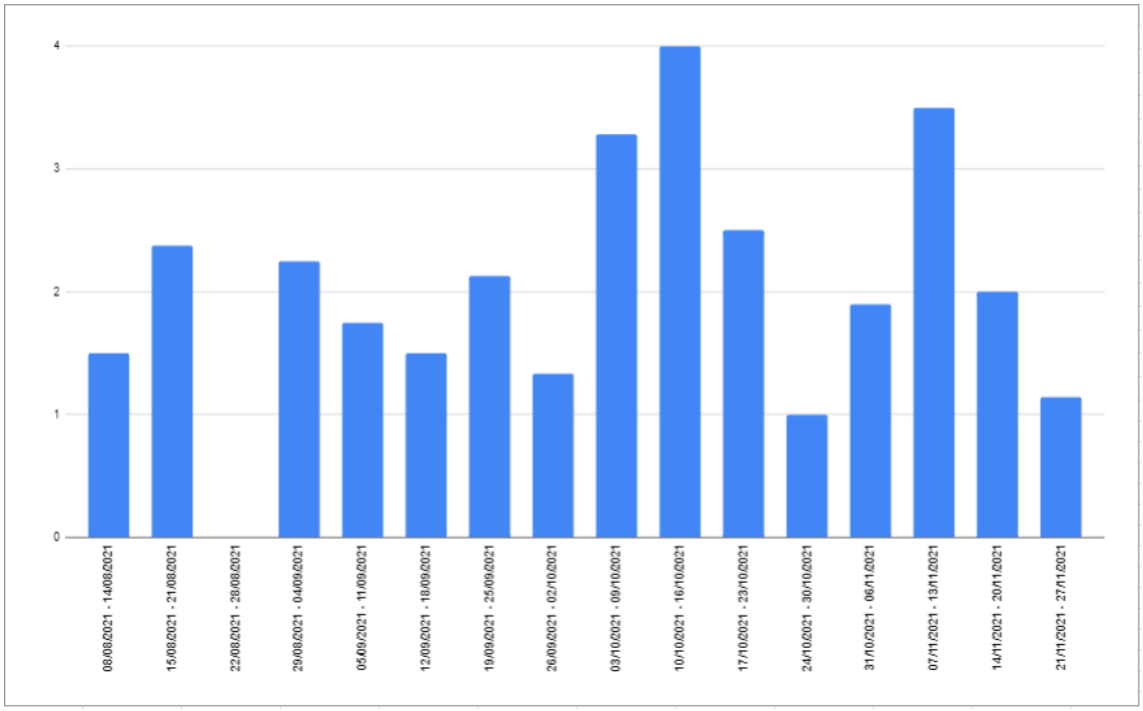
Fonte: elaborada pelo autor

Figura 13 – Gráfico de *cycle time* coluna "Análise"do processo



Fonte: elaborada pelo autor

Figura 14 – Gráfico de *lead time* do processo



Fonte: elaborada pelo autor

Nota-se na figura 10 oscilação na entrega das funcionalidades, normalizada em nível alto no último mês, os gráficos, 11, 12 e 13, representam, a cada ciclo, quantos dias em média uma *story* permanência em cada coluna, valendo ressaltar a constância de 1 dia na fase de desenvolvimento, por fim, a figura 14 que demonstra, a cada ciclo média aproximada de duas que uma atividade levou para ser concebida, construída e validada.

O *Flask*, associado ao *Jinja*[2], ferramenta de construção de *templates* HTML e classes CSS comuns a todo o sistema, promoveram principalmente a aceleração nas últimas semanas de desenvolvimento, mas não somente isso, a capacidade de reconhecer elementos e suas ações ou funcionalidades na tela, no que se refere ao usuário, maior, atendendo assim parte da busca em reduzir a curva de aprendizado do sistema. A figura 15 representa a estrutura, sem dados, de um *template* utilizado para disposição de elementos que proporcionam redirecionamento ao detalhes do referido item, a figura 16 representa estrutura básica de construção, utilizado em grande parte do sistema, de um campo gerado por modelo *Flask-WTF*, já com tratamento de erros previstos.

