Universidade de Brasília – UnB

Faculdade UnB Gama – FGA

Engenharia de Software

Associação para aplicações Agromart de uma CSA em cloud

Autor: Abner Filipe Cunha Ribeiro e Rafael Leão Teixeira de Magalhães

Orientador: Prof. André Luiz Peron Martins Lanna

Brasília, DF

2023

Abner Filipe Cunha Ribeiro e Rafael Leão Teixeira de Magalhães

Associação para aplicações Agromart de uma CSA em cloud

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: Dr. Rudi Henri van Els

Brasília, DF

2023

Abner Filipe Cunha Ribeiro e Rafael Leão Teixeira de Magalhães
Associação para aplicações Agromart de uma CSA em cloud/ Abner Filipe Cunha Ribeiro e Rafael Leão Teixeira de Magalhães. – Brasília, DF, 2023-
51 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. André Luiz Peron Martins Lanna

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

1. Agromart. 2. Cloud. I. Prof. André Luiz Peron Martins Lanna. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Associação para aplicações Agromart de uma CSA em cloud

CDU 02:141:005.6

Abner Filipe Cunha Ribeiro e Rafael Leão Teixeira de Magalhães

Associação para aplicações Agromart de uma CSA em cloud

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.

Prof. André Luiz Peron Martins Lanna
Orientador

Prof. Cristiane Soares Ramos
Convidado 1

Prof. Ricardo Ajax Dias Kosloski
Convidado 2

Brasília, DF
2023

Agradecimentos

Eu, Abner Filipe Cunha Ribeiro quero expressar minha profunda gratidão aos meus professores por sua valiosa contribuição na minha formação acadêmica, bem como aos meus queridos familiares pelo apoio incondicional que me deram diariamente. A dedicação, conhecimento e orientação dos meus professores foram fundamentais para o meu crescimento pessoal. Além disso, o suporte constante e encorajador dos meus familiares foi essencial para superar os desafios ao longo dessa jornada. Também sou grato aos amigos que estiveram ao meu lado, compartilhando experiências e incentivando-me a seguir em frente. A cada um de vocês, meu sincero agradecimento por fazerem parte dessa conquista.

Eu, Rafael Leão Teixeira de Magalhães, gostaria de expressar minha imensa gratidão aos meus amigos, familiares e professores pelo apoio inestimável ao longo da minha jornada. Agradeço aos meus amigos e familiares pelo constante encorajamento, apoio emocional e compreensão durante esse período desafiador. Suas palavras de incentivo e presença ao meu lado fizeram toda a diferença. Aos meus queridos professores, sou profundamente grato pelo conhecimento compartilhado e pelos valiosos feedbacks que moldaram o meu trabalho. Sua dedicação e comprometimento contribuíram significativamente para o meu crescimento acadêmico. A todos vocês, amigos, familiares e professores, meu mais sincero agradecimento por fazerem parte dessa jornada e por tornarem esse momento possível.

Resumo

O AgroMart é um *software* que tem como proposta auxiliar o escoamento da produção dos agricultores, promovendo um ambiente propício para as Comunidades que Sustentam a Agricultura por meio de uma interface *web* e um aplicativo *mobile*. Este trabalho tem como objetivo a associação de aplicações AgroMart de uma Comunidade que sustenta Agricultura na *Cloud* por meio de serviços como o AWS Lambda, o DynamoDB e o Amazon SES reduzindo, assim, os custos de mantimento de alguns serviços por parte da Universidade de Brasília(UnB). Com este trabalho, espera-se contribuir para o AgroMart com evolução, manutenção, documentação e redução de custos.

Palavras-chave: Agromart. Comunidades que sustentam a Agricultura(CSA). Cloud. Redução de custos. AWS. Universidade de Brasília(UnB)

Abstract

AgroMart is a software aimed at assisting farmers in the distribution of their produce, creating a conducive environment for Communitie That Support Agriculture (CSA) through a web interface and a mobile application. This project proposes the integration of AgroMart applications with a cloud-based CSA using AWS services such as AWS Lambda, DynamoDB, and Amazon SES. Additionally, previous improvements such as the automation of provisioning individualized CSAs will be maintained, but leveraging the new cloud resources. The goal is to contribute to AgroMart through evolution, maintenance, documentation, and cost reduction.

Key-words: AgroMart. Communities That Support Agriculture. Cloud. Cost reduction. AWS.

Lista de ilustrações

Lista de abreviaturas e siglas

CSA	Comunidade que Sustenta Agricultura
XP	*Extreme Programming*
AWS	Amazon *Web Services*
SDK	*Software Development Kit*
API	*Application Programming Interface*
URL	*Uniform Resource Locator*
FaaS	*Function as a Service*
UNB	Universidade de Brasília
Amazon SES	Amazon *Simple Email Service*
POC	*Proof of concept*
TCC	Trabalho de Conclusão de Curso
CMS	*Content Managment System*
AWS SAM	*AWS Serverless Application Model*

Sumário

1 Introdução

O setor aropecuário além do seu impacto local, também trouxe protagonismo à produção agropecuária brasileira frente ao cenário internacional. Por conta disso, o Brasil consolidou sua posição como um dos principais players do mercado global de produção e exportação de alimentos.

O setor agropecuário familiar concomitantemente a isso desempenha um papel fundamental, servindo de meio de geração de empregos e produção de alimentos para comércios locais. Além da produção excedente que gera riquezas que impulsionam a ecnomoia do país, a produção voltada para o consumo próprio das famílias contribui para suprir necessidades básicas e movimentação da economia local. (GUILHOTO; OUTROS, 2017).

Em meio à pandemia de COVID-19, a agricultura familiar precisou se adaptar a novas circunstâncias. As Comunidades que Sustentam a Agricultura (CSAs), também conhecidas como Agricultura Apoiada pela Comunidade, surgiram como uma alternativa promissora para apoiar a produção local de alimentos e criar espaços de interação entre áreas urbanas e rurais. Por meio de uma contribuição mensal fixa, os co-agricultores recebem regularmente uma variedade de produtos agrícolas frescos e sazonais, cultivados de forma sustentável, enquanto os agricultores obtêm uma renda mais estável e estabelecem uma conexão mais próxima com a comunidade. Essa abordagem fortalece tanto a segurança alimentar quanto a sustentabilidade, estabelecendo uma relação mais consciente com a natureza(MEIRELES, 2018). O agricultor deixa de vender seus produtos através de intermediários e conta com a participação das pessoas para a organização e financiamento de sua produção. Quem escolhe fazer parte de uma CSA, deixa de ser apenas um consumidor e se torna um co-agricultor. Passa a colaborar para o desenvolvimento sustentável da região estimulando o comércio justo e valorizando a produção local, podendo conhecer de perto de onde vem o seu próprio alimento(MATRES, 2023).

Assim, é importante ressaltar a relevância social e econômica dos agricultores familiares, cuja produção abastece as famílias de baixa renda e contribui significativamente para a economia do país. Além disso, a pandemia trouxe à tona a necessidade de adaptar as práticas agrícolas, resultando no surgimento das CSAs como uma solução inovadora. Por meio dessas comunidades, consumidores e agricultores estabelecem uma relação mais direta e sustentável, promovendo o consumo consciente e fortalecendo os laços entre a cidade e o campo. Nesse contexto, a agricultura familiar desempenha um papel fundamental na garantia de alimentos frescos, saudáveis e locais, bem como na sustentabilidade ambiental e no desenvolvimento das comunidades rurais.

Visando aproximar a relação entre pequenos agricultores e co-agricultores, foi desenolvido através de um *hackathon*, o *software* denominado Agromart. Essa aplicação promove um ambiente individualizado para as Comunidades que Sustentam a Agricultura (CSAs), que facilita a interação entre os membros, proporcionando um ambiente sustentável de produção e direcionamento do excedente dessa produção de forma rápida e fácil.

Com o objetivo de estabelecer uma conexão mais estreita entre os pequenos agricultores e os co-agricultores, foi desenvolvida a solução de *software* denominada Agromart. Essa plataforma tem como finalidade facilitar a interação entre os membros, promovendo um ambiente propício para as Comunidades que Sustentam a Agricultura (CSAs) e auxiliando no escoamento da produção dos agricultores.

1.1 História do AgroMart

O Agromart surgiu em um *Hackaton* realizado na UNB no campus do Gama, em 2020. A ideia surgiu a partir de uma reportagem de 2020 do Globo Rural, onde uma produtora de Góias criou uma 'barraca da honestidade' para melhorar as vendas de hortaliças e verduras (RURAL, 2020).

A versão disponibilizada ao fim do evento foi um aplicativo inicial que permitia aos agricultores divulgar suas lojas, barracas ou pontos de venda, juntamente com outras informações adicionais. Por sua vez, os co-agricultores poderiam visualizar as lojas mais próximas. O principal objetivo era confirmar a disponibilidade dos produtos e obter informações sobre pagamentos (CORREA; VELUDO, 2021).

Após o evento, os idealizadores uniram-se aos professores da faculdade e consultaram profissionais da área. Através de entrevistas, pesquisas e um profundo entendimento das regras de negócio envolvidas nas Comunidades que Sustentam a Agricultura (CSAs), um novo projeto foi desenvolvido, visando a inovação com uma plataforma web, um aplicativo exclusivo para co-agricultores e um design aprimorado. Dessa forma, o projeto tornou-se mais adequado às necessidades diárias e regras de uma CSA (CORREA; VELUDO, 2021).