[2] Disponível em https://jinja.palletsprojects.com/en/3.0.x/

Figura 15 – Código base para cartões de redirecionamento

```
<a href="">
    <div class="cartao minicard">
        <div>
            <p class="minicard-titulo">Título</p>
            <p class="minicard-subtitulo">Subtítulo</p>
        </div>
        <div>
            <span class="material-icons minicard-icon">description</span>
        </div>
    </div>
</a>
```

Fonte: elaborada pelo autor

Figura 16 – Código base para campos de formulário

```
<div>
    {{ form.campo.label }}
    {% if form.campo.errors %}
        {{ form.campo(class="input-error") }}
        {% for error in form.campo.errors %}
            <span class="span-erro">{{ error }}</span>
        {% endfor %}
    {% else %}
        {{ form.campo }}
    {% endif %}
</div>
```

Fonte: elaborada pelo autor

O sistema baseou-se na arquitetura MVC para estruturação dos arquivos, sendo *app* a pasta principal, onde modelos, controladores, *templates*, arquivos estáticos e formulários ficam registrados, externo a ela temos arquivos de configuração de ambiente e versionamento, bem como arquivo raiz de execução do produto.

A pasta *controllers* tem com objetivo armazenar os arquivos, que obedecem a nomenclatura "{elemento}_controller.py", responsáveis por definir as rotas acessíveis pelo cliente, bem como as telas que devem ser construídas e comportamentos necessário em condição aos dados recebidos.

Quanto às telas, fica como responsabilidade da pasta *templates* armazená-las, arquivos HTML que seguem a nomenclatura "{elemento}_{função}.html", em *static* definem-se arquivos de imagem, JavaScript e estilo para as telas supracitadas. A pasta *forms* armazena os arquivos, orientados à nomenclatura "{elemento}_form.py", detentores das classes construtoras de formulários para as telas do sistema.

Ficando *models*, responsável pelos arquivos de definição de tabelas da base de dados, utilizados tanto pelo ORM na construção da base de dados quanto pelo mesmo nas rotas para interações com a base de dados.

Por fim, o arquivo "_init_.py", responsável por unir todas as pastas anteriores e configurações de framework em uma instância *Flask*, utilizada para execução de todo o sistema.

4.3 Relatório dos testes

Todo ciclo de desenvolvimento do sistema estava correlacionada a baterias de teste manuais, que buscavam encontrar defeitos e eventuais falhas em inserções indevidas de dados pelo usuário, ademais, testes automáticos foram construídos para verificar bom funcionamentos das funcionalidades de construção das informações para exposição em telas, como valores e quantidades, a detecção e redirecionamento corretos caso o usuário não esteja autenticado, evitando acessos indevidos e por fim o bom funcionamento de rotas abertas, como a de login.

Para tal, utilizou-se a biblioteca *pytest*, uma das mais utilizadas pela comunidade *Python*, a pastas padrão e destinada aos mesmos é a chamada *tests*, onde definiu-se o arquivo de configuração, responsável por disponibilizar variável de testes, que representa uma versão do produto, porém configurada para ambiente de testes, em arquivo separado definem-se os testes a serem executados, veja a figura 17 para exemplo de implementação de teste, pelo comando ’pytest’, recebendo o retorno da confirmação do sucesso em todos os testes definidos, demonstrado na figura 18.

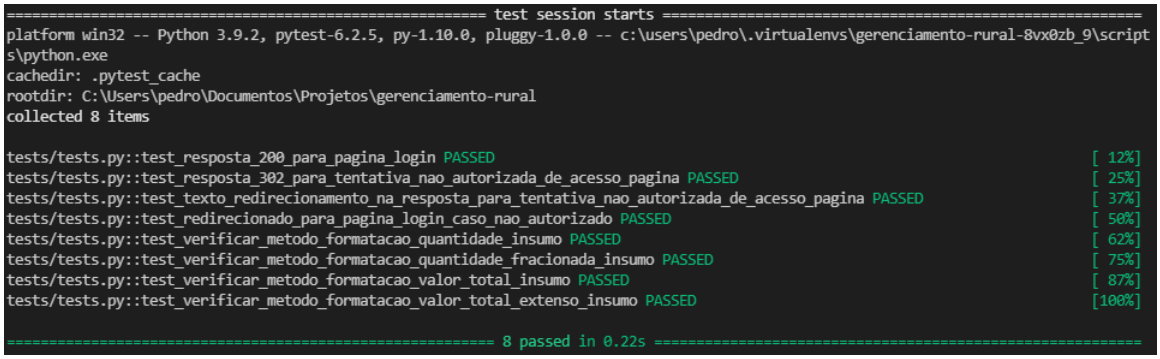
Figura 17 – Exemplo de código de teste

```
def test_resposta_302_para_tentativa_nao_autorizada_de_acesso_pagina(client):
    request = client.get('/')

    assert request.status_code == 302
```