Em seguida, o Projeto Agromart passou por outra iteração, com uma implantação individualizada e automatizada de um ambiente de CSA (FREITAS; CELLA, 2023).

Atualmente o projeto passa por outra iteração onde (BOTTINO; AUGUSTINI, 2023) se propõem a melhorar a segurança do Agromart, bem como a criar um módulo de integração de pagamento. O projeto é coordenado por professores da UnB-FGA, que continuam seu desenvolvimento em colaboração com estudantes.

1.2 Problema

O Agromart é um projeto que utiliza a arquitetura em nuvem para fornecer serviços relacionados ao agronegócio. Atualmente, a arquitetura do Agromart está super provisionada, resultando na alocação de uma grande quantidade de recursos em nuvem, especificamente utilizando a plataforma Heroku. No entanto, esses recursos não estão sendo plenamente utilizados, o que leva a altos gastos por parte dos envolvidos no projeto.

A alocação excessiva de recursos em nuvem no Agromart, por meio do Heroku, resulta em um desperdício financeiro significativo. A falta de utilização total dos recursos disponíveis implica em um custo desnecessário para os envolvidos com o projeto. Esses gastos excessivos podem comprometer a viabilidade financeira do Agromart, além de reduzir a eficiência do projeto como um todo.

Além disso, a atual arquitetura do Agromart conta com uma API dicionário responsável pelo redirecionamento das CSAs. Essa API, embora cumpra sua função, pode ser otimizada em termos de custo. Atualmente, é utilizado um *container* Node para a implementação dessa API, o que implica em uma alocação de uma máquina exclusiva para operação correta.

Uma solução viável para resolver o problema do super provisionamento de recursos e reduzir os gastos associados é a refatoração da arquitetura do Agromart. Propõe-se substituir a API dicionário, responsável pelos redirecionamentos das CSA's, por uma função lambda na plataforma AWS. Essa mudança permitirá reduzir significativamente o custo de manutenção do open source pela UNB para zero, uma vez que as funções lambda na AWS oferecem um modelo de cobrança baseado no consumo efetivo.

A implementação dessa solução trará benefícios tanto financeiros quanto operacionais para o Agromart. Ao utilizar funções lambda na AWS, o projeto poderá dimensionar dinamicamente seus recursos de acordo com a demanda, evitando assim o super provisionamento e reduzindo os gastos desnecessários.

1.3 Objetivos

1.3.1 Objetivo Geral

O objetivo principal deste trabalho é a resolução do super provisionamento de recursos em nuvem e os altos gastos associados ao projeto. A substituição da API dicionário por funções lambda na AWS emerge como uma solução visando a redução de custos e otimização da utilização dos recursos em nuvem. Essa mudança não apenas diminuirá os gastos desnecessários, mas também proporcionará maior flexibilidade, escalabilidade e

controle sobre a arquitetura do projeto.

1.3.2 Objetivos Específicos

- Criação de uma documentação referente a refatoração da arquitetura utilizando FaaS:
  - Diagrama de arquitetura
  - Diagrama de Sequência
- Alteração do *deploy* da api dicionário para função Lambda na AWS e migração da base de dados relacional para NoSql
  - Criação da conta na AWS
  - Criação do *budget* para controle de gastos e notificação via email
  - Configuração do docker utilizando lambda
  - migração do *nodemailer* para o Amazon SES
- Alteração do *Mobile* para chamar as funções Lambda via SDK
- Criação de *pipeline* da Lambda (CI/CD)
- Garantir funcionamento da aplicação
  - Criação dos testes unitários
  - Regressivo para eventuais correções

1.4 Planejamento

Visando a utilização do Scrum e uma melhor organização nas entregas, foi criado o seguinte *roadmap* para o TCC2, que poderá ter sofrido alterações de acordo com o decorrer do projeto:

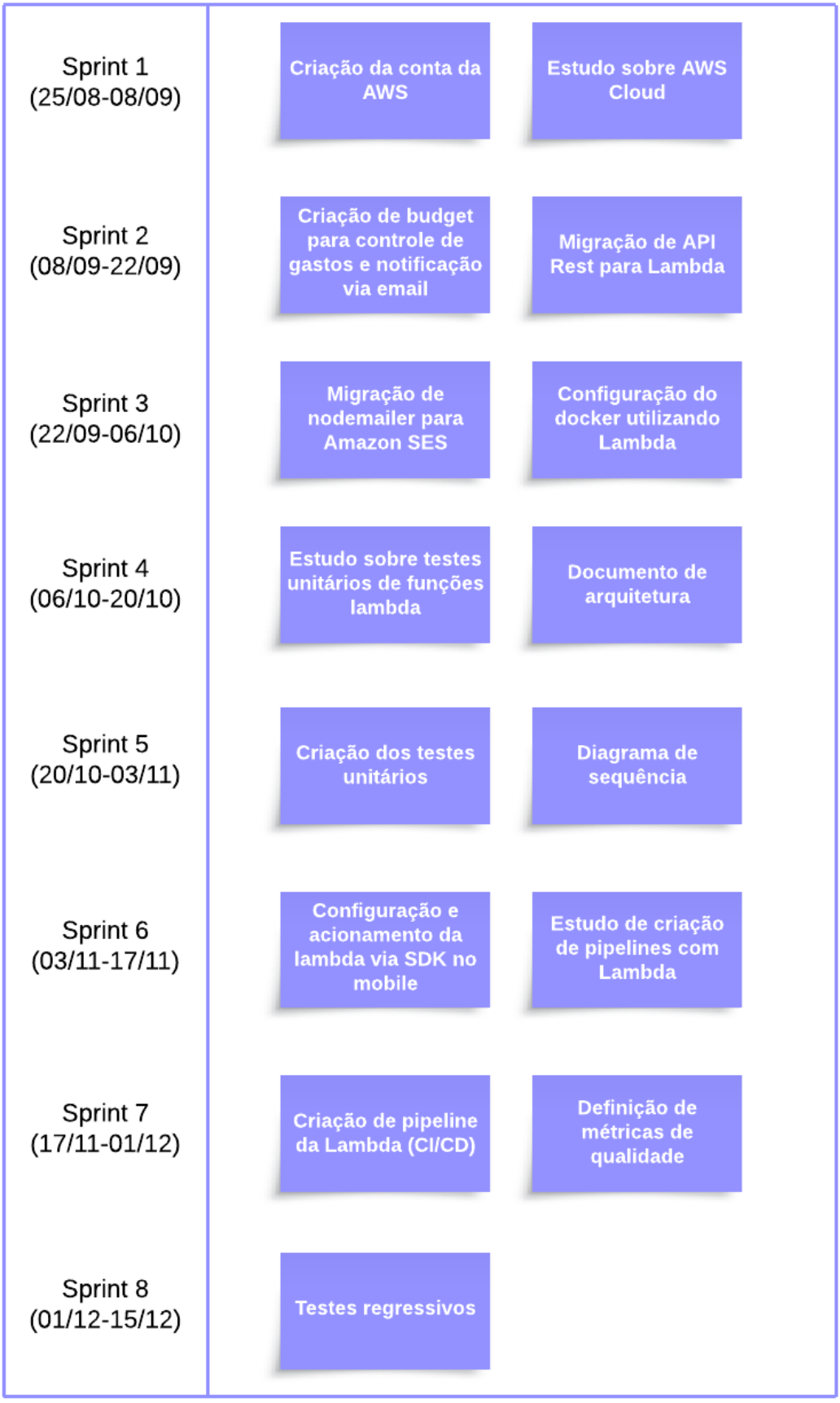

Figura 1 – Roadmap para o TCC2

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, Proof of Concept, Desenvolvimento Ágil de Software, Scrum, XP, Kanban, Cloud Computing, Serverless Computing, Deploy de Software, Ambientes de Software, Manutenção Evolutiva e Testes de Software.

2.1 Engenharia de Software

A Engenharia de Software é a aplicação de princípios, técnicas e métodos científicos para o desenvolvimento de *software* de qualidade, eficiente e confiável(IEEE Computer Society, 2006). É uma disciplina que envolve a aplicação de métodos e técnicas para desenvolver *software* de forma sistemática. Ela abrange todas as etapas do ciclo de vida do *software*, desde a análise de requisitos até a manutenção contínua do sistema. A Engenharia de Software utiliza abordagens e práticas para garantir a qualidade, eficiência e confiabilidade do *software*, bem como a satisfação do cliente.

Os profissionais de Engenharia de *Software* empregam uma variedade de ferramentas, linguagens de programação e metodologias para desenvolver e gerenciar projetos de *software*. Eles trabalham em equipe para analisar, projetar, implementar e testar sistemas de *software*, garantindo que eles atendam aos requisitos dos usuários. Além disso, a Engenharia de Software também lida com a gestão de projetos, incluindo o planejamento, alocação de recursos e controle de qualidade.

Ao adotar processos e práticas adequadas, a Engenharia de *Software* contribui para o sucesso dos projetos e a satisfação dos clientes, promovendo o avanço tecnológico e a inovação na área de software.