Fonte: elaborada pelo autor

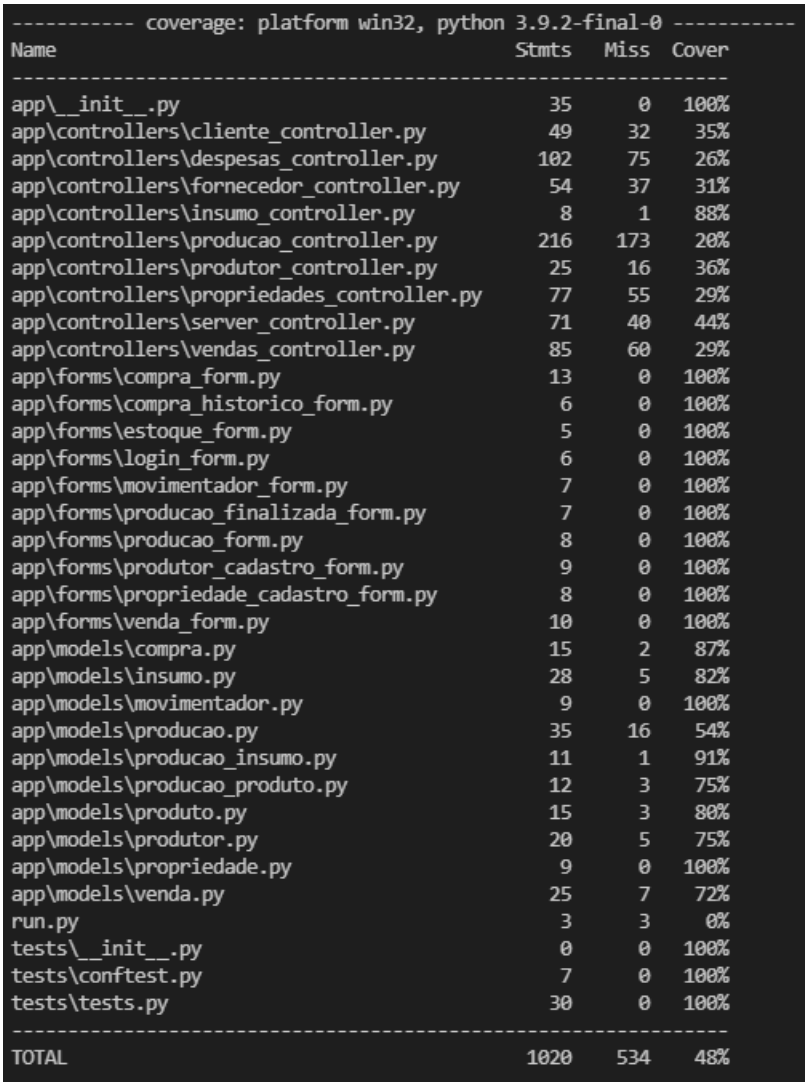
Figura 18 – Relatório de testes



Fonte: elaborada pelo autor

A figura 19 apresenta a execução dos testes com relatório de cobertura, podendo observar uma cobertura geral de 48%, número reduzido principalmente por funcionalidades desenvolvidas nos *controllers*, sendo possível e muito desejável em trabalhos futuros aprimorar esta taxa, buscado a cobertura completa.

Figura 19 – Cobertura dos testes



```
---------- coverage: platform win32, python 3.9.2-final-0 ----------
Name                                              Stmts   Miss  Cover
---------------------------------------------------------------------
app\__init__.py                                      35      0   100%
app\controllers\cliente_controller.py                49     32    35%
app\controllers\despesas_controller.py              102     75    26%
app\controllers\fornecedor_controller.py             54     37    31%
app\controllers\insumo_controller.py                  8      1    88%
app\controllers\producao_controller.py              216    173    20%
app\controllers\produtor_controller.py               25     16    36%
app\controllers\propriedades_controller.py           77     55    29%
app\controllers\server_controller.py                 71     40    44%
app\controllers\vendas_controller.py                 85     60    29%
app\forms\compra_form.py                             13      0   100%
app\forms\compra_historico_form.py                    6      0   100%
app\forms\estoque_form.py                             5      0   100%
app\forms\login_form.py                               6      0   100%
app\forms\movimentador_form.py                        7      0   100%
app\forms\producao_finalizada_form.py                 7      0   100%
app\forms\producao_form.py                            8      0   100%
app\forms\produtor_cadastro_form.py                   9      0   100%
app\forms\propriedade_cadastro_form.py                8      0   100%
app\forms\venda_form.py                              10      0   100%
app\models\compra.py                                 15      2    87%
app\models\insumo.py                                 28      5    82%
app\models\movimentador.py                            9      0   100%
app\models\producao.py                               35     16    54%
app\models\producao_insumo.py                        11      1    91%
app\models\producao_produto.py                       12      3    75%
app\models\produto.py                                15      3    80%
app\models\produtor.py                               20      5    75%
app\models\propriedade.py                             9      0   100%
app\models\venda.py                                  25      7    72%
run.py                                                3      3     0%
tests\__init__.py                                     0      0   100%
tests\conftest.py                                     7      0   100%
tests\tests.py                                       30      0   100%
---------------------------------------------------------------------
TOTAL                                              1020    534    48%
```

Fonte: elaborada pelo autor

4.4 Implantação

Para implantação do produto final foi utilizada uma máquina virtual sobre o servidor interno do Instituto Federal - *Campus* Vilhena, onde o repositório[3] foi clonado, tanto este quanto o banco de dados tiveram sua implantação realizada em um *container Docker*. Após a compilação para produção e execução do sistema, tornou-se possível acessar a aplicação através do *link* público[4].

[3] Disponível em https://github.com/pedroferronato/gerenciamento-rural
[4] Disponível em https://ruralfinance.fslab.dev/

4.5 Demonstração do software

Por se tratar de um sistema aberto qualquer um pode acessá-lo pelo *link* citado anteriormente, realizar seu cadastro na plataforma e desfrutar das funcionalidades de registro de propriedade, produção, estimativas de preço para produtos, estoque e movimentações financeiras

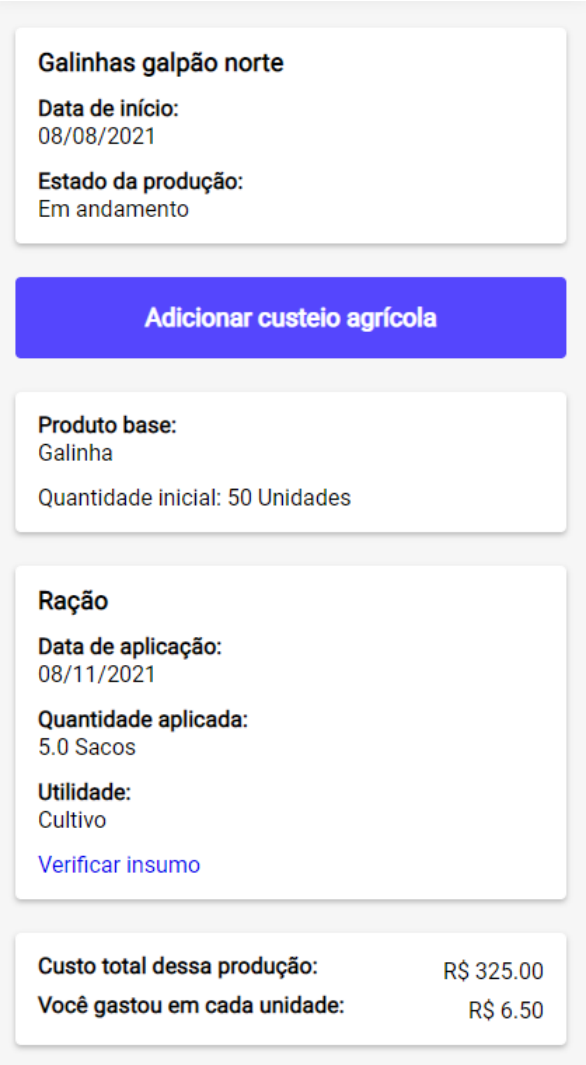
A imagem 20 demonstra a página de listagem de todas as produções ativas no momento em uma determinada propriedade.

Figura 20 – Página de listagem de produções ativas

Fonte: elaborada pelo autor

A imagem 21 demonstra um fragmento da página de detalhes de uma determinada produção, exibindo intervenções realizada pelo produtor e estimativa de gasto realizado para produção de cada produto final. Nomeia-se custeio as interferências a uma produção, como no caso, aplicação de ração para alimentação.