2.2 *Proof of Concept* - POC

A Prova de Conceito consiste em um modelo que busca atestar a veracidade e eficácia de algum conceito teórico. Na área de tecnologia a POC é bastante utilizada, especialmente por profissionais de Tecnologia da Informação (TI), para ter uma evidência documentada de que um software é bem-sucedido no que se propõe a fazer(TAYRANE, 2022).

2.3 Desenvolvimento Ágil de Software

O desenvolvimento ágil é um conjunto de metodologias de gerenciamento de projetos que se concentram na colaboração, na entrega rápida e na adaptação ao mudança. Os métodos ágeis são baseados na ideia de quebrar projetos grandes em pequenas tarefas gerenciáveis, que podem ser entregues em iterações curtas. Os métodos ágeis também enfatizam a comunicação e a colaboração entre os membros da equipe, e a capacidade de se adaptar às mudanças nas necessidades dos clientes(MARTIN, 2002).

O Desenvolvimento Ágil de Software se concentra na entrega contínua de valor aos clientes. Diferente das metodologias tradicionais, que são baseadas em planejamento rígido e em fases sequenciais, o desenvolvimento ágil valoriza a interação constante com os stakeholders e a capacidade de resposta a mudanças ao longo do processo.

Uma das principais características do desenvolvimento ágil é a adoção de iterações curtas e incrementais, conhecidas como sprints. Em vez de tentar definir todos os requisitos e detalhes do projeto no início, o desenvolvimento ágil permite que as equipes ajustem e priorizem os requisitos à medida que avançam, adaptando-se às necessidades em constante evolução dos clientes.

A colaboração e a comunicação efetiva são fundamentais no desenvolvimento ágil. As equipes multidisciplinares trabalham em conjunto, compartilhando conhecimentos, tomando decisões coletivas e se adaptando rapidamente às mudanças. Métricas de progresso e revisões regulares garantem transparência e feedback contínuo, permitindo que as equipes aprendam, melhorem e se ajustem ao longo do processo.

Em resumo, o desenvolvimento ágil de software promove a flexibilidade, a colaboração e a entrega de valor contínuo. Com sua abordagem iterativa e foco no cliente, ele se tornou uma alternativa popular às metodologias tradicionais, permitindo que as equipes desenvolvam software de maneira ágil, eficiente e adaptável às necessidades do mercado em constante mudança.

2.3.1 Scrum

O Scrum é uma metodologia ágil que se concentra na comunicação, colaboração e iteração. É uma abordagem eficaz para o desenvolvimento de *software*, pois permite que a equipe se adapte às mudanças nas necessidades do cliente e forneça um produto de alta qualidade em um curto espaço de tempo(SUTHERLAND; SCHWABER, 2014). Ele se baseia em ciclos de trabalho chamados de *sprints*, que têm duração fixa e geralmente duram de duas a quatro semanas. Durante cada *sprint*, uma série de atividades são realizadas para garantir a entrega contínua de valor ao cliente.

2.3.2 *Extreme Programming* - XP

O XP é um método de desenvolvimento de *software* iterativo e incremental que enfatiza a comunicação, simplicidade, feedback, coragem e respeito. XP é baseado na ideia de que o *software* é melhor desenvolvido em pequenas equipes que trabalham em conjunto e se comunicam constantemente. XP também enfatiza a importância de manter o *software* simples e fácil de entender, e de obter feedback dos usuários o mais cedo possível(BECK, 2000). Através de práticas como programação em pares, testes automatizados, integração contínua e padronização de código, o XP visa melhorar a qualidade do *software*, reduzir riscos e aumentar a satisfação do cliente. Sua abordagem flexível e adaptativa permite uma maior capacidade de resposta às mudanças e uma maior eficiência na entrega de valor aos usuários finais.

2.3.3 Kanban

Kanban é uma metodologia de gerenciamento de projetos que se concentra na visualização do fluxo de trabalho e na melhoria contínua. Ele é baseado na ideia de que o trabalho é melhor realizado em pequenos lotes e que o fluxo de trabalho deve ser constantemente ajustado para melhorar a eficiência e a eficácia(ANDERSON, 2012). Originário do sistema de produção da Toyota, o Kanban é baseado em princípios como visualização, limitação de trabalho em progresso e melhoria contínua. Ele utiliza um quadro Kanban, dividido em colunas que representam os estágios do processo, onde as tarefas são movidas de uma coluna para outra à medida que são concluídas.

Com a aplicação do Kanban, equipes podem aumentar a transparência, melhorar o gerenciamento do fluxo de trabalho e reduzir o tempo de entrega. A metodologia promove um ambiente de trabalho mais organizado, flexível e orientado por dados, permitindo que a equipe acompanhe o progresso, faça ajustes rápidos e promova a melhoria contínua do processo.

2.4 *Cloud Computing*

A computação em nuvem é a entrega de serviços computacionais—incluindo servidores, armazenamento, bancos de dados, rede, software, análise, inteligência artificial e serviços de aplicativos—sob demanda pela Internet com pagamento em uso(GARTNER, 2014). Também conhecida como cloud computing, é um paradigma que revolucionou a forma como recursos computacionais são entregues e utilizados. Em vez de depender de infraestrutura local, os serviços em nuvem oferecem acesso sob demanda a recursos compartilhados, como servidores, armazenamento, redes e aplicativos, por meio da internet.

Existem três principais modelos de serviço em nuvem: Infraestrutura como Serviço

(IaaS), Plataforma como Serviço (PaaS) e *Software* como Serviço (SaaS). No modelo IaaS, os usuários têm controle total sobre a infraestrutura, como servidores virtuais, armazenamento e redes. No modelo PaaS, os usuários podem desenvolver, implantar e gerenciar aplicativos sem se preocupar com a infraestrutura subjacente. Já no modelo SaaS, os usuários têm acesso a aplicativos prontos para uso, que são executados na infraestrutura do provedor de nuvem(CHAPPELL, 2013).

A computação em nuvem oferece uma série de benefícios, incluindo escalabilidade, elasticidade, disponibilidade e segurança. Os recursos em nuvem podem ser facilmente dimensionados de acordo com as necessidades do usuário, permitindo que eles se adaptem a picos de demanda ou reduzam recursos quando não forem mais necessários. Além disso, os provedores de nuvem garantem a disponibilidade contínua dos serviços, com acordos de nível de serviço que estabelecem tempos de atividade mínimos. A segurança também é uma prioridade na computação em nuvem, com os provedores implementando medidas de proteção avançadas para garantir a confidencialidade e integridade dos dados dos usuários.

Além dos três principais modelos de serviço em nuvem, também temos o FaaS(Função como serviço), que é o caso das funções lambda na plataforma AWS (Amazon Web Services) que serão utilizadas neste projeto(DigitalOcean, 2023).

2.4.1 *Serverless Computing*

A computação *serverless*, é um modelo de computação em nuvem em que os desenvolvedores podem executar código sem se preocupar com a infraestrutura subjacente. Nesse modelo, é possível criar e executar aplicações e serviços sem a necessidade de gerenciamento de infraestrutura. A aplicação continua sendo executada em servidores, mas totalmente gerenciada pelo provedor escolhido. Você não precisa mais provisionar, escalar e manter servidores para executar aplicações, bancos de dados e sistemas de armazenamento(KLEPPMANN, 2019).

A computação *serverless* oferece diversos benefícios, como escalabilidade automática, pagamento apenas pelo uso real e simplificação do processo de desenvolvimento. Os desenvolvedores podem se concentrar na lógica de negócios de suas aplicações, enquanto o provedor cuida do dimensionamento automático dos recursos necessários para executar as funções. Além disso, o modelo *serverless* permite uma arquitetura mais modular e desacoplada, facilitando a criação de sistemas flexíveis e altamente escaláveis. Com sua natureza *pay-as-you-go* e a redução de preocupações com infraestrutura, a computação *serverless* tem ganhado popularidade como uma abordagem eficiente e econômica para desenvolvimento e implantação de aplicações na nuvem.

2.5 *Deploy* de Software

O *deploy* de *software* é o processo de implantação e disponibilização de um aplicativo ou sistema em um ambiente operacional para uso real. Envolve a transferência dos arquivos e recursos necessários para o ambiente de produção, configurando-os de forma apropriada. Durante o *deploy*, são realizadas etapas como compilação, empacotamento, configuração de servidores, bancos de dados e outras dependências(Cloud Academy, 2023).

O *deploy* de *software* é um passo crítico, pois é quando o sistema se torna acessível aos usuários finais. É importante garantir que o processo seja executado de forma consistente, segura e eficiente. Estratégias como o uso de ambientes de homologação, testes automatizados e integração contínua são adotadas para minimizar riscos e assegurar que o *software* implantado esteja funcionando corretamente. Um *deploy* bem-sucedido é fundamental para a entrega de valor aos usuários e para a evolução contínua do *software*.

2.6 Ambientes de Software

Um ambiente de *software* é um conjunto de recursos que são usados para desenvolver, testar e implantar um sistema de *software*. Os recursos em um ambiente de *software* podem incluir hardware, *software*, dados e pessoas. O ambiente de *software* deve ser projetado para atender às necessidades específicas do projeto de software(PRESSMAN, 2013).

- O ambiente de desenvolvimento é onde os desenvolvedores criam, testam e depuram o *software*. Geralmente, inclui ferramentas de desenvolvimento, ambientes de teste e versões iniciais do sistema.