Figura 21 – Fragmento da página de detalhes de uma determinada produção



Fonte: elaborada pelo autor

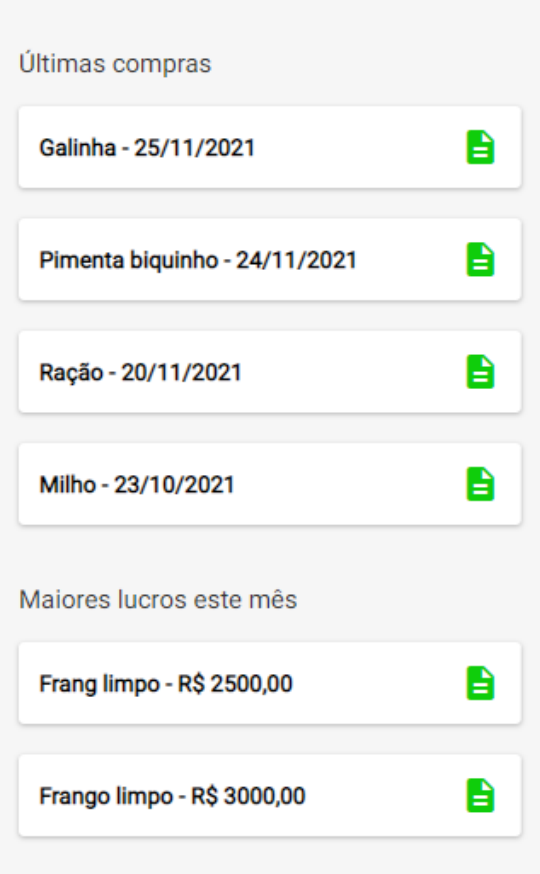
A imagem 22 apresenta o fragmento superior da página de *dashboard*, apresentando cartões referentes, respectivamente, ao caixa mensal atual, sendo considerado a diferença entre vendas e compras, valor total de vendas e valor total de compras.

Figura 22 – Fragmento da página de *dashboard*

Fonte: elaborada pelo autor

A figura 23 apresenta o fragmento sequência da página de *dashboard*, apresentando cartões referentes às últimas compras realizadas e também maiores vendas do mês atual.

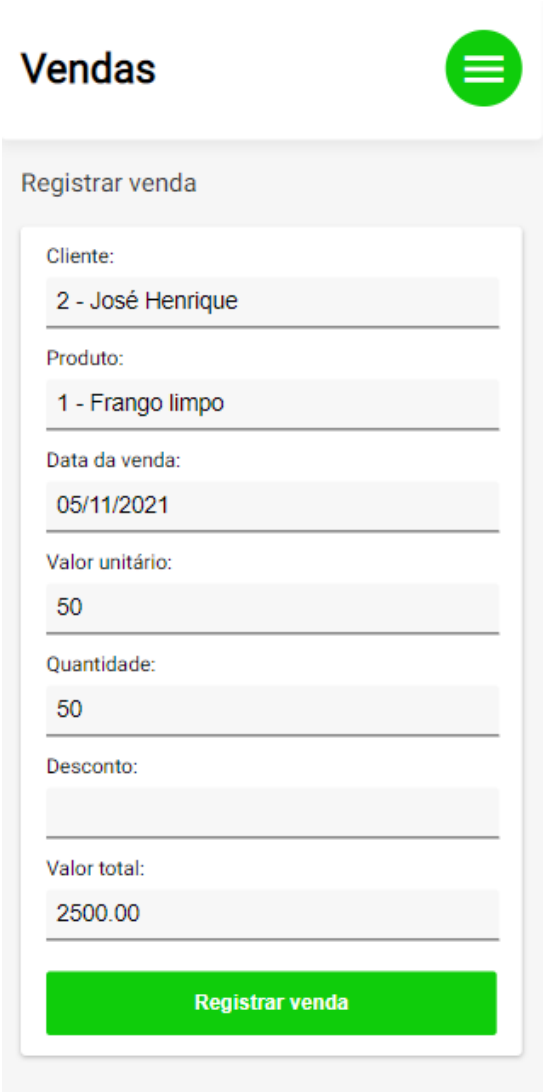
Figura 23 – Fragmento da página de *dashboard* - Cartões inferiores



Fonte: elaborada pelo autor

A seguir, a figura 24, representa a tela de registro de vendas, exemplo de formulário, os campos, "Cliente"e "Produto"se tratam de campos de seleção, alimentados pela base de dados, a fim de manter estoque controlado e dados de registro coeso, para facilitar o processo, data e valor total são preenchidos automaticamente, possibilitando edição em caso de discordância, principalmente a data.