- O ambiente de homologação é uma etapa intermediária entre o ambiente de desenvolvimento e o ambiente de produção. É usado para validar o *software* antes de ser implantado no ambiente de produção, garantindo que esteja de acordo com os requisitos e padrões esperados. Nesse ambiente, são realizados testes mais abrangentes, incluindo testes de integração e aceitação por usuários.

- O ambiente de produção é onde o *software* é implantado para uso real pelos usuários finais. É a versão estável e confiável do sistema, configurada para lidar com o volume de dados e a carga de trabalho esperados. Nesse ambiente, são implementadas medidas de segurança, monitoramento e suporte contínuo para garantir o bom funcionamento do *software* em produção.

Ter ambientes separados é fundamental para garantir a qualidade do *software* e mitigar riscos. Cada ambiente desempenha um papel específico e oferece um ambiente

controlado para diferentes estágios do desenvolvimento e implantação do *software*, permitindo que os desenvolvedores testem, validem e entreguem sistemas confiáveis e eficientes aos usuários finais.

2.7 Manutenção Evolutiva

A manutenção evolutiva é uma das principais atividades envolvidas no ciclo de vida do desenvolvimento de *software*. Trata-se de um processo contínuo que visa aprimorar, adaptar e expandir um sistema já existente, buscando atender às necessidades em constante evolução dos usuários e do ambiente em que o *software* está inserido.

Essa forma de manutenção está relacionada à incorporação de novas funcionalidades, melhorias e atualizações no *software*, permitindo que ele acompanhe as demandas e expectativas dos usuários ao longo do tempo. A manutenção evolutiva é essencial para garantir que o *software* permaneça relevante e competitivo, além de aumentar sua vida útil e maximizar o retorno sobre o investimento realizado em seu desenvolvimento.(MAJDENBAUM, 2004)

Em suma, a manutenção evolutiva é uma atividade essencial para garantir a longevidade e a relevância de um *software* no mercado. Por meio dela, é possível agregar valor ao sistema, adaptando-o às mudanças no ambiente e às necessidades dos usuários, bem como melhorar sua eficiência, usabilidade e desempenho. Ao adotar uma abordagem estruturada e planejada de manutenção evolutiva, as organizações podem maximizar o retorno sobre o investimento no desenvolvimento de *software* e fornecer aos usuários uma experiência contínua e satisfatória.

2.8 Testes de Software

Os testes de software são uma atividade fundamental no processo de desenvolvimento, que consiste em avaliar e validar o comportamento de um software em relação aos requisitos definidos. Esses testes têm o objetivo de identificar possíveis falhas, defeitos ou inadequações no software, garantindo sua qualidade e confiabilidade(BLACK, 2011).

Existem diversos tipos de testes de software, como testes de unidade, testes de integração, testes de sistema e testes de aceitação. Cada tipo de teste possui características e objetivos específicos, abrangendo desde a verificação do funcionamento correto de componentes individuais até a avaliação do sistema como um todo.

Através da realização sistemática de testes, os desenvolvedores podem identificar e corrigir problemas, garantindo que o software atenda às expectativas dos usuários e funcione conforme o esperado. Os testes de software são essenciais para assegurar a qualidade,

a confiabilidade e o desempenho do software, contribuindo para a entrega de um produto final de excelência.

3 Metodologia

Neste capítulo, serão apresentados tópicos explicando como serão utilizados determinados conceitos dentro do projeto. Como por exemplo: metodologia de pesquisa, forma de comunicação da equipe, entre outras ferramentas e estratégias gerenciamento de projeto.

3.1 Metodologia de Pesquisa

O presente trabalho tem como objetivo realizar uma manutenção evolutiva no software Agromart, a fim de reduzir os custos do projeto. Para isso, será adotada uma metodologia de pesquisa aplicada. Ou seja, com ênfase prática na solução de problemas.

A pesquisa aplicada busca soluções para problemas práticos e tem como objetivo gerar conhecimento útil e aplicável ao contexto específico em que se insere. No caso deste trabalho, o problema a ser resolvido é o alto custo do projeto Agromart(CRESWELL, 2013).

Ao adotar uma metodologia de pesquisa aplicada, busca-se não apenas compreender teoricamente o problema, mas também implementar soluções práticas. Isso permite que o trabalho não se limite à teoria, mas ofereça contribuições reais para o campo da engenharia de software, fornecendo recomendações concretas para a melhoria do processo de desenvolvimento e manutenção de software.

3.2 *Proof of Concept* - POC

Neste trabalho, utilizamos a POC como forma de atestar que a migração da api dicionário para a função Lambda resultaria no mesmo resultado, porém reduzindo a zero o custo do projeto por parte da UNB.

3.3 Gerência do Projeto

A gestão de projetos é uma disciplina fundamental para o planejamento, organização, coordenação e controle de projetos. Neste trabalho, serão utilizados pontos positivos de metodologias ágeis, como Scrum, XP e Kanban, para otimizar a gestão do projeto e promover uma abordagem flexível e adaptativa(Project Management Institute (PMI), 2017).

3.3.1 Scrum

Neste trabalho, serão adotados alguns pontos específicos do Scrum. As *sprints* terão uma duração de duas semanas, permitindo um ritmo de trabalho mais ágil. O *Sprint Planning* ocorrerá no início de cada *sprint* para definir as metas e os itens que serão desenvolvidos, gerando o *Sprint Backlog* que será utilizado para acompanhar o progresso e as tarefas ao longo do sprint.

Além disso, será realizado o *Sprint Review* ao final de cada *sprint*, no qual o trabalho realizado será revisado com os *stakeholders* para obter feedback e fazer ajustes necessários. Por fim, o *Product Backlog* será utilizado para gerenciar as demandas e prioridades do projeto.

Esses elementos do Scrum serão aplicados para promover uma abordagem ágil e colaborativa no gerenciamento do projeto, permitindo uma maior eficiência, transparência e entrega de valor contínua ao longo do trabalho.

3.3.2 *Extreme Programming* - XP

O *Extreme Programming* é um processo ágil de desenvolvimento de software que se concentra em velocidade, qualidade e simplicidade(BECK, 2000).

Neste trabalho, serão adotados com mais enfoque três dos 12 pontos específicos do Extreme Programming: programação em pares, colaboração e respeito.

3.3.2.1 Programação em pares

A programação em pares é uma prática em que dois desenvolvedores trabalham juntos em um único código. Eles alternam entre os papéis de piloto (aquele que escreve o código) e navegador (aquele que revisa o código e fornece feedback).

A programação em pares tem vários benefícios, incluindo:

Melhor qualidade do código Maior produtividade Melhor compreensão do código Melhor comunicação entre os desenvolvedores Melhor aprendizado

3.3.2.2 Colaboração

A colaboração é essencial para o sucesso de qualquer projeto de desenvolvimento de *software*. No XP, a colaboração é incentivada entre todos os membros da equipe, incluindo desenvolvedores, gerentes, usuários e clientes.

A colaboração permite que a equipe compartilhe ideias, resolva problemas e aprenda uns com os outros. Também ajuda a garantir que o projeto atenda às necessidades de todos os envolvidos.

3.3.2.3 Respeito

O respeito é outro valor fundamental do XP. Todos os membros da equipe devem ser tratados com respeito, independentemente de seu nível de experiência, função ou título.

O respeito ajuda a criar um ambiente de trabalho positivo e produtivo. Também ajuda a garantir que todos os membros da equipe se sintam confortáveis em compartilhar suas ideias e trabalhar juntos.

3.3.3 Kanban

Neste trabalho, será utilizada a metodologia de gestão visual Kanabn que tem como maior ponto positivo sua capacidade de proporcionar uma visão clara e instantânea do trabalho em andamento, identificar gargalos e problemas de fluxo, e promover um fluxo contínuo de trabalho. Ao mostrar visualmente o trabalho em progresso o kanban limita o trabalho em progresso, evitando sobrecarga e maximizando a eficiência. Além disso, o sistema de *feedback* visual do Kanban facilita a colaboração, a identificação de gargalos e a tomada de decisões baseadas em dados.

3.4 Gerência de Configuração de Software

3.4.1 Repositório

Por ser um projeto *open source* que já vem sendo desenvolvido em outras iterações por diferentes alunos desde a sua criação, será mantida a utilização do repositório no Github. Essa decisão além de facilitar contribuições ao projeto já que é um projeto *open source*, facilita o uso de algumas ferramentas como o Zenhub que tem integração com o Github.

3.4.2 Política de branches

Será mantido o uso do *Gitflow*, que é uma política de *branching* muito utilizada em projetos de software por possuir um modelo que estipula bem claramente a função de cada tipo de *branch* facilitando o desenvolvimento do produto.

- A *branch* master é onde temos as versões estáveis do projeto, o que quer dizer que o código apenas vai para a *branch* master após ter sido testado.
- A *branch* develop é para onde vão, inicialmente, as novas funcionalidades. Após um certo acúmulo dessas funcionalidades é que será gerada uma nova versão estável e isso irá para a *branch* master.

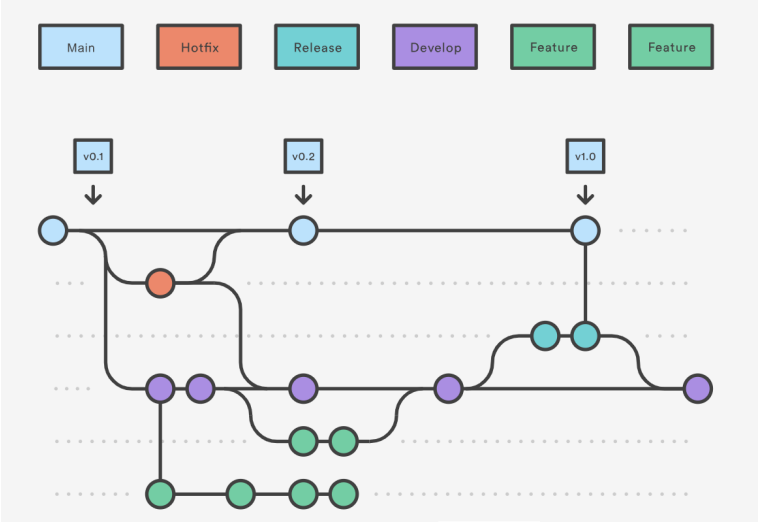

Figura 2 – Gitflow

Fonte: (ATLASSIAN, 2023)

- As branches de features são criadas sempre a partir da develop, e são nelas onde serão desenvolvidas cada funcionalidade separadamente.

- Por fim, temos as branches de Hotfix que são criadas a partir da *branch* master. Esse tipo de *branch* só é criado quando há algum problema em produção e após o problema ser corrigido na *branch* de hotfix, essa *branch* será mesclada diretamente na master e na develop simultaneamente.

3.4.3 Política de Issues

No contexto de desenvolvimento de software, issues são tarefas, problemas ou melhorias que são registradas e rastreadas para acompanhamento e resolução. Elas podem abranger desde a implementação de novos recursos (*FEATURE*), a refatoração de código existente (*REFACTOR*), correção de erros (*BUG*) e criação de documentação (*DOC*). As issues são essenciais para o gerenciamento do projeto, permitindo a priorização e atribuição de trabalho, além de fornecer um registro claro das demandas e atividades realizadas.

No âmbito deste trabalho, será adotada uma política de issues em que todas as *User Stories* (US) que envolvem tarefas de *FEATURE*, *REFACTOR*, *BUG* ou *DOC* devem estar vinculadas a um *Epic*, representando uma unidade maior de trabalho. Essa abordagem ajuda a organizar e agrupar as tarefas relacionadas, permitindo uma visão mais abrangente e facilitando o acompanhamento do progresso. No entanto, nem todas as tarefas estarão vinculadas a uma US, permitindo a inclusão de atividades menores que podem não se enquadrar em um *Epic* específico. Essa política de issues contribui para uma gestão mais estruturada e eficiente do projeto, garantindo a clareza e rastreabilidade das tarefas realizadas.

3.4.4 Política de *commits*

No contexto de controle de versão de *software*, um commit é uma ação de salvar as alterações realizadas em um repositório de código. Ele representa uma unidade de alteração ou conjunto de alterações que foram aplicadas a arquivos específicos. Cada commit geralmente possui uma mensagem descritiva que fornece informações sobre as alterações realizadas.

Neste trabalho, será adotada uma política de *commits* semânticos, que segue uma convenção padronizada para as mensagens dos *commits*. Essa política visa fornecer uma descrição clara e significativa das alterações realizadas em cada commit. Por meio dessa abordagem, as mensagens de commit são estruturadas de forma a indicar o tipo de alteração realizada (como correção de bug, implementação de recurso, refatoração, etc.).

Então um commit que adicione uma funcionalidade nova no código ficaria da seguinte maneira:

"feat: adiciona nova funcionalidade"

3.5 Ferramentas de comunicação

3.5.1 Discord

O Discord é uma plataforma de comunicação em tempo real que permite aos usuários se conectar e interagir por meio de bate-papo por texto, voz e vídeo. Foi lançado em 2015 e teve como objetivo inicial atender às necessidades da comunidade de jogos online, oferecendo uma maneira fácil e eficiente de se comunicar durante as partidas(DISCORD, 2023).

Com o tempo, o Discord expandiu suas funcionalidades e conquistou usuários de diversas áreas, tornando-se uma plataforma popular para grupos de amigos, comunidades e até mesmo ambientes profissionais. Sua interface intuitiva, recursos avançados de voz e a possibilidade de criar servidores personalizados contribuíram para o sucesso do Discord como uma plataforma de comunicação moderna e versátil. Ele será utilizado especialmente como ferramenta para a condução da prática de *pair programming* por possibilitar o compartilhamento de telas.

3.5.2 Whatsapp

O WhatsApp foi fundado por Jan Koum e Brian Acton, que já haviam trabalhado juntos no Yahoo por 20 anos. O WhatsApp se juntou ao Facebook em 2014, mas continua a operar como um app separado com foco no serviço de mensagens rápido e confiável em qualquer lugar do mundo (WHATSAPP, 2023).

Nesse sentido, o app será utilizado para trocas rápidas de informações ou para marcações de reuniões entre a equipe.

3.5.3 Zenhub

O Zenhub é uma ferramenta de gerenciamento de projetos que ajuda as equipes ágeis a visualizar, planejar e executar suas atividades de forma eficiente. Ele integra-se diretamente ao GitHub, permitindo que os times visualizem e gerenciem suas tarefas diretamente no ambiente em que o desenvolvimento de software ocorre. Além de fornecer um quadro Kanban intuitivo para o acompanhamento das tarefas, o Zenhub oferece recursos como estimativas de tempo, rastreamento de problemas, gráficos de velocidade e muito mais. Com essa ferramenta, os times podem ter uma visão clara do andamento do projeto, facilitando a colaboração, o acompanhamento do progresso e a identificação de gargalos, promovendo uma gestão mais eficiente do fluxo de trabalho(ZENHUB, 2023).

3.6 Backlog

O *Backlog* deste projeto utilizará conceitos que são utilizados nas metodologias ágeis. Como por exemplo o fato de as histórias de usuário serem curtas, simples e de fácil entendimento.

- Criação da conta AWS
- Criação de budget para controle de gastos e notificação via email
- Migração de API Rest para Lambda
- Migração de nodemailer para Amazon SES
- Configuração do docker utilizando lambda
- Criação dos testes unitários
- Criação de pipeline da Lambda (CI/CD)
- Configuração e acionamento da Lambda via SDK no mobile
- Diagrama de arquitetura
- Diagrama de sequência
- Regressivo para eventuais correções e assegurar correto funcionamento da aplicação

4 Suporte Tecnológico

Este capítulo visa descrever aspectos técnicos referentes a este trabalho de conclusão de curso, como a linguagem de programação, arquitetura, entre outros.

4.1 Linguagem de programação

JavaScript é uma linguagem de programação interpretada que pode ser usada para adicionar interatividade e funcionalidades a páginas da *web*. Ela é uma das três principais linguagens da web, juntamente com HTML e CSS. JavaScript é usada para criar uma ampla gama de efeitos e funcionalidades em páginas da *web*, incluindo animações, jogos, formulários interativos, gráficos e notificações(MDN, 2023).

O AgroMart usa o JavaScript como base em todos os seus serviços. Isso significa que todas as páginas da *web*, formulários e outros recursos do AgroMart são escritos em JavaScript.

A refatoração do serviço de URL para funções lambda também será escrita em JavaScript. Isso permitirá que o AgroMart crie um serviço de URL mais eficiente e escalável.

4.2 AWS Lambda

AWS Lambda é um serviço de computação sem servidor que permite que você execute código em resposta a eventos. Ele é usado para criar aplicativos escaláveis e de baixa manutenção, sem se preocupar com a infraestrutura subjacente(AWS, 2023c).

Quando um evento ocorre, Lambda inicia um *container* para executar seu código. O *container* é encerrado quando o código é concluído, economizando recursos. Isso permite que Lambda seja escalado automaticamente conforme a demanda, sem a necessidade de gerenciar servidores.

Lambda também é integrado com outros serviços da AWS, permitindo que você crie aplicativos complexos e escaláveis. Você pode usar Lambda para criar APIs, processar dados e executar tarefas em segundo plano.

Aqui estão algumas das vantagens de usar Lambda:

- Escalabilidade automática: Lambda escala automaticamente conforme a demanda, economizando recursos.

- Integração com outros serviços da AWS: Lambda é integrado com outros serviços da AWS, permitindo que você crie aplicativos complexos e escaláveis.
- Fácil de usar: Lambda é fácil de usar e não requer nenhuma experiência com gerenciamento de servidores.
- Pagamento baseado no tempo de execução: Você paga apenas pelo tempo em que seu código está em execução, economizando dinheiro.

4.3 Docker

O Docker é uma plataforma de código aberto que permite criar, gerenciar e executar aplicativos dentro de contêineres. Os contêineres são ambientes isolados que contêm tudo o que um aplicativo precisa para ser executado, incluindo o código, as bibliotecas e as dependências necessáriasz(DOCKER, 2023).

Com o Docker, é possível empacotar um aplicativo junto com suas dependências em um contêiner único, garantindo que ele funcione de maneira consistente em qualquer ambiente, independentemente das diferenças de configuração. Essa abordagem oferece portabilidade, escalabilidade e eficiência, facilitando o desenvolvimento, a implantação e a distribuição de aplicativos em diferentes plataformas e infraestruturas.