Figura 24 – Tela de registro de vendas



Fonte: elaborada pelo autor

5 Considerações finais

Desenvolver esse projeto trouxe este projeto trouxe experiência, solidificação de conhecimentos adquiridos durante todo o curso de análise e desenvolvimento de sistemas e aquisição de muitos mais. Desenvolver um projeto completo seguindo e utilizando um processo bem definido de gerência trouxe estabilidade e noção dos limites de entrega das funcionalidades, produzir desde a modelagem do projeto passando por protótipos e até o sistemas em si consolidou fundamentos adquiridos em todos os anos do curso e não somente isso, apontou a necessidade constante de aprimoramento como profissional da área.

O produto final atende ao seu intuito inicial de auxiliar o produtor rural em suas movimentações financeiras relacionadas às suas produções, porém não é uma solução universal, este projeto ainda detém potencial enorme de evolução atendendo o usuário final em maior espectro.

5.1 Trabalhos futuros

Após o desenvolvimento deste trabalho, a fim de promover a melhora definem-se os seguintes trabalhos futuros:

- Adicionar gráficos a fim de tornar dados contidos no sistema ainda mais visuais.
- Adicionar capacidade de múltiplas propriedades ao sistema.
- Transformar o sistema em produtos independentes, ou seja, construí-lo em diferentes aplicações, API para gerência do banco de dados, sistema mobile nativo para eficiência maior em celulares e aplicação dedicada para WEB, os dois anteriormente citados consumindo a API, aprimorando a eficiência e escalabilidade do sistema.
- Tornar o sistema mais completo para o usuário, registrando alterações intencionais ou acidentais em uma produção, trazendo essas informações em uma linha do tempo para conferência das ocorrências.

Referências

AEGRO. *Aegro*. 2021. [Online; acessado 19-Novembro-2021]. Disponível em: <https://conhecimento.aegro.com.br/contato-p>. Citado na página 27.

AGRISOFT. *Agrisoft*. 2021. [Online; acessado 19-Novembro-2021]. Disponível em: <https://www.agrisoft.com.br>. Citado na página 27.

APOLINÁRIO, D. R. d. F. et al. Desenvolvimento de software na embrapa: Abordagem a partir da teoria ator-rede. Revista Brasileira de Gestão e Inovação, v. 4, n. 3, p. 64–88, 2017. Citado na página 23.

BECKER, E. A tecnologia da informação aplicada à produção de alimentos. Revista da UNIFEBE, v. 7, n. 7, p. 231–240, 2002. Citado na página 25.

CHBAGRO. *CHBAGRO*. 2021. [Online; acessado 19-Novembro-2021]. Disponível em: <https://chbagro.com.br>. Citado na página 27.

CHIOZINI, P. H. P.; CARVALHO, K. A. Estudo sobre os efeitos da implantação de um sistema de gestão informatizado em uma propriedade rural em rio verde – go. UNIRV–Universidade do Rio Verde, 2017. Citado na página 26.

FARMOS. *farmOS*. 2021. [Online; acessado 19-Novembro-2021]. Disponível em: <https://farmos.org>. Citado na página 27.

LITEFARM. *LiteFarm*. 2021. [Online; acessado 19-Novembro-2021]. Disponível em: <https://www.litefarm.org>. Citado na página 27.

MEIRA, C. A. A. et al. Agroinformática: Qualidade e produtividade na agricultura. Embrapa - Cadernos de Ciência Tecnologia, v. 13, n. 2, p. 175–194, 1996. Citado na página 25.

MENEZES, N. N. C. *Introdução à programação com Python*. São Paulo: Novatec Editora, 2010. v. 1. Citado na página 32.

SILVA, D. V. d. S.; SANTOS, F. A. d. O.; NETO, P. S. Os benefícios do uso de kanban na gerência de projetos de manutenção de software. in: Simpósio brasileiro de sistemas de informação (sbsi). Anais [...]. Porto Alegre: Sociedade Brasileira de Computação, v. 8, p. 715–725, 2012. Citado 2 vezes nas páginas 26 e 44.

SILVA, J. E. C. F. d.; BOTELHO, M. F. Cadastro ambiental rural utilizando imagem de drone aerofotogramétrico. Revista Agrogeoambiental, v. 9, n. 2, p. 73–84, 2017. Citado na página 25.

TANIA. *Tania*. 2021. [Online; acessado 19-Novembro-2021]. Disponível em: <https://usetania.org>. Citado na página 27.

Anexos

ANEXO A – Licença MIT