O Docker permitirá que qualquer um que clone o repositório consiga testar localmente a aplicação, especialmente as funções AWS Lambda.

4.4 AWS SAM(Serverless Application Model)

O AWS Serverless Application Model (AWS SAM) é um kit de ferramentas que melhora a experiência do desenvolvedor na criação e execução de aplicativos sem servidor no AWS(AWS, 2023b).

O AWS SAM permitirá executar e testar localmente a estrutura serverless da AWS Lambda.

4.5 DynamoDB

O Amazon DynamoDB é um serviço de banco de dados NoSQL totalmente gerenciado que oferece alta disponibilidade, escalabilidade e desempenho. Ele é projetado para aplicativos que precisam acessar grandes volumes de dados de forma rápida e consistente. O DynamoDB é um banco de dados de chave-valor, o que significa que cada item de dados é identificado por uma chave. O DynamoDB também oferece suporte para armazenamento de documentos e tabelas(AWS, 2023a).

O DynamoDB é um banco de dados totalmente gerenciado, o que significa que a Amazon cuida de toda a infraestrutura. Isso inclui gerenciamento de servidores, armazenamento, rede e segurança. Os desenvolvedores podem se concentrar no desenvolvimento de aplicativos sem se preocupar com a administração do banco de dados.

O DynamoDB é um banco de dados altamente escalonável. Ele pode ser facilmente escalado para atender às demandas de tráfego. O DynamoDB também é um banco de dados altamente disponível. Ele é projetado para permanecer disponível mesmo em caso de falhas de hardware ou software.

O serviço de dicionário de URL que passará a ser funções Lambda não usará mais o banco de dados relacional PostgreSQL e utilizará o DynamoDB espcialmente por ter integração com o AWS Lambda, já que ambos são serviços da AWS com um programa de custos que para as necessidades do AgroMart não chegará a ser cobrado nada.

4.6 Arquitetura

A arquitetura atual do projeto, utiliza uma api dicionário hospedada no Heroku, utilizando o modelo cliente servidor permitindo a aplicação mobile a requisição da url de uma CSA para o redirecionamento adequado ao seu CMS. Entretanto, visando a redução de custos por parte da UNB, foi realizada a migração dessa api para utilização de função Lambda, migração da base de dados relacionais para DynamoDb.

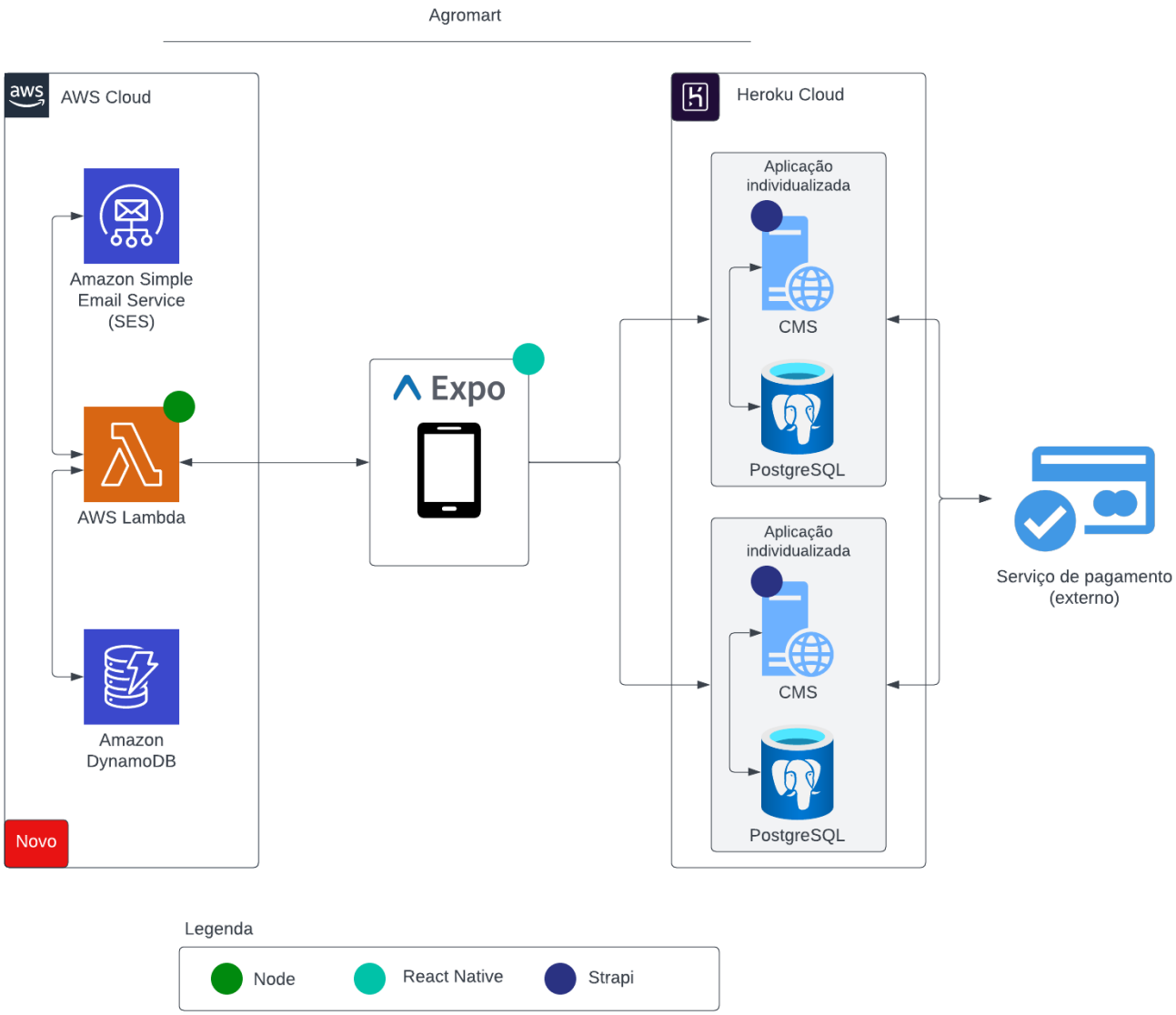

Figura 3 – Nova arquitetura proposta

5 Resultados

Este capítulo tem por objetivo expor os resultados obtidos no decorrer desde trabalho de conclusão de curso

5.1 Pré-desenvolvimento

No ínicio do desenvolvimento deste tcc, alguns problemas inesperados foram enfrentados em etapas anteriores ao desenvolvimento do projeto em si. Alguns destes problemas foram: Repositórios com versões de dependências que já foram descontinuadas e trechos de código que impediam o software de funcionar corretamente.

5.1.1 Manutenção API

Na API foram encontrados problemas referentes à modelagem de entidades, bem como problemas de codificação que impediam alguns pontos do serviço de funcionarem corretamente. Devido à falta de familiaridade com o código deste serviço, um custo de tempo foi gerado para a equipe, onde foi gasto um tempo razoável para conseguir identificar pontos que deixassem o serviço pronto para ser executado corretamente.

5.1.2 Manutenção Mobile

No serviço mobile, que é um cliente feito em *React Native*, problemas mais críticos que o serviço citado anteriormente foram encontrados.

A primeira barreira encontrada foi o fato de que inicialmente o serviço *mobile* não estava funcionando, depois de uma investigação mais a fundo foi identificado que algumas versões de dependências do serviço estavam descontinuadas, dentre elas o próprio *expo* que é uma dependência utilizada no *React Native* para desenvolver, construir, implantar e iterar rapidamente em aplicativos iOS, Android e web a partir da mesma base de código.

Além das versões de dependências atualizadas, foi constatado que a forma como o app gerenciava o tipo de ambiente (desenvolvimento ou produção) não estava corretamente programado para que fosse possível testar a aplicação localmente. Por isso a equipe precisou atualizar a forma como o app controlava se o ambiente era de desenvolvimento ou de produção.

Dependência	Versão anterior	Versão atualizada
expo	44.0.0	49.0.0
@expo/vector-icons	12.0.0	13.0.0
@react-native-async-storage/async-storage	1.15.0	1.18.2
@react-native-picker/picker	2.2.1	2.4.10
expo-device	4.1.1	5.4.0
expo-font	10.0.4	11.4.0
expo-linear-gradient	11.0.3	12.3.0
expo-notifications	0.14.1	0.20.1
expo-permissions	13.1.1	14.2.1
expo-splash-screen	0.14.1	0.20.5
expo-status-bar	1.2.0	1.6.0
expo-updates	0.11.6	0.18.17
lottie-react-native	5.0.1	5.1.6
react	17.0.1	18.2.0
react-dom	17.0.1	18.2.0
react-native	0.64.3	0.72.5
react-native-gesture-handler	2.1.0	2.12.0
react-native-reanimated	2.3.1	3.3.0
react-native-safe-area-context	3.3.2	4.6.3
react-native-screens	3.10.1	3.22.0
react-native-web	0.17.1	0.19.6

Figura 4 – Quadro de dependências atualizadas

Dependência de Desenvolvimento	Versão anterior	Versão atualizada
@babel/core	7.12.9	7.20.0
@types/react	17.0.21	18.2.14
@types/react-dom	17.0.9	18.0.10
typescript	4.3.5	5.1.3

Figura 5 – Quadro de dependências de desenvolvimento atualizadas

5.2 Testes

Durante a atividade de estudo sobre testes unitários de funções lambda previsto no planejamento do TCC1 a equipe constatou que a realização de tais testes não garantiria o correto funcionamento do sistema. Já que os componentes tem muita interação com ambientes de *cloud*, a realização de testes unitários iria isolar ou simular essas interações. Fazendo com que a ideia de migrar para testes de integração se mostrasse muito mais efetiva para assegurar o correto funcionamento do sistema.

5.3 Automatização

Para o cumprimento da ideia principal por trás deste Trabalho de Conclusão de Curso, foi necessária uma alteração no *script* automatizado que antes fazia o deploy de uma CSA utilizando o serviço de api-dicionário instanciado e mantido pela UnB na plataforma Heroku. Essa alteração faz com que o serviço de api-dicionário da forma como existia não tenha mais utilidade, e esta automatização agora aciona um evento de função Lambda da AWS que faz a função antes executada pelo serviço de api-dicionário porém de forma que não haja custos para a UnB.

Uma das conquistas a se citar é o fato de que apesar da mudança no script de *deploy* automático de uma CSA, os pré-requisitos para se utilizar deste não mudaram. O que nos mostra que foi uma mudança que além de ser positiva, não gerou impactos negativos no sentido de dificultar a experiência do usuário.

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

5.4 Amazon SES

Durante a atividade planejada de migração do *nodemailer* para o Amazon SES a equipe constatou que o serviço de *e-mails* da Amazon exige um endereço de *e-mail* verificado pela AWS, bem como uma maneira de o usuário se descadastrar do recebimento dessas mensagens e o Agromart atualmente não possui estrutura para isso.

Devido a esses pontos a equipe julgou que a tentativa de migração do serviço de *e-mails* não valia o esforço e por isso ficou decidido que seria mantido o *nodemailer* que já era previamente utilizado.

5.5 Migração API Dicionário -> AWS Lambda

Tendo em vista a motivação de redução de custos deste trabalho, para remover a necessidade de manter um ambiente que gera custos para apenas 4 operações. Foram criados 4 funções Lambda:

- Listar todas as CSAs cadastradas
- Cadastrar uma CSA

- Resgatar informações de uma CSA pelo id
- Deletar uma CSA pelo id

A api dicionário era um serviço REST criado utilizando Node.js junto com o framework Express. Era uma Api simples que se comunicava com um banco de dados PostgresSql via a ORM squelize para salvar dados básicos sobre cada CSA vinculada ao projeto Agromart.

Com a migração cada uma das rotas do antigo serviço REST virou um evento que pode ser acionado por meio de uma requisição HTTP. Esse evento se comunica com um banco de dados DynamoDB para fazer as operações básicas relacionadas a criação, consulta e deleção de CSAs.

Abaixo encontra-se o diagrama de sequência mostrando a interação do script de deploy automático com a AWS Lambda e o DynamoDB.

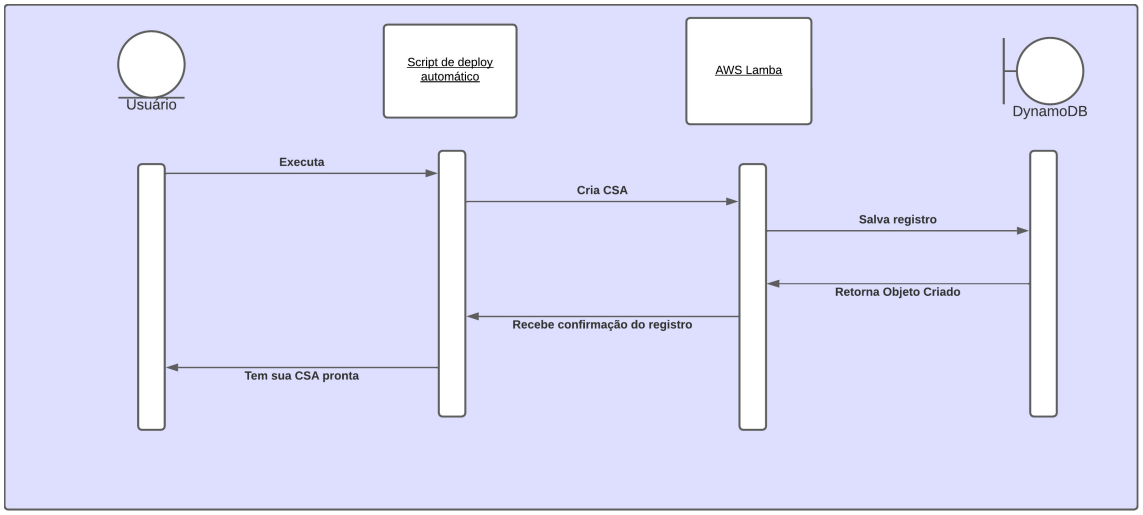

Figura 6 – Diagrama de sequência

5.6 BPMN

O BPMN, ou Business Process Model and Notation, é uma linguagem padronizada para modelagem de processos de negócios. Essa notação gráfica fornece uma maneira compreensível e uniforme de representar os processos operacionais de uma organização, independentemente de sua complexidade. O BPMN utiliza símbolos e elementos visuais para descrever as etapas, decisões, atividades e fluxos de informação que compõem um processo de negócio. Essa abordagem visual facilita a comunicação entre diferentes partes interessadas, desde analistas de negócios até desenvolvedores de software, proporcionando uma compreensão clara e unificada das operações da empresa. Além de servir como uma ferramenta de documentação, o BPMN também desempenha um papel crucial na melhoria

contínua dos processos, permitindo a análise, otimização e automação eficientes para aumentar a eficácia e eficiência operacional das organizações.

Abaixo se encontra o diagrama BPMN do processo de criação de uma CSA como um todo:

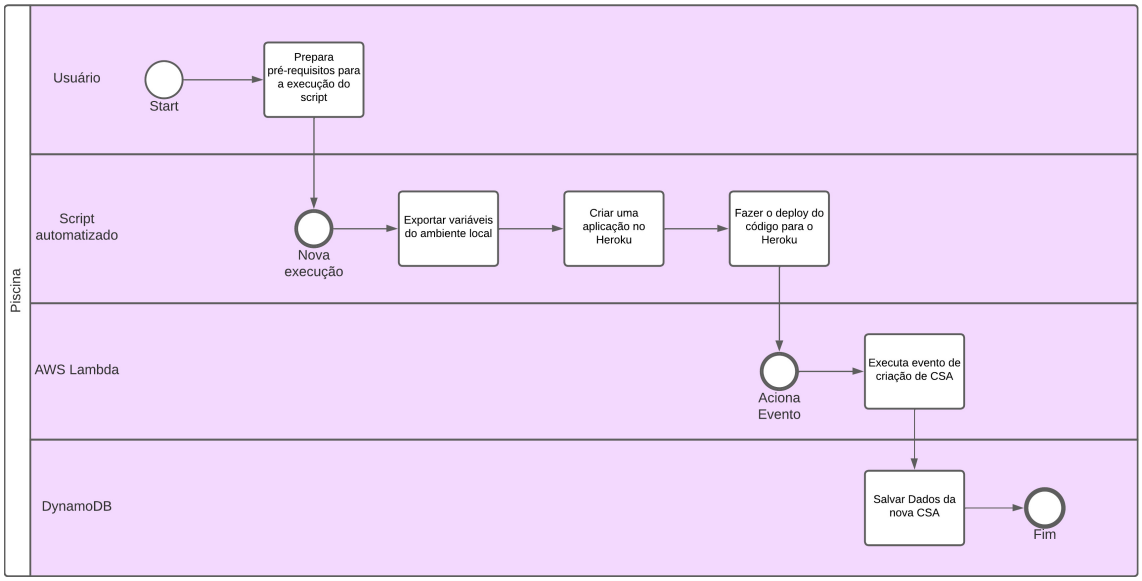

Figura 7 – BPMN

5.7 planejado x executado

De acordo com a figura 1 presente no tópico 1.4 deste trabalho. A equipe elaborou um *roadmap* para o TCC2. Conforme previsto, houveram mudanças na organização das sprints.

Durante a atividade de estudo de testes unitários para funções lambda, foi constatado pela equipe que os testes unitários não fariam tanto sentido para o projeto e ficou definido que fariamos testes de integração. Com isso uma nova atividade surgiu no planejamento enquanto a atividade de implementar os testes unitários deixou de existir.

Da mesma maneira, ao estudar sobre o Amazon SES a equipe julgou melhor por manter o nodemailer. O que fez com que a tarefa de migrar do nodemailer para o Amazon SES fosse substituida apenas pelo estudo sobre a possibilidade de utilizar o Amazon SES.

Além disso foi incluído no planejamento a elaboração de um BPMN para uma melhor visualização do processo de criação de uma CSA que agora passa pelas funções Lamdba.

Abaixo se encontra o *roadmap* atualizado com a real execução no decorrer deste TCC2.

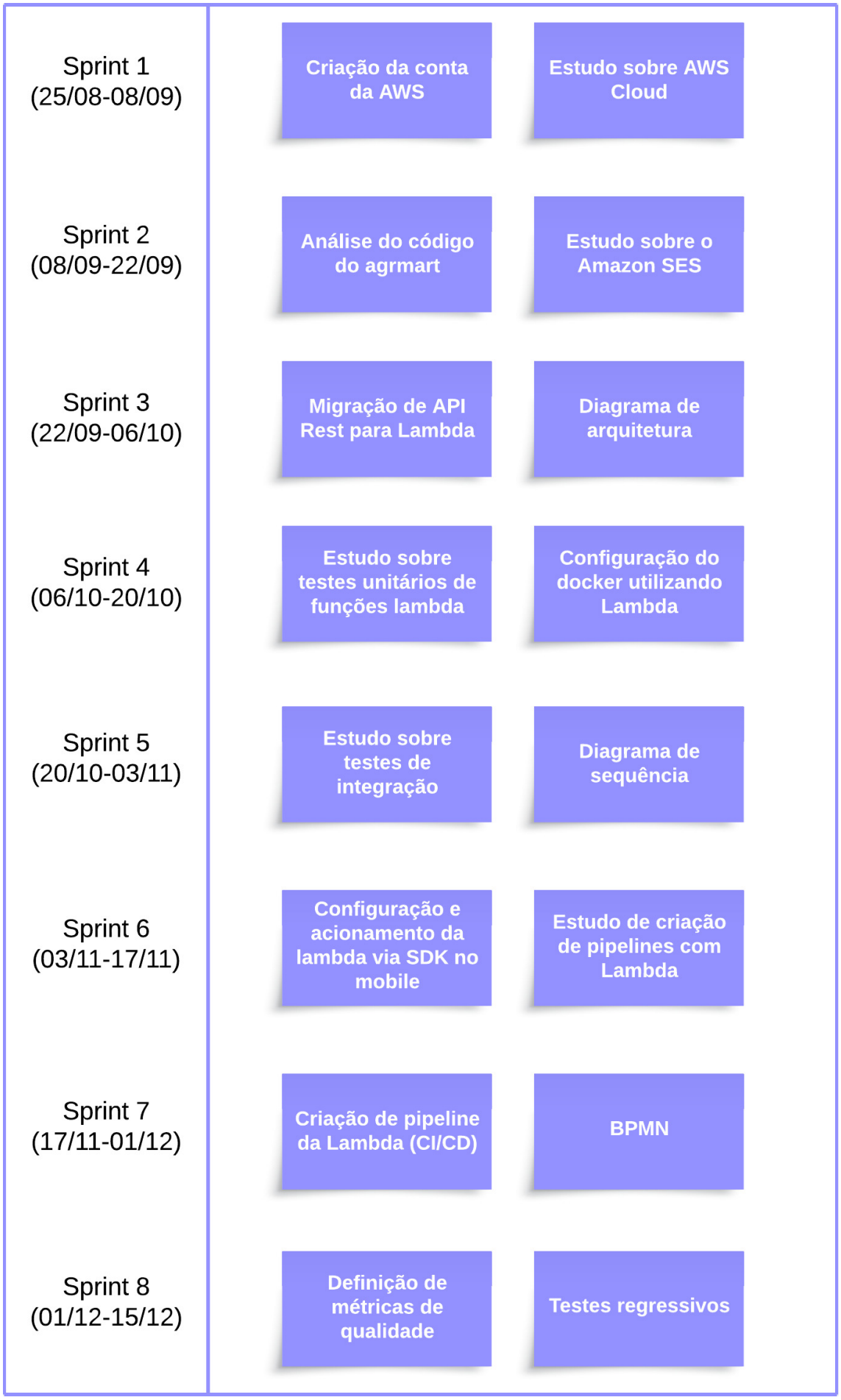

Figura 8 – Quadro com o roadmap efetivamente realizado

6 Considerações Finais

Referências

ANDERSON, D. *Kanban: A Gentle Introduction to Agile Software Development*. [S.l.]: Blue Hole Press, 2012. Citado na página 25.

ATLASSIAN. *Gitflow Workflow*. 2023. Disponível em: <https://www.atlassian.com/br/git/tutorials/comparing-workflows/gitflow-workflow>. Citado na página 34.

AWS. *Amazon DynamoDB*. 2023. Disponível em: <https://aws.amazon.com/pt/dynamodb/>. Citado na página 38.

AWS. *O que é AWS Serverless Application Model (AWS SAM)?* 2023. Disponível em: <https://docs.aws.amazon.com/pt_br/serverless-application-model/latest/developerguide/what-is-sam.html>. Citado na página 38.

AWS. *O que é o AWS Lambda*. 2023. Disponível em: <https://docs.aws.amazon.com/pt_br/lambda/latest/dg/welcome.html>. Citado na página 37.

BECK, K. *Extreme Programming Explained: Embrace Change*. [S.l.]: Addison-Wesley Professional, 2000. Citado 2 vezes nas páginas 25 e 32.

BLACK, R. *Teste de software: uma abordagem prática*. [S.l.]: Bookman, 2011. Citado na página 28.

BOTTINO, B. B.; AUGUSTINI, F. B. S. *Uma evolução do projeto agromart: implantação individualizada e automatizada de um ambiente de csa*. 2023. Citado na página 18.

CHAPPELL, D. *Cloud Computing: Uma abordagem prática*. [S.l.]: Bookman, 2013. Citado na página 26.

Cloud Academy. O que é deploy e como funciona? 2023. Disponível em: <https://cloudacademy.com.br/blog/o-que-e-deploy-e-como-funciona/>. Citado na página 27.

CORREA, B. K. B.; VELUDO, I. G. *Proposta para gestão de co-agricultores e meios de pagamentos na agricultura familiar: uma evolução do agromart*. 2021. Citado na página 18.

CRESWELL, J. W. *Research design: Qualitative, quantitative, and mixed methods approaches*. 4th. ed. Thousand Oaks, CA: Sage Publications, 2013. Citado na página 31.

DigitalOcean. Function as a service (faas): O que é e como funciona? 2023. Disponível em: <https://www.digitalocean.com/community/tutorials/function-as-a-service-faas-o-que-e-e-como-funciona>. Citado na página 26.

DISCORD. *Discord: Sobre nós*. 2023. Disponível em: <https://discord.com/company>. Citado na página 35.

DOCKER. *Docker overview*. 2023. Disponível em: <https://docs.docker.com/get-started/overview/>. 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 na página 18.

GARTNER. *Definição de computação em nuvem*. 2014. Disponível em: <https://www.gartner.com/it-glossary/cloud-computing/>. Citado na página 25.

GUILHOTO, J. J. M.; OUTROS. Os impactos econômicos do agronegócio familiar na economia do brasil. *Revista de Economia e Sociologia Rural*, v. 55, n. 4, p. 635–656, 2017. Citado na página 17.

IEEE Computer Society. Definição de engenharia de software. *IEEE Software*, v. 23, n. 6, p. 10–12, 2006. Citado na página 23.

KLEPPMANN, M. *Serverless Computing: A Practitioner's Guide*. [S.l.]: O'Reilly Media, 2019. Citado na página 26.

MAJDENBAUM, J. L. N. A. R. S. d. E. *Uma análise crítica dos desafios para engenharia de requisitos em manutenção de software*. 2004. <http://www.metodoiron.com.br/iron/wp-content/uploads/2014/03/Artigo_EngReq_ManutencaoSoftware.pdf>. Citado na página 28.

MARTIN, R. C. *Agile Software Development, Principles, Patterns, and Practices*. 3rd. ed. [S.l.]: Prentice Hall, 2002. Citado na página 24.

MATRES. *CSA – Comunidade que Sustenta a Agricultura*. 2023. Disponível em: <https://matres.com.br/pt/areas-de-atuacao/agroecologia/>. Citado na página 17.

MDN. *What is JavaScript?* 2023. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Learn/JavaScript/First_steps/What_is_JavaScript>. Citado na página 37.

MEIRELES, D. Comunidades que sustentam a agricultura (csa): Uma abordagem participativa para o consumo consciente. In: *Anais do 16º Encontro Nacional de Estudos do Consumo*. [S.l.: s.n.], 2018. Citado na página 17.

PRESSMAN, R. S. *Software Engineering: A Practitioner's Approach*. 9th. ed. [S.l.]: McGraw-Hill Education, 2013. Citado na página 27.

Project Management Institute (PMI). *Gerenciamento de Projetos: Guia do Conhecimento do PMI*. [S.l.]: Project Management Institute, 2017. Citado na página 31.

RURAL, G. *Produtora de Goiás cria 'barraca da honestidade' para melhorar as vendas de hortaliças e verduras*. 2020. Disponível em: <https://g1.globo.com/economia/agronegocios/globo-rural/noticia/2020/05/03/produtora-de-goias-cria-barraca-da-honestidade-para-melhorar-as-vendas-de-hortalicas-e-verduras.ghtml>. Citado na página 18.

SUTHERLAND, J.; SCHWABER, K. *Scrum: A Discipline for Agile Software Development*. [S.l.]: Addison-Wesley Professional, 2014. Citado na página 24.

TAYRANE. *POC: o que é a Prova de Conceito e qual a sua importância?* 2022. Disponível em: <https://blog.ploomes.com/poc/>. Citado na página 23.

WHATSAPP. *Sobre o WhatsApp*. 2023. Disponível em: <https://www.whatsapp.com/about>. Citado na página 35.

ZENHUB. *Zenhub - The project management tool teams love*. 2023. Disponível em: <https://www.zenhub.io/>. Citado na página 36.